Deep Generative Models for Phylogenetic Inference with Complex Evolutionary Processes

Ethan Baron[*]
New York University
eb4727@nyu.edu

Alan N. Amin[*]
New York University
alan.nawzad.a@gmail.com

Andrew Gordon Wilson
New York University, Perplexity AI
andrewgw@cims.nyu.edu

Abstract

Phylogenetic inference plays a central role in understanding evolutionary relationships, with applications ranging from tracking pathogen spread to reconstructing the history of life. Conventionally, practitioners obtain posteriors over the phylogenetic tree of species using MCMC methods. However, such likelihood-based methods are only tractable for simple evolutionary models with restrictive assumptions. For more complex and realistic evolutionary models, conventional methods are prohibitively expensive, inaccurate, or even impossible. We instead advocate for a simulation-based inference approach, by using simulated data from an evolutionary model to train a neural network that predicts tree topologies conditioned on sequences. To accurately represent the complex posterior distributions over tree topologies that can arise, we present flexible models that iteratively generate trees using three natural paradigms: top-down, middle-out, and bottom-up. We use a discrete diffusion framework to train these models efficiently on large-scale simulated datasets of phylogenetic trees. For all three generative paradigms, our models fit the data substantially better than the previous state-of-the-art simulation-based method, Phyloformer 2, and obtain more accurate posteriors on real datasets. Finally, our models outperform misspecified conventional methods on data following complex evolutionary processes.

1 Introduction

To understand ancient evolution [93], the origins of pathogens [13], and the function of the immune system [22], biologists collect DNA or protein sequences and try to infer a phylogenetic tree [30]. In addition to the topology of the tree, biologists wish to infer scalar parameters that describe, for example, when a most recent common ancestor has occurred [86], whether there is selective pressure [36], or how fast a pathogen is growing [27]. Just as important as inferring the tree is representing our uncertainty [44]. Bayesian inference provides a natural framework: the posterior represents our belief in the tree, including our uncertainty [30, Ch. 18].

To sample from the posterior, in principle we can use MCMC [43, 83]. However, Bayesian inference tools like MCMC require access to a likelihood, which is only easy to calculate for the simplest models of evolution. More complex models of evolution either (A) have tractable but expensive likelihoods or (B) completely intractable likelihoods.

In models with expensive likelihoods, inference with MCMC is prone to errors [92] and time consuming, in some cases taking weeks to infer a tree with a hundred species [84]. These models include those accounting for different mutation processes at different sites [7, 53], changes of mutation processes over time [89, 19], and different mutation rates at different sites [100] and their correlation between nearby sites [31]. In practice, practitioners speed up inference with strategies like taking the values of evolutionary parameters from previously fit models [56, 57], alternate fitting evolutionary parameters and the tree [71], or relying on simpler models. Unfortunately, these strategies are the sources of many errors in the literature [70, 84, 101].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Other models have intractable likelihoods despite describing ubiquitous biological processes. These include models of sequence insertions and deletions [32], selection-dependence between sites [101], and context-dependent mutation rates [77]. Such models are often dealt with by making strong assumptions [41, 61], discarding data [79, 87], or, more often, ignoring these phenomena entirely.

Unlike likelihood calculation, simulation is simple in nearly all models of evolution, including those with intractable likelihoods. Ideally we could learn to perform inference by training a model on data simulated from the prior [68], a framework known as simulation-based inference. Indeed, this framework has been extremely successful for Bayesian inference in astronomy [21], neuroscience [37], and on tabular data [40]. The main challenge in applying simulation-based inference to this problem is representing a posterior distribution over the combinatorial space of tree topologies, a space of size $(2N-3)!!$. Indeed, inference for scalar evolutionary parameters is mature [34], so here we focus on the hard problem of inferring posteriors over trees.

Previous models use a single forward pass to produce a parametric distribution over trees, limiting expressivity [12]. In this paper, to build flexible models that can support rich posterior distributions, we design iterative generative models that perform multiple forward passes to sample a tree topology from the posterior. In Sec. 4, we describe three natural paradigms for parameterizing iterative generative models for trees: **bottom-up**, **middle-out**, and **top-down**. In Sec. 5, we show we can train large-scale models for each paradigm using a discrete diffusion objective. After training on only 100 GPU hours, our models accurately fit posteriors of trees for data with up to 65 DNA or protein sequences of length > 2500, mapping $> 10^5$ input tokens to a posterior over $> 10^{100}$ topologies.

We show in Sec. 7 that our generative models learn to represent an accurate and calibrated posterior on real and synthetic data in cases where Bayesian inference is tractable, substantially outperforming previous simulation-based models. Moreover, we show that our models successfully learn posteriors under complex evolutionary models where Bayesian inference is intractable, significantly outperforming the conventional strategy of using misspecified models. Our code is available at `https://github.com/AlanNawzadAmin/DeepPhylo`.

2 Background and motivation

In this section, we introduce Bayesian phylogenetic inference, explain why conventional methods cannot be used under realistic evolutionary models, and advocate for simulation-based inference.

2.1 Overview of phylogenetic inference

Let $X_1, \ldots, X_N \in V^L$ be a set of DNA or protein sequences of length L from the genomes of N species (also known as "taxa") descended from a common ancestor, where V is the alphabet of nucleotides or amino acids. We represent how these sequences evolved with a phylogenetic tree T with N leaves, one representing each species. In a phylogenetic tree, each internal node represents the common ancestor of its two children. Phylogenetic trees are bifurcating, meaning each internal node has exactly two children.

In Bayesian phylogenetic inference, we wish to sample from the *posterior* $p(T \mid X_{1:N})$ under some *prior* base distribution over tree topologies $p(T)$ and *likelihood* corresponding to an evolutionary model $p(X_{1:N} \mid T)$. As explained in Sec. 1, we consider inferring the topology of the tree T without predicting branch lengths, though our methods can be easily adapted to accommodate branch length prediction.

In some settings, the root of the tree is not identifiable from sequence data, so T is an unrooted tree. Throughout this paper we consider a rooted T and the nonidentifiability of the root will manifest as its uncertainty in the posterior.

2.2 Priors over tree topologies

There are several classical choices for the prior over tree topologies $p(T)$. The PDA distribution (also known sometimes as simply the uniform model) assigns a uniform probability to each topology, typically producing "unbalanced" trees. In contrast, the Yule model results from assuming a uniform rate of speciation at all nodes, producing relatively balanced trees. In practice, published trees are generally less balanced than Yule trees but more balanced than PDA trees. Ford's alpha model [33] and Aldous' beta model [1] offer two ways to interpolate between these extremes.

2.3 Evolutionary models

The simplest evolutionary model is the Jukes-Cantor (JC) model [48]. The JC model simulates an independent Markov substitution process at each position of the sequence for each species.

Formally, JC parameterizes the evolution of species along a tree T with given branch lengths $r_{u \to v}$ for all parents and children u, v in T. We start with a single species Y at the root, whose sequence is sampled uniformly at random $Y^{(\ell)}_{\text{root}} \sim \text{Uniform}\{A, T, C, G\}$ for all $\ell = 1, 2, \ldots, L$. For each branch, we then simulate an independent uniform Markov substitution process for time $r_{\text{root} \to v}$, mutating $Y^{(\ell)}$ into a different random letter every $\Delta t \sim \text{Exp}(1)$. We continue similarly, simulating the Markov process down the tree T until we have simulated the sequences at all of the leaves $X_1, \ldots, X_N$.

Despite the simplicity of the JC model, it is not obvious how to calculate $p(X_{1:N} \mid T)$, since it requires marginalizing out all ancestral sequences. For this calculation, Felsenstein [29] developed the pruning algorithm, which continues to be the backbone for likelihood calculations in phylogenetic inference. Felsenstein's pruning algorithm can also be applied to calculate likelihoods for some more elaborate models, such as ones that allow different sites to mutate with different rates. However, it strongly relies on the assumption that evolution occurs independently at each position.

Say, when simulating the JC model, the rate of mutation of a letter $Y^{(l)}$ was not independent, but rather depended on the identity of its neighbors $Y^{(l-1)}$ and $Y^{(l+1)}$. Alternatively, say there was a chance for a random chunk of five contiguous letters to be inserted or deleted. These phenomena are ubiquitous in real data, and while it is straightforward to simulate $X_{1:N} \mid T$, computing the likelihood becomes intractable, making typical strategies for posterior inference impossible.

2.4 Simulation-based posterior inference

When we use simple evolutionary models like JC, we can compute likelihoods and perform Bayesian inference with MCMC. However, when likelihood evaluation is expensive or impossible, we cannot perform Bayesian inference with these methods. How can we perform posterior inference when we cannot compute the likelihood $p(X_{1:N} \mid T)$, but we can only sample from this distribution?

We suggest *simulation-based inference* [68], in which we sample many synthetic examples of $(X_{1:N}, T)$ pairs following a given prior and likelihood, and train a generative model to predict T conditioned on $X_{1:N}$. The trained model can then sample from the posterior for T given real sequences $X_{1:N}$! To do so, we need a scalable, model which can learn a full posterior $p(T \mid X_{1:N})$ without access to the likelihood.

3 Related works

There are many works that apply machine learning methods for Bayesian phylogenetic inference. Most of these methods seek to improve upon MCMC methods for Bayesian inference, which are applicable when likelihoods are tractable. These include Hamiltonian Monte Carlo [25], sequential Monte Carlo [90, 64, 18], variational inference [104], and pseudo-likelihood methods [71]. Recent work however has considered likelihood-free inference. Suvorov et al. [85] considered learning to map from $X_{1:N}$ to a single "most-correct" tree T. Finally Blassel et al. [12] suggested learning the posterior distribution $p(T \mid X_{1:N})$.

Can we apply generative models in the literature to our problem? Although many generative models over trees have been proposed for variational likelihood-based inference (VI) [104], summarizing distributions over trees [98], or likelihood-free inference [85], almost all existing methods cannot be used for our problem since

- **they do not learn a full posterior.** Methods that predict quartets [85, 81] or distance matrices [46, 65]; or use a heuristic loss [105] only make point predictions.
- **they do not scale to our setting.** It is not clear how to efficiently learn energy-based models [102] for datasets larger than a few short sequences. And the number of parameters of a split or subsplit network is at least 2^N [52, 103, 104]; when applying these models practitioners restrict support to the small number of splits that were observed in MCMC posteriors, but we don't have that option when we're interested in supporting the full combinatorial space of tree topologies.
- **they require a tractable likelihood to train.** Integrating the continuous encodings of Mimori and Hamada [62] and Duan et al. [28] and training subsplit networks with restricted support [103] is

only possible when optimizing a reverse KL. Also, the GFlowNet framework of Zhou et al. [107] requires a reward model.

- **or they don't condition on sequences.** ARTree [96] and ARTreeFormer [97] summarize MCMC posteriors for a single, **fixed** set of sequences $X_{1:N}$. As a result their architectures do not condition on sequences. And, even with a conditional architecture, training using their paradigm leads to a worse model (see App. A.1).

This leaves, to our knowledge, only Phyloformer 2 [12] as a method appropriate for likelihood-free Bayesian inference. However its flexibility is restricted as it only performs one forward pass to represent a distribution of trees (see App. A.2 for details).

4 Flexible generative models over trees

To perform simulation-based inference for phylogenetic trees, we need a flexible model capable of representing complex posterior distributions over tree topologies. In this section, we start by introducing three natural paradigms for constructing generative models over trees and argue why we should use generative models that use many forward passes to sample a tree.

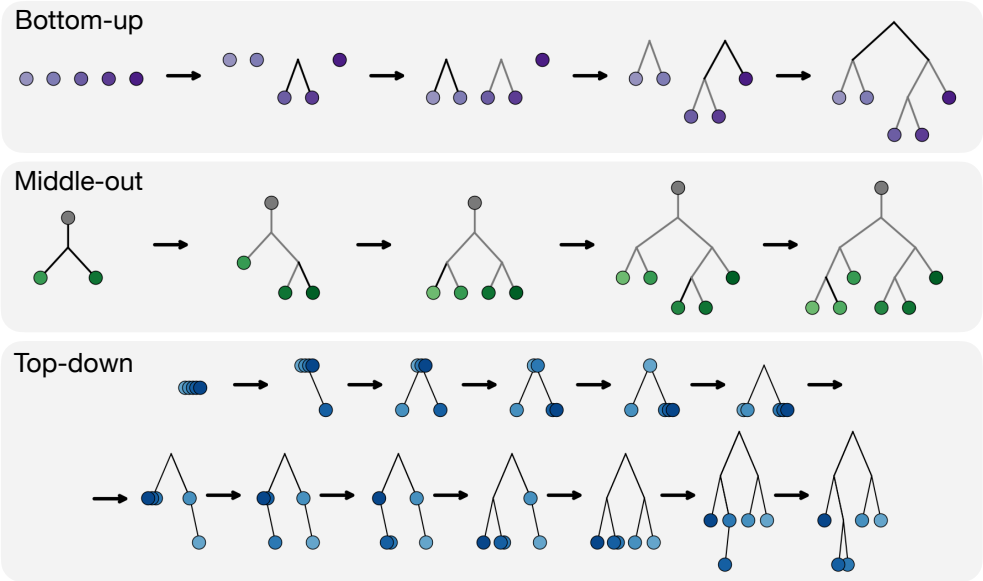


Figure 1: Illustration of three generative paradigms for trees. Each colored circle represents one taxon. For **bottom-up**, we start with each taxon as a separate subtree and repeatedly merge two subtrees until a single bifurcating tree is formed. Dark edges indicate which subtrees were merged in each step. For **middle-out**, we start with a tree containing two taxa and add one taxon at a time. The gray circle depicts the "pseudo-taxon" representing the root of the tree, and dark edges show which taxon was added in each step. For **top-down**, we start with all taxa at the root and percolate them downwards one step at a time until we reach a full bifurcating tree.

4.1 Three generative paradigms for trees

We consider three popular and natural paradigms for parameterizing generative models over trees: **bottom-up**, **middle-out**, and **top-down**. We provide an illustration of the three paradigms in Fig. 1. We discuss alternative paradigms for generative models over trees and their challenges in App. B.

The **bottom-up** paradigm generates trees by starting with each taxon as its own subtree containing one node, and iteratively merging two subtrees together until a complete bifurcating tree is formed.

The **middle-out** paradigm generates trees by adding one taxon to the tree at a time. Specifically, we start by randomly selecting two species and placing them in an unrooted tree, along with a "pseudo-taxon" representing the root of the original tree. We then graft one taxon at a time, attaching it to the current unrooted tree, until all taxa have been added.

The **top-down** paradigm generates trees by percolating taxa down from the root one layer at a time. Specifically, we start with all taxa at the root. At each step, we select one taxon with minimal

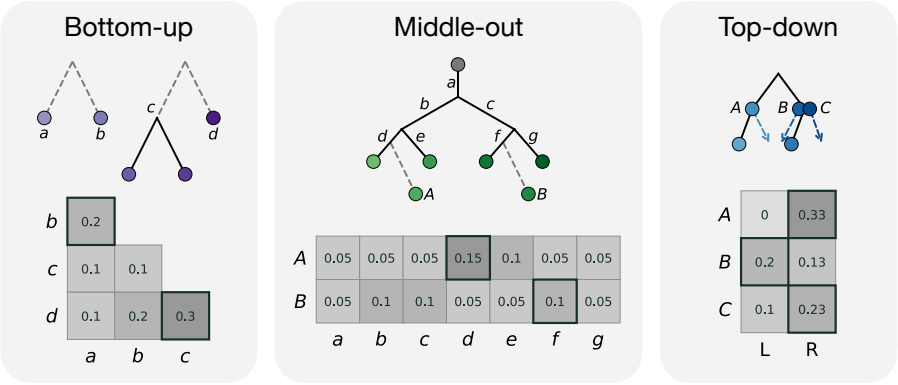


Figure 2: Illustration of sample model predictions for the three generative paradigms. For **bottom-up**, the dashed edges indicate the valid merges consistent with T_0, corresponding to the highlighted cells in the matrix of model predictions. For **middle-out**, the dashed edges indicate the correct attachment edges consistent with T_0, corresponding to the highlighted cells in the matrix of model predictions. For **top-down**, the arrows indicate the correct percolation directions consistent with T_0, corresponding to the highlighted cells in the matrix of t predictions. Taxon A is constrained to percolate right as T_0 is a bifurcating tree.

depth and predict whether to percolate it down to the left or right. We stop percolating when a full bifurcating tree is formed.

All three of these paradigms are explored in the mathematical phylogenetics and generative modeling literature: for example, **bottom-up** has neighbour joining [75] and Phyloformer 2 [12]; **middle-out** has Ford's alpha model [33] and ARTree [96]; and **top-down** has Aldous' beta model [1] and subsplit networks [103]. In none of these paradigms however is there a flexible conditional generative model over trees.

4.2 Achieving flexibility with iterative generative models

How can we develop generative models that are sufficiently flexible to represent arbitrary posterior distributions over tree topologies? We advocate for using *iterative generative models* that perform multiple forward passes through the model to gradually construct a tree one step at a time. Unlike past methods that use a single forward pass to parameterize the full posterior, performing multiple forward passes allows the model to represent arbitrarily complex distributions by conditioning on the intermediate state of the tree before predicting the next step.

Formally, we start the generative process with a structure T_M that depends only on the sequences, without any information about the real tree. We then use a model q_θ to build up the tree one step at a time by repeatedly sampling from the distribution $q_\theta(T_{m-1} \mid T_m, X_{1:N})$. We continue this process for M steps until we have generated the full tree T_0[1].

Fig. 2 illustrates the shape of the model's predictions for a single step of generation under each of the paradigms. For the **bottom-up** paradigm, at each step, the model takes in a set of $m+1$ subtrees, and predicts a distribution of size $\binom{m+1}{2}$ over the next two subtrees to merge. For the **middle-out** paradigm, at each step the model conditions on the current tree and predicts where to attach each missing taxon using a distribution over the tree's edges. For the **top-down** paradigm, at each step the model conditions on the current state of the tree and predicts the probability of each taxon percolating left or right.

Interestingly, a model that naively outputs uniform predictions leads to a different distribution over tree topologies for each of the three paradigms. For **bottom-up** we obtain the Yule distribution, for **middle-out** we obtain the PDA distribution, and for **top-down** we obtain the "almost completely balanced" distribution described in [3]. This might suggest that our choice of generative paradigm depends on our desired prior over tree topologies. However, we show in App. C that all three paradigms can support a full spectrum from PDA to Yule with simple adjustments to the naive uniform predictions, meaning all are sufficiently flexible for our task.

[1]For **middle-out** we could either start with T_M containing two random sequences, or take T_M to be empty and have the first two steps $q_\theta(T_{m-1} \mid T_m, X_{1:N})$ randomly add two sequences. We take the latter convention.

5 Efficiently training our iterative generative models

In this section, we describe how to efficiently train iterative generative models over trees using a discrete diffusion framework, explain our choices for a "noising process" in each paradigm, and provide details on our implementation.

5.1 Training iterative generative models with discrete diffusion

We want to train our models to maximize the likelihood of the simulated data. Letting $T_M, T_{M-1}, \ldots, T_1$ be the states of the tree while generating T_0, the likelihood is:

$$q_\theta(T_0 \mid X_{1:N}) = \sum_{T_1, T_2, \ldots, T_M} \prod_{m=1}^{M} q_\theta(T_{m-1} \mid T_m, X_{1:N}) \tag{1}$$

Unfortunately, directly optimizing such a likelihood is intractable as it requires marginalizing over all possible paths that reach the tree T_0.

Instead, we propose training the model q_θ using the discrete diffusion framework. Specifically, we define a noising process that generates paths $p(T_1, \ldots, T_M \mid T_0)$ by noising T_0; then we train our model to reverse these paths. We do so by optimizing an ELBO of the form: $\log q_\theta(T_0 \mid X_{1:N}) \geq$

$$-M\mathbb{E}_{m\sim \mathrm{U}\{1,M\}, T_m \sim p(\cdot \mid T_0, m)} \left[D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,||\, q_\theta(T_{m-1} \mid T_m, X_{1:N})\right] \tag{2}$$

Training the model now involves simulating a training example $(T_0, X_{1:N})$, applying the noising process $p(T_m \mid T_0)$ for some number of noising steps m, and pushing the model's predictions on T_m closer to the true reverse process that recovers T_0. The ELBO is maximized when $q_\theta(T_{m-1} \mid T_m, X_{1:N}) = \mathbb{E}_{p(T_0 \mid X_{1:N}, T_m)} p(T_{m-1} \mid T_m, T_0)$at which point $q_\theta = p$. We review the derivation of the ELBO and cover related work on discrete diffusion in App. D.

5.2 Rao-Blackwellizing with a closed-form loss

To evaluate the KL in Eqn. 2, we need $p(T_{m-1} \mid T_m, T_0)$. A simple alternative would be to simulate the steps from T_0 up to T_m, and predict the previous step, T_{m-1}, that was visited along that simulation. This approach optimizes the ELBO

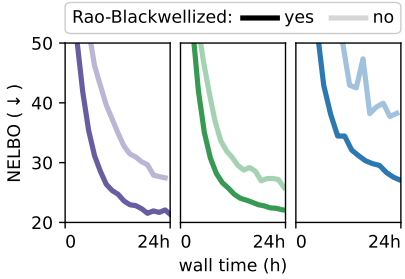


Figure 3: **Our closed form losses are crucial for training.** On the "complex" dataset described below we train models with or without Rao-Blackwellizing the KL estimate in Eqn. 3.

$$\begin{aligned} &-\sum_{m=1}^{M} \mathbb{E}\mathrm{KL}(p(T_{m-1} \mid T_m, T_0) \,||\, q_\theta(T_{m-1} \mid T_m, X_{1:N})) \\ &= -\sum_{m=1}^{M} \mathbb{E}_{T_0, T_1, \ldots, T_m} \log q_\theta(T_{m-1} \mid T_m, X_{1:N}) + \text{const.} \end{aligned} \tag{3}$$

Unfortunately, this approach gives very high variance losses: for every forward pass, we only see a single correct denoising step T_{m-1}.

Instead, we pick noising processes that admit closed-form $p(T_{m-1} \mid T_m, T_0)$. In Eqn. 2 we integrate over all possible paths from T_0 to T_m, giving us not a single correct T_{m-1} but a distribution $p(T_{m-1} \mid T_m, T_0)$ i.e. Rao-Blackwellizing. In Fig. 3, we show that our Rao-Blackwellized training achieves substantially better ELBOs. In the next section, we describe our noising processes and how we compute $p(T_{m-1} \mid T_m, T_0)$.

5.3 Noising process for generative models over trees

To train a discrete diffusion model using Eqn. 2, we must define a noising process $p(T_{1:M} \mid T_0, m)$ carefully so that the target distribution $p(T_{m-1} \mid T_m, T_0)$ is computable closed-form.

For **bottom-up**, at the start of the noising process, we label the root node as "exposed". Then, each noising step involves removing an exposed internal node from the current graph, "exposing" its two children. We select the next node to remove in each step as:

$$p_{\mathrm{BU}}(T_m = \texttt{remove}(T_{m-1}, \texttt{node}) \mid T_{m-1}) \propto |\texttt{descendants}(\texttt{node})|$$

We multiply by the number of descendants of `node`, not counting `node`, to get a tractable form for $p(T_{m-1} \mid T_m, T_0)$ calculated with two tree traversals described in App. E.2.

For **middle-out**, each noising step involves pruning one taxon from the tree and collapsing the edges connecting its two siblings. Formally, we have, for a taxon v:

$$p_{\text{MO}}(T_m = \texttt{prune}(T_{m-1}, v) \mid T_{m-1}) = \frac{1}{N - m + 1}$$

Here, the distribution $p(T_{m-1} \mid T_m, T_0)$ is again uniform over all valid attachments in T_m that are consistent with T_0. Note that for a given taxon, there is only one attachment edge in T_m that is consistent with the ground truth T_0. Note also that our target has multiple non-zero entries corresponding to each of the taxa not in the current tree.

For **top-down**, each noising step involves selecting a taxon of maximal depth and percolating it up one layer to its parent. The noising process continues until all taxa reach the root node, at T_M. Formally, for a tree T_{m-1} with maximum depth d, we have:

$$p_{\text{TD}}(T_m = \texttt{percolate_up}(T_{m-1}, \texttt{taxon}) \mid T_{m-1}) = \begin{cases} 0, & \texttt{depth}(\texttt{taxon}) < d \\ \frac{1}{\texttt{\# of taxa at depth } d}, & \text{otherwise} \end{cases}$$

The distribution $p(T_{m-1} \mid T_m, T_0)$ is uniform over all percolations in T_m that are consistent with T_0.

Noise schedule In the ELBO from eq. (2), we use a uniform distribution over the number of noising steps m. However, in practice some noising steps may be much easier for the model to learn than others. Ideally, to train as efficiently as possible, we would like to sample training examples according to the difficulty, by adapting the noise schedule. In sec. E.1, we describe how to build a noise schedule mirroring the mutual information schedules of Austin et al. [6]. In App. H.1 however we show this makes little difference in practice, so future practitioners may proceed with the uniform noise schedule without having to implement or derive a mutual information schedule.

5.4 Architecture

We adapt the EvoPF architecture from [12]. EvoPF embeds N sequences of length L, $X_{1:N}$, into a tensor of size $[N, L, 128]$ and performs axial attention over the dimensions N and L separately. Finally it performs a mean across the length dimension to get a matrix of $[N, 128]$ embeddings. We make two modifications. We provide full details in App. F and ablations showing these two modifications were crucial for performance in Fig. 4 (details in App. H.2).

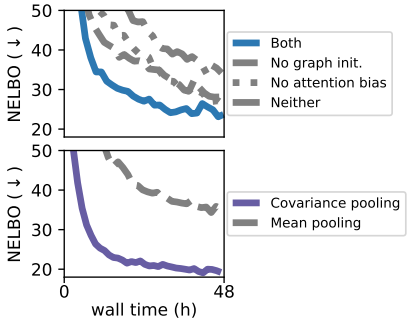


Figure 4: **Our strategies for conditioning on the graph and covariance pooling are crucial for training.** We ablate (Top) our graph conditioning strategies and (Bottom) our covariance pooling. Each is shown on an example paradigm: **top-down** and **bottom-up**.

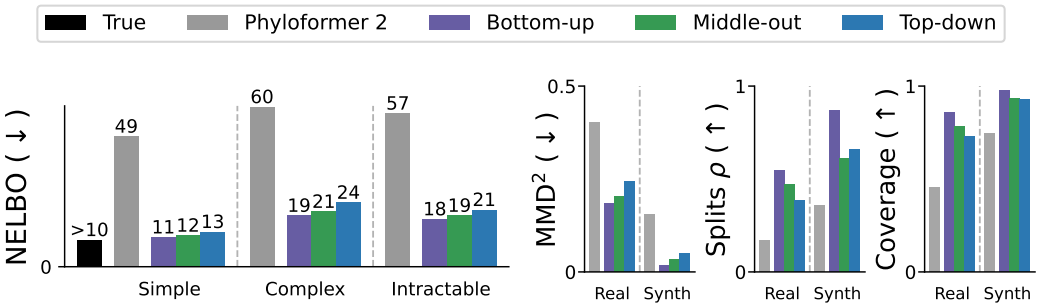
First we need to condition on the current state of the graph T_m. So we initialized species embeddings with graph topology information, and biased attention among species using graph distance.

Second, near initialization, the mean over $L > 1000$ positions returns nearly identical embeddings across taxa: if the embedding at each position represents only its nucleotide, then the mean is only the base composition, nearly the same for every taxon. We developed a new architecture we call **covariance-pooling** which embeds the average similarity between pairs of nodes: if embeddings represent nucleotides, now we get the Hamming similarity.

Also, as noted in Sec. 4.2, each paradigm is capable of representing prior distributions ranging from Yule to PDA. We initialize at the desired prior by multiplying the neural network outputs with these weights. In App. H.1 we show our models can learn these weights even without this initialization.

6 Experiment details

We train our models on synthetic data simulated following a prior $p(T)$ and evolutionary model $p(X_{1:N} \mid T)$. We set $p(T)$ to be a PDA prior on trees to match the priors in MrBayes.



(a) ELBOs on training data

(b) Posterior calibration on real and synthetic datasets

Figure 5: **(Left)** Our models obtain substantially better negative ELBOs than Phyloformer 2. The ELBO of a trivial model that scores all trees uniformly is 157 nats. **(Right)** Our models reconstruct MCMC posteriors on real datasets more accurately than Phyloformer 2.

We consider three types of evolutionary models $p(X_{1:N} \mid T)$ for our experiments. First, we train our models on the simplest model of DNA evolution (**JC**). As described in Sec. 2.3, the JC model yields tractable likelihoods, allowing us to use MrBayes to obtain gold standard MCMC samples from the posterior. Next, we consider two more realistic evolutionary models: a more *complex model* accounting for natural selection in proteins (**CAT**) and a model of DNA evolution that has *intractable* likelihoods because of the presence of insertions and deletions (**InDel**).

We train our models to support datasets with between $N = 25$ and $N = 65$ species, and sequences of length $L = 300$ to $L = 2600$, based on the sizes of 8 classical evolutionary datasets [103]. Our models can therefore be immediately applied to most evolutionary and immunological inference tasks on 1-5 genes. We discuss scaling to larger N and L in Sec. 8. We provide further details on our simulated training datasets in App. G.1.

We use Adam with μP-rescaled per-layer learning rates [99] (base $\mathrm{lr} = 0.01$), CompleteP scaled residual layers [23], 1,000-step linear warmup, EMA 0.9998, gradient-norm clip 500, `bf16-mixed` precision, and an effective batch size of 32 trees per step. Each model trains for 48 hours on 2 A100 GPUs, which is approximately $800,000$ datapoints for each paradigm. Our experiments with Phyloformer 2 are controlled for compute: we train their models according to their default hyperparameters for 48 hours on 2 A100 GPUs, which is approximately $1,100,000$ datapoints. We also reduce the pair-attention dimension in our EvoPF architecture from 256 to 128, giving our models 13.6 million parameters compared to 20.0 million for our baseline, Phyloformer 2.

7 Results

In this section, we provide our empirical results. We compare against Phyloformer 2, the previous state-of-the-art method for simulation-based inference. First, we show that our models learn to fit simulated data much better than Phyloformer 2, as measured by ELBOs. Second, we show that our models match gold standard MCMC posteriors more accurately than Phyloformer 2. Lastly, we show that our models accurately infer the correct tree for the complex and intractable evolutionary models, improving accuracy over the best existing alternative.

We remind the reader that real evolutionary data almost never has a "true" tree that we can observe. Our ground truth always depends on the practitioners' prior $p(T, X_{1:N})$ and our goal is to show that we can perform accurate posterior inference even for priors that have been traditionally challenging to model.

7.1 Our models are state-of-the-art for simulation-based phylogenetic inference

In Fig. 5a, we compare the ELBOs achieved by our models against Phyloformer 2. We find that across datasets and generative paradigms, our models consistently fit the data much better than Phyloformer 2. These results confirm that our flexible parameterization and efficient training via the discrete diffusion ELBO significantly improve the performance of simulation-based inference for phylogenetics. More generally, our work is the first to successfully apply the **top-down** and **middle-out** paradigms to this problem, and we show that both of these paradigms can far surpass the performance of the previous state-of-the-art. In App. H.3 we show through ablations that our

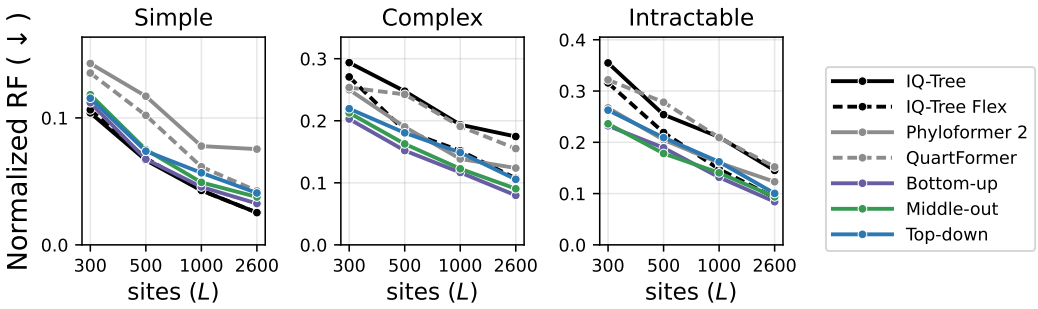


Figure 6: Our models make more accurate predictions than IQ-Tree for the complex and intractable evolutionary models. We hold the number of taxa $N = 45$ fixed and vary the sequence length L. The normalized RF distance of a random tree is roughly 1. In fig. 10, we instead fix $L = 500$ and vary N.

improvement over Phyloformer 2 comes from our new training objective and the flexibility of our framework rather than a pessimistic ELBO or architectural changes.

For the JC data we are able to estimate the true average entropy of the data using MCMC (App. G.4.3) at > 10 nats, which is the uncertainty of a uniform distribution over $e^{10} \approx 20,000$ trees. This demonstrates the significant uncertainty in the true tree even in the simplest synthetic settings, highlighting the importance of fitting a posterior distribution. We see our models have an average KL of < 3 nats (difference between our ELBO and the entropy) suggesting an absolute accurate fit.

On these data **bottom-up** consistently outperformed **middle-out** which in turn outperformed **top-down**. This could be because the many noising steps of **top-down** increase the variance of the loss during training. The superior performance of **bottom-up** may also reflect the bottom-up nature of the likelihood calculation on simple phylogenetic data [29].

7.2 Our models accurately recover calibrated MCMC posteriors on real data

We now take eight classical **real-world** evolutionary datasets from Treebase [69], a collection of inferred trees, originally selected in Lakner et al. [50] (App. G.2 for details). These include diverse data from tetrapod ribosomal DNA to insect *wingless* gene DNA, $N = 27$ to $N = 64$, and $L = 378$ to $L = 2520$, and they are numbered DS1-8 in order of increasing N. We also simulate JC data with the same N and L to each of these datasets. For each dataset, we infer 1000 trees using our JC models, and compare the distribution of our inferred trees to the true posterior estimated by MrBayes under the JC model. We remove constant sites in the sequence and shuffled the order of sites before inferring the tree.

In Fig. 5b we report MMD distance between the posteriors of each model and the gold standard posterior samples from MrBayes for real and synthetic data; we also compare the Spearman correlation of the posterior probabilities of tree partitions and "coverage" of high probability partitions. See App. H.5 for full results, tree samples, and an explanation of these metrics. Our models accurately recover the posterior, obtaining much closer posteriors than the previous state-of-the-art Phyloformer 2. However performance is superior on synthetic data, indicating room for improvement.

7.3 Our models successfully infer trees for complex evolutionary models

For the CAT and InDel datasets, obtaining accurate MCMC posteriors is prohibitively expensive or impossible. To make predictions in these settings, practitioners often resort to point estimates using workhorse maximum likelihood methods such as IQ-Tree. Practitioners use these procedures despite knowing they are misspecified in hopes that the evidence still points to the correct tree.

For each method we infer 8 trees for every set of sequences $X_{1:N}$ and take the majority-rule unrooted consensus tree, allowing for polytomy. We evaluate the unrooted Robinson-Foulds (RF) distance with respect to the true tree. As an additional baseline, we also train Quartformer models, adapting the released code from [82]. See sec. G.6 for more details on our Quartformer implementation.

We show our results for $N = 45$ and $L = 500$ in Fig. 6. We see all of our models outperform Phyloformer 2. Our models also perform just as well as IQ-Tree on the JC data where it is well-specified. On the CAT and InDel datasets, IQ-Tree ignores more complex effects leading to lower-

quality predictions. In contrast, our models are able to leverage their knowledge about complex evolutionary processes and indels to make better predictions than IQ-Tree. This is also true on the complex data where IQ-Tree Flex in principle models the same sort of site-variability and non uniform mutation processes that we're including in our simulated data; the mismatch between the exact form of the complex data prior and the effects IQ-Tree Flex is modelling however allows our models to outperform. App. H.6 shows run-time comparisons, showing our **bottom-up** and **middle-out** models are also faster than IQ-Tree for most N, L settings.

These results demonstrate that our models can give practitioners higher-quality point estimates of the tree under realistic evolutionary models. Moreover, in contrast with bespoke algorithms such as IQ-Tree, our models simply require the practitioner to provide simulated data.

8 Discussion

We develop flexible deep generative models that can be trained to accurately model posteriors over phylogenetic trees. Our models are trained on simulated data only, allowing them to learn posteriors for more realistic evolutionary models where traditional Bayesian approaches are intractable. We found, of the three paradigms, **bottom-up** to be consistently most accurate, but all three are natural starting points for extensions, and future practitioners may find the mathematics of **middle-out** or **top-down** more amenable to their setting. Our contributions open up promising directions for future work: building a universal default tree prior, building better models for inferring evolutionary parameters, and scaling to larger datasets.

A universal tree prior There are a plethora of evolutionary models proposed in the literature. In many cases, practitioners may not know which evolutionary model best explains their data, and resort to using a model that is a poor match for the data. For many protein and DNA evolution datasets, the `GTR+G4+I` prior in MrBayes [43] acts as a default, and many evolutionary parameters come with common and default values used nearly universally for inference [84]. Our framework opens the door for a phylogenetic inference foundation model: by training on simulated data from a variety of evolutionary models, the model automatically selects the appropriate evolutionary process as part of the posterior inference, and can be used across a large variety of applications. The foundation model can also be fine-tuned on simulated data when a particular evolutionary model is desired.

Inferring evolutionary parameters In this paper we focused on inference of the combinatorially-challenging tree topology T. However, in many settings we are also interested in inferring other parameters, such as branch lengths. Our work can be extended using off-the-shelf Bayesian inference methods to infer these Euclidean parameters [90, 68, 34].

Species trees and reticulations We have focused here on inferring the sequence evolution of individual genes. However real evolutionary data has complex phenomena such as recombination or incomplete lineage sorting which require reticulated or species trees [45, 30, Ch. 27, 28]; real sequence data, such as pandemic data, can also come with information about the time it was collected [26]. Future work could extend our models to deal with these data as well.

Larger scale inference We train our models on datasets with up to $N = 65$ sequences of length up to $L = 2600$. These sizes are sufficient for most important phylogenetic tasks. It is a misconception that large-data applications such as pandemic data or full-genome analyses necessarily involve the analysis of $N =$ thousands or $L =$ millions all at once. Instead, practitioners more often infer sub-trees for subsets of N or individual genes $L \approx 1000$ and combine their inferred trees using super-tree methods [91, 47], or species-tree methods [63]. Our method can be combined with these, or scale much further by using methods that have enabled large-scale graph foundation models [9].

Reducing uncertainty We saw in App. H.5 that our posterior predictions were biased to be under-confident, a hallmark of models trained to maximize likelihood. For language models trained to maximize likelihood, the most popular strategy to get high quality samples is reinforcement learning [66] or sampling strategies [42]. Unlike the directions in the rest of this section, adapting these strategies to phylogenetic inference will involve new algorithms, taking care to not to miscalibrate the uncertainty prediction.

Acknowledgements This work was supported by DARPA AIQ HR00112590066, NSF CAREER IIS-2145492, and NSF CDS&E-MSS 2134216. We thank Noor Youssef, Edward Susko, and Andres Potapczynski for helpful discussions.

References

[1] D. Aldous. Probability distributions on cladograms. In D. Aldous and R. Pemantle, editors, *Random Discrete Structures*, pages 1–18, New York, NY, 1996. Springer New York. ISBN 978-1-4612-0719-1.

[2] D. J. Aldous. Mixing time for a markov chain on cladograms. *Comb. Probab. Comput.*, 9(3): 191–204, May 2000.

[3] D. J. Aldous. Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. *Stat. Sci.*, 16(1):23–34, Feb. 2001.

[4] B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Ann. Comb.*, 5(1):1–15, June 2001.

[5] A. N. Amin, N. Gruver, and A. G. Wilson. Why masking diffusion works: Condition on the jump schedule for improved discrete diffusion. In *Frontiers in Probabilistic Inference: Learning meets Sampling*, Apr. 2025.

[6] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Adv. Neural Inf. Process. Syst.*, 34:17981–17993, 2021.

[7] H. Banos, T. K. F. Wong, J. Daneau, E. Susko, B. Q. Minh, R. Lanfear, M. W. Brown, L. Eme, and A. J. Roger. GTRpmix: A linked general time-reversible model for profile mixture models. *Mol. Biol. Evol.*, 41(9):msae174, Sept. 2024.

[8] E. Baron, A. N. Amin, R. Weitzman, D. S. Marks, and A. G. Wilson. A diffusion model to shrink proteins while maintaining their function. *ICLR*, 2026.

[9] M. Bechler-Speicher, Y. Gottlieb, A. Isakov, D. Abensur, A. Tavory, D. Haimovich, I. Guy, and U. Weinsberg. Billion-scale graph foundation models, 2026. URL `https://arxiv.org/abs/2602.04768`.

[10] J. Bertoin. Exchangeable coalescents. Technical report, ETH Z¨urich, 2010.

[11] L. J. Billera, S. P. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees. *Adv. Appl. Math.*, 27(4):733–767, Nov. 2001.

[12] L. Blassel, B. Boussau, N. Lartillot, and L. Jacob. Likelihood-free inference of phylogenetic tree posterior distributions. *arXiv [q-bio.PE]*, Oct. 2025.

[13] J. D. Bloom. The data are insufficient to confidently root the SARS-CoV-2 phylogenetic tree. *Mol. Biol. Evol.*, 42(6):msaf118, June 2025.

[14] M. Bohde, M. Manjrekar, R. Wang, S. Ji, and C. W. Coley. DiffMS: Diffusion generation of molecules conditioned on mass spectra. *arXiv [cs.LG]*, Feb. 2025.

[15] M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Ann. Comb.*, 8(4):409–423, Jan. 2005.

[16] A. V. Brower. Phylogenetic relationships among the nymphalidae (lepidoptera) inferred from partial sequences of the wingless gene. *Proc. Biol. Sci.*, 267(1449):1201–1211, June 2000.

[17] A. Campbell, J. Benton, V. De Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, Oct. 2022.

[18] A. Chen, P. Chlenski, K. Munyuza, A. K. Moretti, C. A. Naesseth, and I. Pe'er. Variational combinatorial sequential Monte Carlo for Bayesian phylogenetics in hyperbolic space. *arXiv [cs.LG]*, July 2025.

[19] S. M. Crotty, B. Q. Minh, N. G. Bean, B. R. Holland, J. Tuke, L. S. Jermiin, and A. V. Haeseler. GHOST: Recovering historical signal from heterotachously evolved sequence alignments. *Syst. Biol.*, 69(2):249–264, Mar. 2020.

[20] B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On computing the nearest neighbor interchange distance. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 125–143. American Mathematical Society, Providence, Rhode Island, Dec. 2000.

[21] M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf. Real-time gravitational wave science with neural posterior estimation. *Phys. Rev. Lett.*, 127(24):241103, Dec. 2021.

[22] W. S. DeWitt, 3rd, L. Mesin, G. D. Victora, V. N. Minin, and F. A. Matsen, 4th. Using genotype abundance to improve phylogenetic inference. *Mol. Biol. Evol.*, 35(5):1253–1265, May 2018.

[23] N. Dey, B. C. Zhang, L. Noci, M. Li, B. Bordelon, S. Bergsma, C. Pehlevan, B. Hanin, and J. Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers. *Neurips 2025*, May 2025.

[24] P. Diaconis and S. Holmes. Random walks on trees and matchings. *Electron. J. Probab.*, 7 (none):1–17, Jan. 2002.

[25] V. Dinh, A. Bilge, C. Zhang, and F. A. Matsen, IV. Probabilistic path Hamiltonian Monte Carlo. *arXiv [q-bio.PE]*, Feb. 2017.

[26] A. J. Drummond, O. G. Pybus, A. Rambaut, R. Forsberg, and A. G. Rodrigo. Measurably evolving populations. *Trends Ecol. Evol.*, 18(9):481–488, Sept. 2003.

[27] A. J. Drummond, A. Rambaut, B. Shapiro, and O. G. Pybus. Bayesian coalescent inference of past population dynamics from molecular sequences. *Mol. Biol. Evol.*, 22(5):1185–1192, May 2005.

[28] C. Duan, Z. Zang, S. Li, Y. Xu, and S. Z. Li. PhyloGen: Language model-enhanced phylogenetic inference via graph structure generation. *38th Conference on Neural Information Processing Systems*, Dec. 2024.

[29] J. Felsenstein. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Syst. Biol.*, 22(3):240–249, Sept. 1973.

[30] J. Felsenstein. *Inferring Phylogenies*. 2004.

[31] J. Felsenstein and G. A. Churchill. A hidden Markov model approach to variation among sites in rate of evolution. *Mol. Biol. Evol.*, 13(1):93–104, Jan. 1996.

[32] W. Fletcher and Z. Yang. INDELible: a flexible simulator of biological sequence evolution. *Mol. Biol. Evol.*, 26(8):1879–1888, Aug. 2009.

[33] D. J. Ford. Probabilities on cladograms: introduction to the alpha model. *arXiv [math.PR]*, Nov. 2005.

[34] M. Fourment, M. Macaulay, C. J. Swanepoel, X. Ji, M. A. Suchard, and F. A. Matsen, IV. Torchtree: flexible phylogenetic model development and inference using PyTorch. *arXiv [q-bio.PE]*, June 2024.

[35] A. Gavryushkin and A. J. Drummond. The space of ultrametric phylogenetic trees. *J. Theor. Biol.*, 403:197–208, Aug. 2016.

[36] N. Goldman and Z. Yang. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.*, 11(5):725–736, Sept. 1994.

[37] P. J. Gonçalves, J.-M. Lueckmann, M. Deistler, M. Nonnenmacher, K. Öcal, G. Bassetto, C. Chintaluri, W. F. Podlaski, S. A. Haddad, T. P. Vogels, D. S. Greenberg, and J. H. Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9(e56261):e56261, Sept. 2020.

[38] M. Havasi, B. Karrer, I. Gat, and R. T. Q. Chen. Edit flows: Flow matching with edit operations. June 2025.

[39] S. B. Hedges, K. D. Moberg, and L. R. Maxson. Tetrapod phylogeny inferred from 18S and 28S ribosomal RNA sequences and a review of the evidence for amniote relationships. *Mol. Biol. Evol.*, 7(6):607–633, Nov. 1990.

[40] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, Jan. 2025.

[41] I. Holmes and W. J. Bruno. Evolutionary HMMs: A Bayesian approach to multiple alignment. *Bioinformatics*, 17(9):803–820, Sept. 2001.

[42] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv [cs.CL]*, Apr. 2019.

[43] J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, Aug. 2001.

[44] J. P. Huelsenbeck, B. Rannala, and J. P. Masly. Accommodating phylogenetic uncertainty in evolutionary studies. *Science*, 288(5475):2349–2350, June 2000.

[45] P. J. Humphries, S. Linz, and C. Semple. Cherry picking: A characterization of the temporal hybridization number for a set of phylogenies. *Bull. Math. Biol.*, 75(10):1879–1890, Oct. 2013.

[46] Y. Jiang, M. Balaban, Q. Zhu, and S. Mirarab. DEPP: Deep learning enables extending species trees using single genes. *Syst. Biol.*, 72(1):17–34, May 2023.

[47] Y. Jiang, D. McDonald, D. Perry, R. Knight, and S. Mirarab. Scaling DEPP phylogenetic placement to ultra-large reference trees: a tree-aware ensemble approach. *Bioinformatics*, 40 (6):btae361, June 2024.

[48] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In *Mammalian Protein Metabolism*, pages 21–132. Elsevier, 1969.

[49] D. P. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In *NeurIPS 2021*, July 2021.

[50] C. Lakner, P. van der Mark, J. P. Huelsenbeck, B. Larget, and F. Ronquist. Efficiency of markov chain monte carlo tree proposals in bayesian phylogenetics. *Syst. Biol.*, 57(1):86–103, Feb. 2008.

[51] A. Lambert and T. Stadler. Birth-death models and coalescent point processes: the shape and probability of reconstructed phylogenies. *Theor. Popul. Biol.*, 90:113–128, Dec. 2013.

[52] B. Larget. The estimation of tree posterior probabilities using conditional clade probability distributions. *Syst. Biol.*, 62(4):501–511, July 2013.

[53] N. Lartillot and H. Philippe. A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. *Mol. Biol. Evol.*, 21(6):1095–1109, June 2004.

[54] N. Lartillot, N. Rodrigue, D. Stubbs, and J. Richer. PhyloBayes MPI: phylogenetic reconstruction with infinite mixtures of profiles in a parallel environment. *Syst. Biol.*, 62(4):611–615, July 2013.

[55] T. Lassmann. Kalign 3: multiple sequence alignment of large data sets. *Bioinformatics*, 36(6): 1928–1929, Oct. 2019.

[56] S. Q. Le and O. Gascuel. An improved general amino acid replacement matrix. *Mol. Biol. Evol.*, 25(7):1307–1320, July 2008.

[57] S. Q. Le, C. C. Dang, and O. Gascuel. Modeling protein evolution with several amino acid replacement matrices depending on site rates. *Mol. Biol. Evol.*, 29(10):2921–2936, Oct. 2012.

[58] N. Ly-Trong, S. Naser-Khdour, R. Lanfear, and B. Q. Minh. AliSim: A fast and versatile phylogenetic sequence simulator for the genomic era. *Mol. Biol. Evol.*, 39(5):msac092, May 2022.

[59] W. Löhr, L. Mytnik, and A. Winter. The aldous chain on cladograms in the diffusion limit. *arXiv [math.PR]*, May 2018.

[60] D. Lüdke, E. R. Raventós, M. Kollovieh, and S. Günnemann. Unlocking point processes through point set diffusion. *arXiv [cs.LG]*, Oct. 2024.

[61] F. A. Matsen, 4th and P. L. Ralph. Enabling inference for context-dependent models of mutation by bounding the propagation of dependency. *J. Comput. Biol.*, 29(8):802–824, Aug. 2022.

[62] T. Mimori and M. Hamada. GeoPhy: Differentiable phylogenetic inference via geometric gradients of tree topologies. *37th Conference on Neural Information Processing Systems*, July 2023.

[63] S. Mirarab, R. Reaz, M. S. Bayzid, T. Zimmermann, M. S. Swenson, and T. Warnow. ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics*, 30(17):i541–8, Sept. 2014.

[64] A. K. Moretti, L. Zhang, C. A. Naesseth, H. Venner, D. Blei, and I. Pe'er. Variational combinatorial sequential Monte Carlo methods for Bayesian phylogenetic inference. *arXiv [stat.ML]*, (Uai), May 2021.

[65] L. Nesterenko, L. Blassel, P. Veber, B. Boussau, and L. Jacob. Phyloformer: Fast, accurate and versatile phylogenetic reconstruction with deep neural networks. *Bioinformatics*, (biorxiv;2024.06.17.599404v1), June 2024.

[66] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. *arXiv [cs.CL]*, Mar. 2022.

[67] M. Owen and J. S. Provan. A fast algorithm for computing geodesic distances in tree space. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(1), Jan. 2011.

[68] G. Papamakarios and I. Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. *30th Conference on Neural Information Processing Systems*, 2016.

[69] W. H. Piel, M. Donoghue, and M. Sanderson. Tree-BASE: A database of phylogenetic information. 2000.

[70] D. Pisani, W. Pett, M. Dohrmann, R. Feuda, O. Rota-Stabelli, H. Philippe, N. Lartillot, and G. Wörheide. Genomic data do not support comb jellies as the sister group to all other animals. *Proc. Natl. Acad. Sci. U. S. A.*, 112(50):15402–15407, Dec. 2015.

[71] S. Prillo, Y. Deng, P. Boyeau, X. Li, P.-Y. Chen, and Y. S. Song. CherryML: scalable maximum likelihood estimation of phylogenetic models. *Nat. Methods*, 20(8):1232–1236, Aug. 2023.

[72] H. Richman, C. Zhang, and F. A. Matsen, IV. Vector encoding of phylogenetic trees by ordered leaf attachment. *arXiv [q-bio.PE]*, Mar. 2025.

[73] A. Y. Rossman, J. M. McKemy, R. A. Pardo-Schultheiss, and H.-J. Schroers. Molecular studies of the bionectriaceae using large subunit rDNA sequences. *Mycologia*, 93(1):100–110, Jan. 2001.

[74] R. Sainudiin and A. Veber. A beta-splitting model for evolutionary trees. *arXiv [math.PR]*, Nov. 2015.

[75] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4(4):406–425, July 1987.

[76] R. Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *Graph Drawing*, Lecture Notes in Computer Science, pages 355–366. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[77] V. B. Seplyarskiy, R. A. Soldatov, E. Koch, R. J. McGinty, J. M. Goldmann, R. D. Hernandez, K. Barnes, A. Correa, E. G. Burchard, P. T. Ellinor, S. T. McGarvey, B. D. Mitchell, R. S. Vasan, S. Redline, E. Silverman, S. T. Weiss, D. K. Arnett, J. Blangero, E. Boerwinkle, J. He, C. Montgomery, D. C. Rao, J. I. Rotter, K. D. Taylor, J. A. Brody, Y.-D. I. Chen, L. de las Fuentes, C.-M. Hwu, S. S. Rich, A. W. Manichaikul, J. C. Mychaleckyj, N. D. Palmer, J. A. Smith, S. L. R. Kardia, P. A. Peyser, L. F. Bielak, T. D. O'Connor, L. S. Emery, NHLBI Trans-Omics for Precision Medicine (TOPMed) Consortium, TOPMed Population Genetics Working Group, C. Gilissen, W. S. W. Wong, P. V. Kharchenko, and S. Sunyaev. Population sequencing data reveal a compendium of mutational processes in the human germ line. *Science*, 373(6558):1030–1035, Aug. 2021.

[78] J. Shi, K. Han, Z. Wang, A. Doucet, and M. K. Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv [cs.LG]*, June 2024.

[79] A. Siepel and D. Haussler. Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. *Mol. Biol. Evol.*, 21(3):468–488, Mar. 2004.

[80] K. Strimmer and A. von Haeseler. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, 13(7):964–969, Sept. 1996.

[81] J. Su and X. Jiao. Quartformer: An accurate deep learning framework for phylogenetic tree construction. *Bioinformatics*, (biorxiv;2025.04.05.646867v1), Apr. 2025.

[82] J. Su and X. Jiao. Quartformer: An accurate deep learning framework for phylogenetic tree construction. *bioRxiv*, page 2025.04.05.646867, Apr. 2025.

[83] M. A. Suchard, P. Lemey, G. Baele, D. L. Ayres, A. J. Drummond, and A. Rambaut. Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10. *Virus Evol.*, 4(1):vey016, Jan. 2018.

[84] E. Susko, L. Lincker, and A. J. Roger. Accelerated estimation of frequency classes in site-heterogeneous profile mixture models. *Mol. Biol. Evol.*, 35(5):1266–1283, May 2018.

[85] A. Suvorov, J. Hochuli, and D. R. Schrider. Accurate inference of tree topologies from multiple sequence alignments using deep learning. *Syst. Biol.*, 69(2):221–233, Mar. 2020.

[86] K. Tamura, F. U. Battistuzzi, P. Billing-Ross, O. Murillo, A. Filipski, and S. Kumar. Estimating divergence times in large molecular phylogenies. *Proc. Natl. Acad. Sci. U. S. A.*, 109(47): 19333–19338, Nov. 2012.

[87] G. Tan, M. Muffato, C. Ledergerber, J. Herrero, N. Goldman, M. Gil, and C. Dessimoz. Current methods for automated filtering of multiple sequence alignments frequently worsen single-gene phylogenetic inference. *Syst. Biol.*, 64(5):778–791, Sept. 2015.

[88] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard. DiGress: Discrete denoising diffusion for graph generation. *arXiv [cs.LG]*, Sept. 2022.

[89] H.-C. Wang, M. Spencer, E. Susko, and A. J. Roger. Testing for covarion-like evolution in protein sequences. *Mol. Biol. Evol.*, 24(1):294–305, Jan. 2007.

[90] L. Wang, A. Bouchard-Côté, and A. Doucet. Bayesian phylogenetic inference using a combinatorial sequential Monte Carlo method. *J. Am. Stat. Assoc.*, 110(512):1362–1374, Oct. 2015.

[91] T. Warnow. Supertree construction: Opportunities and challenges. *arXiv [q-bio.PE]*, May 2018.

[92] N. V. Whelan and K. M. Halanych. Who let the CAT out of the bag? accurately dealing with substitutional heterogeneity in phylogenomic analyses. *Syst. Biol.*, 66(2):232–255, Mar. 2017.

[93] N. V. Whelan, K. M. Kocot, T. P. Moroz, K. Mukherjee, P. Williams, G. Paulay, L. L. Moroz, and K. M. Halanych. Ctenophore relationships and their placement as the sister group to all other animals. *Nat. Ecol. Evol.*, 1(11):1737–1746, Nov. 2017.

[94] T. K. F. Wong, N. Ly-Trong, H. Ren, P. Demotte, H. Baños, A. J. Roger, E. Susko, C. Bielow, N. De Maio, N. Goldman, M. W. Hahn, M. dos Reis, L. Sy Vinh, G. Huttley, R. Lanfear, and B. Q. Minh. IQ-TREE 3: Phylogenomic inference software using complex evolutionary models. *Mol. Biol. Evol.*, (msag117):msag117, May 2026.

[95] W. M. Woodman and T. M. W. Nye. Brownian motion, bridges and bayesian inference in phylogenetic tree space. *arXiv [stat.ME]*, June 2025.

[96] T. Xie and C. Zhang. ARTree: A deep autoregressive model for phylogenetic inference. *arXiv [q-bio.PE]*, Oct. 2023.

[97] T. Xie, Y. Mao, and C. Zhang. Artreeformer: A faster attention-based autoregressive model for phylogenetic inference. *PLOS Computational Biology*, 21(12):e1013768, 2025. doi: 10.1371/journal.pcbi.1013768. URL `https://doi.org/10.1371/journal.pcbi.1013768`.

[98] T. Xie, H. Richman, J. Gao, F. A. Matsen, IV, and C. Zhang. PhyloVAE: Unsupervised learning of phylogenetic trees via variational autoencoders. *arXiv [stat.ML]*, Feb. 2025.

[99] G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv [cs.LG]*, Mar. 2022.

[100] Z. Yang. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J. Mol. Evol.*, 39(3):306–314, Sept. 1994.

[101] N. Youssef, E. Susko, and J. P. Bielawski. Consequences of stability-induced epistasis for substitution rates. *Mol. Biol. Evol.*, 37(11):3131–3148, Nov. 2020.

[102] C. Zhang. Learnable topological features for phylogenetic inference via graph neural networks. *arXiv [stat.ML]*, Feb. 2023.

[103] C. Zhang and F. A. Matsen, IV. Generalizing tree probability estimation via Bayesian networks. *arXiv [stat.AP]*, (NeurIPS):1444–1453, May 2018.

[104] C. Zhang and F. A. Matsen, IV. A variational approach to Bayesian phylogenetic inference. *J. Mach. Learn. Res.*, 25:1–56, 2024.

[105] X. Zhang, S. Ding, C. Yu, J. Zhao, and D. Bu. Accurate and efficient phylogenetic inference through end-to-end deep learning. *Bioinformatics*, (biorxiv;2025.09.30.679045v1), Oct. 2025.

[106] Y. Zhang, D. Yang, and R. Liao. SymmetricDiffusers: Learning discrete diffusion on finite symmetric groups. *arXiv [cs.LG]*, Oct. 2024.

[107] M. Zhou, Z. Yan, E. Layne, N. Malkin, D. Zhang, M. Jain, M. Blanchette, and Y. Bengio. PhyloGFN: Phylogenetic inference with generative flow networks. *International Conference of Learning Representations*, 2024.

A Extended related works

Here we discuss the improvements our models make over two related works, ARTree and Phyloformer 2. This extends the discussion in Sec. 3.

A.1 Improvements upon ARTree

ARTree [96] is a scalable generative model over trees previously used for variational inference and learning representations. It generates trees in the **middle-out** paradigm.

To our knowledge, ARTree has always applied on a fixed set of N sequences a priori $X_1, \ldots, X_N$. Therefore it iteratively added them to the tree in order, that is, X_1 was always added first and X_N always added last. As well, since the sequences themselves $X_{1:N}$ were fixed during training, they never needed to be fed into a neural network.

In our situation we 1) have a **changing** set of sequences we have to use to output a posterior 2) have no canonical order $1, \ldots, N$. We make two changes to accommodate these differences.

1. **Conditioning on sequences:** Our network not only conditions on the state of the tree T_m, but also all of the sequences $X_{1:N}$. Conditioning on a further $> 10^5$ tokens required new architectures to prevent poor training behaviour as we show in App. F and Fig 4.
2. **Rao-Blackwellizing the addition order**: We could have assigned a canonical order to each set of sequences $X_{1:N}$ and added them to the tree in that order as in ARTree. However, this would make the training objective sparse, as there is only one valid addition at each generation step. In Fig. 3 we demonstrate through ablations that integrating over all possible additions makes a large difference in practice.

Our **middle-out** can be thought of as an ARTree model with (1) an architecture that conditions on sequences and (2) integrates over the addition order.

A.2 Improvements upon Phyloformer 2

Phyloformer 2 [12] is the previous state-of-the-art method for simulation-based inference for Bayesian phylogenetic inference. Our work on our **bottom-up** paradigm resembles Phyloformer 2 but improves upon it in several ways:

1. **Flexibility:** Phyloformer 2 uses a neural network backbone they call EvoPF that predicts embeddings for each species, followed by a module they call BayesNJ which parameterizes the distribution over topologies conditioned on the embeddings. BayesNJ uses a symmetric bilinear layer to convert the embeddings into a probability distribution over each merge, using the same embeddings for the full generation process. However, as noted in Section 3.2 of [12], in general the posterior merge probabilities at a given step might depend on the previous merges. We prove that there are simple distributions which cannot be approximated by Phyloformer 2 in Prop. A.2.1. Our models unlock this expressivity by using iterative forward passes through the network, each conditioning on the current state of the tree.
2. **Reliance on canonical merge order**: Another challenge faced by Phyloformer 2 is that it is intractable to sum over the likelihoods of all possible merge orders consistent with a given topology T_0. To solve this, they introduce a canonical merge order that depends on the branch lengths of the tree. As a result, the training objective is sparse, as there is only one valid merge at each generation step.

In App. H.3 we demonstrate through ablations that indeed our **bottom-up** model improves upon Phyloformer 2 by being more flexible and integrating over merge orders.

A.2.1 Proof of inflexibility of Phyloformer 2

We show that there are simple distributions over trees that cannot be represented by BayesNJ, the distribution family used by PF2. We can parameterize the BayesNJ distribution with a set of representations $v_1, \ldots, v_N$, a function that updates representations g, and a function that takes two representations and outputs a logit of their merger f. At each step, we calculate the logits of all possible merges $(f(v_i, v_j))_{1=i<j=N}$, samples from $i, j \sim \text{Categorical}(\text{softmax}(f(v_i, v_j))_{1=i<j=N})$, merge nodes i, j and create a new representation for the merged node $v_{(i,j)} = g(v_i, v_j)$. In practice $v_{1:N}$ are 128-dimensional vectors and g and f are bilinear transformations, but we do not need this fact.

The idea for our example is that dependency between the clades cannot be represented by BayesNJ since representations in one clade are not affected by merges in another. A more expressive merge function or representation-to-logit map cannot fix this. And dependencies can occur in realistic data when the hyperparameters in one clade influence those in another.

For 6 species A, B, C, D, E, F, call $T^1 = ((A, B), C), T^2 = ((A, C), B)$, $T'^1 = ((D, E), F), T'^2 = ((D, F), E)$, and use $*$ to represent the fusion of two trees, so $T^1 * T^2 = (((A, B), C), ((D, E), F))$.

Proposition 1. *Say we have a BayesNJ distribution p where $p(T) < \epsilon$ for any tree T where A, B, C are not separated from D, E, F. Then $|p(T^1 * T'^1)p(T^2 * T'^2) - p(T^1 * T'^2)p(T^2 * T'^1)| < K\epsilon$ for an absolute $K > 0$. Therefore BayesNJ cannot approach a distribution which puts mass on just trees $T^1 * T'^1$ and $T^2 * T'^2$ as the first term stays away from 0 and the second goes to 0 in the bound.*

Proof. BayesNJ constructs a tree in a series of merge steps, just like bottom up, so we get a distribution $p(T_0, T_1, ..., T_6)$. Let n_1 be the first step where a node in A, B, C merges with something else. Then if $\text{merge}(A, B, T_{n_1-1})$ is T_{n_1-1} with A and B merged,

$$p(T_{n_1} = \text{merge}(A, B, T_{n_1-1}) \mid T_{n_1-1}) = \frac{\exp(f(v_A, v_B))}{\sum_{i<j\in\{A,B,C\}} \exp(f(v_i, v_j))} + O(\epsilon),$$

with a similar equality for the merger of A and C. Note this doesn't depend on T_{n_1-1} beyond a factor of $O(\epsilon)$. Calling n_2 the the first step where a node in D, E, F merges we get a similar result. So,

$$\begin{aligned} p(T^1 * T'^1) =& p(T_{n_1} = \text{merge}(A, B, T_{n_1-1}), T_{n_2} = \text{merge}(D, E, T_{n_2-1})) + O(\epsilon) \\ =& \frac{\exp(f(v_A, v_B))}{\sum_{i<j\in\{A,B,C\}} \exp(f(v_i, v_j))} \times \frac{\exp(f(v_D, v_E))}{\sum_{i<j\in\{D,E,F\}} \exp(f(v_i, v_j))} + O(\epsilon). \end{aligned}$$

Similar equalities hold for $T^1 * T'^2$, $T^2 * T'^1$, and $T^2 * T'^2$. □

A.2.2 Correcting leakage bug in Phyloformer 2 code base

In Blassel et al. [12], Phyloformer 2 is shown to outperform IQ-Tree on both tractable and intractable models of evolution. We have identified that this result is tainted by test leakage in their pipeline.

Specifically, their EvoPF architecture is not invariant to the permutation of sequence labels. This is because $\text{flatten}(a_i \otimes a_j) \neq \text{flatten}(a_j \otimes a_i)$ in the outer product mean layer. To generate input datasets, they use AliSim, which outputs rows in a traversal order which contains information about the tree topology. While we make sure to permute sequence labels before passing them into any model, reflecting a real-world application where there is no known ordering, Blassel et al. [12] skip this step. As such, their model learns to leverage the ordering of the leaves in predicting the tree topology.

We loaded the Phyloformer 2 checkpoint from the latest code release and evaluated normalized RF on 100 greedily-sampled trees from the LG+GC test data at `https://zenodo.org/records/13742527/files/paper_test_sets.tar.xz?download=1` with $N = 50, L = 500$. Without shuffling the input sequences, we obtain a score of 0.046, close to the reported value of around 0.048 in Blassel et al. [12]. However, after shuffling the input sequences the normalized RF jumps from 0.046 to 0.238.

We then trained our models and Phyloformer 2 on $N = 50, L = 500$ trees generated from their LG+GC simulated data, but with shuffled sequence order. We trained all models for the same wall time, 28 hours on 2 A100 GPUs, and evaluated on 100 8-consensus trees. IQ-Tree LG+G8 scores a normalized RF of 0.064. Our models scores 0.069, 0.071, and 0.083 for **bottom-up**, **middle-out**, and **top-down** respectively. Our implementation of Phyloformer 2 scores slightly worse, at 0.087. Notably, after fixing the training pipeline to eliminate leakage, Phyloformer 2 does not outperform IQ-Tree which is nearly correctly specified.

B Other potential paradigms for generative models over trees

In our work, we consider three natural paradigms for generative models over trees: **bottom-up**, **middle-out**, and **top-down**. Here, we explore other possible paradigms that could be used to define generative models over trees or discuss the obstacles of using them for flexible simulation-based

inference. The purpose of this appendix is to discuss the breadth of our three paradigms in the context of the wider mathematical phylogenetics literature.

Table A1: Summary of alternative generative modeling approaches for trees and their limitations.

Approach	Status or challenge
Splits and edge contractions	analogous to **top-down**
Leaf delete-regraft	analogous to **middle-out**
Add one leaf at a time and quarter puzzling	analogous to **middle-out**
Coalescents	analogous to **bottom-up**
Dyck paths and permutations	models planar, unlabeled trees
Hyperbolic embeddings	models planar trees
NNI, SPR, and TBR moves	tractability of conditionals
Tree matchings	tractability of conditionals
BHV space	tractability of conditionals

B.1 Paradigms that recover our models

The following paradigms are practical for simulation-based inference, and can fit into one of the three paradigms which we consider.

Splits and edge contractions Many distributions of trees start with all species starting at the root, and then splitting left or right at a certain rate [3, 33]; these can include birth-death models. Indeed this was explored for building generative models in Larget [52] and Zhang and Matsen [103]. However, the number of possible splits of n species is 2^{n-1}; it is therefore a challenge to parameterize a distribution over splits in a flexible way. Indeed Larget [52] and Zhang and Matsen [103] had to restrict the possible splits to those seen in an MCMC run. Similarly, one could imagine corrupting trees by contracting random edges rather than just leaf edges; the de-corruption process would need to deal with the same challenge of parameterizing a distribution of splits of objects. One could parameterize such a distribution by gradually having nodes split one way or another using the diffusion method of Lüdke et al. [60]. This recovers the **top-down** paradigm we explore.

Leaf delete-regraft The "Aldous chain" involves tree corruptions via pruning and regrafting individual leaves [2]. Extensive theory on the process relies on the theory of random Dyck paths [59], which we discuss below. We are not sure if $p(T_m \mid T_0)$ is tractable, however we note that schedule-conditioning this model as described in Amin et al. [5] leads to something analogous to **middle-out**. In particular, Amin et al. [5] advocates improving discrete diffusion by keeping track of how many times each token in a sequence has been corrupted during diffusion. But once a leaf has been corrupted, it no longer contains any information about its original position. Therefore, the task of predicting $q_\theta(T_{m-1} \mid T_m, \vec{s}_M)$, where $\vec{s}_M$ is an indicator of how many times each leaf has been corrupted, simply involves taking the subtree of un-noised leaves and predicting which edge any noised leaf falls in that tree, recovering the exact prediction problem from **middle-out**.

Add one leaf at a time and quartet puzzling Sainudiin and Veber [74] and Ford [33] describe distributions of trees in which one leaf is added at a time; these differ by introducing different hyperparameters and latent variables, and possibly disallowing addition to edges not connected to a leaf. Integer vector encodings of trees describe the order in which individual leaves are attached [72]. Quartet puzzling is also an algorithm that adds one leaf at a time to a tree based on likelihoods of "quartet" splits [80]. These approaches are analogous to the **middle-out** paradigm we explore.

Coalescents One of the most popular models for for phylogenetic trees is the coalescent [10]. These involve joining subtrees at some rate, which is equivalent to the **bottom-up** paradigm we explore.

B.2 Paradigms that model the wrong types of trees

Alternatively, some paradigms are not obviously applicable for our problem since they model the wrong types of trees. We are interested in modeling *non-planar* trees (which do not distinguish between left and right branches) and *labeled* trees (which distinguish the leaves).

Dyck paths and permutations Every labeled planar tree – that is, a tree distinguishing left and right children – can be uniquely described by a Dyck path and a permutation of $\{1, \dots, N\}$, suggesting we could model trees over these two objects [2]. The coalescent point process [51] is another clever approach to efficiently generating trees; every labeled planar tree with branch lengths is characterized by a vector in $(0, 1)^{N-1}$ and a permutation. One issue with this approach is that it is not clear that modeling permutations or Dyck paths is simpler than modeling trees. Another issue is that we wish to model non-planar trees, and each non-planar tree admits 2^{N-1} planar representations. Stochastically choosing a planar representation for each tree would significantly increase variance when training, but canonically choosing a planar representation would introduce discontinuities or artificial structure.

Hyperbolic embeddings Sarkar [76] shows how to embed trees into hyperbolic space with arbitrarily small distortion. Putting the distortion aside, the embedding is not unique: rotations, translations, and the ordering of cones emanating from each internal node do not affect the tree distance matrix. In particular, the ordering of cones can be used to define a planarity of the tree, showing that the symmetries of this embedding cause problems at least as bad as those of the Dyck paths and permutations above.

B.3 Paradigms that do not admit tractable conditionals for discrete diffusion training

The following paradigms are challenging choices for simulation-based inference since they do not allow for efficient training with a discrete diffusion objective as explained in Sec. 5.1. Specifically, to train a diffusion model as in Campbell et al. [17], we need the conditionals $p(T(t) \mid T(0))$ to be tractable for any time t, or to train a diffusion model as in Amin et al. [5], we need $p(T_m \mid T_0)$ to be tractable for any number M, where T_m is the tree after m corruptions[2]. Many stochastic processes on trees are not suitable for a generative model because these conditionals are intractable.

NNI, SPR, and TBR moves Many MCMC methods explore tree-space with proposals based on nearest neighbor interchange (NNI), subtree prune-and-regraft (SPR), and tree bisection and reconnection (TBR) moves. However, calculating whether two trees T_1, T_2 have NNI, SPR, or TBR distance less than or equal to M is NP-Hard [20, 15, 4]. Therefore, if we corrupt our trees using these moves, simply determining if $p(T_m = T' \mid T_0 = T) > 0$ is NP-Hard, making it likely intractable. Meanwhile, other tree distances which are easier to calculate, like RF distance, are not associated with any particular sequence of tree operations to our knowledge.

Tree matchings Diaconis and Holmes [24] showed that trees are in bijection with pairings of the numbers $\{1, \dots, 2N-2\}$. They characterized the eigenspectrum of the action of the symmetric group on this representation, finding deep algebraic structure. However, the exact formulas for calculating $q_\theta(T_m \mid T_0)$, while "solved", are not easily computable in practice.

BHV space On the continuous Billera-Holmes-Vogtmann tree space [11], where topologies change via NNI moves, the remarkable theorem of Owen and Provan [67] demonstrates that geodesic distances are still computable in polynomial time. Nevertheless, recent work has highlighted the intractability of calculating the densities of diffusion processes on this space [95]; tractable distributions meanwhile (GGF of Woodman and Nye [95]) have restricted support. It is not clear if other continuous parameterizations, like that of Gavryushkin and Drummond [35], have more tractable diffusions.

C Expressivity of sampling paradigms

Here we prove simple modifications to the logits can allow all three paradigms to generate trees from PDA to Yule, the spectrum of realistic trees. Note this section is not about proving that our models can represent every sequence distribution (that fact follows from the discussion below Eqn. 2). Rather it is about seeing how each of our paradigms fit simple distributions of trees explicitly.

C.1 Bottom-up

Consider a process that generates a tree T_0 by starting from T_{N-1}, a forest of N singleton components, and building up to a complete tree. At each step, the process merges two components of the forest T_m rooted at v and u leaves to produce T_{m-1}, selecting this pair with probability proportional to $\gamma(n_{\text{leaves}}(v) + n_{\text{leaves}}(u) - 2) + 1$, where $\#\text{leaves}(u)$ is the number of leaves in the subtree rooted

[2] SCUD [5] models can technically be trained with only empirical samples from $p(T_m \mid T_{m-1}, T_0)$. However this training method sacrifices a closed-form ELBO and has extremely high variance in its gradient estimates [8].

at u, for some parameter $\gamma \geq 0$. Each merge creates one internal node of the final tree T_0; the first merge creates the deepest internal node and the last creates the root.

Reversing the merge order, we obtain an ordering $v_1, v_2, \ldots, v_{N-1}$ of the internal nodes of T_0 in which every ancestor appears before its descendants — that is, a linear extension of $\texttt{int}(T_0)$. It will be useful below that paths are uniform over linear extensions.

Proposition 2. *When $\gamma = 0$, $p(T_0)$ is a Yule distribution over trees; when $\gamma \to \infty$, $p(T_0)$ concentrates on combs. When $\gamma = 1$, $p(T_0) = 1/(2N-3)!!$ is the PDA (uniform) distribution over topologies, and each path is equally likely conditioned on the topology: $p(T_1, \ldots, T_{N-1} \mid T_0) = 1/|\mathrm{LE}(\texttt{int}(T_0))|$.*

Proof. The $\gamma = 0$ case is Kingman's coalescent, which is well known to produce the Yule distribution [3]; the $\gamma \to \infty$ case follows since the merge probability concentrates on the pair with largest $a + b$, which always includes the component containing all previously merged taxa.

For $\gamma = 1$, at step m, the forest T_m has $m + 1$ components with leaf counts $n_1, \ldots, n_{m+1}$. The normalizing constant is

$$\sum_{i<j}(n_i + n_j - 1) = ((m+1) - 1)\sum_i n_i - \binom{m+1}{2} = \frac{1}{2}m(2N - (m+1)),$$

since each n_i appears in $(m + 1) - 1$ pairs. Taking the product from $m = 1$ to $m = N - 1$, we get

$$\prod_{m=1}^{N-1} \frac{1}{2}m(2N - (m+1)) = 2^{-(N-1)}(N-1)!\frac{(2N-2)!}{(N-1)!} = (N-1)!(2N-3)!!,$$

with the last equality coming by counting even and odd terms of $(2N - 2)!$. We also note that for $v \in \texttt{int}(T_0)$, $n_{\text{leaves}}(\texttt{left}(v_m)) + n_{\text{leaves}}(\texttt{right}(v_m))$ is the number of descendants v has in $\texttt{int}(T_0)$ plus two: $|\texttt{descendants}_{\texttt{int}(T_0)}(v)| + 2$. Finally,

$$\begin{aligned} p(T_0, T_1, \ldots, T_{N-1}) &= \prod_{m=1}^{N-1} \frac{n_{\text{leaves}}(\texttt{left}(v_m)) + n_{\text{leaves}}(\texttt{right}(v_m)) - 1}{\frac{1}{2}m(2N-(m+1))} \\ &= \frac{\prod_{v \in \texttt{int}(T_0)}(|\texttt{descendants}_{\texttt{int}(T_0)}(v)| + 1)}{(N-1)!(2N-3)!!} \end{aligned}$$

which does not depend on the order of the merges. This then equals $1/(|\mathrm{LE}(\texttt{int}(T_0))|(2N-3)!!)$ by the hook length formula $|\mathrm{LE}(\texttt{int}(T_0))| = |\texttt{int}(T_0)|!/\prod_{v \in \texttt{int}(T_0)}(|\texttt{descendants}_{\texttt{int}(T_0)}(v)| + 1)$. Summing over all $|\mathrm{LE}(\texttt{int}(T_0))|$ linear extensions gives $p(T_0) = 1/(2N-3)!!$. □

C.2 Middle-out

Consider a process that generates a rooted tree by starting with a tree with a single pseudo taxon and grafting one taxon at a time. At step k, starting at $k = 0$, the current tree has k leaves and $\max(2k - 1, 1)$ edges. The new taxon subdivides an edge chosen with probability proportional to α or $1 - \alpha$, where *pendant* edges (those incident to a leaf, not including the pseudo-taxon) receive weight $1 - \alpha$ and all other edges receive weight α, for a parameter $\alpha \in [0, 1)$.

Proposition 3. *When $\alpha = 0$, $p(T_0)$ is a Yule distribution; when $\alpha = \frac{1}{2}$, $p(T_0)$ is PDA; and when $\alpha \to 1$, $p(T_0)$ concentrates on combs. When $\alpha = 1/2$, the distribution does not depend on the order in which taxa are inserted.*

Proof. This is Ford's alpha model [33]. To correspondence to Yule, PDA, and combs is given in [33, Sec 3.12]. Order-independence when $\alpha = 1/2$ follows from the uniform probability of adding to any edge:

$$p(T_{0:N}) = \frac{1}{N!}1 \times 1 \times 3^{-1} \times 5^{-1} \times \cdots \times (2N-3)^{-1} = \frac{1}{N!}p_{\text{PDA}}(T).$$

. □

C.3 Top-down

The Aldous beta model [1] generates a labeled rooted tree by recursively splitting sets of taxa. At a node containing $n \geq 2$ taxa $v_1, \ldots, v_n$, the set is partitioned into two non-empty groups; the probability of each labeled partition $\{A, B\}$ is proportional to $B(|A| + \beta + 1,\ |B| + \beta + 1)$ for $\beta > -2$, where $B(\cdot, \cdot)$ is the beta function. Each group is then split recursively until every group is a singleton. When $\beta = 0$ this gives the Yule distribution; when $\beta = -\frac{3}{2}$ it gives PDA; and as $\beta \to \infty$ it gives the "almost completely balanced" distribution [3].

In our top-down paradigm, we do not split all taxa at a node simultaneously. Instead, taxa percolate down one at a time and all taxa at a given depth split into groups before any taxon proceeds to the next depth. We imagine doing this by picking a random taxa and "revealing" how it split under the Aldous beta model. We can reveal the direction of one taxa at a time by looking at the marginal likelihood: if we've revealed that a taxa, call them $v_1, \ldots, v_a$, have percolated into one group and b taxa, call them $v_{n-b+1}, \ldots, v_n$, have percolated into the other, what is the posterior probability that the next taxa joins the group with a taxa? We call this probability with $M = n - a - b$ taxa remaining

$$f(a, b \mid M) = \frac{p(v_{1:a+1} \in A, v_{n-b+1:n} \in B)}{p(v_{1:a} \in A, v_{n-b+1:n} \in B)}.$$

Note the order of A and B doesn't matter as $f(a, b \mid M) = 1 - f(b, a \mid M)$, and, when $a = b = 0$, there is no distinction between A and B. The procedure is summarized in Alg. 1.

Algorithm 1 Single-step Aldous β

Initialize T with a single root with $\texttt{leaves}(\texttt{root}) = \{v_1, \ldots, v_N\}$
$d \leftarrow 0$
while some node at depth d has ≥ 2 taxa **do**
  $U_d \leftarrow \{u \in T : d_T(u) = d,\ |\texttt{leaves}(u)| \geq 2\}$
  while $\texttt{leaves}(u) \neq \emptyset$ for some $u \in U_d$ **do**
    $u \sim \text{Categorical}\left(|\texttt{leaves}(u)| / \sum_{u' \in U_d} |\texttt{leaves}(u')|\right)$
    $v \sim \text{Uniform}(\texttt{leaves}(u))$
    if u has no children **then**
      create children w, w' of u; $\quad \texttt{child} \leftarrow w$
    else
      $w, w' \leftarrow$ children of u; $\quad a, b \leftarrow |\texttt{leaves}(w)|, |\texttt{leaves}(w')|$
      $M \leftarrow |\texttt{leaves}(u)|$
      $\texttt{child} \leftarrow w$ with probability $f(a, b \mid M)$ else $\texttt{child} \leftarrow w'$
    end if
    $\texttt{leaves}(\texttt{child}) \leftarrow \texttt{leaves}(\texttt{child}) \cup \{v\}$
    $\texttt{leaves}(u) \leftarrow \texttt{leaves}(u) \setminus \{v\}$
  end while
  $d \leftarrow d + 1$
end while
return T

To implement Alg. 1 we only need to know the values of f.

Proposition 4. *The probabilities* (7) *equal*

$$f(a, b \mid M) = \begin{cases} \dfrac{a + \beta + 1}{a + b + 2\beta + 2}, & a, b > 0, \\ \dfrac{B(a + \beta + 2, \beta + 1) - B(n + \beta + 1, \beta + 1)}{B(a + \beta + 1, \beta + 1) - B(n + \beta + 1, \beta + 1)}, & b = 0,\ M > 1, \\ 0, & b = 0,\ M = 1, \end{cases} \tag{4}$$

with the $a = 0$ cases given by symmetry ($f(0, b) = 1 - f(b, 0)$). Every percolation order of the taxa is equally likely conditioned on T_0.

Proof. For a node with n taxa where a have percolated one direction and b the other, with $M = n - a - b$ remaining, define the unnormalized marginal likelihood

$$p(v_{1:a} \in A, v_{n-b+1:n} \in B) \propto S(a,b,M) := \sum_{\substack{\{A,B\}:|A|\geq 1,|B|\geq 1 \\ v_{1:a}\in A, v_{n-b+1:n}\in B}} B(|A|+\beta+1, |B|+\beta+1). \quad (5)$$

Since $|A| = a + j$ and $|B| = b + M - j$ for j of the remaining taxa joining A, we can write

$$S(a,b,M) = \sum_j \binom{M}{j} B(a+j+\beta+1,\ b+M-j+\beta+1), \quad (6)$$

where the sum runs over all j for which $a + j \geq 1$ and $b + M - j \geq 1$. The conditional probability that the next taxon goes to group with a taxa is therefore

$$f(a,b \mid M) = \frac{S(a+1,b,M-1)}{S(a,b,M)}. \quad (7)$$

We evaluate $S(a,b,M)$ in closed form using the integral representation $B(s,t) = \int_0^1 x^{s-1}(1-x)^{t-1}dx$ and the binomial theorem.

Case with $a, b > 0$. When $a \geq 1$ and $b \geq 1$ the constraints $a + j \geq 1$ and $b + M - j \geq 1$ hold for all $j \in \{0, \ldots, M\}$, so from (6)

$$S(a,b,M) = \int_0^1 x^{a+\beta}(1-x)^{b+\beta} \underbrace{\sum_{j=0}^{M} \binom{M}{j} x^j (1-x)^{M-j}}_{=1} dx = B(a+\beta+1, b+\beta+1).$$

Therefore $S(a+1,b,M-1) = B(a+\beta+2, b+\beta+1)$ and

$$f(a,b) = \frac{B(a+\beta+2,b+\beta+1)}{B(a+\beta+1,b+\beta+1)} = \frac{a+\beta+1}{a+b+2\beta+2}.$$

Case with $b = 0$, $M \geq 1$. The constraint $b+M-j \geq 1$ forces $j \leq M-1$, excluding the degenerate all-left split. Writing $n = a + M$,

$$S(a,0,M) = \int_0^1 x^{a+\beta}(1-x)^{\beta} \left[\underbrace{\sum_{j=0}^{M} \binom{M}{j} x^j (1-x)^{M-j}}_{=1} - x^M \right] dx$$
$$= B(a+\beta+1, \beta+1) - B(n+\beta+1, \beta+1).$$

Similarly $S(a+1,0,M-1) = B(a+\beta+2,\beta+1) - B(n+\beta+1,\beta+1)$. Dividing gives the $b = 0$ case of (4). Note when $M = 1$ we have $a + 1 = n$ so $S(a+1,0,0) = B(n+\beta+1,\beta+1) - B(n+\beta+1,\beta+1) = 0$.

Order independence. In the notation of Alg. 1, calling $M_d = \sum_{u\in U_d} n_{\texttt{leaves}}(u)$, and a_u, b_u the number of leaves in u's children for every node u, it is easy to see that, by construction, the probability of the path is the probabilities of the percolation orderings times the probabilities from Aldous' beta model:

$$p(T_{0:M}) = \prod_{d=0}^{d_{\max}} \frac{1}{M_d!} \prod_{u\in U_d} \frac{B(a_u+\beta+1, b_u+\beta+1)}{B(\beta+1,\beta+1) - 2B(a_u+b_u+\beta+1,\beta+1)} = \left(\prod_{d=0}^{d_{\max}} \frac{1}{M_d!} \right) \times p(T_0).$$

The first term $\prod_{d=0}^{d_{\max}-1} \frac{1}{M_d!}$ is $p(T_{1:M} \mid T_0)$ which is therefore independent of the order of the percolation.

□

D Discrete diffusion models

D.1 Background

Discrete diffusion models have been recently highlighted for flexibly fitting complex distributions of discrete objects. However previous diffusion models have largely been built to model data of the form $\text{tokens}^{\text{length}}$ like language [6, 17, 78]. To apply these models to graphs therefore, one could simply model their adjacency matrices $A \in \{0,1\}^{N\times N}$ [88, 14]. Modeling phylogenetic trees as abstract graphs is problematic however as (1) internal nodes of the tree are latent, so there is no ground truth adjacency matrix A, and (2) trees are highly constrained structures, and there is no clear way to incorporate these constraints during generation. Our models avoid these issues by building a diffusion model directly on the space of trees.

Discrete diffusion models have also succeeded at modeling other sorts of discrete objects, like permutation processes [106], and sequences of different lengths [38, 8]. However, these methods do not map to modeling trees.

D.2 Derivation of discrete diffusion ELBO

$$q_\theta(T_0 \mid X_{1:N}) = \sum_{T_1,T_2,\ldots,T_M} \prod_{m=1}^{M} q_\theta(T_{m-1} \mid T_m, X_{1:N})$$

Here, we provide the derivation of the discrete diffusion ELBO in Eqn. 2. We start by taking the logarithm of the expression for the likelihood in Eqn. 1. Then, we multiply and divide each term by $p\left(T_{1:M} \mid T_0\right)$, yielding

$$\begin{aligned}
\log q_\theta(T_0 \mid X_{1:N}) &= \log \sum_{T_{1:M}} p\left(T_{1:M} \mid T_0\right) \frac{\prod_{m=1}^{M} q_\theta(T_{m-1} \mid T_m, X_{1:N})}{p\left(T_{1:M} \mid T_0\right)} \\
&= \log \mathbb{E}_{T_{1:M}\sim p(T_{1:M}\mid T_0)} \left[\prod_{m=1}^{M} \frac{q_\theta(T_{m-1} \mid T_m, X_{1:N})}{p(T_{m-1} \mid T_m, T_0)}\right] \\
&\geq \mathbb{E}_{T_{1:M}\sim p(T_{1:M}\mid T_0)} \left[\sum_{m=1}^{M} \log \frac{q_\theta(T_{m-1} \mid T_m, X_{1:N})}{p(T_{m-1} \mid T_m, T_0)}\right] \\
&= \sum_{m=1}^{M} \mathbb{E}_{T_{m-1},T_m\sim p(T_{m-1},T_m\mid T_0)} \left[\log \frac{q_\theta(T_{m-1} \mid T_m, X_{1:N})}{p(T_{m-1} \mid T_m, T_0)}\right] \\
&= -\sum_{m=1}^{M} \mathbb{E}_{T_m\sim p(T_m\mid T_0)} \left[D_{\text{KL}}(p(T_{m-1} \mid T_m, T_0) \,||\, q_\theta(T_{m-1} \mid T_m, X_{1:N}))\right] \\
&= -M\mathbb{E}_{m\sim\text{Unif}(1,\ldots,M),T_m\sim p(T_m\mid T_0)} \left[D_{\text{KL}}(p(T_{m-1} \mid T_m, T_0) \,||\, q_\theta(T_{m-1} \mid T_m, X_{1:N})\right].
\end{aligned}$$

We used the fact that $p(T_M \mid T_0)$ and $q(T_M \mid X_{1:N})$ are both assumed to have all their mass on a unique starting position; for **top-down** this has all the taxa at the root, for **bottom-up** this has all taxa disconnected, and for **middle-out** the tree has only the pseudo-taxon.

E Additional information on paradigms

For each paradigm we describe the noising schedule, how to calculate the target $p(T_{m-1} \mid T_m, T_0)$, and how to sample from our mutual-information informed noising schedule.

E.1 Noise schedule details

The ELBO in Eqn. 2 requires computing $p(T_{m-1} \mid T_m, T_0)$, the distribution over the previous state given the current noised state and the ground truth. We achieve this by introducing a noise schedule $\pi(T_m \mid T_0)$ and rewriting the ELBO as $\mathbb{E}_{p(T_0,X_{1:N})} \log q_\theta(T_0 \mid X_{1:N}) \geq$

$$-\mathbb{E}_{T_m\sim\pi(\cdot\mid T_0)} \left[w(T_m) \cdot D_{\text{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, q_\theta(T_{m-1} \mid T_m, X_{1:N}))\right] \tag{8}$$

where

$$w(T_m) = \frac{M \cdot p(T_m \mid T_0)}{\pi(T_m \mid T_0)}. \tag{9}$$

A classic heuristic for designing discrete diffusion noise schedules is to choose π to minimize the variance of this Monte Carlo estimator, assuming q_θ has converged to the true reverse process [49, 6]. To make this analytically tractable, we further assume a simple reference posterior $p(T_0 \mid X_{1:N})$. A natural choice is any of the prior distributions described in Sec. 4.2 whose reverse process we can compute in closed form; we use $p_{\mathrm{PDA}}(T_0)$ as a concrete example throughout, though any such prior works equally well.

Under this assumption, the variance-minimizing π sets $w(T_m) \cdot D_{\mathrm{KL}}(\cdots)$ to a constant across (m, T_m), giving

$$\pi(T_m \mid T_0) \propto M \cdot p(T_m \mid T_0) \cdot D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, p_{\mathrm{PDA}}(T_{m-1} \mid T_m)), \tag{10}$$

where $p_{\mathrm{PDA}}(T_{m-1} \mid T_m) = \mathbb{E}_{T_0 \sim p_{\mathrm{PDA}}(T_0)} p(T_{m-1} \mid T_m, T_0)$. The normalizing constant of π is

$$\sum_{m=1}^{M} \sum_{T_m} M \cdot p(T_m \mid T_0, m) \cdot D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, p_{\mathrm{PDA}}(T_{m-1} \mid T_m)) = -M \log p_{\mathrm{PDA}}(T_0), \tag{11}$$

which is $-M \log(2N-3)!!$. Under this choice,

$$w(T_m) = \frac{\log p_{\mathrm{PDA}}(T_0)}{D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, p_{\mathrm{PDA}}(T_{m-1} \mid T_m))}.$$

Marginalizing over $T_0 \sim p_{\mathrm{PDA}}$, the quantity $p(T_m \mid T_0) \cdot D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, p_{\mathrm{PDA}}(T_{m-1} \mid T_m))$ becomes the mutual information $\mathrm{MI}(T_{m-1}; T_0 \mid T_m)$: the amount of information about T_0 that is lost at step m. This mirrors the mutual information schedules of Austin et al. [6] and concentrates training on steps where the model must add the most information about T_0 in a single step.

To implement this schedules, we need two things:

- The form of $p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ to calculate $w(T_m)$.
- A procedure to sample from $\pi(T_m \mid T_0)$.

Sampling will be handled below but note App. C gives us the first goal: we've already built procedures q to sample sequences of trees such that $\prod_{m=0}^{M-1} q(T_{m-1} \mid T_m) = p_{\mathrm{PDA}}(T_0) \times p_{\mathrm{noising}}(T_{1:M} \mid T_0)$. Therefore, these $p(T_{m-1} \mid T_m)$ must be $p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ as we will see below.

E.2 Bottom-up

E.2.1 Distribution over paths

It will be convenient to derive the distribution over paths $p(T_1, \ldots, T_{N-1} \mid T_0)$. The path $(T_m)_m$ is characterized by an order of internal vertices which are removed from T_0, $v_1, \ldots, v_{N-1}$; on the other hand every ordering of the internal vertices of T_0 $v_1, \ldots, v_{N-1}$ corresponds to a unique path as long as the ancestors of every node appear before it in the ordering. An ordering of the internal vertices that satisfies this rule is known in the literature as a *linear extension* of the subtree of T_0 made of its internal nodes, $\texttt{int}(T_0)$. We call $\mathrm{LE}(T)$ the set of linear extensions of a tree T. We now show our choice of noising process makes *every linear extension equally likely*; this is the property that will allow us to derive the quantities we need below.

Proposition 5. *The probability of a path $(T_m)_m$ that removes vertices in order $v_1, \ldots, v_{N-1}$ is*

$$p(T_1, \ldots, T_{N-1} \mid T_0) = 1/|\mathrm{LE}(\texttt{int}(T_0))|.$$

Proof. Recall for a root of a subtree v, $p(\texttt{remove } v \mid T_m) \propto |\texttt{descendants}(v)|$, excluding v as a descendant. The denominator is the sum of the descendants of all roots, so it's simply the number of

nodes in the tree, minus the roots: $2N - 2(m+1)$. Finally,

$$\begin{aligned} p(T_1, \ldots, T_{N-1} \mid T_0) &= \prod_{m=0}^{N-2} p(\texttt{remove}\ v \mid T_m) \\ &= \prod_{m=0}^{N-2} \frac{|\texttt{descendants}(v_{m+1})|}{2N - 2(m+1)} \\ &= \frac{\prod_{v \in \texttt{int}(T_0)} |\texttt{descendants}(v)|}{2^{N-1}(N-1)!} \end{aligned}$$

which does not depend on the order of the vertices. One can verify this also matches the classical hook length formula by noting $|\texttt{descendants}(v)|$ is twice the number of descendants in $\texttt{int}(T_0)$ plus 1, $|\texttt{descendants}_{\texttt{int}(T_0)}(v)| + 1$, and that $|\mathrm{LE}(\texttt{int}(T_0))| = |\texttt{int}(T_0)|! / \prod_{v \in \texttt{int}(T_0)} (|\texttt{descendants}_{\texttt{int}(T_0)}(v)| + 1)$. □

E.2.2 Calculating the target

We can now derive $p(T_{m-1} \mid T_m, T_0)$. The set of removed nodes $\{v_1, \ldots, v_m\}$ forms a rooted subtree of $\texttt{int}(T_0)$ containing the root; call it R_m. Its leaves are exactly the candidates for the last node removed. Since all linear extensions of $\texttt{int}(T_0)$ are equally likely, the probability that node v was removed last is simply the fraction of linear extensions of R_m that end in v: $p(T_{m-1} = \texttt{add}(T_m, v) \mid T_m, T_0) = \frac{|\mathrm{LE}(\texttt{remove}(R_m, v)|}{|\mathrm{LE}(R_m)|}$.

Proposition 6. *For a leaf v of R_m,*

$$p(T_{m-1} = \texttt{add}(T_m, v) \mid T_m, T_0) = \frac{1}{m} \prod_{u \in \texttt{ancestors}(v, R_m)} \frac{|\texttt{descendants}_{R_m}(u)| + 1}{|\texttt{descendants}_{R_m}(u)|}, \tag{12}$$

where $\texttt{descendants}_{R_m}(u)$ counts the descendants of u in R_m, not including u itself.

Proof. By the hook length formula for rooted trees, $|\mathrm{LE}(R_m)| = m! / \prod_{u \in R_m} (|\texttt{descendants}_{R_m}(u)| + 1)$. Removing a leaf v from R_m decreases the numerator from $m!$ to $(m-1)!$, removes the factor $|\texttt{descendants}_{R_m}(v)| + 1 = 1$ from the denominator, and decrements $|\texttt{descendants}_{R_m}(u)|$ by 1 for every strict ancestor u of v. Taking the ratio gives Eqn. 12. □

This can be calculated with a post-order traversal of R_m to compute $|\texttt{descendants}_{R_m}(v)|$ for all v, then a pre-order pass to accumulate the product over ancestors. The cost of computing $p(T_{m-1} \mid T_m, T_0)$ for every possible T_{m-1} is therefore $O(m)$.

E.2.3 Noise schedule

The PDA backwards steps By Prop. 2, the PDA coalescent ($\gamma = 1$) also produces uniform linear extensions conditioned on T_0. Therefore the reverse model $p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ merges subtree roots u, w in the forest T_m with probability proportional to $n_{\mathrm{leaves}}(u) + n_{\mathrm{leaves}}(w) - 1$.

Sampling from π We write

$$\begin{aligned} \pi(T_m \mid T_0) \propto & p(T_m \mid T_0) D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \| p_{\mathrm{PDA}}(T_{m-1} \mid T_m)) \\ &= \sum_{\tilde{T}_{m-1}} p(T_m, \tilde{T}_{m-1} \mid T_0) \log \frac{p(\tilde{T}_{m-1} \mid T_m, T_0)}{p_{\mathrm{PDA}}(\tilde{T}_{m-1} \mid T_m)} \\ &= \sum_{\tilde{T}_{1:M} \mid \tilde{T}_m = T_m} p(\tilde{T}_{1:M} \mid T_0) \log \frac{p(\tilde{T}_{m-1} \mid \tilde{T}_m, T_0)}{p_{\mathrm{PDA}}(\tilde{T}_{m-1} \mid \tilde{T}_m)} \\ &=: \sum_{\tilde{T}_{1:M} \mid \tilde{T}_m = T_m} p(\tilde{T}_{1:M} \mid T_0) \tilde{q}(m \mid \tilde{T}_{0:M}). \end{aligned}$$

It turns out that $\tilde{q}(m \mid \tilde{T}_{0:M})$ is an unnormalized distribution with a fixed normalizing constant as

$$\sum_{m=1}^{M} \log \frac{p(\tilde{T}_{m-1} \mid \tilde{T}_m, T_0)}{p_{\mathrm{PDA}}(\tilde{T}_{m-1} \mid \tilde{T}_m)} = \log \frac{p(\tilde{T}_{1:M} \mid T_0)}{p_{\mathrm{PDA}}(T_0, \tilde{T}_{1:M})} = -\log p_{\mathrm{PDA}}(T_0).$$

Therefore, calling $q(m \mid \tilde{T}_{1:M}) = \tilde{q}(m \mid \tilde{T}_{1:M}) / \log(2N-3)!!$ we can simply sample from $\pi(T_m \mid T_0)$ by simple sampling a noising path $T_{1:M} \sim p(\tilde{T}_{1:M} \mid T_0)$ and selecting $m \sim q(m \mid T_{1:M})$.

The wrinkle is that $q(m \mid T_{1:M})$ need not be positive! The solution is to instead rejection sample from the proposal distribution

$$\hat{\pi}(T_m \mid T_0) \propto p(T_m \mid T_0) \mathbb{E}_{p(T_{m-1} \mid T_m, T_0)}[-\log p_{\mathrm{PDA}}(T_{m-1} \mid T_m)].$$

With a bit of algebra, one can show that $\tilde{\pi}(T_m \mid T_0)$ has normalizing constant $-\mathbb{E}_{p(T_{1:M} \mid T_0)} \log p_{\mathrm{PDA}}(T_{1:M}, T_0) = \log(2N-3)!! + \log |\mathrm{LE}(\texttt{int}(T_0))|$ and, as above,

$$\tilde{\pi}(T_m \mid T_0) = \sum_{\tilde{T}_{1:M} \mid \tilde{T}_m = T_m} p(\tilde{T}_{1:M} \mid T_0) \hat{q}(m \mid \tilde{T}_{0:M}),$$

where $\hat{q}(m \mid T_{0:M}) = -\log p_{\mathrm{PDA}}(T_{m-1} \mid T_m) / (\log(2N-3)!! + \log |\mathrm{LE}(\texttt{int}(T_0))|)$. This gives Alg. 2.

Algorithm 2 Importance sampling the **bottom-up** noise schedule

Require: Tree T_0 with N leaves
  repeat
    Draw a noising path $T_{1:M} \sim p(T_{1:M} \mid T_0)$
    Draw $m \in \{1, \ldots, N-1\}$ with probability $q(m \mid T_{0:M}) \propto -\log p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$
    Compute acceptance ratio $\rho \leftarrow \frac{D_{\mathrm{KL}}(p(T_{m-1} \mid T_m, T_0) \,\|\, q_{\mathrm{PDA}}(T_{m-1} \mid T_m))}{\mathbb{E}_{p(T_{m-1} \mid T_m, T_0)}[-\log q_{\mathrm{PDA}}(T_{m-1} \mid T_m)]}$
    Accept with probability ρ
  until accepted
  return T_m

Is this an efficient way to sample? The average acceptance probability is $\mathbb{E}_{\hat{\pi}} \rho = \frac{\log(2N-3)!!}{\log(2N-3)!! + \log |\mathrm{LE}(\texttt{int}(T_0))|} \geq \frac{\log(2N-3)!!}{\log(2N-3)!! + \log(N-1)!} \geq 1/2$. This lower bound shows we shouldn't expect to reject, on average, more than once per sample.

E.3 Middle-out

E.3.1 Distribution over paths

The noising process removes one taxon at each step, selecting uniformly among the $N - m$ taxa present in T_m. A path from T_0 to T_N is a sequence of N distinct taxa removed at each step. The fully noised state T_N is a tree containing only the pseudo-taxon. Clearly we have

Proposition 7. *Every removal sequence has the same probability:* $p(T_{1:N} \mid T_0) = 1/N!$.

E.3.2 Calculating the target

At state T_m, the tree contains $N - m$ taxa and the pseudo-taxon. To reverse one noising step, we reattach one of the m pruned taxa to the appropriate edge of T_m. For each pruned taxon there is exactly one edge in T_m such that grafting produces a tree consistent with T_0.

Proposition 8. *The reverse distribution is uniform over the m valid re-attachments:*

$$p(T_{m-1} \mid T_m, T_0) = \frac{1}{m}.$$

Proof. Follows from Prop. 7. □

E.3.3 Noise schedule

The PDA backwards steps By Prop. 3, the PDA Ford alpha model ($\alpha = 1/2$) produces uniform paths $T_{1:M}$ conditioned on T_0. Therefore the reverse model $p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ picks a taxon to add and branch in the tree uniformly at random:

$$p_{\mathrm{PDA}}(T_{m-1} \mid T_m) = \frac{1}{m\,(2(N-m)-1)}.$$

Sampling from π As for **bottom-up** we can write

$$\pi(T_m \mid T_0) = \sum_{\tilde{T}_{1:M} \mid \tilde{T}_m = T_m} p(\tilde{T}_{1:M} \mid T_0) q(m \mid T_0)$$

where $q(m \mid T_0) = \frac{\log \frac{p(\tilde{T}_{m-1} \mid \tilde{T}_m, T_0)}{p_{\mathrm{PDA}}(\tilde{T}_{m-1} \mid \tilde{T}_m)}}{\log(2N-3)!!}$. In this case, for $m = 1, \ldots, N-1$, $q(m \mid T_0) = \frac{\log(2(N-m)-1)}{\log(2N-3)!!}$ is positive, so it's a probability distribution. Therefore we can sample from π by sampling a noising path and selecting a step according to q. This gives Alg. 3.

Algorithm 3 Importance sampling the **middle-out** noise schedule

Require: Tree T_0 with N leaves
  Draw a noising path $T_{1:M} \sim p(T_{1:M} \mid T_0)$
  Draw $m \in \{1, \ldots, N-1\}$ with probability $q(m \mid T_{0:M}) \propto \log(2(N-m)-1)$.
  return T_m

E.4 Top-down

E.4.1 Distribution over paths

The noising process percolates taxa v upward from depth $d_{T_0}(v)$ to the root, one level at a time. Since only taxa at maximal depth are eligible, the process decomposes into *rounds*: round d (for $d = d_{\max}, d_{\max}-1, \ldots, 1$, where $d_{\max}$ is the maximum leaf depth in T_0) moves every taxon currently at depth d up to depth $d-1$.

Let $V_d = \{u \in \mathtt{int}(T_0) : d_{T_0}(u) \geq d\}$ be the taxa that are percolated upwards in round d. At the start of round d, all of these taxa are at depth d: those originally at depth exactly d have remained there, while those originally deeper have been brought to depth d by earlier rounds. Within round d, the taxa are moved from depth d to $d-1$ one at a time, each selected uniformly from the taxa still at depth d.

Proposition 9. *Every sequence $T_{1:M}$ has the same probability: $p(T_{1:M} \mid T_0) = 1/\prod_{d=1}^{d_{\max}} |V_d|!$.*

Proof. Within round d, each of the orderings of the taxa in V_d is equally likely. □

Note $|V_d|$ is the same as M_{d-1} from Prop. 4.

E.4.2 Calculating the target

At state T_m, some rounds have been completed and one is partially processed. Let d be the depth of the current round (the maximal depth of any taxon in T_m). Let $M_m = |\{v : d_{T_m}(v) = d\text{–}1 \text{ and } d_{T_0}(v) \geq d\}|$ be the number of taxa that have percolated up in T_m in round d.

Proposition 10. *The reverse distribution is uniform over M_m valid downward moves:*

$$p(T_{m-1} \mid T_m, T_0) = \frac{1}{M_m}.$$

Proof. Follows from Prop. 9. □

E.4.3 Noise schedule

The PDA backwards steps By Prop. 4, the PDA single-taxa splitting model ($\beta = -3/2$) also produces uniform paths $T_{1:M}$ conditioned on T_0. Therefore the reverse model $p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ picks a taxon at the current depth at random and percolates down according to f, following the procedure described in Alg. 1.

Sampling from π Say T_m is on round d with a_u, b_u nodes percolated for every node $u \in U_d$, and $\bar{a}_u$, $\bar{b}_u$ nodes not yet percolated which belong to the group with a_u and b_u taxa in T_0. Call $M_u = \bar{a}_u + \bar{b}_u$ the number of leaves at u in T_m, $|\texttt{leaves}(u)|$, $N_u = \bar{a}_u + \bar{b}_u + a_u + b_u$ the number of leaves in the subtree rooted at u in T_0, $n_{\mathrm{leaves}}(u)$, and $A_u = a_u + \bar{a}_u$, $B_u = b_u + \bar{b}_u$.

At every $u \in \texttt{int}(T_0)$, $p_{\mathrm{PDA}}(A_u, B_u) := \frac{B(A_u+\beta+1, B_u+\beta+1)}{B(\beta+1,\beta+1) - 2B(N_u+\beta+1,\beta+1)}$ is, intuitively, the amount of information created at node u. In particular note that $\prod_{u\in\texttt{int}(T_0)} p_{\mathrm{PDA}}(A_u, B_u) = p_{\mathrm{PDA}}(T_0) = 1/(2N-3)!!$. First we sample a u according to how much information is created at that node. Define, for $u \in \texttt{int}(T_0)$,

$$\pi_u(T_m \mid T_0) = \frac{1}{-\log p_{\mathrm{PDA}}(A_u, B_u)} \sum_{T_{m-1} \text{ decreases } M_u} p(T_m, T_{m-1} \mid T_0) \log \frac{p(T_m \mid T_{m-1}, T_0)}{p_{\mathrm{PDA}}(T_m \mid T_{m-1})}.$$

Then clearly $\pi(T_m \mid T_0) = \sum_{u\in\texttt{int}(T_0)} \frac{-\log p_{\mathrm{PDA}}(A_u,B_u)}{\log(2N-3)!!} \pi_u(T_m \mid T_0)$. Now say $u \in U_d$, and $m_1, \ldots, m_2$ are all in stage d, so T_{m_1} is the start of stage d and T_{m_2} is the end, both deterministic. Then $\log(2N-3)!! \sum_{T_m \text{ depth } d} \pi_u(T_m \mid T_0)$

$$\begin{aligned}
&= \sum_{T_m \text{ depth } d} \sum_{T_{m-1}} p(T_m, T_{m-1} \mid T_0) \mathbb{1}(T_{m-1} \text{ decreases } M_u) \log \frac{p(T_m \mid T_{m-1}, T_0)}{p_{\mathrm{PDA}}(T_m \mid T_{m-1})} \\
&= \mathbb{E}_{T_{m_1+1:m_2} \sim p(T_{m_1+1:m_2} \mid T_0)} \sum_{m=m_1+1}^{m_2} \mathbb{1}(T_{m-1} \text{ decreases } M_u) \log \frac{p(T_m \mid T_{m-1}, T_{m_2})}{p_{\mathrm{PDA}}(T_m \mid T_{m-1})} \\
&= -\mathbb{E}_{T_{m_1+1:m_2} \sim p(T_{m_1+1:m_2} \mid T_0)} \log p_{\mathrm{PDA}}(A_u, B_u) \\
&= -\log p_{\mathrm{PDA}}(A_u, B_u).
\end{aligned}$$

The third equality follows from summing the log probabilities of revealing the splitting of u one node at a time. Therefore, $\pi_u(T_m \mid T_0)$ is a probability distribution! To sample from π therefore we can sample u in proportion to $-\log p_{\mathrm{PDA}}(A_u, B_u)$ then sample from π_u.

Now how do we sample from π_u? Calling $I_u(a_u, b_u) = -\frac{\bar{a}_u}{M_u} \log f(a_u, b_u \mid M_u) - \frac{\bar{b}_u}{M_u} \log f(b_u, a_u \mid M_u)$ (and $I_u(0,0) = 0$) we have, with a bit of algebra,

$$\pi_u(T_m \mid T_0) \propto p(T_m \mid T_0) \frac{M_u}{\sum_{u'} M_{u'}} I_u(a_u, b_u).$$

If $u \in U_d$, then $p(T_m \mid T_0)$ is a hypergeometric over $(M_u, a_u)_{u\in U_d}$:

$$\begin{aligned}
\binom{\sum_u N_u}{\sum_u M_u}^{-1} \prod_{u\in U_d} \binom{A_u}{a_u}\binom{B_u}{b_u} &= \left(\binom{\sum_u N_u}{\sum_u M_u}^{-1} \prod_{u\in U_d} \binom{N_u}{M_u} \right) \\
&\quad \prod_{u\in U_d} \times \mathrm{HyperGeom}(N_u, A_u, N_u - M_u)(a_u) \\
&= \left(\binom{\sum_u N_u}{\sum_u M_u}^{-1} \binom{N_u}{M_u}\binom{\sum_{u'\neq u} N_{u'}}{\sum_{u'\neq u} M_{u'}} \right) \\
&\quad \times \mathrm{HyperGeom}(\sum_{u'\neq u} N_{u'}, (A_{u'})_{u'\neq u}, \sum_{u'\neq u} M_{u'})(M_{u'})_{u'\neq u} \\
&\quad \times \prod_{u\in U_d} \mathrm{HyperGeom}(N_u, A_u, N_u - M_u)(a_u).
\end{aligned}$$

This also shows that once we've sampled the variables affected by the weight in front of $p(T_m \mid T_0)$ — $M_u, \sum_u M_u$ and a_u — then we can sample $(M_{u'})_{u' \neq u}$ and $(a_{u'})_{u' \neq u}$ from the hyper geometrics above. Now we only need to sample from a distribution proportional to

$$\frac{\binom{\sum_{u' \neq u} N_{u'}}{\sum_{u' \neq u} M_{u'}} \binom{N_u}{M_u}}{\binom{\sum_u N_u}{\sum_u M_u}} \frac{M_u}{\sum_{u'} M_{u'}} \hat{\pi}_u(a_u, M_u)$$

where $\hat{\pi}_u(a_u, M_u) = I_u(a_u, b_u)\text{HyperGeom}(N_u, A_u, N_u - M_u)(a_u)$. But that pre-factor is

$$\frac{\binom{\sum_{u' \neq u} N_{u'}}{\sum_{u' \neq u} M_{u'}} \binom{N_u}{M_u}}{\binom{\sum_u N_u}{\sum_u M_u}} \frac{M_u}{\sum_{u'} M_{u'}} = \binom{\sum_{u' \neq u} N_{u'}}{\sum_{u' \neq u} M_{u'}} \frac{B(\sum_u M_u, \sum_u (N_u - M_u) + 1)}{B(M_u, N_u - M_u + 1)}$$

which is a beta binomial

$$\text{BetaBin}\left(\sum_{u' \neq u} N_{u'}, M_u, N_u - M_u + 1\right)\left(\sum_{u' \neq u} M_{u'}\right)$$

To sample from $\hat{\pi}_u(a_u, M_u)$ we build the $A_u \times B_u$ table explicitly. This gives Alg. 4.

Algorithm 4 Importance sampling the **top-down** noise schedule

Require: Tree T_0 with N leaves

Draw $u \sim \text{Categorical}(-\log p_{\text{PDA}}(A_u, B_u)/\log(2N-3)!!)$. Call $d = d_{T_0}(u)$.

Draw $a_u, M_u \sim \text{Categorical}\left(\hat{\pi}_u(a_u, M_u)/\sum_{a,M} \hat{\pi}_u(a, M)\right)$.

Draw $\sum_{u' \in U_d \setminus \{u'\}} M_{u'} \sim \text{BetaBin}\left(\sum_{u' \in U_d \setminus \{u'\}} N_{u'}, M_u, N_u - M_u + 1\right)$.

Draw $(M_{u'})_{u' \in U_d \setminus \{u'\}} \sim \text{HyperGeom}\left(\sum_{u' \in U_d \setminus \{u'\}} N_{u'}, (A_{u'})_{u' \in U_d \setminus \{u'\}}, \sum_{u' \in U_d \setminus \{u'\}} M_{u'}\right)$.

Draw $(a_{u'})_{u' \in U_d \setminus \{u'\}} \sim \prod_{u' \in U_d \setminus \{u'\}} \text{HyperGeom}(N_{u'}, A_{u'}, N_{u'} - M_{u'})$.

For each $u \in U_d$ assign a_u taxa uniformly at random to one child and b_u to the other.

return T_m

F Architecture details

Our model q_θ must condition on the sequences $X_{1:N}$ and current state of the tree T_m. To condition on the sequences, we adapt the EvoPF architecture from Phyloformer 2 [12]. EvoPF embeds N sequences of length L, $X_{1:N}$, into a tensor of size $[N, L, 128]$ and performs axial attention over the dimensions N and L separately. After a trunk made of 12 layers, we turn embeddings from each pair of node into a set of logits.

F.1 An Embedding trunk that conditions on the tree T_m

We pass sequence embeddings through twelve EvoPF layers modified to condition on the tree structure T_m. We make two modifications that allow us to condition on T_m:

Internal nodes We include internal nodes of T_m in our calculation, so our tensor is of size $[N + \#\texttt{internal_nodes}(T_m), L, 128]$. Say $h_v^{(k)} \in \mathbb{R}^{(L+1)\times 128}$ is the embedding of node or leaf v at layer k. The $+1$ is from an extra position we also add to the beginning of each sequence which will allow us to read out a single vector.

For an internal node v we initialize $h_v^{(0)}$ using topological information and information about nearby leaves.

- For topological information on v we use the depth and the number of leaves in the subtree $\texttt{depth}(v), |\text{leaves}(v)|$. We call these features $f_{\text{topo},v}$.
- For every position ℓ, we average the embeddings of all leaves in the subtree of the node $\frac{1}{|\text{leaves}(v)|}\sum_{w \in \text{leaves}(v)} h_{w,\ell}^{(k)}$. We call these features $f_{\text{subtree},v,\ell}^{(0)}$.

- For every position ℓ, we average the embeddings of all leaves according to the Dirichlet energy features from Xie and Zhang [96]. This can be summarized as multiplying by a $|T_m| \times |T_m|$ matrix A: $\sum_{w \in T_m} A_{v,x} h^{(k)}_{w,\ell}$). We call these features $f^{(0)}_{\text{Dirichlet},v,\ell}$.

We feed all of these features through an MLP at every position

$$h^{(0)}_{v,\ell} \leftarrow \text{MLP}_{0,\theta}\left(f_{\text{topo},v}, f^{(0)}_{\text{subtree},v,\ell}, f^{(0)}_{\text{Dirichlet},v,\ell}\right).$$

After each layer, each internal node v has graph information injected into it as

$$h^{(k)}_{v,\ell} \leftarrow h^{(k)}_{v,\ell} + \text{MLP}_{k,\theta}\left(f_{\text{topo},v}, f^{(k)}_{\text{subtree},v,\ell}, f^{(k)}_{\text{Dirichlet},v,\ell}\right)$$

where $\text{leaves}(v)$ denotes the leaf descendants of v in T_m.

Attention bias During attention over N we bias the attention using the graph distance in T_m (which can be ∞): for nodes v, u, position ℓ, and head h,

$$\text{Attention}_{v,u,\ell,h} \leftarrow \text{Attention}_{v,u,\ell,h} + \text{embed}_{\theta,h}(\texttt{dist}(u, v \mid T_m)).$$

If u and v are leaves then we also bias the attention using their Hamming distance:

$$\text{Attention}_{v,u,\ell,h} \leftarrow \text{Attention}_{v,u,\ell,h} + \text{MLP}_{\theta,h}(\texttt{sinusoidal}(\texttt{hamming}(X_u, X_v))).$$

F.2 Readout

After the trunk we arrive at a vector $\texttt{embed}_\theta(v) = h^{(12)}_{v,\ell} \in \mathbb{R}^{128}$ for every vertex or leaf v in T_m and position ℓ which depends on both T_m and $X_{1:N}$. Now we need a strategy to first pool over the length dimension and then a strategy for each paradigm to get a set of scores for each possible T_{m-1} which we then normalize.

Covariance pooling along the length dimension EvoPF [12] pools over the length dimension by taking a mean. We found this resulted in embeddings that were nearly identical for different nodes, especially when sequences were long. This made training extremely unstable.

We suggest a fix inspired by classical phylogenetics methods. To motivate the method, first note at initialization, for a leaf v, $h^{(12)}_{v,\ell}$ is roughly the embedding of the letter at position ℓ in v. When we take a mean, $\frac{1}{L+1}\sum_\ell h^{(12)}_{v,\ell}$ then our embedding is simply the average composition of v. For randomly generated trees, each nucleotide or amino acid appears in roughly the same frequency across sequences, so this embedding is nearly identical across all v. To build useful embeddings, phylogeneticists instead build a matrix of average similarity across sequences, which we emulate by taking

$$\texttt{embed}(v, w) := \frac{1}{L+1}\sum_\ell h^{(12)}_{v,\ell} \circ h^{(12)}_{w,\ell} - \left(\frac{1}{L+1}\sum_\ell h^{(12)}_{v,\ell}\right) \circ \left(\frac{1}{L+1}\sum_\ell h^{(12)}_{w,\ell}\right)$$

for all pairs $v, w \in T_m$. At initialization, entries of $h_{u,w}$ are large for leaves v, w which have similar sequences, and small for those with dissimilar sequences. Note this pooling step can not straightforwardly be dropped into EvoPF – it is specific to our stretegy of iteratively building up the tree.

After pooling we apply a self-attention layer to $\{h_{v,w}\}_{v,w \in T_m}$. Note the number of pairs of sequences is typically small compared to the length L, so this computation has negligible cost in our overall architecture.

Scoring T_{m-1} For all pairs of nodes $v, w \in T_m$ we now have a vector of similarities $\texttt{embed}(v, w)$. We simply choose the right pair of nodes to compare to compute un-normalized logits.

- **bottom-up**: for each pair of subtree roots v, w in the current forest,

$$\text{logit}(v, w) = \texttt{MLP}\left(\texttt{embed}(v, w)\right).$$

- **middle-out**: here we need to score each unattached taxon t against each parent-to-child edge (v, w) in T_m. To do so, before pooling the length dimension we define edge embeddings

$$h^{(12)}_{(v,w),\ell} = W_\theta[h^{(12)}_{v,\ell}, h^{(12)}_{w,\ell}]$$

for a matrix $W_\theta \in \mathbb{R}^{128\times 256}$ so that our pooling step gives us $\texttt{embed}(t, (v, w))$. Finally we take

$$\text{logit}(t, (v, w)) = \texttt{MLP}\left(\texttt{embed}(t, (v, w))\right).$$

- **top-down**: each taxon t at the deepest layer of T_m sits at a host internal node with left and right children $\texttt{left}(t), \texttt{right}(t)$; the percolation logits are

$$\text{logit}_{\text{left}}(t) = \texttt{MLP}\left(\texttt{embed}(t, \texttt{left}(t))\right),$$
$$\text{logit}_{\text{right}}(t) = \texttt{MLP}\left(\texttt{embed}(t, \texttt{right}(t))\right).$$

if t has only one child then the embedding of the missing child is set to be a fixed vector plus $-\texttt{embed}(t, \texttt{child}(t))$. Note for top-down there can be multiple taxa at a node; we therefore represent nodes and taxa that may occupy a node separately in our trunk.

G Experiment details

G.1 Additional information on training data

We simulate data from three priors by first simulating a tree, and then simulating sequences conditional on the tree.

G.1.1 Dataset size

For each set of data point, we sample

$$N \sim \text{Uniform}(25, 65), L \sim \text{Uniform}(300, 2600).$$

Then we generate a tree with N species and sequences with length L before any insertions or deletions.

G.1.2 Trees

MrBayes [43] and CAT [53] both have uniform (proportional to distinguishable arrangements (PDA)) priors on tree topologies. We simulate PDA trees using the Ford-α process with $\alpha = 1/2$ [33].

G.1.3 Sequences and branch lengths

All simulations were done using AliSim [58], part of the IQ-TREE software used under the GPL-2.0 license. We use L to denote the length of the sequences.

Simple model To simulate from the Jukes-Cantor model, we passed `JC` to the model flag in AliSim. We simulate branch lengths independently from $\text{Exp}(1)/10$ to match the default MrBayes prior.

Complex model We simulate from a CAT model [53] which models site-wise biases in mutation rates in amino acid sequences. This model has branch lengths $(r_{u\to v})_{u,v}$, class diversity parameter $\alpha \in (0, \infty)$, and a symmetric generator matrix Q representing relative mutation rates as global parameters. Then each site ℓ mutates according to its rate g_ℓ and its stationary distribution $\vec{\pi}_\ell$. In particular, we (1) modify Q to get a stationary rate $\vec{\pi}_\ell$, $\tilde{R}_\ell = Q\text{diag}(\vec{\pi}_\ell) - \text{diag}(Q\vec{\pi}_\ell)$; (2) normalize the rates $R_\ell = \tilde{R}_\ell/(-\vec{\pi}_\ell^T \text{diag}(\tilde{R}_\ell))$; and finally (3) multiply by the rate $Q_\ell = g_\ell \times R_\ell$.

We simulate parameters from a CAT prior as used in PhyloBayes MPI [54]. See the manual at `https://github.com/bayesiancook/phylobayes/tree/master`.

- $Q_{ij} \sim \text{Exp}(1)$ for $i < j$. This matches the `-gtr` flags.
- $\alpha \sim \text{Exp}(1) \times 10$.
- $r_{u\to v} \sim \text{Exp}(1)/\mu$ independently where $\mu \sim \text{Exp}(1) \times 10$.
- $(g_\ell)_{\ell=1}^L \sim \text{Gamma}(\theta, 1)/\theta$, discretized to 4 components, which has mean 1 and variance $1/\theta$, where $\theta \sim \text{Exp}(1)$.
- $(\vec{\pi}_\ell)_\ell^L \sim \text{DirichletProcess}(\alpha, \text{Dirichlet}(\eta_1, \ldots, \eta_{20}))$ where each $\eta_b \sim \text{Exp}(1)$. This matches the `-cat` flag. We simulate class probabilities using the standard stick-breaking procedure up to 100 classes.

Finally we can simulate this prior using AliSim by passing `GTR20{...}+FMIX{...}+G4{...}` filling in the sampled parameters for Q, the probabilities and classes from the sampled stick breaking process, and θ.

Intractable model To simulate from a realistic prior including indels, we first simulate from a standard model of DNA mutation, `GTR{q_matrix}+GC{g_std}+I{invar_rate}`, where `q_matrix` is sampled from a Dirichlet as in the CAT model, `g_std` $\sim \text{Exp}(1)$, and `invar_rate` $\sim \text{Uniform}(0, 0.4)$. AliSim will simulate the stationary DNA distribution, what we called $\vec{\pi}$ in the CAT model, from an empirical distribution. We also simulate branch lengths from $\text{Exp}(1)/10$.

To add insertions and deletions, we pass `-indel ins_rate,del_rate` where we simulate `ins_rate` $\sim \text{Uniform}(0, 0.01)$ and `del_rate` $\sim \text{Uniform}(0, 0.05)$. This simulates insertions up to 1% and deletions up to 5% as frequently as substitutions; AliSim then samples insertion and deletion sizes from an empirically fit Zipfian distribution [32].

After simulating, we align sequences using `kalign3` [55] with default parameters before performing inference. Note models do not see the ground truth insertion and deletion labels generated during simulation, only those inferred by `kalign3`.

Missing-ness To simulate missing-ness, we draw a $p \sim \text{Beta}(1.25, 8.75)$ and give each non-gap position an independent $\text{Bernoulli}(p)$ chance of being masked.

G.2 Real sequence data

We downloaded sequence datasets from Lakner et al. [50]. We use DS1 originally taken from Hedges et al. [39], DS6 original taken from Brower [16], and DS9 originally taken from Rossman et al. [73].

G.3 Inference with likelihood-based methods

We compared our models to MCMC sampling with MrBayes and a maximum likelihood based method, IQ-Tree.

G.3.1 MrBayes

We used code from `https://nbisweden.github.io/MrBayes/download.html` under a GPL-3.0 license. We ran MrBayes with a uniform stationary distribution `prset statefreqpr=fixed(equal)` and exponential branch lengths `prset brlenspr=unconstrained:exponential(10.0)`. Our gold standard runs were run until the convergence criterion, the standard deviation in subsplit probabilities across two runs, was less than 0.01, sampling every 1000 steps.

G.3.2 IQ-Tree

We use IQ-Tree3 [94] code from `https://github.com/Cibiv/IQ-TREE` under a GPL-2.0 license. We ran IQ-tree with 16 CPUs with the flag `-nt AUTO` and otherwise used default settings. We ran IQ-Tree with `-m JC` for both DNA datasets and `-m Poisson` for the protein dataset. Our "flexible" IQ-Tree model was `-m GTR+G+I` for the DNA dataset and `-m LG+F+G+I` for the protein dataset. We attempted to use the more flexible `-m LG+C60+F+G+I` for the protein dataset but this took over an hour to generate a single tree for a dataset with $N = 45$ and $L = 1000$. To generate 8 trees to build a consensus, we ran IQ-Tree with 8 different seeds.

G.4 Evaluation metrics

We describe how we compare trees and their distributions using metrics derived from the Robinson-Foulds distance.

G.4.1 Robinson-Foulds distance

The Robinson-Foulds (RF) metric is often used for comparing trees due to its ease of computation [30, Ch. 30]. The metric can be used to compare rooted and unrooted trees. Since JC and CAT are reversible, the position of the root is not identifiable, so we use the unrooted RF distance to compare all trees.

To compute the RF distance, we compute the partition features of each tree: for a binary partition $P_1, P_2 \subset \{1, \ldots, N\}$ a tree has that feature if there is an edge in the unrooted tree separating all

nodes P_1 from all nodes P_2. Then the distance between two trees is simply the symmetric difference of their features.

G.4.2 Comparing tree distributions using RF MMD

To compare two distributions p, q, using a kernel k, we can calculate

$$\mathrm{MMD}_k^2(p,q) = \mathbb{E}_{T_1\sim p,T_2\sim p}k(T_1,T_2) + \mathbb{E}_{T_1\sim q,T_2\sim q}k(T_1,T_2) - 2\mathbb{E}_{T_1\sim p,T_2\sim q}k(T_1,T_2).$$

We can build kernels using the RF distance: we can think of RF as embedding a tree T into $v(T) \in \mathbb{R}^{2^{N-1}}$ with dimensions indexed by binary partitions of $\{1,\dots,N\}$, where, for a partition $P_1, P_2 \subset \{1,\dots,N\}$, $v(T)_{\{P_1,P_2\}} = 1$ if there is an edge separating all nodes P_1 from all nodes P_2. Then the RF distance between trees T, T', is simply $\|v(T) - v(T')\|$.

We could compare distributions with the linear kernel $k(T,T') = v(T)^T v(T')$; the MMD in this case is simply the distance between the vectors of subsplit probabilities $\|\mathbb{E}_{T\sim p}v(T) - \mathbb{E}_{T\sim q}v(T)\|$. However, two distinct distributions can have zero MMD simply by matching the average subsplit probabilities. To more flexibly detect distributional differences we look at a more flexible kernel $k(T,T') = \exp(-\|v(T)-v(T')\|^2/2\sigma^2)$ where we chose $\sigma^2 = (N-3)/2$ so $\|v(T)-v(T')\|^2/2\sigma^2$ is twice the "relative RF" distance between T and T'.

G.4.3 JC marginal likelihood estimate

We wanted to estimate the average entropy $-\mathbb{E}_{T_0,X_{1:N}} \log p(T_0 \mid X_{1:N})$ on the simulated JC data where p is the true prior model. Then we could bound the absolute error of our models as

$$\mathbb{E}_{X_{1:N}} D_{\mathrm{KL}}(p(T_0 \mid X_{1:N}) \| q_\theta(T_0 \mid X_{1:N})) \le \mathbb{E}_{T_0,X_{1:N}} \log p(T_0 \mid X_{1:N}) - \mathrm{ELBO}(q_\theta).$$

The entropy isn't tractable so we estimated it two ways.

First we ran MrBayes with sequences $X_{1:N}$ to get a large number of posterior samples $T^1, \dots T^M$ and simply estimated $p(T_0 \mid X_{1:N}) \approx \hat{p} = \frac{\#\{m|T^m=T_0\}}{M}$. This is an under-estimate as $\mathbb{E}\log\hat{p} \approx \log p(T_0 \mid X_{1:N}) - \frac{1}{2}\left(\frac{\mathrm{std}\hat{p}}{\hat{p}}\right)^2$. After a simulation, if $\frac{1}{2}\left(\frac{\mathrm{std}\hat{p}}{\hat{p}}\right)^2$ was greater than 0.1 then we re-ran the simulation for twice as long, repeating until the estimated error was less than 0.1. We ran MCMC with `mcmc nruns=4 nchains=4 ngen=2000000 samplefreq=500` and increased `ngen` as necessary.

Second we estimated the likelihood by first decomposing

$$\log p(T_0 \mid X_{1:N}) = \log p(T_0) + \log p(X_{1:N} \mid T_0) - \log p(X_{1:N}).$$

Under the PDA prior, $\log p(T_0) = -\log \#\{\text{trees of size } N\}$. The next two terms require marginalizing over branch lengths, and $\log p(X_{1:N})$ also involves marginalizing over trees. These were estimated using the stepping-stone algorithm in MrBayes. This algorithm again gives an underestimate for each of these quantities and reports a standard deviation; again, if the standard error was larger than 0.1 then we re-ran the simulation for twice as long. We ran the stepping stone algorithm, with or without a fixed tree, with `ss ngen=1000000 nruns=4 nchains=4 samplefreq=500 nsteps=50 alpha=0.4` and increased `ngen` as necessary.

For 100 sampled tree-sequence pairs, our first method failed to sample the true tree after 10^9 generations for 15/100 pairs. On the remaining pairs, which typically exclude those high-entropy trees, our first method gave an average of 9.4 and the second method gave an average of 8.9; these were further apart than our 0.1 threshold likely because the simple approximations above were not accurate. On all the data, our second method gave an estimate of 10.8 nats with a standard error of 0.5. We therefore report a value of > 10 nats in the main text. In principle we could perform the same estimation with CAT, but the computational cost is prohibitive: even this simulation took roughly 2 hours on 8 CPUs to evaluate a single tree-sequence pair.

G.5 Phyloformer 2

We trained Phyloformer 2 [12] for 48 hours on two A100s — the same as our models. We used the training architecture and parameters from the supplementary information of `https://openreview.net/forum?id=18pFYYqFiG`, including the branch-length canonicalization. However we removed the branch length prediction head. In particular the model was trained with a learning rate of 0.0001,

with a batch size of 64 and precision of `fp32`. The settings are similar to our model except we use lower precision and half the batch size. In the original code the trained on data with a fixed N and L; we modified their code to train on varied N and L every batch just like our model.

G.6 Quartformer

We trained Quartformer [82] models for 48 hours on two A100 GPUs, consistent with our models. The architecture details follow from their paper, including sampling blocks of 16 taxa and taking a 3-way cross-entropy loss for all 1820 quartets. Our only changes are:

- We use a a learned column MLP mean-pooled over sites rather than their histogram approach, which would be intractable for the larger vocabulary of amino acids
- Replacing the post-normalization with pre-normalization blocks and final LayerNorm, to improve training stability.

To sample a tree, we sample 16-taxa blocks until each taxon appears at least 40 times and store the predicted outputs. Following Su and Jiao [82], the predictions are then assembled using wQFM.

H Supplementary results

H.1 Training strategy ablations

Here we ablate choices in our training strategy.

Noise schedule Above we derived noise schedules so that de-noising steps were generated in proportion to where information is generated when $p(T_0)$ is the PDA distribution; this makes the variance in our loss estimate 0 at the beginning of training. Compared to sampling m uniformly however, Fig. 7 there is little, if any, benefit.

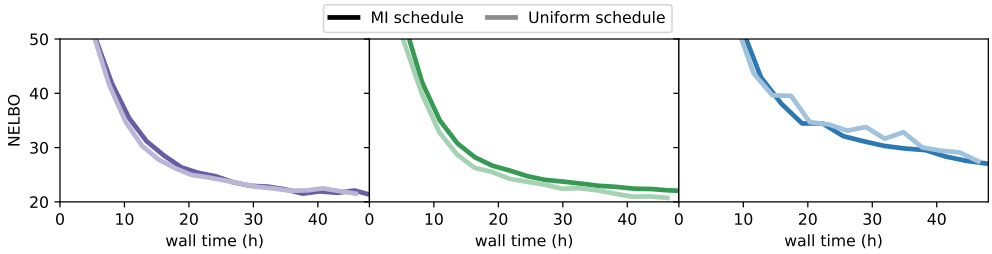


Figure 7: **The bespoke mutual information schedules do not make much of a difference when training.** On the "complex" data, with the same architecture and training parameters we train models for 24 hours on 2 A100s. We train models either with our derived mutual information schedule or by sampling the sampling step m uniformly between 1 and M.

We hypothesize this is because during training the model diverges very sharply away from PDA which was the basis of the noising process, starting from a trivial average likelihood of near 150 nats (the entropy of PDA on our data) to around 20 nats. Since PDA is the best approximation to $p(T_0 \mid X_{1:N})$ that does not use any information about $X_{1:N}$, a better noising distribution must include information about the sequences themselves. On one hand, this suggests that a better schedule would require even more complicated analysis, but on the other, it suggests we can be satisfied with the simple uniform schedule.

Prior corrected initialization In App. C we showed that all three paradigms can express distributions between PDA and Yule. We consider PDA data and note that, while our **middle-out** paradigms is initialized at the PDA distribution, **top-down** and **bottom-up** are not. We hypothesized that this may harm training, so we multiplied the outputs of our neural networks by our derived PDA posterior $\log p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ for **bottom-up** and **top-down**. However, Fig. 8 we see that this improves **top-down**, whose PDA posterior is somewhat complicated, early in training. However, after one day of training there is no detectable benefit to doing so, suggesting these paradigms can learn the PDA correction in practice.

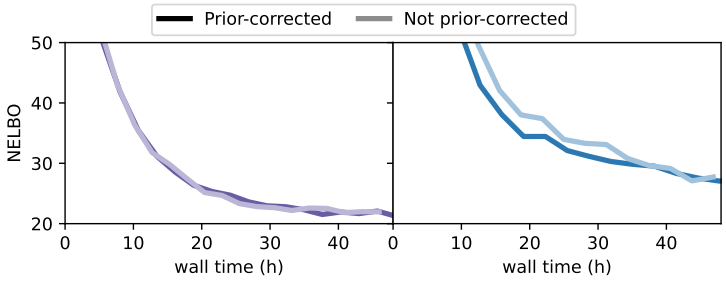


Figure 8: **The prior corrected outputs do not make much of a difference when training.** On the "complex" data, with the same architecture and training parameters we train models for 24 hours on 2 A100s. We train **top-down** or **bottom-up** models either by multiplying outputs by $\log p_{\mathrm{PDA}}(T_{m-1} \mid T_m)$ or leaving them biased at initialization..

H.2 Architecture ablations

Graph conditioning We ran ablations demonstrating the importance of our two strategies for injecting graph information into the model. We removed $f^{(0)}_{\mathrm{subtree},v,\ell}$ and $f^{(0)}_{\mathrm{Dirichlet},v,\ell}$ from the initialization of internal node initializations, and we removed $\mathrm{embed}_{\theta,h}(\texttt{dist}(u, v \mid T_m))$ from each attention layer. In Fig. 4, on the "complex" data, with the same architecture and training parameters we trained these models for 48 hours.

Pooling strategy ablations We show that an architecture with mean pooling performs substantially worse than one with covariance pooling. For a bottom-up model, we replace our readout with a mean $\texttt{embed}(v) = \frac{1}{L+1}\sum_\ell h^{(12)}_{v,\ell}$. We apply two dense layers and then read out, for each pair of subtree roots v, w in the current forest,

$$\mathrm{logit}_\theta(u, v) = \texttt{MLP}\left([\texttt{embed}(v) + \texttt{embed}(w), \texttt{embed}(v) \circ \texttt{embed}(w)]\right).$$

We trained these models in Fig 4, on the "complex" data, with the same architecture and training parameters for 48 hours.

H.3 Phyloformer 2 ablations

In the main text we saw that our models significantly outperform Phyloformer 2. We proposed that this was due to the increased flexibility of our models and our improved objectives: our models take in a partial tree T_m and do a full forward pass to predict a posterior over the next move T_{m-1}, meanwhile Phyloformer 2 constructs fixed embeddings for every sequence $X_{1:n}$ in one forward pass and these embedding vectors $\mathbb{R}^{N\times 128}$ must parameterize a distribution over trees that must predict a single "correct" path. However, there are two other plausible hypotheses for our improvements we ablate here.

Hypothesis 1: Our Phyloformer 2 ELBO is pessimistic Phyloformer 2 originally also predicted branch lengths. Without this prediction, the branch-length based canonicalization introduces an irreducible error in the ELBO. Note this can only explain the ELBO result and not the MMD or RF distance results where we compare actual samples from Phyloformer 2.

To describe the source of the pessimism, we need to introduce the canonicalization. As mentioned in the text, for a particular tree T, there are many possible paths $T = T_0, \ldots, T_M$. Phyloformer 2 defines a deterministic "correct" path $P(T, \vec{b}) = T_0, \ldots, T_M$ where $\vec{b}$ are the branch lengths and the cherry with the shortest branch lengths is removed every step (see Blassel et al. [12] for details). Then we can write an ELBO

$$\mathbb{E}_{T, X_{1:N}\sim p} \log q_\theta(T \mid X_{1:N}) \geq \mathbb{E}_{T, X_{1:N}, \vec{b}\sim p} \log q_\theta(P(T, \vec{b}) \mid X_{1:N}). \tag{13}$$

In Phyloformer 2 their goal was to predict $\log q_\theta(T, \vec{b} \mid X_{1:N})$ so the ELBO Eqn. 13 could be made tight if the model learned $q_\theta(T_0, \ldots, T_M, \vec{b} \mid X_{1:N}) = 0$ if $(T_0, \ldots, T_M) \neq P(T_0, \vec{b})$. In our case however, $P(T, \vec{b})$ is a random variable and the best prediction is $q_\theta(T_0, \ldots, T_M \mid X_{1:N}) = p(T_0 \mid X_{1:N}) p(P(T_0, \vec{b}) = (T_0, \ldots, T_M) \mid T_0, X_{1:N})$. The second term creates an irreducible gap in the ELBO of size $\mathbb{E}_{T, X_{1:N}\sim p}\mathrm{Entropy}(P(T, \vec{b}) \mid T, X_{1:N})$.

To ablate this effect, we replace the true branch lengths used to generate the tree with branch lengths predicted from T and $X_{1:N}$, i.e. we replace $P(T, \vec{b})$ with a $P(T, X_{1:N})$ so it is possible for the ELBO to be tight; i.e. $\text{Entropy}(P(T, X_{1:N}) \mid T, X_{1:N}) = 0$. To do so, we first estimate the distance between every pair of DNA sequences X_n and $X_{n'}$ as $b_{n,n'} = -\frac{3}{4} \times \log(1 - \frac{4}{3} \times \text{fraction mismatches})$ – this estimator converges to the exact distance under the JC model when $L \to \infty$. Then, traversing the tree from the bottom-up, for every node v with children ℓ and r, and other node w, we estimate $b_{v,w} = \frac{1}{2}(b_{\ell,w} + b_{r,w} - b_{\ell,r})$. Given these branch lengths $\vec{b}(T_0, X_{1:N})$, we then construct $P(T, \vec{b}(T_0, X_{1:N}))$

Hypothesis 2: Our architecture improvements explain the improved performance Here we port the improvements we made to our architecture to Phyloformer 2. In particular, we add μP-rescaled per-layer learning rates, initialization, and attention normalization [99], CompleteP scaled residual layers [23], EMA 0.9998, gradient-norm clip 500, and `bf16-mixed` precision. We scan learning rates $0.001, 0.003, 0.01, 0.03, 0.1$ each for 12 hours, and find that 0.003 is optimal. We see the loss is at each step is similar to before, but the throughput increased roughly by $2\times$.

Ablation results Fig. 9 shows the results of the ablations described above compared to the original Phyloformer 2 and **bottom-up** results. We see hypotheses 1 and 2 are not true: evaluating a non-pessimistic ELBO, or porting our architectural improvements to Phyloformer 2 did not majorly affect training. The model trained with a non-pessimistic ELBO may have done slightly worse than the original Phyloformer 2 because the canonicalization we propose may have been harder to learn.

On the other hand, as in App. H.1, we see Rao-Blackwellizing the objective results in a substantially better fit. However there is still a significant gap to our **bottom-up** model, which likely comes from our other contribution, the increased flexibility.

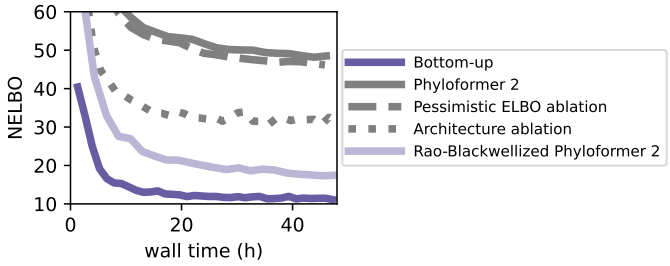


Figure 9: **Our improvements over Phyloformer 2 came from our improved objectives and the flexibility of our models.** We ablate the architectural improvements we've made and the form of the canonicalized objective. These changes don't make a large change, while Rao-Blackwellize does, but not enough to close the gap to our model.

H.4 Additional Normalized RF Results

In fig. 6, we showed that our models are more accurate than IQ-Tree for complex evolutionary models where IQ-Tree is misspecified. There, we held the number of taxa $N = 45$ fixed and varied the length of the sequences L. Here, we instead hold $L = 500$ fixed and vary the number of taxa N.

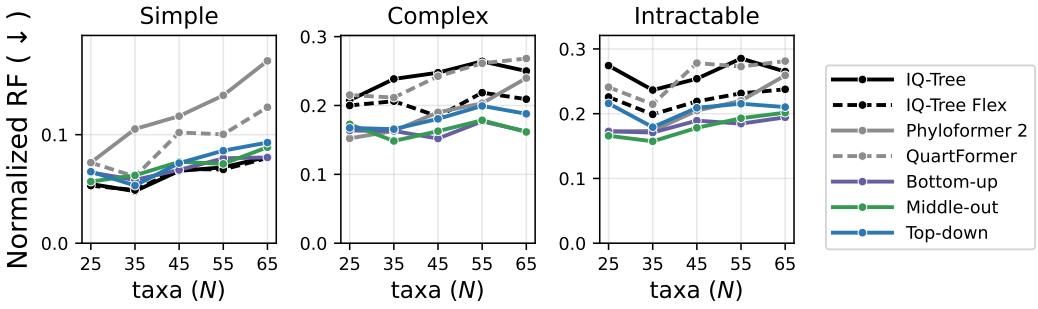


Figure 10: Our models make more accurate predictions than IQ-Tree for the complex and intractable evolutionary models. We hold the sequence length $L = 500$ fixed and vary the number of taxa N. The normalized RF distance of a random tree is roughly 1.

H.5 Posterior calibration

On the JC data we have access to a computationally cheap likelihood $p(X_{1:N} \mid T)$ and can approximate the ground truth posterior with MrBayes. This allowed us to quantify an approximate average KL between our inferred posteriors $q_\theta(T_0 \mid X_{1:N})$ and the true posterior. Here we ask whether *samples* from $q_\theta(T_0 \mid X_{1:N})$ look like samples from the true posterior. We also ask how much this depends on using our synthetic data versus real data which can be out-of-distribution. This extends the analysis in Sec. 7.2.

We consider classical evolutionary datasets labeled DS1-8 [50] numbered in order of increasing N and, for each dataset, simulate JC data with the same N and L. Then we compare our inferences to those of a long MrBayes run on these 16 data using 3 methods.

- *How far is $q_\theta(T_0 \mid X_{1:N})$ from the true posterior?* We compare them with MMD. MMD measures the average distance between trees sampled from separate distributions and subtracts the average distance between trees sampled from the same distribution, with the best MMD score of 0 achieved when the two distributions are identical. See App. G.4.2 for more details.
- *If we make conclusions about our confidence of evolutionary events $q_\theta(T_0 \mid X_{1:N})$ will they match the confidence of those from the true posterior?* In a tree, every parent-child branch partitions the sequences $X_{1:N}$ into two groups: those that are descendants of the parent, and everything else. Whether two species often belong to the same split or different split determines their related-ness. Therefore the posterior likelihood of a split is regularly used to make conclusions about our confidence of species-relatedness. So, for every possible split that appears in any observed tree, we count how likely it is to appear in the MrBayes posterior versus samples from our posterior.
- *Do samples from $q_\theta(T_0 \mid X_{1:N})$ look like those from the true posterior?* We plot samples for visual comparison.

In Fig. 11 we plot MMDs for the real and synthetic DS 1-8. We see we substantially outperform Phyloformer 2. We also see that our MMDs are near 0 for the synthetic data, but all models find the real data much more challenging.

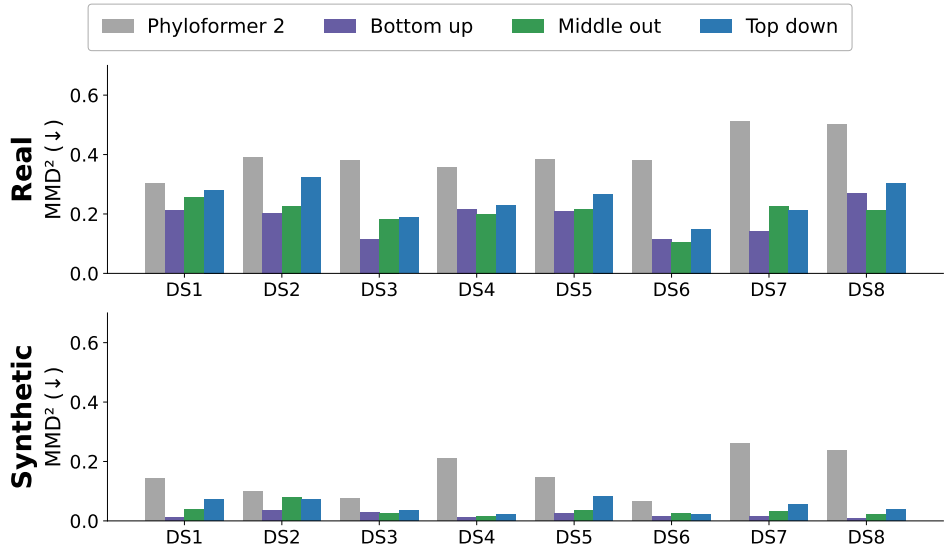


Figure 11: **Our model matches the MrBayes posterior distribution.** We plot MMDs on DS1-8 between 1000 samples from each model and MrBayes. The top row is a reproduction of Fig. 5b included for comparison.

In Fig. 12 we plot the posterior probabilities of each observed split from MrBayes and our models (each point is a split). We report the Spearman correlation (ρ) as well as the coverage of likely splits (we take all splits with a posterior probability ≥ 0.9 under MrBayes and report the average posterior

probabilities of these splits under our model). We see all models are under-confident (below the diagonal line), likely because the likelihood objective encourages models to spread-out. This is in particular true for the real data, suggesting that our models' errors come largely from abstaining to make confident predictions on OOD data. Again, we see our models are much more calibrated than Phyloformer 2, and that models are less calibrated on real data. However, we see that the coverage of our models remains high: most high probability splits in MrBayes are also high probability under our models.

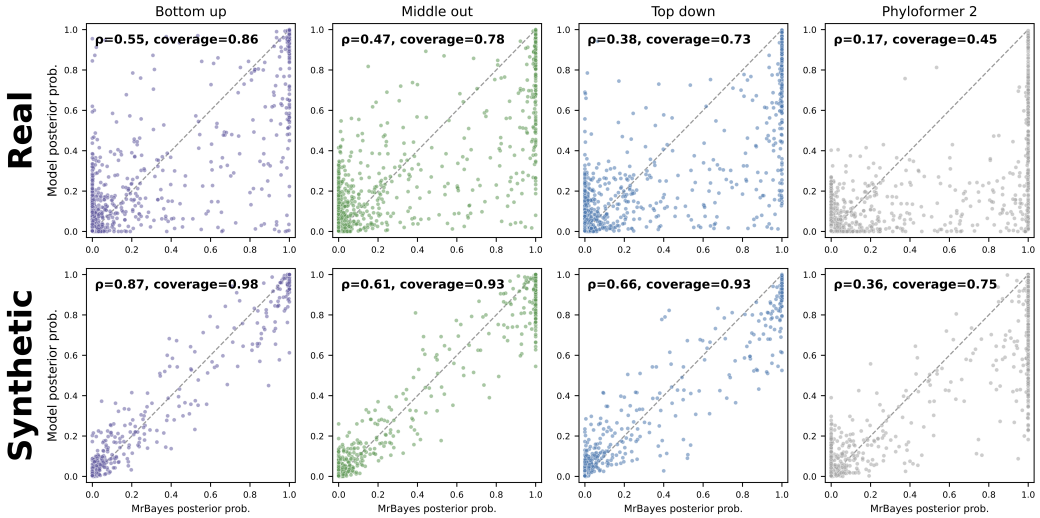


Figure 12: **Our models have accurate posterior split probabilities.** We plot posterior probabilities of observed splits on all of DS1-8 between 1000 samples from each model and MrBayes.

In Fig. 13 and 14 we plot the consensus and sample trees from each method for DS3 real and synthetic data. This is a setting in which the uncertainty was low, making it easier to visualize. We see, as above, that our models are under-confident and perform better on synthetic data. Trees from our models however clearly better respect the original tree than those from Phyloformer 2.

H.6 Model sampling timing comparisons

Our **bottom-up** and **middle-out** models samples scale as N steps of $\times O(N^4 + NL^2)$ per step. Phyloformer 2 in principle scales the same way but only requires one step. Despite the extra cost of our model, it is cheaper than the cheapest default settings of IQ-Tree for most parameter settings, making it appropriate for use in standard phylogenetics workflows. The exception is our **top-down** model, which needs many more iterative steps to generate a tree, between $\Theta(N \log N)$ and $\Theta(N^2)$ depending on the depths of the nodes in the tree (average $\Theta(N^{3/2})$ for PDA). The extra compute our models spend during sampling allow them to represent much more flexible distributions, bringing these resources to bear on this problem.

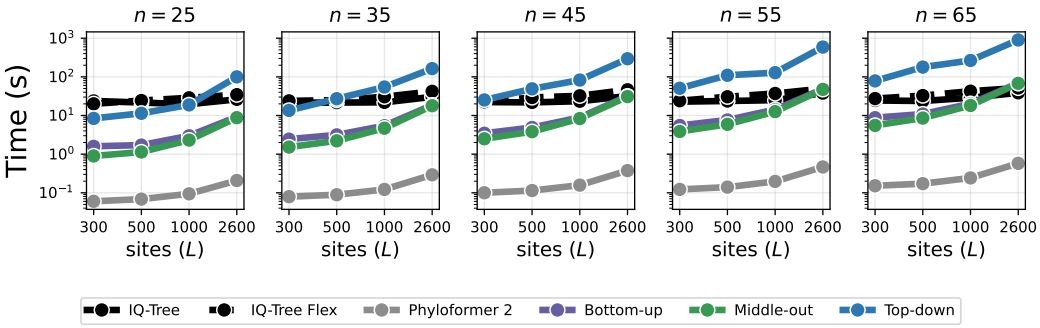


Figure 15: Comparison of sampling runtimes for our models and baselines on an A100 GPU.



Figure 13: Consensus and inferred trees on DS3. We plot the consensus over 500 sequences. We colour branches that don't appear in the MrBayes consensus black. We root trees by picking a fixed random out-group among the nodes. Clades are shaded grey to help in comparing trees. Species are ordered to allow us to show the MrBayes consensus tree as a planar tree.



Figure 14: Consensus and inferred trees on a synthetic DS3. We plot the consensus over 500 sequences. We colour branches that don't appear in the MrBayes consensus black. We root trees by picking a fixed random out-group among the nodes. Clades are shaded grey to help in comparing trees.

NeurIPS Paper Checklist

1. **Claims**

   Answer: [Yes]

   Justification: See Sections 3–7.

2. **Limitations**

   Answer: [Yes]

   Justification: See Section 8.

3. **Theory assumptions and proofs**

   Answer: [Yes]

   Justification: See Appendix.

4. **Experimental result reproducibility**

   Answer: [Yes]

   Justification: We provide code to reproduce our experiments at `https://github.com/AlanNawzadAmin/DeepPhylo`.

5. **Open access to data and code**

   Answer: [Yes]

   Justification: We provide code to reproduce our experiments at `https://github.com/AlanNawzadAmin/DeepPhylo`.

6. **Experimental setting/details**

   Answer: [Yes]

   Justification: See Section 6.

7. **Experiment statistical significance**

   Answer: [No]

   Justification: Retraining the models with different seeds would be too expensive.

8. **Experiments compute resources**

   Answer: [Yes]

   Justification: We describe compute resources in Section 6.

9. **Code of ethics**

   Answer: [Yes]

   Justification: We comply with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Answer: [Yes]

    Justification: See Section 1.

11. **Safeguards**

    Answer: [N/A]

    Justification: NA.

12. **Licenses for existing assets**

    Answer: [Yes]

    Justification: We use code for Phyloformer 2 from `https://openreview.net/forum?id=18pFYYqFiG`, which has the CC BY 4.0 license.

13. **New assets**

    Answer: [N/A]

    Justification: NA.

14. **Crowdsourcing and research with human subjects**

Answer: [N/A]

Justification: NA.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Answer: [N/A]

Justification: NA.

16. **Declaration of LLM usage**

Answer: [N/A]

Justification: No declaration required.