Likelihood-free inference of phylogenetic tree posterior distributions

Luc Blassel[*]
CQSB
Sorbonne Université, Paris, France

Noémie Sauvage
CQSB
Sorbonne Université, Paris, France

Pierre Barrat-Charlaix
CQSB
Sorbonne Université, Paris, France

Bastien Boussau
LBBE
Université Lyon 1, Villeurbanne, France

Nicolas Lartillot
LBBE
Université Lyon 1, Villeurbanne, France

Laurent Jacob[*]
CQSB
Sorbonne Université, Paris, France

Abstract

Phylogenetic inference, the task of reconstructing how related sequences evolved from common ancestors, is a central objective in evolutionary genomics. The current state-of-the-art methods exploit probabilistic models of sequence evolution along phylogenetic trees, by searching for the tree maximizing the likelihood of observed sequences, or by estimating the posterior of the tree given the sequences in a Bayesian framework. Both approaches typically require to compute likelihoods, which is only feasible under simplifying assumptions such as independence of the evolution at the different positions of the sequence, and even then remains a costly operation. Here we present the first likelihood-free inference method for posterior distributions over phylogenies. It exploits a novel expressive encoding for pairs of sequences, and a parameterized probability distribution factorized over a succession of subtree merges. The resulting network provides well-calibrated estimates of the posterior distribution leading to more accurate tree topologies than existing methods, even under models amenable to likelihood computation. We further show that its edge against likelihood-based methods increases under models of sequence evolution with intractable likelihoods.

1 Introduction

Reconstructing the evolutionary history of a set of contemporaneous sequences is a central task in genomics [18]: it is used to understand how extant species have evolved from common ancestors [52], how bacterial resistances to drugs have emerged and been disseminated [1], and how epidemics are spreading [12]. A key object in this endeavor is the phylogeny, a bifurcating tree summarizing the succession of transformations of the sequence that lead to the current observed diversity from a single ancestor.

Modern phylogenetic reconstruction is dominated by approaches based on probabilistic models of sequence evolution. Model-based approaches offer a principled framework for inference by making assumptions about evolutionary processes transparent, while also providing methods for model checking, criticism and iterative model elaboration [9].

[*]luc.blassel@sorbonne-universite.fr laurent.jacob@cnrs.fr

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

Models of sequence evolution are typically continuous time Markov processes parameterized by the phylogeny and by the rates at which amino-acids or nucleotides undergo substitutions. Existing reconstruction methods look for the phylogeny maximizing the likelihood of the sequences [27, 31] or, in a Bayesian perspective, aim at sampling from the posterior distribution of the phylogeny given the sequences through Monte Carlo strategies [14, 15, 2] or approximate this posterior distribution through variational inference [48, 20, 49, 4].

Across all these approaches, a major hurdle is the computational cost of evaluating the likelihood function, which is required either for maximizing the likelihood, computing acceptance probabilities in sampling strategies, or computing the evidence lower bound (ELBO) objective in the case of variational inference. For any single phylogeny, the likelihood is computed through a costly pruning algorithm [6] and many such evaluations are required to heuristically explore the set of possible tree topologies—$(2N-5)!!$ for a phylogeny over N leaves. Furthermore, just making this computation feasible has resulted in a focus on probabilistic models that make simplifying assumptions such as independence and identical distribution of the evolutionary process at each position in the sequence, or the absence of natural selection. These simplifications are known to produce unrealistic sets of sequences [43], artifacts in reconstructed phylogenies [42], and impede our ability to understand the evolutionary history of living species.

Simulation-based or likelihood-free inference has emerged as a powerful paradigm for estimation under probabilistic models under which likelihood evaluations are intractable but sampling is cheap [3, 26]. This paradigm has leveraged advances in deep learning to approximate posterior distributions by neural networks trained over data simulated under probabilistic models [10, 26]. Among these methods, neural posterior estimation [NPE, 26] defines a family of distributions parameterized by a neural network whose weights are then optimized to approximate the posterior. In addition to working around the need for likelihood evaluation, NPE is amortized: training the network can take time but performing inferences with the trained network is typically very fast. By contrast, existing variational inference and Monte Carlo samplers require new computation every time a new inference is done. On the other hand, NPE requires to define a parameterized family of distributions that is appropriate for the posteriors of phylogenies given sequences, which is not straightforward.

Here we introduce Phyloformer 2, an NPE for phylogenetic reconstruction, with the following contributions:

- We propose a parameterized family of posterior distributions on phylogenies given a set of sequences, factorized through a succession of pairwise mergings. Optimizing the weights of our network to maximize the log-probability within this family yields a likelihood-free estimate of the corresponding posterior that can then be used for sampling trees. To our knowledge, this is the first likelihood-free posterior estimation method trained end-to-end from sequences to the phylogeny beyond quartets.
- To extract the parameters of the approximate posterior distribution from the input sequences, we introduce a novel evoPhyloFormer (evoPF) architecture akin to the EvoFormer module used in Alphafold 2 [17], that is both more scalable and expressive than the one used in Phyloformer. The overall architecture allows us to scale up to over 250 sequences of length 500 or more than 350 sequences of length 250 on a single V100 GPU with 16Gb of VRAM.
- On data generated under a probabilistic model of sequence evolution with a tractable likelihood, Phyloformer 2 point estimates match both state-of-the-art likelihood-based and likelihood-free reconstruction methods in topological accuracy. PF2 produces calibrated estimates of the posterior which, when summarized, yield more accurate trees than state-of-the-art methods. Because it is likelihood-free, it is the first well-specified estimator under models with intractable likelihoods. We show that with such models, its topological error is lower than the one incurred by—misspecified—likelihood-based estimators.
- Because Phyloformer 2 is amortized, once trained, it performs inference 1 to 2 orders of magnitude faster than the—less accurate—state-of-the-art likelihood-based estimators.

Related work

Initial attempts to phylogenetic NPE were restricted to quartets, *i.e.*, topologies over four leaves, allowing them to cast the problem as a classification over the three possible topologies [39, 51, 41]. In this case, a vector embedding of the input sequences was extracted by a neural network and used to

produce three scalar outputs, and the probability of each topology was simply modeled as a softmax over these three outputs. Since the number of possible topologies grows super-exponentially with the number of leaves, this strategy cannot be generalized to larger numbers of sequences and even if it could, treating all topologies as separate classes would disregard the fact that some are more similar than others. In addition, Grosshauser et al. [11] showed that the method of Zou et al. [51] underperformed on more difficult tasks with short sequences and long evolution times.

Alternatively, Nesterenko et al. [29] proposed Phyloformer, a likelihood-free inference method defined for any number of leaves. Phyloformer doesn't estimate phylogenies, but evolutionary distances, *i.e.*, sum of branch lengths on the phylogeny between pairs of leaves. Phylogenies can be reconstructed from such distances with the neighbor joining algorithm [NJ, 37] , but the authors observed that this strategy led to a limited topological reconstruction accuracy. In addition, Phyloformer only allows for point estimates—namely the median of the posterior distribution of evolutionary distances—rather than the entire distribution. Finally, the network applied axial self-attention [13] to all pairs of sequences, which led to a large memory footprint even using a linear approximation of self-attention [19].

Phyloformer 2 addresses each of these limitations of Phyloformer by estimating the entire posterior distribution of phylogenies—instead of a single estimate of evolutionary distances— thanks to its BayesNJ module, and by using a novel evoPF encoding that is both more lightweight and more efficient than Phyloformer.

2 Background and notation

2.1 Notation

Let $x = \{x_1, \dots, x_N\}$ be a set of N sequences of L letters in some fixed alphabet representing organic compounds (*e.g.*, 20 possible amino acids for proteins, 4 possible nucleotides for DNA). We assume that the sequences are aligned, *i.e.*, correspond to N initial sequences of possibly different lengths whose positions were matched to minimize some score quantifying how similar all sequences are at each position, potentially by introducing gaps denoted by a special character in the alphabet [18].

A phylogeny $\theta = (\tau, \ell)$ over x is an unrooted binary tree τ with N leaves and a set ℓ of branch lengths in $\mathbb{R}_+^{2N-3}$. The tree τ can equivalently be represented by a succession of merges. Intuitively, given N species, one chooses two species, pair them to form a cherry, replace them by a single species representing their ancestor, and proceed recursively with the $N-1$ resulting species until the entire tree has been produced.

Formally, we denote this succession of merges as $\{m^{(k)}\}_{k=1}^{N-3}$ where $m^{(k)} \in \mathcal{C}^{(k)} \triangleq \{(v_i^{(k)}, v_j^{(k)}) \in \mathcal{S}_{(k)}^2 | i \neq j\}$ is a pair of two distinct elements in $\mathcal{S}_{(k)}$ the set of "mergeable" nodes at the kth merge, *i.e.*, either leaves or internal nodes whose children were already merged, and $\mathcal{C}^k$ is the set of candidate merges at step k. In order to build $\mathcal{S}_{(k)}$, we define$\mathcal{S}_{(1)}$ to be the set of leaves in τ and for $k > 1$, $\mathcal{S}_{(k)} \triangleq \{\{\mathcal{S}_{(k-1)} \cup u^{(k-1)}\} \setminus (v_i^{(k-1)}, v_j^{(k-1)})\}$, and $u^{(k)}$ is the common neighbor in τ to the nodes merged in $m^{(k)}$—which is always defined because we start from leaves and recursively replace pairs of elements by their common neighbor in the binary tree. We further denote $\ell^{(k)} = \{\ell_i^{(k)}, \ell_j^{(k)}\} \in \mathbb{R}_+^2$ the k-th cherry, *i.e.*, the set of two branch lengths connecting $m^{(k)}$ to $u^{(k)}$. In our context, the N leaves will represent the taxa with sequences in x, $u^{(k)}$ is the common ancestor of $m^{(k)}$, and $\ell^{(k)}$ is the set of evolutionary times between this ancestor and its two descendants.

2.2 Neural posterior estimation

For a probabilistic model $p(x|\theta)$ of the data x given the parameter θ and a prior distribution $p(\theta)$, NPE provides a way to estimate the posterior $p(\theta|x)$ in cases where evaluating $p(x|\theta)$ for a given (x, θ) is too costly or intractable, but where it is possible to sample from this model. It relies on a family of distributions $q_\psi(\theta|x)$ whose parameters ψ are provided by a neural network acting on the data x—by abuse of notation we denote $\psi(x)$ the parameters output by this neural network for an input x. NPE builds its approximation of $p(\theta|x)$ by looking for the q_ψ minimizing the average

Kullback-Leibler (KL) divergence with the true posterior:

$$\mathbb{E}_{p(x)}\left[\mathrm{KL}(q_\psi(\theta|x)||p(\theta|x))\right] = \mathbb{E}_{p(x)}\left[\mathbb{E}_{p(\theta|x)}\left[\log p(\theta|x) - \log q_\psi(\theta|x)\right]\right],$$

where the average is taken over the marginal $p(x) = \int p(x|\theta)p(\theta)d\theta$. Since the $p(\theta|x)$ term does not depend on ψ, minimizing this average KL divergence with respect to ψ is equivalent to maximizing the average approximate log-likelihood q_ψ over the joint distribution $p(x,\theta) = p(x|\theta)p(\theta)$ [see *e.g.*, 33]. This is generally achieved by minimizing a Monte Carlo approximation $\sum_{i=1}^{n} \log q_{\psi(x_i)}(\theta_i|x_i)$ of this average, built over a large number of examples $\{(x_i,\theta_i)\}_{i=1}^{n}$ sampled from $p(x,\theta)$ by successively sampling a θ_i from the prior and an x_i from the model given θ_i.

Consequently, NPE is guaranteed to converge to the true posterior $p(\theta|x)$, provided that the family q_ψ is expressive enough to represent $p(\theta|x)$ and that the optimization algorithm finds the optimum. More precisely, its approximation error depends both on the expressivity of the chosen family of distributions, *i.e.*, on the average KL divergence $\mathbb{E}_{p(x)}\left[\mathrm{KL}(q_{\psi^*(x)}(\theta|x)||p(\theta|x))\right]$ between the best distribution $q_{\psi^*(x)}$ in the family, and the true posterior for each x, and on the expressivity of the neural network, *i.e.*, on its ability to map every x to the corresponding best parameters $\psi^*(x)$.

3 Methods

Phyloformer 2 combines two novel modules. The first one, evoPF, encodes distribution parameters $\psi(x)$ from a set of aligned related sequences x, which is referred to as a multiple sequence alignment (MSA). The second one, BayesNJ, defines a family of posterior distributions $q_{\psi(x)}(\theta = (\tau,\ell)|x)$ on phylogenetic trees, parameterized by the output of evoPF. Figure 1a-b represents the overall architecture.

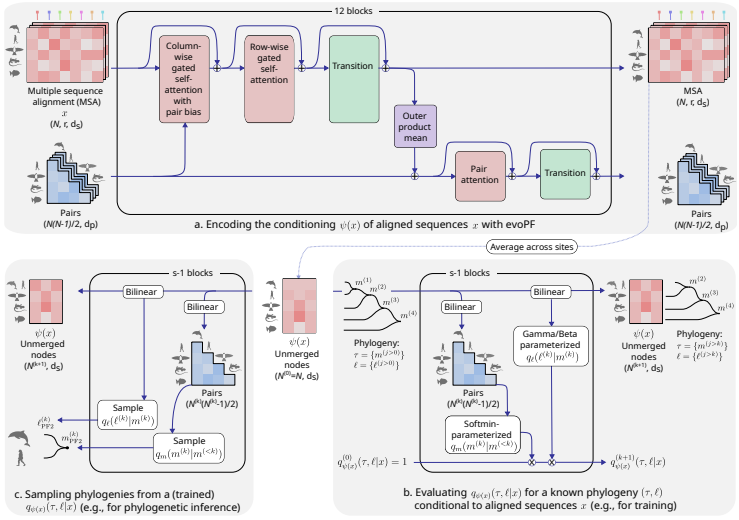

Figure 1: Architecture of Phyloformer 2. ***Panel a***: evoPF, an EvoFormer-inspired module updating $N \times L$ embeddings for a set of aligned sequences (MSA) x and $N(N-1)/2$ embeddings for pairs of sequences. Each of its blocks applies self-attention within both the MSA and representation, and ensures information sharing between them. After 12 blocks, we extract one embedding for each sequence by averaging the MSA embeddings across sites. ***Panel b***: BayesNJ (Algorithm 11) computes the posterior probability of a phylogeny given an MSA represented by the sequence and pair embeddings provided by evoPF. The probability is a product over a recursive operation where two taxa are merged into their parent, and the taxon representation is updated accordingly. ***Panel c***: at inference time, we apply the same succession of operations as for evaluating the probability, but either sampling or taking the modes of the distributions (Algorithm 12).

3.1 Encoding tree distribution parameters with evoPF

Our encoder should process a set x of aligned sequences and be expressive enough to capture sufficient information on their evolutionary relationships. Nesterenko et al. [29] encoded x with

one embedding for each aligned position in each pair of aligned sequences. This approach avoided the need to flatten information across positions, while maintaining a representation at the pair level, consistently with their predicting pairwise evolutionary distances. They noted that tuning down the architecture to one embedding for each position in each sequence—boiling down to the MSA transformer [34]—dramatically improved the method scalability but strongly affected the accuracy of their trained network. Inspired by this trade-off, we introduce evoPF, an encoder that maintains a single embedding per pair, and a separate representation with one embedding per position within each sequence (Figure 1a).

EvoPF is a transpose version of the EvoFormer module in Alphafold 2 [17]—whose objective was to capture spatial distances between pairs of aligned positions rather than evolutionary distances between pairs of homologous sequences—with a few simplifications. The MSA stack maintains an embedding for each position within each sequence in x. It relies on axial attention, alternating one layer of column-wise and one layer of row-wise gated self-attention. The row-wise mechanism takes each sequence separately, and applies self-attention between embeddings of all its positions, allowing information to flow within each sequence. Symmetrically, the column-wise mechanism lets the information flow between sequences by treating each position separately and applying self-attention over the embeddings of all sequences at this position. The two self-attention layers are followed by a transition layer applying a linear function and a ReLU to each embedding in x separately.

Parallel to this MSA stack, we maintain an embedding for each pair of sequences in x, updated in each evoPF block by applying self-attention between the embeddings of all pairs, followed by a transition layer like in the MSA stack. By contrast EvoFormer relied on triangular attention where pair (i, j) only attended to pairs (i, k) and (k, j). The MSA stack affects the pair stack through the addition of an outer product mean of sequence embeddings to the pair embeddings. Conversely, the pair stack affects the MSA stack by biasing its column-wise attention. We provide a complete description of evoPF in Algorithms 1 to 9 in Appendix A.1.

3.2 Defining a proper probability distribution over phylogenies with BayesNJ

After 12 evoPF blocks, we average the MSA embeddings across positions, yielding a single vector embedding per sequence. These embeddings constitute our encoding $\psi(x)$ of the sequences x, and our next goal is to define a family of distributions over phylogenies, conditional on x, and whose parameters are functions of $\psi(x)$. To this end, we define $q_{\psi(x)}\left(\theta = (\tau, \ell)|x\right)$ factorized over the succession of merges in τ, *i.e.* $q_{\psi(x)}\left(\theta = (\tau, \ell)|x\right) = \Pi_{k=1}^{N-3} q_m(m^{(k)}|m^{(<k)}) q_\ell\left(\ell^{(k)}|m^{(\leq k)}\right)$, where $m^{(<k)}$ denotes the set of merges with indices smaller than k, and where the probability of each merge is further decomposed into a topological factor q_m representing the probability of merging these two particular clades and a branch-length factor q_ℓ representing the probability of observing these evolutionary distances between the merged clades and their parents. Both q_m and q_ℓ are parameterized by $\psi(x)$ as detailed in 3.3. A caveat of this factorization across $N-3$ merges is that most phylogenies θ can be obtained by several distinct successions of merges from the leaves. For example, a balanced binary tree with four leaves a, b, c, d merging (a, b) and (c, d) can be obtained by merging any of the two groups first, and these two orders have no reason to lead to the same $\Pi_{k=1}^{N-3} q_m(m^{(k)}|m^{(<k)}) q_\ell\left(\ell^{(k)}|m^{(\leq k)}\right)$ in general. Properly defining a probability distribution over phylogenies therefore requires to sum over all possible orders of merge, which is not feasible even for moderately large N.

Alternatively, we must ensure that for a given phylogeny our distribution assigns a non-zero probability to a single merge order and, to be able to evaluate the probability of any phylogeny, that this order can be recovered efficiently from the phylogeny. We achieve this by designing a canonical merge order such that (i) we can guarantee that our sampling procedure always generates merges in this order and (ii) our evaluation procedure always processes merges in this order. Point (i) requires that the merge order does not depend on the stochastic parts of the sampling procedure, point (ii) requires that we can recover the canonical order efficiently for any given phylogeny. We achieve these two points by ensuring that at every step k, we select the merge corresponding to the two closest nodes currently available in $\mathcal{S}_{(k)}$—*i.e.*, whose children have already been merged. Conversely when sampling a phylogeny (τ, ℓ) from our $q_{\psi(x)}(\tau, \ell|x)$, we ensure that the distance between merged nodes (i, j) is larger than the distance between any previous merges done while (i, j) was a possible merge—*i.e.*, after their children were merged. A similar strategy was used in the likelihood-based Sequential Monte Carlo sampler [2].

3.3 BayesNJ parameterization of topological and branch-length components

We parameterize the topological component of q_ψ by a softmin across pairwise scores computed from a symmetric bilinear function over embeddings of pairs of mergeable nodes at step k:

$$q_m\left(m^{(k)}|m^{(<k)}\right) = \frac{e^{-\text{score}_{m^{(k)}}}}{\sum_{m'\in\mathcal{C}^{(k)}} e^{-\text{score}_{m'}}}, \text{ and } \forall m=(u,v)\in\mathcal{C}^{(k)}, \text{score}_m = \mathbf{v}_u^\top A \mathbf{v}_v, \quad (1)$$

where the symmetric matrix A is a learnable parameter and $\mathbf{v}_u$ denotes the embedding for node u in the encoding $\psi(x)$. At $k=1$ these embeddings are those provided by evoPF, and at each successive merge we remove two current leaves and create an embedding for their ancestor which becomes a new leaf (Figure 1b, Algorithm 11). The topological component $q_m(m^{(k)}|m^{(<k)})$ is therefore a probabilistic version of the $\arg\min$ over pairwise distances that is used in classical hierarchical reconstruction algorithms such as NJ. Furthermore, q_m relies on learnable scores rather than estimates of the evolutionary distances.

In order to ensure that each sampled merge $m^{(k)}$ leads to a longer cherry (sum of branch lengths) than those in all merges previously selected while $m^{(k)}$ was a possible choice—as decided in 3.2 to ensure that $p_{\psi(x)}$ is a proper probability distribution over phylogenies—we re-parameterize the two branch lengths $(\ell_i^{(k)}, \ell_j^{(k)}) \in \ell^{(k)}$ as their sum $s^{(k)} = \ell_i^{(k)} + \ell_j^{(k)}$ and ratio $r^{(k)} = \ell_i^{(k)}/s^{(k)}$. We update a matrix of constraints as we merge pairs indicating the minimal length that the sum of branch lengths at any new merge must attain for the sampled tree to be consistent with our canonical order. We then model the probability $q_s(s^{(k)}|m^{(\leq k)})$ as a Gamma distribution shifted by the constraint, and the probability $q_r(r^{(k)}|m^{(\leq k)})$ of the branch length ratio as a Beta distribution, *i.e.*,

$$\begin{aligned} q_s\left(s^{(k)}|m^{(\leq k)}\right) &= c_{m^{(k)}} + \text{Gamma}\left(\alpha_{\text{G}}^{(k)}, \lambda_{\text{G}}^{(k)}\right), \\ q_r\left(r^{(k)}|m^{(\leq k)}\right) &= \text{Beta}\left(\alpha_{\text{B}}^{(k)}, \beta_{\text{B}}^{(k)}\right), \end{aligned} \quad (2)$$

where $c_{m^{(k)}}$ is the current constraint on the sum of branch lengths for merge $m^{(k)}$ and whose parameters $(\alpha_{\text{G}}^{(k)}, \lambda_{\text{G}}^{(k)}, \alpha_{\text{B}}^{(k)}, \beta_{\text{B}}^{(k)})$ are produced by bilinear functions of the embedding of the two merged nodes $(u,v) = m^{(k)}$—similar to the score used in (1) for q_m. We use a symmetric form for $(\alpha_{\text{G}}^{(k)}, \lambda_{\text{G}}^{(k)})$ and an asymmetric one for $(\alpha_{\text{B}}^{(k)}, \beta_{\text{B}}^{(k)})$. The results of the bilinear forms are passed to a softplus function $\text{softplus}(x) = \log(1+e^x)$ to ensure their positivity, and we further add 1 for all parameters but $\lambda_{\text{G}}^{(k)}$, as they must be larger than 1. We finally obtain the joint probability $q_\ell(\ell^{(k)}|m^{(\leq k)})$ of the two branch lengths as $q_s(s^{(k)}|m^{(\leq k)})q_r(r^{(k)}|m^{(\leq k)})/s^{(k)}$, where the $1/s^{(k)}$ factor arises from the determinant of the Jacobian of the change of variables from $(s^{(k)}, r^{(k)})$ to $\ell^{(k)}$ (see Appendix A.2).

Algorithm 11 summarizes our procedure to evaluate the posterior $q_{\psi(x)}(\theta=(\tau,\ell)|x)$ of a phylogeny τ given set of sequences x. We use this procedure during the training phase to compute the loss function of Phyloformer 2, and at inference time when we want to evaluate the posterior probability of a phylogeny using a trained network. Of note, when evaluating $q_{\psi(x)}(\theta=(\tau,\ell)|x)$ the choice of one merge over several possibilities at each step is determined by the data, and does not depend on the network parameters. We therefore don't need to differentiate through discrete operations during training even though the loss evaluation depends on a succession of discrete choices. By contrast, sampling from $q_{\psi(x)}(\theta=(\tau,\ell)|x)$ does require a sequence of discrete decisions that depend on the network parameters but we never need to differentiate through this process.

Once the network is trained, sampling from the posterior given a set of sequences x (Algorithm 12) is very similar to the posterior evaluation procedure described above. The tree is iteratively built from evoPF embeddings of x by successively sampling merges from the softmin-parametrized conditional-merge probabilities (1). Similarly, we obtain branch-lengths at a step k by sampling their sum from the shifted Gamma distribution and their ratio from the Beta distribution, whose parameters are obtained from evoPF embeddings (2). We can also use the trained network to build a greedy approximation of the maximum *a posteriori* (MAP) estimator (thereafter referred to as greedy MAP), by using the mode of the estimated distributions, *i.e.*, replacing the softmin in (1) with $\hat{m}^{(k)}_{\text{greedyMAP}} = \arg\min_{m\in\mathcal{C}^{(k)}} \text{score}_m$ to sample the most probable merge at each step k,

and using the Gamma and Beta modes for branch lengths, *i.e.* $\hat{s}^{(k)}_{\text{greedyMAP}} = (\alpha^{(k)}_{\text{G}} - 1)/\lambda^{(k)}_{\text{G}}$ and $\hat{r}^{(k)}_{\text{greedyMAP}} = (\alpha^{(k)}_{\text{B}} - 1)/(\alpha^{(k)}_{\text{B}} + \beta^{(k)}_{\text{B}} - 2)$.

Of note, the resulting q_ψ has limited expressivity for two reasons. First, the relative probabilities for merging each of the currently available pairs are computed based on the initial embeddings (*i.e.*, the embeddings of those nodes that have not been chosen thus far are not updated as the recursion proceeds). In the true posterior, on the other hand, the relative posterior probability of being the next smallest cherry for a given pair of nodes in principle depends on previous merges. An update of the vector of embeddings at each step of the recursion could be implemented in a future version, but would be considerably more computationally intensive. Second, we model branch lengths posteriors with specific parametric distributions (Gamma and Beta) whereas the true posterior has no reason to match these analytical forms in general.

4 Experiments

4.1 Faster and more accurate point estimates of phylogenies under a tractable model

We trained Phyloformer 2 (PF2) over a large dataset simulated under similar priors to Nesterenko et al. [29]. This dataset contains $\approx 1.3 \cdot 10^6$ 50-taxa trees simulated under a rescaled birth-death process—effectively corresponding to the prior $p(\theta)$—and, for each tree θ, an MSA x simulated under the LG+G8 probabilistic model of evolution $p(x|\theta)$ (LG substitution matrix of Le & Gascuel [24] combined with gamma evolution rates [47], also see Appendix A.3.1).

The likelihood under this model is tractable, making it a favorable setting for maximum-likelihood methods—FastTree [31] and IQTREE [27] in this experiment. We also include FastME [25], a much faster but less accurate method that only requires to compute likelihoods of branches between pairs of sequences.

The PF2 model was then fine-tuned on tree/MSA pairs with sizes ranging from 10 to 170 taxa, simulated under the same priors as the main training set (see Appendices A.3.1 and A.4). This fine-tuning step is necessary to avoid some overfitting to the number of taxa (see also Figure 8). All MSAs have the same sequence length since overfitting to that metric seems to be less of an issue (see Figure 10).

We then inferred greedy MAP trees (see Section 3.3) on a test set of sequence alignments simulated under the same priors as the training set, with 10 to 200 taxa, obtained from Nesterenko et al. [29]. Figure 2a compares all tested methods using the normalized Robinson-Foulds distance between simulated and inferred tree—a classical metric to compare topologies counting the proportion of branches that are present in only one of two phylogenies [35]. PF2 is a marked improvement over PF, with better topological accuracy across the whole range of tree/MSA size. For trees with 10 to 130 leaves PF2 matches IQTree and FastTree in terms of topological accuracy, both state-of-the-art maximum-likelihood tree reconstruction methods. By leveraging its posterior estimation, PF2 is able to outperform both IQTree and FastTree on trees up to 150 tips on the tractable LG+GC evolution model. To do so, we sampled 100 trees from the estimated posterior and summarized them with the greedy-consensus (GC) tree as implemented in the `dendropy` [38] python package.

In addition to being as topologically accurate, PF2 produces single-tree estimates much faster than maximum-likelihood estimators: by one order of magnitude compared to FastTree, two compared to IQTREE. For trees with more than 100 leaves PF2 is faster than PF. Although PF2 is still memory intensive compared to maximum-likelihood approaches, it scales better than PF allowing PF2 to infer larger trees, faster, despite having 1000 times more parameters than PF (see Figure 2c).

In order to disentangle the effects of the evoPF module and the BayesNJ loss, we trained a PF2_{MAE} model using evoPF but replacing the BayesNJ module by a mean absolute error (MAE) loss on pairwise distances, as was done with PF. After a similar number of training steps as PF, we inferred distance matrices from the PF test MSAs, and trees from these matrices using FastME [25]. PF2_{MAE} yields trees with the same topological accuracy as PF, measured by the RF distance (See Figure 6). This seems to indicate that most of the topological accuracy gain shown in Figure 2a is due to the BayesNJ loss. Both PF and PF2_{MAE} yield similar Kuhner-Felsenstein [KF, 21] distances—a metric similar to RF but weighting each inconsistent branch by the square of its length, and each consistent pair by the squared difference of lengths.

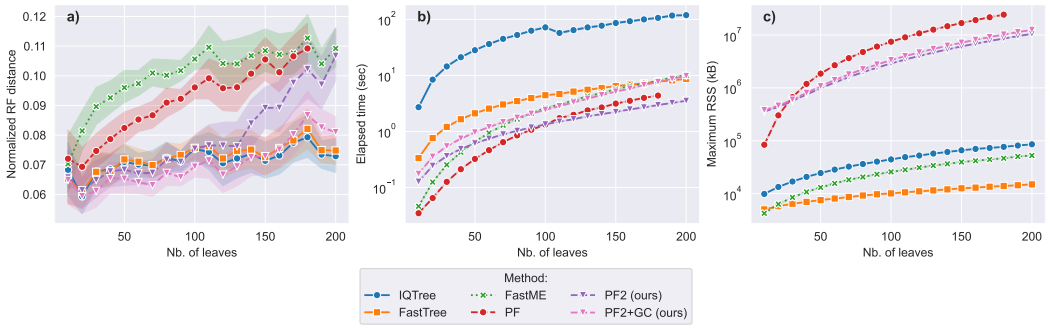

Figure 2: **(a)** Topological performance for Phyloformer 2, measured by the normalized Robinson-Foulds distance. The alignments for which trees were inferred were taken from the original Phyloformer paper [29] and were simulated under the LG+GC sequence model. **(b)** Runtime and **(c)** Memory usage for Phyloformer 2. The same GPU model as the original Phyloformer study was used to run Phyloformer 2 inference. Results for other methods are reported from the original Phyloformer paper [29]. Both single-tree estimates using the greedy-MAP and 100-sample posterior summary trees using the greedy-consensus tree are shown for PF2 (respectively labeled PF2 and PF2+GC)

Finally, similar experiments on empirical data are not possible in phylogenetics where the real tree is never observed but we follow Nesterenko et al. [29] and plot for several datasets the distribution of RF distances between the inferred tree for MSAs of individual gene families and an estimate of the species tree obtained from a concatenate of gene alignments (Figure 3). Since all methods perform inference under the LG model, we expect them to behave similarly—*i.e.*, either all well for datasets where inference under LG leads to a good estimate, or all poorly otherwise. This is what we observe on all the Zhou et al. [50] datasets (Figure 3b). For four out of five datasets (ShenA9 excepted), PF2 greedy-MAP estimates are as accurate as all other methods, and summarizing posterior samples (with the GC or the MCC) yields more accurate tree estimates than all other methods. This effect is further accentuated in the cyanobacteria dataset (Figure 3a) where both PF2 greedy-MAP and posterior summary trees are more accurate than all other methods.

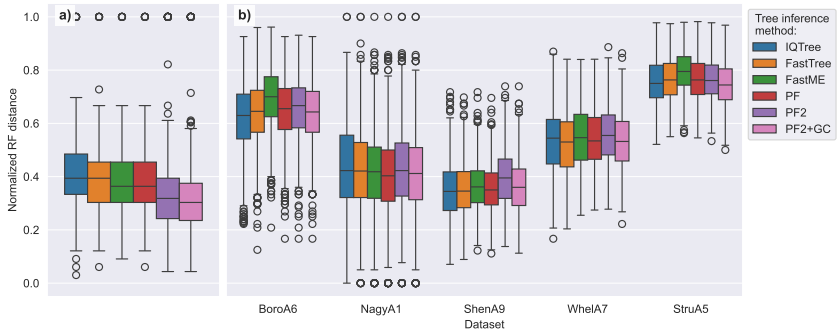

Figure 3: Topological reconstruction accuracy on empirical datasets. The topological accuracy is measured as the normalized Robinson-Foulds distance between the inferred gene tree, and a reference species tree inferred on the concatenation of all gene MSAs using maximum-likelihood estimation. This experiment is done on a cyanobacteria dataset introduced in Szöllősi et al. [40] (left), and a set of diverse protein datasets introduced in Zhou et al. [50].This follows the same experimental setup as Nesterenko et al. [29]. Both greedy-MAP trees (PF2) and greedy-consensus (PF2+GC) posterior summary trees over 10^3 samples are shown for our method.

4.2 Increased advantage under intractable probabilistic models of evolution

The main motivation for NPE is to allow well-specified inference under models whose likelihood is intractable. We assess the performance of PF2 in this setting on the same benchmark used in Nesterenko et al. [29], that includes the Cherry model [32], allowing for dependencies between

evolution at distinct positions in the sequences, and the SelReg model [5] allowing for heterogeneous amino acid distributions between positions. Of note, a mistake in the generation of the Cherry dataset in [29] inflates the amount of dependency to an unrealistic level (see Appendix A.5) but we keep it as it still provides a proof of concept and to avoid the computational burden of re-training PF on a new dataset. We further add a new dataset with sequences evolved under a Potts model [7] capturing co-evolutions in the response regulator receiver domain (PF00072) protein family (see Appendix A.3.1).

Under all three models, a fine-tuned PF2 model significantly outperforms the equivalent PF model in RF distance and is comparable to PF in terms of KF score (see Figure 7). For all models and metrics, PF2 (and PF where applicable) outperforms all other reference methods.

4.3 Estimating the phylogenetic posterior distribution

A major advance of PF2 compared to existing likelihood-free phylogenetic inference methods including PF is its ability to represent entire posterior distributions—as opposed to point estimates—over full phylogenies—as opposed to quartets. As shown in Figures 2 and 3 summarizing posterior samples yields higher quality estimates than the greedy-MAP approximation. We assess the quality of this estimation by comparing against samples obtained from a long MCMC run of RevBayes [15], a standard tool for Bayesian phylogenetic inference. We ran 10 parallel MCMC runs on a single 50 sequence alignment for $50,000$ iterations and $5,000$ burn-in iterations. We used a uniform prior on tree topologies and an Exponential distribution with $\lambda = 10$ for branch lengths, and LG+G8 as the sequence evolution model. For a fairer comparison, we trained an instance of PF2 on tree/MSA pairs simulated under these priors before sampling from the posterior.

A common way to compare distributions of topologies is through the branches present in sampled trees. Every branch in a phylogeny defines a bipartition of the leaves, making them comparable across all possible trees and providing a softer metric than, *e.g.*, the frequencies of full topologies. Figure 4a shows that RevBayes produces a hard posterior distribution where most branches appear in either all or none of the sampled topologies. PF2 is conservative in its estimate of uncertainty (less certain than it could afford to be), which is expected, given that NPE is trying to minimize the KL divergence from the true distribution to the proposal, which induces a mode-covering behavior, i.e. covers all regions of non-negligible probability mass under the true posterior, at the expense of covering other regions with close to zero posterior probability mass. PF2 provides a softer posterior but a large agreement with RevBayes, as most branches sampled in all RevBayes trees have a frequency larger than 0.6 in PF2, and those not sampled have a frequency lower than 0.1. This is also consistent with our observation that the greedy MAP version of PF2 provides good point estimates, since sequentially sampling the highest probability merges is very likely to select the right branches.

We further investigated the calibration of PF2's posterior probabilities on 248 simulations of trees and alignments with 50 sequences. We computed branch posterior probabilities over $10,000$ trees sampled from PF2 for each alignment and compared these probabilities to the frequency with which they correspond to branches in the true simulated tree. Figure 4b shows that PF2 is generally well-calibrated, with a tendency to slightly overestimate the support of low probability bipartitions, and to underestimate the support of high probability bipartitions. These experiments further suggest that the particular analytical distributions we chose to use to describe branch lengths do not jeopardize PF2's ability to estimate the posterior. Future investigation will enable us to assess whether another parametrization may improve upon our current choice. Overall, PF2 appears to provide estimates of posterior tree distributions that should be useful in practice. Furthermore it does so several orders of magnitude faster than MCMC methods like RevBayes by leveraging GPU parallelism (see Appendix A.6).

5 Conclusion

We introduced Phyloformer 2, a phylogenetic neural posterior estimator combining two novel components: evoPF, an expressive and efficient encoding for aligned sequences, and BayesNJ, a factorization of the tree probability over successive merges. Phyloformer 2 can provide MAP estimates that match or outperform existing likelihood-based and likelihood-free phylogenetic reconstruction methods in topological accuracy while running at least ten times faster. By summarizing posterior samples, PF2 can yield even better trees while still being as fast as the fastest maximum-likelihood

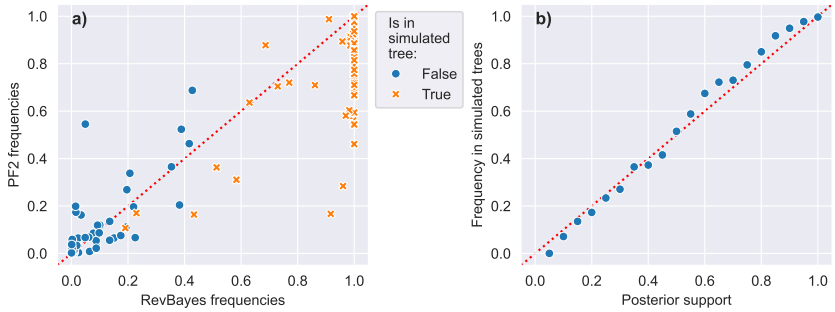

Figure 4: **a)**: Comparison of split frequencies over samples from the posterior of a single 50 sequences MSA, between RevBayes MCMC (x-axis) and PF2 (y-axis). The orange cross marker indicates splits that are present in the tree along which the MSA used for sampling has been simulated. For fair comparison, this instance of PF2 was trained on data simulated under an Uniform topology prior with an exponential distribution of branch lengths. This prior is readily available in RevBayes unlike our custom rescaled Birth-Death prior used to train other PF2 instances.**b)**: simulation-based calibration comparing branch supports, *i.e.*, their frequency in samples from the posterior estimate given by a trained PF2 (x-axis) to the frequency with which splits with a given PF2 support are true, i.e. found in the tree used for simulation (y-axis) Support values are binned by steps of 0.05.

method. The magnitude of this improvement under models with intractable likelihoods suggests that simplifying assumptions such as homogeneity or site independence that are typically necessary to run likelihood-based inference methods can have a major effect on the quality of phylogenetic reconstruction. The possibility introduced by PF2 of doing inference under more realistic models could therefore be critical in this field. Furthermore, our experiments show that PF2 approximates posterior distributions in an amortized fashion, with a good calibration, so that branch posterior probabilities provide a useful estimate of their accuracy. Summarizing posterior samples into single tree-estimates also yield higher quality trees than single greedy-MAP estimates, at the cost of some execution time. This execution time remains orders of magnitude lower than existing MCMC methods.

One major limitation of Phyloformer 2 is its scalability, preventing its usage on much more than 250 sequences on 16Gb of VRAM. Even though the new inference possibilities under realistic models offered by PF2 are already very valuable for middle-size phylogenies, future work should explore more efficient encoders [46, 44] or existing heuristics to build larger trees [45, 16].

In addition, like other NPE methods, Phyloformer 2 could currently produce poor estimates with no warning when presented with inputs far from its training data, which can happen even when doing well-specified inference—on data generated under the same $p(x, \theta)$ used for training—if some parameter areas have low probability under the prior. Likelihood-based methods are immune to this specific issue, as they do not require a learning phase. By contrast, both likelihood-based and likelihood-free methods perform model-based phylogenetic inference, and as such Phyloformer 2 is not immune to model mis-specification. Both of these issues could be mitigated by providing an assessment of the uncertainty of its prediction [8, 22]

In the future, we expect Phyloformer 2 to reveal most of its potential by allowing inference under more realistic and complex scenarios [23]. Other promising directions are to handle unaligned sequences, and inference under broader probabilistic models of evolution embedding phylogenies such as population dynamics and co-evolution.

Acknowledgments

This work was funded by the Agence Nationale de la Recherche (ANR-20-CE45-0017). It was granted access to the HPC/AI resources of IDRIS under the allocation AD010714229R2 made by GENCI. The taxon silhouettes in Fig. 1 are modified from public domain images in the PhyloPic database. The authors also thank Vincent Garot and Evan Gorstein.

Code and Data Availability

All the code used in this study is available at the following repository: gitlab.in2p3.fr/deelogeny/wp1/phyloformer-2

References

[1] Aminov, R. I. and Mackie, R. I. Evolution and ecology of antibiotic resistance genes. *FEMS Microbiology Letters*, 271(2):147–161, 06 2007. ISSN 0378-1097. doi: 10.1111/j.1574-6968.2007.00757.x. URL https://doi.org/10.1111/j.1574-6968.2007.00757.x.

[2] Bouchard-Côté, A., Sankararaman, S., and Jordan, M. I. Phylogenetic inference via sequential monte carlo. *Systematic Biology*, 61(4):579–593, 01 2012. ISSN 1063-5157. doi: 10.1093/sysbio/syr131. URL https://doi.org/10.1093/sysbio/syr131.

[3] Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. doi: 10.1073/pnas.1912789117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1912789117.

[4] Duan, C., Zang, Z., Li, S., Xu, Y., and Li, S. Z. Phylogen: Language model-enhanced phylogenetic inference via graph structure generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GxvDsFArxY.

[5] Duchemin, L., Lanore, V., Veber, P., and Boussau, B. Evaluation of methods to detect shifts in directional selection at the genome scale. *Molecular Biology and Evolution*, 40(2):msac247, 12 2022. ISSN 1537-1719. doi: 10.1093/molbev/msac247. URL https://doi.org/10.1093/molbev/msac247.

[6] Felsenstein, J. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376, 1981.

[7] Figliuzzi, M., Barrat-Charlaix, P., and Weigt, M. How Pairwise Coevolutionary Models Capture the Collective Residue Variability in Proteins? *Molecular Biology and Evolution*, 35 (4):1018–1027, April 2018. ISSN 0737-4038. doi: 10.1093/molbev/msy007. URL https://doi.org/10.1093/molbev/msy007.

[8] Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

[9] Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2004.

[10] Greenberg, D., Nonnenmacher, M., and Macke, J. Automatic posterior transformation for likelihood-free inference. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2404–2414. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/greenberg19a.html.

[11] Grosshauser, M., Zaharias, P., and T., W. Re-evaluating deep neural networks for phylogeny estimation: the issue of taxon sampling. *Recomb2021*, 2021.

[12] Hadfield, J., Megill, C., Bell, S. M., Huddleston, J., Potter, B., Callender, C., Sagulenko, P., Bedford, T., and Neher, R. A. Nextstrain: real-time tracking of pathogen evolution. *Bioinformatics*, 34(23):4121–4123, December 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty407. URL https://doi.org/10.1093/bioinformatics/bty407.

[13] Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *CoRR*, abs/1912.12180, 2019. URL http://arxiv.org/abs/1912.12180.

[14] Huelsenbeck, J. P. and Ronquist, F. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 08 2001. ISSN 1367-4803. doi: 10.1093/bioinformatics/17.8.754. URL https://doi.org/10.1093/bioinformatics/17.8.754.

[15] Höhna, S., Landis, M. J., Heath, T. A., Boussau, B., Lartillot, N., Moore, B. R., Huelsenbeck, J. P., and Ronquist, F. Revbayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic Biology*, 65(4):726–736, 05 2016. ISSN 1063-5157. doi: 10.1093/sysbio/syw021. URL https://doi.org/10.1093/sysbio/syw021.

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

[17] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, Aug 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

[18] Kapli, P., Yang, Z., and Telford, M. J. Phylogenetic tree building in the genomic age. *Nature Reviews Genetics*, 21(7):428–444, July 2020. ISSN 1471-0064. doi: 10.1038/s41576-020-0233-0. URL http://www.nature.com/articles/s41576-020-0233-0. Publisher: Nature Publishing Group.

[19] Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[20] Koptagel, H., Kviman, O., Melin, H., Safinianaini, N., and Lagergren, J. Vaiphy: a variational inference based algorithm for phylogeny. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14758–14770. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/5e956fef0946dc1e39760f94b78045fe-Paper-Conference.pdf.

[21] Kuhner, M. K. and Felsenstein, J. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol*, 11(3):459–468, May 1994.

[22] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[23] Latrille, T., Lanore, V., and Lartillot, N. Inferring long-term effective population size with mutation–selection models. *Molecular Biology and Evolution*, 38(10):4573–4587, 06 2021. ISSN 1537-1719. doi: 10.1093/molbev/msab160. URL https://doi.org/10.1093/molbev/msab160.

[24] Le, S. Q. and Gascuel, O. An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution*, 25(7):1307–1320, July 2008. ISSN 0737-4038. doi: 10.1093/molbev/msn067. URL https://doi.org/10.1093/molbev/msn067.

[25] Lefort, V., Desper, R., and Gascuel, O. Fastme 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. *Molecular biology and evolution*, 32(10):2798–2800, 2015.

[26] Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. Benchmarking simulation-based inference. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 343–351. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/lueckmann21a.html.

[27] Minh, B. Q., Schmidt, H. A., Chernomor, O., Schrempf, D., Woodhams, M. D., von Haeseler, A., and Lanfear, R. Iq-tree 2: New models and efficient methods for phylogenetic inference in the genomic era. *Molecular Biology and Evolution*, 37(5):1530–1534, 02 2020. ISSN 0737-4038. doi: 10.1093/molbev/msaa015. URL https://doi.org/10.1093/molbev/msaa015.

[28] Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., and Bateman, A. Pfam: The protein families database in 2021. 49(D1):D412–D419. ISSN 1362-4962. doi: 10.1093/nar/gkaa913. URL https://europepmc.org/articles/PMC7779014.

[29] Nesterenko, L., Blassel, L., Veber, P., Boussau, B., and Jacob, L. Phyloformer: Fast, accurate, and versatile phylogenetic reconstruction with deep neural networks. *Molecular Biology and Evolution*, 42(4):msaf051, 03 2025. ISSN 1537-1719. doi: 10.1093/molbev/msaf051. URL https://doi.org/10.1093/molbev/msaf051.

[30] Pagnani, A. and Barrat-Charlaix, P. Generative continuous time model reveals epistatic signatures in protein evolution. ISSN 2692-8205. URL https://www.biorxiv.org/content/10.1101/2025.09.17.676821v1.

[31] Price, M. N., Dehal, P. S., and Arkin, A. P. Fasttree 2 – approximately maximum-likelihood trees for large alignments. *PLOS ONE*, 5(3):1–10, 03 2010. doi: 10.1371/journal.pone.0009490. URL https://doi.org/10.1371/journal.pone.0009490.

[32] Prillo, S., Deng, Y., Boyeau, P., Li, X., Chen, P.-Y., and Song, Y. S. Cherryml: scalable maximum likelihood estimation of phylogenetic models. *Nature Methods*, 20(8):1232–1236, Aug 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01917-9. URL https://doi.org/10.1038/s41592-023-01917-9.

[33] Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE transactions on neural networks and learning systems*, 33(4):1452–1466, 2020.

[34] Rao, R. M., Liu, J., Verkuil, R., Meier, J., Canny, J., Abbeel, P., Sercu, T., and Rives, A. Msa transformer. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8844–8856. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/rao21a.html.

[35] Robinson, D. and Foulds, L. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981. ISSN 0025-5564. doi: https://doi.org/10.1016/0025-5564(81)90043-2. URL https://www.sciencedirect.com/science/article/pii/0025556481900432.

[36] Rosset, L., Netti, R., Muntoni, A. P., Weigt, M., and Zamponi, F. adabmDCA 2.0—A Flexible but Easy-to-Use Package for Direct Coupling Analysis. In Khan, S. M. and Pazos, F. (eds.), *Protein Evolution: Methods and Protocols*, pp. 83–104. Springer US. ISBN 978-1-0716-4828-5. doi: 10.1007/978-1-0716-4828-5_6. URL https://doi.org/10.1007/978-1-0716-4828-5_6.

[37] Saitou, N. and Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[38] Sukumaran, J. and Holder, M. T. Dendropy: a python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571, 06 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq228. URL https://doi.org/10.1093/bioinformatics/btq228.

[39] Suvorov, A., Hochuli, J., and Schrider, D. R. Accurate Inference of Tree Topologies from Multiple Sequence Alignments Using Deep Learning. *Systematic Biology*, 69(2):221–233, 09 2019. ISSN 1063-5157. doi: 10.1093/sysbio/syz060. URL https://doi.org/10.1093/sysbio/syz060.

[40] Szöllősi, G. J., Rosikiewicz, W., Boussau, B., Tannier, E., and Daubin, V. Data from: Efficient exploration of the space of reconciled gene trees. URL https://datadryad.org/stash/dataset/doi:10.5061/dryad.pv6df.

[41] Tang, X., Zepeda-Nuñez, L., Yang, S., Zhao, Z., and Solís-Lemus, C. Novel symmetry-preserving neural network model for phylogenetic inference. *Bioinformatics Advances*, 4(1):vbae022, 02 2024. ISSN 2635-0041. doi: 10.1093/bioadv/vbae022. URL https://doi.org/10.1093/bioadv/vbae022.

[42] Telford, M. J., Wise, M. J., and Gowri-Shankar, V. Consideration of RNA Secondary Structure Significantly Improves Likelihood-Based Estimates of Phylogeny: Examples from the Bilateria. *Molecular Biology and Evolution*, 22(4):1129–1136, April 2005. ISSN 0737-4038. doi: 10.1093/molbev/msi099. URL https://doi.org/10.1093/molbev/msi099.

[43] Trost, J., Haag, J., Höhler, D., Jacob, L., Stamatakis, A., and Boussau, B. Simulations of sequence evolution: how (un) realistic they are and why. *Molecular Biology and Evolution*, 41 (1):msad277, 2024.

[44] Wang, Y., Lu, J., Jaitly, N., Susskind, J., and Bautista, M. A. Simplefold: Folding proteins is simpler than you think, 2025. URL https://arxiv.org/abs/2509.18480.

[45] Warnow, T. Supertree construction: Opportunities and challenges, 2018. URL https://arxiv.org/abs/1805.03530.

[46] Wohlwend, J., Reveiz, M., McPartlon, M., Feldmann, A., Jin, W., and Barzilay, R. Minifold: Simple, fast, and accurate protein structure prediction. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=1p9hQTbjgo. Featured Certification.

[47] Yang, Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *Journal of Molecular Evolution*, 39(3):306–14, September 1994.

[48] Zhang, C. and IV, F. A. M. Variational bayesian phylogenetic inference. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJVmjjR9FX.

[49] Zhou, M. Y., Yan, Z., Layne, E., Malkin, N., Zhang, D., Jain, M., Blanchette, M., and Bengio, Y. PhyloGFN: Phylogenetic inference with generative flow networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hB7SlfEmze.

[50] Zhou, X., Shen, X.-X., Hittinger, C. T., and Rokas, A. Evaluating Fast Maximum Likelihood-Based Phylogenetic Programs Using Empirical Phylogenomic Data Sets. 35(2):486–503. ISSN 0737-4038. doi: 10.1093/molbev/msx302. URL https://doi.org/10.1093/molbev/msx302.

[51] Zou, Z., Zhang, H., Guan, Y., and Zhang, J. Deep residual neural networks resolve quartet molecular phylogenies. *Molecular biology and evolution*, 37(5):1495–1507, 2020.

[52] Álvarez Carretero, S., Tamuri, A. U., Battini, M., Nascimento, F. F., Carlisle, E., Asher, R. J., Yang, Z., Donoghue, P. C. J., and dos Reis, M. A species-level timeline of mammal evolution integrating phylogenomic data. *Nature*, 602(7896):263–267, February 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04341-1. URL https://www.nature.com/articles/s41586-021-04341-1. Number: 7896 Publisher: Nature Publishing Group.

A Technical Appendices and Supplementary Material

A.1 Model architecture

Notation: i, j and k denote sequence indices, t and l denote residue indices. p and r denote pair indices in $\mathcal{P}$ where $\mathcal{P} = \{\{i, j\} :, 1 \leq i < j \leq N\}$.
Capitalized functions (e.g. Linear) are parametrized and learnable, whereas lowercase functions (e.g. sigmoid) are not.

The evoPF module takes as input a multiple sequence alignments $\{\mathbf{x}_{it}\}$, where $\mathbf{x}_{it}$ is the one-hot encoded vector of the tth character of sequence i. EvoPF produces an initial set of sequence embeddings $\{\mathbf{s}_{it}\}$ where each residue is represented by an embedding of size $c_s = 128$, and a set of sequence pair embeddings $\{\mathbf{z}_p\}$ of fixed size $c_z = 256$ for each unordered pair of sequences (i, j). These embeddings are refined through the evoPF MSA and pair stacks respectively. The final block of the evoPF module returns a set of fixed-size c_s sequence embeddings $\{\mathbf{v}_i\}$, and the set of pair-embeddings $\{\mathbf{z}_p\}$ of fixed-size c_z.

Algorithm 1 The evoPF module

function EVOPF ($N_{blocs} = 12$, $\{\mathbf{x}_{it}\}$)
  $\{\mathbf{s}_{it}\} \leftarrow$ MSAEmbedder($\{\mathbf{x}_{it}\}$)
  $\{\mathbf{z}_p\} \leftarrow$ PairEmbedder($\{\mathbf{x}_{it}\}$)
  for $b \in [1, \ldots, N_{blocs}]$ **do**
    $\{\mathbf{s}_{it}\} \leftarrow \{\mathbf{s}_{it}\} +$ MSAColAttentionWithPairBias($\{\mathbf{s}_{it}\}, \{\mathbf{z}_p\}$)
    $\{\mathbf{s}_{it}\} \leftarrow \{\mathbf{s}_{it}\} +$ MSARowAttention($\{\mathbf{s}_{it}\}$)
    $\{\mathbf{s}_{it}\} \leftarrow \{\mathbf{s}_{it}\} +$ MSATransition($\{\mathbf{s}_{it}\}$)
    $\{\mathbf{z}_p\} \leftarrow \{\mathbf{z}_p\} +$ OuterProductMean($\{\mathbf{s}_{it}\}$)
    $\{\mathbf{z}_p\} \leftarrow \{\mathbf{z}_p\} +$ PairAttention($\{\mathbf{z}_p\}$)
    $\{\mathbf{z}_p\} \leftarrow \{\mathbf{z}_p\} +$ PairTransition($\{\mathbf{z}_p\}$)
  end for
  $\mathbf{z}_p \leftarrow$ Linear($\mathbf{z}_p$)
  $\mathbf{v}_i \leftarrow$ mean$_t$($\mathbf{s}_{it}$)
  return $\{\mathbf{z}_p\}, \{\mathbf{v}_i\}$ $\mathbf{z}_p \in \mathbb{R}^{c_z}$, $\mathbf{v}_i \in \mathbb{R}^{c_s}$
end function

A.1.1 Embedding modules

Algorithm 2 MSA Embedding module

function MSAEMBEDDER ($\{\mathbf{x}_{it}\}$, $c_s = 128$)
  $\mathbf{a}_{it} \leftarrow$ Conv2D($\mathbf{x}_{it}$) $a_{it} \in \mathbb{R}^{c_s}$
  $\mathbf{s}_{it} \leftarrow$ reLU($\mathbf{a}_{it}$)
  return $\{\mathbf{s}_{it}\}$
end function

Algorithm 3 Pair Embedding module

function PAIREMBEDDER ($\{\mathbf{x}_{it}\}$, $c_z = 256$)
  $a_{it} \leftarrow$ reLU(Conv2D($\mathbf{x}_{it}$)) $a_{it} \in \mathbb{R}^{c_z}$
  $\mathbf{z}_p \leftarrow$ mean$_t$($a_{it} + a_{jt}$) $p = \{i, j\} \in \mathcal{P}$
  return $\{\mathbf{z}_p\}$
end function

A.1.2 The evoPF MSA stack

Algorithm 4 MSA Stack - Column-wise pair-biased gated self-attention

function MSACOLATTENTIONWITHPAIRBIAS ($\{\mathbf{s}_{it}\}, \{\mathbf{z}_p\}, N_{head} = 4, c = c_s/N_{head}$)
- $\mathbf{s}_{it} \leftarrow \text{LayerNorm}(\mathbf{s}_{it})$
- $\mathbf{q}^h_{it}, \mathbf{k}^h_{it}, \mathbf{v}^h_{it}, \leftarrow \text{LinearNoBias}(\mathbf{s}_{it})$ — $\mathbf{q}^h_{it}, \mathbf{k}^h_{it}, \mathbf{v}^h_{it} \in \mathbb{R}^c, 1 \leq h \leq N_{head}$
- $b^h_{ij} \leftarrow \text{LinearNoBias}(\mathbf{z}_p)$ — $p = \{i, j\} \in \mathcal{P}$
- $\mathbf{g}^h_{it} \leftarrow \text{sigmoid}(\text{Linear}(\mathbf{s}_{it}))$ — $\mathbf{g}_{it} \in \mathbb{R}^c$
- $\mathbf{a}^h_{ijt} \leftarrow \text{softmax}_j \left(\frac{1}{\sqrt{c}} \mathbf{q}^{h\top}_{it} \mathbf{k}^h_{jt} + b^h_{ij} \right)$
- $\mathbf{o}^h_{it} \leftarrow \mathbf{g}^h_{it} \odot \sum_j \mathbf{a}^h_{ijt} \mathbf{v}^h_{jt}$
- $\tilde{\mathbf{s}}_{it} \leftarrow \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{it})\right)$ — $\tilde{\mathbf{s}}_{it} \in \mathbb{R}^{c_s}$
- **return** $\{\tilde{\mathbf{s}}_{it}\}$

end function

Algorithm 5 MSA Stack - Row-wise gated self-attention

function MSAROWATTENTION ($\{\mathbf{s}_{it}\}, N_{head} = 4, c = c_s/N_{head}$)
- $\mathbf{s}_{it} \leftarrow \text{LayerNorm}(\mathbf{s}_{it})$
- $\mathbf{q}^h_{it}, \mathbf{k}^h_{it}, \mathbf{v}^h_{it}, \leftarrow \text{LinearNoBias}(\mathbf{s}_{it})$ — $\mathbf{q}^h_{it}, \mathbf{k}^h_{it}, \mathbf{v}^h_{it} \in \mathbb{R}^c, 1 \leq h \leq N_{head}$
- $\mathbf{g}^h_{it} \leftarrow \text{sigmoid}(\text{Linear}(\mathbf{s}_{it}))$ — $\mathbf{g}_{it} \in \mathbb{R}^c$
- $\mathbf{a}^h_{itl} \leftarrow \text{softmax}_l \left(\frac{1}{\sqrt{c}} \mathbf{q}^{h\top}_{it} \mathbf{k}^h_{il} \right)$
- $\mathbf{o}^h_{it} \leftarrow \mathbf{g}^h_{it} \odot \sum_l \mathbf{a}^h_{itl} \mathbf{v}^h_{il}$
- $\tilde{\mathbf{s}}_{it} \leftarrow \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{it})\right)$ — $\tilde{\mathbf{s}}_{it} \in \mathbb{R}^{c_s}$
- **return** $\{\tilde{\mathbf{s}}_{it}\}$

end function

Algorithm 6 MSA Stack - Transition

function MSATRANSITION ($\{\mathbf{s}_{it}\}, n = 4$)
- $\mathbf{s}_{it} \leftarrow \text{LayerNorm}(\mathbf{s}_{it})$
- $\mathbf{a}_{it} \leftarrow \text{Linear}(\mathbf{s}_{it})$ — $\mathbf{a}_{it} \in \mathbb{R}^{nc_s}$
- $\mathbf{s}_{it} \leftarrow \text{Linear}(\text{reLU}(\mathbf{a}_{it}))$
- **return** $\{\mathbf{s}_{it}\}$

end function

Algorithm 7 Communication - Outer product mean

function OUTERPRODUCTMEAN ($\{\mathbf{s}_{it}\}, c = 32$)
- $\mathbf{s}_{it} \leftarrow \text{LayerNorm}(\mathbf{s}_{it})$
- $\mathbf{a}_{it} \leftarrow \text{Linear}(\mathbf{s}_{it})$ — $\mathbf{a}_{it} \in \mathbb{R}^{c_s}$
- $\mathbf{o}_p \leftarrow \text{flatten}(\text{mean}_t(\mathbf{a}_{it} \otimes \mathbf{a}_{jt}))$ — $p = \{i, j\} \in \mathcal{P}, \mathbf{o}_p \in \mathbb{R}^{c^2}$
- $\mathbf{z}_p \leftarrow \text{SymmetricLinear}(\mathbf{o}_p)$ — $\mathbf{z}_p \in \mathbb{R}^{c_z}$
- **return** $\{\mathbf{z}_p\}$

end function

A.1.3 The evoPF pair stack

Algorithm 8 Pair Stack - Gated self-attention

function PAIRATTENTION ($\{\mathbf{z}_p\}, N_{head} = 4, c = c_z/N_{head}$)
  $\mathbf{z}_p \leftarrow \text{LayerNorm}(\mathbf{z}_p)$
  $\mathbf{q}_p^h, \mathbf{k}_p^h, \mathbf{v}_p^h, \leftarrow \text{LinearNoBias}(\mathbf{z}_p)$ $\quad \mathbf{q}_p^h, \mathbf{k}_p^h, \mathbf{v}_p^h \in \mathbb{R}^c, 1 \leq h \leq N_{head}$
  $\mathbf{g}_p^h \leftarrow \text{sigmoid}(\text{Linear}(\mathbf{z}_p))$ $\quad \mathbf{g}_p \in \mathbb{R}^c$
  $\mathbf{a}_{pr}^h \leftarrow \text{softmax}_r \left(\frac{1}{\sqrt{c}} \mathbf{q}_p^{h\top} \mathbf{k}_r^h \right)$ $\quad p, r \in \mathcal{P}$
  $\mathbf{o}_p^h \leftarrow \mathbf{g}_p^h \odot \sum_r \mathbf{a}_{pr}^h \mathbf{v}_r^h$
  $\tilde{\mathbf{z}}_p \leftarrow \text{Linear}\left(\text{concat}_h(\mathbf{o}_p^h)\right)$ $\quad \tilde{\mathbf{z}}_p \in \mathbb{R}^{c_z}$
  return $\{\tilde{\mathbf{z}}_p\}$
end function

Algorithm 9 Pair Stack - Transition

function PAIRTRANSITION ($\{\mathbf{z}_p\}, n = 4$)
  $\mathbf{z}_p \leftarrow \text{LayerNorm}(\mathbf{z}_p)$
  $\mathbf{a}_p \leftarrow \text{Linear}(\mathbf{z}_p)$ $\quad \mathbf{a}_p \in \mathbb{R}^{nc_z}$
  $\mathbf{z}_p \leftarrow \text{Linear}(\text{reLU}(\mathbf{a}_p))$
  return $\{\mathbf{z}_p\}$
end function

A.1.4 BayesNJ algorithms

Notation reminder:

- $\mathbf{v}_i$: Embedding of node i.
- $\mathcal{S}_{(k)}$: Set of active "mergeable" nodes at step k.
- $\mathcal{C}_{(k)}$: Set of candidate merges (i.e. node pairs) at step k.
- $m^{(k)}$: Pair of indices (i, j) of merged nodes at step k.
- $c_m^{(k)}$: Constraint on cherry-length for merge m at step k.
- ℓ_i: Length of branch from node i to its parent.

Algorithm 10 Helper functions for BAYESNJ

function GETDISTRIBUTIONS (Embeddings of merged nodes $\mathbf{v}_i, \mathbf{v}_j$)
  $\left(\tilde{\alpha}_{\text{G}}^{(k)}, \tilde{\lambda}_{\text{G}}^{(k)}\right) \leftarrow \text{SymmetricBilinear}_G(\mathbf{v}_i, \mathbf{v}_j)$
  $\alpha_G^{(k)} \leftarrow 1 + \text{softplus}(\tilde{\alpha}_G^{(k)}) + \varepsilon$
  $\lambda_G^{(k)} \leftarrow \text{softplus}(\tilde{\lambda}_G^{(k)}) + \varepsilon$
  $\left(\tilde{\alpha}_{\text{B}}^{(k)}, \tilde{\beta}_{\text{B}}^{(k)}\right) \leftarrow \text{Bilinear}_B(\mathbf{v}_i, \mathbf{v}_j)$
  $\left(\alpha_{\text{B}}^{(k)}, \beta_{\text{B}}^{(k)}\right) \leftarrow 1 + \text{softplus}\left(\tilde{\alpha}_{\text{B}}^{(k)}, \tilde{\beta}_{\text{B}}^{(k)}\right) + \varepsilon$
  return $\text{Gamma}(\alpha_G^{(k)}, \lambda_G^{(k)}), \text{Beta}(\alpha_B^{(k)}, \beta_B^{(k)})$
end function

function UPDATESTATE ($\mathcal{S}_{(k)}$, $\{c_m^{(k)}\}$, $\{\mathbf{v}_i\}$, $m^{(k)}$, parent index u, cherry length $s^{(k)}$)
  $i, j \leftarrow m^{(k)}$
  $\mathbf{v}_u \leftarrow \text{SymmetricBilinear}(\mathbf{v}_i, \mathbf{v}_j)$ *Compute parent embedding*
  $\mathcal{S}_{(k+1)} \leftarrow \left(\mathcal{S}_{(k)} \cup \{u\}\right) \setminus m^{(k)}$ *Update mergeable nodes*
  $\{c_m^{(k+1)}\} \leftarrow \begin{cases} 0, & u \in m \\ \max\left(c_m^{(k)}, s^{(k)}\right), & u \notin m \end{cases} \quad \forall m \in \mathcal{C}_{(k+1)}$ *Update constraints*
  return $\mathcal{S}^{(k+1)}, \{c_m^{(k+1)}\}, \{\mathbf{v}_i, i \in \mathcal{S}_{(k+1)}\}$
end function

Evaluating $q_{\psi(x)}\left(\theta = (\tau, \ell)|x\right)$ with BayesNJ:

Algorithm 11 BayesNJ log-probability evaluation

function BAYESNJ (Embeddings $\{\psi(x)\} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, merges $\{m^{(k)}\}$, branch lengths $\{\ell^{(k)}\}$)
  $\mathcal{S}_{(1)} \leftarrow \{1, \ldots, N\}$
  $\{c_m^{(1)}\} \leftarrow \{0 \,\forall m \in \mathcal{C}_{(1)}\}$ *Initialize constraints*
  $\mathcal{L}_\tau \leftarrow 0, \mathcal{L}_\ell \leftarrow 0$ *Initialize log-probabilities*

  for all $k \in [1, \ldots, N-2]$ **do**
    # Topological component
    if $k \leq N-3$ **then**
      $\{\text{score}_{m=\{i,j\}}\} \leftarrow \{\text{SymmetricBilinear}(\mathbf{v}_i, \mathbf{v}_j)\}$ *Score all pairs $m = \{i, j\} \in \mathcal{C}(k)$*
      $\{q_m(m)\} \leftarrow \text{softmin}(\{\text{score}_m\})$
      $\mathcal{L}_\tau \mathrel{+}= \log(q_m(m^{(k)}))$
    end if

    # Branch length component
    $i, j \leftarrow m^{(k)}$
    $s^{(k)} \leftarrow \ell_i^{(k)} + \ell_j^{(k)}$
    gamma, beta $\leftarrow$ GETDISTRIBUTIONS$(\mathbf{v}_i, \mathbf{v}_j)$
    $\mathcal{L}_\ell \mathrel{+}= \log \text{PDF}_{\text{Gamma}}\left(s^{(k)} - c_{m^{(k)}}^{(k)} | \text{gamma}\right) + \log \text{PDF}_{\text{Beta}}\left(\ell_i^{(k)}/s^{(k)} | \text{beta}\right) - \log(s^{(k)})$

    # Prepare next iteration
    $u \leftarrow N + k$ *Define new parent's index*
    $\mathcal{S}_{(k+1)}, \{c_m^{(k+1)}\}, \{\mathbf{v}_i\} \leftarrow$ UPDATESTATE$(\mathcal{S}_{(k)}, \{c_m^{(k)}\}, \{\mathbf{v}_i\}, m^{(k)}, u, s^{(k)})$
  end for

  # Deal with final branch length
  $u \leftarrow 2N - 2$;
  $\{w\} \leftarrow \mathcal{S}_{(N-1)} \setminus \{u\}$ *last un-merged node*
  gamma, _ $\leftarrow$ GETDISTRIBUTIONS$(\mathbf{v}_u, \mathbf{v}_w)$
  $\mathcal{L}_\ell \mathrel{+}= \log \text{PDF}_{\text{Gamma}}(\ell_w - c_{\{u,w\}}^{(N-1)} | \text{gamma})$

  return $\mathcal{L}_\tau, \mathcal{L}_\ell$
end function

BayesNJ computes the probability of unrooted trees, however in order to compute the branch-length component of the 3 branches i, j, w remaining after N-3 merges, a "virtual" $(N-2)^{\text{th}}$ merge is applied. The topological component of this merge $q_m(m^{(N-2)}) = 1$ and it is simply used to define a parent embedding u necessary to compute the branch-length component $q_\ell(m^{(N-2)})$ for branches (u, i) and (u, j). To compute the probability of the remainng (u, w) branch's length, we simply use the Gamma component modeling the sum of the cherry and ignore the Beta component, modeling the balance factor.

Sampling from $q_{\psi(x)}\left(\theta = (\tau, \ell)|x\right)$ with BayesNJ

Algorithm 12 Sampling from the posterior
For the greedy MAP approximation, we simply replace the softmin of scores with an argmin to sample merges, and replace samplings from Gamma and Beta distributions with the corresponding mode.

function SAMPLINGBAYESNJ (Embeddings $\{\psi(x)\} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$)
  $\mathcal{S}_{(1)} \leftarrow \{1, \ldots, N\}$
  $\{c_{ij}^{(1)}\} \leftarrow \{0, \forall (i,j) \in \mathcal{C}_{(1)}\}$ *Initialize constraints*
  $M \leftarrow [], B \leftarrow []$ *Initialize merge and branch length sequences*

  for all $k \in [1, \ldots, N-2]$ **do**
    # Topological component
    $\{\text{score}_{ij}\} \leftarrow \{\text{SymmetricBilinear}(\mathbf{v}_i, \mathbf{v}_j)\}$ *Score all pairs $(i,j) \in \mathcal{C}_{(k)}$*
    $m^{(k)} \sim \text{softmin}(\{\text{score}_{ij}\})$ *Sample merge $m = \{i, j\}$*

    # Branch length component
    $i, j \leftarrow m^{(k)}$
    gamma, beta $\leftarrow$ GETDISTRIBUTIONS$(\mathbf{v}_i, \mathbf{v}_j)$
    $s^{(k)} \sim \text{gamma} + c_{m^{(k)}}^{(k)}$ *Sample sum*
    $r^{(k)} \sim \text{beta}$ *Sample ratio*
    $\ell_i^{(k)} \leftarrow r^{(k)} \cdot s^{(k)}$
    $\ell_j^{(k)} \leftarrow s^{(k)} - \ell_i^{(k)}$

    # Save sampled values
    if $k \leq N-3$ **then**
      $M[k] \leftarrow m^{(k)}$ *last merge is only used to compute branch-lengths*
    end if
    $B[i] \leftarrow \ell_i^{(k)}, B[j] \leftarrow \ell_j^{(k)}$

    # Prepare next iteration
    $u \leftarrow N + k$ *Define new parent's index*
    $\mathcal{S}_{(k+1)}, \{c_m^{(k+1)}\}, \{\mathbf{v}_i\} \leftarrow$ UPDATESTATE$(\mathcal{S}_{(k)}, \{c_m^{(k)}\}, \{\mathbf{v}_i\}, m^{(k)}, u, s^{(k)})$
  end for

  # Deal with final branch length
  $u \leftarrow 2N - 2$;
  $\{w\} \leftarrow \mathcal{S}_{(N-1)} \setminus \{u\}$ *last un-merged node*
  gamma, _ $\leftarrow$ GETDISTRIBUTIONS$(\mathbf{v}_u, \mathbf{v}_w)$
  $\ell_w \sim \text{gamma} + c_{\{u,w\}}^{(N-1)}$
  $B[w] \leftarrow \ell_w$

  return M, B
end function

BayesNJ also samples unrooted trees, the final $(N-2)^{\text{th}}$ merge is a "virtual" one and is only necessary to sample branch lengths in the same fashion as described in Algorithm 11

A.2 Reparameterization of the branch length probability q_ℓ

Because we want to constrain the sum of branch lengths in our probability distribution, we reparameterize $\ell = (\ell_i, \ell_j)$ as $(s, r) = g(\ell) = (\ell_i + \ell_j, \ell_i/(\ell_i + \ell_j))$ and model the distribution of $q_{(s,r)}(s, r) = q_s(s)q_r(r)$, as detailed in 3.3. In order to recover the probability q_ℓ from $q_s(s)q_r(r)$,

we must account for this change of variable:

$$q_\ell(\ell) = q_{(s,r)}\left(g(\ell)\right)\left|\det J_g(\ell)\right|,$$

where

$$J_g(\ell) = \begin{pmatrix} 1 & 1 \\ \frac{\ell_j}{(\ell_i+\ell_j)^2} & -\frac{\ell_i}{(\ell_i+\ell_j)^2} \end{pmatrix}$$

is the Jacobian matrix of g, whose determinant is therefore $-1/(\ell_i+\ell_j) = -1/s$. As a result,

$$q_\ell(\ell) = q_s(s)q_r(r)/s.$$

A.3 Training runs

Model	Starting weights	Embedding dimensions $(c_s\|c_z)$	Training data	Loss function	Training wall-time	Batch size	Scheduled epochs	Target LR	Warmup	Selected step
$PF2_{Base}$	Random	(128|256)	BD,LG+G8	BayesNJ	60h	16	30	10^{-4}	10^4	78,000
$PF2_{MAE}$	Random	(128|256)	BD,LG+GC	MAE	26h	16	30	$5\cdot10^{-4}$	10^3	51,000
PF2	$PF2_{base}$	(128|256)	BD,LG+G8,multi	BayesNJ	14h	1-71	30	10^{-6}	0.5%	26,000
$PF2_{Cherry}$	$PF2_{base}$	(128|256)	BD,Cherry	BayesNJ	20h	15	30	10^{-5}	0.5%	19,000
$PF2_{SelReg}$	$PF2_{base}$	(128|256)	BD,SelReg	BayesNJ	20h	15	30	10^{-5}	0.5%	20,000
$PF2_{Potts}$	Random	(128|256)	BD,Potts	BayesNJ	60h	32	6	10^{-4}	10%	87,000
$PF2_{Uniform}$	Random	(128|256)	U+Exp,LG+G8	BayesNJ	60h	16	5	10^{-4}	10^4	78,153

Table 1: Training run parameters. All runs were in a distributed data-parallel setting using 4 H100 GPUs. Final models are shown with their name in bold. More details on the training data in Appendix A.3.1 and Figure 5.

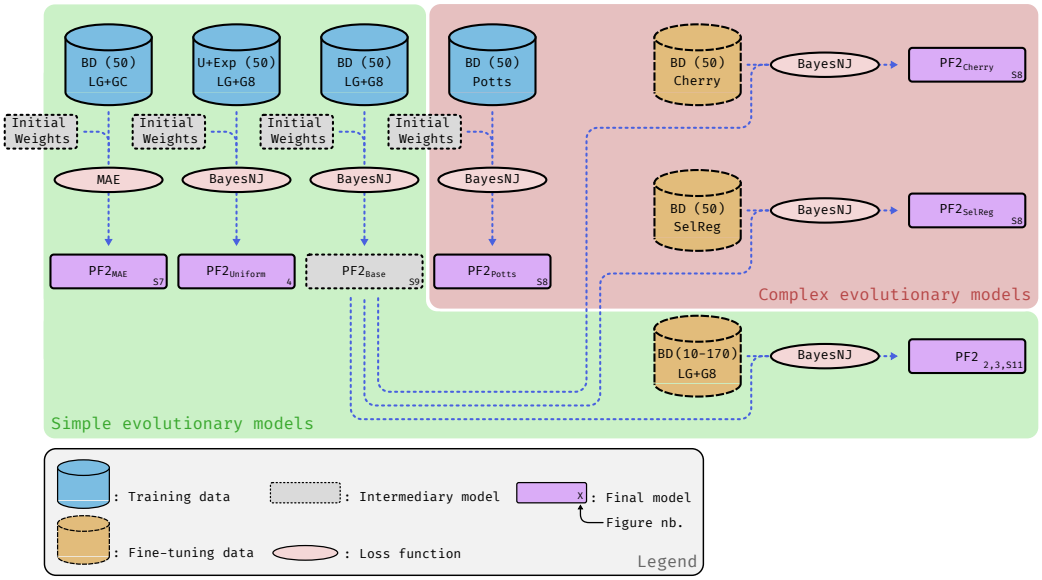

Figure 5: Training setting for all PF2 instances. PF2 instances are shown in rectangles, with a full outline and purple fill if they are final models used for inference in the main results, and a dotted outline and gray fill if they are used as the starting model for fine tuning runs. The loss functions used for each training or fine-tuning runs are shown in ovals. Finally the training (blue) and fine-tuning (yellow) datasets are also shown as cylindrical shapes. For each dataset, the number of training and validation examples is shown in the top section of the cylinder. The simulation priors are shown in the body of each cylinder : (1) the tree topology prior with the training tree size in parentheses, (2) the MSA evolutionary model. The tree priors are either *(BD)*: rescaled Birth-death, described in [29], or *(U+Exp)*: Uniform tree topology with $\lambda = 10$ exponentially distributed branch lengths available in RevBayes [15]. The number of figure in which the performance of a given model is studied is shown in the bottom right corner of the corresponding rectangle. Datasets are described in Appendix A.3.1

A.3.1 Training datasets

From [29]: The 200k LG+GC, Cherry and Selreg datasets, used for training and fine-tuning models were obtained from the original paper. A full description of these datasets is available in the supplementary material of [29]

LG+G8 dataset: This is the main dataset used to train PF2. the 7,966,499 training trees (and 10,000 validation trees) are simulated with 50 leaves, following the empirically rescaled birth-death procedure described in [29]. MSA were simulated along these trees using IQTree's alisim tool [27] under the LG evolutionary model. Rate heterogeneity across sites was modeled with an 8 category discrete Gamma. Insertions-deletion events were added using alisim, with identical rates of $2 \cdot 10^{-3}$, and insertion-deletion length was modeled with the default options: a zipfian distribution with an exponent of 1.7 and a maximum size of 100. These parameters were chosen to yield MSAs with 10% of gaps on average on trees sampled from our prior distribution, which seems consistent with the gap content of empirical data.

Multi-size LG+G8 dataset: This dataset was used to fine-tune PF2 to limit the effect of overfitting to the training. Tree-MSA pairs were generated with 10 to 170 leaves (with a step-size of 10) using the same procedure as the main LG+G8 dataset described above. We simulated 20,000 training examples and 1,000 validation examples for each tree size.

Uniform topology training dataset: This dataset was used to train PF2 under the prior used in RevBayes [15] to estimate the posterior distribution shown in Figure 4. 10^6 training (resp. 5000 validation) trees were simulated with RevBayes with uniformly distributed topologies and branch lengths sampled from an exponential distribution with a mean of 0.2. MSAs were simulated along those trees using the same procedure as the main LG+G8 and the multi-size LG+G8 datasets described above.

Potts Model training dataset: Pagnani & Barrat-Charlaix [30] introduced a continuous-time sequence evolution model parametrized by a given Potts model. This can be used to simulate realistic, family-specific MSAs along a phylogenetic tree that accurately reflect real epistatic effects. We used this method to generate 2 million (tree,MSA) pairs under a Potts model fit using `adabmDCA` [36] on the response regulator PF00072 PFAM family [28]. 5,000 of these were used as a separate testing set (see. Figure 7), 5,000 as validation examples and 1.5 million as training examples. This was used to train a PF2 instance from scratch.

A.4 Fine tuning on different tree-sizes

In our current implementation, the BayesNJ module only accepts batches with MSA with the same number of taxa. In order to avoid wasting GPU memory we use a dynamic batch-size during fine-tuning, where the number of examples for a batch with MSAs of a certain size is set to use as much GPU memory as it can. Since batch-size and learning rate are tightly linked, we define the learning-rate schedule for MSAs with 50 sequences and rescale the learning rate applied at a certain batch by the ratio of the current batch-size and the one for MSAs with 50 sequences. We also set a maximum batch-size equal to the one chosen for MSAs of 50 sequences, this ensures that the model sees enough batches for small MSAs during fine-tuning.

A.5 The Cherry dataset in Nesterenko et al. [29] contains unrealistic amounts of coevolution between sites

The Cherry dataset simulated in Nesterenko et al. [29] was simulated by using an incorrect rate matrix. As a result, the amount of co-evolution among pairs of coevolving residues was superior to what can be found in empirical data. The cause of this high amount of coevolution is a mistake in the use of the Cherry matrix [32]. Instead of using the Cherry rate matrix to model pairs of interacting sites, the authors used the product of the rate matrix with its stationary frequencies. In standard models of molecular evolution, it is customary to represent a reversible substitution rate matrix Q as a product of an exchangeability matrix R and stationary frequencies F: $Q = R \times F$. The Cherry dataset in [29] was actually simulated according to a matrix $Q' = R \times F \times F = Q \times F$. The resulting data provides an example of data with extreme amounts of coevolution, which we use as

an example of strong departure from standard models of sequence evolution as implemented in e.g., IQ-Tree [27].

A.6 Break even analysis

We analyze the total compute time needed to obtain inferred trees for PF2 and ML methods in Table 2. While PF2's training time on 7.9M (tree,alignment) pairs was 62h on GPUs, the simulation of the 7.9M pairs took approximately 1450h of CPU time. Similarly, PF2's fine-tuning time on 16×20k (tree,alignment) pairs was 8h30 on GPUs while the simulation of these pairs took 288 CPU hours. This is because we discard all alignments that contain identical sequences, which forces us to simulate many more pairs ($6\times$ more for MSAs with 50 sequences, and up to $60\times$ more for alignments with 170 tips) than we actually use. Other simulation protocols may be much shorter, and we plan to improve this step of our pipeline. In the following, we report break-even points both with and without the simulation. Because of this cost, if we only infer trees for MSAs of 50 tips, PF2 would break even in terms of compute time after 230k tree inferences with IQTree with the LG+GC model (9042 tree inferences without simulation), 16k with IQTree and ModelFinder (629), or 4.3M FastTree inferences (169047). If we focus on trees of 100 tips this goes down to 91k IQTree with LG+GC (3577), 8k IQTree with ModelFinder (315) and 2.2M FastTree inferences (86489). While the break-even points are quite high for point estimation, this changes radically when sampling a large number of trees from the learned posterior distribution. Since PF2 can sample from the posterior in parallel using GPUs it is very fast, and for trees with 50 tips we reach the break even point w.r.t RevBayes in only 11 sampling runs of 10k iterations. In practice it might be even lower because some of the MCMC samples must be discarded during the burn-in phase.

	Simulation	Training	1 tree ($n = 50$)	1 tree ($n = 100$)	10k Samples ($n = 50$)
PF2	1735h	71h	0.6s	1.4s	38.8s
IQTree	-	-	28.9s	73.1s	-
IQTree+MF	-	-	405.5s	826.6s	-
FastTree	-	-	2.1s	4.4s	-
RevBayes	-	-	-	-	172h

Table 2: Time per inference and upfront tasks for PF2 vs ML and MCMC methods. Simulation times are estimated from Snakemake pipeline reports and are reported in CPU time. In practice this is highly parallelisable so wall time would be much shorter. Here we compare PF2 and other methods on 2 tasks. (1) Point estimations (for 50-sequence and 100-sequence MSAs) where we compare ourselves to maximum-likelihood methods: IQTree, IQTree with ModelFinder and FastTree, and (2) Sampling 10k trees from the posterior for a 50-sequence MSA, where we compare ourselves to RevBayes. Times for maximum likelihood methods are obtained from [29]. The mean time to sample 10k trees from the posterior distribution, replicated over 50 50-sequence MSAs, using PF2 is reported. PF2 samples from the posterior in batches of 500 parallel samples. For the RevBayes estimate, we measured how many trees were able to be sampled during a fixed 20h duration and extrapolate. Simulation and Training time for PF2 are the sum of the times for the main training and fine-tuning phases. Reported times are in seconds unless specified otherwise.

A.7 Supplementary Figures

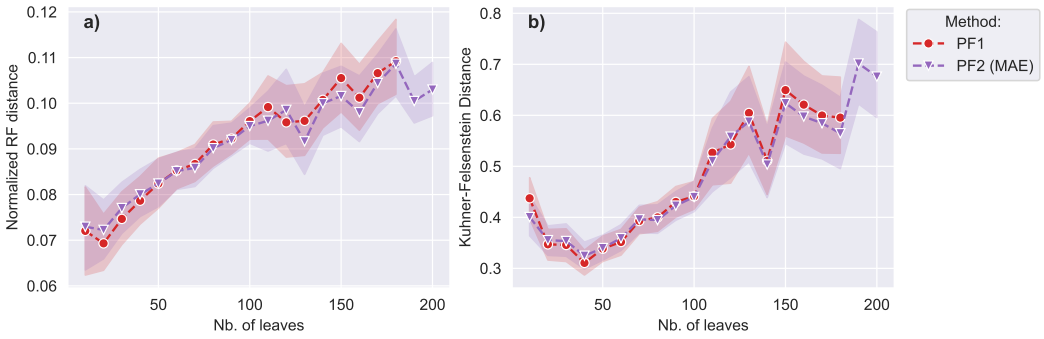

Figure 6: How does PF2 trained with an L1 loss compare to PF? PF2 was trained under the same conditions as the original PF on LGGC trees with an L1 loss on pairwise phylogenetic distances. The Robinson-Foulds distance (left) shows the topological reconstruction accuracy, while the Kuhner-Felsenstein distance (right) takes both topology and branch-lengths into account.

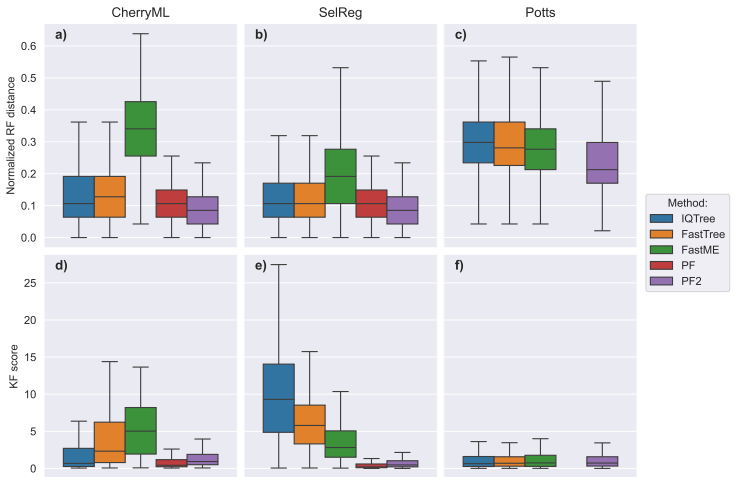

Figure 7: Average PF2 performance under intractable-likelihood models, measured on 50-tip trees. The tree reconstruction performance was measured under 3 different sequence-evolution models for which the likelihood-computation is hard or intractable. Each sequence model is shown in a column, from left to right: Cherry [32] which simulates pairs of co-evolving sites, SelReg [5] which simulates sites under different selective scenarii, and a Potts model fit on the PFAM PF00072 family (also see A.3.1). PF (red) and PF2 (purple) represent instances of the model that have been fine-tuned on data simulated under the corresponding sequence evolution model. Since the Potts model was not used in [29], PF performance measures do not exist for this dataset. PF2 trees in this plot are greedy-MAP trees. For each dataset the error bar shows the 95% CI computed with 1000 bootstrap samples.

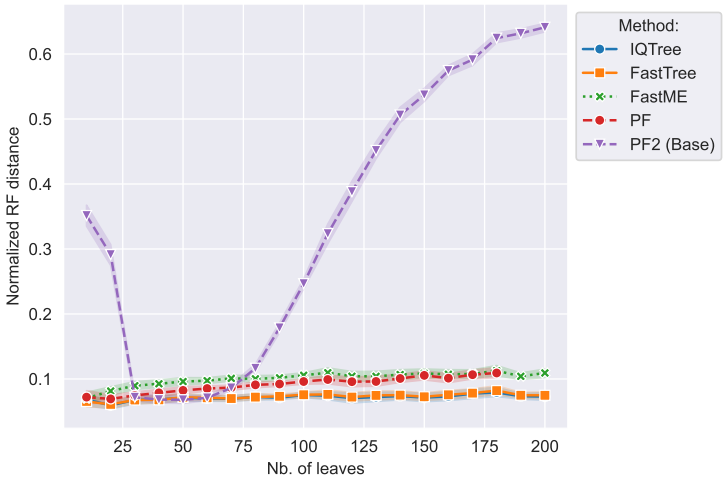

Figure 8: Topological performance for non fine-tuned topology-only Phyloformer 2, measured by the normalized Robinson-Foulds distance. The MSA dataset and compared method results are the same as Figure 2

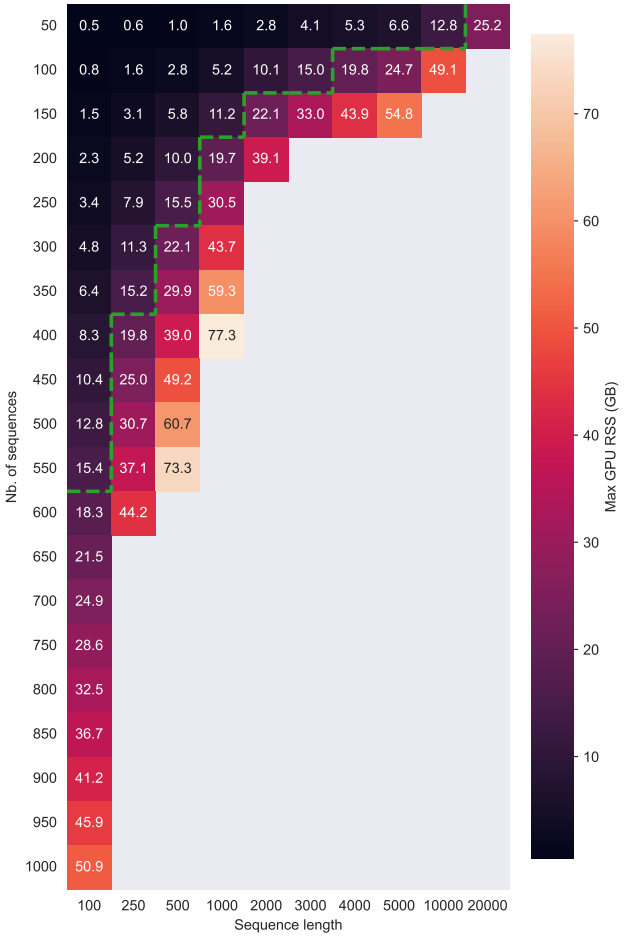

Figure 9: PF2 memory usage (in GB) scaling w.r.t number of sequences and sequence length measured on an H100 GPU. The dashed line indicates what cases are runnable on a 16GB VRAM GPU. These measures are for single-tree estimates using the greedy-MAP, however since most of the memory cost of PF2 is due to the embedding module EvoPF, sampling in parallel should not impact memory usage too much (see Figure 2c)

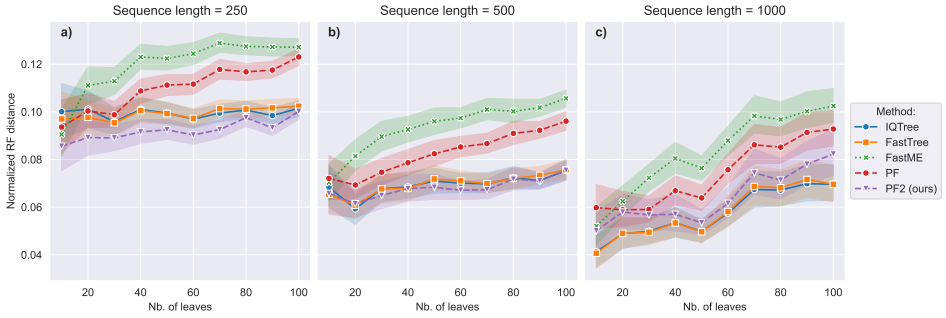

Figure 10: Topological reconstruction accuracy for PF2 and other methods for different sequence lengths. MSAs of different lengths are simulated on the same set of phylogenetic trees for the 3 panels.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe that all the claims made in the abstract are supported by the article.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the discussion and empirical data sections, limitations of the method are discussed.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This article does not introduce any new theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides a link to an online repository containing the training and data simulation scripts necessary to reproduce this study.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The article provides a link to an online git repository allowing access to the code and all necessary scripts to reproduce results. The testing data is also supplied. Training data will be uploaded to the repo.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full sets of hyperparameters are listed in the supplementary material. Data simulating scripts are also supplied in the code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All point estimates are show with bootstraped 95% CIs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computational resources are shown in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that is the case.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper does not discuss societal impact of the work as we do not think it is warranted.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: we believe this paper does not pose such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All data sources, and software are cited in the text.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No human subjects or crowdsourcing was used in the course of this study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: see previous answer.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: no LLMs were used to develop the method described in the paper, nor were they used during writing.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.