Nucleic Acids Research, 2023, **51**, 10909–10923
https://doi.org/10.1093/nar/gkad805
Advance access publication date: 11 October 2023
Computational Biology

Fusang: a framework for phylogenetic tree inference via deep learning

Zhicheng Wang[1,2,†], Jinnan Sun[3,†], Yuan Gao[1], Yongwei Xue[1], Yubo Zhang[4], Kuan Li[1], Wei Zhang[4,5], Chi Zhang[6], Jian Zu[3,*] and Li Zhang[1,*]

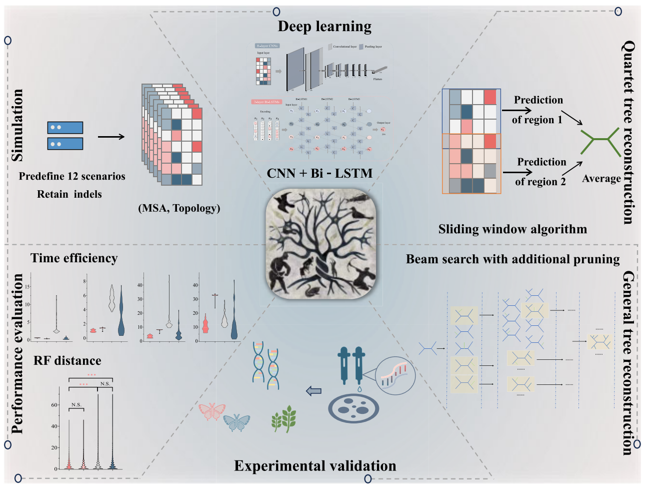
[1]Chinese Institute for Brain Research, Beijing 102206, China
[2]Academy for Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China
[3]School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China
[4]Peking-Tsinghua Center for Life Sciences, Academy for Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China
[5]State Key Laboratory of Protein and Plant Gene Research, School of Life Sciences, Peking University, Beijing 100871, China
[6]Key Laboratory of Vertebrate Evolution and Human Origins, Institute of Vertebrate Paleontology and Paleoanthropology, Center for Excellence in Life and Paleoenvironment, Chinese Academy of Sciences, Beijing 100044, China

[*]To whom correspondence should be addressed. Tel: +86 010 81912680; Fax: +86 010 81912644; Email: zhangli@cibr.ac.cn
Correspondence may also be addressed to Jian Zu. Email: jianzu@xjtu.edu.cn
[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Abstract

Phylogenetic tree inference is a classic fundamental task in evolutionary biology that entails inferring the evolutionary relationship of targets based on multiple sequence alignment (MSA). Maximum likelihood (ML) and Bayesian inference (BI) methods have dominated phylogenetic tree inference for many years, but BI is too slow to handle a large number of sequences. Recently, deep learning (DL) has been successfully applied to quartet phylogenetic tree inference and tentatively extended into more sequences with the quartet puzzling algorithm. However, no DL-based tools are immediately available for practical real-world applications. In this paper, we propose Fusang (http://fusang.cibr.ac.cn), a DL-based framework that achieves comparable performance to that of ML-based tools with both simulated and real datasets. More importantly, with continuous optimization, e.g. through the use of customized training datasets for real-world scenarios, Fusang has great potential to outperform ML-based tools.

Graphical abstract



Introduction

A phylogenetic tree is a branching diagram that models genealogical history to represent the evolutionary relationships of genes or species (1). In most cases, a phylogenetic tree has a bifurcated structure that conveys the concept of speciation as a lineage splitting event producing two separate species. The idea and the important role of phylogenetic tree inference were mentioned in Charles Darwin's *Origin of Species* (2), and this process of constructing phylogenetic trees still affects many aspects of evolutionary biology research, such as population genetics (3), comparative genomics (4) and tumor evolution

Received: January 10, 2023. **Revised:** August 17, 2023. **Editorial Decision:** September 14, 2023. **Accepted:** September 20, 2023

studies (5). However, true phylogenies are unavailable in most cases, which causes difficulties for many related studies.

Statistical inference is currently the main approach for reconstructing phylogenetic trees. Many statistical algorithms have been proposed for phylogenetic tree inference based on multiple sequence alignment (MSA). In general, these algorithms can be classified into two categories: distance-based and character-based algorithms (1). In distance-based algorithms, a distance matrix containing the pairwise distances between sequences is calculated and then used for tree inference. Typically, a clustering algorithm is used to infer hierarchical relationships. This category includes algorithms such as UPGMA (6) and neighbor joining (7). Maximum parsimony, maximum likelihood estimation and Bayesian inference methods represent common character-based algorithms. These algorithms rely on predefined models and parameter sets to evaluate the likelihood levels of parameters, including phylogenetic trees, for a given MSA. They search the tree space and select the tree that best fits the given data. However, several issues have long limited these algorithms. In phylogenetic tree inference cases, the unknown parameters associated with a model must be estimated based on either observed data or theoretical assumptions. When the model and parameters deviate from the real-world situation, the resulting tree can be inaccurate. For example, tumor phylogenetics are quite different from species phylogenetics (8). In addition, it is difficult to construct an exact evolutionary model for insertion and deletion (indel) (9), so most tools such as RAxML simply treat indels as undetermined characters that may result in the loss of useful evolutionary information. Third, heterozygous rates across sequences may result in long branch attraction (10).

Recently, great efforts have been made to apply DL methods to phylogenetic tree inference (11,12). Two major challenges have been encountered: how to address a large number of sequences and how to handle MSAs with varying sequence lengths. To overcome the first challenge, Suvorov et al. focused on quartet trees with fixed sequence lengths (like 1000 bases) and achieved very good performance with sophisticatedly simulated nucleotide training datasets. To address more sequences, Zou et al. integrated quartet puzzling (13), a form of stepwise addition (9), and successfully reconstructed a phylogenetic tree with a small Robinson-Foulds (RF) distance from the real topology of 19 amino acid sequences. However, this pipeline loses the probability distributions of trees from the DL model after the quartet puzzling process, and sometimes results in multifurcated trees. For the second challenge, Zou et al. treated the length of the input sequence as the channel information of a high-dimensional convolution kernel. However, this approach tends to lose information. For example, information is lost if MSAs with lengths of one thousand and one million are handled in the same way.

In this work, we propose Fusang, a new DL-based framework that aims to provide an evolvable toolkit for daily phylogenetic tree inference applications. The current version of Fusang is designed to perform phylogenetic tree inference on nucleotide MSAs with fewer than 40 input sequences and sequence lengths below 10 000. Fusang performs very well in terms of accuracy and speed with both simulated and real datasets on a benchmark GPU server, although a GPU is not required for phylogenetic tree inference. Additionally, we have developed an online server (http://fusang.cibr.ac.cn) and a Linux command-line package for common phylogenetic tree inference. Fusang represents the first DL-based tool for daily phylogenetic tree inference applications, and the algorithmic differences between Fusang, Tree_learning, and PhyDL are shown in Table S1.

New versions designed for amino acid sequences and specific scenarios are expected in continuous future updates.

Materials and methods

The workflow of Fusang is shown in Figure S1. First, we simulated data and used them for DL model training. Then, we used the DL model to predict subquartet trees and integrated the results to obtain the complete phylogeny. Finally, we validated our methods with both simulation data and real biological data.

Hardware configuration

All the tests were performed on the benchmark GPU server, which has one Intel(R) Xeon(R) Gold 6140 CPU with a base clock speed of 2.30 GHz and 12 cores and one Tesla V100 SXM2 GPU with 32 GB of video RAM. For phylogenetic tree inference, Fusang can run on a server with only a CPU, but having a GPU as well is a plus for acceleration.

Simulation data

Before performing the simulation, we visualized the branch length patterns of several published datasets (Figure S2, the data sources are listed in Table S2 sheet 1). To quantitatively measure the relative differences between pairs of branch length distributions, we used the Jensen−Shannon divergence (JSD) measure. Suppose that $P(X)$ and $Q(X)$ are two branch length distributions; the equations for calculating the JSD between them are listed as follows:

$$D\left(P, Q\right) = \sum P\left(X\right) \log_2 \frac{P\left(X\right)}{Q\left(X\right)} \tag{1}$$

$$\begin{aligned} JSD\left(P, Q\right) = & \frac{1}{2} D\left(P\left(X\right), \frac{P\left(X\right)+Q\left(X\right)}{2}\right) \\ & + \frac{1}{2} D\left(Q\left(X\right), \frac{P\left(X\right)+Q\left(X\right)}{2}\right) \end{aligned} \tag{2}$$

$$0 \leq JSD\left(P, Q\right) \leq 1 \tag{3}$$

Here, a higher JSD represents larger differences between the two tested distributions. We list the JSDs between the branch length distributions of five published datasets (Table S2 sheet 2).

The choice of the strategy for simulating the '(MSA, Phylogeny)' pair for supervised deep learning is not trivial but essential because it affects the performance achieved on realistic data. To ensure that Fusang performs well in common applications, our simulation parameter settings were close to the properties of real-world biological sequences instead of covering extreme cases of quartet tree spaces, as Suvorov et al. did. Generally, we predefined 12 test scenarios with the aim of covering three sequence types (coding, noncoding, and standard sequences for a mixture of coding and noncoding sequences), two sequence lengths (200 and 1000 for the root sequence), and two branch length distributions (uniform and gamma). We named these 12 scenarios in the form of regular expressions '[SCN][1–2][UG]' as mentioned in the Results section. We generally have 11 main parameters for each scene, which are listed as 12 columns in Table S3. Two (mean pairwise divergence and I/S ratio) of these 11 parameters were selected as scales to define different potential evolutionary scenarios. The first scale is the mean pairwise divergence, which we controlled within a reasonable range (Table S3, 'Mean pairwise divergence') for different scenarios according to the previous discussion of suitable sequence divergence values for phylogenetic tree inference (14). The second scale is the I/S ratio, which we set (Table S3 'INDEL rate') for different scenarios according to previous research (15). Except for the mean pairwise divergence and I/S ratio, the settings of the other 9 main parameters referred to previous research conducted on Tree_learning (11) and PhyDL (12). For example, the 'Length', 'Indel length distribution', 'Internal branch length', 'External branch length', 'Max indel length', 'Proportion of invariable sites', and 'Gamma shape' parameters in Table S3 refer to the parameter settings of Tree_learning (11), and the 'Leaf nodes N of the larger tree' parameter in Table S3 refers to PhyDL (12).

Except for the main parameters, the other simulation settings were the same as those of the original Tree_learning project (11), which selected evolutionary models from 'JC', 'TIM', 'TIMef', 'GTR' and 'UNREST' and generated state frequencies from the Dirichlet distribution. In addition, we simulated an ~15% LBA tree (16) in our training dataset by following the suggestions of PhyDL (12).

A schematic of our simulation procedure is shown in Figure S3, and to simplify the description, we list the details for generating a single '(MSA, Phylogeny)' pair, which can be encoded using the rule ('A': '0', 'T': '1', 'C': '2', 'G': '3', '-': '4', 'N': '4') and used as one data point in the subsequent DL modeling procedure. We can simply repeat the following Algorithm M times to generate a dataset with M '(MSA, Phylogeny)' pairs. The algorithm for generating a single '(MSA, Phylogeny)' pair is described as follows.

First, we generated a phylogenetic tree topology with randomly selected N ($4 < N < 40$) leaf nodes using the ETE3 software package (17). Second, we sampled an expected mean pairwise divergence d_1 from a predefined distribution, which is recorded in Table S3 ('Mean pairwise divergence'). Third, we assigned a branch length to each branch that was sampled from a uniform or gamma distribution. Then, we calculated the mean pairwise divergence d_2 and scaled the branch length by a factor of $\frac{d_1}{d_2}$ so that the mean pairwise divergence of N leaf nodes was equal to d_1. Fourth, we randomly sampled 4 leaf nodes from this phylogeny, which provided the topology and branch length of a quartet tree. Finally, we used Indelible (18) to simulate the MSA of the given phylogeny.

After completing the simulation, we evaluated our simulation data. First, we visualized the branch length distribution patterns of all 12 scenarios (Figure S2), and all plots showed long-tailed distribution patterns, supporting the idea that most branch lengths are small and few branch lengths are large, which is also the pattern seen in most real biological data. Second, we calculated the JSD of our simulation relative to five publicly available datasets (Table S2 sheet 3) and found relatively small JSD values. Third, as shown in Table S4, we compared the JSD of Fusang relative to public datasets to be smaller than that of Tree_learning (simulation script https://github.com/SchriderLab/Tree_learning/blob/master/INDELible/indelible_controlgen_INDEL001.R).

Finally, through the algorithm above, we generated '(MSA, Phylogeny)' pairs, and we used the following steps to build our DL dataset. First, we encoded MSAs with the following rule '('A': '0', 'T': '1', 'C': '2', 'G': '3', '-': '4', 'N': '4')' to generate data, and each MSA was encoded as a $4 \times L$ matrix (L is the length of the MSA). Second, we encoded the phylogeny with the following rule: '('((1, 2), 3, 4)': 0, '((1, 3), 2, 4)': 1, '((1, 4), 2, 3)': 2)' (1, 2, 3, and 4 are the lexicographic orders of the names of the 4 MSA sequences) to generate the dataset labels. Then, each phylogeny was encoded as a class number (0, 1 or 2). Third, for each '(MSA, Phylogeny)' pair, we performed encoding as described above. In the final dataset, each data point had a shape of $M \times 4 \times L$ (M is the dataset size), and the label was an M-dimensional vector. The phylogenetic tree inference process for the dataset with 12 scenarios was separate and followed the same procedure described above but with different parameter settings (Table S3). Finally, we conducted data partitioning to divide each dataset into a training set, validation set and test set; the details are shown in Table S5.

DL model training

The structure of the deep learning network used in this article is shown in Figure 1. The specific shape of each layer's network structure can be seen in Table S6. An introduction to a specific deep learning network architecture can be found in the 'Results: Design principles' section. For DL training, we used the backpropagation (19) algorithm to optimize the parameters of the DL network during the process of minimizing the cross-entropy loss function. Suppose that we have N samples, the true distribution of each sample is $\mathrm{y_i} = (y_{i1}, y_{i2}, y_{i3})$ and the distribution predicted by DL is $\hat{\mathrm{y}}_\mathrm{i} = (\hat{\mathrm{y}}_{\mathrm{i1}}, \hat{\mathrm{y}}_{\mathrm{i2}}, \hat{\mathrm{y}}_{\mathrm{i3}})$. The loss function of the optimization procedure is as follows.

$$\mathrm{Loss} = -1/\mathrm{N}\sum_{i=1}^{N}\sum_{j=1}^{3} y_{ij}\log\left(\hat{\mathrm{y}}_{\mathrm{ij}}\right) \tag{4}$$

During the training procedure, we used the Adam optimizer (20) to minimize the loss function, which is another posterior probability approximation in theory (21). Importantly, compared to traditional likelihood methods, DL can learn complex evolutionary patterns from data instead of setting evolutionary parameters, which are hard to estimate correctly. We also incorporated the label smoothing strategy (22), which smoothed the target distribution during the learning process. As a result, extreme deviations could be reduced due to the smoothing of the target distribution. Plots of the loss curves produced on the validation set during training are shown in Figure S4, and they gradually converged in all 12 scenarios, which indicates the robustness of our DL model.

DL model prediction

To use DL in the subsequent phylogenetic tree inference process, we needed to generate predictions for all subquartet trees, as schematically shown in Figure 2.(1). For DL model prediction, the input tensor had shapes of $\mathrm{N} \times 4 \times 240$ for the six '[SCN]1[UG]' scenarios and $\mathrm{N} \times 4 \times 1200$ for the six '[SCN]2[UG]' scenarios. N indicates the number of quartet trees that we needed to predict. In the following section ('Phylogenetic tree inference'), we describe the process of sampling quartet MSAs from the original MSA and using DL to predict the results. The value '4' represents four quartet tree sequences. The values of 240 and 1200 represent quartet MSAs

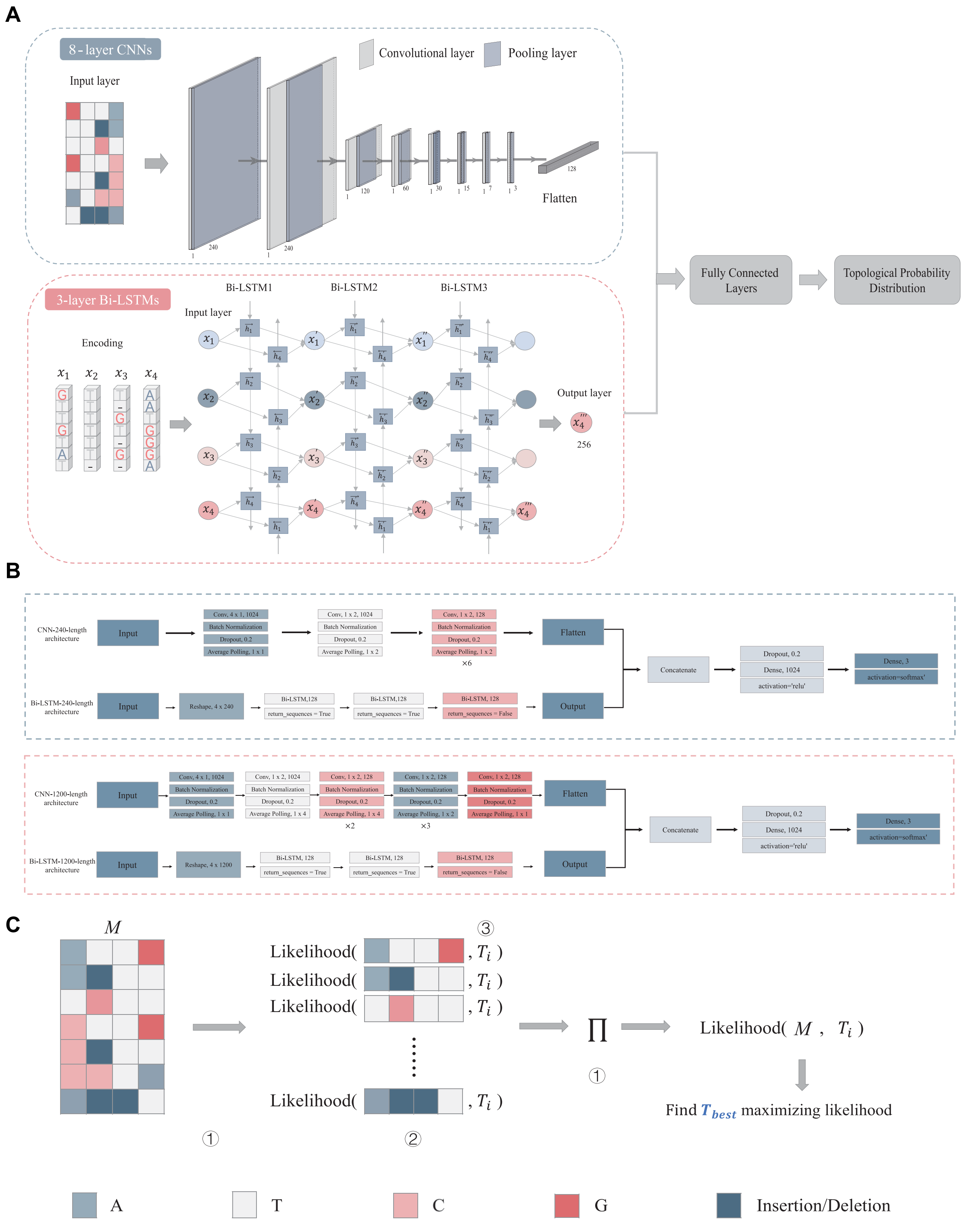
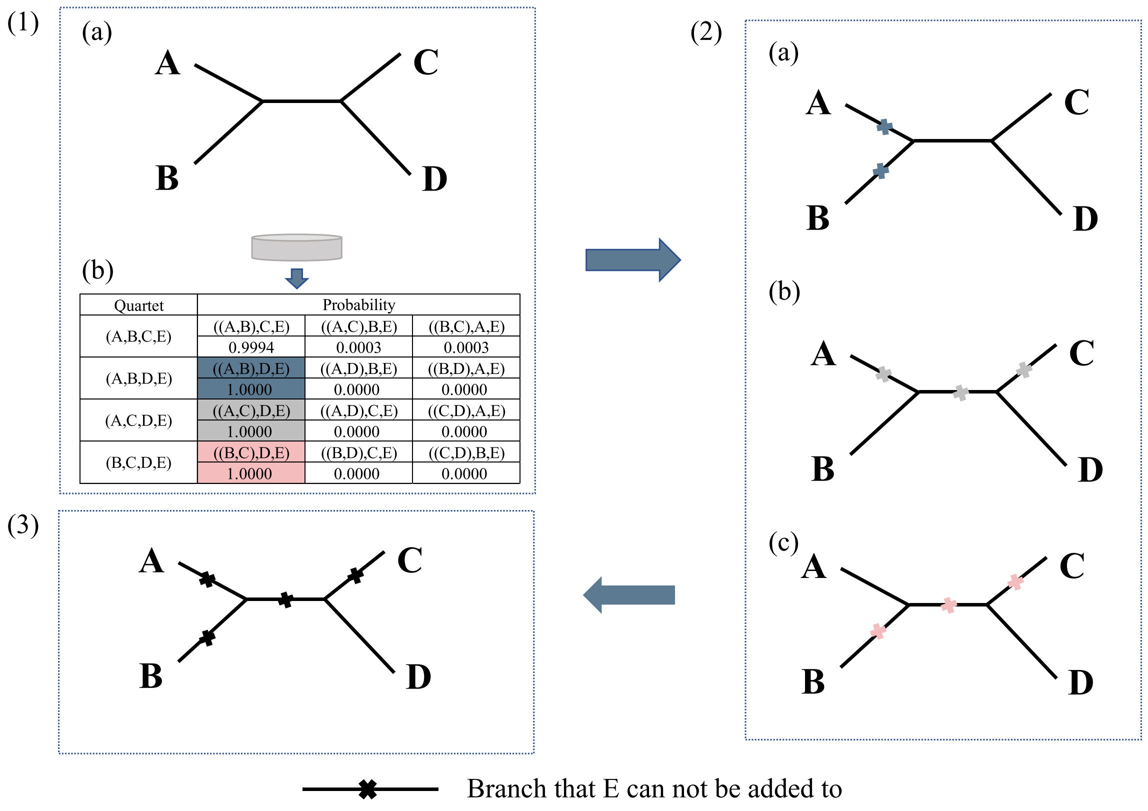


Figure 1. DL structure of Fusang. (**A**) DL structure: an algorithm combining 8-layer CNNs and 3-layer Bi-LSTMs was used to feed the results of the CNN and Bi-LSTM together into a fully connected network to obtain the probability distributions of three possible topologies (the network structure of the CNN with a model of length 240 is shown as an example); (**B**) details of the parameters of the models with lengths of 1200 and 240 for the network with 8-layer CNNs and 3-layer Bi LSTMs, where the presence of 'x i' under a partial structure block indicates that the partial structure is repeated i times; (**C**) general ML structure and its limitations: ①The assumption of site-independent evolution largely increases the incurred computational costs. ②It is difficult to predefine the likelihood of insertion and deletion.



Quartet	Probability		
(A,B,C,E)	((A,B),C,E)	((A,C),B,E)	((B,C),A,E)
	0.9994	0.0003	0.0003
(A,B,D,E)	((A,B),D,E)	((A,D),B,E)	((B,D),A,E)
	1.0000	0.0000	0.0000
(A,C,D,E)	((A,C),D,E)	((A,D),C,E)	((C,D),A,E)
	1.0000	0.0000	0.0000
(B,C,D,E)	((B,C),D,E)	((B,D),C,E)	((C,D),B,E)
	1.0000	0.0000	0.0000

Figure 2. Schematic of the pruning process according to the topologies with the top-n probabilities (n equals 3 in this schematic). (1) Before pruning. (**a**) This schematic shows that during the inference process, we have already reconstructed ((A, B), C, D), so we need to add E in next iteration. The 'cylinder (database)' icon represents the probability table of all possible quartet topologies. If we have k nodes, this table has C_k^4 rows and 3 columns. Each row represents a quartet combination of k nodes, and each column represents a possible topology of the quartet. (**b**) The table extracted from the table shown in subfigure (a), which contains rows representing the combination (x, y, z, E); the dictionary order of x, y, z should be smaller than E. The three columns represent the topologies ((1, 2), 3, 4), ((1, 3), 2, 4), ((2, 3), 1, 4), respectively, and 1, 2, 3, 4 represent the dictionary order of the quartet of each row. (2) Three pruning steps. (**a**) Pruning branches according to ((A, B), D, E), which has one of the top-n probabilities. (**b**) Pruning branches according to ((A, C), D, E), which has one of the top-n probabilities. (c) Pruning branches according to ((B, C), D, E), which has one of the top-n probabilities. (3) After pruning.

with fixed lengths of L. The prediction result has a shape of N × 3. N has the same meaning as that described above. The value '3' represents the one-hot encoding result of three possible categories of the classification problem. Utilizing the DL model prediction module, we can reconstruct the general phylogenies. The details of this process are discussed in the next section.

Comparison of DL and ML methods

First, we retrieved the training set, validation set and test set with indels under S2G parameters through simulation. For the Fusang and Tree_learning algorithms, we first used the training set and validation set for model training, and the resulting models were tested using the test set to obtain the prediction accuracy of the model. For IQ-TREE and RAxML, we used the data from the test set for phylogenetic inference to obtain the prediction accuracy of the algorithms. To ensure that the comparison of model accuracy is meaningful, we used the same set of data for indel removal to obtain datasets without indels under S2G parameters and used the same method for training and prediction to determine the prediction accuracy of the four algorithms. In addition, we also compared the Fusang and Tree_learning models trained with indel data using the indel-free test set and vice versa.

Phylogenetic tree inference

Three important steps are required for general phylogenetic tree inference: solving the MSAs of four leaf nodes with fixed lengths, solving the MSAs of four leaf nodes with variable lengths, and solving general MSAs with variable lengths. Our strategy solves these three problems in a step-by-step manner. The details for solving the problem of multiple-leaf-node phylogenetic trees with variable lengths are as follows.

For a given MSA with arbitrary leaf nodes of arbitrary lengths, first, we sample every possible MSA with 4 leaf nodes from the original MSA. For example, for an MSA with n leaf nodes, we sample C_n^4 MSAs with 4 leaf nodes from the original MSAs with all n leaf nodes. Then, after performing encoding using the rules '('A': '0′, 'T': '1′, 'C': '2′, 'G': '3′, '-': '4′, 'N': '4′)', we obtain a matrix with a shape of $C_n^4 \times 4 \times L$, where L is the arbitrary length of this MSA. Second, we divide every MSA with 4 leaf nodes into many sub-MSAs (all are 4-leaf-node MSAs with fixed lengths) using the sliding window algorithm, as mentioned later in this section, and obtain an input tensor for DL prediction (see the 'DL model prediction' section) with a size of $(W \times C_n^4) \times 4 \times 240$ (W is the number of sliding windows, and '240' is for the $L < 1200$ case) or $(W \times C_n^4) \times 4 \times 1200$ (when $L > 1200$). Third, we use the DL model to predict and obtain the result with a shape of $(W \times C_n^4) \times 3$ (the meaning of this expression is mentioned in the 'DL model prediction' section). Because we assume that the majority of the MSA regions support the true topology, we integrate the phylogenetic results of each subregion of the original MSA. We reshape the $(W \times C_n^4) \times 3$ output to $C_n^4 \times 3 \times W$ and then calculate the average of all W subregions, which is to say that we calculate the average of the last dimension and obtain a $C_n^4 \times 3$ result.

For the sliding window algorithm, we divide the original MSA into many sub-MSAs with or without overlap, calculate the phylogenies of these sub-MSAs, and then integrate the results of the sub-MSAs to generate the phylogeny of the whole MSA. For the default version of Fusang, if the length of the given MSA is smaller than 1200, we use sliding windows with lengths of 240, and if the length of the MSA is larger than 1200 and smaller than 10K, we use sliding windows with lengths of 1200. The number of windows W and the step size S are calculated as follows.

$$W = \left\lfloor \frac{\text{MSA length} \times C}{\text{window size}} \right\rfloor + 1 \qquad (5)$$

$$S = \left\lfloor \frac{\text{MSA length} - \text{length of sliding window}}{W} \right\rfloor \qquad (6)$$

In this formula, C is a hyperparameter that determines the number of windows in which, on average, a nucleotide base is covered. Usually, C can be any positive real number such as 1, 1.5, 2, and so on. For example, if C is equal to 2, it means that, on average, a base will be covered by two sliding windows. Therefore, we can use the sliding windows starting from the first MSA base, the distance between the previous window and the next window is S, and the last sliding window covers the last position of the MSA. In the default setting, C is equal to 1. With the sliding window method, Fusang can handle variable MSA lengths and retain the main information. Additionally, the speed can be optimized after performing parallel optimization.

For handling variable species, a modified stepwise addition algorithm with beam search is used. The schematic of this procedure is shown in Figure S5.

Assume that we need to reconstruct a phylogenetic tree with n leaf nodes and that we have already obtained the $C_n^4 \times 3$ predicted probability distribution (such as that in Figure 2.(1), as mentioned in the previous part of this section); the next steps are as follows.

Generally, we iteratively add new nodes to a reconstructed subtree until we have added all nodes. As shown in Figure S5, first, we reconstruct T_4 by randomly selecting 4 species and calculate their topology. Second, we iteratively add nodes to T_4 (from N_5 to N_n) and obtain the final tree after we have added all nodes. For this procedure, in each iteration, a new node is added to the current tree on the branch when it maximizes the tree score. The definition of the tree score for T_i is the sum of the probability of every quartet tree q_i sampled from it. The formula is as follows.

$$\text{Score}\,(T_i) = \sum_{q_i} P\,(q_i) \qquad (7)$$

In most cases, if a tree with more than four species has a higher probability, the quartet trees sampled from it should have higher probabilities. Therefore, we use the sum of the probabilities from all quartet trees sampled from a given tree to evaluate its relative probability. When the number of species increases, the possible tree space becomes excessively large. Therefore, we use the beam search algorithm, as shown in Figure S5, to iteratively obtain trees with 5 to n species. As shown in Figure S5, we use a beam size of 3 as an example in the schematic, which is to say that in each iteration, we select only the trees with the three highest tree scores. For example, when we expand T_4 to T_5, we add N_5 to all five branches of T_4 and obtain 5 T_5 candidate trees. Instead of adding N_6 to all five candidate trees, we select only the candidate trees with the 3 highest tree scores, which enables us to mask the search spaces of the other two candidate trees. Then, when we expand T_5 to T_6, we add N_6 to all seven branches of every candidate T_5 and obtain 21 candidate trees (each of the three candidate trees from the last iteration has seven new candidate trees for this iteration) of T_5, and we select only the 3 candidate trees with the 3 highest tree scores among the total 21 trees. Therefore, in this iteration, 18 possible search paths are pruned. Then, we repeat this procedure until we obtain the final tree. We investigated the influence of varying the beam size from 1 to 5 on the performance of the tree inference process for typical simulation data. As shown in Figure S6, we found that a high beam size improves the probability of finding the true phylogeny with a tradeoff of a high computational cost. Thus, we choose to use the beam size 1 as the default value, which is equal to that in the greedy search.

In addition, to accelerate the search process, we use a pruning algorithm in each iteration of the stepwise addition algorithm. In general, we first cut useless branches, as shown in Figure 2, instead of directly calculating the tree scores of all possible branches (as shown in Figure S5). Then, the algorithm is based on the pruned results. If we cut many branches and only a few branches (using the following rule: possible branches $\leq$ beam size) can add the next node (e.g. in Figure 2, we can only add the next node to one branch, and one is smaller than the beam size of 3), we directly proceed to the next iteration of the beam search. Otherwise, if (possible branches $>$ beam size), we perform the original beam search on the remaining branches. If we have just one branch out of a total of 5 branches after pruning, we can add the next node to 4 branches. However, the beam size is 3, so we must calculate the tree scores of the remaining 4 possibilities and select the top 3.

The details of the pruning algorithm are as follows. Suppose that we are now in the $(n + 3)$th iteration of inference and that we have reconstructed the phylogenies of nodes 1, 2, ..., n. In the next step, we add node $(n + 1)$ to the current phylogeny. For pruning purposes, we first calculate all possible quartet trees having nodes 1, 2, ..., $n + 1$, select the top-n trees from them, and mask the edges according to these top-n trees.

A schematic of this procedure is shown in Figure 2. For example, we have already reconstructed the phylogeny '((A, B), C, D)', and we need to add E to this tree with 4 leaf nodes to obtain a tree with five leaf nodes. Then, we can use the phylogenetic information of quartets (A, B, C, E), (A, B, D, E), (A, C, D, E) and (B, C, D, E) to mask some branches. For example, if ((A, B), D, E) is a quartet topology possessing a top-3 probability, then we mask all edges between A and B. If we add E to the edges between A and B, the sampled quartet tree of A, B, D, E cannot be ((A, B), D, E), which has a very low tree score. In our experiment, we set the top 10% (instead of the top 3 setting in the schematic) as the threshold for selecting the phylogenetic tree to mask branches. We compared our pruning algorithm to the algorithm without pruning (Figure 3A) and observed a similar performance, which supports the

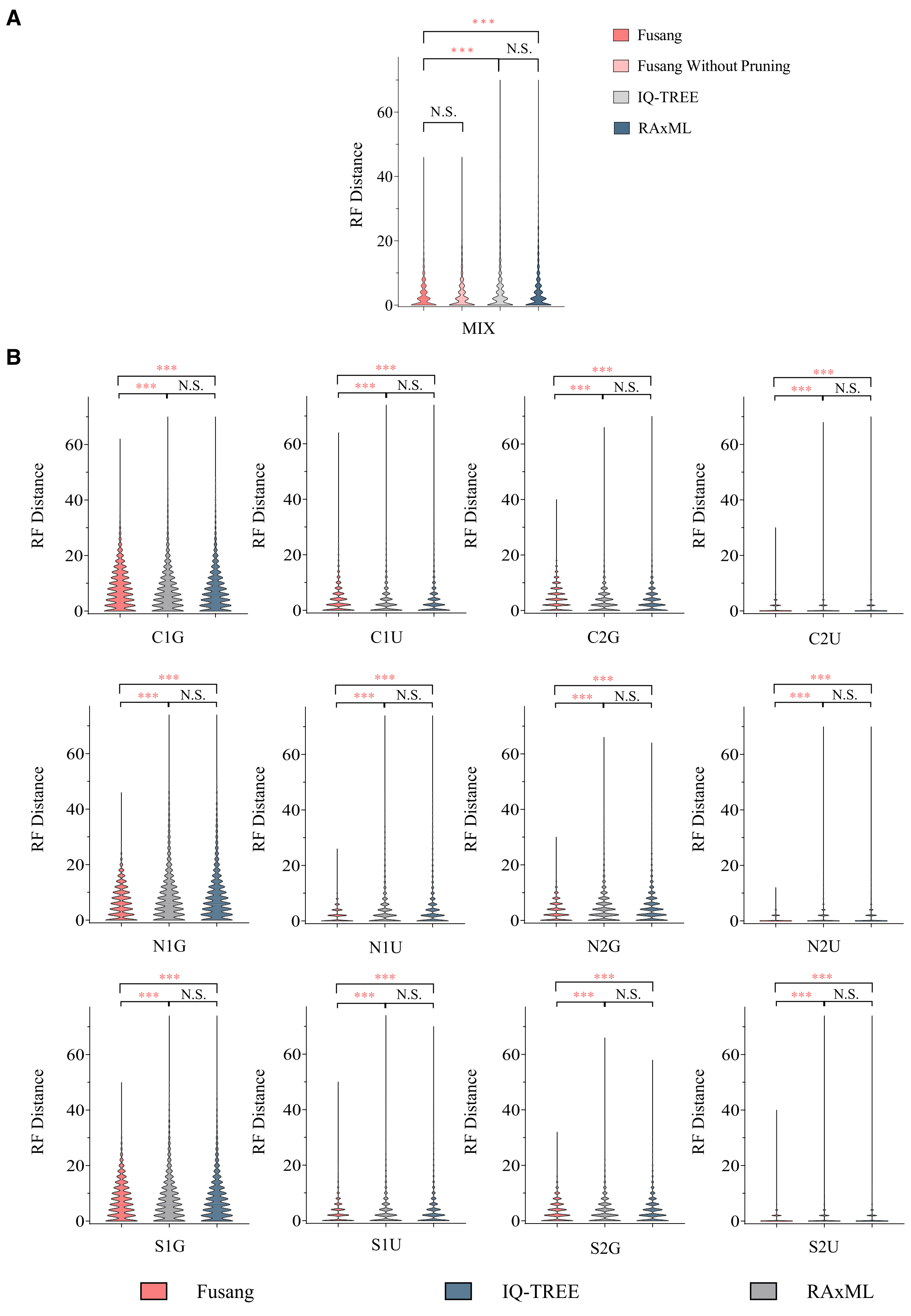


Figure 3. Performance achieved on simulations by the Fusang, IQ-TREE, and RAxML algorithms. (**A**) Performance achieved on simulated data by the Fusang, Fusang without pruning (default version), IQ-TREE and RAxML algorithms, where the data are a mixture of all 12 scenarios. Violin plots are drawn from the overall RF distances obtained on the simulated data by the Fusang, Fusang without pruning, IQ-TREE, and RAxML algorithms versus the different algorithms for each case based on the *P* values of the simulated RF distances (two-sided Mann−Whitney *U* test). (**B**) Performance achieved on simulated data by the Fusang, IQ-TREE, and RAxML algorithms for each of the 12 scenarios (S1G, S2G, S1U, S2U, N1G, N2G, N1U, N2U, C1G, C2G, C1U and C2U). Violin plots are drawn from the overall RF distances obtained on the simulated data by the Fusang, IQ-TREE and RAxML algorithms for each of the six simulated scenarios versus the different algorithms for each case based on the *P* values of the simulated RF distances (two-sided Mann−Whitney *U* test). Note: NS: $P > 0.05$; *: $0.01 < P \leq 0.05$; **: $0.001 < P \leq 0.01$; ***: $P \leq 0.001$.

validity of our pruning design.

In summary, in our search for a phylogeny with arbitrary leaf nodes, we use a modified stepwise addition algorithm to add nodes individually. To avoid utilizing a massive search space (which is an NP-hard problem), we use the beam search process, and to accelerate the beam search, we use a pruning algorithm to mask several edges in every iteration. Therefore, after every iteration, we have fewer than or exactly three candidate trees for the next iteration. A comparison between Fusang and Fusang without pruning is shown in Figure 4, which shows the accelerated performance provided by pruning.

Although we utilize the pruning algorithm, when there are too many species (>40), the computational efficiency of the current Fusang framework is low. Therefore, for these cases, we combine Fusang with divide-and-conquer algorithms that integrate the information of the reconstructed subtrees and the distance matrix to generate a complete large tree (23,24).

The pseudocode of our algorithm is as follows.

Fusang validation

We evaluated the performance of Fusang from the following angles. First, we tested the classification accuracy of the DL model on 12 datasets of '[SCN][1–2][UG]' and calculated the classification accuracy of Fusang in all 12 scenarios. In addition, the accuracy formula used to evaluate the performance of the Fusang DL model is as follows.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{8}$$

In the formula above, TP = the number of true-positive cases, TN = the number of true-negative cases, FP = the number of false-positive cases, and FN = the number of false-negative cases.

Furthermore, two standard scenarios (S2U and S2G, with all 4-leaf-node data and a fixed length of 240 or 1200) were selected to compare the 4-leaf-node inference accuracies of Fusang and previous methods (Tree_learning, IQ-TREE, and RAxML).

Second, the default version of the whole Fusang framework was tested when the number of leaf nodes was less than 40. We used simulation data with variable leaf nodes and variable sequence lengths, and by default, each parameter setting contained 27 200 MSAs, covering 29 multispecies protocols (4–30, 35 and 40, respectively). In this case, we used the RF distance as a criterion for judging the accuracy of the general phylogenetic tree inference results. Moreover, Fusang was combined with NJMerge-2 (23) to reconstruct the 100-leaf-node tree to verify the scalability of Fusang while comparing it with IQ-TREE and RAxML in terms of their inference results to assess the performance of Fusang.

Third, we tested examples of real biological data, including data from public databases and real trees generated by artificial mutations. Among them, the inferred phylogenies of 10 10-leaf-node subtrees were compared, while Fusang and NJMerge-2 were combined to reconstruct 100-leaf-node trees. In addition, we performed phylogenetic tree inference on rice and butterfly species, and the results were compared with those of IQ-TREE and RAxML to judge the performance of Fusang. Based on this assessment, we tested the performance of the default version of Fusang and a specific version where new parameters were trained for the scene to judge Fusang's adaptability. Finally, we tested Fusang's speed in typical scenarios, evaluating the running time required to build phylogenetic trees under different species and comparing the results with those of IQ-TREE and RAxML to judge Fusang's computational efficiency.

Results

Design principles

Fusang represents an integration and upgrade of previous efforts regarding the development of DL frameworks for phylogenetic tree inference (11,12). Figure 5 illustrates the design principle for Fusang, including the process of simulating training datasets (Figure 5A), a schematic of the DL architecture (Figure 5B), and the strategy used for phylogenetic tree inference (Figure 5C). Altogether, Fusang consists of two functional components, a DL model training process involving quartet trees with fixed sequence lengths and a phylogenetic tree inference procedure of MSA with varying numbers of sequences and sequence lengths.

To simulate the training datasets, we started by solving the phylogenies of 4 sequences, which were treated as a classification problem since only three unrooted topologies were available: ((A, B), C, D), ((A, C), B, D) and ((A, D), B, C). We used supervised learning with an MSA as input and the corresponding topology as the label in the form of a one-hot vector that represented the probability distributions of the three topologies. Then, we trained Fusang to learn a mapping from the MSA to the target probability distributions of the three possible phylogenies. To generate an unbiased training dataset, we used a simulation strategy (Figure 5A and Materials and Methods) identical to that employed in previous work (11,12,18) with modified parameters. Generally, we predefined 12 test scenarios with the three-letter naming rule in the form of the regular expression '[SCN][1–2][UG]'. First, S (both coding and noncoding), C (coding), and N (noncoding) represent the simulation of general, coding, and noncoding sequences, respectively. These three statuses were defined by the mean pairwise divergence measure and the ratio of indel to substitution (I/S ratio) (Table S3). Second, 1 (200 bases) and 2 (1000 bases) represent the length settings of the 'root' sequence for simulation purposes. The resulting sequences were filled with gaps or truncated to 240 or 1200 bases, respectively. Third, U (uniform distribution) and G (gamma distribution) represent the types of distributions on which we tested branch length sampling. More details of our simulation are provided in the Materials and Methods section.

Regarding the DL model, in previous studies, the classic convolutional neural network (CNN) (25) and CNN-based residual neural network (ResNet) (26) were used to extract features from MSAs to generate the best topologies. The schematic of Fusang's DL model is shown in Figure 5B, which includes a CNN module for extracting phylogenetic features and a recurrent neural network (Bi-LSTM) (27) module for extracting the sequential features of MSAs. The aim of Bi-LSTM is to learn the order and sequential information of MSAs, so we did not perform data augmentation to shuffle the order of the 4 sequences in the same MSA, which would have led to redundant labeling as in PhyDL (12). Meanwhile, the addition of the Bi-LSTM structure can ensure that there are more parameters to improve the learning ability, and the bidirectional recurrent network structure of Bi-LSTM can address the issue of results bias due to different orders of inputs, thus

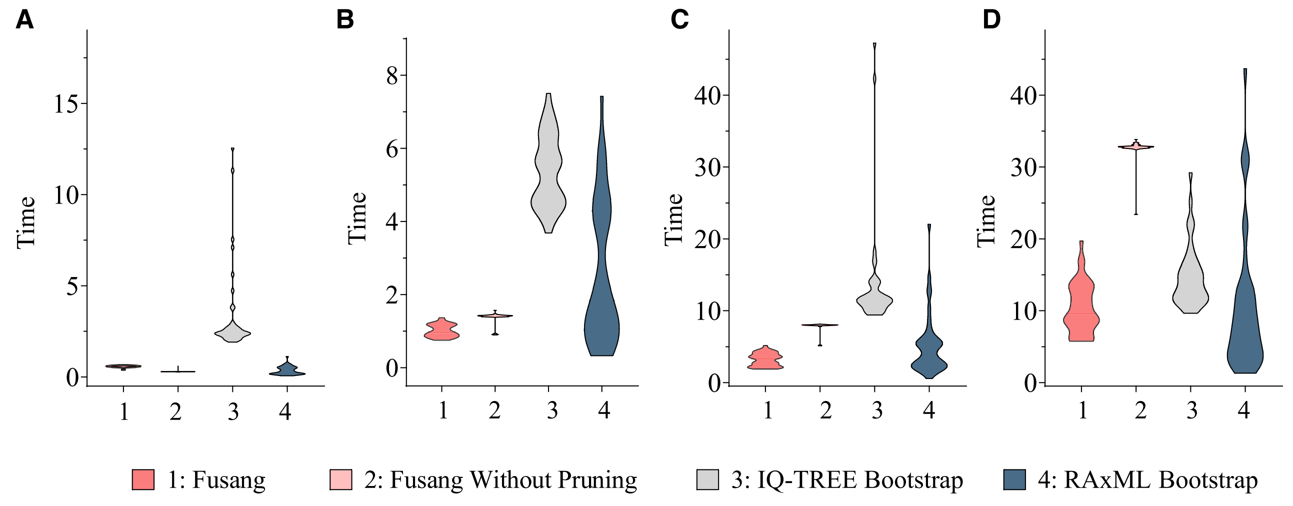


Figure 4. Running times of Fusang, Fusang without pruning (default version), IQ-TREE with bootstrapping and RAxML with bootstrapping for different leaf nodes. (**A**) Running times for 10-leaf-node trees; (**B**) running times for 20-leaf-node trees; (**C**) running times for 30-leaf-node trees; and (**D**) running times for 40-leaf-node trees.

```
Algorithm
Input: $A$: $C_n^4$ probability distributions of quartet trees sampled from $n$ leaf nodes
Output: $T_n$: A phylogenetic tree with $n$ species
/* Define a function for evaluating the tree score of a candidate tree */
function$Score(C_i, N_i)$
$score = 0$
Sample all quartet trees that have node $N_i$ from $C_i$
/* $C_i^3$ trees in total */
Sum the log probabilities of these trees, and assign this value to $p$
$score = p$
return $score$
end function
Initialize$Nodes = [1, n]$
/*The list of nodes that need to be added to the phylogenetic tree*/
Start with $T_4$
/* Select the quartet tree with the highest probability in $A$ */
Remove nodes from $Nodes$ if the nodes are already in $T_4$.
The tree score $S_4$ of $T_4$ is equal to its probability.
/* Reconstruct $T_5$ to $T_n$ iteratively */
initialize candidate tree beam$CTB_4$
add $(T_4, S_4)$ to$CTB_4$
for $i \in [5, n]$ do
initialize candidate tree beam$CTB_5$
$node = Nodes[i]$
/* Beam search */
for every tree $(T, S)$ in the candidate tree beam $CTB_{i-1}$
for each$branch \in T$
add the node to the current branch, and obtain candidate tree $C_i$
calculate its score, $S_{C_i} = Score(C_i, node) + S$
add $(C_i, S_{C_i})$ to$CTB_i$
Only keep the trees with the top-n tree scores in $CTB_i$
/* n equals the size of the beam */
Remove $node$ from $Nodes$
The final phylogenetic tree is the tree with the highest tree score in $CTB_n$
```

improving the performance of the model. Both features were integrated for the final phylogenetic tree inference process. Our CNN structure was the same as that of Tree_learning (11), and three-layer Bi-LSTM (27) was used for our module. Finally, two feature vectors were concatenated and fed into a fully connected network to output a predicted topology distribution. In contrast with the techniques of previous studies, Fusang integrates both CNN and Bi-LSTM models, while Suvorov et al. and Zou et al. used different CNN models.

The structures of the CNN and Bi-LSTM modules are shown in Figure 1A and address the limitations of ML approaches (Figure 1C). The shape of the input tensor was $4 \times L$, corresponding to a quartet MSA with a fixed length L. We represented each MSA as a matrix with rows corresponding to the sequence and columns corresponding to the loci in the alignment, with each character encoded as a number (‘A’: ‘0′, ‘T’: ‘1′, ‘C’: ‘2′, ‘G’: ‘3′, ‘-’: ‘4′, ‘N’: ‘4′). The CNN model consisted of 8 convolutional layers: the first convolutional layer

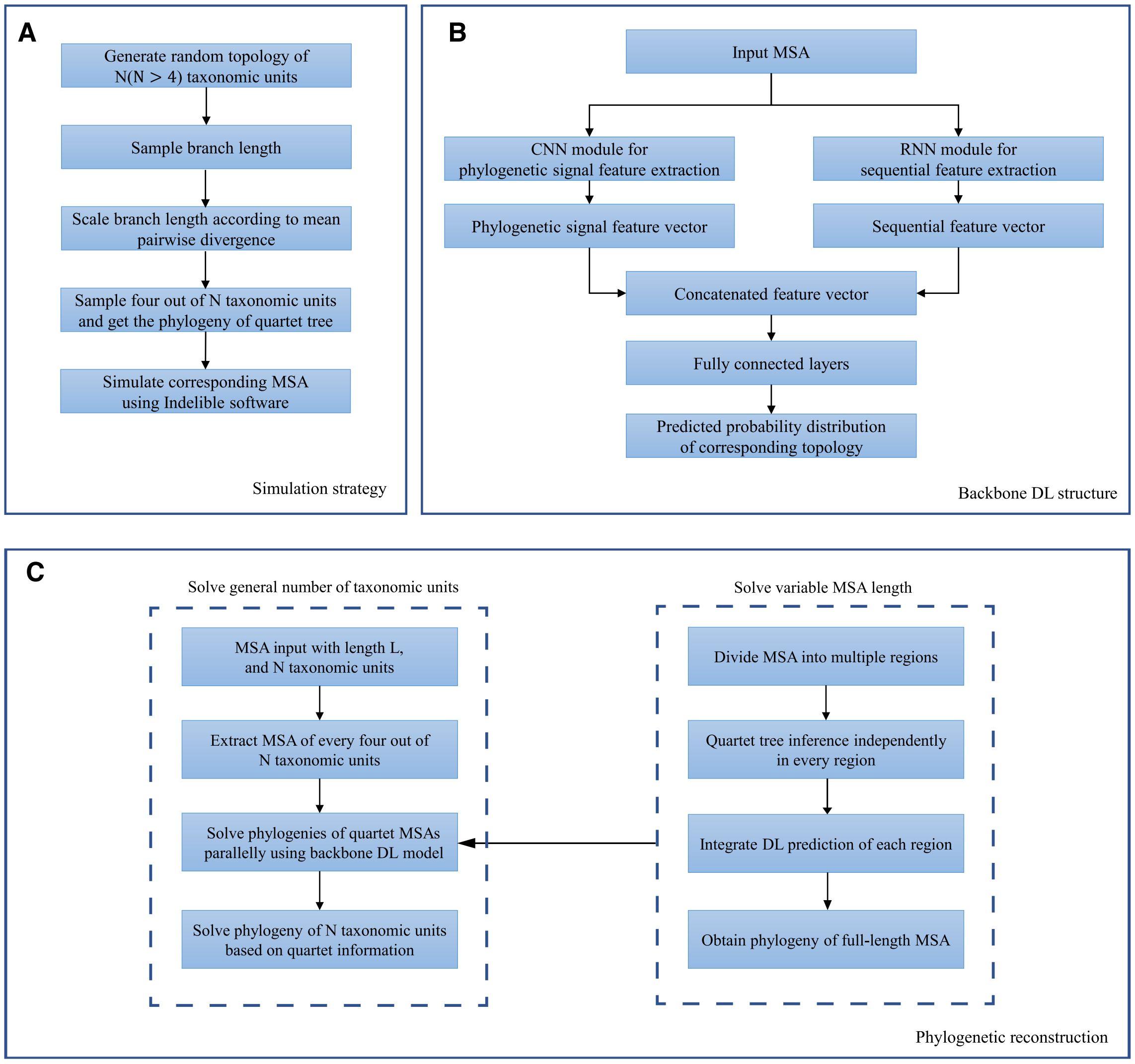

A
Generate random topology of N(N > 4) taxonomic units
Sample branch length
Scale branch length according to mean pairwise divergence
Sample four out of N taxonomic units and get the phylogeny of quartet tree
Simulate corresponding MSA using Indelible software
Simulation strategy
B
Input MSA
CNN module for phylogenetic signal feature extraction
RNN module for sequential feature extraction
Phylogenetic signal feature vector
Sequential feature vector
Concatenated feature vector
Fully connected layers
Predicted probability distribution of corresponding topology
Backbone DL structure
C
Solve general number of taxonomic units
MSA input with length L, and N taxonomic units
Extract MSA of every four out of N taxonomic units
Solve phylogenies of quartet MSAs parallelly using backbone DL model
Solve phylogeny of N taxonomic units based on quartet information
Solve variable MSA length
Divide MSA into multiple regions
Quartet tree inference independently in every region
Integrate DL prediction of each region
Obtain phylogeny of full-length MSA
Phylogenetic reconstruction

Figure 5. Design principle of Fusang. (**A**) Strategy for simulating training datasets; (**B**) schematic diagram of the DL model of Fusang using a CNN combined with Bi-LSTM and (**C**) strategy for the general phylogenetic tree inference case based on the stepwise addition algorithm and the sliding window algorithm.

was set to 4 × 1, and the sizes of the subsequent seven convolutional layers were set to 1 × 2. The total number of filters for the first and second convolutional layers was 1024, and the number of remaining convolutional layers was set to 128. For a model of length 1200, the first average pooling layer had a size of 1 × 1, followed by 3 average pooling layers with sizes of 1 × 4, another 3 average pooling layers with sizes of 1 × 2, and finally an average pooling layer with a size of 1 × 1. For a model with a length of 240, the first average pooling layer was set to 1 × 1, and the next seven layers of average pooling steps were equal in size (all set to 1 × 2). The specific CNN structure is shown in Figure 1B. The Bi-LSTM module consisted of three Bi-LSTM networks, with each unidirectional unit set to 128, and the results of each Bi-LSTM were fed into the next one until the last Bi-LSTM output the final result. The results of the CNN and Bi-LSTM modules were concatenated via a fully connected layer with ReLU activation and 1024 nodes. The output layer had three nodes corresponding to each of the three possible topologies. In our experiment, we set L to 240 for six '[SCN]1[UG]' scenarios and 1200 for six '[SCN]2[UG]' scenarios. These lengths were 1.2 times the root lengths in INDELible (18) for hosting simulated insertions.

For the model with a length of 1200, the shape of the initial input layer of the CNN was (4, 1200, (1), and the output shape of the eight-layer CNN was changed to (1,2, 128). To input it into the fully connected network with the Bi-LSTM network later, we called the Flatten function to process the output result into a shape of (256). The shape of the initial input layer of the Bi-LSTM network was (4, 1200), and the output shape of the three-layer Bi-LSTM network became (256). Then, we connected the CNN output processed by the Flatten function with the Bi-LSTM network output; at this time, the shape was (512). In the hidden layer of the fully connected network, the data shape was changed to (1024), and finally, shape (3) was output. For the model with a length of 240, the shape of the initial input layer of the CNN was (4, 240, (1), and the shape of the output obtained through the 8-layer CNN network was changed to (1, 2, 128). To input it into the fully connected network with the Bi-LSTM network later, we called the Flatten function to process the output result into a shape of (128).

The shape of the initial input layer of the Bi-LSTM network was (4, 240), and the output shape of the three-layer Bi-LSTM network became (256). Next, we connected the CNN output processed by the Flatten function with the Bi-LSTM network output, at which time the shape was (384). In the hidden layer of the fully connected network, the data shape was changed to (1024), and finally, shape (3) was output. The DL network outputs a three-dimensional vector, which is a probability distribution of the three possible topologies. The shape of the specific network output of each layer in the Fusang algorithm can be found in Table S6.

For phylogenetic tree inference, we used the strategy shown in Figure 5C to handle MSAs with variable numbers of sequences and sequence lengths for general phylogenetic tree inference applications. To reconstruct multiple sequences, we used a different version of the stepwise addition algorithm (9). In brief, we started from a randomly selected quartet phylogeny and iteratively added the remaining species. For each iteration, we added the target sequence to a different branch of the unrooted tree, scored all possible quartet subtrees of the candidate topology using our DL model, and kept the topology with the highest score, which was the sum of all quartet tree probability scores, until we had constructed the whole phylogenetic tree. For MSAs with variable lengths, we used a sliding window algorithm (Materials and Methods: Phylogenetic tree inference with variable MSA lengths). We divided the original MSA into sub-MSAs with the same length as our DL model, scored the candidate phylogeny based on the sub-MSAs, and used the average score.

Notably, Fusang can be customized to fit different scenarios by using different simulated training datasets. Here, we provide an initial version for common applications. This default version is designed to handle MSAs with fewer than 40 sequences and 10 000 nucleotides. To address MSAs with more than 4 sequences, we combined Fusang with the divide-and-conquer algorithm (23,24), which is different from the quartet puzzling of a previous study (12) concerning phylogenetic tree inference.

Classification accuracy of the quartet DL model

First, we compared Fusang with other methods in two standard scenarios (S2U and S2G). Fusang (97.30% for S2U and 90.70% for S2G) had similar accuracies to those of Tree_learning (97.30% for S2U and 90.68% for S2G) and outperformed IQ-TREE (89.51% for S2U and 82.05% for S2G) and RAxML (89.27% for S2U and 81.68% for S2G). This indicates that our DL architecture is powerful in terms of classifying 4-leaf-node phylogenies.

Second, we tested the generalization abilities of different Fusang models by applying the models to different data scenarios. Specifically, we divided all 12 models into 6 long-length models and 6 short-length models and tested their generalization capabilities. We tested the 6 long-length models with all six long-length data scenarios and obtained a 6 × 6 matrix as a result (Table S7 sheet 1; Long-Length Model). Similarly, we tested the 6 short-length models with all six short-length data scenarios and obtained a 6 × 6 matrix as well (Table S7 sheet 1; Short-Length Model). In general, Fusang produced better classification results for long sequences and the noncoding case (which has more indels), both of which provide more sequence divergence information. In addition, most scenarios yielded promising classification performance, supporting the good generalizability of the model, and no large differences were observed overall when choosing different models (all P values > 0.2, Table S7 sheet 2). This indicates that model selection does not greatly affect the resulting model performance under the settings of the scenario. Therefore, we selected S1G and S2G as default models for short and long sequences, respectively, due to their maximal average DL classification accuracies (Table S7 sheet 3).

Third, one important advantage of Fusang is the use of indel information. To investigate how indel information benefits the phylogenetic tree inference process, we trained Fusang as well as Tree_learning with and without indel information in default S2G dataset so that Tree_learning models can be trained as well. Finally, we obtained four models, including Fusang S2G Indel, Fusang S2G Noindel, Tree_learning S2G Indel and Tree_learning S2G Noindel. The Noindel training set was generated from the indel training set by removing indels, so these two sets shared identical phylogenies. The loss function is shown in Figure S7, and finally each model reached the optimum, e.g. the EarlyStopping state. We then applied the four models on the corresponding test sets as well as the likelihood methods. The results are listed in Table S8. As shown in Table S8, we compare the prediction accuracies of Fusang, Tree_learning, IQ-TREE, and RAxML utilizing datasets with and without indel information. ML methods such as RAxML usually treat indels as undetermined characters in the MSA, which may result in the loss of valuable evolutionary information related to indels. As expected, their changes in the performance of the different methods between MSAs with or without indels are relatively smaller than those of DL methods, which largely improve when indel information is added. Moreover, the accuracy of the model trained with indel information is much higher than that of the model trained without indel information. These results clearly support that indel information significantly increases the accuracy of phylogenetic inference and that our indel simulation is effective, although further exploration will be needed to determine what the best indel simulation is that we can achieve. In contrast, it is difficult to extract divergence times from single nucleotide changes due to reverse mutation, and indels are troublesome for likelihood calculations (9).

Fourth, to further verify the advantage of integrating Bi-LSTM with a CNN, we tested Fusang and Tree_learning on the same MSAs with different sequential inputs. Notably, Fusang's CNN module is identical to that of Tree_learning, but Fusang has an extra Bi-LSTM module. In total, we simulated 100 cases of long-length quartet trees under the S2G parameters. Each quartet tree yielded 24 ($A_4^4 = 4! = 24$) combinations of sequence input orders. As a result, we detected incorrect topologies in 10 out of 100 test cases. Among them, Fusang yielded fewer incorrect topologies than Tree_learning in six cases, more in two cases and equal quantities in two cases (File S1). According to the test results, Fusang has marginal advantages in terms of sequential feature extraction. This marginal effect can be enlarged when dealing with more sequences and longer sequences that include more information.

In conclusion, Fusang achieved very good performance with simulated quartet trees and exhibited two major improvements. First, Fusang was able to use indel information that is inapplicable in ML-based tools. Second, Fusang integrated Bi-LSTM and CNN modules and obtained marginal advantages in sequential feature extraction.

General phylogenetic tree inference performance

First, we tested the performance of two default models (S1G and S2G) using a mixture of data from all 12 scenarios with 1740 MSAs from 29 multileaf-node (4–30, 35 and 40 leaf nodes) datasets. For each dataset, we simulated 5 MSAs for each parameter set and 60 MSAs in total. Figure S8 shows the distributions of the number of sequences and length for different parameter settings. Fusang significantly outperformed IQ-TREE and RAxML in terms of accuracy, while Fusang without pruning did not change the accuracy significantly (Figure 3A). This result supports the notion that the default model works well for general cases. Second, we conducted separate testing with datasets derived from 12 scenarios using the 12 corresponding models with larger datasets, which means the name of used model is the same as the name of test scenario. For this test, each parameter setting contained 27 200 MSAs. Figure S9 shows the distributions of the number of sequences and length for different parameter settings. As shown in Figure 3B, Fusang achieved better performance in all 12 scenarios, and no significant difference was observed between the two ML methods. The RF distances and the properties of the test data can be found in Table S9.

To reduce the computational cost, we integrated a heuristic search algorithm, i.e. NJMerge-2, into the phylogenetic tree inference process (23,24) for the simulated cases of a 100-leaf-node tree (https://github.com/Jerry-0591/Fusang/tree/main/public%20data/Fusang_100_taxa). We divided the original 100-leaf-node MSA into ten 10-leaf-node sub-MSAs, reconstructed 10-leaf-node subtrees, and calculated the distance matrix of the original MSA using Bio. Phylo package (28) in Biopython (29) and integrated the information to reconstruct the whole tree. In this case, the performance of Fusang was slightly worse than that of the ML-based methods in terms of the mean RF distance, but its standard deviation was smaller, as shown in Figure S10, indicating that the current version of Fusang is suitable for MSAs with fewer than 40 leaf nodes.

Performance with real biological data

Although Fusang has very good performance with simulated datasets, it is more important to assess its performance with real datasets. To fully evaluate Fusang, we tested it on extreme cases with large sequences or high levels of species divergence, a 100-leaf-node tree with long sequences, genome- and transcriptome-level analyses and a real phylogeny generated by experimental evolution.

First, we tested the extreme cases extracted from the Pfam database (version 35.0) (30). We collected 9444 MSAs with fewer than 40 leaf nodes and sequence lengths below 10 000 from Pfam. Then, we defined two datasets of extreme cases: the top 100 trees with the highest standard deviations for sequence divergence and the top 100 trees with the highest standard deviations for species divergence, which were extracted from TimeTree (31) (more details are listed in the supplementary materials). As shown in Figure S11 and Table S10, Fusang achieved comparable performance to that of the ML-based methods, which supports its ability to deal with extreme cases.

Second, we performed tests on a vertebrate species tree (100 sequences, MSA for chromosome one, 352 245 148 bp) collected from the UCSC database (http://hgdownload.cse.ucsc.edu/goldenpath/hg38/multiz100way/). Initially, we constructed ten 10-leaf-node subtrees and found that the RF distance of Fusang was slightly greater than those of IQ-TREE and RAxML (Table S11). However, both ML-based methods failed to generate the 100-leaf-node tree due to insufficient RAM error produced by our server with 1.5 TB of RAM (File S2). Instead, we combined Fusang with NJMerge-2 (23) to successfully generate a 100-leaf-node tree with an RF distance of 66. It outperformed the neighbor joining algorithm (7), whose RF distance was 90.

Third, we performed phylogenetic tree inference at the genome and transcriptome levels, including for *Oryza* (11 sequences, whole-genome alignment, 359187370 bp), butterflies (16 sequences, MSA for chromosome twenty-one, 13 580 644 bp), insects (4 sequences, whole-genome alignment, 133 202 631 bp), and vertebrates (8 sequences, whole-genome alignment, 463 540 596 bp). As shown in Figure S12, Fusang output similar but different phylogenies from those of the ML-based methods, with RF distances of 2 for *Oryza* and butterflies, 0 for insects and 4 for vertebrates. Notably, Fusang's vertebrate tree was identical to the UCSC tree (https://hgdownload.soe.ucsc.edu/goldenPath/fr3/multiz8way/fr3.8way.nh). In addition, we tested transcriptome data consisting of 6528 genes conserved in 11 *Oryza* species. As shown in Figure S13, Fusang was slower than RAxML but faster than IQ-TREE. Additionally, Fusang outputs significantly different topologies from those of both ML-based methods, although the true topologies are unknown.

To determine a precise ground truth, we conducted evolutionary experiments in the laboratory. The details of our experimental design, sequencing results (File S3), and test data are listed in the supplementary materials (the 'Experimental phylogeny' section). In this part, we tested two Fusang models: the default S2G model and a customized model. For the customized model, we used the mean pairwise divergence and I/S ratio as scales to generate a new training dataset based on the parameters (Table S12) estimated from experimental data. As shown in Table S13, this model achieved a smaller RF distance than the S2G model and ML-based methods. The inference result of different methods is shown in Figure S14, and the numbers of mutations, insertions and deletions observed during the experiment are shown in Table S14. According to the results (Table S13), Fusang can evolve to handle specific scenarios with customized training datasets. More details can be found in the supplementary materials section 'Experimental phylogeny' including the true experimental phylogeny (Figure S15), schematic of simulation for training customized model (Figure S16), analysis of experimental data (Figure S17), and loss curve of training customized model (Figure S18).

Computational time analysis

Here, we discuss the required computational time from three aspects: the training time, the time needed to generate the training dataset, and the time efficiency of the phylogenetic tree inference process in typical scenarios.

First, the details of the training times required by Fusang in all 12 scenarios are listed in sheet 1 of Table S15, which includes the total training time, the average training time per epoch, the number of training epochs, and the amount of data in the dataset. The training times of all short-length Fusang models were quicker than those of the long-length Fusang models due to their smaller model sizes.

Second, we performed an analysis on the dataset containing all 12 scenarios to determine the time required for generating training data. Each scenario contained 10 000 samples. For this analysis, we calculated the time needed to generate the phylogenetic trees, simulate the corresponding MSAs, and finally encode them in the '(data, label)' format, which can be directly used for DL training. The results are shown in sheet 2 of Table S15. On average, across all 12 scenarios, the proposed method needed ∼0.0752 s to generate a '(data, label)'-formatted sample for deep learning model training.

In addition, to determine the efficiency of the whole phylogenetic tree inference process, we simulated four independent typical datasets to test the speed of Fusang in different scenarios: D1 (10 species, 5616 bp on average), D2 (20 species, 6577 bp on average), D3 (30 species, 6750 bp on average), and D4 (40 species, 6791 bp on average). Each contained 40 MSAs. We then compared the running times of Fusang, IQ-TREE and RAxML. As shown in Table S15 sheet 3, Fusang achieved comparable computational efficiency for cases with fewer than 20 species. As the number of species increased, a slight increase was observed in the running time of Fusang due to the properties of the stepwise addition algorithm and the beam search procedure. In addition, bootstrapping generally must be performed when heuristic searches are performed for ML methods (32). If we take this into account, Fusang's running time is comparable to that of the maximum likelihood methods, as shown in Figure 4. More details regarding our test scripts and data used in the 'Performance evaluation' section can be found in the supplementary materials (Data and scripts).

Discussion

We wrote a theoretical derivation (Supplementary materials) to describe the statistical inference aspect of the Fusang framework that supports the notion that, statistically speaking, the goal with Fusang is similar to that with ML methods, but Fusang has the potential for handling specific evolutionary cases better by training specific DL models, as we have shown in the experimental phylogeny case. In addition, DL frameworks are potentially more accurate and efficient than ML frameworks when the training dataset fits well with the real data. In addition, when an MSA has more indels, such as those in our test data N1G and N2U, DL performs better than the ML methods. To investigate the effect of indel information on phylogenetic tree inference, we trained different models with or without indel information and found that all the models trained with indel information were more accurate than the models trained without indel information (Table S8). This is a great advantage of DL frameworks, which take indel information into account; in contrast, ML methods usually treat indels as undetermined characters that may result in the loss of useful evolutionary information. Fusang represents the first DL framework whose performance is comparable to that of traditional ML frameworks in general phylogenetic tree inference cases with variable MSA sequences and lengths.

Recent efforts (11,12) have laid a solid foundation for DL frameworks. Suvorov et al. showed that a DL framework works better than an ML framework for 4-leaf-node phylogeny. Furthermore, Zou et al. successfully constructed a 19-leaf-node tree based on amino acid sequences with an enhanced DL framework. However, neither approach is immediately available for phylogenetic tree inference in general cases. Fusang is able to handle cases with variable MSA sequences (≤40) and lengths (≤10K).

To address variable MSA sequences, Fusang utilizes a modified stepwise addition algorithm inspired by Zou et al. (12). In contrast with the previously developed quartet puzzling (13) algorithm, we used a beam search to find the best topology based on the probability distribution provided by DL. To address variable MSA lengths, Fusang adopts a sliding window to split the original sequence and summarizes the results by voting. This strategy is obviously more suitable than those used in previous methods that tend to lose information. More importantly, the voting process can run in parallel to yield improved speed.

Since the branch length calculation relies on the settings of the evolutionary parameters (33), the current version of Fusang provides only the tree topology. We recommend available tools such as RAxML-NG (34) for branch length calculations. An example case can be seen in the supplementary materials ('Data and scripts').

The current version of Fusang can only support most general phylogenetic tree inference cases. In regard to MSAs with strong discordant signals, such as introgression or sequences with extremely high or low divergence, specific training datasets must be further integrated. Fusang achieves computational efficiency comparable to that of IQ-TREE and RAxML when the number of MSA sequences is small. When the number of species is large and the MSA length reaches the genome level, Fusang exhibits a decrease in efficiency. Therefore, we set the limitations of (≤40 species, ≤10K MSA length) for the current version of Fusang. We will continue our efforts to improve Fusang so that it can cover more scenarios.

On a real data test, Fusang also produced relatively comparable results. For example, for the experimental phylogeny, Fusang significantly improved the result when it used the estimated scale from the experimental data, which supports the great potential regarding the applications of DL in phylogenetic tree inference.

As previously mentioned, Fusang's performance can be significantly improved with scenario-specific training dataset simulations. One important optimization of Fusang is that it should independently simulate training datasets for different scenarios. In our research, we mainly treated the mean pairwise divergence and I/S ratio as scales to simulate different evolutionary cases and demonstrate the ability of Fusang to reconstruct different scales. We suppose that with a more specific and advanced simulation strategy, Fusang could be used for more specific phylogenetic tree inference tasks. For example, with a suitable training dataset, Fusang could outperform traditional ML frameworks in tumor cell evolution cases since tumor evolution is entirely different from most predefined evolutionary models. Additionally, optimizing the model structure and branch adding algorithms would contribute to making Fusang more accurate and efficient.

Data availability

The source code for the DL training and phylogenetic tree inference processes can be downloaded from https://github.com/Jerry-0591/Fusang. Figshare repository: https://figshare.com/articles/software/Fusang_source_code/23921916. The hardware and software environment requirements are listed in https://github.com/Jerry-0591/Fusang/blob/main/Environment_setting.md. The command line software and online server are available at (http://fusang.cibr.ac.cn/). More details about the source data and scripts can be found in the supplementary materials ('Data and scripts').

Supplementary data

Supplementary Data are available at NAR Online.

Acknowledgements

We thank Dr Yong E. Zhang at the Institute of Zoology, Chinese Academy of Sciences for his insightful suggestions. We thank the Computational Core and Genomics Center at the Chinese Institute for Brain Research, Beijing, for their technical support.

Author contributions: L.Z. conceived the idea and designed the study. Z.C.W. and J.N.S. wrote the code and performed the analyses, including data simulation, model training and results validation. Y.W.X. and Z.C.W. built the online server. Y.G. generated the real phylogeny with evolutionary experiments. Y.B.Z. and K.L. collected publicly available phylogenetic datasets. J.Z., C.Z., Z.C.W. and J.N.S. generated and confirmed the mathematical part. L.Z., J.N.S., Z.C.W., J.Z., C.Z. and W.Z. wrote the paper. All the authors have read and approved the final manuscript.

Funding

Chinese Institute for Brain Research Internal Funding and the National Natural Science Foundation of China [32170642 to L.Z.]; Shaanxi Fundamental Science Research Project for Mathematics and Physics [22JSY039 to J.Z.]; National Natural Science Foundation of China [11971375 to J.Z.]; National Natural Science Foundation of China [32170420 to W.Z.]. Funding for open access charge: NSFC.

Conflict of interest statement

None declared.

References

1. Yang,Z. and Rannala,B. (2012) Molecular phylogenetics: principles and practice. *Nat. Rev. Genet.*, **13**, 303–314.
2. Darwin,C. (1909) *The Origin of Species*. PF Collier & son New York.
3. Kingman,J.F.C. (1982) On the genealogy of large populations. *J. Appl. Probab.*, **19**, 27–43.
4. Singh,N.D., Larracuente,A.M., Sackton,T.B. and Clark,A.G.(2009) Comparative genomics on the Drosophila phylogenetic tree. *Annual Review of Ecology, Evolution, and Systematics*. Vol. **40**, pp. 459–480.
5. Caravagna,G., Heide,T., Williams,M.J., Zapata,L., Nichol,D., Chkhaidze,K., Cross,W., Cresswell,G.D., Werner,B. and Acar,A.J.N.G. (2020) Subclonal reconstruction of tumors by using machine learning and population genetics. *Nature*, **52**, 898–907.
6. Sneath,P.H. and Sneath,P. (1962) *The Construction of Taxonomic Groups*. Cambridge University Press.
7. Saitou,N. and Nei,M.(1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol.*, **4**, 406–425.
8. Schwartz,R. and Schaffer,A.A. (2017) The evolution of tumour phylogenetics: principles and practice. *Nat. Rev. Genet.*, **18**, 213–229.
9. Yang,Z. (2014) *Molecular Evolution: A Statistical Approach*. Oxford University Press.
10. Kapli,P., Flouri,T. and Telford,M.J. (2021) Systematic errors in phylogenetic trees. *Curr. Biol.*, **31**, R59–R64.
11. Suvorov,A., Hochuli,J. and Schrider,D.R. (2019) Accurate inference of tree topologies from multiple sequence alignments using deep learning. *Syst. Biol.*, **69**, 221–233.
12. Zou,Z., Zhang,H., Guan,Y. and Zhang,J.(2020) Deep residual neural networks resolve quartet molecular phylogenies. *Mol. Biol. Evol.*, **37**, 1495–1507.
13. Strimmer,K. and Haeseler,A.v. (1996) Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, **13**, 964–969.
14. Makowsky,R., Cox,C.L., Roelke,C. and Chippindale,P.T.(2010) Analyzing the relationship between sequence divergence and nodal support using Bayesian phylogenetic analyses. *Mol. Phylogenet. Evol.*, **57**, 485–494.
15. Chen,J.-Q., Wu,Y., Yang,H., Bergelson,J., Kreitman,M. and Tian,D. (2009) Variation in the ratio of nucleotide substitution and indel rates across genomes in mammals and bacteria. *Mol Biol Evol.*, **26**, 1523–1531.
16. Felsenstein,J. (1978) Cases in which parsimony or compatibility methods will be positively misleading. *Syst. Biol.*, **27**, 401–410.
17. Huerta-Cepas,J., Serra,F. and Bork,P. (2016) ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Mol. Biol. Evol.*, **33**, 1635–1638.
18. Fletcher,W. and Yang,Z (2009) INDELible: a flexible simulator of biological sequence evolution. *Mol. Biol. Evol.*, **26**, 1879–1888.
19. Rumelhart,D.E., Hinton,G.E. and Williams,R.J. (1986) Learning representations by back-propagating errors. *Nature*, **323**, 533–536.
20. Kingma,D.P. and Ba,J. (2014) Adam: a method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.
21. Sbert,M., Chen,M., Poch,J. and Bardera,A. (2018) Some order preserving inequalities for cross entropy and Kullback–Leibler divergence. *Entropy (Basel)*, **20**, 959.
22. Szegedy,C., Vanhoucke,V., Ioffe,S., Shlens,J. and Wojna,Z. (2016) *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp.2818–2826.
23. Molloy,E.K. and Warnow,T. (2019) TreeMerge: A New Method for Improving the Scalability of Species Tree Estimation Methods. *Bioinformatics*, **35**, i417–i426.
24. Molloy,E.K. and Warnow,T. (2018) *RECOMB International conference on Comparative Genomics*. Springer, pp. 260–276.
25. Fukushima,K. and Miyake,S. (1982) *Competition and Cooperation in Neural Nets*. Springer, pp. 267–285.
26. He,K., Zhang,X., Ren,S. and Sun,J. (2016) *Computer Vision and Pattern Recognition*. pp .770–778.
27. Hochreiter,S. and Schmidhuber,J.J.N.c. (1997) Long short-term memory, *Neural Comput.*, **9**, 1735–1780.
28. Talevich,E., Invergo,B.M., Cock,P.J. and Chapman,B.A. (2012) Bio.Phylo: a unified toolkit for processing, analyzing and visualizing phylogenetic trees in biopython. *BMC Bioinformatics*, **13**, 209.
29. Cock,P.J., Antao,T., Chang,J.T., Chapman,B.A., Cox,C.J., Dalke,A., Friedberg,I., Hamelryck,T., Kauff,F., Wilczynski,B., *et al.*. (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.
30. 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., *et al.* (2021) Pfam: the protein families database in 2021. *Nucleic Acids Res.*, **49**, D412–D419.
31. Hedges,S.B., Dudley,J. and Kumar,S. (2006) TimeTree: a public knowledge-base of divergence times among organisms. *Bioinformatics*, **22**, 2971–2972.

32. Shen,X.X., Li,Y., Hittinger,C.T., Chen,X.X. and Rokas,A. (2020) An investigation of irreproducibility in maximum likelihood phylogenetic inference. *Nat. Commun.*, **11**, 6096.
33. Yang,Z. (2006) *Computational Molecular Evolution*. Oxford University PressOxford.
34. Kozlov,A.M., Darriba,D., Flouri,T., Morel,B. and Stamatakis,A. (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, **35**, 4453–4455.

Received: January 10, 2023. **Revised:** August 17, 2023. **Editorial Decision:** September 14, 2023. **Accepted:** September 20, 2023