Using Machine Learning Techniques for Phylogenetics

A report submitted for the course
COMP8604, Project Work in Computing

12 pt research project, S1 2021–2022

By:
Chen Yang

Supervisor:
Dr. Minh Bui

Australian National University

School of Computing
College of Engineering and Computer Science (CECS)
The Australian National University

October 2024

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the University Academic Misconduct Rules;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or Wattle site;
- is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling.

October, Chen Yang

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor Minh Bui as well as his PhD student Trong Nhan Ly. They gave me the opportunity to do this project. They also helped me a lot in solving theoretical and technical problems. I also thank Alina Leuchtenberger and Arndt von Haeseler for helpful advise on the F-zoneNN and providing access to the CIBIV clusters.

Abstract

Phylogenetics is the study of analyzing genes to infer evolutionary relationships among a set of species. Maximum likelihood and maximum parsimony are two typical methods for phylogenetic inference, but each of these two methods could perform well or badly for different phylogenetic tree types, resulting in inconsistent phylogenetic trees. This thesis trained a neural network to infer the phylogenetic tree type from a four-taxon alignment, which potentially provides insight into which tree reconstruction method is suitable to the alignment.

Key words: **Phyloginetic inference, machine learning, neural network, maximum likelihood, parsimony, sequence simulation**

Table of Contents

Chapter 1

Introduction

Phylogenetics trees, also known as evolutionary trees, are the basic structures depicting the evolutionary relationships among a set of organisms (Felsenstein (2004)). Many evolutionary processes that happened in the past are unobservable, therefore the main task of phylogenetics is to reconstruct phylogenetic trees from multiple sequence alignments, which represent the genomic data of present-day organisms1.1.

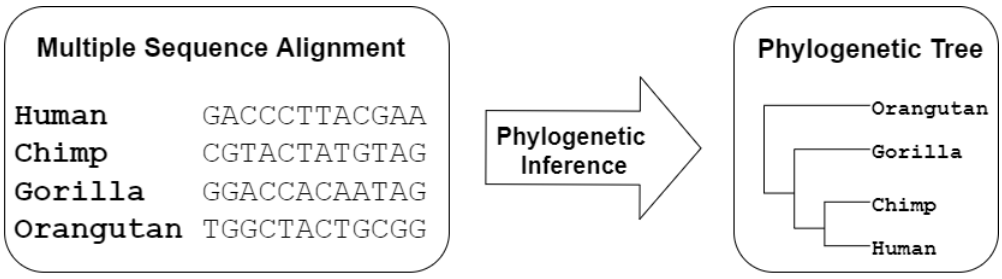

Figure 1.1: Phylogenetic inference process.

Real-world DNA sequences can vary in length, when comparing the DNA sequences of different species, many sites might be mismatched. A sequence alignment is a hypothesis about homology of multiple residues in protein or nucleotide sequences (Lemey et al. (2009)). Based on this fundamental assumption, character homology, aligned DNA sequences are required for many phylogenetics analysis. One common method is to insert gaps, which represent deletions or a combination of insertions and deletions, to the DNA sequences to make them match up. Therefore, the input data for this task is the whole-genome sequence alignments, or multiple sequence alignments (MSAs).

If the sequences accumulate a few substitutions, they will remain lots of similarities and can be relatively easy to align. But in reality, sequences can change considerably

and it can be hard to find pairs of similar nucleotides. The gaps and repeats of the sequences also cause problems, they can increase the ambiguity of pairs of nucleotides. Dynamic programming can be used to find the best possible alignment. Unfortunately, the time complexity grows exponentially with the number of species, so many heuristics or approximate alternatives are currently used for multiple sequence alignment to reduce the time complexity.

A model of evolution is indispensable for phylogenetic inference. Use of one model of evolution or another may change the results of a phylogenetic analysis (Sullivan et al. (2005)). Model of evolution is also named substitution model, which provides a statistical description of the mutation process. The model can be described with a time-homogeneous stationary Markov process. In general, a complex model with more parameters can fit the data better than a simple model, but if more parameters are estimated from the small amount of data, more errors may be included in each parameter. Therefore, a more complex model with more parameters may not increase the accuracy and produce more accurate estimation of branch-length. When the model assumed is wrong, branch-lengths may be underestimated, while the strength of rate variation among sites may be overestimated.

1.1 Phylogenetic Methods

Maximum Likelihood (ML) and Maximum Parsimony (MP) are two typical methods for phylogenetic inference (Felsenstein (2004)).

The principle idea of the Maximum Likelihood approach is to seek a phylogenetic tree and estimate the parameters of a substitution model that maximizes the probability of observing the given sequences. The best-fit model and tree topology can be selected using statistical techniques. The likelihood function is the conditional probability of the data given a model with a set of parameters and the tree topology with branch length (Lemey et al. (2009)). To find the best tree, we may need to take into account all possible trees in the tree space. The number of unrooted binary tree topologies increases factorially with the number of taxa, which means finding the best tree topology with ML is NP-hard (Chor and Tuller (2005)). Besides, the probability of getting the observed alignments does not assure the phylogenetic tree is true.

The basic idea of MP is to minimize the number of evolution changes (Lemey et al. (2009),Felsenstein (2004)). Doing phylogenetic analysis under parsimony method can be separated into two steps: first, determining the number of character changes required by any given tree; second, searching over all possible tree topologies for the trees minimize the number of changes. In parsimony analysis, the minimum cost of mutation, or genetic distance of each site can be calculated by evaluating all possible tree structures.

Summing the distance of each site to obtain the total tree length for each possible tree, and then choosing the tree that minimizes the total length. The number of possible tree topologies is factorial with the number of species. MP is also computationally expensive when the number of species is relatively large, but this method is less computationally demanding than ML. Another problem with this method is the assumption under parsimony analysis is unclear, performance of this method could be very good or extremely bad based on specific situation.

1.2 Machine Learning

Machine learning is a branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data. Traditional programming takes input data and human designed program to compute the output, while machine learning typically trained with input data and target output data to get a trained model to solve the task (BISHOP (2016)).

Neural Network (NN) is a machine learning algorithm, which uses a program to simulate the behaviour of biological neurons. Commonly humans design the architecture and topology of the network, and determine the training algorithm, while all network parameters (weights) are initialized randomly. Then the neural network can make small modifications to weights on each presentation of data during the training process. After many epochs of training, the neural network can learn and reproduce some properties of the training set. One big advantage of neural network is that once a network has been trained, it can produce output from the input fast, although the training process might be computationally expensive (BISHOP (2016)).

Random forest is another machine learning algorithm. It is a combination of decision tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges to a limit as the number of trees in the forest becomes large (Biau and Scornet (2016)). In random forest algorithm, the training set of each decision tree is sampled from the whole training set, and in each training process the tree is trained to make decisions only based on a few features instead of all features. Therefore, each tree predictor is a weak predictor, but when many trees are trained and their outputs are averaged, the random forest algorithm does not overfit, but produces a limiting value of the generalization error.

Compared with neural network, random forest method has some advantages. First, the input data could be high dimension data, reducing dimension or selecting features are not necessary, because in each step the trees only use a part of features to make the decision. Second, random forest algorithm is unlikely to overfit, because randomness

is introduced during the training process. Third, all trees can be trained in parallel to speed up the training process. However, random forest methods tend to fit the error if noise in the training data is relatively large. This problem potentially restricts the application of random forest in phylogeneyics, because the randomness in evolution can introduce noise in the data.

1.3 Existing Machine Learning Approaches to solve Phylogenetics Problems

Inferring phylogenetic trees with classical methods such as ML and MP is NP-Hard. Moreover, the ever-growing amount of genomic data makes many, if not all, existing heuristic search-based inference tools intractable. Machine learning has great potential to deal with that problem. Therefore, some machine learning-based methods have been introduced to deal with that problem.

The first step in developing machine learning-based methods is generating training and testing data. Simulated data has been widely used since the true phylogenies of empirical data sets are unknown. Therefore, sequence simulation plays an important role in phylogenetics. Typical sequence simulation algorithms require a phylogenetic tree and an evolutionary model as inputs to generate sequences which evolved along the tree based on the model of evolution. Seq-Gen is one popular sequence simulation tool proposed by Rambaut and Grass (1997). That program simulates a sequence evolving down each branch of the tree using a transition probability matrix. For a given branch length, this matrix describes the probability of a site changing from a state to other states or remaining unchanged. Moreover, the branch length of the tree is assumed to denote the mean number of nucleotide substitutions per site.

AliSim (Ly-Trong et al. (2022)) is a fast, memory-efficient, and realistic sequence simulator, which was recently introduced for simulating large-scale genomic data from hundreds of evolutionary models. To speed up the simulation, AliSim implemented an adaptive approach that combines the two conventional probability matrix and rate matrix approaches (Schoeniger and von Haeseler (1995)). Moreover, AliSim offers more realistic simulations that allows users to mimic the evolutionary processes of an MSA or generate model parameters from empirical distributions extracted from a large data set. Compared with Seq-Gen, AliSim supports many more evolutionary models, consumes less memory and time for simulating large data set.

PhyDL (Zou et al. (2019)) is a machine learning-based method for phylogenies inference based on a deep residual network. In maximum likelihood methods, choosing an inappropriate substitution model can lead to an inaccurate phylogenetic inference, so the

authors employed a deep residual network to infer a four-taxon tree topology directly from the input data without any assumption on the model of evolution. They simulated amino acid sequences from randomly generated trees for training and testing their network. The alignments were then encoded into one-hot vectors to feed into the network. Their model outputs 3 scores represent the likelihood that taxon 0 is a sister of taxon 1, taxon 2 or taxon 3, respectively. For four-taxon trees, this information can denote the topology of the tree. For the training process, their model converges fast for data set containing normal trees, but slowly for those trees containing long branch attraction (**??**). However, this model performs better than classical methods in different test sets, even the test data have different distribution or sequence length with the training data.

Another method to infer the tree topologies from multiple sequence alignments (Suvorov et al. (2019)) is based on convolution network (CNN). Different zones of parameter space can affect the accuracy of classical methods (ML and MP), so they employed a neural network model to infer the tree topology from an MSA. They simulated data set from trees with uniformly-sampled branch lengths. This model takes sequence alignments as input, and each character is encoded as a number (A:0, T:1, C:2, G:3, -:4). This model uses 2D convolution blocks to extract features, so they avoid encoding the input data to high dimensional data. Their research shows CNN are more accurate than classical methods on phylogenesis inference. But their model indicates the CNN model has limitations on this task, it may not perform that well for some difficult cases if the amount of training data is limited.

F-zoneNN (Leuchtenberger et al. (2020)) is a neural network used to distinguish Felsensetin zone from Farris zone. Here Felsenstein zone and Farris zone denote different combinations of branch-length parameters 1.2. The maximum parsimony method performs badly in the Felsenstein zone, while maximum likelihood performs badly in the Farris zone (Felsenstein (1978)). Therefore, they trained a simple NN model to distinguish Felsensetin zone from Farris zone from a given alignment. For the alignments with four taxa, each site has 256 unique site-patterns, but they collapsed input data to 15 pattern categories. Unlike the methods above, their model cannot output tree topology; instead, this model only helps to make a judgement. They claimed their method as a mixed method. They first employed the F-zoneNN to determine the tree type. If the tree is a Farris-type tree, they will apply the maximum parsimony method to reconstruct the phylogenetic tree, otherwise, if the tree is a Felsenstein-type tree, they will use the maximum likelihood method. Compared with classical maximum likelihood and maximum parsimony methods, the mixed method can generate more accurate and balanced result. Their research shows potential future research, which is the combination of machine learning-based methods and classical methods.

ModelRevelator (Burgstaller-Muehlbacher et al. (2021)) is a machine learning method for model selection. The ModelRevalator contains two individual neural networks. The first network is NNmodelfind that is based on ResNet-18 (He et al. (2016)). The in-

put alignments of this network are reshaped to 3D matrices like image data, because ResNet-18 is typically used for computer vision tasks. The NNmodelfind can select one substitution model for the input from six base models. The second network is NNalphafind. The input data of this network are sequence alignments, and this network uses bidirectional LSTM with an attention layer to process the data, while this network structure is typically used in natural language processing tasks. The NNalphafind can compute α parameter that denotes the distribution of branch-length. And this network can also estimate if the model has rate heterogeneity. Their result demonstrated that neural networks can marginally improve the accuracy and speed up the phylogenetic inference. It is interesting that they tried different network architectures. ResNet-18 did not perform well for α estimation, but the underlying reason is unclear, and finding some other network architectures that can produce better results is still an open question.

ModelTeller (Abadi et al. (2020)) is a method to predict the substitution model based on the Random Forest learning algorithm. ModelTeller can predict a single model, or predict a model for branch-length optimization with a fixed GTR+I+G topology, or predict a model for branch-length optimization with a fixed topology given by the user. ModelTeller shows higher accuracy on branch-length estimation compared with some existing methods, but the training data must be extended over a large range of possible realistic mutation processes so that the trained model is applicable for empirical data.

In conclusion, recent studies indicate some problems when applying machine learning to phylogenetic inference. First, to get better performance, the training data should be distributed to a large range, which can represent as more realistic mutation processes as possible. Second, a specific network architecture that adapts to phylogenetic tasks needs to be found.

1.4 Contributions of this Thesis

In this paper, I explored the method proposed by Leuchtenberger et al. (2020), and combined their method with Alisim, trained a neural network with the data generated by AliSim to distinguish between alignments derived from a Felsenstein-type tree and a Farris-type tree. The visualization of tree topologies is shown in Figure 1.2.

My contributions in this research are:

- I generated training data with AliSim and proposed new data pre-processing method by e.g., extract alignment pattern frequencies with my own Python script.
- I improved F-zoneNN, including changing hyperparameters and improving the implementation of this neural network model. To this end, I developed my own

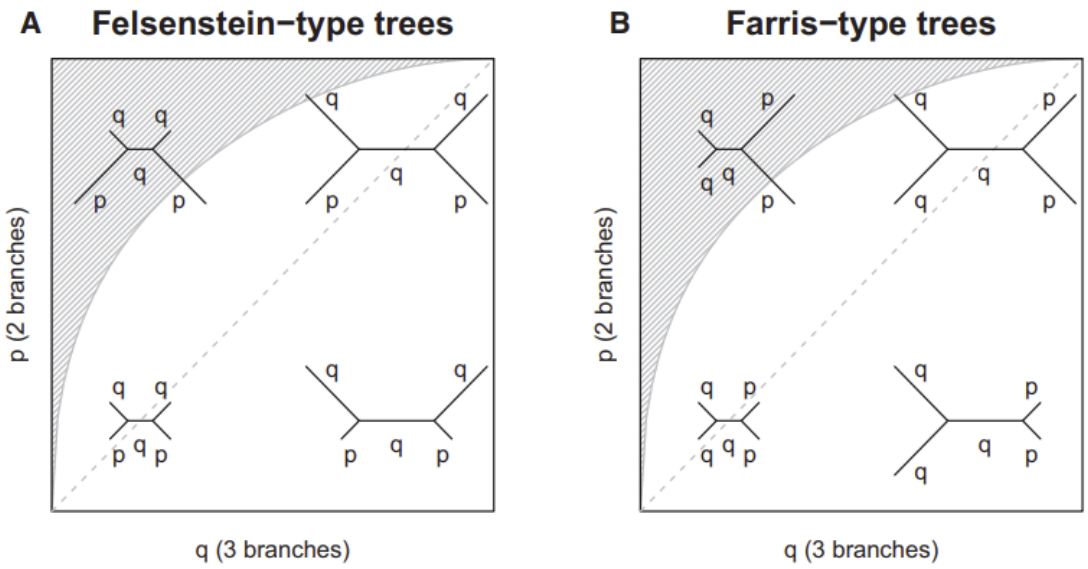

Figure 1.2: (Leuchtenberger et al. (2020))Visualization of Felsenstein-type (A) and Farris-type tree topologies (B).

Python script to train the NN model using PyTorch (compared with the original script using TensorFlow).

- I mastered high performance computing environments including Google Colab and the cluster system at the Center for Integrative Bioinformatics Vienna and utilised them to train and test the resulting NN model.

1.5 Organization of this Thesis

In chapter 1, I introduce basic concepts of phylogenetics, some typical phylogenetic methods and Machine Learning methods. Then I review some related works and summarize the contributions of this paper.

In Chapter 2, I show my overall workflow and details of my methods and contributions. In Chapter 3, I show my experiment results, including the computational time, results from my training process and results from my testing process. In Chapter 4, I conclude my experiment results.

Chapter 2

Methods

2.1 Overall Workflow

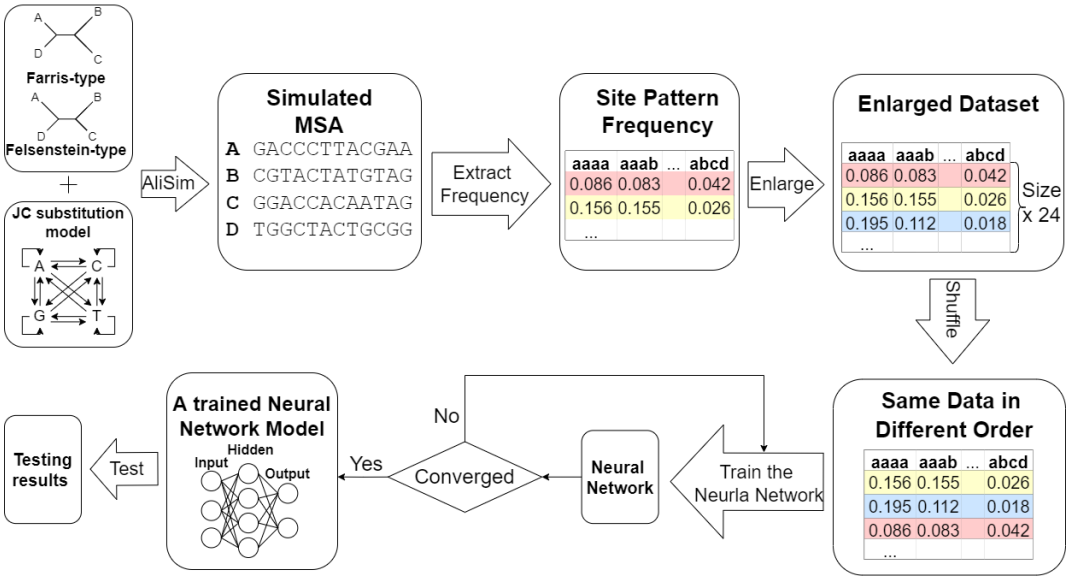

Figure 2.1: The overall workflow consisting of 3 main phases: (1) generating data; (2) training; and (3) testing the neural network.

My overall workflow is shown in Figure 2.1. This figure illustrates how I generate my training data, train and test the neural network model. I will discuss details about how I generate my testing data in section 2.4.

2.2 Generating Training Data

Training data generation process involves 4 steps: (1) Simulating MSAs from phylogenetic trees and a Markov substitution model with AliSim; (2) Extracting site pattern frequencies form sequence alignments; (3) Enlarging the data set; and (4) Shuffling the data.

2.2.1 Simulating Sequence

Phylogenetic data are relatively complex due to the wide range of distribution and randomness of the data. For a complex neural network model, if the size of training set is small, the model is likely to overfit the training data and lose generalization ability. To avoid overfitting, I need to generate a large training data set. Therefore, I chose AliSim as the sequence simulation method for this research due to its time and memory efficiency compared with existing simulation tools.

For the phylogenetic trees, I randomly generated them following Farris and Felsenstein types as shown in Figure 1.2. I independently varied branch length p (2 branches) and q (3 branches) from 0.005 to 0.745 at increments of 0.01. These combinations of parameters create $75 \times 75 \times 2$ different trees, here times 2 because I need to generate two types of trees.

For model of evolution, I applied the Jukes-Cantor (JC) model (Jukes and Cantor (1969)) for simplicity. For each tree, I generated 1,000 different alignments with 1,000 sites per sequence, which resulted in $11,250,000$ MSAs in total.

2.2.2 Extracting Site-pattern Frequencies

Some previous methods (Suvorov et al. (2019)) can only accept input alignments with fixed sequence length, which is impractical for real-world data since the sequence length is various among different alignments. I converted the input alignments into site pattern frequencies to make my neural network model adaptable to empirical data.

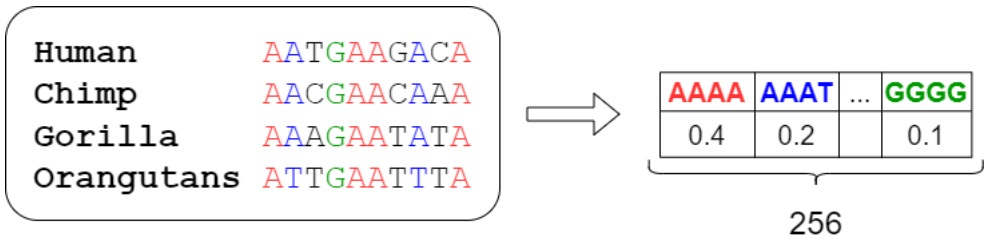

Figure 2.2: Extracting site-pattern frequencies from the sequence alignments.

For DNA data, every site has 4 different states, which are A, T, C, G, and for four taxa, there are $4^4 = 256$ unique site-patterns. To extract frequency form the site-patterns, I counted the number of each unique site-pattern and divided this number by the sequence length 2.2. However, the sequence length is not much larger than the number of site-patterns, for certain sequence alignments, the frequency of some site-patterns might be close to 0 or equal to 0. That problem may lead to some unused neurons, thus, reducing the efficiency of the neural network model.

Due to the symmetries in the JC model, I can reduce the 256 site-patterns to 15 distinct pattern categories, which are aaaa, aaab, aaba, aabb, aabc, abaa, abab, abac, abba, abbb, abbc, abca, abcb, abcc, abcd, where a, b, c, and d denote different nucleotides. The dimension reduction can improve the efficiency and compute speed of neural network model.

The data set is now represented in a table. Each row presents an alignment whereas columns record pattern-category frequencies, p, q, sequence length, and the label (Farris-type or Felsenstein-type).

2.2.3 Enlarging Data Set

The tree with four taxa has four tip nodes. In sequence simulation step, I just set taxa A, B, C, D on node 1, 2, 3, 4, respectively. When carrying out the simulation-based training of the neural network model, I am aware if a taxon belongs to the shorter branch or the longer branch. For example, for a Farris-type tree, taxon A and taxon D belong to the branches with length q, and taxon B and taxon C belong to the branches with length p, but the network cannot get this information during the testing process. The neural network should be able to accurately classify alignments independent of the order of the taxa.

To achieve this, I permuted the training data. For four taxa, there are $4! = 24$ different permutations, which are ABCD, ABDC, ACBD, ACDB, ADBC, ADCB, BACD, BADC, BCAD, BCDA, BDAC, BDCA, BCAD, BCDA, CABD, CADB, CBAD, CBDA, CDAB, CDBA, DABC, DACB, DBAC, DBCA, DCAB, DCBA. Instead of applying permutation directly on simulated sequences, I relabelled the extracted pattern-category frequencies to save the processing time. At the end of this step, the data set was enlarged by 24 times.

2.2.4 Shuffling Data

In each iteration, the neural network model is trained by a small batch of data, the gradient on this batch of data is assumed to be a good estimate of the total gradient.

But this assumption may not hold if the training data are not shuffled. Normally the data is sorted by the label. For my data set, the data is sorted by the branch length of the tree. If I did not shuffle the data, at the beginning of the training process, training data would always come from the tree with short branch length, while at the end of the training process, training data would come from the tree with long branch length. This problem could cause the model do not converge or overfit to the training data. Therefore, I need to shuffle the data to reduce variance and overfitting, and make sure that the neural network model has good generalization ability. After this step, the size of the data set did not change, but the order of data was changed randomly.

2.2.5 Parallelising the Computing Processes

To accelerate the data processing step, I divided the whole data set into 20 subsets so that I could process these subsets in parallel in the high-performance computing cluster at CIBIV. In the sequence simulation step, I divided the total tree files into 20 subsets, and simulated alignments for each subset in parallel. Similarly, the sequence simulation step, I extracted the frequencies and permuted each subset in parallel. The data set was divided into different subsets based on the label and branch length, the data should not just be shuffled within each subset. Therefore, I concatenated all permuted subsets before the shuffle step.

2.3 Training the Neural Network Model

The network architecture and setting of hyperparameters are shown in table 2.1. I generated a small validation set to optimize hyperparameters. The process of generating the validation data is totally the same as generating the training data, the only difference is for each combination of branch length p and q, only one sequence alignment was generated, therefore the size of the validation data set is 270,000.

During training, I computed the accuracy of the validation data set every 10,000 iterations. When the average accuracy does not change much, it means the model has converged.

Figure 2.3 shows the network architecture of F-zoneNN. This network takes a 15-D vector contains site-pattern frequencies as input, and outputs a number in range 0 to 1 corresponding to two tree types, Felsenstein-type and Farris-type, respectively.

Table 2.1: Architecture and Hyperparameters of F-zoneNN.

F-zoneNN	
Number layers	10 (8 hidden layers)
Number of nodes in layers	15, 64, 128, 256, 512, 1,024, 408, 208, 96, 1
Activation function(hidden layer)	ReLU
Activation function(output layer)	Sigmoid
Learning rate	0.0001
Batch size	64
Loss function	Binary Cross Entropy
Optimizer	Adam
Epochs trained	1

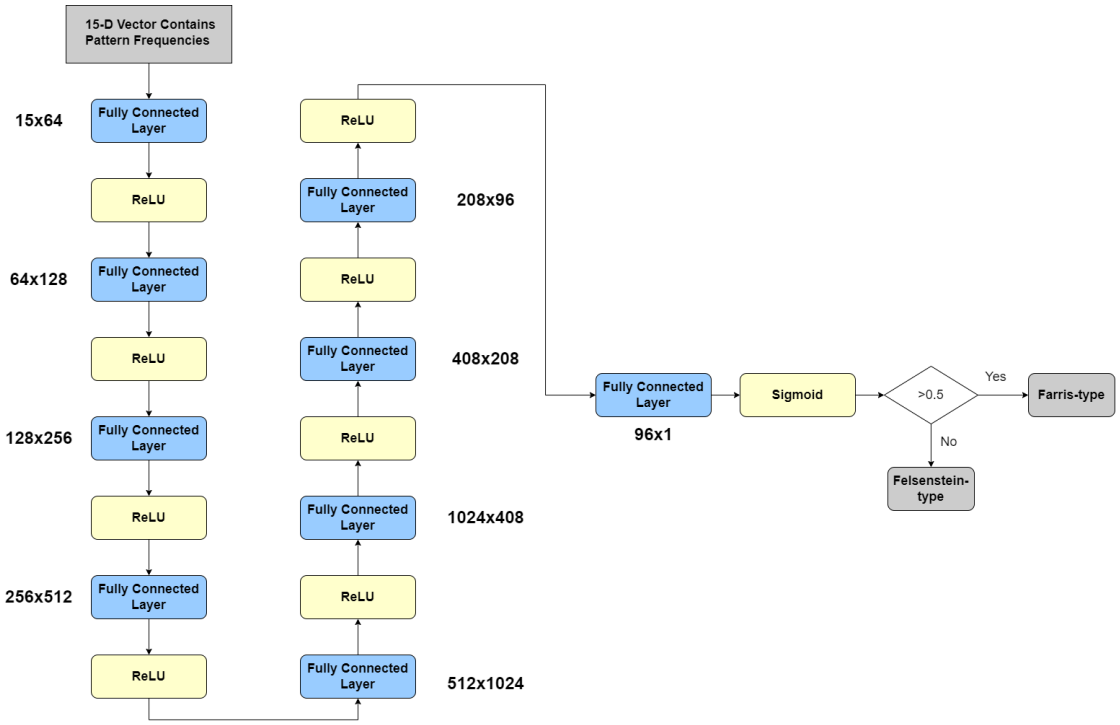

Figure 2.3: The main architecture of F-zoneNN.

2.4 Generating Testing Data

The process of generating testing data is quite similar to the process of generating training data. I also applied the JC substitution model, but the tree structures are different. I varied the branch length p and q between 0.025 and 0.725 at increments of 0.05, creating a total of $15 \times 15 \times 2 = 450$ Phylogenetic trees, and for each tree I used AliSim to simulate 200 different multiple sequence alignments of sequence length 1,000.

In addition, I did not shuffle the testing data. I want to test the accuracy of each combination of parameters (branch length p and q), so the data sorted by branch length is better for testing.

2.5 Testing the Model

I tested the neural network model on my testing data set. To test the performance of the model on different combinations of branch length, I divided the testing set into $12 \times 15 = 225$ subsets based on the combination of branch length p and q, then computed the accuracy of the model on each subset. When computing the accuracy, I computed the average accuracy, the accuracy of Farris-type and the accuracy of Felsenstein-type, respectively.

2.6 Hardware

I run the data generation process on the CIBIV cluster, then trained and tested my neural network model on Google Colab. The detailed information of these two computer clusters is shown in table 2.2 and table 2.3. Note that I only used GPU to accelerate the computation for training and testing process. Besides, Goolge Colab allows different configurations, here I only shows the environment I used for this experiment.

Table 2.2: Information of CIBIV cluster.

CIBIV cluster	
Number of servers	116
Total number of vCPUs	1850
CPU	AMD Opteron(tm) Processor 6380 AMD EPYC 7501 32-Core Processor AMD Opteron(tm) Processor 6348 AMD Ryzen 7 1700X Eight-Core Processor
Total RAM	11.89 TB

2.7 My Contributions

Compared with F-zoneNN (Leuchtenberger et al. (2020)), I used different methods to generate training data. In previous research, the alignments were created by sampling 1,000 pattern-categories from the multinomial distribution MD(p, q). In my experiment, I simulated the alignments with AliSim and extracted frequencies form the simulated

Table 2.3: Information of Google Colab.

Google Colab	
Number of vCPUs	4
Number of cores per CPU	2
CPU	Intel(R) Xeon(R) CPU @ 2.00GHz
CPU RAM	25.46 GB
GPU	NVIDIA Tesla T4
GPU RAM	16 GB

alignments. Therefore, my data process approach is more generalized and suitable for realistic data. Besides, my experiment proved that the AliSim has strong potential to generate large data sets for machine learning methods.

I also made some modifications to the implementation of the neural network model. First, the original model was implemented by TensorFlow, but the version of TensorFlow they used has not been supported by the up-to-date Python version, while I implement this neural network model based on another PyThon library, PyTorch. My implementation are supported by the latest Python version. Moreover, my implementation is more simple and efficient compared with their implementation.

In addition, I mastered high performance computing environments. I learned how to use computer clusters to process data, train and test the neural network model.

All codes developed in this project are available at GitLab https://gitlab.cecs.anu.edu.au/u7201888/fzone_net

Chapter 3

Results

3.1 Computational Time

Table 3.1 shows the running time of each step. The hardware I used is discussed in section 2.6. The results show that the data generation and training phases required long processing time. But for testing process, the neural network took only 30 seconds to test 2,160,000 data, which was only 14 microsecond per testing data.

Table 3.1: Running time.

Process	Running Time (Per job)	Number of Jobs
Simulate alignments	60 hours	20 jobs
Extract frequencies	26 hours	20 jobs
Permute data	2 minutes	20 jobs
Concatenate permuted data	2 hours	1 job
Shuffle data	18 minutes	1 job
Train the NN	6.7 hours	1 job
Test the NN	30 seconds	1 job

3.2 Results from the Training

Figure 3.1 illustrates the accuracy during training. This accuracy was computed on my validation data set as mentioned in section 2.3.

The average accuracy shows that the model converged after tens of thousands of iterations. But the accuracy of Farris-type and Felsenstein-type varied a lot during training,

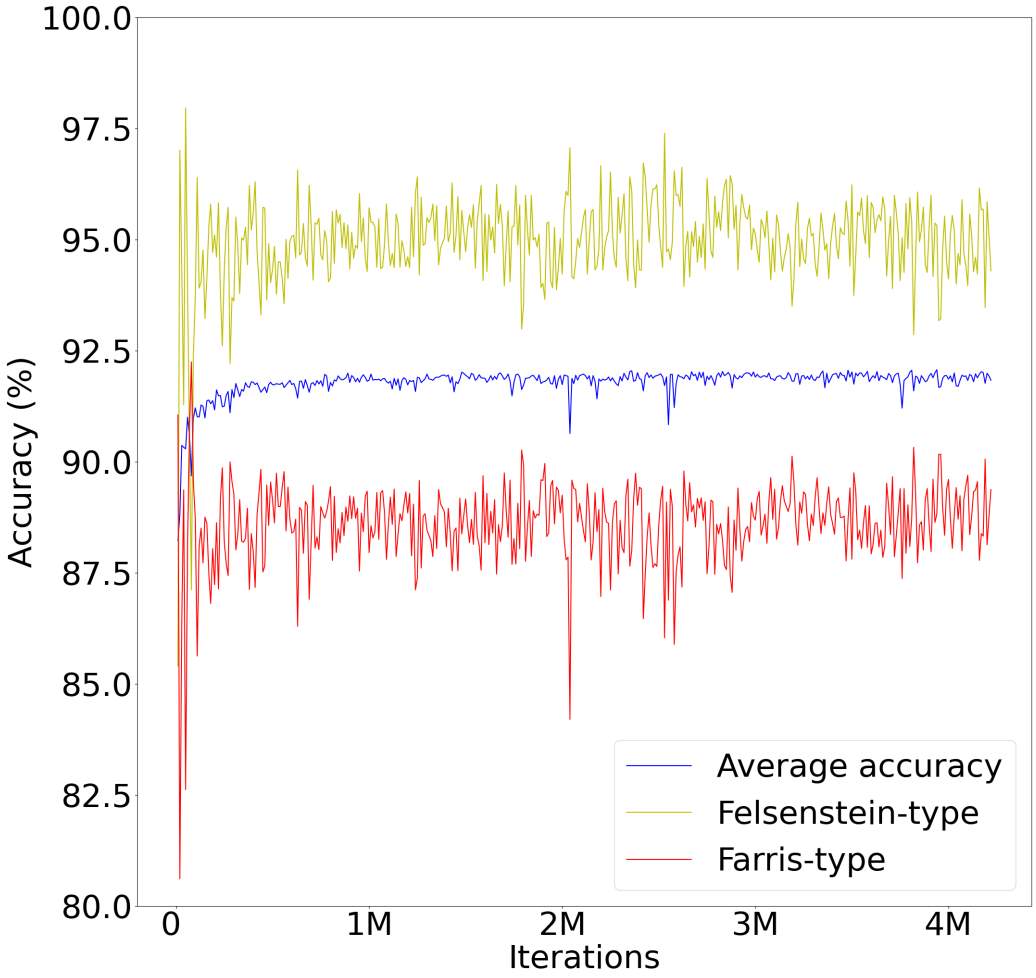

Figure 3.1: Accuracy during training. This model was trained on the data set generated by AliSim 2.2.

and the model always got higher accuracy on Felsenstein-type alignments.

Note that when the branch length p and q are equal, the structure of Farris-type tree and Felsenstein-type tree are identical. That means for this data the model predicts that to Farris-type or Felsenstein-type are both correct. However, I only allow two different labels in my data set. Therefore, in this case, I expected the model just randomly predict the alignments to be Farris-type or Felsenstein-type with equal probability. This

randomness is the main reason caused by the variation of accuracy of Farris-type and Felsenstein-type, and the average accuracy is sufficient to illustrate the good convergence of the model.

3.3 Results from the Testing

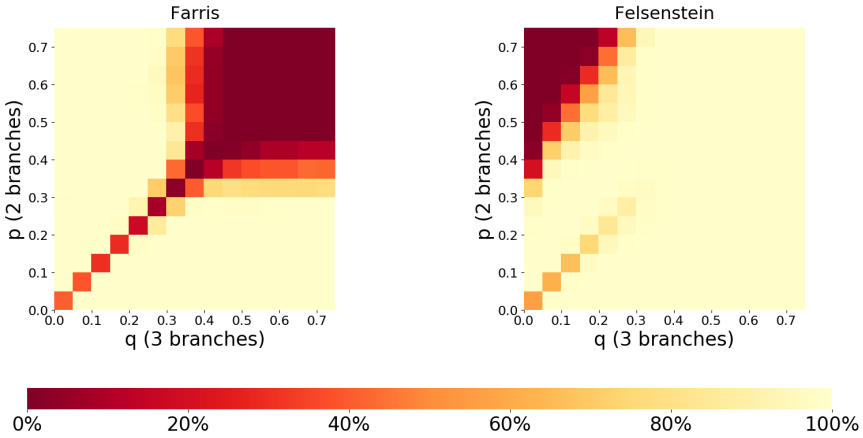

Figure 3.2: The results from the script provided by the paper (Leuchtenberger et al. (2020)), which was retrained on the subset of the author's training data and tested on the author's testing data. Accuracy of the neural network model to infer the correct tree type under a Farris-type tree (left) and Felsenstein-type tree (right). The average accuracy of Farris-type is 69.08%, the average accuracy of Felsenstein-type is 87.92% and the total average accuracy is 78.50%.

To verify my neural network model, I trained my model and the original model on a subset (about 25% of the total data set) of the training data provided by the paper Leuchtenberger et al. (2020) (the author's training data), and then tested the two models on the testing set provided by the paper (the author's testing data). Figure 3.2 and 3.3 show the testing results.

For verification purpose, I only employed a subset rather than the whole data set provided with the paper to save the time. Specifically, I trained my model and the original F-zoneNN model on a same sub-data set, to test if the behaviors of the two models are similar. Figure 3.2 and figure 3.3 illustrate that the result of my model matched well with the result of the F-zoneNN model. The slight difference in accuracy is mainly caused by the randomness in initializing weights.

Then I trained and tested the model on the training set and testing set generated by

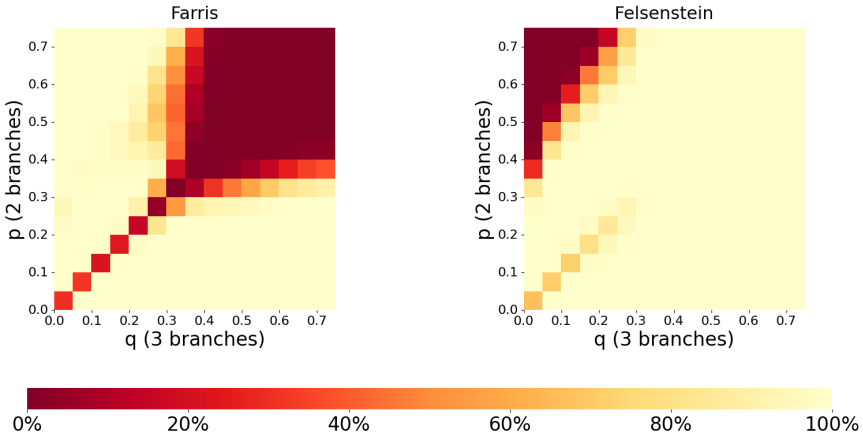

Figure 3.3: The results from my neural network model, which was also trained on the subset of the author's training data and tested on the author's testing data. Accuracy of the neural network model to infer the correct tree type under a Farris-type tree (left) and Felsenstein-type tree (right). The average accuracy of Farris-type is 65.23%, the average accuracy of Felsenstein-type is 89.23% and the total average accuracy is 77.24%.

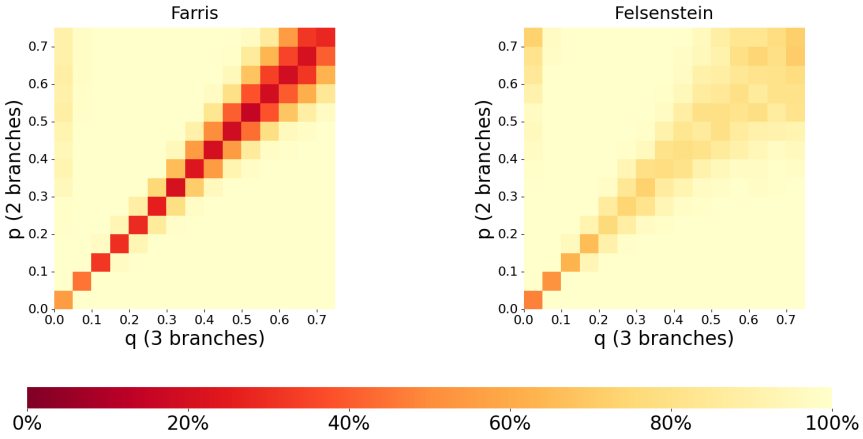

Figure 3.4: The results form my neural network model, which was trained and tested on the data set generated by AliSim. Accuracy of the neural network model to infer the correct tree type under a Farris-type tree (left) and Felsenstein-type tree (right). The average accuracy of Farris-type is 88.63%, the average accuracy of Felsenstein-type is 94.22% and the total average accuracy is 91.42%.

AliSim, which was discussed in section 2.2 and 2.4. My neural network can distinguish alignments originated from Felsenstein-type or Farris-type trees. Felsenstein-type alignments and Farris-type alignments were successfully identified to a high degree of

accuracy (91.42%). A few misclassifications occurred primarily on the diagonal, where the two tree types are not distinguishable because the branch length $p = q$ and the tree structures of Farris-type tree and Felsenstein-type trees are identical. Moreover, when the branch length p and q are close but not equal, it is difficult for the neural network to make correct predictions and we noticed that in this region the model tends to label these trees as Felsenstein-type, that is the reason why this model get higher accuracy on Felsenstein-type than Farris-type. That means the classification of this neural network model was imbalanced. Therefore, when applying this model on realistic data set, the accuracy of this model might be inconsistent. If the number of Felsenstein-type trees are much smaller than the number of Farris-type trees, the accuracy may be lower than the accuracy on a balanced data set.

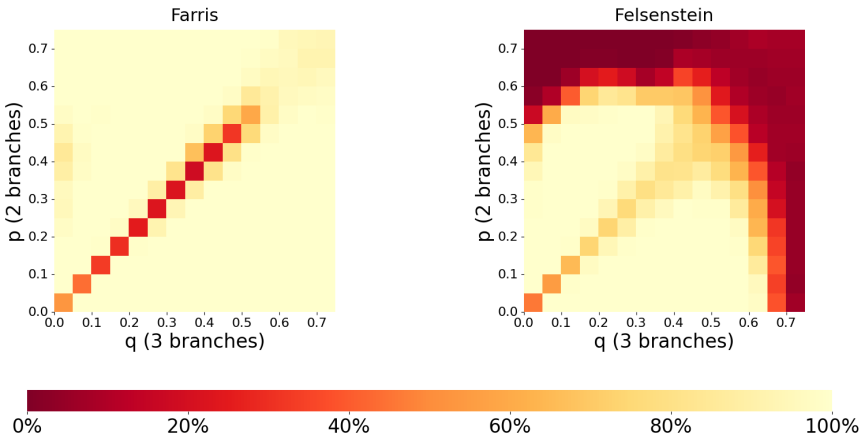

Figure 3.5: The results form my neural network model, which was trained on the data set generated by AliSim and tested on the author's testing data set. Accuracy of the neural network model to infer the correct tree type under a Farris-type tree (left) and Felsenstein-type tree (right). The average accuracy of Farris-type is 94.99%, the average accuracy of Felsenstein-type is 61.43% and the total average accuracy is 78.21%.

Then I tested my model on the author's testing data. The result is shown in Figure 3.5. I observed accuracy dropped sharply when p and q were greater than 0.5. This problem was caused by the different understanding of parameters p and q (Alina Leuchtenberger, personal communication). In the original paper (Leuchtenberger et al. (2020)), p and q values represent the observed distance, while in my experiment I understood p and q as the expected distances. The observed distance is expressed as the number of nucleotide differences per site. The observed distance is a very intuitive measure, but if the degree of divergence is high, observed distances are generally not very informative with regard to the number of substitutions that actually occurred, it may underestimate the actual number of substitutions. This problem is caused by the following effect. Assume that two or more mutations take place consecutively at the same site in the sequence, for example, an A is replaced by a C, and then by a G. As a result, although two substitutions have

occurred, only one difference is observed (A to G). Moreover, in case of a back-mutation (A to C to A), no difference is observed. The expected distance represents the actual number of substitutions per site that occurred (Lemey et al. (2009)). For JC model, the relationship between observed distance and expected distance is:

$$D_s = \frac{3}{4}(1 - e^{-\frac{4}{3}l})$$

where the D_s represents the observed distance, and the l represents the expected distance.

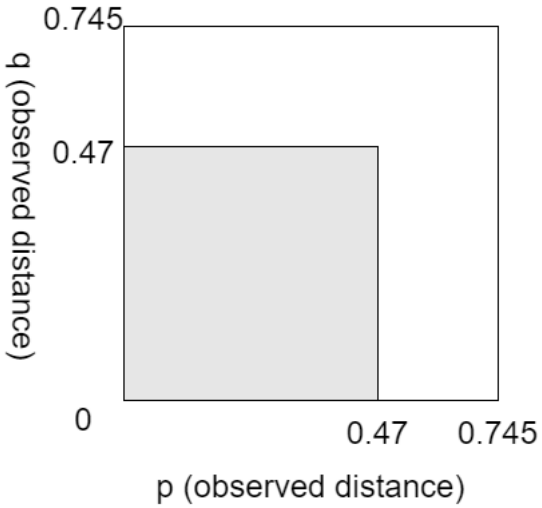

Figure 3.6: The parameter space. The larger square is the parameter space of the author's data set, and the shadowed square is the parameter space of my data set.

Due to this reason, in my experiment the parameters were actually distributed in a narrow region in parameter space. Using the equation above, I set the maximum expected distance to be 0.745, which equals 0.47 in observed distance (shown in figure 3.6). Therefore, in Figure 3.5, this model did not perform well when p and q (observed distance) were greater than 0.5.

Chapter 4

Conclusion

I have implemented a neural network model to classify the Farris-type tree and Felsenstein-type tree, which is a well-known problem in phylogenetic inference. The testing result on the testing set generated by AliSim illustrates that my neural network can achieve high accuracy and be applied to infer the tree type for phylogenetic inference methods. The neural network has strong potential to be applied to other phylogenetic methods to increase the accuracy and efficiency.

However, I also found some possible future works that are extended from this research. First, I need to deal with the problem I overlooked with the branch length. I can enlarge the branch length to make the data distributed in a wider range in parameter space.

Another potential future work is about the network structure. I extracted the frequencies form the alignments that allows the input alignments to have arbitrary sequence length. But the type of site-patterns increase exponentially with increment of the number of taxa. When the number of taxa is relatively large, the number of parameters of the neural network model could be extremely large and hard to be trained. Besides, this neural network does not allow the input alignments to have gaps (one nucleotide is deleted instead of replaced by another one). To make the network accept input alignments with gaps, the number of site-pattern should be increased from 256 to 625. To solve this problem, I need to test some other network architectures.

Regarding evolutionary model, in this research, I assumed JC substitution model, because the JC model is simple and easy to compute, but this model may not reflect all realistic evolutionary processes. I can try to use more complex evolutionary models including insertions and deletions.

Bibliography

Abadi, S.; Avram, O.; Rosset, S.; Pupko, T.; and Mayrose, I., 2020. Modelteller: Model selection for optimal phylogenetic reconstruction using machine learning. *Molecular Biology and Evolution*, 37, 11 (2020), 3338–3352. doi:10.1093/molbev/msaa154. [Cited on page 6.]

Biau, G. and Scornet, E., 2016. A random forest guided tour. *TEST*, 25, 2 (2016), 197–227. doi:10.1007/s11749-016-0481-7. [Cited on page 3.]

BISHOP, C. M., 2016. SPRINGER-VERLAG NEW YORK. [Cited on page 3.]

Burgstaller-Muehlbacher, S.; Crotty, S. M.; Schmidt, H. A.; Drucks, T.; and von Haeseler, A., 2021. Modelrevelator: Fast phylogenetic model estimation via deep learning. (2021). doi:10.1101/2021.12.22.473813. [Cited on page 5.]

Chor, B. and Tuller, T., 2005. Maximum likelihood of evolutionary trees: hardness and approximation. *Bioinformatics*, 21, Suppl 1 (2005), i97–i106. doi:10.1093/bioinformatics/bti1027. [Cited on page 2.]

Felsenstein, J., 1978. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology*, 27, 4 (1978), 401. doi:10.2307/2412923. [Cited on page 5.]

Felsenstein, J., 2004. In *Inferring phylogenies.*, 664. Sinauer Associates, Inc. [Cited on pages 1 and 2.]

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016). doi:10.1109/cvpr.2016.90. [Cited on page 5.]

Jukes, T. H. and Cantor, C. R., 1969. *Evolution of protein molecules.* Academic Press. [Cited on page 10.]

Lemey, P.; Salemi, M.; and Vandamme, A.-M., 2009. In *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing.*, 723. United States of America by Cambridge University Press, New York. [Cited on pages 1, 2, and 22.]

Leuchtenberger, A. F.; Crotty, S. M.; Drucks, T.; Schmidt, H. A.; Burgstaller-Muehlbacher, S.; and von Haeseler, A., 2020. Distinguishing felsenstein zone from farris zone using neural networks. *Molecular Biology and Evolution*, 37, 12 (2020), 3632–3641. doi:10.1093/molbev/msaa164. [Cited on pages 5, 6, 7, 14, 19, and 21.]

Ly-Trong, N.; Naser-Khdour, S.; Lanfear, R.; and Minh, B. Q., 2022. Alisim: A fast and versatile phylogenetic sequence simulator for the genomic era. *Molecular Biology and Evolution*, 39, 5 (2022), in press. doi:10.1093/molbev/msac092. [Cited on page 4.]

Rambaut, A. and Grass, N. C., 1997. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Bioinformatics*, 13, 3 (1997), 235–238. doi:10.1093/bioinformatics/13.3.235. [Cited on page 4.]

Schoeniger, M. and von Haeseler, A., 1995. Simulating efficiently the evolution of dna sequences. *Bioinformatics*, 11, 1 (1995), 111–115. doi:10.1093/bioinformatics/11.1.111. [Cited on page 4.]

Sullivan, J.; Abdo, Z.; Joyce, P.; and Swofford, D. L., 2005. Evaluating the performance of a successive-approximations approach to parameter optimization in maximum-likelihood phylogeny estimation. *Molecular Biology and Evolution*, 22, 6 (2005), 1386–1392. doi:10.1093/molbev/msi129. [Cited on page 2.]

Suvorov, A.; Hochuli, J.; and Schrider, D. R., 2019. Accurate inference of tree topologies from multiple sequence alignments using deep learning. *Systematic Biology*, 69, 2 (2019), 221–233. doi:10.1093/sysbio/syz060. [Cited on pages 5 and 10.]

Zou, Z.; Zhang, H.; Guan, Y.; and Zhang, J., 2019. Deep residual neural networks resolve quartet molecular phylogenies. *Molecular Biology and Evolution*, 37, 5 (2019), 1495–1507. doi:10.1093/molbev/msz307. [Cited on page 4.]