The Australian National University
2600 ACT | Canberra | Australia

Australian National University

School of Computing

College of Engineering, Computing and Cybernetics (CECC)

Phylogenetic Model Selection via Machine Learning

— 24 pt research project (S1/S2 2024)

A report submitted for the course
COMP8800, Computing Research Project

By:
Yanghe Dong

Supervisor:
Assoc. Prof. Minh Bui
Associate Supervisor:
Nhan Ly-Trong

October 2024

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the Academic Integrity Rule;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or LMS course 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.

I acknowledge that I am expected to have undertaken Academic Integrity training through the Epigeum Academic Integrity modules prior to submitting an assessment, and so acknowledge that ignorance of the rules around academic integrity cannot be an excuse for any breach.

October (2024), Yanghe Dong

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Associate Professor Minh Bui, for welcoming me into such an accomplished lab. His vast knowledge, creative insights and supportive attitude have been crucial in helping me complete this thesis.

I also want to extend my thanks to my labmates Thomas Wong and Huaiyan Ren. Their help and kindness have been a constant source of encouragement.

I owe a special debt of gratitude to Nhan Ly-Trong, the senior member of our group. Over the past year, he has made the successful transition from PhD student to postdoctoral researcher, and at the same time, he has helped me evolve from someone with little research experience to someone with a firmer grasp of the field. He has been incredibly patient, answering my many, sometimes naive, questions and offering invaluable advice. I also greatly appreciate his careful review of my drafts, which made me realise just how much I still have to improve in my writing.

Lastly, I would like to thank my friends Haiqing Zhu and Xinyu Dong for their companionship during this period.

This research was undertaken with the use of the National Computational Infrastructure (NCI Australia), which is supported by the National Collaborative Research Infrastructure Strategy (NCRIS).

Abstract

Phylogenetic inference, which reconstructs evolutionary trees from DNA or amino acid sequences, is crucial for understanding the evolutionary histories of species on Earth. Model selection is a fundamental step in this process, determining the best-fit model for the data. However, classic maximum likelihood-based methods for model selection are computationally intensive. This study introduces a machine learning-based framework for amino acid model selection, consisting of three components: (1) *protFinder* for selecting the best-fit substitution model, (2) *RHASFinder* for identifying the appropriate rate heterogeneity model, and (3) *protFFinder* for determining the use of empirical pre-estimated frequencies. Our framework is an order of magnitude faster than the widely used ModelFinder, while maintaining comparable accuracy.

Table of Contents

Chapter 1

Introduction

Phylogenetics studies the evolutionary relationships among taxa (groups of organisms) by reconstructing phylogenetic trees, typically from molecular data such as multiple sequence alignments (MSAs) of nucleotides or amino acids. This type of analysis not only allows researchers to explore the evolutionary history of various species but also aids in predicting how organisms may evolve in the future. For example, during the COVID-19 pandemic, phylogenetic analysis was instrumental in tracing the origins of the virus and informing efforts to prevent and control its spread (Attwood et al., 2022; Zhao et al., 2022).

The stochastic process of evolution, reflected in the changes between nucleotides or amino acids over time, is commonly modeled by a probabilistic model—the continuous-time Markov chain. Different constraints on the rate matrix of the Markov chain give rise to various models of evolution.

Model selection is a preparatory step in phylogenetic tree reconstruction. It involves selecting the model that best explains the evolutionary history of an MSA. Maximum likelihood is a widely used method for this task, as it searches the model space to identify parameters that maximise the likelihood of observing the given MSA. However, the main drawback of maximum likelihood-based methods is the computational inefficiency. Calculating likelihoods for large genomic datasets can take several days. Additionally, the suitability of common model selection criteria, such as the Akaike Information Criterion (AIC) (Akaike, 1974) and Bayesian Information Criterion (BIC) (Schwarz, 1978), remains debated in phylogenetics. Susko and Roger (2020) suggest that AIC may be inaccurate for closely related sequences or when model parameters are near a boundary, and that the formula of BIC may be problematic. As a result, there is a growing demand for innovative approaches that can bypass the computational burden of likelihood calculations and the limitations of information criteria.

Machine learning, particularly deep learning, has been transforming various fields in recent years. Phylogenetics is a relatively new domain for machine learning to make a significant impact, with most relevant research emerging since 2020. Among these efforts, only four existing studies have applied machine learning to model selection. Abadi et al. (2020) and Burgstaller-Muehlbacher et al. (2023) focus on nucleotide model selection, while the work of Kulikov et al. (2024) explores both nucleotide and amino acid models, although it is limited to MSAs with only four taxa. When the topic of this thesis was chosen, there was no existing work focusing on amino acid model selection. Although Nguyen Huy and Vinh (2024) proposed ModelDetector several months ago, their work involved substitution model selection in a relatively simple data setting, and the absence of a test set raises concerns about overfitting. Therefore, this thesis aims to address the following research question:

How can machine learning methods be applied to amino acid model selection to improve efficiency?

In this thesis, I proposed a machine learning-based framework for amino acid model selection, consisting of three components: (1) protFinder for selecting the best-fit substitution model, (2) RHASFinder for identifying the appropriate rate heterogeneity model, and (3) protFFinder for determining the use of empirical preestimated frequencies. Compared to the state-of-the-art maximum likelihood-based ModelFinder, the machine learning framework demonstrates competitive accuracy on simulated data while offering significantly greater efficiency, except in the case of very short alignments.

All the source code and scripts used in this research are publicly available here.

1.1 Contribution

The main contributions of this thesis are summarised below.

- Estimated empirical distributions of evolution model parameters from real amino acid alignments.
- Designed and trained protFinder, a neural network for selecting the best amino acid substitution model.
- Designed and trained RHASFinder, the first machine learning model for classifying rate heterogeneity models.
- Designed and trained protFFinder, the first machine learning model to determine the use of empirical amino acid frequencies.
- Conducted a comprehensive benchmarking of these machine learning models against the widely used maximum likelihood-based ModelFinder algorithm (Kalyaanamoorthy et al., 2017) on simulated data.
- Evaluated and discussed the performance of protFinder on real data.

Chapter 2

Background

This chapter provides the essential background in phylogenetics and machine learning to help readers better understand the context of this study.

2.1 Phylogenetics

2.1.1 Multiple Sequence Alignment

A multiple sequence alignment (MSA) is the result of aligning nucleotide or amino acid sequences (Figure 2.1). Each row represents a taxon (typically a species or gene), and each column is referred to as a site. MSAs are based on the concept of homology, which means that nucleotides or amino acids at the same site are assumed to have evolved from the same ancestral state. Gaps ('–') that represent deletions or insertions are commonly used to achieve the alignment as the original sequences are of different lengths. Figure 2.1 shows an example of an amino acid sequence alignment. There are 20 types of amino acids, each represented by a capital letter abbreviation. The generation of alignments is now typically performed using computer programs such as MAFFT (Katoh and Standley, 2013) and Muscle (Edgar, 2022), and these alignments are essential data for most phylogenetic analyses. In this thesis, the terms 'MSA' and 'alignment' are used interchangeably to refer to the same concept.

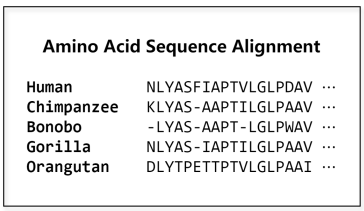


Figure 2.1: Example of an amino acid MSA of great ape species.

2.1.2 Phylogenetic Tree

A phylogenetic tree represents the evolutionary relationships among species or taxa (Figure 2.2). It is typically bifurcating, meaning that each internal node splits into two descendant branches. External nodes (leaves) represent present-day taxa while internal nodes represent extinct ancestors. A phylogenetic tree has two main components: topology and branch lengths. Topology refers to the overall structure of the tree, showing the order of branching and how taxa are related. On the other hand, branch lengths often represent evolutionary distances measured in terms of substitutions per site. Figure 2.2 shows an example of phylogenetic tree with species in Figure 2.1.

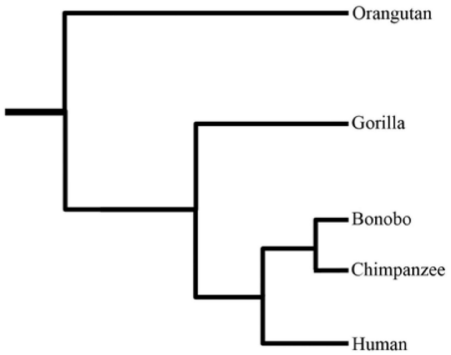


Figure 2.2: The phylogenetic tree of great ape species (Yousaf et al., 2021).

2.1.3 Model of Sequence Evolution

During the process of evolution, mutations can occur, reflected in substitutions of nucleotides or amino acids in sequences. While it may seem intuitive that more similar sequences are more closely related, the evolutionary distance—defined as the expected number of substitutions per site—cannot be directly inferred by simply counting the differences between sequences. This is due to the phenomenon of multiple substitutions at the same site (known as multiple hits), which can obscure changes and lead to an underestimation of the actual number of substitutions. To account for this, the substitution process is modelled using a continuous-time stationary Markov chain.

The Markov chain is characterised by its rate matrix $Q = \{q_{ij}\}$ where q_{ij} is the instantaneous rate of change from state i to j. In this context, a state refers to one of the four nucleotides or one of the 20 amino acids. The transition probability matrix $P(t) = \{p_{ij}(t)\} = e^{Qt}$ describes the probability that state i changes to j after a certain time t. The assumption of stationarity implies that the frequencies of nucleotides or amino acids remain constant over time.

In most cases, and as in this thesis, the substitution models under consideration are time-reversible. A Markov chain is time-reversible if and only if $\pi_i q_{ij} = \pi_j q_{ji}$ for all $i \neq j$ where π_i is the frequency of nucleotide or amino acid i. This property is equivalent to the

fact that Q can be written as the product of a symmetric matrix $S = \{s_{ij}\}$ (referred to as amino acid exchangeabilities in amino acid substitution models) and a diagonal matrix $\Pi = \{\pi_i\}$ which contains the equilibrium frequencies. Different substitution models arise from different constraints on S and Π. For example, the simplest nucleotide substitution model, the JC model (Jukes et al., 1969), assumes equal s_{ij} and equal π_i.

In the case of amino acids, there are 20 possible states, making rate matrix Q a 20×20 matrix. As a result, there are far more free parameters in an amino acid substitution model than in a nucleotide model. Because of this complexity, empirical models, which are pre-estimated from large datasets of protein sequences, are typically used for amino acid alignments (e.g., LG (Le and Gascuel, 2008) and WAG (Whelan and Goldman, 2001)). These empirical models come with predefined matrices S and Π, thereby avoiding the need for parameter estimation during the phylogenetic analysis.

It is biologically unrealistic to assume that all sites in a sequence evolve at the same rate. To account for rate heterogeneity across sites (RHAS), a common approach is to model the rates using a gamma distribution, where the rate parameter β is set equal to the shape parameter α (Figure 2.3). This model is denoted by the suffix $+\Gamma$, e.g., LG$+\Gamma$. The parameter α is inversely related to the degree of rate variation, i.e., low α denotes high rate variation whereas large α means more homogeneous rates. In practice, a discrete gamma model with four rate categories is typically used as an approximation, as it is computationally efficient and performs equally well (Yang, 2014). In the discrete model, the continuous gamma distribution is divided into k equal-probability bins, and the rate assigned to each site in a bin is the mean rate of that bin. For clarity, in this thesis, $+\Gamma$ refers to the discrete gamma model with four ($k = 4$) categories.

Another type of RHAS model is the FreeRate model (Soubrier et al., 2012). In contrast to the Γ model, FreeRate models not only allow for unequal proportions of sites in each category but also estimate the rates directly from the MSA. A FreeRate model with n categories is denoted as $+R_n$. While FreeRate models often provide a better fit than Γ, the parameter estimation process is more computationally intensive.

If some sites in the alignment are considered invariant, the model is extended with +I. When the predefined amino acid frequencies Π do not fit a given MSA, the frequencies observed in the alignment should be used instead, represented by +F.

Those additional parameters increase the complexity of models of evolution. For example, starting with the original LG model, it can be extended to various versions including LG+F, LG+Γ, LG+R_n, LG+I, LG+F+Γ, LG+F+R_n, LG+F+I, LG+F+I+Γ, LG+F+I+R_n, LG+I+Γ, LG+I+R_n, to better account for the complex nature of evolutionary processes.

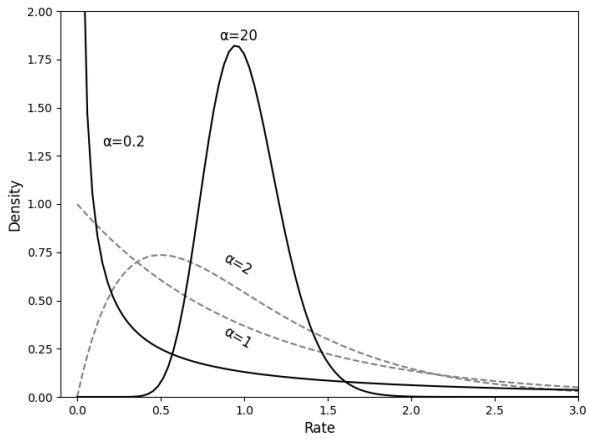


Figure 2.3: Probability density function of the gamma distribution with α of 0.2, 1, 2 and 20.

2.1.4 Model Selection

The fit of a model to an MSA can be measured by the likelihood function, $L = P(D|\theta, t)$, which is the probability of observing the data D (i.e., the MSA) given a hypothesis defined by the model parameters θ and the phylogenetic tree t. The tree t does not necessarily represent the final estimate of relationships among taxa but is only used for model parameter estimation and likelihood calculation. The tree is typically reconstructed using a likelihood-free method such as maximum parsimony (Farris, 1970; Fitch, 1971) or neighbour-joining (Saitou and Nei, 1987), to reduce runtime. Model selection remains robust as long as the starting tree is reasonable (Posada and Crandall, 2001). Since sites in the alignment are typically assumed to evolve independently, the likelihood of the hypothesis is calculated as the product of the likelihoods for each individual site, which involves summing over all possible ancestral states at each internal node. Maximum likelihood estimation seeks to find the model parameters θ and t that maximise the likelihood function L, and the corresponding log maximum likelihood $l = \ln(L)$ is often used to avoid numerical underflow.

The likelihood ratio test (LRT) statistic (Vuong, 1989) is used to compare nested models, with one model being the special case of the other. LRT is defined as $LRT = 2(l_1 - l_0)$, where l_1 is the log maximum likelihood of the more complex model and l_0 is the log maximum likelihood of the simpler model. The LRT statistic is asymptotically distributed as χ^2_{p-q}, where p and q are the number of parameters in two models.

For comparing non-nested models, various information criteria are used in phylogenetics, which combine the likelihood with a penalty based on the number of free parameters p in the model. The Akaike information criterion (Akaike, 1974) is defined as $AIC = -2l+2p$,

where a smaller AIC score indicates a better-fitting model. To account for small sample sizes, the corrected AIC (AICc) (Sugiura, 1978; Hurvich and Tsai, 1989) is defined as $AIC_c = AIC + \frac{2p(p+1)}{n-p-1}$, where where n is the sequence length. Another commonly used criterion is the Bayesian Information Criterion (BIC) (Schwarz, 1978), which is defined as $BIC = -2l + p \log n$. BIC imposes a higher penalty on models with more parameters.

These selection criteria require computationally expensive likelihood calculations, making classic model selection methods time-consuming.

Once the best-fit model is selected, it is used to reconstruct the phylogenetic tree, typically via maximum likelihood-based methods such as IQ-TREE (Minh et al., 2020) or RAxML-NG (Kozlov et al., 2019).

2.2 Machine learning

2.2.1 Basic Concepts

A machine learning algorithm is an algorithm that is able to learn from data (Goodfellow et al., 2016a). In a supervised classification task, the algorithm produces a function (i.e., machine learning model) that maps input data to categorical labels. The term 'supervised' indicates that each input sample is paired with its correct label. The dataset is typically split into training, validation, and test sets.

The model is trained on the training set, learning patterns from the data. The validation set is used to evaluate the model's performance during training and to tune hyperparameters—the parameters that cannot be learned from the data but are set before training. The validation set helps ensure that the model does not become too complex and overfit the training data. Overfitting occurs when a model performs well on the training data but fails to generalise to new data. Finally, once the best model is selected, its performance is evaluated on a test set, which consists of data that the model has not seen before.

2.2.2 Random Forest Classifier

A Random Forest classifier is a supervised classification model which combines multiple decision trees (Quinlan, 1986). Each decision tree is trained on a random subset of the training set and it splits the data into branches based on their values. The terminal nodes (leaves) of the tree represent class labels. The final prediction of the Random Forest classifier is the majority vote of all decision trees. This approach effectively mitigates the overfitting problem that may occur with a single decision tree.

2.2.3 Deep Learning

Deep learning is a kind of machine learning (Goodfellow et al., 2016b). While traditional machine learning models rely on algorithms explicitly designed by humans, deep learning uses neural networks to automatically learn and solve problems by discovering patterns in the data. As the number of layers and the number of units within each layer increase, neural networks can model functions of very high complexity. Although training a neural network typically requires more sophisticated hyperparameter tuning and greater computational resources compared to traditional models like the Random Forest classifier, it often achieves significantly better performance on a wide range of tasks.

Feedforward Neural Network

The feedforward neural network, also called multilayer perceptron (MLP), is one of the most basic types of network. It consists of a series of fully connected layers, where each neuron (unit) in one layer is connected to every neuron in the next layer (Figure 2.4). In a feedforward network, the information flows forward through multiple layers to the output layer without any feedback connections.

Let the input vector $\mathbf{x}$ have n dimensions $x_1 \dots x_n$. In each intermediate layer (called hidden layer), the network applies a weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$ to the output of the previous layer. Then an activation function Φ is used to introduce non-linearity, which is necessary for networks to approximate complex functions. Therefore, the output of the $(p+1)$-th hidden layer is $h_{p+1} = \Phi(\mathbf{W} \cdot h_p + \mathbf{b})$, where h_p is the output of the p-th hidden layer.

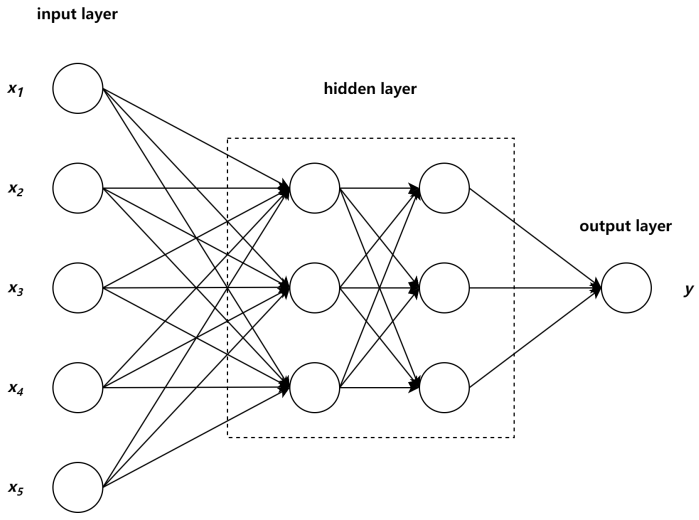


Figure 2.4: A multilayer perceptron. Each circle represents a neuron (unit).

The network learns its weights and biases through the backpropagation algorithm (Kelley, 1960). The loss function L measures the difference between the network's predicted output and the true label. Then gradients of L with respect to weights and biases are calculated, starting from the output layer and propagating backward through the network. These gradients are used to update the weights and biases using an optimisation algorithm. The learning rate determines the size of each update step. The whole process is repeated many times (epochs) to progressively minimise the loss.

Convolutional Neural Network

Convolutional neural networks (CNNs) (LeCun et al., 2015) are a kind of network designed for processing grid-like data such as images. CNNs are widely used in many deep learning architectures and have achieved significant success in various real-world applications.

CNNs involve the mathematical operation called convolution. In this operation, a kernel of a specified size slides over the 2D input in a manner similar to a sliding window. At each position, the kernel performs element-wise multiplication with the input it covers, and the results are summed to produce the output at the corresponding location. This is illustrated in Figure 2.5.

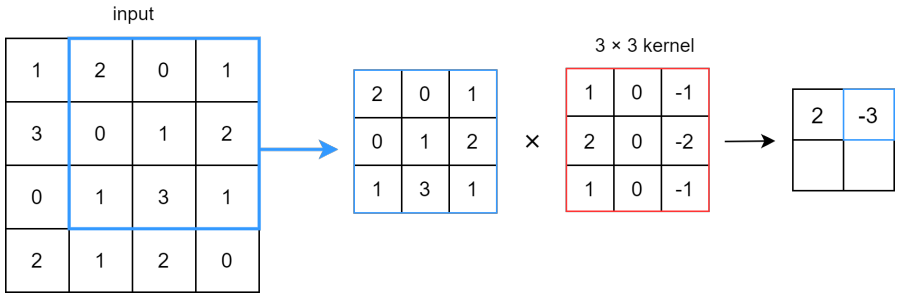


Figure 2.5: Convolution of a single input channel with a 3×3 kernel.

The input to a convolutional layer can have multiple channels, and convolution is performed on each channel with a separate kernel. The output of each kernel is summed across all channels to produce a single output called feature map. If n output channels are required, this process is repeated with n different sets of kernels. Each set of kernels produces a corresponding feature map.

A convolutional layer is typically followed by an activation function, and after several such convolutional blocks, a pooling layer is applied. The pooling layer uses a pooling function to summarising nearby values. For example, max pooling produces the maximum value within a neighborhood while average pooling computes the average. Pooling helps reduce the size of the feature map, and make the representation approximately invariant to small translations of the input (Goodfellow et al., 2016b).

Regularisation Techniques

Regularisation techniques aim to reduce overfitting in machine learning models. One of the most basic approaches is to add a penalty term to the loss function based on the size of the model's parameters. For example, L2 regularisation introduces the L2 norm of the parameters, $\|\theta\|_2$. The strength of this regularisation is controlled by a hyperparameter called the L2 coefficient. By incorporating this penalty, the network learns to balance minimising the loss with controlling model complexity.

Samples are fed into the network in batches (i.e., subsets of the training set) during training. Batch normalisation (Ioffe, 2015) first normalises each sample $\mathbf{x}$ in a batch by computing $\mathbf{x}' = \frac{\mathbf{x}-\mu}{\sigma}$, where μ is a vector of means and σ is a vector of standard deviations of each unit. After normalization, the data is scaled and shifted using $\gamma\mathbf{x}' + \beta$, where γ and β are learned during training. Batch normalisation helps mitigate overfitting and also accelerates the training process.

Dropout (Srivastava et al., 2014) is a simple but powerful technique to prevent overfitting. During training, each unit in the network is randomly set to zero (dropped out) with a probability p, where p is a hyperparameter. This method requires each unit to perform well regardless of other units in the model, and prevent the network from relying heavily on specific units.

Chapter 3

Related Work

Over the years, many tools have been developed to perform phylogenetic model selection and improve accuracy, some of which are widely used globally. More recently, several studies have applied machine learning techniques to address the computationally intensive nature of maximum likelihood-based methods. In this chapter, I review some of the key maximum likelihood-based tools and discuss how machine learning-based approaches are being applied in phylogenetics.

3.1 Classic Methods for Model Selection

The first widely adopted tool for DNA model selection, MODELTEST, was introduced by Posada and Crandall in 1998. This program evaluates substitution models by taking log likelihood scores and applying hierarchical likelihood ratio tests (hLRT) or calculating AIC. It supports six substitution models along with +I and +Γ. However, MODELTEST is restricted to nested models due to its reliance on hLRT, and requires prior likelihood calculations from other software. Moreover, the validity of using the χ^2 distribution for LRT statistics has been questioned when comparing models with different assumptions about rate heterogeneity or base frequencies (Whelan and Goldman, 1999). The model selection process is also highly sensitive to the order of pairwise comparisons, potentially leading to different outcomes depending on whether simpler or more complex models are evaluated (Posada and Buckley, 2004).

For amino acid model selection, ProtTest (Abascal et al., 2005) was the first software introduced, initially featuring eight substitution models, later expanded to fifteen (Darriba et al., 2011), with options for rate heterogeneity and empirical frequencies. ProtTest ranks models using AIC, AICc, or BIC.

jModelTest (Darriba et al., 2012) follows as an enhanced tool for DNA model selection using maximum likelihood. It introduced a hill-climbing algorithm for clustering models

hierarchically, and also a filter algorithm, significantly reducing the number of models that need to be evaluated. This heuristic laid the foundation for other algorithms seeking efficiency in model selection.

Building on the advancements of jModelTest, Smart Model Selection (SMS) (Lefort et al., 2017) applies an improved heuristic algorithm for protein model selection. It ranks empirical matrices according to the similarity between the amino acid frequencies of the alignment and those pre-estimated within the matrices, thus avoiding redundant likelihood calculations for both +F and -F variations. SMS demonstrates accuracy comparable to ProtTest while being more than twice as fast.

ModelFinder (Kalyaanamoorthy et al., 2017), integrated into IQ-TREE, represents the current state-of-the-art algorithm in model selection using maximum likelihood. It accommodates hundreds of substitution models, including those for DNA, proteins, and codons. ModelFinder introduced the FreeRate model for rate heterogeneity, which improves the flexibility of rate distributions by allowing arbitrary shapes and rates, leading to better model-data fit. Results have shown that it is accurate regardless of the information criterion (AIC, AICc or BIC), and it outperforms both jModelTest and ProtTest in speed and accuracy, making it the preferred tool in many phylogenetic analyses today.

ModelTest-NG (Darriba et al., 2020) was developed by some of the same authors who created jModelTest and ProtTest. The core functionalities of these earlier tools are preserved. However, ModelTest-NG is a complete reimplementation from scratch, designed to be much more efficient and better at handling large MSAs. It performs comparably to ModelFinder on protein data, and it is more accurate but slightly slower on simulated DNA data.

Different loci (i.e., groups of sites or genes) in a multiple sequence alignment may evolve under different substitution models. To account for that, advanced models such as partitioned models (Nylander et al., 2004) and mixture models (Le et al., 2008) have been introduced. Partitioning involves identifying independent substitution models for different groups of sites (typically different genes) in an MSA, which can increase tree reconstruction accuracy if the partitioning scheme is appropriate. PartitionFinder (Lanfear et al., 2012) is a widely used algorithm that iteratively merges partitions to improve the model fit. To that end, PartitionFinder evaluates the model fit of merging each pair of partitions into a new one, then chooses the best pair to merge, thus, reducing the number of partitions by one at each iteration. This process is repeated until the model fit does not increase. Results show that PartitionFinder outperforms approaches relying solely on biological intuition (e.g., codon positions) and another hierarchical clustering method (Li et al., 2008), providing more accurate partitioning schemes with significantly reduced computation time.

Although partitioned models account for heterogeneity across sites, the accuracy heavily relies on user-defined site blocks. Mixture models offer a more flexible alternative by treating each site as being governed by a mixture of different models with weights. MixtureFinder (Ren et al., 2024), implemented in IQ-TREE, uses an iterative approach

to estimate the best-fit mixture model for DNA alignments. The algorithm starts by identifying the best single-class (non-mixture) substitution model and then incrementally adds classes, optimising the mixture model by testing substitution models for each additional class. Results demonstrate that a multi-class mixture model often provides a superior fit and improves topology compared to a single-class model, suggesting that mixture models hold significant potential for more accurate phylogenetic inference.

While classic model selection methods continue to be widely used, they have some limitations. These approaches often rely on predefined model assumptions, which may not adequately capture the complexity of real evolutionary processes. Additionally, as datasets have grown larger and more complex, the computational demands of these methods have become problematic. Tools like ModelFinder can be very time-consuming when handling large genomic datasets, especially as the number of models to evaluate increases. There is a pressing need for more efficient approaches, such as those integrating machine learning techniques.

3.2 Machine learning Methods for Model Selection

The application of machine learning to phylogenetic model selection is a relatively new area of research. Currently, four studies in this domain exist, each of which is summarised in this section.

ModelTeller (Abadi et al., 2020) is an early attempt at using machine learning for model selection. It employs a Random Forest regressor (Breiman, 2001) to predict the ranking of 24 nucleotide substitution models (six base models with or without +I and/or +Γ) based on their potential to enhance the accuracy of branch length estimation. The rankings are determined by calculating the Branch Score (Kuhner and Yamato, 2015) between the phylogenetic trees inferred from each model and the true tree. ModelTeller slightly outperforms maximum likelihood in accuracy while significantly reducing computational time. However, there are a few limitations in its methodology. One concern is the simulation scheme, where some parameters were either randomly selected or kept constant instead of being generated from empirical distributions. This approach may result in unrealistic simulated data. Additionally, ModelTeller's use of Rate4Site (Pupko et al., 2002) instead of the +I+Γ model to account for rate heterogeneity across sites may also raise concerns, as Rate4Site was designed for amino acid alignments, and its applicability to DNA remains unvalidated. Besides, some of the extracted features are redundant; for example, the number of transitions can be inferred from substitution counts. Although ModelTeller performs well in selecting the optimal model for estimating branch lengths, it may not consistently choose the best model for accurately reconstructing the tree topology.

ModelRevelator (Burgstaller-Muehlbacher et al., 2023) is the first study to employ deep learning in phylogenetic model selection. It uses one neural network to identify the best DNA model from six candidates and a second network to determine whether +Γ

should be included and, if so, to estimate the α parameter. ModelRevelator performs comparably to, and sometimes better than, ModelFinder on simulated data while being more time-efficient. Nevertheless, ModelRevelator has certain limitations. It does not consider +I or FreeRate models. And its performance drops when applied to simulated data with different parameter distributions.

Machine learning was applied to amino acid model selection for the first time by Kulikov et al. (2024). This study uses fully connected neural networks to predict the best model for DNA and protein alignments with four taxa. The input consists of site-pattern frequencies, and the networks consider five DNA models and four protein models, all with +I+Γ. The networks achieve accuracy comparable to ModelFinder on simulated data. However, the simulations are based on parameter values sampled from uniform distributions and use excessively long sequence lengths to incorporate more site patterns, making the dataset unrealistic. Additionally, the networks' limitation to four-taxon alignments restricts its practical utility.

The recent ModelDetector (Nguyen Huy and Vinh, 2024) focuses on amino acid model selection, covering nine substitution models with +I+Γ. While it uses the same network architecture as ModelRevelator, it introduces a different feature extraction method that speeds up training. Although ModelDetector achieves extremely high accuracy on the simulated validation set, the absence of a test set raises concerns about potential overfitting. Additionally, the exclusion of commonly used FreeRate models further limits its applicability.

In summary, while machine learning has shown promise in model selection, significant challenges remain. These approaches often perform well on simulated data but lack validations on real datasets. Additionally, the lack of gap handling and the limited scope of models considered (e.g., excluding FreeRate models and empirical frequencies) hinder their practical adoption. Current methods also face difficulties in scaling beyond simple evolutionary scenarios. As of now, ModelFinder remains the state-of-the-art tool for phylogenetic model selection due to its flexibility, comprehensive model range, and proven performance on diverse empirical datasets.

3.3 Other Machine Learning Applications in Phylogenetics

Machine learning techniques have also been applied to various aspects of phylogenetics, and they often achieve competitive accuracy compared to classic approaches while offering significant reductions in runtime. This section discusses some of the applications apart from model selection.

3.3.1 Topology Prediction

Machine learning has been increasingly applied to topology inference. Since the number of possible unrooted tree topologies grows double-factorially with the number of taxa (e.g., a 12-taxon MSA has over 654 million possible topologies), many studies focus on

unrooted 4-taxon trees (quartets), which have only three possible topologies. Larger trees are constructed using quartet amalgamation techniques like quartet puzzling (Strimmer and Von Haeseler, 1996).

Suvorov et al. (2020) introduced convolutional neural networks for quartet topology prediction, while PhyDL (Zou et al., 2020) uses a residual neural network for the same purpose. They both achieved similar accuracy to classic methods, as well as being significantly faster. As neural networks typically require inputs of fixed size, Suvorov et al. (2020) padded shorter alignments with random values, which may introduce noise and potentially affect the model's performance. Furthermore, accuracy drops significantly when test data diverges from the conditions seen in training. Zaharias et al. (2022) found the accuracy of PhyDL combined with quartet amalgamation is lower than classical methods when applied to short alignments with long branch lengths.

Moving beyond quartet-based methods, Phyloformer (Nesterenko et al., 2022), a neural network with attention mechanisms (Vaswani, 2017), predicts pairwise distances for input into clustering algorithms like neighbour-joining. Azouri et al. (2024) introduced reinforcement learning for tree reconstruction. Both methods perform comparably to maximum likelihood on small MSAs, with performance dropping as the size of MSAs increases.

3.3.2 Choice of Inference Methods

In addition to topology prediction, machine learning has also been applied to branch length estimation (Suvorov and Schrider, 2022), site-specific evolutionary rate inference for DNA alignments (Silvestro et al., 2024) and the choice of appropriate phylogenetic inference methods.

F-zoneNN (Leuchtenberger et al., 2020) is a deep fully connected network designed to distinguish between 4-taxon alignments derived from Felsenstein-type and Farris-type trees. The concepts of the Felsenstein and Farris zones refer to regions within the parameter space of evolutionary models where different phylogenetic methods perform better or worse. Correctly identifying them allows for the selection of inference methods that improve reconstruction accuracy. Building on this work, the same authors proposed FarFelDiscerner (Leuchtenberger and von Haeseler, 2024), a simplified and more interpretable model that achieves higher accuracy than F-zoneNN. However, both networks are designed for 4-taxon alignments without gaps, simulated under the simplest JC model. This raises concerns about their generalizability to more complex evolutionary scenarios.

Chapter 4

Methods

My framework consists of three main components: (1) protFinder, a neural network, for selecting the best-fit substitution model, (2) RHASFinder, another neural network, for identifying the appropriate rate heterogeneity model, and (3) protFFinder, a random forest classifier, for determining the use of empirical pre-estimated frequencies. The entire workflow is developed in Python, with the neural networks implemented using PyTorch 1.10.0. As shown in Figure 4.1, given an input amino acid MSA, various features are extracted and fed into three classifiers to make predictions. The results are then combined to be the final predicted model.

The development of my framework consists of four major steps: (1) data generation, (2) feature extraction, (3) classifier implementation and training and (4) evaluation. These steps are detailed in the following.

4.1 Data Generation

Empirical data is often preferred for training neural networks; however, in phylogenetics, the true evolutionary history of sequences is unknown. This uncertainty means that the actual substitution model and phylogenetic tree can never be definitively determined. Consequently, researchers typically rely on simulated data that aims to broadly represent the diversity of empirical data. In this study, I simulated the training, validation and test sets based on the empirical EvoNAPS database (https://github.com/Cibiv/EvoNAPS).

The EvoNAPS database contains 21,800 amino acid MSAs, along with the corresponding models of evolution and phylogenetic trees inferred by ModelFinder, avoiding the need for additional parameter estimation. Analysis of this dataset shows that seven substitution models (LG, WAG, JTT (Jones et al., 1992), Q.bird, Q.mammal, Q.plant and Q.pfam (Minh et al., 2021)) account for 86.95% of the database, and these are among the most commonly used models. For rate heterogeneity across sites (RHAS) models, Γ and three

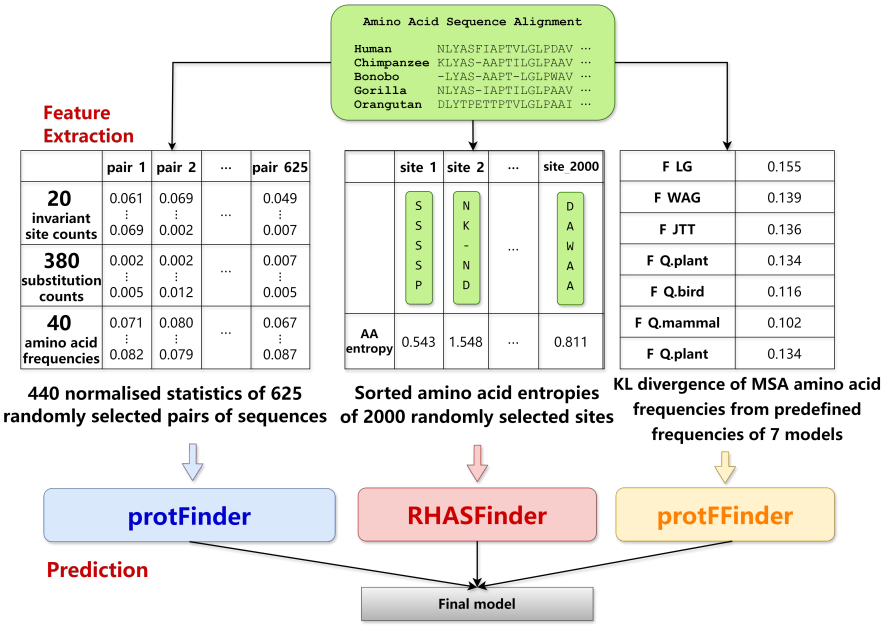


Figure 4.1: Workflow of the amino acid model selection framework.

FreeRate models—R2, R3 and R4—cover 90.26% of the database. Consequently, the data used in this study covers these seven substitution models, four RHAS models, as well as the inclusion of invariant sites (+I) and empirical frequencies (+F), resulting in 15,327 empirical MSAs from EvoNAPS. This number is slightly lower than expected due to the exclusion of problematic MSAs that were discarded during preprocessing.

Aside from the fact that the true labels of empirical alignments are unknown, even if ModelFinder's existing results are treated as ground truth, the limited number of empirical alignments may still be insufficient to effectively train machine learning models. Moreover, running ModelFinder on additional datasets requires substantial time and computational resources, which exceed my available capacity. Therefore, using simulated data also helps make for the scarcity of real data.

4.1.1 Generation of Training and Validation Sets

I used AliSim (Ly-Trong et al., 2022) to simulate alignments based on model parameters sampled from empirical distributions, which were fitted from those 15,327 real alignments. The simulation process shown in Figure 4.2 includes the following steps:

1. Parameter Selection and Outlier Removal: From the EvoNAPS database, I first extracted parameters, including branch lengths, amino acid frequencies (+F), the proportion of invariant sites (+I), the shape parameter alpha for the gamma distribution (+Γ), and rates and weights for the FreeRate models (+R). Outliers were removed to ensure robust distribution estimation.

2. Fitting Empirical Distributions: Each parameter was fitted to one of 39 candidate continuous distributions. The Kolmogorov-Smirnov test (An, 1933) was used to determine the best fit. External and internal branch lengths were estimated separately due to their different evolutionary patterns, with external branches typically being longer.

3. Generation of Random Phylogenetic Trees: Random tree topologies were generated under the Yule-Harding model (Udny Yule, 1925) for five different taxon sizes (8, 16, 32, 64 and 128). AliSim was used for topology generation. Then branch lengths sampled from the fitted distributions were assigned to the topologies.

4. Simulation of Amino Acid Sequence Alignments: The sequence length was set to 1000 amino acids. Given the seven substitution models, four RHAS models, +I, +F, and five different taxon sizes, there was a total of 700 combinations. For each combination, 350 replicates were generated for the training set and 30 for the validation set, resulting in 245,000 training data and 21,000 validation data. These settings are shown in Table 4.1.

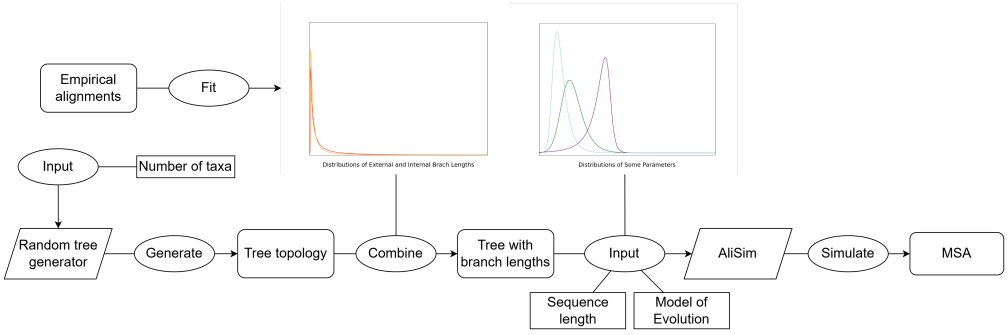


Figure 4.2: Simulation workflow of training and validation sets

Table 4.1: Settings of training, validation and test sets

Setting	Training & Validation Sets	Test Set S1
# Substitution Model	7	7
RHAS Model	[×, Γ, R2, R3, R4]	[×, Γ, R2, R3, R4]
Invariant Sites	[✓, ×]	[✓, ×]
Empirical Frequencies	[✓, ×]	[✓, ×]
Number of Taxa	[8, 16, 32, 64, 128]	[8, 16, 32, 64, 128, 256, 512]
Number of Sites	1000	[100, 500, 1000, 2000, 3000, 4000, 5000]

4.1.2 Generation of Test Sets

The first test set $S1$ was simulated in essentially the same way as training and validation sets, but more taxon sizes (256 and 512) and more sequence lengths (100, 500, 2000, 3000, 4000 and 5000) were incorporated to examine the generalisation of the classifiers (Table 4.1). Five replicates were generated for each of the 6,860 combinations, making $S1$ a size of 34,300.

The second test set, $S2$, was generated using the mimicking functionality in AliSim. Specifically, AliSim internally runs IQ-TREE to infer a maximum likelihood tree and identify the best-fit model from a real alignment, then using the inferred information as input to generate new alignments. In this study, the information in EvoNAPS was directly used, avoiding running IQ-TREE again. During that process, gaps from the real alignment are replicated in the simulated data. The size of the alignment is also preserved. This approach aims to replicate the evolutionary history of the original empirical alignments. For each empirical alignment, two replicates were generated, resulting in 30,654 test data. The $S2$ dataset is imbalanced and more closely resembles the empirical dataset compared to $S1$. Additionally, $S2$ includes gaps, which are not present in the training and validation sets.

Another option to evaluate our framework is using real alignments. However, testing on real data is more complicated due to the unknown ground truth and the complex nature of evolution processes (e.g., a single substitution model was demonstrated to be inadequate to represent the complicated evolution in empirical data (Ren et al., 2024)). Therefore, I will discuss the testing results on real data in a dedicated section 5.7 below.

4.2 Feature Extraction

In this section, I describe the feature extraction methods employed for protFinder, RHASFinder and protFFinder. These methods extract fixed-size features from alignments and can be applied to alignments of any size.

4.2.1 Feature for protFinder

Since the substitution rate matrix is the core determinant of a substitution model, pairwise substitutions are the key features for protFinder. Moreover, amino acid frequencies can be useful to mitigate the disturbance of +F. I randomly sampled (with replacement) 625 pairs of sequences from each alignment and compute 440 statistics for each pair. These statistics included 20 invariant site counts (e.g., $A \leftrightarrow A$, $R \leftrightarrow R$), 40 amino acid counts in the two sequences (i.e., 20 amino acids $\times$ 2 sequences), and 380 directional pairwise substitution counts (e.g., $A \rightarrow R$ and $R \rightarrow A$). All statistics were normalised by the sequence length, resulting in an input shape of 625×440.

Some values for the number of random pairs, ranging from 10,000 to 225, were tested. Although 625 is fewer than the total number of unique pairs when the number of taxa

is greater than 25, it was found to be sufficient for the network to achieve comparably high accuracy while maintaining fast feature extraction.

4.2.2 Feature for RHASFinder

The classic approach for distinguishing RHAS models involves calculating site-specific substitution rates using maximum likelihood, followed by examining their distribution, which is computationally intensive. For RHASFinder, I seek a statistic that reflects the characteristics of site-specific rates: a value that reaches its minimum when only one amino acid is present at a site and its maximum when all 20 amino acids are equally represented.

I chose site-specific Shannon entropy (Shannon, 1948) of amino acid frequencies as the feature for RHASFinder. For each site, entropy S is calculated as $S = -\sum_{i=1}^{20} f_i \log f_i$ where f_i is the count of amino acid $i \in [1, 20]$ normalised by the number of taxa (i.e., the frequency of the amino acid i). $S = 0$ when only one amino acid is present at a given site ($f_i = 1$ for some i, and $\sum_{j \neq i} f_j = 0$), and it reaches its maximum value of $\log(20)$ when all amino acids are uniformly distributed ($f_i = \frac{1}{20}\ \forall i \in [1, 20]$).

To standardise the input size, I randomly selected 2,000 sites (with replacement) from each alignment and compute the corresponding entropy values, which were then sorted in ascending order. This process produced a feature vector of length 2,000, and the sorting step turned out to significantly enhance RHASFinder's performance. Similarly, some numbers of sites were tested, and it was determined that 2,000 sites were sufficient for the network to generalise well to longer alignments.

4.2.3 Feature for protFFinder

The key to distinguishing between +F and without +F lies in the amino acid frequencies. To capture this difference, seven Kullback–Leibler (KL) divergences (Kullback and Leibler, 1951) were computed as

$$D_{KL} = \sum_{i=1}^{20} f_{MSA,i} \log \frac{f_{MSA,i}}{f_{ref,i}}$$

where $f_{MSA,i}$ represents the frequency of the i-th amino acid in the alignment, and $f_{ref,i}$ is the predefined frequency of the i-th amino acid in one of the seven substitution models. This allowed the amino acid frequencies of the alignment to be compared against those of all the candidate models.

If the true substitution model were known, it would only be necessary to compare the amino acid frequencies of the MSA against those of the true model. However, since neither protFinder nor ModelFinder is certain to achieve 100% accuracy in substitution model selection, relying solely on previous results is not advisable. Additionally, it was observed that using only the amino acid frequencies of the MSA as feature led to a slight decrease in performance.

4.2.4 Gap Handling

To handle gaps in the test set $S2$, two methods are considered:

- Replace: Gaps are replaced with the most frequent amino acid at the corresponding site.
- Ignore: Gaps are excluded from the analysis. For protFinder, in each pair of sequences, sites with gaps are removed, and amino acid frequencies are calculated from the remaining sites, normalised by the new sequence length. For RHASFinder, gaps are removed for each site, and amino acid frequencies are normalised by the remaining number of taxa.

4.3 Classifier Implementation and Training

This section outlines the architectures of the two neural networks, protFinder and RHASFinder, and also presents hyperparameter tuning and training settings of all three classifiers. Automatic Mixed Precision in Pytorch was employed to accelerate neural network training and reduce memory usage.

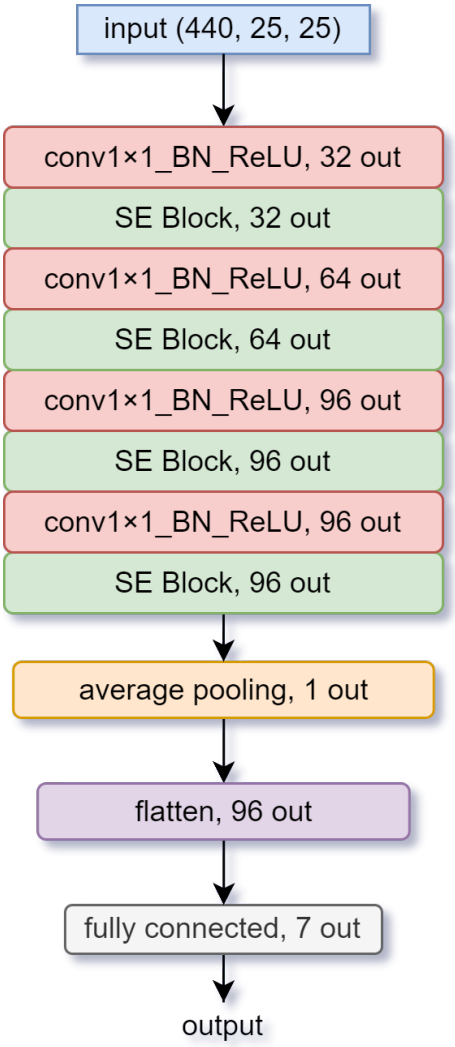
During the development process, numerous neural network architectures have been explored, including fully connected networks, convolutional networks, LSTM (Hochreiter, 1997), attention mechanisms, ResNet (He et al., 2016), ViT (Dosovitskiy, 2020) and different combinations of these, along with modifications to individual layers. It turned out that very complex networks with tens of millions of parameters did not necessarily yield the best results. Some hyperparameters were tuned in earlier networks (e.g., optimiser) or automatically adjusted during training (e.g., learning rate). Here I only present the hyperparameter tuning stage for the final architecture, as detailed below, and do not address modifications to network layers or other hyperparameters that were tuned earlier.

4.3.1 protFinder

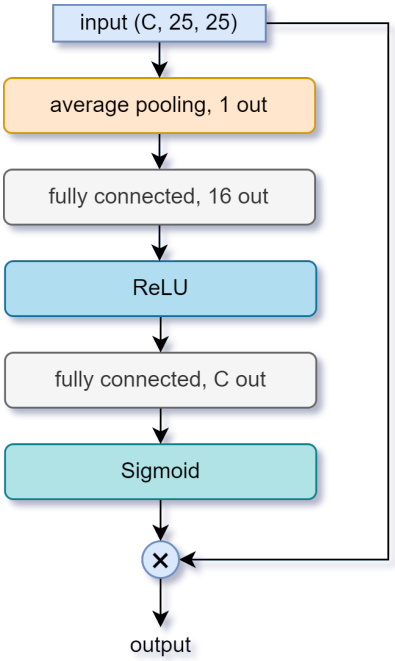
Architecture of protFinder

protFinder is a combination of convolutional layers and Squeeze-and-Excitation (SE) blocks (Hu et al., 2017) (Figure 4.3a). The network structure starts with four Conv-SE blocks (Figure 4.3b), each composed of one convolutional layer followed by an SE block. An SE block begins by applying global average pooling to compress each channel into a single value. This output is then passed through a fully connected layer with a ReLU activation function (Agarap, 2018), followed by another fully connected layer with a Sigmoid activation function, producing a 1D output, namely channel-wise weights. This output is subsequently multiplied element-wise with the original input of the SE block. After these Conv-SE blocks, an average pooling layer is used to reduce the input size before it is passed to a fully connected layer, which outputs the final 7-class prediction.

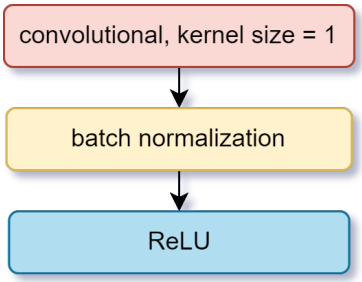
Each convolutional layer is also followed by a Batch Normalisation layer and a ReLU activation function to stabilise and accelerate training (Figure 4.3c).



(a) protFinder



(b) SE block



(c) conv1×1_BN_ReLU

Figure 4.3: Architecture of protFinder and its components

Hyperparameter Tuning for protFinder

The input to protFinder, consisting of 440 statistics derived from 625 randomly selected sequence pairs, is reshaped to $440 \times 25 \times 25$ (channels, width, height). Table 4.2 shows the hyperparameter tuning process for protFinder. An uncommon aspect of protFinder is its use of a 1×1 kernel in the convolutional layers. Convolutions with a kernel size

Table 4.2: Hyperparameter tuning of protFinder. The final ones are shown in bold.

Hyperparameter	**Values**
Batch size	40, **64**, 128
Number of squeezed channels in SE block	8, **16**, 32
Kernel size of convolutional layers	$\mathbf{1 \times 1}$, 2×1, 2×2
Number of channels in last convolutional layer	**96**, 128, 256

of 1 function similarly to fully connected layers, as they do not process local spatial information within the individual channels but instead operate across the channel dimension. These 1×1 convolutions focus on the relationships between the 440 statistics at each spatial location (sequence pair), without explicitly modeling interactions between sequence pairs. The subsequent average pooling layer aggregates the spatial information across the 25×25 grid.

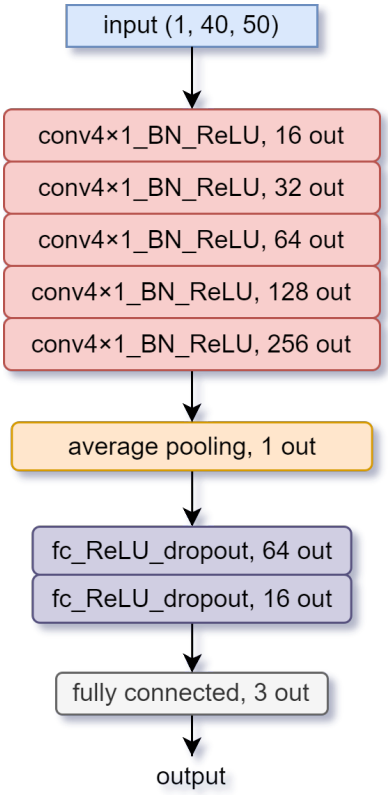
The AdamW optimiser (Loshchilov, 2017) is employed with an initial learning rate of 0.001. L2 regularisation is applied to all network parameters with a coefficient of 0.0001 to prevent overfitting. A learning rate scheduler is used to adapt the learning rate dynamically: if the validation accuracy does not improve for 5 epochs, the learning rate is reduced by a factor of 0.1. Early stopping is also employed, where training is stopped if there is no improvement in validation accuracy for 10 consecutive epochs.

4.3.2 RHASFinder

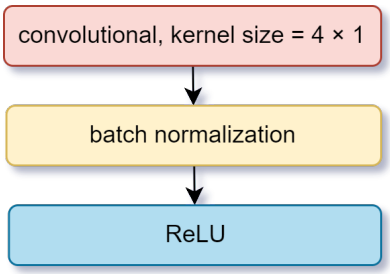
It is important to note that while rate homogeneity and the four RHAS models—Γ, R2, R3 and R4—each has an equal number of alignments, the classification task of RHASFinder was reduced to three classes. R2, R3, and R4 were grouped into a single class R because distinguishing between them proved challenging after several attempts. A similar difficulty was observed with ModelFinder. This adjustment introduced an imbalance in the dataset, leading to low testing accuracy for the Γ model, because the network learned to predict the majority class (R) more often to minimise the training loss. To mitigate this effect, class weights of $[3, 3, 1]$ were applied to the None, Γ, and R classes during training. This strategy was designed to encourage the network to pay more attention to the None and Γ classes. It turned out that the accuracy for Γ significantly improved.

Architecture of RHASFinder

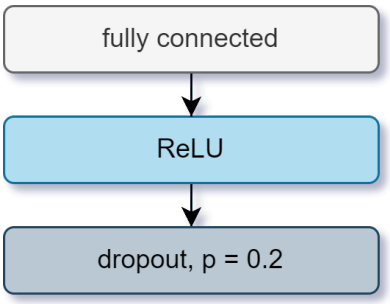
RHASFinder consists of five convolutional layers followed by three fully connected layers (Figure 4.4a). Similar to protFinder, each convolutional layer is followed by a Batch Normalisation layer and a ReLU activation function (Figure 4.4b). For the fully connected layers, a ReLU activation function is followed by a dropout layer to further prevent overfitting (Figure 4.4c).



(a) RHASFinder



(b) conv4×1_BN_ReLU



(c) fc_ReLU_dropout

Figure 4.4: Architecture of RHASFinder and its components

Hyperparameter Tuning for RHASFiner

Table 4.3 shows the process of hyperparameter tuning. The optimiser, learning rate schedule, and early stopping criteria are the same as those used for protFinder.

Table 4.3: Hyperparameter tuning of RHASFinder. The final ones are shown in bold.

Hyperparameter	**Values**
Batch size	40, 64, **128**, 256
L2 penalty	0, **0.0001**, 0.001
Padding method of convolutions	**same**, valid
Pooling method	**average**, max
Kernel size of convolutional layers	**4 × 1**, 4 × 2, 4 × 4, 5 × 5
Number of channels in last convolution layer	128, **256**, 512
Dropout	**0.2**, 0.3

4.3.3 protFFinder

As random forest classifiers do not have a neural network-like architecture, this section only presents the hyperparameter tuning for protFFinder (Table 4.4), which focuses on n_estimators and max_depth.

A grid search with 5-fold cross validation was conducted to find best hyperparameters of protFFinder. Grid search exhaustively tries every combination of the provided hyperparameter values. In 5-fold cross-validation, the data is evenly divided into five subsets, and the process is repeated five times, each time using a different subset as the validation set and the remaining four as the training set. For this process, the original training and validation sets used for the neural networks were combined to form the entire dataset.

Table 4.4: Hyperparameter tuning of protFFinder. The final ones are shown in bold.

Hyperparameter	**Values**
n_estimators	10, 50, 100, **150**, 200, 250, 300
max_depth	10, 20, **30**, 40, 50, 60, 70, 80, 90, 100

n_estimators is the number of trees and max_depth indicates the maximum depth of each tree. As n_estimators increases, the accuracy tends to improve but may eventually level off. Larger values for max_depth may lead to overfitting as the trees become too complex.

4.4 Evaluation

This section outlines the hardware used for this research and explains how the performance of protFinder, RHASFinder, and protFFinder was evaluated. The detailed results are in the next chapter.

The performance of three classifiers was benchmarked against the state-of-the-art ModelFinder. For a fair comparison, the candidate models examined by ModelFinder were restricted to the same set used by my classifiers. It is important to note that ModelFinder was only tested on MSAs with up to 2000 sites, as the time cost for longer MSAs was prohibitively high. Model selection in ModelFinder was based on BIC.

4.4.1 Hardware Setup

All experiments and measurements were conducted either on the server at the Center for Integrative Bioinformatics Vienna (CIBIV) or the Gadi supercomputer at the National Computational Infrastructure (NCI). The detailed system configurations are provided in Table 4.5. I first ran tasks including feature extraction and testing ModelFinder on CIBIV. However, due to the outdated NVIDIA driver on CIBIV, I had to move to Gadi to train and test my machine learning models.

Table 4.5: Configurations of CIBIV and Gadi

Server	Operating System	CPU	GPU
CIBIV	openSUSE Leap 15.5	AMD Opteron Processor 6380	N/A
Gadi	Rocky Linux 8.10	Intel Xeon Platinum 8268 CPU	NVIDIA V100

The training of both protFinder and RHASFinder was conducted on Gadi, utilising 4 NVIDIA V100 GPUs, 48 CPU cores, and 210GB of reserved memory. For testing, GPU-based experiments used 1 NVIDIA V100 GPU with 12 CPU cores and 48GB of reserved memory, while CPU-based tests used 1 CPU core with 4GB of reserved memory. Feature extraction and ModelFinder execution were performed on CIBIV using a single CPU core. No multi-threading was applied in any of the experiments. The time cost measurements in the next chapter refer to walltime.

4.4.2 Evaluation Metrics

Three classifiers and ModelFinder were tested on the test set $S1$. The accuracy (i.e., the proportion of correct predictions) for each substitution model and RHAS model, across different taxon sizes and site sizes, will be presented in the next chapter to highlight the discrepancies among models and the impact of the number of sites and taxa on performance.

protFinder was also evaluated on the test set $S2$, which is an imbalanced dataset designed to mimic the complex patterns observed in empirical MSAs. In cases where a multi-class dataset is highly imbalanced, using Macro F1 score or Weighted F1 score is generally more appropriate than accuracy as a performance indicator. While the weighted F1 score adjusts for class imbalance by assigning weights based on the number of instances in each class, it is unsuitable here because all substitution models should be treated with equal importance. The macro F1 score, which is the arithmetic mean of class-wise F1 scores, was used to examine profFinder's performance. It is calculated as $\frac{2}{n}\sum_1^n \frac{Precision \times Recall}{Precision + Recall}$ where n is the number of classes. Precision is defined as the number of true positives divided by the total number of true positives and false positives, and recall is the number of true positives divided by the total number of true positives and false negatives.

Chapter 5

Results

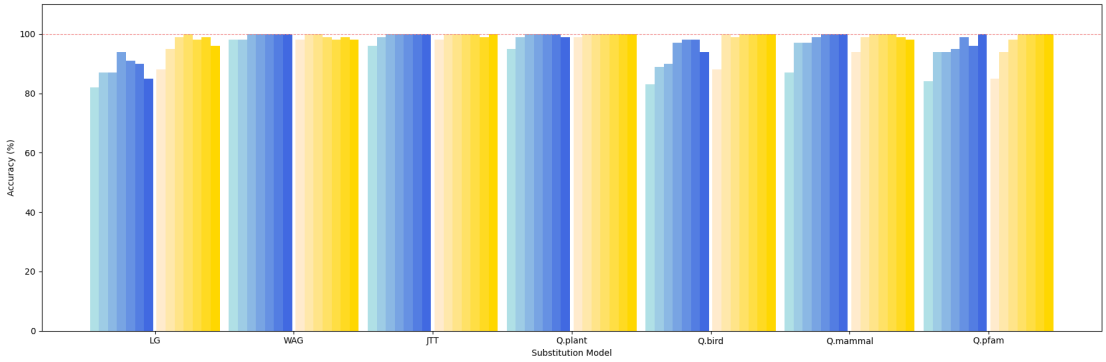
This chapter presents a comparison of the results from protFinder, RHASFinder and protFFinder against the maximum-likehood based ModelFinder.

5.1 Substitution Model Prediction

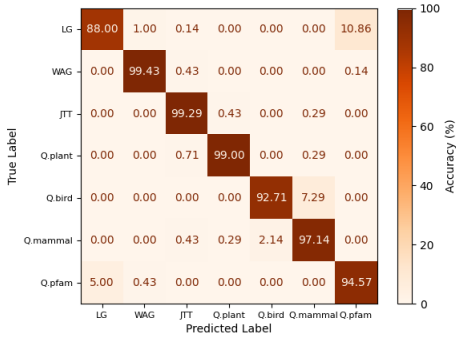
This section shows protFinder's performance tested on three sequence length settings: 1,000 sites, which corresponds to the sequence length of the alignments used to train protFinder; 2,000 sites, the longest sequence length I was able to run ModelFinder on; and 100 sites, to evaluate protFinder's performance on very short alignments.

Figure 5.1a presents the accuracy of protFinder and ModelFinder on MSAs with 1000 sites, categorised by substitution model and number of taxa (n_{taxa}). In general, accuracy improves as the number of taxa increases, with ModelFinder achieving nearly 100% accuracy when $n_{taxa} \geq 64$. Similarly, protFinder reaches nearly 100% accuracy for four of the models, while maintaining over 90% accuracy for Q.bird and Q.pfam. For the LG model, however, protFinder's accuracy drops to between 80% and 90%. When $n_{taxa} \leq 32$, the performance of both methods is negatively impacted, but they remain effective overall.

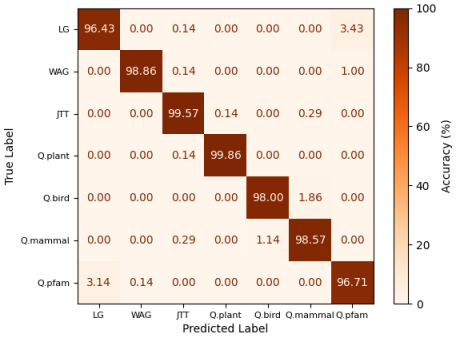
The specific misclassifications can be observed in Figures 5.1b and 5.1c. Both methods consistently exhibit confusion between LG and Q.pfam, as well as between Q.bird and Q.mammal, with this issue being more obvious for protFinder. To investigate the similarities among these models, their matrices of amino acid exchangeabilities are visualised using principal component analysis (PCA) (Pearson, 1901). After being reduced to two principal components, the distribution of the seven substitution models is displayed in Figure 5.2. It is evident that LG and Q.pfam are close to each other, as are Q.bird and Q.mammal. This explains the observed misclassifications, as the models are essentially similar, making them more difficult for both methods to distinguish accurately.



(a) Accuracy of protFinder (blueish) and ModelFinder (yellowish) on 1000-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.



(b) Confusion matrix for protFinder.



(c) Confusion matrix for ModelFinder.

Figure 5.1: Accuracy and confusion matrices for protFinder and ModelFinder on 1000-site MSAs.

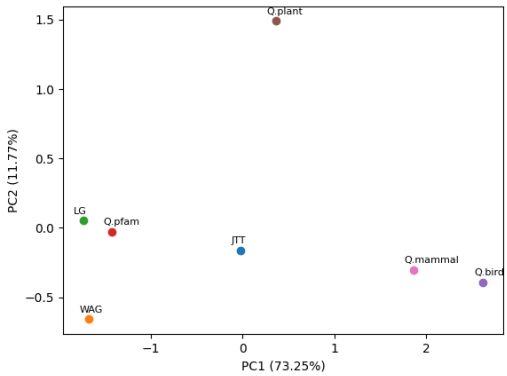
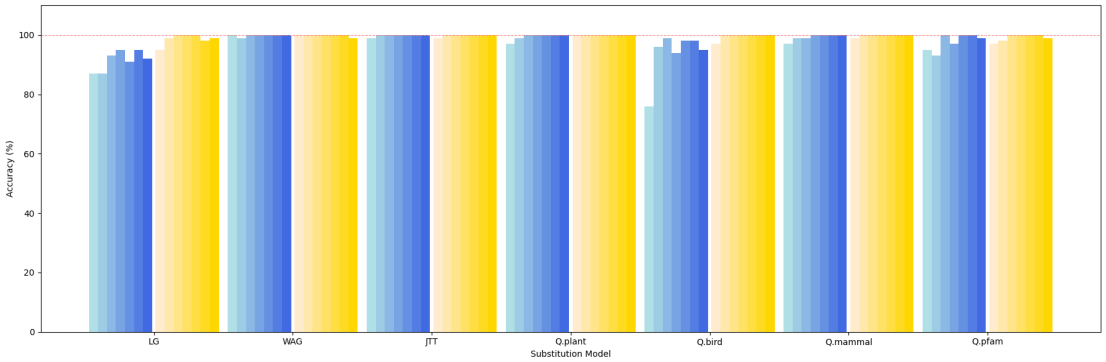
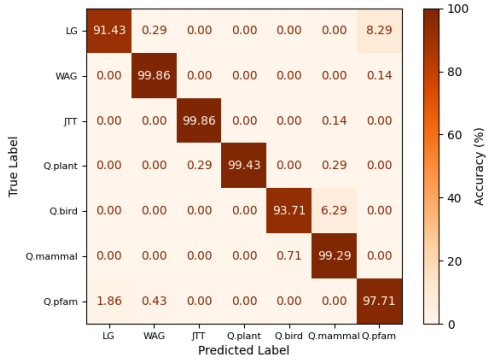


Figure 5.2: PCA result of matrices of amino acid exchangeabilities.

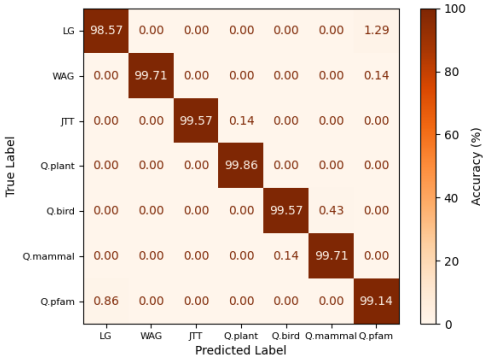
As shown in Figure 5.3, the increased sequence length from 1000 to 2000 sites has a positive impact on both methods' accuracy across all substitution models. Longer alignments provide more information for the network to learn from, allowing protFinder to improve its performance and reduce the gap between it and ModelFinder. ModelFinder, on the other hand, shows modest improvements as it already performs closely to optimal for many models and taxon sizes in the 1000-site case. The misclassification pattern observed with 1000-site MSAs is also present in Figures 5.3b and 5.3c. protFinder's performance on the LG model is relatively low across all n_{taxa}, while the majority of misclassifications for Q.bird occur in smaller MSAs with 8 taxa.



(a) Accuracy of protFinder (blueish) and ModelFinder (yellowish) on 2000-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.

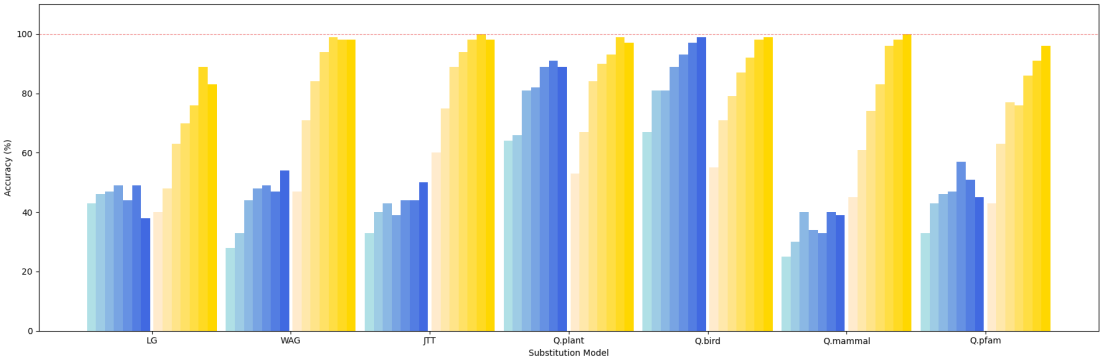


(b) Confusion matrix for protFinder.

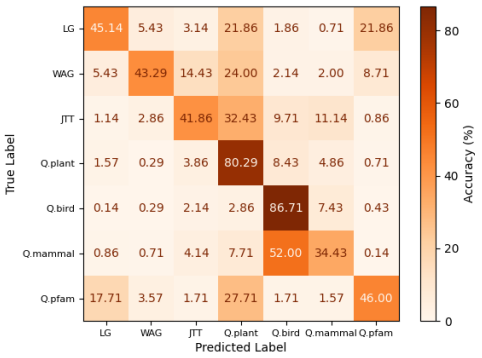


(c) Confusion matrix for ModelFinder.

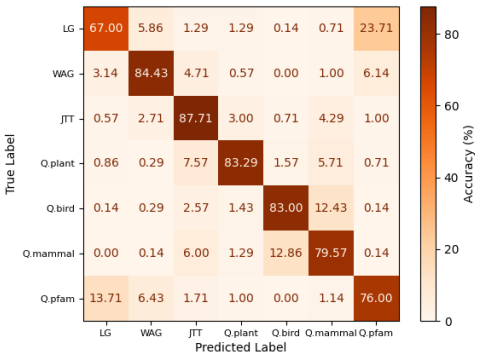
Figure 5.3: Accuracy and confusion matrices for protFinder and ModelFinder on 2000-site MSAs.



(a) Accuracy of protFinder (blueish) and ModelFinder (yellowish) on 100-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.



(b) Confusion matrix for protFinder.



(c) Confusion matrix for ModelFinder.

Figure 5.4: Accuracy and confusion matrices for protFinder and ModelFinder on 100-site MSAs.

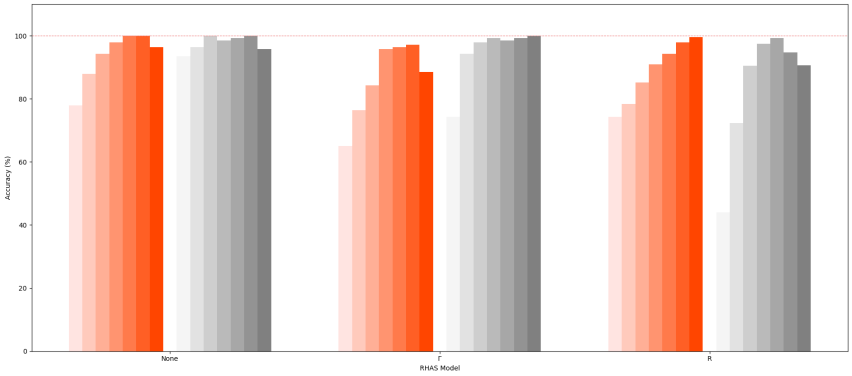
In my tests, as the number of sites decreases to 500, the overall accuracy of both methods declined slightly, and protFinder remains comparable to ModelFinder. However, for the 100-site MSAs, as shown in Figure 5.4, ModelFinder demonstrates an overwhelming advantage over protFinder, with the exception of the Q.plant and Q.bird models. The poor performance of protFinder suggests that the network struggles to extract sufficient phylogenetic information from such short alignments using the current feature. While protFinder processes pairwise information, ModelFinder examines the whole alignment, potentially leading to information loss for protFinder. Although ModelFinder retains a higher degree of robustness, its accuracy also drops significantly on MSAs with smaller n_{taxa}.

Furthermore, the misclassification pattern of protFinder across 1000 and 2000-site MSAs is completely disrupted on the 100-site alignments (Figure 5.4b). A probable reason is that with longer sequence alignments, the patterns of substitution become more distinct, and subtle differences between models can be better captured using the extracted feature,

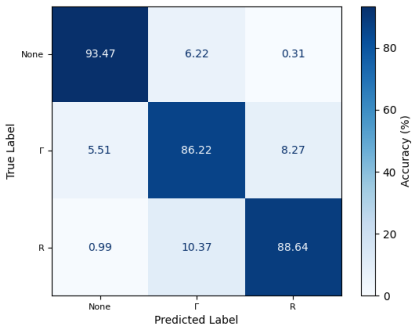
allowing protFinder to maintain consistent classification behaviour. In contrast, shorter alignments introduce greater variance, making it more difficult for the network trained on 1000-site MSAs to generalise effectively. These results indicate that short sequence lengths present challenges for both machine learning-based and maximum likelihood-based model selection approaches.

5.2 RHAS Model Prediction

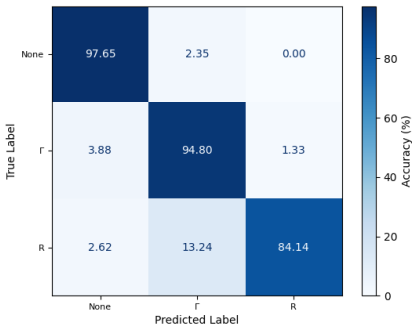
Figure 5.5 illustrates that RHASFinder and ModelFinder perform comparably for alignments with $n_{taxa} \geq 64$ for the None and Γ models. Both methods experience a decline in accuracy when the number of taxa is small. Overall, RHASFinder is outperformed by ModelFinder on the Γ model, while it shows slightly better performance on the R model. This discrepancy may be attributed to the residual effects of data imbalance, even with the application of class weights.



(a) Accuracy of RHASFinder (reddish) and ModelFinder (greyish) on 1000-sites MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.



(b) Confusion matrix for RHASFinder.

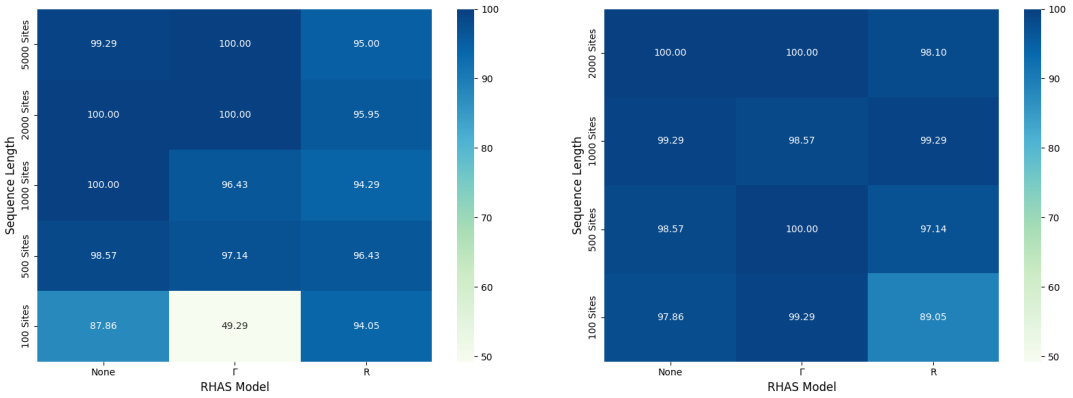


(c) Confusion matrix for ModelFinder.

Figure 5.5: Accuracy and confusion matrices for protFinder and ModelFinder on 100-site MSAs.

Unlike the classification of substitution models, the number of sites has a limited impact on the accuracy of both methods for the RHAS models. As shown in Figure 5.6, the accuracy of both RHASFinder and ModelFinder for all three RHAS models generally varies as the sequence length increases from 500 to 2000 sites.

The main exceptions are the performance of RHASFinder on 100-site alignments for the None and Γ models, and ModelFinder on 100-site alignments for the R model, where a more pronounced drop in accuracy is observed. This again indicates that the insufficient information in 100-site alignments negatively affects the performance of both methods, though with different emphases. A fixed number of random sites (2000) is selected with replacement to calculate the input of RHASFinder, which likely explains the consistent performance of RHASFinder across different sequence lengths. It also indicates that 2000 sites are enough for the network to generalise well for longer alignments, as seen with the 5000-site MSAs in Figure 5.6a.



(a) RHASFinder. (b) ModelFinder.

Figure 5.6: Accuracy of RHASFinder and ModelFinder for 128-taxon MSAs with different number of sites and RHAS models.

5.3 With/Without +F Prediction

Figure 5.7 shows the accuracy of protFFinder and ModelFinder for classifying with and without +F. Both methods perform almost perfectly on long alignments. However, ModelFinder's accuracy shows a steep drop on 100-site MSAs, while protFFinder's performance is close to random guessing.

This is likely due to the unreliable KL divergences derived from the 100-site alignments. KL divergence measures the difference between two distributions, but its reliability heavily depends on having sufficient data to accurately estimate the amino acid frequencies. With only 100 sites, the observed amino acid frequencies may not accurately reflect the true evolutionary process.

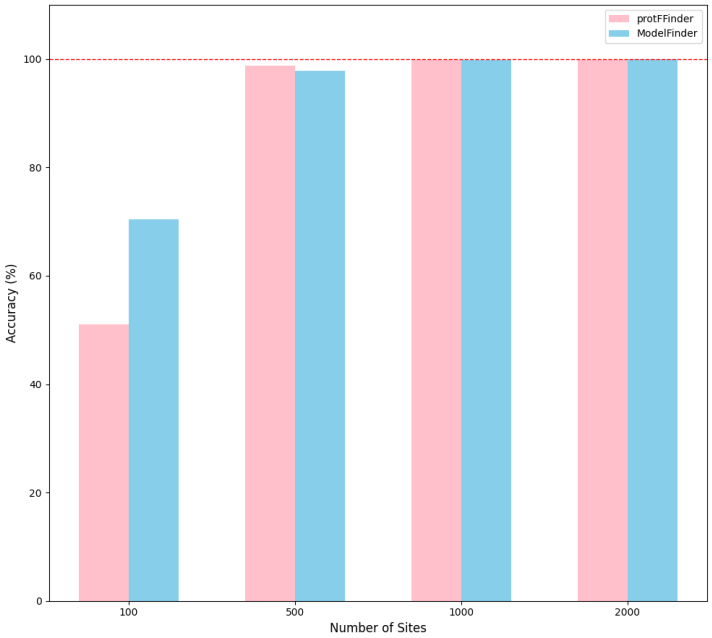
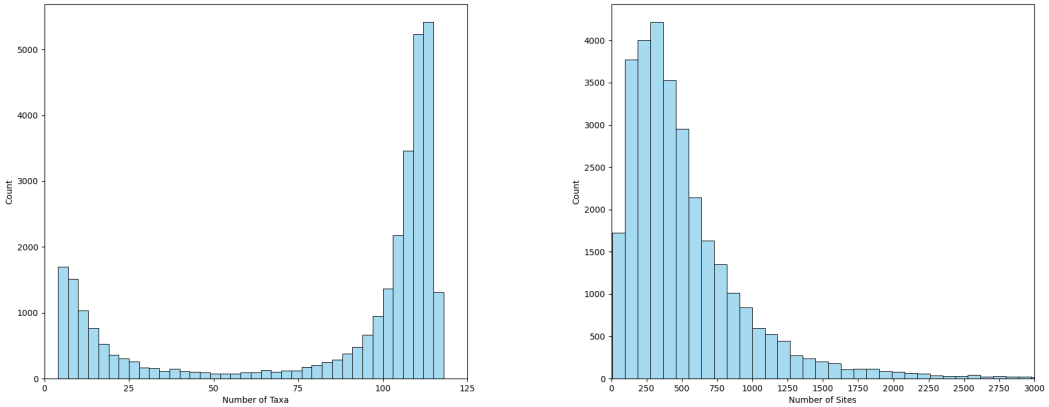


Figure 5.7: Accuracy of protFFinder and ModelFinder on MSAs with varying sequence lengths

5.4 Generalisation to MSAs with Gaps

To evaluate the generalisation of my framework on gapped alignments, I tested protFinder on the $S2$ test set which contains MSAs with gaps and various number of sites and number of taxa as summarised in Figure 5.8.



(a) (b)

Figure 5.8: Count of MSAs by number of taxa (left) and number of sites (right).

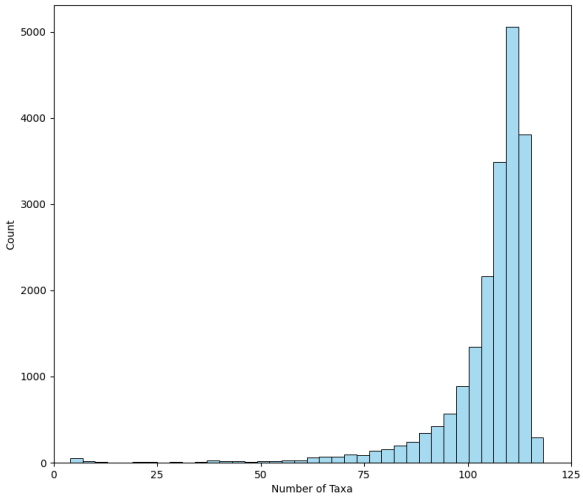
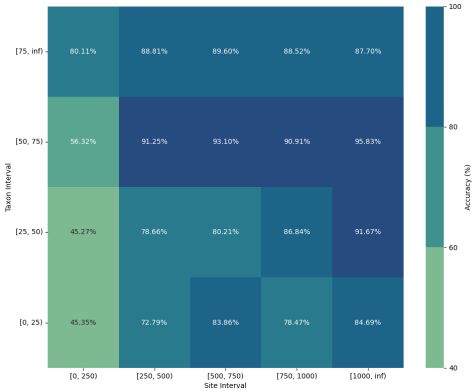
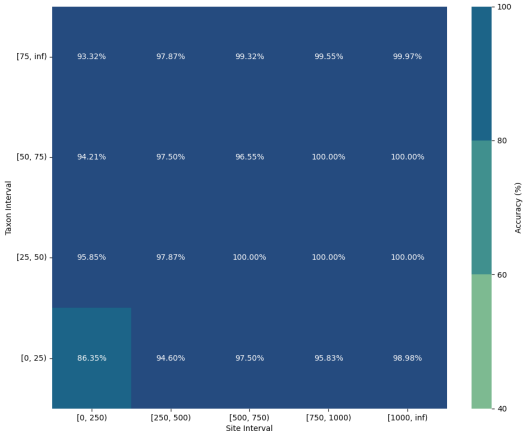


Figure 5.9: Count of MSAs of Q.bird model by number of taxa

(a) protFinder: gaps are ignored



(b) protFinder: gaps are replaced

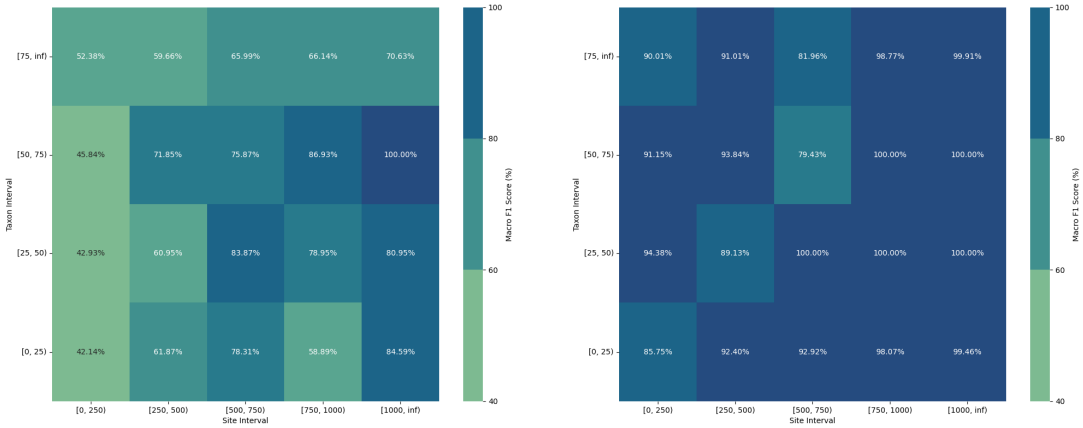


(c) ModelFinder

Figure 5.10: Accuracy of protFinder with two gap-handling methods and ModelFinder, and the number of MSAs by site interval and taxon interval

To present the performance, site intervals and taxon intervals are used. Figure 5.10 displays the accuracy of protFinder and ModelFinder in each interval. ModelFinder performs well on $S2$, with over 95% accuracy on most alignments (Figure 5.10c), which demonstrates the robustness of maximum likelihood-based methods on relatively small alignments with gaps. The high degree of randomness in the number and position of gaps poses challenges for protFinder, which was trained on MSAs without gaps. The two gap-handling methods produces similar results, and in general, accuracy increases with alignment size (Figure 5.10a and 5.10b).

However, when examining the substitution models in $S2$, Q.bird accounts for the largest proportion at 64.51%. In this case, the macro F1 score provides a more accurate reflection of protFinder's overall performance, as discussed in Section 4.4.2.



(a) protFinder (b) ModelFinder

Figure 5.11: Macro F1 score of protFinder (ignoring gaps) and ModelFinder.

Figure 5.11 shows the macro F1 scores of protFinder and ModelFinder. ModelFinder has over 90% score for most alignments, indicating minimal bias in selecting the substitution model. However, when comparing Figure 5.10a with Figure 5.11a, we observe that most scores of protFinder decrease, particularly in the top two rows ($n_{taxa} \geq 50$) of the heatmaps.

Further comparison of Figure 5.8a and Figure 5.9 reveals that the Q.bird model dominates alignments with $n_{taxa} \geq 50$. The significantly lower macro F1 score compared to overall accuracy for these alignments indicates that the model performs well on Q.bird but poorly on one or more other models. This is consistent with Figure 5.4a, where the accuracy for classifying the Q.bird model is the highest among all models. Additionally, the gap between the macro F1 score and accuracy narrows as the number of sites increases, as seen from left to right in the first row of Figures 5.10a and 5.11a. This is because protFinder's advantage in classifying Q.bird diminishes as the number of sites increases, as demonstrated in Figures 5.1a and 5.3a.

5.5 Time Measurement

5.5.1 Runtime comparison between our framework, ModelFinder, and a combination approach

The workflow consists of two steps: preprocessing and prediction. The preprocessing step involves converting alignments into integer matrices and performing feature extraction. The time required for file conversion is minimal and scales proportionally with the size of the MSA.

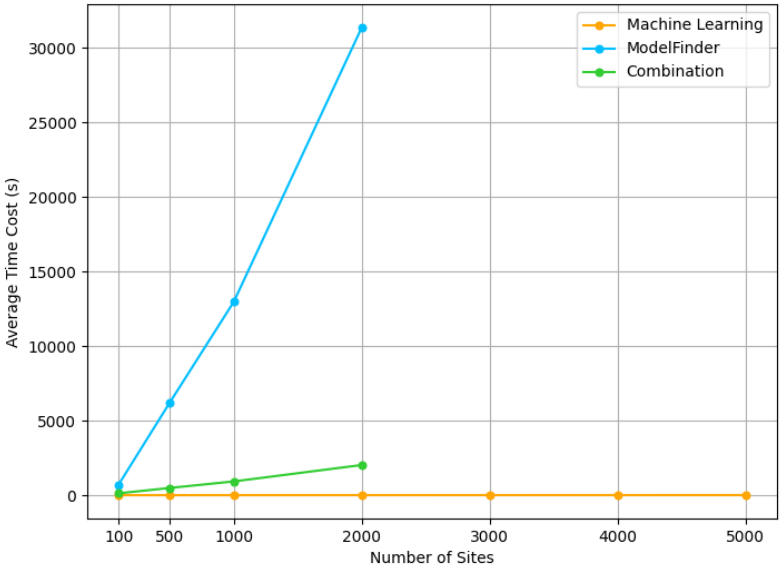


Figure 5.12: Average runtimes of three methods for 512-taxon MSAs. The orange line represents the total time of file conversion, feature extraction and prediction using protFinder, RHASFinder and protFFinder with a GPU. The blue line indicates the time cost of ModelFinder. The green line shows the combination approach.

The runtime of my framework is the total runtime of data preprocessing and prediction of the three classifiers. While the classifiers can predict the substitution model, the RHAS model (None, Γ or R), and with/without +F, ModelFinder offers additional capabilities. It can distinguish between R2/3/4 models, provide with/without +I, and most importantly, estimate parameters. It would be unfair if I directly compare the runtime of my framework and ModelFinder. Therefore, a combination approach is applied.

First, the three machine learning models are used to predict the substitution model, None/Γ/R, and with/without +F. Based on these predictions, ModelFinder is then

employed to determine the final, complete model. This approach reduces the number of models that ModelFinder needs to examine—from 140 down to 2 if the RHAS model is None or Γ, or 6 if it is R. It is now more reasonable to compare the runtimes of ModelFinder and the combination approach as that approach can both predict the best-fit model and estimate its parameters.

Figure 5.12 shows the average runtimes (per alignment) of the pure machine learning framework, the combination approach, and ModelFinder when tested on 512-taxon alignments of different sequence lengths. In general, the machine learning framework and the combination approach are much faster than ModelFinder. For example, ModelFinder takes about 8.7 hours to infer the best-fit model and estimate its parameters for one alignment of 2000 sites, while the combination approach and the pure machine learning framework require 7.9 seconds and 33.8 minutes, respectively (Figure 5.12), which are 3,973.2x and 15.5x speedups.

5.5.2 Runtime Analysis of protFinder

This section presents the time cost of extracting the feature used for protFinder, along with the time cost of its prediction.

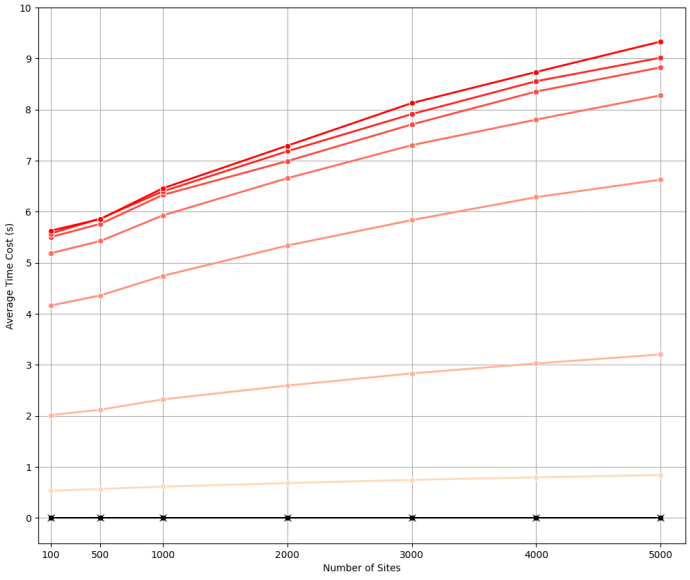


Figure 5.13: Average time cost of protFinder for one MSA. The reddish lines represent the feature extraction time, with shading indicating the number of taxa (from light to dark): 8, 16, 32, 64, 128, 256, and 512. The prediction time using CPU and GPU is indicated by black −×− and −□−, respectively.

The input of protFinder includes 440 statistics derived from 625 randomly selected sequence pairs. As the number of pairs is fixed at 625, we may expect the feature extraction time is regardless of the number of taxa. However, the actual runtime is influenced by both the number of sites and the number of taxa as shown in Figure 5.13. It is, in fact, due to my optimised implementation of feature extraction, as described in the following.

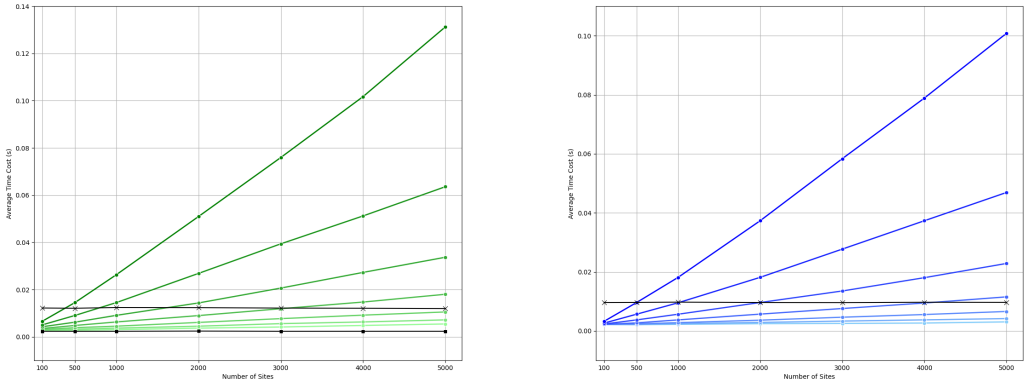
To compute amino acid frequencies for each pair, the frequencies are first calculated for all sequences and then retrieved as needed. For the computationally expensive step of calculating invariant and replacement counts, we only compute for the unique pairs (note that (seq_1, seq_2) is different from (seq_2, seq_1)), and store them for reuse. As the number of taxa increases, the number of unique pairs occurring may also grows, resulting in a higher time cost.

The increase in time cost when moving from 8 to 16 taxa and from 16 to 32 taxa is much larger compared to later intervals (e.g., 64 to 128 taxa). This is due to the rapid increase in the number of unique sequence pairs and the fixed number of selected pairs. When transitioning from 8 to 16 taxa, the number of unique pairs increases from 56 to 240, which is a considerable jump, and it is likely that all unique pairs are involved in the 625 selected pairs. Similarly, moving from 16 to 32 taxa increases the number of unique pairs from 240 to 992, requiring more computation. As the number of taxa continues to grow, the absolute number of unique pairs still increases, but the total number of pairs is constrained by the fixed 625 pairs. Consequently, statistics are calculated for most of the 625 selected pairs, and n_{taxa} has a minimal impact on the time cost.

An interesting observation is that prediction times using CPU and GPU are nearly identical (around 0.0035 seconds). This is likely because protFinder has a small number of parameters (42,311), and CPU-based predictions are already very fast. Additionally, the alignments are processed one at a time, with a batch size of 1. It does not fully utilise the parallel processing capabilities of the GPU and suffers from high overhead due to frequent data transfers between the CPU and GPU, which outweighs the potential computation speed advantage of the GPU.

5.5.3 Runtime Analysis of RHASFinder and protFFinder

Feature extraction times for both RHASFinder and protFFinder (Figure 5.14) are exactly proportional to the number of sites and the number of taxa. This behaviour is expected for protFFinder due to the need to calculate overall amino acid frequencies. Although the number of randomly selected sites is fixed at 2000 for RHASFinder, the time cost still increases with the total number of sites in the MSA because the site-specific entropy is computed for all sites before selecting the 2000 used as input. Additionally, RHASFinder's prediction speed on the GPU is five times faster than on the CPU, which aligns with the fact that RHASFinder has much more parameters (193,171) compared to protFinder.



(a) RHASFinder. (b) protFFinder.

Figure 5.14: Average time cost of RHASFinder and protFFinder for one MSA. The greenish and blueish lines represent the feature extraction time, with shading indicating the number of taxa (from light to dark): 8, 16, 32, 64, 128, 256, and 512. The prediction time using CPU and GPU is indicated by black $-\times-$ and $-\square-$, respectively. Only the CPU is used for protFFinder prediction.

5.6 Exploration of With/Without +I Prediction

In addition to predicting the substitution model, RHAS model, and empirical frequencies, we have also made some attempts to classify alignments with or without invariant sites. Several features were tested on the validation set using Support Vector Machines (Cortes, 1995), Random Forest classifiers, and various neural networks. Among these, a neural network achieved the highest accuracy of 77.58%, comparable to ModelFinder, which reached 77.82%.

The relatively low accuracy of both methods may stem from the interaction between +I and RHAS models. In the case of +I, a proportion of the sites remain unchanged over time, whereas +Γ and +R account for varying rates of evolution across sites. When these models are applied simultaneously, sites with very low evolutionary rates might be misinterpreted as invariant (+I) when they are actually evolving slowly under the Γ or R models. Potential solutions include developing features that better distinguish between truly invariant sites and those evolving at very low rates, or integrating +I into RHAS model prediction to help the network learn to account for this interaction.

5.7 Challenges of Using Empirical MSAs

Although the true models of evolution for empirical alignments are unknown, it is still useful to evaluate the network's performance by comparing its prediction to the output of the state-of-the-art ModelFinder method, which can serve as the ground truth.

Performance of protFinder on real data

Tested on the empirical alignments of EvoNAPS, protFinder (trained on MSAs that were simulated from empirical distributions) achieves 71.23% overlap in predictions compared with ModelFinder. This result is reasonable, as simulated data generally simplifies the underlying evolutionary processes compared to real-world data. For example, real alignments typically arise from complex biological events such as gene duplications, transfers, and losses (DTL) (Szöllősi et al., 2015), recombination (Griffiths et al., 2004), incomplete lineage sorting (ILS) (Maddison, 1997), which are usually overlooked or neglected in simulation processes. Moreover, Ren et al. (2024) has shown that a single substitution model may be insufficient to capture the complexity of evolution in real data.

To improve the performance of protFinder, I also tried to re-train the model by (1) using simulated data that mimics the empirical data, and (2) directly using empirical data.

A new network trained on simulated data that mimics real alignments

Using AliSim's mimicking functionality (described in Section 4.1.2), 16 replicates were generated for each empirical alignment from EvoNAPS, resulting in 214,578 training data and 30,654 validation data. Then a new network was trained using this simulated data. It shows improved performance on real data, with a prediction overlap of 78.66% compared to ModelFinder.

A new network trained on real data

The 15,327 empirical MSAs were split into training, validation, and test sets in a $7 : 1.5 : 1.5$ ratio, maintaining the same proportion of substitution models across all sets. The network trained on this real-data set achieves a prediction overlap of 85.55% compared to ModelFinder. Notably, the total number of real alignments used to train this new network is only 4.4% of the 245,000 simulated alignments used to train protFinder. This result demonstrates that network and the features are indeed capable of capturing evolutionary information from real MSAs.

These above results underscore the significant gap between simulated and empirical data in phylogenetics. The performance of networks trained on simulated data when tested on real alignments indicates that simulated data, even when designed to mimic empirical characteristics, may not fully capture the complexity of real-world evolutionary processes. The poor performance on the other data type reflects not a limitation of this machine learning method, but rather the inherent differences between simulated and empirical data.

Chapter 6

Concluding Remarks

6.1 Conclusion

The machine learning framework proposed in this study addresses the selection of three key components of evolutionary models: amino acid substitution model, rate heterogeneity model, and the use of empirical frequencies. Overall, the framework achieves accuracy comparable to the state-of-the-art ModelFinder while offering a significant speed advantage, though it remains less robust for very short alignments.

This study not only incorporates the most complex data settings among existing machine learning-based model selection methods—by simulating both +F and various rate heterogeneity models—but also introduces the first machine learning models capable of distinguishing between with or without +F and different rate heterogeneity models.

The combination of machine learning and ModelFinder discussed in Section 5.5.1 demonstrates the practical application of this framework. Another approach is to pass the top-k substitution models or a 95% confidence set predicted by protFinder to ModelFinder, with these values customisable by the user. This method can significantly speed up ModelFinder while guarantee the accuracy on even short alignments.

Explorations using real MSAs highlight the discrepancies between simulated and real data in phylogenetics. Trost et al. (2024) achieved 99% accuracy in distinguishing empirical alignments from simulated ones using neural networks, underscoring the demand for methods that can simulate more realistic data.

6.2 Future Work

Considering the areas where the machine learning framework can be improved in comparison to ModelFinder, future enhancements may include:

- Improve With/Without +I Prediction: Although the network achieves accuracy comparable to ModelFinder, as discussed in Section 5.6, there is lots of space for improvement. The starting point will be integrating +I into RHAS model classification and designing networks that can capture the potential interactions between +I and +Γ and $+R_n$.
- Estimate Model Parameters: In addition to classification tasks, machine learning models can also be used to predict the parameters such as α for Γ, the invariant proportion for +I, frequencies for +F, and the more complex weights and rates for $+R_n$. By incorporating parameter estimation, the framework could perform independently, providing complete evolutionary models and much more efficiency over classic methods.
- Examine Resulting Trees: An important step is to compare the similarity between trees inferred from models that are predicted by the machine learning framework and those selected by ModelFinder. Since the ultimate goal of phylogenetic inference is to reconstruct trees, comparing them will provide insight into the practical effectiveness of the framework. If the differences are minimal, as has often been observed in nucleotide model selection, it would strongly enhance the utility of this machine learning approach.
- Train on Larger Datasets: Expand training to larger datasets that may contain more complex evolutionary patterns, which may improve network performance by addressing the data scarcity in the current EvoNAPS database. Additionally, explore different sequence length settings and investigate features that can better capture the variance and underlying information in very short alignments.

Appendix A

Appendix: Results of protFinder

This appendix presents the results of protFinder on MSAs with 3000, 4000 or 5000 sites, as well as the comparison between protFinder and ModelFinder on MSAs with 500 sites.

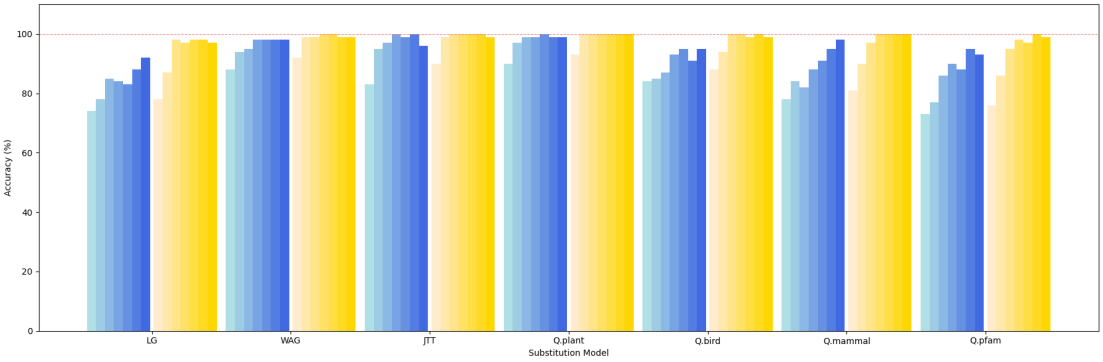


Figure A.1: Accuracy of protFinder (blueish) and ModelFinder (yellowish) on 500-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.

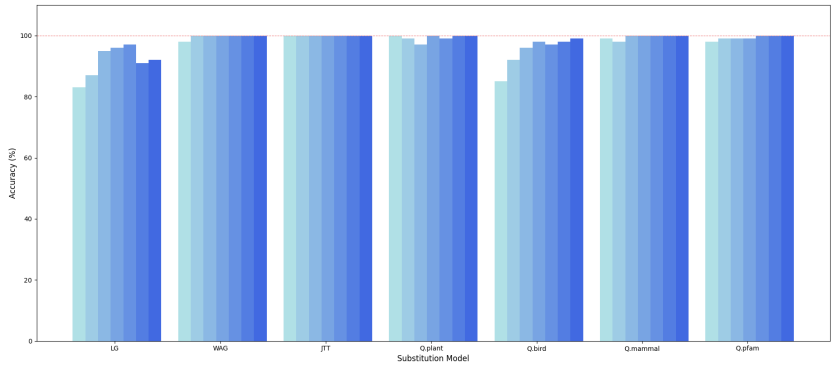


Figure A.2: Accuracy of protFinder on 3000-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.

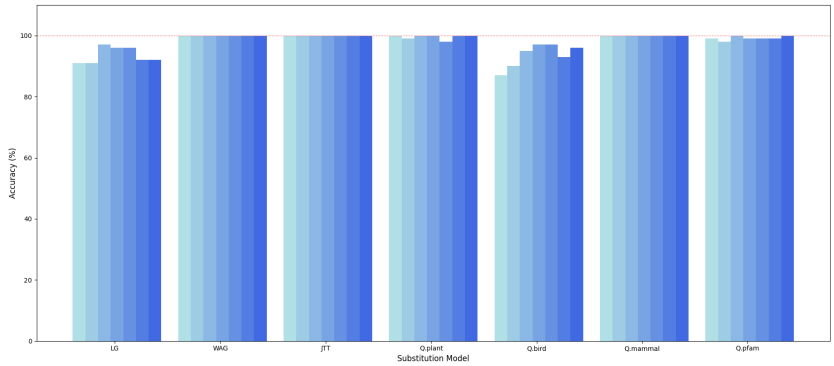


Figure A.3: Accuracy of protFinder on 4000-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.

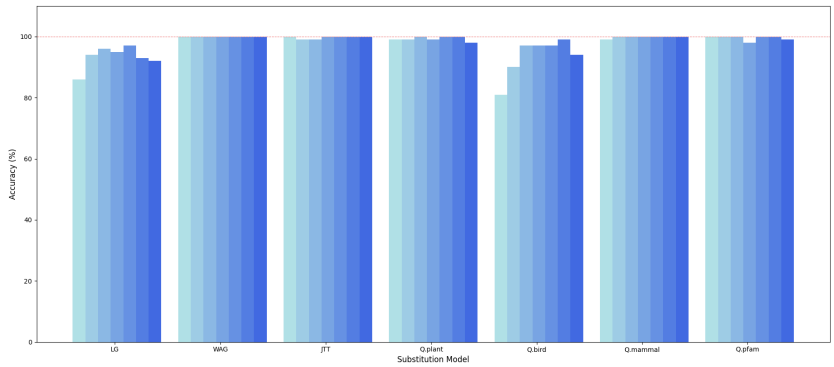


Figure A.4: Accuracy of protFinder on 5000-site MSAs. The shade of colors indicates number of taxa from left to right: 8, 16, 32, 64, 128, 256 and 512.

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. [Cited on pages 2 and 13.]

Abascal, F.; Zardoya, R.; and Posada, D., 2005. Prottest: selection of best-fit models of protein evolution. *Bioinformatics*, 21, 9 (2005), 2104–2105. [Cited on page 11.]

Agarap, A., 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, (2018). [Cited on page 22.]

Akaike, H., 1974. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19, 6 (1974), 716–723. [Cited on pages 1 and 6.]

An, K., 1933. Sulla determinazione empirica di una legge didistribuzione. *Giorn Dell'inst Ital Degli Att*, 4 (1933), 89–91. [Cited on page 19.]

Attwood, S. W.; Hill, S. C.; Aanensen, D. M.; Connor, T. R.; and Pybus, O. G., 2022. Phylogenetic and phylodynamic approaches to understanding and combating the early sars-cov-2 pandemic. *Nature Reviews Genetics*, 23, 9 (2022), 547–562. [Cited on page 1.]

Azouri, D.; Granit, O.; Alburquerque, M.; Mansour, Y.; Pupko, T.; and Mayrose, I., 2024. The tree reconstruction game: phylogenetic reconstruction using reinforcement learning. *Molecular Biology and Evolution*, 41, 6 (2024). [Cited on page 15.]

Breiman, L., 2001. Random forests. *Machine learning*, 45 (2001), 5–32. [Cited on page 13.]

Burgstaller-Muehlbacher, S.; Crotty, S. M.; Schmidt, H. A.; Reden, F.; Drucks, T.; and von Haeseler, A., 2023. Modelrevelator: Fast phylogenetic model estimation via deep learning. *Molecular Phylogenetics and Evolution*, 188 (2023), 107905. [Cited on pages 2 and 13.]

Cortes, C., 1995. Support-vector networks. *Machine Learning*, (1995). [Cited on page 42.]

Darriba, D.; Posada, D.; Kozlov, A. M.; Stamatakis, A.; Morel, B.; and Flouri, T., 2020. Modeltest-ng: a new and scalable tool for the selection of dna and protein evolutionary models. *Molecular biology and evolution*, 37, 1 (2020), 291–294. [Cited on page 12.]

Darriba, D.; Taboada, G. L.; Doallo, R.; and Posada, D., 2011. Prottest-hpc: fast selection of best-fit models of protein evolution. In *Euro-Par 2010 Parallel Processing Workshops: HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31–September 3, 2010, Revised Selected Papers 16*, 177–184. Springer. [Cited on page 11.]

Darriba, D.; Taboada, G. L.; Doallo, R.; and Posada, D., 2012. jmodeltest 2: more models, new heuristics and high-performance computing. *Nature methods*, 9, 8 (2012), 772. [Cited on page 11.]

Dosovitskiy, A., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, (2020). [Cited on page 22.]

Edgar, R. C., 2022. Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. *Nature Communications*, 13, 1 (2022), 6968. [Cited on page 3.]

Farris, J. S., 1970. Methods for computing wagner trees. *Systematic Biology*, 19, 1 (1970), 83–92. [Cited on page 6.]

Fitch, W. M., 1971. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20, 4 (1971), 406–416. [Cited on page 6.]

Goodfellow, I.; Bengio, Y.; and Courville, A., 2016a. *Deep Learning*. MIT Press. http://www.deeplearningbook.org. [Cited on page 7.]

Goodfellow, I.; Bengio, Y.; and Courville, A., 2016b. *Deep Learning*. MIT Press. http://www.deeplearningbook.org. [Cited on pages 8 and 9.]

Griffiths, A. J. F.; Wessler, S. R.; Lewontin, R. C.; Gelbart, W. M.; Suzuki, D. T.; and Miller, J. H., 2004. *Introduction to Genetic Analysis and Testbank*. Freeman Company, W. H. ISBN 9780716786634. https://openlibrary.org/books/OL37936589M/Introduction_to_Genetic_Analysis_and_Testbank. [Cited on page 43.]

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. [Cited on page 22.]

Hochreiter, S., 1997. Long short-term memory. *Neural Computation MIT-Press*, (1997). [Cited on page 22.]

Hu, J.; Shen, L.; and Sun, G., 2017. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507 (2017). http://arxiv.org/abs/1709.01507. [Cited on page 22.]

Hurvich, C. M. and Tsai, C.-L., 1989. Regression and time series model selection in small samples. *Biometrika*, 76, 2 (1989), 297–307. [Cited on page 7.]

Ioffe, S., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, (2015). [Cited on page 10.]

Jones, D. T.; Taylor, W. R.; and Thornton, J. M., 1992. The rapid generation of mutation data matrices from protein sequences. *Bioinformatics*, 8, 3 (1992), 275–282. [Cited on page 17.]

Jukes, T. H.; Cantor, C. R.; et al., 1969. Evolution of protein molecules. *Mammalian protein metabolism*, 3, 24 (1969), 21–132. [Cited on page 5.]

Kalyaanamoorthy, S.; Minh, B. Q.; Wong, T. K.; Von Haeseler, A.; and Jermiin, L. S., 2017. Modelfinder: fast model selection for accurate phylogenetic estimates. *Nature methods*, 14, 6 (2017), 587–589. [Cited on pages 2 and 12.]

Katoh, K. and Standley, D. M., 2013. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30, 4 (2013), 772–780. [Cited on page 3.]

Kelley, H. J., 1960. Gradient theory of optimal flight paths. *Ars Journal*, 30, 10 (1960), 947–954. [Cited on page 9.]

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, 21 (2019), 4453–4455. [Cited on page 7.]

Kuhner, M. K. and Yamato, J., 2015. Practical performance of tree comparison metrics. *Systematic biology*, 64, 2 (2015), 205–214. [Cited on page 13.]

Kulikov, N.; Derakhshandeh, F.; and Mayer, C., 2024. Machine learning can be as good as maximum likelihood when reconstructing phylogenetic trees and determining the best evolutionary model on four taxon alignments. *Molecular Phylogenetics and Evolution*, 200 (2024), 108181. [Cited on pages 2 and 14.]

Kullback, S. and Leibler, R. A., 1951. On information and sufficiency. *The annals of mathematical statistics*, 22, 1 (1951), 79–86. [Cited on page 21.]

Lanfear, R.; Calcott, B.; Ho, S. Y.; and Guindon, S., 2012. Partitionfinder: combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Molecular biology and evolution*, 29, 6 (2012), 1695–1701. [Cited on page 12.]

Le, S. Q. and Gascuel, O., 2008. An improved general amino acid replacement matrix. *Molecular biology and evolution*, 25, 7 (2008), 1307–1320. [Cited on page 5.]

Le, S. Q.; Lartillot, N.; and Gascuel, O., 2008. Phylogenetic mixture models for proteins. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363, 1512 (2008), 3965–3976. [Cited on page 12.]

LeCun, Y.; Bengio, Y.; and Hinton, G., 2015. Deep learning. *nature*, 521, 7553 (2015), 436–444. [Cited on page 9.]

Lefort, V.; Longueville, J.-E.; and Gascuel, O., 2017. Sms: smart model selection in phyml. *Molecular biology and evolution*, 34, 9 (2017), 2422–2424. [Cited on page 12.]

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. [Cited on page 15.]

Leuchtenberger, A. F. and von Haeseler, A., 2024. Learning from an artificial neural network in phylogenetics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, (2024). [Cited on page 15.]

Li, C.; Lu, G.; and Ortí, G., 2008. Optimal data partitioning and a test case for ray-finned fishes (actinopterygii) based on ten nuclear loci. *Systematic biology*, 57, 4 (2008), 519–539. [Cited on page 12.]

Loshchilov, I., 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, (2017). [Cited on page 24.]

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), msac092. [Cited on page 18.]

Maddison, W. P., 1997. Gene trees in species trees. *Systematic biology*, 46, 3 (1997), 523–536. [Cited on page 43.]

Minh, B. Q.; Dang, C. C.; Vinh, L. S.; and Lanfear, R., 2021. Qmaker: fast and accurate method to estimate empirical models of protein evolution. *Systematic Biology*, 70, 5 (2021), 1046–1060. [Cited on page 17.]

Minh, B. Q.; Schmidt, H. A.; Chernomor, O.; Schrempf, D.; Woodhams, M. D.; Von Haeseler, A.; and Lanfear, R., 2020. Iq-tree 2: new models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution*, 37, 5 (2020), 1530–1534. [Cited on page 7.]

Nesterenko, L.; Boussau, B.; and Jacob, L., 2022. Phyloformer: towards fast and accurate phylogeny estimation with self-attention networks. *bioRxiv*, (2022), 2022–06. [Cited on page 15.]

Nguyen Huy, T. and Vinh, L. S., 2024. An efficient deep learning method for amino acid substitution model selection. *bioRxiv*, (2024), 2024–06. [Cited on pages 2 and 14.]

Nylander, J. A.; Ronquist, F.; Huelsenbeck, J. P.; and Nieves-Aldrey, J., 2004. Bayesian phylogenetic analysis of combined data. *Systematic biology*, 53, 1 (2004), 47–67. [Cited on page 12.]

Pearson, K., 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2, 11 (1901), 559–572. [Cited on page 29.]

Posada, D. and Buckley, T. R., 2004. Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Systematic biology*, 53, 5 (2004), 793–808. [Cited on page 11.]

Posada, D. and Crandall, K. A., 1998. MODELTEST: testing the model of DNA substitution. *Bioinform.*, 14, 9 (1998), 817–818. doi:10.1093/BIOINFORMATICS/14.9.817. https://doi.org/10.1093/bioinformatics/14.9.817. [Cited on page 11.]

Posada, D. and Crandall, K. A., 2001. Selecting the best-fit model of nucleotide substitution. *Systematic biology*, 50, 4 (2001), 580–601. [Cited on page 6.]

Pupko, T.; Bell, R. E.; Mayrose, I.; Glaser, F.; and Ben-Tal, N., 2002. Rate4site: an algorithmic tool for the identification of functional regions in proteins by surface mapping of evolutionary determinants within their homologues. *Bioinformatics*, 18, suppl_1 (2002), S71–S77. [Cited on page 13.]

Quinlan, J. R., 1986. Induction of decision trees. *Machine learning*, 1 (1986), 81–106. [Cited on page 7.]

Ren, H.; Wong, T. K.; Minh, B. Q.; and Lanfear, R., 2024. Mixturefinder: Estimating dna mixture models for phylogenetic analyses. *bioRxiv*, (2024), 2024–03. [Cited on pages 12, 20, and 43.]

Saitou, N. and Nei, M., 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4, 4 (1987), 406–425. [Cited on page 6.]

Schwarz, G., 1978. Estimating the dimension of a model. *The annals of statistics*, (1978), 461–464. [Cited on pages 1 and 7.]

Shannon, C. E., 1948. A mathematical theory of communication. *The Bell system technical journal*, 27, 3 (1948), 379–423. [Cited on page 21.]

Silvestro, D.; Latrille, T.; and Salamin, N., 2024. Toward a semi-supervised learning approach to phylogenetic estimation. *Systematic Biology*, (2024), syae029. [Cited on page 15.]

Soubrier, J.; Steel, M.; Lee, M. S.; Der Sarkissian, C.; Guindon, S.; Ho, S. Y.; and Cooper, A., 2012. The influence of rate heterogeneity among sites on the time dependence of molecular rates. *Molecular biology and evolution*, 29, 11 (2012), 3345–3358. [Cited on page 5.]

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15, 1 (2014), 1929–1958. [Cited on page 10.]

Strimmer, K. and Von Haeseler, A., 1996. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Molecular biology and evolution*, 13, 7 (1996), 964–969. [Cited on page 15.]

Sugiura, N., 1978. Further analysis of the data by akaike's information criterion and the finite corrections: Further analysis of the data by akaike's. *Communications in Statistics-theory and Methods*, 7, 1 (1978), 13–26. [Cited on page 7.]

Susko, E. and Roger, A. J., 2020. On the use of information criteria for model selection in phylogenetics. *Molecular biology and evolution*, 37, 2 (2020), 549–562. [Cited on page 1.]

Suvorov, A.; Hochuli, J.; and Schrider, D. R., 2020. Accurate inference of tree topologies from multiple sequence alignments using deep learning. *Systematic biology*, 69, 2 (2020), 221–233. [Cited on page 15.]

Suvorov, A. and Schrider, D. R., 2022. Reliable estimation of tree branch lengths using deep neural networks. *bioRxiv*, (2022), 2022–11. [Cited on page 15.]

Szöllősi, G. J.; Tannier, E.; Daubin, V.; and Boussau, B., 2015. The inference of gene trees with species trees. *Systematic biology*, 64, 1 (2015), e42–e62. [Cited on page 43.]

Trost, J.; Haag, J.; Höhler, D.; Jacob, L.; Stamatakis, A.; and Boussau, B., 2024. Simulations of sequence evolution: how (un) realistic they are and why. *Molecular biology and evolution*, 41, 1 (2024), msad277. [Cited on page 45.]

Udny Yule, G., 1925. A mathematical theory of evolution, based on the conclusions of dr. jc willis, frs. *Philosophical Transactions of the Royal Society of London Series B*, 213 (1925), 21–87. [Cited on page 19.]

Vaswani, A., 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, (2017). [Cited on page 15.]

Vuong, Q. H., 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: journal of the Econometric Society*, (1989), 307–333. [Cited on page 6.]

Whelan, S. and Goldman, N., 1999. Distributions of statistics used for the comparison of models of sequence evolution in phylogenetics. *Molecular Biology and Evolution*, 16, 9 (1999), 1292–1299. [Cited on page 11.]

Whelan, S. and Goldman, N., 2001. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular biology and evolution*, 18, 5 (2001), 691–699. [Cited on page 5.]

Yang, Z., 2014. *Molecular evolution: a statistical approach*. Oxford University Press. [Cited on page 5.]

Yousaf, A.; Liu, J.; Ye, S.; and Chen, H., 2021. Current progress in evolutionary comparative genomics of great apes. *Frontiers in Genetics*, 12 (2021), 657468. [Cited on page 4.]

Zaharias, P.; Grosshauser, M.; and Warnow, T., 2022. Re-evaluating deep neural networks for phylogeny estimation: the issue of taxon sampling. *Journal of Computational Biology*, 29, 1 (2022), 74–89. [Cited on page 15.]

Zhao, N.; Zhou, N.; Fan, H.; Ding, J.; Xu, X.; Dong, X.; Dong, X.; Xu, D.; Min, X.; Yu, Y.; et al., 2022. Mutations and phylogenetic analyses of sars-cov-2 among imported covid-19 from abroad in nanjing, china. *Frontiers in Microbiology*, 13 (2022), 851323. [Cited on page 1.]

Zou, Z.; Zhang, H.; Guan, Y.; and Zhang, J., 2020. Deep residual neural networks resolve quartet molecular phylogenies. *Molecular biology and evolution*, 37, 5 (2020), 1495–1507. [Cited on page 15.]