Machine learning to infer the process of coevolution under the light of evolution

Rama Ballesteros Rocío

UNIL | Université de Lausanne

Faculté de biologie
et de médecine

Département de Biologie Computationnelle

Machine learning to infer the process of coevolution under the light of evolution

Thèse de doctorat ès sciences de la vie (PhD)

présentée à la

Faculté de biologie et de médecine
de l’Université de Lausanne

par

Rocío RAMA BALLESTEROS

Master en Ingeniería Informática, Université de Granada

Jury

Lausanne
(2022)

UNIL | Université de Lausanne
Faculté de biologie et de médecine

Ecole Doctorale
Doctorat ès sciences de la vie

Imprimatur

Vu le rapport présenté par le jury d'examen, composé de

Président·e	Monsieur	Prof.	Jan Roelof	**Van der Meer**
Directeur·trice de thèse	Monsieur	Prof.	Nicolas	**Salamin**
Expert·e·s	Monsieur	Prof.	Giovanni	**Ciriello**
	Monsieur	Prof.	Daniele	**Silvestro**

le Conseil de Faculté autorise l'impression de la thèse de

Rocio Rama Ballesteros

Master in computational science and engineering, University of Granada, Espagne

intitulée

Machine learning to infer the process of coevolution under the light of evolution

Lausanne, le 11 novembre 2022

pour le Doyen
de la Faculté de biologie et de médecine

[illegible]

Prof. Jan Roelof Van der Meer

Table of Contents

ABSTRACT

Coevolution is an important component of the evolutionary biology and describes the reciprocal changes that occur between biological entities as they depend on each other. It is one of the mechanisms driving biodiversity when interactions occur between organisms, and, at the molecular level, it can reveal information about the function and structure of a protein. These coordinated changes between sites along sequences tend to occur to improve or maintain functional and structural interactions, or because of evolutionary processes like compensatory mutations and epistasis. Because of the high throughput sequencing revolution, it is now possible to examine the available genomic databases - encompassing thousands of proteins - to detect coevolution and improve the insights of the genomic data. Nevertheless, current methods inferring coevolution have some limitations and they require large computing time to analyze the data. In my thesis, I used the power of machine learning techniques to infer coevolution in large databases in an easy and fast way. First, I investigated the limitations of the current methods inferring coevolution by testing the effect of the level of divergence on their performance to detect pairs of coevolving sites, comparing their key properties and downsides. Secondly, I developed a machine learning model based on Convolutional Neural Networks (CNN) to detect the signature of coevolution while accounting for the evolutionary history of the sequences. Finally, I provided a user-friendly pipeline to run the model and infer coevolution in any given alignment with its phylogenetic tree. I simulated genomic data based on a large genomic database to train the CNN. I used the model trained on a subset of the bony vertebrate Selectome dataset to detect signature of coevolution in 217 proteins.

Overall, my work provides a novel approach based on machine learning techniques to better detect and understand the signature of coevolution, opening the door to investigate other compelling machine learning approaches to take advantage of the large genomic data that is becoming available nowadays.

RÉSUMÉ

La coévolution est une composante importante de la théorie de l'évolution et décrit les changements réciproques qui se produisent entre les entités biologiques lorsqu'elles dépendent les unes des autres. C'est l'un des mécanismes moteur de la biodiversité lorsque des organismes interagissent entre eux. A plus fine échelle, au niveau moléculaire, la coévolution peut révéler des informations sur la fonction et la structure des protéines. Ainsi, des changements entre différents sites d'une séquence qui ont tendance à se produire de manière coordonnée sont la trace de coévolution, permettant d'améliorer ou maintenir les interactions fonctionnelles et structurelles. Grâce à la révolution du séquençage à haut débit, il est désormais possible d'exploiter les bases de données génomiques disponibles, couvrant des milliers de gènes et de protéines afin de détecter les signatures de la coévolution. Cependant, les méthodes actuelles qui infèrent la coévolution ont certaines limites et prennent trop de temps pour analyser les données. Par conséquent, dans cette thèse, j'exploite la puissance des techniques d'apprentissage automatique utilisant des réseaux de neurones convolutifs (CNN) pour inférer la coévolution de manière simple et rapide sur des bases de données génomiques à large échelle. Dans un premier temps, j'ai étudié les limites des méthodes d'inférence de coévolution actuelles (plmDCA, CoMap et Coev) en fonction du niveau de divergence, en comparant leurs avantages et leurs inconvénients respectifs. Deuxièmement, j'ai développé un modèle d'apprentissage automatique basé sur des CNN pour détecter la signature de la coévolution au sein d'une protéine en exploitant le signal laissé par l'évolution sur des sites qui ont des changements coordonnés. Enfin, j'ai développé un pipeline facile à utiliser pour exécuter le modèle et détecter de la coévolution à partir d'un alignement donné et de son arbre phylogénétique. J'ai simulé des données génomiques basées sur l'ensemble de données bony vertebrate Selectome et je les ai utilisées pour entraîner le CNN. Après avoir entraîné le modèle, je l'ai testé sur le jeu de données réel bony vertebrate Selectome, détectant 217 protéines sous signature d'une coévolution.

Dans l'ensemble, mon travail fournit une approche puissante basée sur des techniques d'apprentissage automatique pour mieux détecter et comprendre la signature de la coévolution, ouvrant la porte à d'autres approches d'apprentissage automatique utilisables sur des données génomiques afin de répondre à des questions de biologie évolutive.

ACKNOWLEDGEMENTS

Thank you to everyone for all the support that I have received, for always having faith on me.

I am specially very thankful to Nicolas. I don’t have enough words to thank you for always supporting me and trusting my work. Thank you for always bringing up the best part of me.

Thank you to all my colleagues, you have been day by day supporting me in this process of doing research. From the beginning, thank you Anna, thank you for being you, you have taught me so many things, that I could never be grateful enough. You are a role model to follow! Oriane, so many afternoons in the office, thank you for opening my eyes to the biology world and for making me be a bit more like a biologist. Xavier, Linda, Pablo, Alberto, Lucy, Sara, Sarah, Kana, Emilie, Theo, Diego and Thibault. THANK YOU for always listening to me, for always having time for a coffee break, for helping me out every time I needed it, thank you all of you.

My Spanish friends in Geneva, you became my family here. Barbacuqui, you are incredible: Miguel, Eddie, Silvia, Marpe, David, Luisa, and specially Estrella and Dani, thank you for everything. You have always been by my side, and thank you for making the Covid period easier to live through.

My friends from Spain: Elena, Paqui, Marisa, Ana, Fran, Ruben, Vane, and Juan, thank you for listening to my desperate audios through Whatsapp.

And of course, my family, I couldn’t have made it without your support. My parents, always there, supporting me to finish my PhD dream, listening to me when I cried, for all our talks about my thesis project. My sister, my role model to follow, thank you for all your advice. And Íñigo, thank you. You have been the best partner I could ask for. Thank you for our thousand talks about my project, for your understanding, for your interminable patient and for your support.

I feel very lucky of having been a part of this lab because I was able to meet such smart and amazing people here. It doesn’t matter what you do in your life if you are not around the people that you love.

*“Nothing in biology makes sense
except in the light of evolution.”*
Theodosius Dobzhansky

GENERAL INTRODUCTION

One hundred and sixty years ago, Charles Darwin (Darwin, 1859) proposed the theory of evolution through natural selection, and although it is now a central paradigm for biological research (Koonin & Wolf, 2012; Davydov et al., 2015) the exact mechanisms surrounding the core of the theory as well as the relationship between organisms are still at the focus of intensive research worldwide.

Organisms are related to each other within what biologists call the Tree of Life (Hug et al., 2016), a metaphor to express the evolutionary relationships existing between living beings. Each entity of these living beings are species that originated from the same common ancestor. Although species that originate from the same common ancestor are often represented as independent evolutionary entities (Fontaneto et al., 2007), it is known that interactions between species do occur through evolution (Poisot et al., 2015). Coevolution, the evolutionary process subtending these interactions, is an essential component of evolutionary theory and describes the reciprocal evolutionary changes that occur between pairs of organisms or biomolecules as they depend on one another (Yip et al., 2008).

The concept of coevolution was briefly described at the species level for the first time by Charles Darwin in The Origin of Species (Darwin, 1859). He demonstrated this idea in detail in Fertilization of Orchids (Darwin, 1877) when studying the process of pollination in orchids where he observed that the length of the pollinator's proboscis was related to the size of the orchid's corolla (Fig-1). This suggests that the evolution of orchid flower traits is strongly linked with the evolution of the pollinator's proboscis and that they are adapted to one another. Coevolution has been highly studied since then (Ehrlich & Raven, 1964), particularly in biological systems involving host-parasite interactions, where the parasites exploit the host to grow, reproduce and spread while providing benefits to the host to evolve (for example in the case of *Drosophila* and the fungal toxins (Trienens & Rohlfs, 2011)). These interactions between parasites and their host species induce high selective constraints, meaning that a change happening in one species will quickly trigger the change in the other species, allowing the relationship to be maintained and species to survive (Woolhouse et al., 2002; Papkou et al., 2016).

Fig-1 Source: Smithsonian Gardens[1]. In the left: The orchid *Angraecum sesquipedale var. angustifolium.* In the middle: an illustration of a hawk moth interacting with the orchid by Emily Damstra. In the left: a moth with extended probiscis © kqedquest.

Coevolution is one of many mechanisms driving the biodiversity of living organisms (Laine 2009). However, this process does not only apply to interactions between entire organisms but can also explain, at the cellular level, how biomolecules interact with each other. At the molecular level, coevolution between parts of a molecule can reveal important information about the function and structure of a protein (Carmona et al., 2015). As a result, at the molecular scale, DNA (Deoxyribonucleic acid) sequences are of the utmost importance to studying signals of coevolution.

Coevolution at the molecular level

DNA is a polymer consisting of two-nucleotide chains that carry the genetic information to develop, live and reproduce living organisms and viruses (Tropp, 2012). Every living organism contains thousands of genes in its genome: long sequences of nucleotides that are transcribed into sequences of amino acids to create proteins. Each protein has a structure which allows it to interact with other proteins or other molecules enabling its function within an organism. Through time sequences may change and coordinated changes tend to occur between pairs of sites within a sequence to improve or to maintain the function or structure of proteins (Fares & Travers, 2006). Due to the conserved function or structure of a protein, mutations do not happen randomly and are an indication of the evolutionary constraints acting on a protein. Coevolution at the molecular level can be observed along a sequence when the modification of one site (nucleotide when referring to genes or amino acids when referring to proteins) will trigger the modification of another site (Yip et al., 2008).

[1] https://smithsoniangardens.wordpress.com/2015/12/18/on-display-highlights-from-the-smithsonian-gardens-orchid-collection/

Coevolution can occur between a pair of sites within a molecule. For example, if two sites are in contact in the 3-dimensional structure of the protein, structural constraints will occur between these pairs to maintain the correct folding of the protein. This implies that any mutation perturbing the 3D structure will be limited and if a mutation does occur, it will need to be compensated to recreate the correct 3D structure (Ivankov et al., 2014). This compensation is another coevolutionary process. Additionally, if the coevolving pairs are in an active part of the protein, which is involved in the expression of a specific function, they will also be under a functional constraint, and any change will need to be compensated to maintain the function of the protein, which will lead to another level of coevolution. Likewise, coevolutionary events occurring within a single protein may also be observed between pairs of proteins interacting with one another. For instance, in allosteric communication, a change of an essential site will affect the protein-protein interaction, and the other protein will be driven to adapt to maintain efficient communication (Baussand & Carbone, 2009). However, regardless of the underlying mechanisms involved, the initial task to understand the role played by coevolution in shaping protein sequences is to be able to detect which signals to extract from the genomic or protein data to correctly infer coevolving pairs of sites.

There are several methods available in the literature to detect coevolution between pairs of sites at the molecular level. The most popular methods rely on the idea that the structural constraints imposed by the protein 3D structure lead to the covariance between the patterns of amino acid frequencies found in the columns of the multiple sequence alignments (MSA) (Chiu & Kolodziejczak, 1991; Weigt et al., 2009; Jones et al., 2012). Therefore, these methods take into account MSAs of homologous DNA or amino acid sequences and look at the covariation of the frequency patterns of amino acids or nucleotides found in any two columns of the MSA.

Structural methods like mutual information (MI) estimate the distance between the two probability distributions of amino acids at two different sites in the MSA (Chiu & Kolodziejczak, 1991; Burger & Van Nimwegen, 2010). Direct coupling analysis (DCA) and protein sparse inverse co-variance (PSICOV) have improved over MI and removed indirect associations between sites by inverting a site-site covariance matrix (Weigt et al., 2009; Jones et al., 2012). Due to the computational demands of these methods, methods are improving their performance by making use of pseudo-likelihood estimation, like plmDCA (Ekeberg et al., 2014) or the very similar Gremlin (Kamisetty et al., 2013). Furthermore, other methods such as metaPSICOV (Jones et al., 2015), start to take advantage of machine learning techniques by combining the coevolving pairs resulting from other methods to obtain more accurate predictions of true coevolution pairs of sites.

Detecting covariance between sites with the methods outlined above has been used to help predict protein structure (Kamisetty et al., 2013; Hopf et al., 2015; Zerihun & Schug, 2018; Senior et al., 2019). However, sequences do not evolve independently, there are evolutionary relationships between them which can lead to covariances between sites. This can create patterns in the MSA that are similar to structural constraints (Qin & Colwell, 2018) and bias the prediction of coevolution (Horta1 & Weigt, 2021). Therefore, several approaches have been attempted to correct this effect of phylogenetic correlations and thus improve the inference of coevolving sites (Qin & Colwell, 2018). Some of these methods re-weight sequences according to their similarities measured by Hamming distances (Marks et al., 2011), through row-column weighting (Gouveia-Oliveira & Pedersen, 2007), or by correcting the coevolving sites score with Average Product correction (APC) (Dunn et al., 2008). Although these methods have improved the inference of coevolving sites, they only remove or correct the inherent evolutionary signal present in the sequences. The evolutionary history that affects the similarities between the sequences (Caporaso et al., 2008; Dib et al., 2014; Dutheil, 2012; Marmier et al., 2019; Yeang & Haussler, 2007) can also reflect the interplay of mechanisms that lead to coevolution between sites. As a result, several methods taking into account phylogeny have been proposed in recent years.

In the literature, several authors have proposed statistical and combinatorial methods based on phylogenetic profiles (Dutheil et al., 2005; Baussand & Carbone, 2009; Dib et al., 2014) to infer coevolving sites. While the methods described above see the phylogeny as a noisy signal that may be removed, these methods rely on the idea that coevolving pairs of sites evolve in a co-dependent manner, and that there is an evolutionary pressure forcing coevolving changes to occur on the same branches of a phylogenetic tree. One way to incorporate phylogeny is based on reconstructing the ancestral state of each site to map the changes events onto the phylogenetic tree (CoMap, Dutheil et al., 2005). Another method, Coev, uses a Markov model to study the process of evolution along a phylogenetic tree for any pair of sites based on a substitution matrix describing the transitions between positions, along the branches (Dib et al., 2014; Meyer et al., 2019; Yamada et al., 2019). These methods detect that two sites are under coevolution when they change in a coordinated way during their evolution.

Taking into account the phylogenetic tree when inferring coevolution, reveals essential information such as selective pressure or evolutionary constraints acting on a pair of sites, which can explain how and why they are coevolving. Looking at these coordinated changes between a pair of sites through their phylogeny, it is possible to decipher if it is a random change or if evolutionary constraints or selective pressure are forcing this change to occur.

Moreover, it is possible to estimate the site-specific substitution rate which improves the prediction of coevolving pairs of sites (Dutheil, 2012). Furthermore, studies are showing that high divergence is needed to better estimate the sites under coevolution (Dutheil, 2012). This effect is because the low divergence in the phylogenetic tree will induce a smaller number of mutations and will possibly reduce the amount of information contained in the MSA. Besides, with low divergence, a random change at two different sites can be over-represented in the MSA if it happens near the root of the tree, as is shown in (Dutheil, 2012). While structural methods try to overcome this problem by having large numbers of sequences in the alignment, it is not always possible and it will depend on the protein family under study, e.g. in (Dib et al., 2018). However, one of the drawbacks of the methods using the phylogenetic tree to infer coevolution, is that they usually require more computing time than the structural methods (Meyer, Dib, & Salamin, 2019).

Machine learning

The exponential growth of sequenced genomes facilitates the determination of which proteins coevolve, leading to the investigation of patterns explaining the evolutionary process of coevolution. The methods currently available are often limited by the number of sequences required to increase prediction accuracy and by the length of time needed to run the analysis. The availability of databases with thousands of proteins (Moretti et al., 2014; Mistry et al., 2021), gives us the possibility to have a tool to analyze and study coevolution in a scalable and efficient way.

The emergence of high-throughput genomics has exploited artificial intelligence techniques such as machine learning algorithms (Pezoulas et al., 2021), which have proven to be a powerful tool in this era of big data analysis in many fields such as in physics, by calculating all the variables needed to describe physical actions (swing of a double pendulum or the flicker of a flame) by analyzing videos (Z.-K. Liu et al., 2022); in language recognition, by allowing the communication between signers and non-signers through a smart glove in a VR space (Wen et al., 2021); and in medicine, by detecting tongue cancer using endoscopic images (Heo et al., 2022). Machine learning algorithms create a mathematical model based on a known dataset that is used to train a model to make predictions about the data without being specifically programmed for it (Bishop & Nasrabadi, 2006). These methods are helpful when the dataset to analyze is too large and/or complex for standard methods. There are mainly four types of machine learning: supervised, unsupervised, semi-supervised learning and reinforcement learning (Fig-2). Supervised machine learning methods train a model to fit a labeled dataset, where the true label of each dataset is known (for example: Decision Trees, Linear Regression, Neural Networks). In contrast, unsupervised machine learning methods are used when the data is unlabeled and the model learns to group the data based on detected similar patterns (for example: K-means clustering). Semi-supervised machine learning methods combine these two approaches, and the model is free to explore the patterns in the data. In reinforcement learning, the model takes a sequence of decisions to achieve a goal (such as finding the optimal path to exit a maze (Yu et al., 2019), and it will learn how to succeed based on reward or punishment depending on the action taken (for example: Q-Learning, Deep Adversarial Networks).

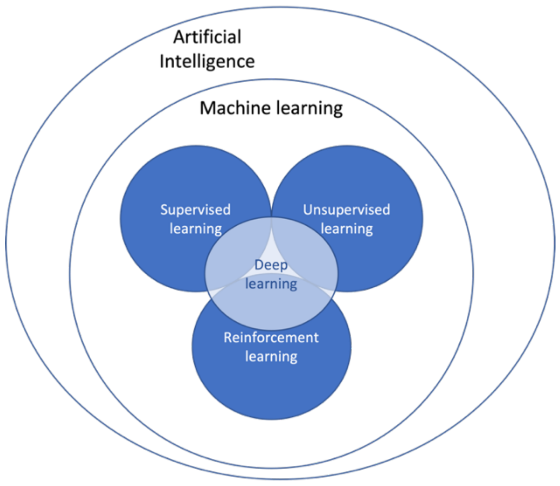


Fig-2 Diagram artificial intelligence sub-fields

Deep Learning is a subset of machine learning based on multiple layers of Artificial Neural Networks. One of its popular algorithms is Recurrent Neural Networks (RNNs), a type of Artificial Neural Network that contains feedback loops allowing it to store information within the network (Goodfellow et al., 2016). Its architecture can be seen as a graph with directed cycles in memory, allowing it to learn information related to the past. RNNs are trained to recognize the sequential characteristics of data and they are very effective for speech processing, Natural Language Processing (Lavanya & Sasikala, 2021), and time series prediction (Che et al., 2018). However, one of the most popular Deep Learning methods is Convolutional Neural Networks (CNNs), which are also a type of Artificial Neural Network where the "neurons" correspond to receptive fields very similar to the real neurons of the visual cortex of a biological brain (Goodfellow et al., 2016). CNNs are trained with bidimensional matrices (usually an image), and they are very effective in artificial vision problems. They typically have three main layers: a convolutional layer in which features/patterns are extracted from input data, a pooling layer to reduce the size of the data while preserving the critical features, and a fully connected layer where all the outputs from the previous layer are connected, and an output/prediction is provided. CNNs have proven to be a powerful methodology for detecting patterns and classifying them accordingly. Even though it started as a method to classify images (Lecun et al., 1998; Krizhevsky et al., 2012), it has been expanding its applications to other non-image problems (Abdel-Hamid et al., 2014; Zhang & LeCun, 2015). CNNs are however always used in situations where the data is translation invariant, the data has spatial features and it is possible for the network to ignore positional shifts or the translation of the target, and its detection is key for the prediction/classification of the data (Goodfellow et al., 2016). Nevertheless, neural networks in general, have always been considered a "black box": it is not possible to understand the parameters learned by the network. Nonetheless, studies are trying to better understand the most important parts of the network by applying gradients to explain the decision/output. Some of them are guided back-propagation (Springenberg et al., 2015.), deconvolutional networks (Zeiler et al., 2010), grad-CAM (Selvaraju et al., 2017) or guided grad-CAM (Selvaraju et al., 2017). Contributions of these studies better facilitate the visualization of feature layers and the better understanding of the classification decision (Zeiler & Fergus, 2014; Sheu, 2020).

Deep learning methods have thus proven their potential in biology and are accelerating the research and innovation in many areas: estimating species extinction risk (Zizka et al., 2022), identifying diseases (J. Liu et al., 2021), classifying metagenomes (Manning et al., 2019), inferring hybridization between organisms (Blischak et al., 2021), or the incredible advance predicting the structure of proteins with AlphaFold (Jumper et al., 2021). Nowadays, predicting the structure of proteins is a hot topic, where the number of new methods using deep learning to infer pairs of amino-acid sites in contact has increased. For example, the use of the popular CNN (RaptorX, DeepMetaPSICOV, Filter-DCA; Källberg et al., 2012; Kandathil et al., 2019; Muscat et al., 2020). Nevertheless, all of them take as input-features some measures of coevolution between sites, without taking into account the phylogeny, and these sites may not be close in the 3D structure, biasing their 3D structure prediction (Mehari B. Zerihun & Schug, 2017). However, in molecular evolution, there have not been many advances in the use of these powerful methods. CNNs have been used to build phylogenetic trees by inferring quartets from sequence data (Zou et al., 2020), but there has been no attempt to use the pattern recognition of CNN to learn about the processes driving molecular evolution.

The objectives of this thesis

On a broad scale, the main goal of my thesis was to propose meaningful ways to sort and interpret biologically the signature of coevolution based on machine learning techniques while taking into account the evolutionary constraints. After considering the limitations of the current methods available to infer coevolution, I simulated a dataset variable enough to train my machine learning model. First, I transformed the genomic data into a 2D matrix. Then, by using Convolutional Neural Networks, I was able to detect the signature pattern of coevolution in a controlled simulated dataset.

In **Chapter 1** I aimed to understand the differences between methods inferring coevolution and how divergence affects their predictions. By simulating datasets (amino acid MSAs and their phylogenetic trees) under different levels of divergence, I benchmarked the following methods: plmDCA, CoMap and Coev. I found that plmDCA is outstanding compared to CoMap and Coev.

In **Chapter 2**, I developed a new method to infer coevolution, taking into account the phylogeny, based on Convolutional Neural Networks (CNNs). First, I studied how to transform the information of MSAs and their phylogenetic tree into a type of data suitable and useful to learn the signature pattern of coevolution. Second, I implemented a new method based on convolutional neural networks to predict coevolution considering the phylogeny. Finally, I simulated several datasets varying the divergence and the number of species to test our method. I achieved 90% of accuracy when applying the CNN to infer coevolution.

In **Chapter 3**, I described the implementation of the method described in **Chapter 2**, Coev-Asymmetric-CNN. I provided a pipeline for the user to execute the method and I showed how to transform the data to input it into the model. Moreover, I tested Coev-Asymmetric-CNN on the bony vertebrates Selectome database. I evaluated its performance by varying the size of the training dataset. Furthermore, I identified 217 proteins under coevolution. Finally, I shed light on the limitations of this method and raised awareness to better understand the data when inferring coevolution.

Finally, annexes 1 and 2 contain additional projects that I set up during my thesis and on which I worked alongside two Master's students under my supervision.

Annex 1: Master First-step project, Karim Saied. "Data extraction and machine learning in features involved in coevolution" (2017)

Annex 2: Master First-step project, Léonard Jequier (2018). "Automatically classifying clownfish pictures datasets at the species level"

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, *22*(10), 1533–1545. https://doi.org/10.1109/TASLP.2014.2339736

Baussand, J., & Carbone, A. (2009). A combinatorial approach to detect coevolved amino acid networks in protein families of variable divergence. *PLoS Computational Biology*, *5*(9). https://doi.org/10.1371/journal.pcbi.1000488

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, Issue 4). Springer.

Blischak, P. D., Barker, M. S., & Gutenkunst, R. N. (2021). Chromosome-scale inference of hybrid speciation and admixture with convolutional neural networks. *Molecular Ecology Resources*, *June 2020*, 1–13. https://doi.org/10.1111/1755-0998.13355

Burger, L., & Van Nimwegen, E. (2010). Disentangling direct from indirect co-evolution of residues in protein alignments. *PLoS Computational Biology*, *6*(1), e1000633.

Caporaso, J. G., Smit, S., Easton, B. C., Hunter, L., Huttley, G. A., & Knight, R. (2008). Detecting coevolution without phylogenetic trees? Tree-ignorant metrics of coevolution perform as well as tree-aware metrics. *BMC Evolutionary Biology*, *8*(1), 1–25. https://doi.org/10.1186/1471-2148-8-327

Carmona, D., Fitzpatrick, C. R., & Johnson, M. T. J. (2015). Fifty years of co-evolution and beyond: integrating co-evolution from molecules to species. *Molecular Ecology*, *24*(21), 5315–5329. https://doi.org/10.1111/MEC.13389

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, *8*(1), 1–12. https://doi.org/10.1038/s41598-018-24271-9

Chiu, D. K. Y., & Kolodziejczak, T. (1991). Inferring consensus structure from nucleic acid sequences. *Bioinformatics*, *7*(3), 347–352.

Darwin, C. (1859). *The origin of species by means of natural selection*. Pub One Info.

Darwin, C. (1877). *The various contrivances by which orchids are fertilised by insects*. John Murray.

Davydov, I. I., Robinson-Rechavi, M., & Salamin, N. (2015). State aggregation for fast likelihood computations in phylogenetics. *BioRxiv*, 1–15.

https://doi.org/10.1093/bioinformatics/btw632

Dib, L., Salamin, N., & Gfeller, D. (2018). Polymorphic sites preferentially avoid co-evolving residues in MHC class I proteins. *PLoS Computational Biology*, *14*(5), 1–19. https://doi.org/10.1371/journal.pcbi.1006188

Dib, L., Silvestro, D., & Salamin, N. (2014a). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dib, L., Silvestro, D., & Salamin, N. (2014b). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dunn, S. D., Wahl, L. M., & Gloor, G. B. (2008). Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, *24*(3), 333–340. https://doi.org/10.1093/bioinformatics/btm604

Dutheil, J., Pupko, T., Jean-Marie, A., & Galtier, N. (2005). A model-based approach for detecting coevolving positions in a molecule. *Molecular Biology and Evolution*, *22*(9), 1919–1928. https://doi.org/10.1093/molbev/msi183

Dutheil, J. Y. (2012). Detecting coevolving positions in a molecule: Why and how to account for phylogeny. *Briefings in Bioinformatics*, *13*(2), 228–243. https://doi.org/10.1093/bib/bbr048

Ehrlich, P. R., & Raven, P. H. (1964). Butterflies and plants: a study in coevolution. *Evolution*, 586–608.

Ekeberg, M., Hartonen, T., & Aurell, E. (2014). Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences. *Journal of Computational Physics*, *276*, 341–356. https://doi.org/10.1016/j.jcp.2014.07.024

Fares, M. A., & Travers, S. A. A. (2006). A novel method for detecting intramolecular coevolution: Adding a further dimension to selective constraints analyses. *Genetics*, *173*(1), 9–23. https://doi.org/10.1534/genetics.105.053249

Fontaneto, D., Herniou, E. A., Boschetti, C., Caprioli, M., Melone, G., Ricci, C., & Barraclough, T. G. (2007). Independently Evolving Species in Asexual Bdelloid Rotifers. *PLOS Biology*, *5*(4), e87. https://doi.org/10.1371/JOURNAL.PBIO.0050087

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gouveia-Oliveira, R., & Pedersen, A. G. (2007). Finding coevolving amino acid residues using row and column weighting of mutual information and multi-dimensional amino acid representation. *Algorithms for Molecular Biology*, *2*(1), 1–12. https://doi.org/10.1186/1748-7188-2-12/COMMENTS

Heo, J., Lim, J. H., Lee, H. R., Jang, J. Y., Shin, Y. S., Kim, D., Lim, J. Y., Park, Y. M., Koh, Y. W., Ahn, S.-H., Chung, E.-J., Lee, D. Y., Seok, J., & Kim, C.-H. (2022). Deep learning model for tongue cancer diagnosis using endoscopic images. *Scientific Reports*, *12*(1), 1–10. https://doi.org/10.1038/s41598-022-10287-9

Hopf, T. A., Morinaga, S., Ihara, S., Touhara, K., Marks, D. S., & Benton, R. (2015). Amino acid coevolution reveals three-dimensional structure and functional domains of insect odorant receptors. *Nature Communications*, *6*(1), 1–7.

Horta1, E. R., & Weigt, M. (2021). On the effect of phylogenetic correlations in coevolution-based contact prediction in proteins. *PLoS Computational Biology*, *17*(5), 1–17. https://doi.org/10.1371/journal.pcbi.1008957

Hug, L. A., Baker, B. J., Anantharaman, K., Brown, C. T., Probst, A. J., Castelle, C. J., Butterfield, C. N., Hernsdorf, A. W., Amano, Y., Ise, K., Suzuki, Y., Dudek, N., Relman, D. A., Finstad, K. M., Amundson, R., Thomas, B. C., & Banfield, J. F. (2016). A new view of the tree of life. *Nature Microbiology 2016 1:5*, *1*(5), 1–6. https://doi.org/10.1038/nmicrobiol.2016.48

Ivankov, D. N., Finkelstein, A. V., & Kondrashov, F. A. (2014). A structural perspective of compensatory evolution. *Current Opinion in Structural Biology*, *26*(1), 104–112. https://doi.org/10.1016/j.sbi.2014.05.004

Jones, D. T., Buchan, D. W. A., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, *28*(2), 184–190. https://doi.org/10.1093/bioinformatics/btr638

Jones, D. T., Singh, T., Kosciolek, T., & Tetchner, S. (2015). MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, *31*(7), 999–1006.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*. https://doi.org/10.1038/s41586-021-03819-2

Källberg, M., Wang, H., Wang, S., Peng, J., Wang, Z., Lu, H., & Xu, J. (2012). Template-based protein structure modeling using the RaptorX web server. *Nature Protocols 2012*

7:8, *7*(8), 1511–1522. https://doi.org/10.1038/nprot.2012.085

Kamisetty, H., Ovchinnikov, S., & Baker, D. (2013). Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era. *Proceedings of the National Academy of Sciences*, *110*(39), 15674–15679.

Kandathil, S. M., Greener, J. G., & Jones, D. T. (2019). Prediction of interresidue contacts with DeepMetaPSICOV in CASP13. *Proteins: Structure, Function and Bioinformatics*, *87*(12), 1092–1099. https://doi.org/10.1002/prot.25779

Koonin, E. V., & Wolf, Y. I. (2012). Evolution of microbes and viruses: a paradigm shift in evolutionary biology? *Frontiers in Cellular and Infection Microbiology*, *2*, 119. https://doi.org/10.3389/FCIMB.2012.00119/BIBTEX

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Lavanya, P. M., & Sasikala, E. (2021). Deep Learning Techniques on Text Classification Using Natural Language Processing (NLP) In Social Healthcare Network: A Comprehensive Survey. *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, 603–609. https://doi.org/10.1109/ICSPC51351.2021.9451752

Lecun, Y., Bottou, L., Bengio, Y., & Ha, P. (1998). LeNet. *Proceedings of the IEEE*, *November*, 1–46.

Liu, J., Li, M., Luo, Y., Yang, S., Li, W., & Bi, Y. (2021). Alzheimer's disease detection using depthwise separable convolutional neural networks. *Computer Methods and Programs in Biomedicine*, *203*, 106032. https://doi.org/10.1016/j.cmpb.2021.106032

Liu, Z.-K., Zhang, L.-H., Liu, B., Zhang, Z.-Y., Guo, G.-C., Ding, D.-S., & Shi, B.-S. (2022). *Deep learning enhanced Rydberg multifrequency microwave recognition*. *2022*. https://doi.org/10.1038/s41467-022-29686-7

Manning, T., Wassan, J. T., Palu, C., Wang, H., Browne, F., Zheng, H., Kelly, B., & Walsh, P. (2019). Phylogeny-Aware Deep 1-Dimensional Convolutional Neural Network for the Classification of Metagenomes. *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018*, 1826–1831. https://doi.org/10.1109/BIBM.2018.8621543

Marks, D. S., Colwell, L. J., Sheridan, R., Hopf, T. A., Pagnani, A., Zecchina, R., & Sander, C. (2011). Protein 3D structure computed from evolutionary sequence variation. *PLoS ONE*, *6*(12). https://doi.org/10.1371/journal.pone.0028766

Marmier, G., Weigt, M., & Bitbol, A. F. (2019). Phylogenetic correlations can suffice to infer protein partners from sequences. *PLoS Computational Biology*, *15*(10), 1–24. https://doi.org/10.1371/journal.pcbi.1007179

Meyer, X., Dib, L., & Salamin, N. (2019). CoevDB: A database of intramolecular coevolution among protein-coding genes of the bony vertebrates. *Nucleic Acids Research*, *47*(D1), D50–D54. https://doi.org/10.1093/nar/gky986

Meyer, X., Dib, L., Silvestro, D., & Salamin, N. (2019). Simultaneous Bayesian inference of phylogeny and molecular coevolution. *Proceedings of the National Academy of Sciences*, *116*(11), 5027–5036. https://doi.org/10.1073/pnas.1813836116

Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., & Bateman, A. (2021). Pfam: The protein families database in 2021. *Nucleic Acids Research*, *49*(D1), D412–D419. https://doi.org/10.1093/NAR/GKAA913

Moretti, S., Laurenczy, B., Gharib, W. H., Castella, B., Kuzniar, A., Schabauer, H., Studer, R. A., Valle, M., Salamin, N., Stockinger, H., & Robinson-Rechavi, M. (2014). Selectome update: quality control and computational improvements to a database of positive selection. *Nucleic Acids Research*, *42*(D1), D917–D921. https://doi.org/10.1093/NAR/GKT1065

Muscat, M., Croce, G., Sarti, E., & Weigt, M. (2020). FilterDCA: Interpretable supervised contact prediction using inter-domain coevolution. *PLoS Computational Biology*, *16*(10), 1–19. https://doi.org/10.1371/journal.pcbi.1007621

Papkou, A., Gokhale, C. S., Traulsen, A., & Schulenburg, H. (2016). Host–parasite coevolution: why changing population size matters. *Zoology*, *119*(4), 330–338. https://doi.org/10.1016/J.ZOOL.2016.02.001

Pezoulas, V. C., Hazapis, O., Lagopati, N., Exarchos, T. P., Goules, A. V., Tzioufas, A. G., Fotiadis, D. I., Stratis, I. G., Yannacopoulos, A. N., & Gorgoulis, V. G. (2021). Machine Learning Approaches on High Throughput NGS Data to Unveil Mechanisms of Function in Biology and Disease. *Cancer Genomics & Proteomics*, *18*(5), 605–626. https://doi.org/10.21873/CGP.20284

Poisot, T., Stouffer, D. B., & Gravel, D. (2015). Beyond species: why ecological interaction networks vary through space and time. *Oikos*, *124*(3), 243–251. https://doi.org/10.1111/OIK.01719

Qin, C., & Colwell, L. J. (2018). Power law tails in phylogenetic systems. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(4), 690–695. https://doi.org/10.1073/pnas.1711913115

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., & Bridgland, A. (2019). Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins: Structure, Function, and Bioinformatics*, *87*(12), 1141–1148.

Sheu, Y. H. (2020). Illuminating the Black Box: Interpreting Deep Neural Network Models for Psychiatric Research. *Frontiers in Psychiatry*, *11*, 1091. https://doi.org/10.3389/FPSYT.2020.551299/BIBTEX

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*.

Trienens, M., & Rohlfs, M. (2011). Experimental evolution of defense against a competitive mold confers reduced sensitivity to fungal toxins but no increased resistance in Drosophila larvae. *BMC Evolutionary Biology*, *11*(1), 1–11. https://doi.org/10.1186/1471-2148-11-206/TABLES/3

Tropp, B. E. (2012). *Molecular biology: genes to proteins*. Jones & Bartlett Publishers.

Weigt, M., White, R. A., Szurmant, H., Hoch, J. A., & Hwa, T. (2009). *PNAS-2009-Weigt-67-72*. *106*(1). https://doi.org/10.1073/pnas.0805923106

Wen, F., Zhang, Z., He, T., & Lee, C. (2021). AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove. *Nature Communications 2021 12:1*, *12*(1), 1–13. https://doi.org/10.1038/s41467-021-25637-w

Woolhouse, M. E. J., Webster, J. P., Domingo, E., Charlesworth, B., & Levin, B. R. (2002). Biological and biomedical implications of the co-evolution of pathogens and their hosts. *Nature Genetics 2002 32:4*, *32*(4), 569–577. https://doi.org/10.1038/ng1202-569

Yamada, K., Davydov, I. I., Besnard, G., & Salamin, N. (2019). Duplication history and molecular evolution of the rbcS multigene family in angiosperms. *Journal of Experimental Botany*, *70*(21), 6127–6139.

Yeang, C. H., & Haussler, D. (2007). Detecting coevolution in and among protein domains. *PLoS Computational Biology*, *3*(11), 2122–2134. https://doi.org/10.1371/journal.pcbi.0030211

Yip, K. Y., Patel, P., Kim, P. M., Engelman, D. M., Mcdermott, D., & Gerstein, M. (2008). An integrated system for studying residue coevolution in proteins. *Bioinformatics*, *24*(2), 290–292. https://doi.org/10.1093/bioinformatics/btm584

Yu, X., Wu, Y., & Sun, X.-M. (2019). A Navigation Scheme for a Random Maze using Reinforcement Learning with Quadrotor Vision. *2019 18th European Control Conference (ECC)*, 518–523. https://doi.org/10.23919/ECC.2019.8795690

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). Deconvolutional networks. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2528–2535. https://doi.org/10.1109/CVPR.2010.5539957

Zerihun, Mehari B., & Schug, A. (2017). Biomolecular coevolution and its applications: Going from structure prediction toward signaling, epistasis, and function. *Biochemical Society Transactions*, *45*(6), 1253–1261. https://doi.org/10.1042/BST20170063

Zerihun, Mehari Bayou, & Schug, A. (2018). RNA Structure Prediction Guided by Coevolutionary Information. *Biophysical Journal*, *114*(3), 436a.

Zhang, X., & LeCun, Y. (2015). *Text Understanding from Scratch*. 1–9. http://arxiv.org/abs/1502.01710

Zizka, A., Andermann, T., & Silvestro, D. (2022). IUCNN – Deep learning approaches to approximate species' extinction risk. *Diversity and Distributions*, *28*(2), 227–241. https://doi.org/10.1111/ddi.13450

Zou, Z., Zhang, H., Guan, Y., Zhang, J., & Liu, L. (2020). Deep Residual Neural Networks Resolve Quartet Molecular Phylogenies. *Molecular Biology and Evolution*, *37*(5), 1495–1507. https://doi.org/10.1093/MOLBEV/MSZ307

Chapter 1

Benchmarking methods to infer sites under coevolution

"All models are wrong,
but some are useful."
George E. P. Box

Benchmarking methods to infer sites under coevolution

ABSTRACT

Identifying the mechanisms behind the process of coevolution is still an important question in molecular evolution. At the molecular level, coevolution can be detected when a modification at one site along the sequence triggers the modification at another site. In this context, coevolution can reveal important information about the function and structure of a protein, as these coordinated changes tend to occur to improve or maintain functional and structural interactions. Various methods have been proposed to predict coevolving pairs of sites. Most only consider coevolution as a measure of distance between amino acids. From a multiple sequence alignment (MSA) containing large numbers of sequences, they extract a measure of the correlation between sites to predict those that are close to each other in the protein 3D structure. They often use corrections and weight the sequences to reduce the impact of the phylogenetic relationships. Another type of methods aims at inferring the evolutionary constraints between pairs of sites (amino acids or nucleotides) using both a MSA and a phylogenetic tree to identify pairs of sites under coevolution. In this article, we focus on how to identify the pattern of molecular coevolution based on the role of evolutionary divergence. Using simulations, we evolve amino-acid sequences containing pairs of sites coevolving along phylogenetic trees. We then characterize the ability of each method to correctly detect coevolving sites under varying levels of sequence divergence and phylogenetic signals. Our study shows that divergence is important and has an impact on the way these methods infer coevolving pairs, and thus, it should be taken into consideration when predicting coevolution.

1. INTRODUCTION

Reciprocal evolutionary changes that occur between biological entities as they depend on each other are an important component of evolutionary biology (Yeang & Haussler, 2007). This biological process, called coevolution, can take place at the scale of molecules and the identification of coordinated changes between residues of a protein can reveal important information about the functional, structural and evolutionary constraints acting on them. Structural constraints occur when amino acids are in contact in the 3D structure of the protein and their proximity helps to maintain the protein folding (Carmona et al., 2015). Functional constraints occur when sites are implicated in a protein active site or are part of the interaction with other proteins (Yeang & Haussler, 2007). Finally, evolutionary constraints may come from other mechanisms to maintain the protein function through evolutionary time scales, such as epistasis or compensatory mutations (Dutheil et al., 2010). However, regardless of the underlying mechanisms involved, the initial task to understand the role played by coevolution in shaping protein sequences is to be able to correctly detect coevolving pairs of sites.

A variety of methods are available to infer coevolution between pairs of sites along a protein. The most popular methods are those looking at the covariation of the frequency patterns of amino-acids between sites using a local or global statistical model (e.g. maximum entropy). They all assume independence between protein sequences. One of the first measures to be proposed was mutual information (MI), which estimates the distance between the two probability distributions of amino acids at two different sites in a multiple sequence alignment (MSA; Chiu & Kolodziejczak, 1991; Burger & Van Nimwegen, 2010). However, MI can be biased by indirect associations between sites. Other methods, such as the most widely used direct coupling analysis (DCA) (Weigt et al., 2009) and protein sparse inverse covariance (PSICOV) (Jones et al., 2012), have been developed to remove these indirect effects. Estimating coevolution on a large MSA is computationally intensive and these methods have been improved using pseudo-likelihood estimation, like plmDCA (Ekeberg et al., 2014) or the very similar Gremlin (Kamisetty et al., 2013). Furthermore, methods, such as metaPSICOV (Jones et al., 2015), start to take advantages of machine learning techniques by combining results from other methods to obtain more accurate predictions of coevolution. Although methods based on deep learning algorithms are actively being developed, the scores provided by plmDCA is still a core feature added to most models.

The biochemical constraints imposed to maintain the interactions between sites in close proximity lead to covariance between the patterns of amino acid frequencies found in the columns of the MSA (Qin & Colwell, 2018). Detecting covariance between sites with the methods outlined above has therefore been used to help predicting amino acids in contact (Morcos et al., 2011; De Juan et al., 2013; Ekeberg et al., 2014; Zerihun & Schug, 2018), protein structure (Hopf et al., 2015; Kamisetty et al., 2013; Senior et al., 2019; Zerihun & Schug, 2018) and interfaces between proteins (Ackerman et al., 2012; Uguzzoni et al., 2017), or to identify interactions with other proteins (Malinverni et al., 2017).

However, protein sequences do not evolve independently and this will affect the similarities that we observe between sequences (Yeang & Haussler, 2007; Caporaso et al., 2008; Dutheil, 2012; Dib et al., 2014; Marmier et al., 2019). Evolutionary relationships between the sequences can create patterns in the MSA that are similar to structural constraints (Qin & Colwell, 2018) and bias the prediction of coevolution if the phylogenetic tree is not accounted for (Horta1 & Weigt, 2021). Structural methods assume that evolutionary and structural processes are independent. Corrections have been introduced to remove the effect of evolution with the goal of improving the predictions of coevolving sites (Qin & Colwell, 2018). Sequences are weighted according to their similarities measured by Hamming distances and the scores of pairs of sites are corrected with Average-Product Correction (APC) (Dunn et al., 2008). Still, the goal of these improvements is to remove or correct for the inherent evolutionary signal present in the data, which is due to the phylogenetic relationships between sequences.

Another approach is to account for this phylogenetic signal while estimating coevolution. In recent years, several phylogenetic methods have been proposed to estimate coevolving sites in a MSA. In all these methods, the protein structure is not considered at all. The relevant information used is the identity of the amino-acids or nucleotides and the way they changed through the branches of the phylogenetic tree (but see Dutheil, 2012). Several authors have proposed statistical and combinatorial methods based on phylogenetic profiles (Dutheil & Galtier, 2007; Baussand & Carbone, 2009; Dib et al., 2014) to infer coevolving sites. They rely on the idea that coevolving pairs of sites evolve in a co-dependent manner, which means that coevolving changes will occur on the same branches of a phylogenetic tree. For example, CoMap maps the substitution events onto the phylogenetic tree to fully incorporate the evolutionary history of each site (Priya & Shanker, 2021). Other methods, like Coev (Dib et al., 2014), use a Markov model to study the process of evolution along a phylogenetic tree for any pair of sites by inferring through maximum likelihood the substitution matrix that describes the transitions between positions, along the branches, that occur in a coordinated way during sequence evolution (Meyer, Dib, & Salamin, 2019; Yamada et al., 2019). One of the downsides of these methods is that they usually require more computing time than the structural methods.

More importantly, evolutionary processes can reflect the interplay of mechanisms that lead to coevolution between sites. For instance, divergence, which describes the evolutionary rates of a protein family and is defined as the average substitution rate at a given site, can play an important role because of its impact on the substitutions that are expected to occur within a MSA.

It was indeed shown (Dutheil, 2012) that low sequence divergence will reduce significantly the amount of information contained in the MSA to estimate coevolution. Moreover, with low divergence, random substitution occurring at two different sites can be over-represented in the MSA if it happened near the root of the phylogenetic tree (Dutheil, 2012). The impact of divergence could be more acute when large number of sequences are required to statistically show a signal of coevolution. This is the case for most structural methods that require large MSAs to improve the estimation of the frequencies of each amino-acid pair. However, increasing sequence numbers can sometimes only be achieved by adding sequences that are evolutionarily very similar depending on the protein family under study. For instance, the study of coevolution in the protein family MHC-I (Dib et al., 2018) was done with both a high number of human MHC-I sequences, but it was complemented with available mammalian sequences that increased the statistical power to detect sites under coevolution due to the higher variability of the latter dataset.

There is a need to compare and better understand the performance of the different methods to predict sites under coevolution. Some studies have used synthetic data, based on simulations of sequence evolution, to assess the accuracy of DCA in detecting coevolution not necessarily constrained by protein structure (Marmier et al., 2019). However, there has been no study analyzing in detail the effect of divergence on the accuracy of predicting pairs of sites as coevolving. We therefore have little knowledge about the behaviour of the different types of methods (i.e. structural vs phylogenetic) to analyse data with variable amount of sequence divergence and whether this factor will play a role in our ability to detect patterns of coevolution. Here, we compare three methods based on these two different approaches: Direct Coupling Analysis (DCA), which is the most widely used approach to predict structural dependencies, using the plmDCA software (Ekeberg et al., 2014), CoMap, which estimates the correlated evolution across branches of a phylogenetic tree between pairs of sites, and Coev, which models the evolution along a phylogenetic tree of coevolving pairs of sites. We used simulations to evolve amino-acid sequences along phylogenetic trees representing varying levels of evolutionary divergence (Dib et al., 2014) and we assessed the ability of each method to correctly detect coevolving sites under varying levels of sequence divergence and different tree topologies.

2. METHODS

2.1. Simulated data

2.1.1. Creating phylogenetic trees

We simulated two sets of phylogenetic trees with either 100 and 500 leaves, which we will refer to in this work as species. For each tree size, we varied the topologies by creating, in each simulation, 10 random phylogenetic trees using the R function **pbtree** from the phytools library (Revell, 2012). The birth and death parameters were set to 1 and 0, respectively. The trees were all binary, rooted and ultrametric.

Our main objective was to investigate the effect of sequence divergence on the ability to detect pairs of sites under coevolution. We selected six levels of divergence (0.005, 0.01, 0.02, 0.04, 0.08 and 0.16) that corresponded to the expected number of amino-acid substitutions on the branches of the phylogenetic trees (i.e., total sum of branch lengths divided by number of branches in the tree). These values were selected to cover scales of divergence observed in real phylogenetic trees inferred from amino acids. Such datasets typically present divergences between 0.006 and 0.15 substitutions per branch on average based on the data for bony vertebrates available in the Selectome database ((Moretti et al., 2014); more details in Suppl. Data 6.1. Calculating average branch length; Supp.Fig-1.1). We extended these values towards low and high amount of evolutionary changes to investigate more extreme cases of sequence divergence in our simulations. We modified the branch lengths of the ultrametric trees created by the function **pbtree** by drawing random values from an exponential distribution with a scale value given by the six levels of divergence described above (i.e., For a scale β, the mean of the distribution is $1/\beta$) and assigned branch lengths to each edge of the simulated trees.

For the larger phylogenetic trees (500 species), we investigated the effect of varying amounts of phylogenetic signal by transforming each simulated tree using Pagel's tree transformation (Blomberg et al., 2003; M. Pagel, 1999) as implemented in the **rescale** function from the R library **geiger** (Pennell et al., 2014). The parameter λ defines the scaling factor applied to the internal branches of the tree to down-weight the covariance structure corresponding to the phylogenetic tree and therefore reduces the phylogenetic signal. We scaled the branches of the trees with $\lambda \in \{0,0.3,0.6,0.8, 1\}$. If $\lambda = 1$, the phylogenetic trees were identical to the ones simulated, while $\lambda = 0$ lead to phylogenetic trees corresponding to a star phylogeny with species being fully independent from each other (and thus giving a covariance of 0 between the species).

2.1.2. Simulating multiple sequence alignments

We used each simulated phylogenetic tree (scaled and/or transformed) to create a MSA of 100 amino acids in length. We simulated two different parts to create each MSA: i) a set of 80 independent sites that evolved under standard models of amino acid substitutions; ii) a set of 10 pairs of sites that evolved under a model of coevolution.

For the 80 independent sites, we use two models. First, we used the LG model of evolution (Le & Gascuel, 2008) to simulate the sequences of amino acids (hereafter referred to as LG simulations). We used a homogeneous matrix of rate transition with all substitutions being identical and we assumed that each amino acid had the same equilibrium frequency but included a Gamma distribution (*alpha* = 2) to represent the variation in the rate of evolution of each site. Second, we simulated independent sites evolving according to the CAT model (Lartillot & Philippe, 2004, 2006; Lartillot et al., 2007; Quang et al., 2008). The CAT model creates classes of independent sites that evolved according to different equilibrium frequencies therefore creating patterns of amino acid frequencies across the alignment without any structural constraints. For the CAT model, the sites of the MSA were grouped in 20 classes[1] that had different equilibrium frequencies derived from available empirical datasets (Quang et al., 2008) .

We combined the 80 independent sites with 10 pairs of coevolving sites that were created by using the Coev model of evolution (Dib et al., 2014). The goal was to create true pairs of coevolving sites that are derived from evolutionary mechanisms and have no structural constraints associated. Each pair was created by drawing at random a coevolution profile and setting the *d* and *s* parameters that govern the rate of change in the coevolution model to 100 and 1, respectively (see Dib et al., 2014, for details about the model). The *d* parameter is the rate of transition of one amino-acid that will allow a pair of amino-acid that are not in the coevolving profile to move to a pair that is in the coevolving profile, while the *s* parameter is

[1] https://github.com/nsalamin/rocio/blob/master/scripts/simulator/README.md

the reverse transition. There are two additional parameters driving the substitution of single amino acids within each profile, and we set these to 1.

The simulations were done using a custom R script written by us and available at https://github.com/nsalamin/rocio.

2.2. Inferring coevolution

2.2.1. plmDCA

The algorithm plmDCA was used as implemented in the MATLAB version described in (Ekeberg et al., 2014). It is implemented with a weight parameter $w = 1 - p$, p being the Hamming distance between two sequences of a MSA below which they are assumed to be similar. For the small dataset with 100 species, we used the default parameter $w = 0.2$ (p = 0.8). For the larger dataset with 500 species, we tested the effect of the reweighting by using the default parameter $w = 0.2$ against no reweighting with $w = 0$ ($p = 1$). We also slightly modified the MATLAB code to output the scores before and after the APC correction. When not specified, the plmDCA scores are shown with APC correction and with the reweighting factor $w = 0.2$.

2.2.2. CoMap

We used CoMap v1.5.2 (Dutheil et al., 2005) to estimate the pairs of coevolving sites while accounting for the phylogenetic tree underlying their evolution. CoMap uses an independent model of evolution to map the substitutions occurring at each site on the branches of a phylogenetic tree. The vectors of substitutions rates (one value per branch of the tree) for each site are then compared in a pairwise analysis using a Pearson correlation. We used the LG model to estimate substitutions and added a Gamma distribution to account for rate variation between sites. We optimized the branch lengths and the model parameters using the full-derivatives algorithm and we used the marginal reconstruction of ancestral states as implemented in CoMap. We tested for coevolution between each pair of sites using the Pearson correlation and we used the significance level with a threshold of 0.05 to assess if a pair of sites was coevolving (p-*value* ≤ 0.05) or not (p-*value* > 0.05).

The p-*value* estimated by CoMap is based on a randomization procedure to take into account the rate of evolution of the sites of the MSA (see Dutheil et al., 2005 for details). The number of categories used to estimate the significance threshold for a given rate of evolution is fixed and can lead to sites falling in categories with very few randomized values. This can bias the *p-value* estimation and we modified the approach to solve this issue by adding either a parametric distribution of rates or by ensuring that the bin size for each category was large enough to include enough randomized patterns. The R script to perform the estimation of the new *p-values* for CoMap is available at https://github.com/nsalamin/rocio. In all our results, we used the new procedure to obtain the *p-values* rather than the default one returned by CoMap.

2.2.3. Coev

We finally used Coev (Dib et al., 2014), another phylogenetic method, to estimate the pairs of coevolving sites in the simulated datasets. The Coev method was used to simulate the data and we would expect a perfect match between the simulations and the estimations done. As with the previous methods, we performed pairwise analysis between each site of the alignment and we did not remove the constant sites. We used the maximum likelihood implementation of Coev and the rate of transition from one profile to another, parameters *d* and *s*, were estimated during the maximum likelihood optimization. The parameters *r1* and *r2* were set to 1.

The coevolving score between two sites was calculated using the Akaike information ($\Delta AIC = AIC_{independent\ model} - AIC_{Coev}$) as indicated in Dib et al., 2014. Higher ΔAIC values indicated that the signal of coevolution was stronger between the pair of sites tested than the null model of no coevolution. However, ΔAIC values can be biased in this specific model comparison and evidence for a coevolution should also be associated with a small s/d ratio (see Dib et al., 2014, for details).

2.3. Assessing the performance of the methods

The performance of each method was measured using precision and recall (PR) curves. We used this approach because our simulations are highly imbalanced with far more pairs of sites evolving under the independent model than pairs under coevolution (for a simulation with 100 bp with 80 independent sites and 20 coevolving ones, we had 4,940 pairwise comparisons that were true negatives versus 20 that were true positives). For plmDCA, we used the score to rank each pair of sites, while for CoMap, we combined the correlation coefficient and the *p-value* to rank each pair of sites. Finally, for Coev, we used both the ΔAIC value and the ratio of the parameters *s* and *d* to rank the pairs of sites. For each divergence levels simulated, we estimated the numbers of true positives (*TP*; coevolving pairs with a score higher than the threshold), false negative (*FN*; coevolving pairs with a score lower than the threshold), true negatives (*TN*; non-coevolving pairs with a score lower than the threshold) and false positive (*FP*; non-coevolving pairs with a score higher than the threshold). It allowed us to calculate the precision as $P = TP/(TP + FP)$ and Recall as $R = TP/(TP + FN)$ for each simulations that we averaged to obtain the mean PR curve for each level of divergence as well as the mean area under the PR curve (*AUC*; $AUC = 1$ means a perfect predictor, and $AUC < 0.5$ means a worse than random predictor). Finally, we also calculated the *F1* score, estimated as $2 * ((P * R)/(P + R))$, which is the harmonic average of the precision and recall, for each threshold value at each divergence level (the $F1 = 1$ means a perfect predictor, and $F1 = 0$ means a bad predictor). All analyses were done using custom scripts written in R and python. The scripts are available at https://github.com/nsalamin/rocio.

3. RESULTS

We tested the performance of several methods to infer coevolving pairs of sites along a MSA. Our goal was to investigate the ability of structural and phylogenetic methods to estimate coevolution due to a pure evolutionary process and to assess the effect of varying levels of sequence divergence on their performance. We used simulated datasets to benchmark the different methods and provide an estimate of true and false positives rates. This allowed us to characterize the properties of the methods compared. We report here the results first for the three different methods on the smaller data size containing 100 sequences and then for plmDCA for the larger dataset of 500 sequences.

3.1. Accuracy of methods on the 100 sequences dataset

We used plmDCA to estimate the score of coevolution for each simulated pair of sites. We showed in Fig-1.1 the different scores obtained by this method for coevolving and non-coevolving pairs of sites when the true negative pairs were simulated with the LG model and the true positive pairs with the Coev model. At all levels of divergence, the mean score of coevolving or non-coevolving pairs remained very similar, but the variance was much larger at low divergence than at higher divergence levels. This led to a larger overlap between the scores of the two types of pairs of sites at lower divergence. The overlap was evident with divergence levels of 0.005, 0.01, and 0.02 (Fig-1.1), where it was not possible to differentiate coevolving pairs from the others. The difference between the two distributions of scores increased with increasing divergence (levels of 0.04, 0.08 and 0.16; Fig-1.1) until a full disconnection occurred at the most extreme level of divergence (Fig-1.1). This pattern was similar for the simulations done with the CAT model to create the independent pairs of sites (Supp.Fig-1.5).

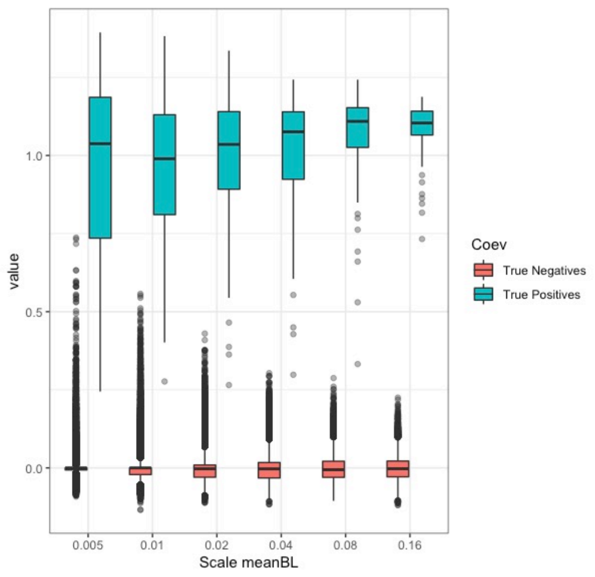


Fig-1.1 Boxplots of the scores obtained by plmDCA for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the LG model.

The PR curves and *F1* scores for plmDCA are shown in Fig-1.2. For the simulations under the independent model LG, the performance to classify coevolving and non-coevolving pairs of sites was very high, but with a slight decrease in performance with lower levels of divergence (Fig-1.2). The *AUC* was nevertheless high at all levels of divergence and ranged from 0.95 for a mean branch length of 0.005 to 0.99 for the higher levels (Table-1.1). The pattern was very similar with the simulations done with the CAT model to create independent sites (Table-1.1), although the *AUC* values were slightly lower than for the LG model.
We used the *F1* score to detect the threshold value that best allowed to distinguish between coevolving and non-coevolving pairs. This threshold increased when the levels of divergence decreased (Fig-1.2), while the *F1* score itself decreased when the level of divergence decreased. A *F1* score of 1 is achieved with the divergence scales of 0.04, 0.08 and 0.16, which reflect the pattern seen in Fig-1.1. We saw a similar pattern for the simulations done under the CAT model (see Supp.Fig-1.6).

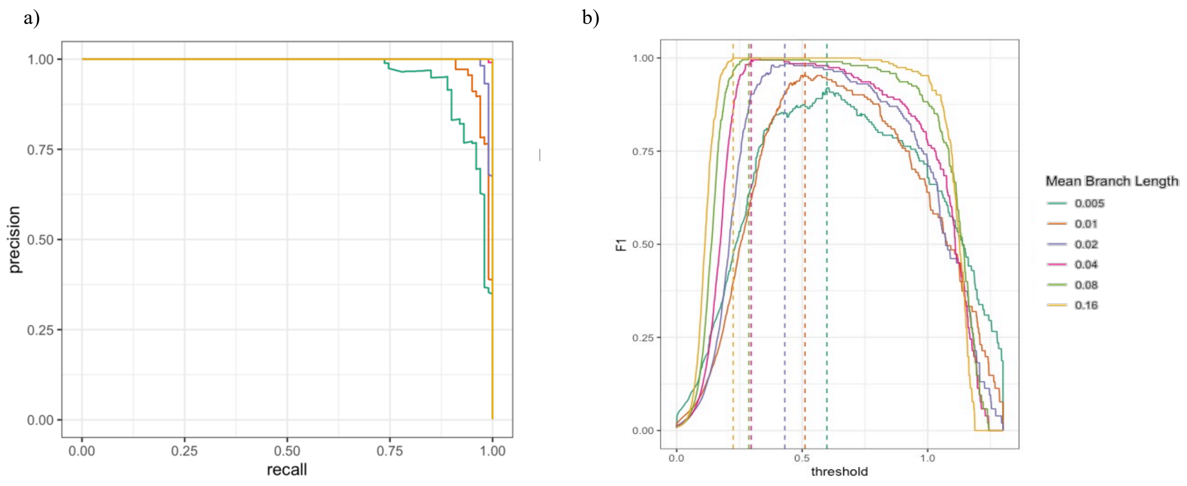


Fig-1.2 The results from the LG model are shown here. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for plmDCA scores. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for plmDCA. The y-axis represents the F1 score and x-axis represents the plmDCA scores. Each vertical line shows the plmDCA score threshold to achieve the best balance between precision and recall.

Mean branch length

	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
plmDCA LG	0.961	0.982	0.995	0.995	0.995	0.995
plmDCA CAT	0.938	0.984	0.993	0.995	0.995	0.995

Table-1.1 Area Under the Curve for the plmDCA analyses across the 6 levels of divergence for the simulations done with the LG and CAT models.

We analyzed the same simulated datasets with CoMap, which takes a phylogenetic approach to estimate pairs of coevolving sites. We showed in Fig-1.3 the distributions of correlation coefficients (after removing constant sites) obtained for coevolving and non-coevolving pairs. Contrary to plmDCA, the mean correlation changed between levels of divergence (increasing for non-coevolving pairs of sites with higher levels of divergence and decreasing for coevolving pairs of sites), while the variance stayed more similar, especially for the non-coevolving pairs (Fig-1.3). As a consequence, there was not a clear threshold to distinguish between coevolving or non-coevolving pairs of sites across all the divergence scales tested. Finally, at low divergence scales (0.005 to 0.04; Fig-1.3), some pairs of sites that are not coevolving showed a correlation of 1. This was due to the low substitution rates found in the sites involved in these pairs, which led to high correlations because all branches have similar, albeit low, rates of substitutions. At higher divergence scales, this effect disappeared because the probability of non-coevolving sites to change on different branches becomes higher, therefore leading to a smaller correlation coefficient than for coevolving pairs (which by assumption of the Coev model used to simulate the data must change on the same branches).

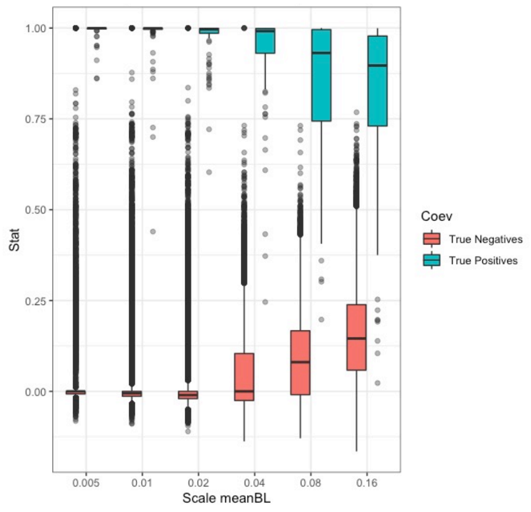


Fig-1.3 Boxplots of the correlation nscores obtained by CoMap for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the LG model.

The performance of using the correlation coefficient provided by CoMap for the simulations using the LG model (Fig-1.4) reflected the behavior already seen in Fig-1.3.At low divergence scales, we observed low precision, because of a higher number of false positives, but the recall was 1 because very few false negatives were found. Increasing the level of divergence increased the precision, but with a decreasing recall (Fig-1.4 a). However, the ability of this method to correctly predict the coevolving sites increased with higher levels of divergence, as shown by the AUC value (Table-1.2). The simulations with the CAT model gave similar, albeit slightly better, results (Supp.Fig-1.7; Supp.Table-1.1).

An advantage of CoMap over plmDCA is that it provides more information about the pairs of sites being evaluated than only a score. We combined the correlation coefficient with the estimated value of the minimum number of changes expected to occur through the tree for each pair of sites (*Nmin* parameter in CoMap output) and the *p-value* of the correlation coefficient. When combining both the correlation coefficient and the *p-value*, the estimated *AUC* did not improve (Table-1.2 and Supp.Fig-1.3). This is because correlation coefficients that are equal to 1 will most often have *p-values* smaller than 0.05 (Supp.Fig-1.4) and both values provided therefore very similar information. On the other hand, the false positive pairs that are associated with correlation coefficients of 1 or close to 1 at low divergence scales (Fig-1.4 a) are associated with low minimum number of expected substitutions across the phylogenetic tree. We removed the pairs of sites that were associated with a *Nmin* < 1.5 (i.e., one or less expected substitutions on these sites across the phylogenetic tree) and found that the performance of CoMap drastically improved at low levels of divergence (i.e. 0.005, 0.01, 0.02 and 0.04) with *AUC* values higher than 0.93 for all scales (Fig-1.5; Table-1.2). The performance at high levels of divergence (0.08 and 0.16) did not improve given that the value of minimum number of changes at these scales is higher (*Nmin* > 1.5) than at low scales.

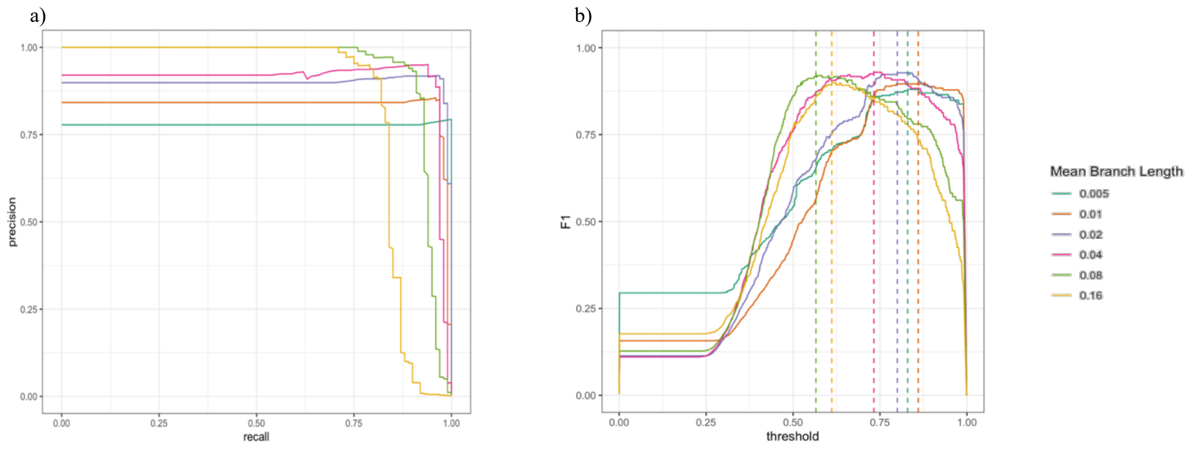


Fig-1.4 Results from LG model and CoMap are shown here: taking into account only the correlation coefficient and the significant pairs of sites. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for CoMap correlation coefficient. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for CoMap correlation coefficients. The y-axis represents the F1 score and x-axis represents the CoMap correlation coefficient. Each vertical line shows the CoMap correlation coefficient threshold to achieve the best balance between precision and recall.

Mean branch length

	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
Correlation coefficient	0.724	0.83	0.918	0.898	0.938	0.846
Correlation coefficient and *p-value*	0.77	0.825	0.874	0.879	0.936	0.887
Correlation coefficient and Nmin value	0.936	0.965	0.989	0.97	0.941	0.846
Correlation coefficient, *Nmin* and *p-value*	0.933	0.959	0.974	0.947	0.939	0.887

Table-1.2 Area Under the Curve for the CoMap analyses across the 6 levels of divergence for the simulations done with the LG model.

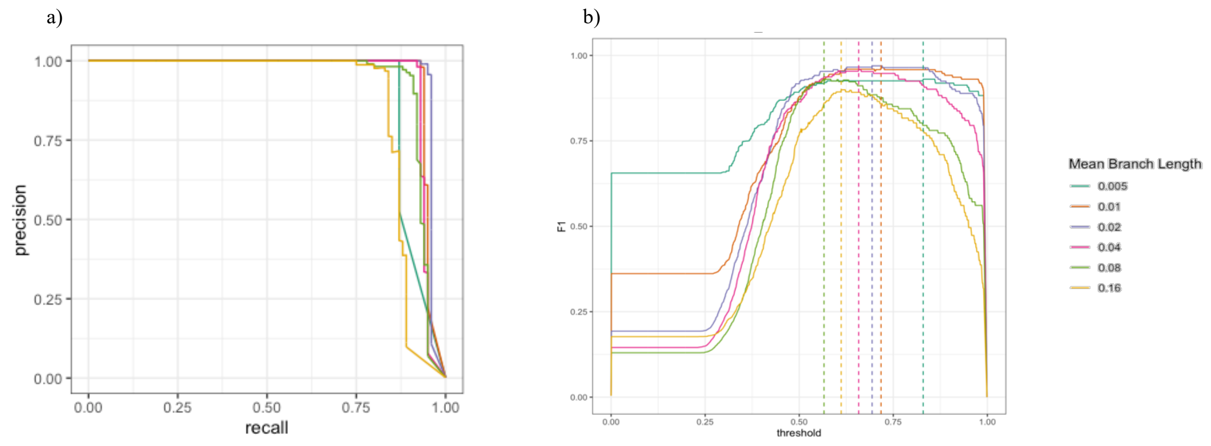


Fig-1.5 Results from LG model and CoMap are shown here: taking into account only the correlation coefficient and the significant pairs of sites with a Nmin > 1.5. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for CoMap correlation coefficient. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for CoMap correlation coefficients. The y-axis represents the F1 score and x-axis represents the CoMap correlation coefficient. Each vertical line shows the CoMap correlation coefficient threshold to achieve the best balance between precision and recall.

We analyzed the LG simulated dataset with the Coev model, which takes also a phylogenetic approach to estimate pairs of coevolving sites but uses a Markov model of evolution to estimate using maximum likelihood the probability that the pairs of sites are evolving under coevolution. We show in Fig-1.6 the distributions of ΔAIC values (after removing constant sites) obtained for coevolving and non-coevolving pairs. Contrary to plmDCA, but similar to CoMap, the mean correlation varies between levels of divergence, while the variance stayed more similar, especially for the non-coevolving pairs (Fig-1.6). As a consequence, there was not a clear threshold to distinguish between coevolving or non-coevolving pairs of sites across all the divergence scales tested. Additionally, we also show in Supp.Fig-1.10 the distributions of *s/d* values to detect pairs of sites under coevolution and we found that there was not a clear threshold to distinguish between non-coevolving pairs and the coevolving ones.

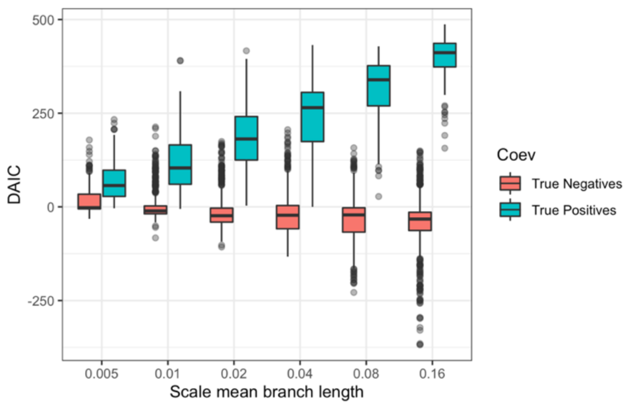


Fig-1.6 Boxplots of the ΔAIC scores obtained by Coev for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the LG model.

Similar to CoMap, Coev also provides more information about the pairs of sites than only a score. First, we used the ΔAIC measure to select the best model of evolution and to detect pairs of sites under coevolution. We applied this approach to the simulated dataset based on the LG model and we found levels of performance that were slightly better with Coev (Fig-1.7) than with CoMap when only taking into account the significant correlation coefficient.

Then again, plmDCA performs better in general. Nevertheless, if we look at the AUC when using as a score the ΔAIC value from Coev (Table-1.3), at high scale (mean branch length of 0.16), Coev performs as good as plmDCA and better than CoMap (Table-1.1, Table-1.2 and Table-1.3).

Moreover, we also used the s/d ratio measure to select the best model of evolution and to detect pairs of sites under coevolution. We also applied it on the LG model (Fig-1.7) and we found that Coev is slightly better than CoMap when only taking into account the significant correlation coefficient, but with high mean branch length its behavior is not good. Looking into these results, we saw that the ΔAIC and the s/d ratio are complementary values. At high divergence, we see that the top pairs predicted as coevolving pairs, have the same s/d ratio, being not possible to differentiate between true positives and false positives. However, if we take into account the top pair of sites based on the s/d ratio and also having the highest ΔAIC, then the top pairs of sites are all true coevolving pairs of sites.

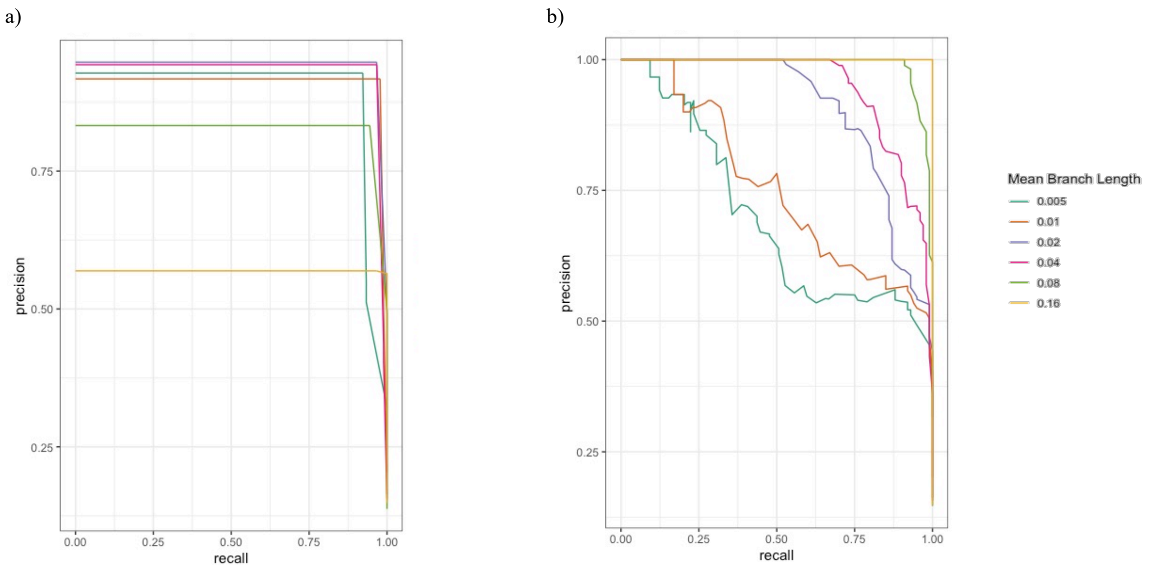


Fig-1.7 Results from LG model and Coev are shown here. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for Coev considering the s/d score. The y-axis represents the precision and the x-axis represents the recall. b) PR curve for Coev considering the ΔAIC. The y-axis represents the precision and the x-axis represents the recall.

Mean branch length

	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
Coev with s/d ratio	0.893	0.909	0.934	0.93	0.813	0.562
Coev with ΔAIC	0.695	0.752	0.902	0.945	0.986	0.996

Table-1.3 Area Under the Curve for the Coev analyses across the 6 levels of divergence for the simulations done with the LG model.

3.2. Accuracy of structural methods with varying levels of phylogenetic signal

We studied the effect that the evolutionary process had on the plmDCA predictions. Our goal was to assess whether the corrections for phylogenetic relationships implemented in plmDCA was effectively removing covariation due to the evolutionary process. We used larger alignments than in the previous simulations to ensure that the statistical properties of the method were met (i.e. it is suggested that 5 times more sequences than sites should be present; (Kamisetty et al., 2013)). However, we could not apply CoMap or Coev on these larger alignments because either limitations in the memory usage of the software, for the former, or due to prohibitive computational costs, for the latter. We first checked that the performances observed with the larger dataset containing 500 species were similar to those found on the dataset with 100 species. The *AUC* of the PR curves for each level of divergence showed the same decrease with lower levels of divergence (Table-1.4) and the overall *AUC* for lower divergence was lower than for the smaller dataset (Table-1.1).

Mean branch length

	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
plmDCA LG	0.844	0.929	0.968	0.995	1	1
plmDCA CAT	0.861	0.933	0.976	0.994	0.999	1

Table-1.4 Area Under the Curve for the plmDCA analyses, for 500 species, across the 6 levels of divergence for the simulations done with the LG and CAT models.

We next looked at the impact of various levels of phylogenetic signal, defined here by the parameter λ, which can vary from 0 to 1. We showed in Fig-1.8 the different scores obtained by the plmDCA method on coevolving and non-coevolving pairs of sites when the true negative pairs were simulated with the LG model and the true positive pairs with the Coev model. When $\lambda = 1$, we observed the same behavior as with the previous small alignments (see Fig-1.1) except that the mean of the coevolving pairs increased with the divergence. When $\lambda = 0$, distributions showed much less outliers and the predictions made by plmDCA were almost perfect. Independent sequences seemed to help the method correctly detecting the coevolving pairs of sites. For each λ higher than 0, simulations with mean branch length less than 0.04 tended to overlap more than with mean branch length higher than 0.04.

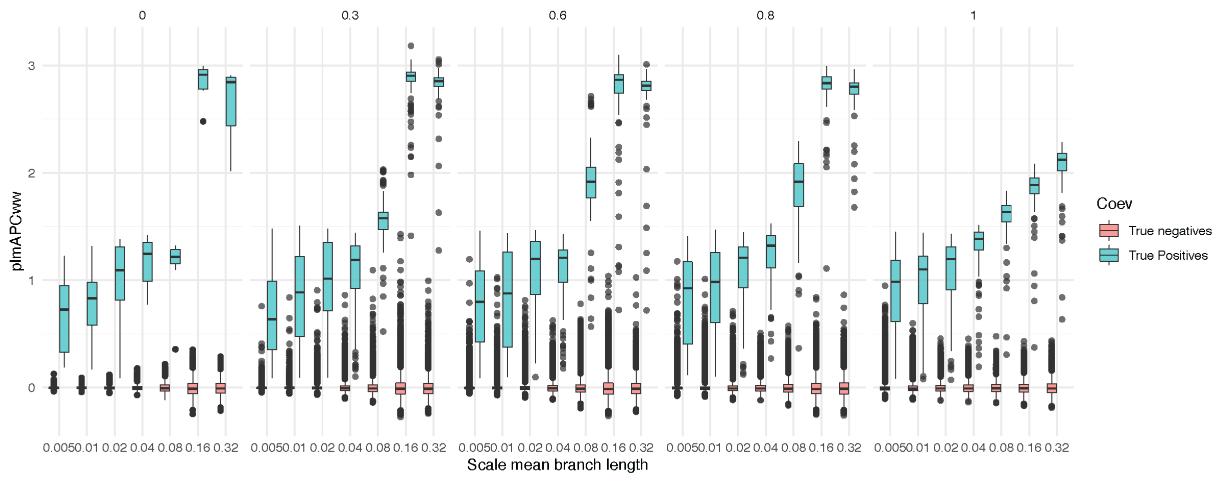


Fig-1.8 Boxplots showing the distribution of plmDCA scores for coevolving and non-coevolving pairs according to mean branch length. Different λ are shown from left to right; the value at the top of the plot indicates the value of λ. The LG model is shown here.

We quantified the ability for DCA to discriminate between coevolving and non-coevolving pairs using AUROC according to the mean branch length and λ. In Fig-1.9, we observed that for mean branch length higher than 0.04, the AUROC was almost 1 for all λ. For mean branch length smaller than 0.04, the AUROC decreased with λ, except for $\lambda > 0.6$, where the AUROC increased again but without reaching 1. We confirmed that plmDCA gave high AUROC values (in between 0.8 and 1) but that detecting coevolving pairs in low divergence alignments with a $\lambda > 0$ is a more difficult task for plmDCA. In Supp.Fig-1.11, we compared plmDCA with Mutual Information (MI) and showed that MI was performing better, with a AUROC higher than 0.97 for all alignments. Moreover, the results were similar whether the simulations were done with the CAT or LG models.

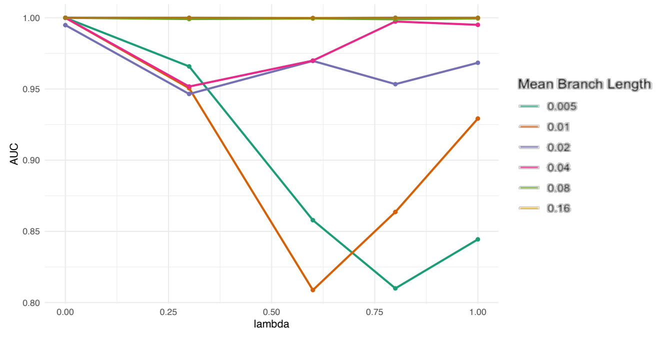


Fig-1.9 AUROC for plmDCA scores to mean branch length. The y-axis is the AUC value and the x-axis is the λ value. The green, orange, purple, pink, light green, yellow and brown colors represent each level of divergence (mean branch length). The LG model is shown here.

We compared different versions of plmDCA scores, depending on the phylogenetic corrections used, with and without the sequences weighting, and with and without the APC correction. Fig-1.10 shows the AUROC of precision recall for different branch length and λ = 1 for different corrections. The AUROC is increasing for all mean branch lengths when the APC correction is applied (with the weighting of sequences: pink against orange and without: purple against green). However, the AUROC is decreasing for low mean branch length when the weighting is applied (with APC: pink against purple, without APC: orange against green). In the supplementary data, results are shown for all λ and for the CAT model as well (see Supp.Fig-1.11 and Supp.Fig-1.12)

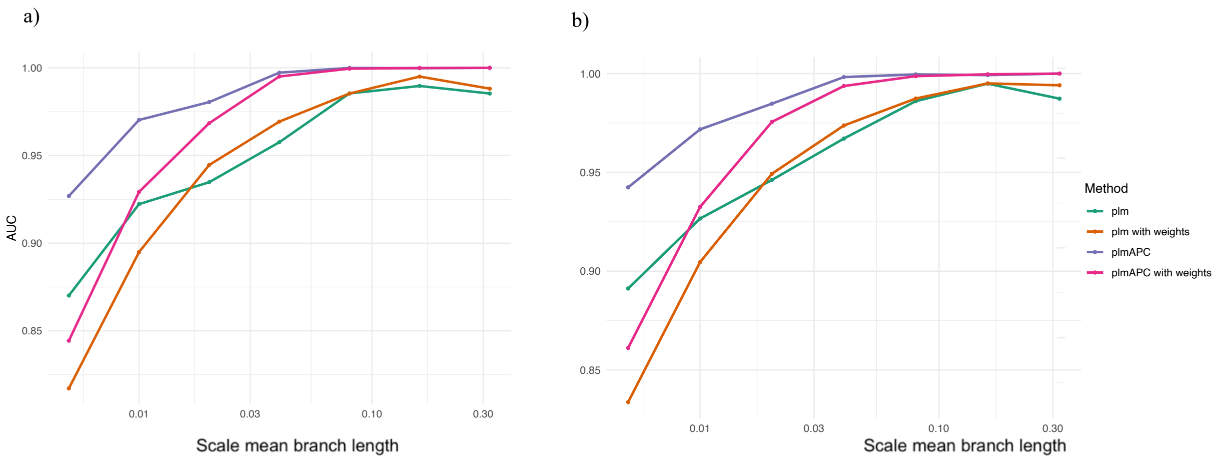


Fig-1.10 AUROC Precision and Recall according to the mean branch length (logscale) for different versions of plmDCA depending on the corrections used. The green curve shows the plmDCA scores with no reweighting and no APC correction. Orange is when sequences are weighted according to their Hamming distances. Purple is when the APC correction is applied. Finally pink curve shows the plmDCA scores with both weighting and APC correction. a) LG model; b) CAT model.

4. DISCUSSION

In this work, we investigated the effects of sequence divergence on the way methods, like plmDCA, CoMap or Coev, predict pairs of sites as coevolving: we highlight the good performance of plmDCA independently of the divergence level; we do an extensive evaluation of CoMap performance according to different parameters; we evaluate the performance of Coev; and we identify the limitations of these methods based on the six divergence levels simulated with Coev.

Similar to previous studies in which controlled synthetic data was generated (Marmier et al., 2019), we utilize evolutionary models (LG and CAT) to simulate data only based on evolutionary constraints and we show that plmDCA is able to infer with good accuracy, overall, all the pairs of sites under coevolution as reported in Marmier et al., 2019.

Amino acids in contact need to keep this interaction to maintain the protein 3D structure, which induce correlations in amino acids between homologous sequences. Methods like DCA are able to infer these correlations, thus, they are highly used to predict the structure of proteins (Muscat et al., 2020). Moreover, these methods remove correlated sites not in contact, while the results showed in this study suggest that these pairs are not only associated with structural constraints, but they could also still be the signature of genuine coevolving sites and should be considered for other studies not structure related.

Even though we simulate data under the CAT model assuming it would generate a false coevolving pattern (due to the pair-site variation equilibrium state frequencies utilized by the model) plmDCA is able to still infer the coevolving pairs of sites. Given that we were using a high number of site categories, in the future it would be interesting to reduce it and see how plmDCA behaves.

Interestingly, it seems that there is a threshold score based on the divergence level, allowing a clear differentiation between pairs with and without coevolution. For low divergence (average branch length of 0.005 and 0.01) we show that the threshold score should be bigger than 0.5 while for high divergence (average branch length of 0.16, 0.08 and 0.04) the threshold score should be above 0.2. For average divergence the threshold score is above 0.4.

However, even if we can define a threshold at different divergence levels, at low scales there is an unavoidable overlap between pairs of sites of different classes. These findings support the notion that divergence should be considered to better understand the plmDCA predictions. Notwithstanding, given that plmDCA is only given a score to classify the sites, it is harder to understand what is happening in these sites.

When using larger alignments, plmDCA performance slightly decreases, in case of low mean branch length. However, the number of samples used and the changes are not significant to draw any conclusion.

We observe that increasing the phylogenetic dependence between sequences (i.e., increasing λ) decreased the performance of plmDCA even when the scores were corrected for the phylogeny. When there is no dependence between sequences, plmDCA is really accurate for all mean branch lengths. This suggests that the low performance of plmDCA on small branch length trees is due to the shared history of the sequences that is not well taken into account by the corrections.

When looking into the effects of the corrections, we realized that the APC has a positive effect but that the reweighting gives worst results. More surprisingly, the performance of plmDCA with and without phylogenetic corrections is lower than the simplest Mutual Information in almost every case (see Supp.Fig-1.11 and Supp.Fig-1.12). The APC correction used in plmDCA helps increasing the performance but is not enough to reach MI's good performance on our datasets that contain only pairs of coevolving sites. On the contrary, the reweighting decreases or does not improve the predictions as already mentioned in Hockenberry & Wilke, 2019. Because it downweights similar sequences, it also drastically reduces the amount of information of the MSA. This seems to affect the performance, especially when similar sequences are not due to an over-representation of one organism but are due to the evolutionary process itself.

We also show that increasing the number of sequences does not help with dealing with a reduced amount of information in the MSA, for low divergence cases.

In all experiments, the results between CAT and LG models are very similar. The type of evolutionary model used seems not to matter for creating randomly correlated pairs.

Contrary to the single output provided by plmDCA, CoMap provides wide information about the predicted pairs. Nevertheless, CoMap's performance is not as good as expected, given that it is taking evolutionary constraints into account with the phylogenetic tree as an input.

In line with the results given by plmDCA, we show that divergence has an impact on the way CoMap infers coevolution. We find that at low divergence levels, where there are more conserved sites, the Pearson correlation score, given by CoMap, is biased by nearly constant sites. Whereas past studies have shown that it should not be the case (Dutheil, 2012), we see that in pairs of sites where there is one substitution on one branch, CoMap gives a significant (*p-value* < 0.05) and high correlation (*correlation* ~ 1). We found that it occurred in datasets with long basal branches, at low scales, where a single change can happen across several independent sites, near the root of the tree, and lead to a false prediction by CoMap (see Supp.Fig-1.2 and Dutheil, 2012).

It is important to note that because CoMap provides the *Nmin* statistic, it is possible to detect these types of substitutions, at long basal branches. With the *Nmin* statistic, we understand better the process behind the prediction and if the number of expected substitutions on these sites are small (i.e., one or less) then we can remove the pair of sites, and reduce the number of false positives cases.

Nevertheless, when the divergence is too high, in case of high mean branch length (0.16), the tree branches are long and it is not possible to correct with the *Nmin* statistic. The minimum number of changes is increasing at the same time as the length of the branches increase. CoMap get lost with so many possible changes and there is not a clear threshold to differentiate between pairs of sites with or without coevolution.

One approach to avoid falsely interpreting nearly constant sites as coevolving would be the use of the Jaccard distance instead of the Pearson correlation. The advantage of the Jaccard distance (defined as

$$J(A,B) = \frac{s_a \cap s_b}{s_a \cup s_b}$$

) is that it does not count common patterns due to no change.

Last but not least, analogous to the results that we see from CoMap and plmDCA, we find that divergence has also an impact on Coev's performance.

Similar to CoMap, Coev also utilize the phylogenetic tree to infer coevolving pairs of sites and thus, it provides more statistics to better understand the predicted pairs such as the s/d ratio. Therefore, we evaluated Coev taking into account two of its main statistics: the ΔAIC value and the s/d ratio. At low scales (mean branch length of 0.005 and 0.01), Coev cannot infer most of the positions under coevolution correctly when considering the ΔAIC value. However, when increasing the average branch length, Coev has as good performance as plmDCA. Contrary to the results considering the ΔAIC value, when evaluating the behavior of Coev with the s/d ratio, we appreciate that it has a good performance in most of the mean branch length scenarios, except when the mean branch length is high (0.16). We also saw a similar behavior with CoMap, where the accuracy of the model starts to drop when the mean branch length is too high (0.16).

However, if we look a bit deeper in the top pairs of sites predicted by Coev (see Supp.Table-1.2), when the mean branch length is 0.005 or 0.16, we see that the top 13 pairs of sites, with the lowest s/d ratio, have the same s/d ratio, meaning that it is not possible to distinguish correctly which are the true coevolving pairs of sites. Therefore, this may be one of the reasons why the AUC values are lower (Table-1.3). One approach to avoid this kind of situation and being able to correctly select the true coevolving pairs of sites, is to order the top values based first on the s/d ratio and secondly on the ΔAIC highest values. Like this (see Supp.Table-1.2), the top values are the true coevolving sites. This new measure, combining the s/d ratio and the ΔAIC, would avoid the impact of the divergence when predicting coevolving pairs of sites as suggested by Dib et al., 2014,.

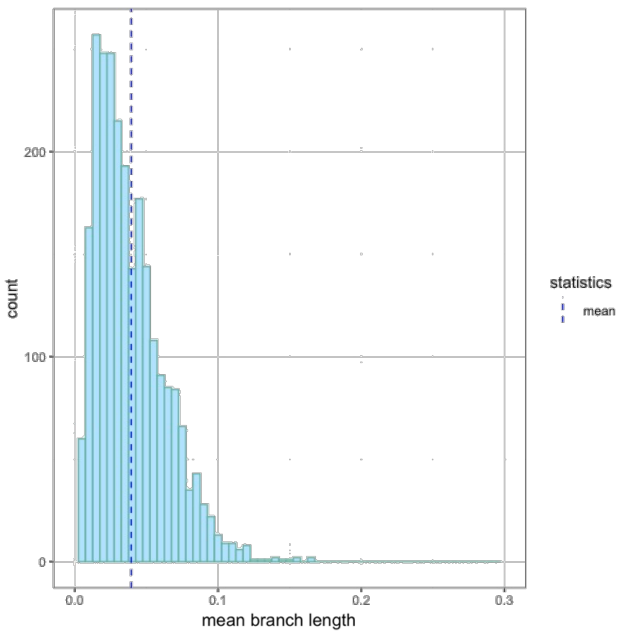
In conclusions, Coev and CoMap are methods which provide more information about the predicted coevolving pairs of sites. This would improve our knowledge about the evolutionary pressures pushing for this process to occur and, moreover, these methods provide more measurement to control that the predictions are reliable. Nevertheless, Coev and CoMap took longer to run and compare each pair of sites of the full alignment in comparison with plmDCA.

Last but not less, the simulations used in this work create only coevolving pairs, omitting triplets or bigger chains of coevolution. Work needs to be done to compare both type of coevolution inference with more complex chains of correlations between sites. In particular simulating more complex chain is still a research topic.

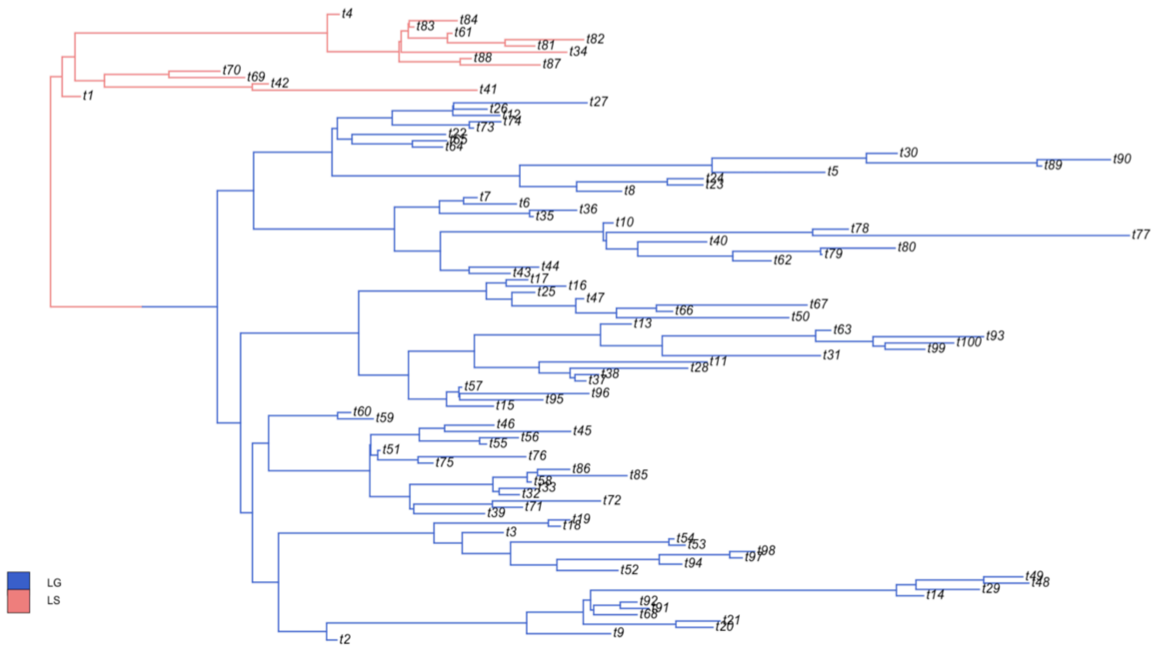
6. SUPPLEMENTARY DATA

6.1. Calculating average branch length

We calculated the average tree branch length from a dataset containing more than 15k samples. We used the "bony vertebrates" dataset Euteleostomi, from the positive selection database Selectome (Proux et al., 2009; Moretti et al., 2014) as it has been used for coevolution study in other studies (Meyer, Dib, & Salamin, 2019; Meyer, Dib, Silvestro, et al., 2019). This dataset was used under studies to infer coevolution in nucleotides. To calculate the average branch length for amino acids, we had to translate the codon-based alignment to amino acid base. The tree branch length corresponding to the alignment was re-estimated, using PhyML (Guindon et al., 2010), based on the new amino acid alignment. Finally, we selected only the samples containing between 50 and 150 species. In Supp.Fig-1.1 we show the average branch length distribution for the amino acids, where the percentiles of 0.25 and 0.75 are, respectively: 0.006 and 0.15.

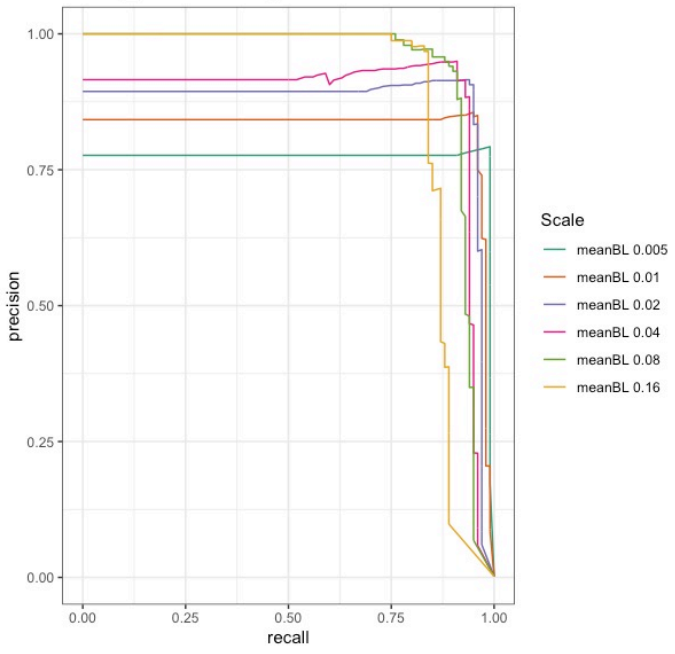


Supp.Fig-1.1 Histogram showing the mean branch length distribution for amino acids in the Euteleostomi dataset

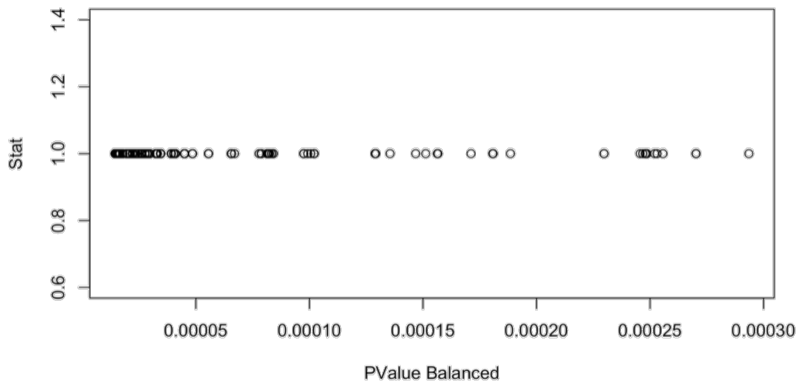


Supp.Fig-1.2 Changes happening for site 35 and 62 at meanBL scale of 0.005

6.2. CoMap LG results

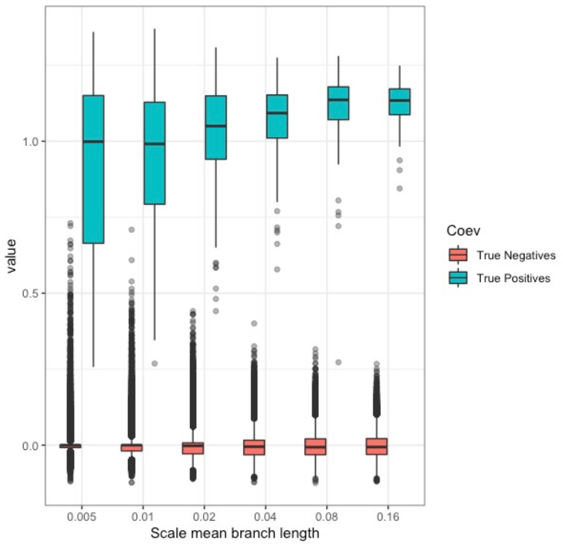


Supp.Fig-1.3 PR Curve from CoMap with the LG model is shown here: taking into account only the correlation coefficient and the significant pairs of sites. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). The y-axis represents the precision and the x-axis represents the recall.

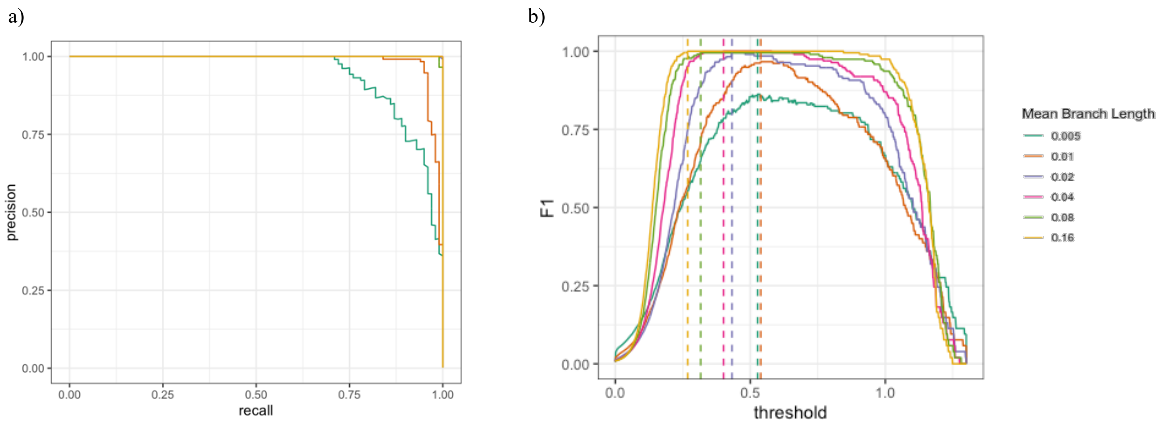


Supp.Fig-1.4 Relationship between correlation coefficient value (Stat) and PValue when Stat = 1.

6.2. plmDCA CAT results

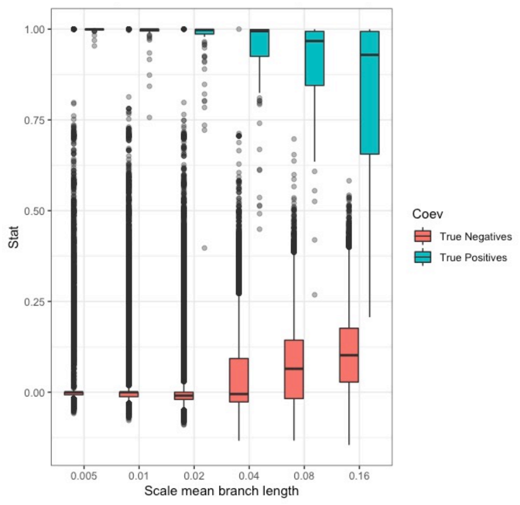


Supp.Fig-1.5 Boxplots of the scores obtained by plmDCA for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the CAT model.



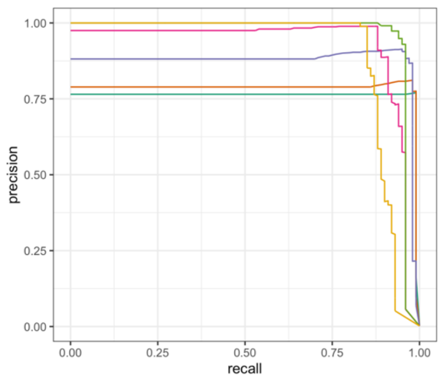
Supp.Fig-1.6 The results from the CAT model are shown here. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for plmDCA scores. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for plmDCA. The y-axis represents the F1 score and x-axis represents the plmDCA scores. Each vertical line shows the plmDCA score threshold to achieve the best balance between precision and recall.

6.3. CoMap CAT results

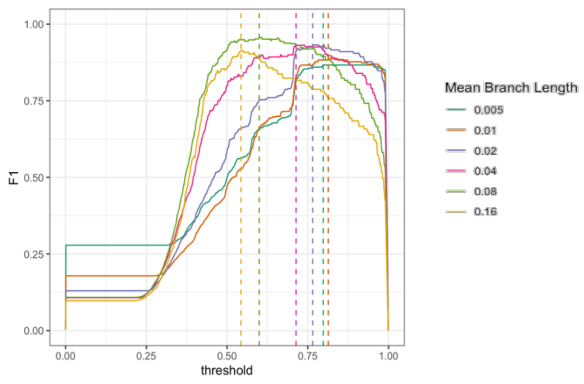


Supp.Fig-1.7 Boxplots of the correlation scores obtained by CoMap for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the CAT model.

a)

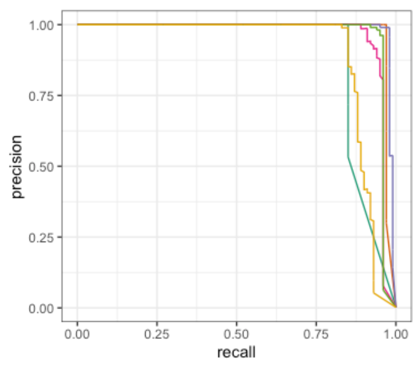


b)

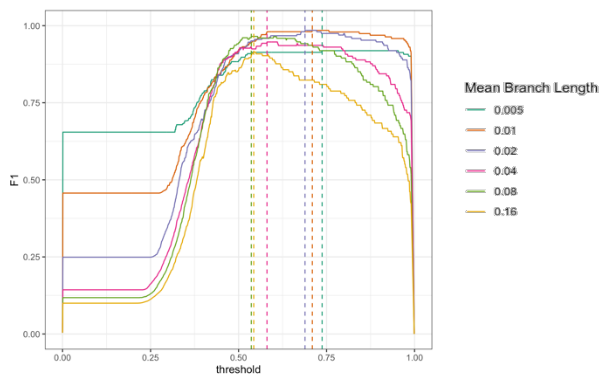


Supp.Fig-1.8 Results from CAT model and CoMap are shown here: taking into account only the correlation coefficient and the significant pairs of sites. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for CoMap correlation coefficient. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for CoMap correlation coefficients. The y-axis represents the F1 score and x-axis represents the CoMap correlation coefficient. Each vertical line shows the CoMap correlation coefficient threshold to achieve the best balance between precision and recall.

a)



b)



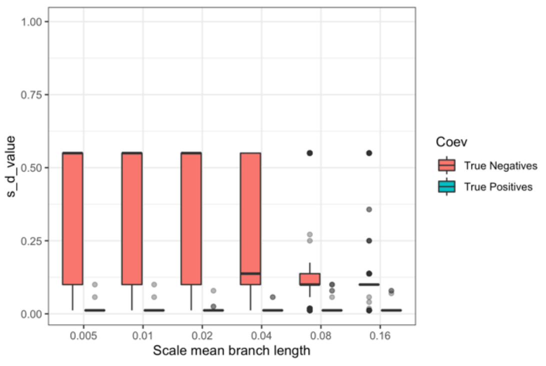
Supp.Fig-1.9 Results from CAT model and CoMap are shown here: taking into account only the correlation coefficient and the significant pairs of sites with a Nmin > 1.5. The green, orange, purple, pink, dark blue and yellow colors represent each level of divergence (mean branch length). a) PR curve for CoMap correlation coefficient. The y-axis represents the precision and the x-axis represents the recall. b) F1 score plot for CoMap correlation coefficients. The y-axis represents the F1 score and x-axis represents the CoMap correlation coefficient. Each vertical line shows the CoMap correlation coefficient threshold to achieve the best balance between precision and recall.

Mean branch length

	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
Correlation coefficient	0.758	0.783	0.87	0.93	0.97	0.9
Correlation coefficient and *p-value*	0.758	0.783	0.87	0.93	0.97	0.9
Correlation coefficient and Nmin value	0.923	0.982	0.986	0.967	0.974	0.906
Correlation coefficient, *Nmin* and *p-value*	0.923	0.982	0.986	0.967	0.974	0.906

Supp.Table-1.1 Area Under the Curve for the CoMap analyses across the 6 levels of divergence for the simulations done with the CAT model.

6.4. Coev LG results



Supp.Fig-1.10 Boxplots of the s/d scores obtained by Coev for the true coevolving (in blue) and true non-coevolving (in red) pairs of sites at different levels of divergence with the simulations done with the LG model.

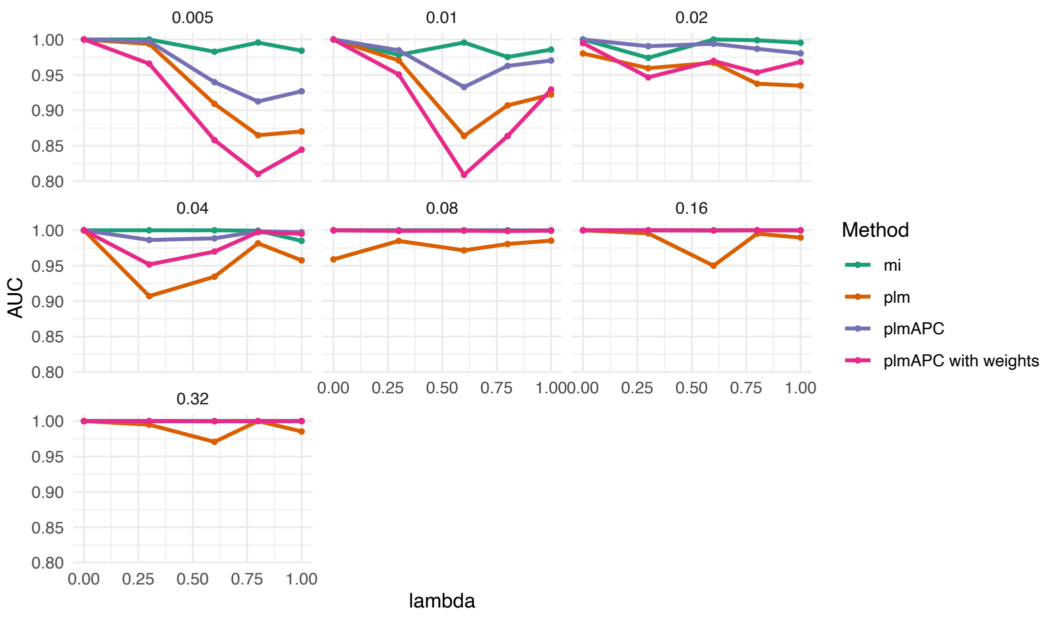
Site 1	Site 2	Delta AIC	s/d ratio	Coev
15	16	460.79	0.01176	1
5	6	446.08	0.01176	1
7	8	440.40	0.01176	1
17	18	438.66	0.01176	1
11	12	419.10	0.01176	1
1	2	418.35	0.01176	1
3	4	395.47	0.01176	1
9	10	387.85	0.01176	1
13	14	253.93	0.01176	1
19	20	223.35	0.01176	1
16	17	145.11	0.011765	0
6	7	113.63	0.011765	0
2	3	70.34	0.011765	0

Supp.Table-1.2 Coev results: top 15 values one simulation and mean branch length 0.16

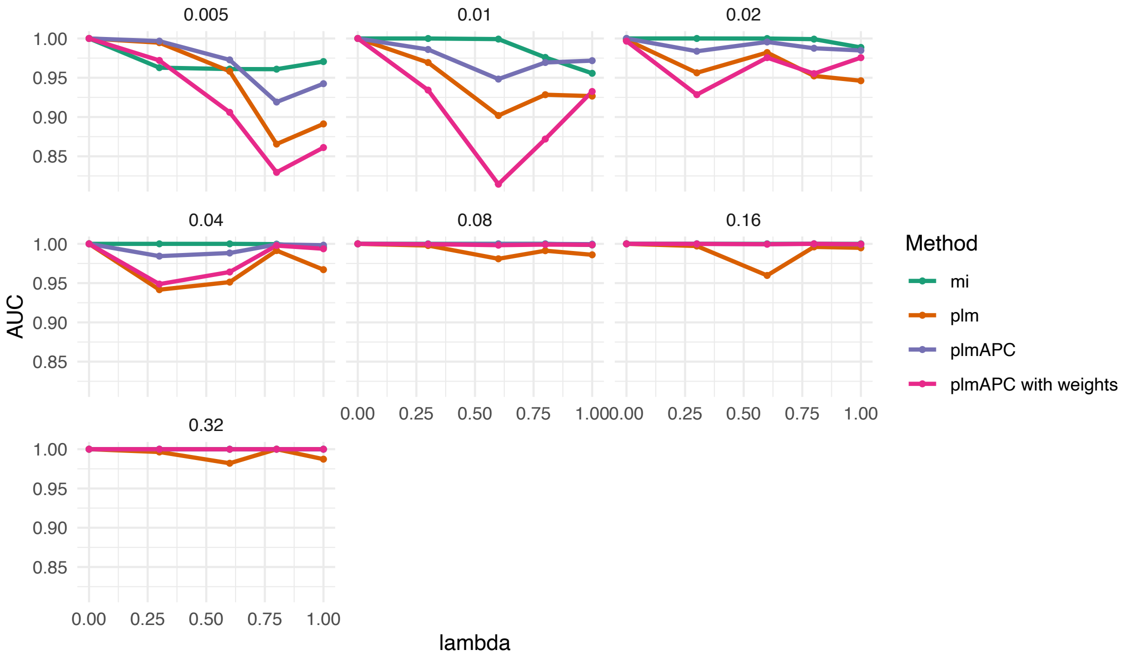
Site 1	Site 2	Delta AIC	s/d ratio	Coev
9	10	223.30	0.01176	1
3	4	207.12	0.01176	1
7	8	192.17	0.01176	1
5	6	92.92	0.01176	1
10	11	82.96	0.01176	0
15	16	50.65	0.01176	1
13	14	50.65	0.01176	1
19	20	34.39	0.01176	1
1	2	33.53	0.01176	1
11	12	33.53	0.01176	1
17	18	11.01	0.011765	1
34	35	3.83	0.011765	0
43	44	3.29	0.011765	0

Supp.Table-1.3 Coev results: top 15 values one simulation and mean branch length 0.005

6.5. plmDCA results with 500 species



Supp.Fig-1.11 AUROC precision recall for different λ and mean branch length with Coev and LG. Colors show different methods used. MI is the Mutual Information without the APC correction but normalized with the pair entropy.



Supp.Fig-1.12 AUROC precision recall for different λ and mean branch length with Coev and CAT. Colors show different methods used. MI is the Mutual Information without the APC correction but normalized with the pair entropy.

REFERENCES

Ackerman, S. H., Tillier, E. R., & Gatti, D. L. (2012). Accurate Simulation and Detection of Coevolution Signals in Multiple Sequence Alignments. *PLoS ONE*, *7*(10). https://doi.org/10.1371/journal.pone.0047108

Baussand, J., & Carbone, A. (2009). A combinatorial approach to detect coevolved amino acid networks in protein families of variable divergence. *PLoS Computational Biology*, *5*(9). https://doi.org/10.1371/journal.pcbi.1000488

Blomberg, S. P., Garland, T., & Ives, A. R. (2003). Testing for phylogenetic signal in comparative data: Behavioral traits are more labile. *Evolution*, *57*(4), 717–745. https://doi.org/10.1111/j.0014-3820.2003.tb00285.x

Burger, L., & Van Nimwegen, E. (2010). Disentangling direct from indirect co-evolution of residues in protein alignments. *PLoS Computational Biology*, *6*(1), e1000633.

Caporaso, J. G., Smit, S., Easton, B. C., Hunter, L., Huttley, G. A., & Knight, R. (2008). Detecting coevolution without phylogenetic trees? Tree-ignorant metrics of coevolution perform as well as tree-aware metrics. *BMC Evolutionary Biology*, *8*(1), 1–25. https://doi.org/10.1186/1471-2148-8-327

Carmona, D., Fitzpatrick, C. R., & Johnson, M. T. J. (2015). Fifty years of co-evolution and beyond: integrating co-evolution from molecules to species. *Molecular Ecology*, *24*(21), 5315–5329. https://doi.org/10.1111/MEC.13389

Chiu, D. K. Y., & Kolodziejczak, T. (1991). Inferring consensus structure from nucleic acid sequences. *Bioinformatics*, *7*(3), 347–352.

De Juan, D., Pazos, F., & Valencia, A. (2013). Emerging methods in protein co-evolution. *Nature Reviews Genetics*, *14*(4), 249–261. https://doi.org/10.1038/nrg3414

Dib, L., Salamin, N., & Gfeller, D. (2018). Polymorphic sites preferentially avoid co-evolving residues in MHC class I proteins. *PLoS Computational Biology*, *14*(5), 1–19. https://doi.org/10.1371/journal.pcbi.1006188

Dib, L., Silvestro, D., & Salamin, N. (2014). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dunn, S. D., Wahl, L. M., & Gloor, G. B. (2008). Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, *24*(3), 333–340. https://doi.org/10.1093/bioinformatics/btm604

Dutheil, J., & Galtier, N. (2007). Detecting groups of coevolving positions in a molecule: A clustering approach. *BMC Evolutionary Biology*, *7*(1), 1–18. https://doi.org/10.1186/1471-2148-7-242

Dutheil, J., Pupko, T., Jean-Marie, A., & Galtier, N. (2005). A model-based approach for detecting coevolving positions in a molecule. *Molecular Biology and Evolution*, *22*(9), 1919–1928. https://doi.org/10.1093/molbev/msi183

Dutheil, J. Y. (2012). Detecting coevolving positions in a molecule: Why and how to account for phylogeny. *Briefings in Bioinformatics*, *13*(2), 228–243. https://doi.org/10.1093/bib/bbr048

Dutheil, J. Y., Jossinet, F., & Westhof, E. (2010). Base pairing constraints drive structural epistasis in ribosomal RNA sequences. *Molecular Biology and Evolution*, *27*(8), 1868–1876. https://doi.org/10.1093/molbev/msq069

Ekeberg, M., Hartonen, T., & Aurell, E. (2014). Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences. *Journal of Computational Physics*, *276*, 341–356. https://doi.org/10.1016/j.jcp.2014.07.024

Guindon, S., Dufayard, J. F., Lefort, V., Anisimova, M., Hordijk, W., & Gascuel, O. (2010). New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology*, *59*(3), 307–321. https://doi.org/10.1093/SYSBIO/SYQ010

Hockenberry, A. J., & Wilke, C. O. (2019). Phylogenetic weighting does little to improve the accuracy of evolutionary coupling analyses. *Entropy*, *21*(10), 19–22. https://doi.org/10.3390/e21101000

Hopf, T. A., Morinaga, S., Ihara, S., Touhara, K., Marks, D. S., & Benton, R. (2015). Amino acid coevolution reveals three-dimensional structure and functional domains of insect odorant receptors. *Nature Communications*, *6*(1), 1–7.

Horta1, E. R., & Weigt, M. (2021). On the effect of phylogenetic correlations in coevolution-based contact prediction in proteins. *PLoS Computational Biology*, *17*(5), 1–17. https://doi.org/10.1371/journal.pcbi.1008957

Jones, D. T., Buchan, D. W. A., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, *28*(2), 184–190. https://doi.org/10.1093/bioinformatics/btr638

Jones, D. T., Singh, T., Kosciolek, T., & Tetchner, S. (2015). MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, *31*(7), 999–1006.

Kamisetty, H., Ovchinnikov, S., & Baker, D. (2013). Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(39), 15674–15679. https://doi.org/10.1073/PNAS.1314045110/-/DCSUPPLEMENTAL/SD02.XLS

Lartillot, N., Brinkmann, H., & Philippe, H. (2007). Suppression of long-branch attraction artefacts in the animal phylogeny using a site-heterogeneous model. *BMC Evolutionary Biology*, *7*(SUPPL. 1), 1–14. https://doi.org/10.1186/1471-2148-7-S1-S4/FIGURES/5

Lartillot, N., & Philippe, H. (2004). A Bayesian Mixture Model for Across-Site Heterogeneities in the Amino-Acid Replacement Process. *Molecular Biology and Evolution*, *21*(6), 1095–1109. https://doi.org/10.1093/MOLBEV/MSH112

Lartillot, N., & Philippe, H. (2006). Computing Bayes Factors Using Thermodynamic Integration. *Systematic Biology*, *55*(2), 195–207. https://doi.org/10.1080/10635150500433722

Le, S. Q., & Gascuel, O. (2008). An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution*, *25*(7), 1307–1320. https://doi.org/10.1093/MOLBEV/MSN067

M. Pagel. (1999). Inferring the historical patterns of biological evolution. *Nature*, *401*(October 1999), 877–884.

Malinverni, D., Lopez, A. J., De Los Rios, P., Hummer, G., & Barducci, A. (2017). Modeling Hsp70/Hsp40 interaction by multi-scale molecular simulations and coevolutionary sequence analysis. *ELife*, *6*. https://doi.org/10.7554/ELIFE.23471

Marmier, G., Weigt, M., & Bitbol, A. F. (2019). Phylogenetic correlations can suffice to infer protein partners from sequences. *PLoS Computational Biology*, *15*(10), 1–24. https://doi.org/10.1371/journal.pcbi.1007179

Meyer, X., Dib, L., & Salamin, N. (2019). CoevDB: A database of intramolecular coevolution among protein-coding genes of the bony vertebrates. *Nucleic Acids Research*, *47*(D1), D50–D54. https://doi.org/10.1093/nar/gky986

Meyer, X., Dib, L., Silvestro, D., & Salamin, N. (2019). Simultaneous Bayesian inference of phylogeny and molecular coevolution. *Proceedings of the National Academy of Sciences*, *116*(11), 5027–5036. https://doi.org/10.1073/pnas.1813836116

Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., Zecchina, R., Onuchic, J. N., Hwa, T., & Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, *108*(49), E1293–E1301. https://doi.org/10.1073/pnas.1111471108

Moretti, S., Laurenczy, B., Gharib, W. H., Castella, B., Kuzniar, A., Schabauer, H., Studer, R. A., Valle, M., Salamin, N., Stockinger, H., & Robinson-Rechavi, M. (2014). Selectome update: quality control and computational improvements to a database of positive selection. *Nucleic Acids Research*, *42*(D1), D917–D921. https://doi.org/10.1093/NAR/GKT1065

Muscat, M., Croce, G., Sarti, E., & Weigt, M. (2020). FilterDCA: Interpretable supervised contact prediction using inter-domain coevolution. *PLoS Computational Biology*, *16*(10), 1–19. https://doi.org/10.1371/journal.pcbi.1007621

Pennell, M. W., Eastman, J. M., Slater, G. J., Brown, J. W., Uyeda, J. C., Fitzjohn, R. G., Alfaro, M. E., & Harmon, L. J. (2014). geiger v2.0: an expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. *Bioinformatics*, *30*(15), 2216–2218. https://doi.org/10.1093/BIOINFORMATICS/BTU181

Priya, P., & Shanker, A. (2021). Coevolutionary forces shaping the fitness of SARS-CoV-2 spike glycoprotein against human receptor ACE2. *Infection, Genetics and Evolution*, *87*(November 2020), 104646. https://doi.org/10.1016/j.meegid.2020.104646

Proux, E., Studer, R. A., Moretti, S., & Robinson-Rechavi, M. (2009). Selectome: a database of positive selection. *Nucleic Acids Research*, *37*(suppl_1), D404–D407. https://doi.org/10.1093/NAR/GKN768

Qin, C., & Colwell, L. J. (2018). Power law tails in phylogenetic systems. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(4), 690–695. https://doi.org/10.1073/pnas.1711913115

Quang, L. S., Gascuel, O., & Lartillot, N. (2008). Empirical profile mixture models for phylogenetic reconstruction. *Bioinformatics*, *24*(20), 2317–2323. https://doi.org/10.1093/BIOINFORMATICS/BTN445

Revell, L. J. (2012). phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, *3*(2), 217–223. https://doi.org/10.1111/j.2041-210X.2011.00169.x

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., & Bridgland, A. (2019). Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction

(CASP13). *Proteins: Structure, Function, and Bioinformatics*, *87*(12), 1141–1148.

Uguzzoni, G., Lovis, S. J., Oteri, F., Schug, A., Szurmant, H., & Weigt, M. (2017). Large-scale identification of coevolution signals across homo-oligomeric protein interfaces by direct coupling analysis. *Proceedings of the National Academy of Sciences of the United States of America*, *114*(13), E2662–E2671. https://doi.org/10.1073/pnas.1615068114

Weigt, M., White, R. A., Szurmant, H., Hoch, J. A., & Hwa, T. (2009). *PNAS-2009-Weigt-67-72*. *106*(1). https://doi.org/10.1073/pnas.0805923106

Yamada, K., Davydov, I. I., Besnard, G., & Salamin, N. (2019). Duplication history and molecular evolution of the rbcS multigene family in angiosperms. *Journal of Experimental Botany*, *70*(21), 6127–6139.

Yeang, C. H., & Haussler, D. (2007). Detecting coevolution in and among protein domains. *PLoS Computational Biology*, *3*(11), 2122–2134. https://doi.org/10.1371/journal.pcbi.0030211

Zerihun, M. B., & Schug, A. (2018). RNA Structure Prediction Guided by Coevolutionary Information. *Biophysical Journal*, *114*(3), 436a.

Chapter 2

Supervised Deep Learning to infer coevolution under the light of evolution

*"Your brain does not manufacture thoughts.
Your thoughts shape neural networks."*

Deepak Chopra

Supervised Deep Learning to infer coevolution under the light of evolution

ABSTRACT

Coevolution can be seen as a molecular evolutionary process where reciprocal evolutionary changes occur between molecules (amino acids or nucleotides) as they depend on each other. We transform each site of a given MSA into a vector representing the substitutions occurring along each branch of their phylogenetic tree, obtaining a matrix similar to an image with its pixels where we can visualize the changes co-occurring between two sites of the MSA and detect if sites are coevolving. Due to the advances in machine learning and that Convolutional Neural Networks (CNNs) are a popular model used to detect patterns in bidimensional matrices (usually for image classification), in this study we introduce a model with an architecture based on two 1D-asymmetrical separable convolutions to detect the signature of coevolution. The model is trained and tested under different simulated genomic datasets varying the levels of divergence and the number of sequences (in total 18000 MSAs with their phylogenetic trees), containing 2 distinct labeled classes: is there any signature of coevolution or not.

1. INTRODUCTION

The use of machine learning techniques is nowadays becoming ubiquitous in many areas of science, providing effective solutions to problems that were difficult to tackle in very different fields of research. In biology, these techniques have been used for decades to classify, predict, or do regression with high dimensional data (e.g., Decision Tree, Random Forest, Support Vector Machine). Nevertheless, new approaches like Deep Learning, which have seen a surge in their applications in scientific areas such as in physics (Z.-K. Liu et al., 2022), language recognition (Wen et al., 2021) or medicine (Heo et al., 2022), have only recently been applied to biological problems. The developments have, however, a wide range of applications in biology, from estimating species extinction risk (Zizka et al., 2022), identifying diseases (J. Liu et al., 2021), classifying metagenomes (Manning et al., 2019), inferring hybridization between organisms (Blischak et al., 2021), or predicting the structure of proteins (Jumper et al., 2021).

Deep Learning is a subset of machine learning based on multiple layers of Artificial Neural Networks. One of its popular algorithms is Recurrent Neural Networks (RNNs), a type of Artificial Neural Network that contains feedback loops allowing to store information within the network (Goodfellow et al., 2016). RNNs are trained to recognize the sequential characteristics of data and they are very effective for text classification. Its architecture can be seen as a graph with directed cycles in memory, allowing to learn information related to the past. RNN have been used to predict the mutations of influenza A viruses (Yin et al., 2020) or to infer the genome-wide map of per-base recombination rates from polymorphism data (Adrion et al., 2019). However, one of the most popular Deep Learning methods is Convolutional Neural Networks (CNNs), which are also a type of Artificial Neural Networks. They typically have three main layers: a convolutional layer in which features/patterns are extracted from input data, a pooling layer to reduce the size of the data while preserving the critical features, and a fully connected layer where all the outputs from the previous layer are connected, and an output/prediction is provided (Goodfellow et al., 2016). CNNs are trained with bidimensional matrices (usually an image), and they are very effective in artificial vision problems. They have proved to be a powerful methodology for detecting patterns and classifying according to them. Even though it started as a method to classify images (Lecun et al., 1998; Krizhevsky et al., 2012), it has been expanding its applications to other non-image problems (Abdel-Hamid et al., 2014; Zhang & LeCun, 2015). CNNs are however always used in situations where the problem is translation invariant, the data has spatial features and its detection is key for the prediction/classification of the data (Goodfellow et al., 2016). Moreover, it is possible to visualize the feature layers to better understand the data and the classification decision (Zeiler & Fergus, 2014). CNNs have been used to predict host phenotype from metagenomic data (Reiman et al., 2020) or to calculate the evolutionary distances (Z. Liu et al., 2020), outperforming traditional methods.

The application of deep learning in molecular evolution have so far been limited. CNN have been used to build phylogenetic trees by inferring quartets from sequence data (Zou et al., 2020), but there has been no attempt to use the pattern recognition of CNN to learn about the processes driving molecular evolution. In this context, an essential mechanism in molecular evolution are the joint substitutions occurring dependently at two amino-acid sites of a protein. The detection of such coevolution between sites within molecular sequences can help predict amino acids that are in contact in the protein 3D structure (Morcos et al., 2011; Ekeberg et al., 2012; De Juan et al., 2013), better define the structure of proteins (Kamisetty et al., 2013; Hopf et al., 2015; Mehari Bayou Zerihun & Schug, 2018; Senior et al., 2019), identify interfaces between proteins (Ackerman et al., 2012; Uguzzoni et al., 2017), or detect interactions with other proteins (Malinverni et al., 2017).

Several methods have been developed to predict sites under coevolution (Dunn et al., 2008; Ekeberg et al., 2012; Jones et al., 2012; Kamisetty et al., 2013). These methods take a multiple sequence alignment (MSA) as an input and predict pairs of sites in contact using a statistical model that looks at the covariation of the frequency pattern of amino acids between sites. The assumption is that the covariation between sites is due to structural constraints maintaining the interactions between sites that are in close proximity in the 3D structure of the protein (Qin & Colwell, 2018).

Recently, there has been an increase in the use of Machine Learning techniques to infer pairs of amino-acid sites in contact, but all of them took as input-features some measures of coevolution between sites. Restricted Boltzmann machine (Adhikari et al., 2018) or the popular CNN (RaptorX, DeepMetaPSICOV, Filter-DCA; Källberg et al., 2012; Kandathil et al., 2019; Muscat et al., 2020) are some of these techniques. In the case of Filter-DCA, it uses the predicted standard DCA scores as an input for a CNN which utilizes "structural filters", created from the average contact patterns, to predict sites in contact. Even though these methods have proven their success, the coevolving sites that they use as an input may not be only close in the 3D structure and it can bias their 3D structure prediction (Mehari B. Zerihun & Schug, 2017) due to all the top pair of sites inferred are not necessary in contact in the 3D structure. Further, the covariance structure used to detect coevolution between sites can also arise from evolutionary processes and it is important to account for the underlying phylogenetic tree that represents the evolutionary history of the sequence at hand (Dutheil, 2012; **Chapter 1**).

Accounting for phylogenetic relationships to predict coevolution between sites in a MSA requires modeling the joint substitution process of two sites along the branches of the tree. This is important because the same pattern of amino-acid frequencies in an MSA can be explained differently depending on the underlying evolutionary history of the sequences. This is shown, for instance, in Fig-2.1. It illustrates that coevolution between a pair of sites predicted from a MSA could be due to be a simple random change in the protein sequence when including phylogenetic information (Fig-2.1 C left tree) which reduces the validity of interpreting this as coevolution (Dutheil, 2012); or if the changes have co-occurred multiple times (Fig-2.1 C right tree) through the phylogeny, which increases the validity of interpreting this as coevolution, having a stronger signal. One approach to account for the effect of the evolutionary relationships, taken by the method implemented in CoMap (Dutheil et al., 2005), is to infer the pattern of evolution through the stochastic mapping of the substitution events occurring for each site along each branch of the phylogenetic tree. It then uses the correlation between the vectors of site changes per branch to estimate coevolution between pairs of sites. Other approaches, like Coev (Dib et al., 2014), use Maximum Likelihood estimation of a Markov model to study the process of evolution along a phylogenetic tree for any pairs of sites based on a substitution matrix that describes the transitions between positions, along the branches, that change in a coordinated way during sequence evolution.

The latter approaches are computationally expensive and can only with difficulty be applied to large scale genomic data (Meyer et al. 2019). In contrast, CNN have been applied to large scale data and based on the demonstrated advantages of using CNN, in combination with the fact that the pattern of coevolution present in our data is translation invariant, we propose a novel model, Coev-Asymmetric-CNN, based on a CNN to detect the signal of coevolution in an MSA while incorporating the phylogenetic tree associated with the sequences. Our model detects a coevolution signal when co-substitutions are happening between at least two different sites and does not detect a coevolution signal when there are no co-substitutions patterns. Another aspect of Coev-Asymmetric-CNN is that it can highlight the branches where the co-substitutions are happening, profiting from the activation maps derived from the trained CNN. We first describe the model that we developed before using simulated datasets to assess the properties of our approach and test its performance. Moreover, we explore which sites are under coevolution thanks to the activation maps of the CNN.

2. MATERIALS AND METHODS

We first describe the different steps to transform the data, composed of an MSA and a phylogenetic tree, into a 2D matrix representing the number of substitutions occurring at each site along each branch of the tree. The 2D matrix is then used as an input to the CNN architecture. Afterward, we describe the CNN model and the procedure to train our model.

2.1. Representing substitutions in a 2D matrix

We transform each column of the MSA into a vector representing the substitutions occurring along each branch of the phylogenetic tree. This is done by estimating the likelihood of the ancestral states at each node of the phylogenetic tree using the R library *phangorn* (Schliep, 2011). We applied a simple substitution model assuming equal probabilities for each state and equal substitution rates. The rates were, optimized, but we fixed the length of each branch to the value given in input (options *bf=NULL, optQ=T, optRate=T, optEdge=F* in phangorn). This could be modified to fit any type of model of amino-acid evolution. Once the ancestral states were estimated, we mapped the changes (or no changes) occurring along each branch n ($n \in [1, B]$) to obtain a vector of length B. This was done for all sites L and the final input matrix M_{LxB} (L = sequence length, B = number of branches) had values $m_{i,n}$ that were the types of substitutions occurring for site *i* on branch n (Fig-2.1). For amino acid datasets, which contain 20 different states, there are a total of 380 possible substitution types represented by the $m_{i,n}$ values and the value 0, which indicates no change (Fig-2.1 A, with the left phylogeny in Fig-2.1 C, where there is no signal of coevolution; Fig-2.1 D, with the right phylogeny in Fig-2.1 C where there is signal of coevolution).

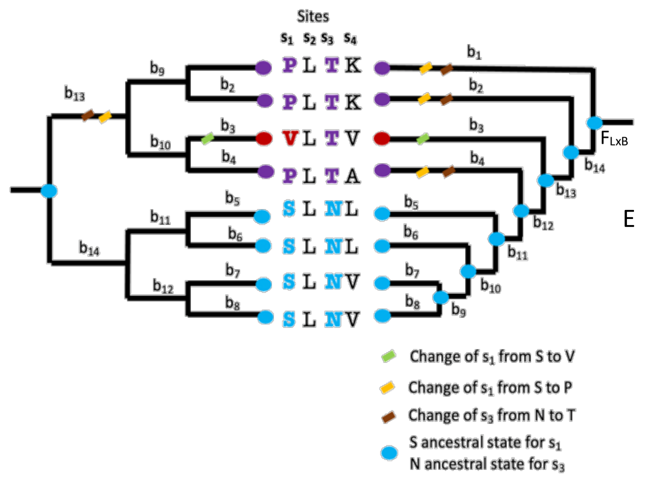
A

$$M_{LxB} = \begin{array}{c|cccccccccccccc} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ \hline s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ s_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ s_4 & 0 & 0 & 32 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 46 & 0 & 0 \end{array}$$

$$F_{LxB} = \begin{array}{c|cccccccccccccc} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ \hline s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ s_4 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

B

C



D

$$M_{LxB} = \begin{array}{c|cccccccccccccc} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ \hline s_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_3 & 5 & 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_4 & 0 & 0 & 32 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 46 & 0 & 0 \end{array}$$

$$\begin{array}{c|cccccccccccccc} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ \hline s_1 & 4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_3 & 4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_4 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

E

Fig-2.1 A) Matrix of changes M_{LxB}, for the MSA and the left phylogenetic tree, where each value is a type of change. B) Frequency changes matrix, F_{LxB}, for the MSA and the left phylogenetic tree. C) MSA with two different possible phylogenetic trees, in the left with only one change happening deep in the tree, and in the right, where multiples changes have occurred. D) Matrix of changes M_{LxB}, for the MSA and the right phylogenetic tree, where each value is a type of change. E) Frequency changes matrix, F_{LxB}, for the MSA and the right phylogenetic tree.

In the event of coevolution due to evolutionary processes, we would expect substitutions co-occurring across pairs of sites (co-substitutions) repeatedly along multiple branches (Dutheil, 2012; Dib et al., 2014), generating a pattern in the matrix M_{LxB}. Such a pattern is visible when two specific substitution types at sites *i* and *j* (e.g., in Fig-2.1 right tree, substitution S -> P at site i=1 and substitution N-> T at site j=3) appear together in multiple branches (e.g., b_1, b_2, b_3 and b_4) of the phylogenetic tree. The number of occurrences of these co-substitutions is important to detect coevolving pairs of sites because the signal of coevolution will be stronger if the number of these occurrences is high. We transformed the matrix M_{LxB} into a matrix F_{LxB} by counting the number of times each substitution type occurred at the same site. The values $f_{i,n}$ of this F matrix are the number of times the type of change encoded in $m_{i,n}$ has occurred for site *i* through the phylogenetic tree (Fig-2.1 B; Fig-2.1 E) For instance, since substitution 1 occurred 4 times at site s_1, we substituted every $m_{1,j}$ =1 into $f_{1,j}$=4. The transformation has the advantage of increasing the signal of coevolution in the data. It also removes the issue that substitution types are encoded with integer values when they represent, in fact, categorical variables. This would bias the CNN model if not accounted for.

2.2. Convolutional Neural Network Model

We proposed a CNN (hereafter referred to as Coev-Asymmetric-CNN) to learn the pattern of coevolution present in the F_{LxB} matrix. This pattern can vary between rows and columns depending on the MSA and phylogenetic tree, although the coevolution process will create a defined structure linking some rows and columns as described above (Fig-2.1). It is essential that the filters used in the CNN do not break this data structure. We therefore implemented Coev-Asymmetric-CNN using two 1D convolutional layers, based on the idea of spatial separable convolutions (Králik & Ladányi, 2021), sometimes also called asymmetric convolutions (S. Y. Lo et al., 2018). The data structure is maintained in spatial separable convolutions because the kernel is split across its spatial axes, which are, in our case, the sites and branches represented in the rows and columns of the F matrix, respectively. A typical 2D kernel K, for instance of size LxB, will then be separated into two smaller kernels C (eqn. 1) that can be applied sequentially to the input by convolution (eqn. 2), obtaining the same effect than the original 2D kernel K_{LxB} (Králik & Ladányi, 2021). The 2D kernel is obtained by matrix multiplication of the two 1D vectors (eqn. 1), but the 1D vectors are applied to the F_{LxB} matrix using two convolutions (eqn. 2; we use the symbol * for the convolution).

The network learns the filters C_{Lx1} and C_{1xB}, such that:

$$K_{LxB} = C_{Lx1} \times C_{1xB} \qquad (1)$$

Given an input matrix F_{LxB} and these two filters, the output of the two convolutions will be a scalar s:

$$F_{LxB} * K_{LxB} = (F_{LxB} * C_{Lx1}) * C_{1xB} = \mathrm{s} \qquad (2)$$

The final scalar *s* that is obtained gives the strength of the signal of coevolution present in the data F given the two filters C that were defined. The key is then to let the CNN learn the patterns in these filters by providing appropriate training datasets (see below for details).

In summary, our Coev-Asymmetric-CNN model consisted of one input layer of size LxB with a single channel, two asymmetric convolutional layers of size Lx1 and 1xB containing each 400 filters and a fully connected layer (Fig-2.2) with two outputs values corresponding to a final prediction of coevolution or not. It resulted in a network containing 32,689 parameters to train (weights and biases). After each convolution layer, we used a ReLU nonlinearity activation function (Schmidhuber, 2015). The results from the second convolution were then flattened and fed into the fully connected network. We further used a dropout (Srivastava et al., 2014) to help reduce overfitting of the CNN and a LogSoftMax activation function was applied to the output for class prediction. We did not use pooling in our model since co-substitutions occurring across sites at several branches have a positional dependency in the F matrix representing the process of coevolution. Pooling would have given the same prediction score for any shuffled input matrices, although those would no longer have represented the coevolution pattern that we were looking for. Finally, we did not normalize the input data, because our F_{LxB} matrix is sparse. Most sites do not change over most branches, especially for slowly evolving amino-acid sequences. A normalization step would make the values associated with the true substitutions very small and similar to each other rendering the signal of coevolution fuzzy and impossible to detect.

The Coev-Asymmetric-CNN was implemented using Pytorch (Paszke et al., 2019). We trained it using an Adam optimizer (Kingma & Ba, 2015), a learning rate of 0.008 and a cross-entropy loss function. We applied mini-batches of size 64 and the convolution was performed with a stride equal to 1 and a padding of 0. The dropout in the fully connected layer was set to 0.25.

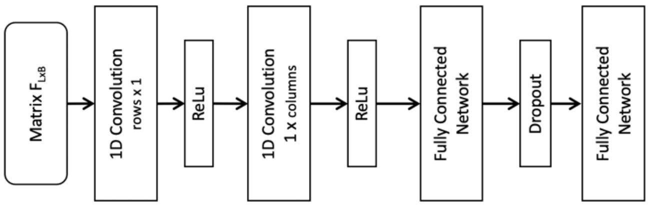


Fig-2.2 Example network architecture for Coev-Asymmetric-CNN

2.3. Activation maps

After the CNN has been trained, we can input a F matrix to our CNN to obtain a prediction score defining if the MSA contained any signal of coevolution or not. If coevolution is detected by the CNN, it is important to get further information about which are the sites that may be under coevolution. Thus, we proposed a pipeline to identify the most probable sites to be under coevolution given a F input matrix, using the activation maps from the second convolutional layer (Fig-2.3).

First, we repeated the input F matrix as many times as the number of filters used in the second convolutional layer. In our case, there were 400 filters in the second convolutional layer and the input F matrix was repeated 400 times. At the end, the new input matrix had, in our case, a dimension of *400 x 100 x 298*. We then applied the second convolutional layer to this new input matrix (F matrix repeated 400 times) to obtain the activation maps of size *400 x 100 x 1*. Each of the 400 activation maps (also known as feature maps) correspond to the activation of the different sites from the input F matrix. We averaged the 400 activation maps to obtain a single final averaged activation map of size *100 x 1*. Finally, to identify the sites that were responsible for the strong activations and thus the signal of coevolution, we ranked the 100 sites based on the averaged activation value and selected those with higher values to represent the sites more likely to be under coevolution.

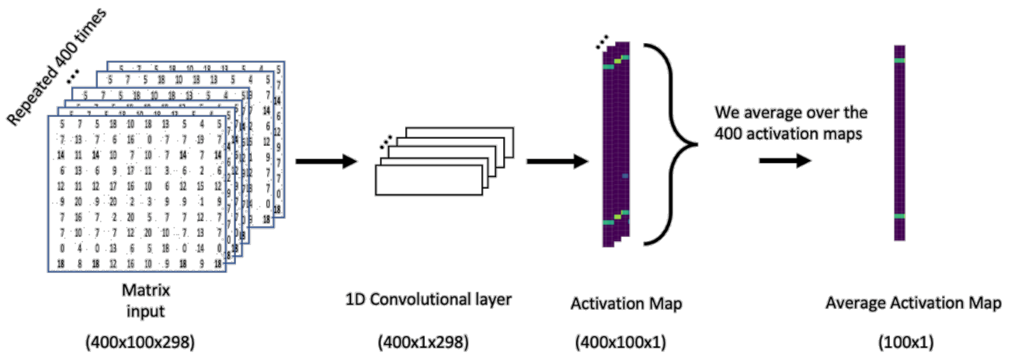


Fig-2.3 Pipeline to rank the sites to get the top most probable sites under coevolution. First we have the input matrix of size 400x100x298, (the 2D F matrix repeated 400 times), then, we applied the 400 learnt filters of size 1x298 to each dimension of our input. After that, we get the activation map of size 400x100x1 (each of the 400 dimensions correspond to one specific filter from the previous step). Finally, we average over the 400 activation maps to get the average activation map of size 100x1. This last output is a vector where the 100 values correspond to the activation value of the 100 sites.

2.4. Datasets and simulations

We used simulations to assess the performance of our Coev-Asymmetric-CNN and tested its performance while varying the number of sequences and the level of sequence divergence. All the datasets that we used to train or test the CNN were fully balanced. Half of the samples were simulated with a certain number of pairs of sites coevolving and the other half with alignments having only independent sites. The total number of datasets used in each case is described below.

We selected different levels of divergence that were realistic for amino acid sequences of protein-coding genes by taking alignments from the Selectome database (Proux et al., 2009). We used the Euteleostomi data, containing around 10,000 MSAs and their corresponding phylogenetic trees as a representative dataset. We estimated the mean branch lengths (i.e., total branch lengths of the maximum likelihood tree divided by the number of branches in the tree) for each MSA, which showed a median value of 0.04 and a 95% percentile covering the values from 0.009 to 0.12. Based on this data, we defined six levels of divergence, expressed as the mean branch lengths in number of expected substitutions, in our simulations (0.005, 0.01, 0.02, 0.04, 0.08 and 0.16), to perform our simulations. We also explored different sizes of datasets by including phylogenetic trees with 150, 100 and 50 sequences, which led to a combination of 18 sets of simulations (three tree sizes times six levels of divergence).

Our simulations started by creating random phylogenetic trees for each tree size (i.e., 150, 100 and 50 sequences). We accounted for topological variation by creating for each size 1,000 random phylogenetic trees using the R function pbtree from the phytools library (Revell, 2012). The birth and death parameters were set to 1 and 0, respectively. The trees were all binary, rooted and ultrametric. We transformed the branch lengths of each of these trees by drawing random values from an exponential distribution with a scale value given by the six levels of divergence described above (i.e., a scale β, which gave the mean of the exponential distribution as $1/\beta$). We assigned a new branch length to each edge of the simulated trees drawn from this distribution, therefore scaling the trees accordingly.

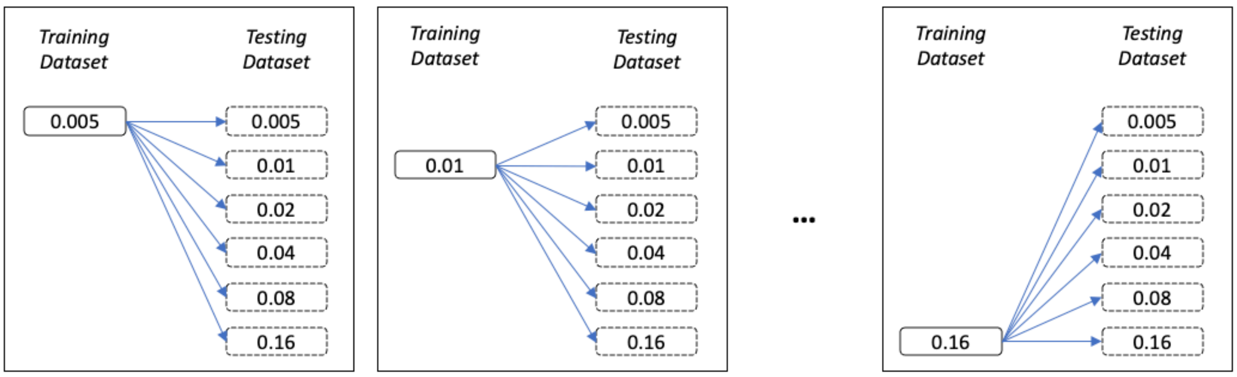
We simulated an MSA of 100 amino acids in length on each scaled phylogenetic tree, which led to 1,000 pairs of simulated MSA and phylogenetic trees. We used a balanced training dataset containing half of the datasets created with some sites of the MSA evolving under coevolution and the other half with sites evolving fully under an independent model of evolution. For the MSA without coevolution signal, we generated a set of 100 independent amino acids that evolved under the standard LG model (Le & Gascuel, 2008). We used a homogeneous matrix of rate transition with all substitutions being identical and we assumed that each amino acid had the same equilibrium frequency. We included a Gamma distribution to represent the variation in the rate of evolution of each site with an alpha parameter set to 4. For the MSA with signal of coevolution, we partitioned the MSA into two parts: i) a set of *x* independent sites that evolved under the LG model as above; ii) a set of *y* pairs of sites, with *y* varying between samples according to a Poisson distribution with a mean of 4, that evolved under a model of coevolution (if the value drawn from the distribution was equal 0, we changed it to 1, to always have at least one pair of sites under coevolution). We used the Coev model to simulate these *y* pairs of sites (Dib et al., 2014) and each pair was created by drawing at random a coevolution profile and setting the *d* and *s* parameters that govern the rate of change in the coevolution model to 100 and 1, respectively. Therefore, if we have an alignment with a sequence length of 100 amino acids and $y=5$ pairs under coevolution, we will have $x = 100 - y * 2 = 90$ independent sites evolving under the LG model. All simulated datasets were transformed into a frequency matrix F_{LxB} as described above.

2.4.1. Training and testing datasets

We assessed the performance of our Coev-Asymmetric-CNN by training and testing it on different types of datasets based on the simulations that we described above. We describe here the three cases that we investigated.

Case A. We studied the performance of our model to predict MSA under coevolution by first looking at the effect of sequence divergence while fixing the number of sequences in the data. We studied it under two different scenarios. For each scenario, we used 1,000 datasets that were randomly split into 70% for training, 10% for validation and 20% for testing. The input order of each training dataset was shuffled to ensure that the matrices were processed randomly by the CNN.

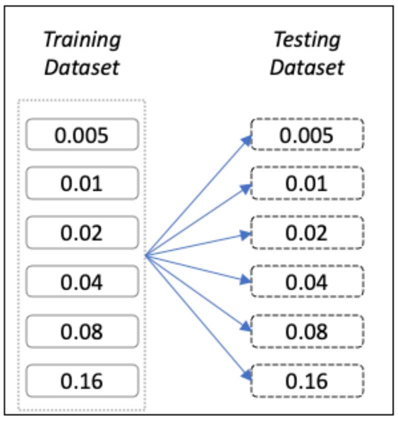
In the first scenario, we used the datasets with 100 sequences and we trained the model under six levels of sequence divergence separately (Fig-2.4). Each dataset used in the training phase contained 400 MSA with coevolution and 400 MSA without coevolution. We thus obtained 6 different trained CNNs, one for each level of sequence divergence. We then measured the performance of our approach by applying each of the six CNNs to a testing dataset containing 100 MSAs with and 100 MSAs without coevolution simulated under each level of divergence. We repeated the same procedure to train CNNs for datasets containing 150 and 50 sequences.



Mean branch length: [0.005, 0.01, 0.02, 0.04, 0.08, 0.16]

Fig-2.4 Shows how the training was done in Case A. First training over 100 sequences for a mean branch length of 0.005. A second training over 100 sequences for a mean branch length of 0.01 and so on. *The training data set is going through all the different levels of divergence.

In the second scenario, we used the datasets with 100 sequences and we trained the CNN by mixing the six levels of sequence divergence (Fig-2.4). For each level of divergence, datasets included in the training phase contained again 400 MSAs with and 400 MSAs without coevolution. In total, the training dataset contained 4,800 samples (MSA and phylogenetic trees) and we obtained a single trained CNN. We measured the performance by applying the network to a testing dataset containing 600 MSAs with and 600 MSAs without coevolution (100 MSAs for each level of divergence). We repeated the same procedure for 150 and 50 sequences.



Mean branch length: [0.005, 0.01, 0.02, 0.04, 0.08, 0.16]

Fig-2.5 Shows how the training was done in sub-case A, for 100 sequences. Training over 100 sequences mixing all divergence levels. Testing over the test dataset for 100 sequences

Case B. We then looked at the performance of a CNN trained on datasets of a given size, when applied to datasets of either smaller or larger sizes (Fig-2.6). We started by training our CNN on datasets composed of 100 sequences under each specific divergence level (400 MSAs used for training each time) and tested its performance over the same level of divergence for datasets of 50 and 150 sequences (100 MSAs used for testing each time). We repeated the same procedure for 150 and 50 sequences.

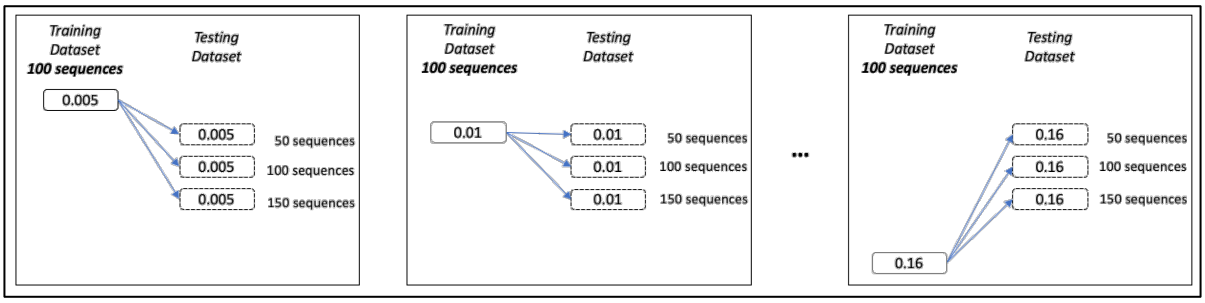


Fig-2.6 Shows how the training was done in Case B. First training with a specific divergences level for 100 sequences and tested over 100, 50 and 150 sequences for that divergence level. *The training data set is going through all the different divergence levels.

Case C. We trained our CNN by equally mixing all datasets of 150, 100 and 50 sequences at all divergence levels (Fig-2.7). The total size of the training datasets was 14,400 MSAs (800 MSAs for each size and level of divergence, half of them with and half of them without coevolution). The total size of the testing datasets was 3,600 MSAs (200 MSAs for each size and level of divergence, half of them with coevolution and half of them without coevolution).

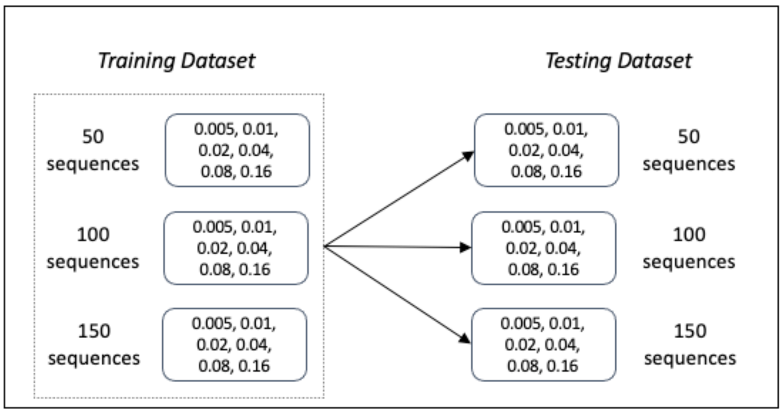


Fig-2.7 Shows how the training was done in Case C. Training dataset containing all levels of divergence (mean branch length of 0.005, 0.01, 0.02, 0.04, 0.08, 0.16) and all number of sequences (50, 100, 150). Testing dataset with all data, differentiating between the number of sequences: 50, 100 and 150 sequences and all levels of divergence.

In both cases B and C, the input data is not necessarily of the same size as the built CNN. In such situations, it is common in applications of CNN to use padding (Hashemi, 2019) to ensure a comparable size between each dataset. We used padding for the Coev-Asymmetric-CNN by adding zeros values after the last column until the input size of the smaller matrices corresponded to the largest one used (here 150 sequences, 298 branches/columns; Fig-2.8).

		Matrix	
Number sequences	**Sequence length**	**L**	**B**
150	*100*	*100*	*298*
100	100	100	198
50	100	100	98

Table-2.1 Matrix dimension from the F matrices. The final matrix' size to train the CNN is the biggest one, with 150 sequences, a sequence length of 100, and a matrix size of 100x298

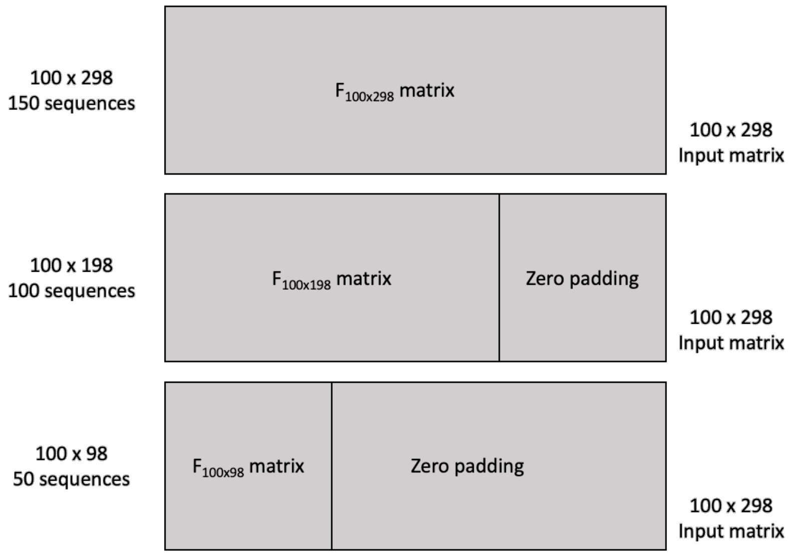


Fig-2.8 Padding F_{LxB} matrix. In the case where we have 150 sequences, the F matrix has a size of 100x298. In the case where we have 0 sequences, the F matrix has a size of 100x198, and because it is smaller than the biggest matrix, we padded it to have the same size, 100x298. In the case where we have 50 sequences, the F matrix has a size of 100x98, and we padded it too, to have the same size of the biggest matrix, 100x 298

3. RESULTS AND DISCUSSION

We developed a novel approach that used a CNN model to predict coevolution given a MSA and its associated phylogenetic tree. This model was implemented using asymmetric convolutions, without a pooling layer to maintain the strict spatial features of the data. We studied the effect of sequence divergence on the accuracy of the CNN to predict coevolution and investigated the ability of the CNN to cope with varying number of sequences. We showed that padding influenced the accuracy of the model, but that the direction of the effect depended on the size difference between the training and testing datasets. Our results indicated that our CNN could reach a high performance and that its accuracy can increase when the training dataset included a variable set of sequence divergence in the samples.

3.1. Case A: effect of sequence divergence

We assessed the performance of our CNN when the level of sequence divergence differed between the training and testing datasets. It has been shown (**Chapter 1**; (Dib et al., 2018; Dutheil, 2012)) that sequence divergence can have an effect on the accuracy of methods to detect coevolution because it affects the pattern of substitution across the sites of a MSA. We examined the behavior of our CNN with datasets containing 100 sequences, but the results were similar with the two other data sizes tested (see Supp.Fig-2.6; Supp.Fig-2.7). We trained six different CNNs, one for each level of divergence, and we applied each of these CNN on testing datasets covering all six levels of divergence (Fig-2.4). We showed that the CNN trained on the same level of divergence as the testing dataset reached a high accuracy to predict coevolution (Fig-2.9).

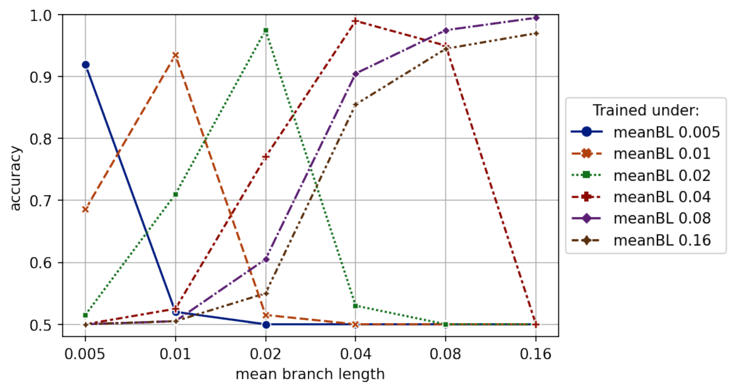


Fig-2.9 For 100 sequences. The y-axis represents the accuracy of the trained CNN on each tested dataset. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue, orange, green, red, purple and brown colors represent the trained CNN for the divergence level of 0.005, 0.01, 0.02, 0.04, 0.08 and 0.16 respectively. Each point corresponds to the accuracy of each trained CNN tested under the different levels of divergence.

The accuracy went from 0.9 for the CNN trained on a divergence of 0.005 expected number of substitutions per site to 1.00 for the CNN trained on 0.16 expected number of substitutions per site. Moreover, the intermediate levels of divergence showed an almost linear increase in accuracy between these two values (Fig-2.9). However, when a specific CNN was applied to testing datasets with either larger or smaller levels of divergence than it was trained on, the accuracy dropped sharply. The drop was more extreme with CNN trained on low levels of divergence, whereas CNN trained on higher levels of divergence showed more capabilities to predict coevolution in a wider range of testing datasets (Fig-2.9). Table-2.2 showed the distribution of false positive and false negative predictions obtained for all simulations. Although the accuracy to predict coevolution dropped when the level of divergence of the trained CNN and testing datasets diverged, the reasons for this decrease was different depending on the direction of the difference in levels of divergence. When the CNN was trained with a specific level of divergence, applying the CNN on a testing data set with higher level of divergence gave more false negatives results, whereas testing data set with lower levels of divergence gave more false positives results (Table-2.2). We looked in more details into these false negatives and positives cases and calculated, for each matrix F_{LxB}, the highest value $f_{i,j}$ found in each matrix. Fig-2.10 showed the distribution of these highest $f_{i,j}$ values for the dataset with 100 sequences. We observed a clear distinction in the frequency pattern between alignments with or without coevolution signal and the difference between the two types of datasets increased with higher levels of divergence (Fig-2.10). This pattern was maintained for datasets of 50 or 150 sequences (see Supp.Fig-2.3; Supp.Fig-2.4).

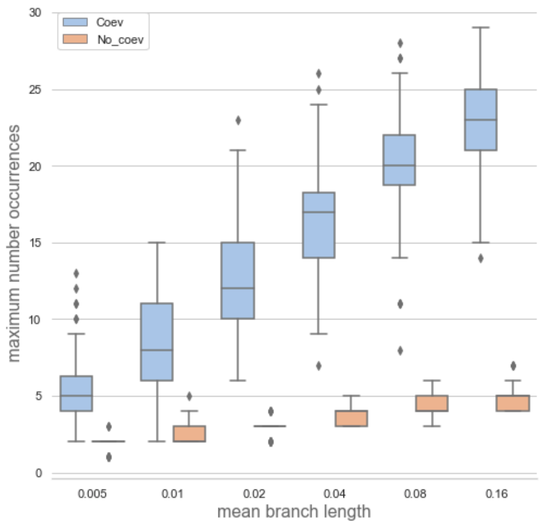


Fig-2.10 Distribution maximum number of occurrences for 100 sequences. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of divergence, from 0.005 to 0.16. At each level of divergence there is the distribution of the maximum number of occurrences for each site under coevolution in blue and for each site under non coevolution inn orange.

We related this pattern with the number of false positive and negative cases (Table-2.2) to explain the different scenarios. For instance, when the CNN was trained with low levels of divergence, it was able to learn that the number of changes expected were higher for cases where there was coevolution because most sites did not change and the F matrix was mostly filled with zeros (see Supp.Fig-2.3). Consequently, when the CNN was tested on higher level of divergence, the input F matrices displayed higher numbers of changes than the training datasets. Any change was thus detected as coevolution, leading to many false positive cases (Fig-2.9; Table-2.2). In contrast, when the CNN was trained with high levels of divergence, it learnt that coevolution was associated with high numbers of changes and that the F matrices had more changes than at lower divergence levels. Therefore, when the trained CNN was tested on lower levels of divergence, it predicted that there was no coevolution only because the number of changes in the input F matrices was low, leading to many false negatives (Fig-2.9; Table-2.2).

As a result, we saw that we overfitted the CNN if we trained it at only one specific level of divergence. It was already shown in (J. E. Lo et al., 2021) that a homogeneous training dataset may have an impact on the generalization of the model. Thus, including in the training dataset all the possible training datasets should improve the performance of the CNN (see results below), and better generalize to all possible scenarios (variability in the number of sequences and divergence).

Train under		**Mean branch length**					
		0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
0.005	**FP**	2	100	100	100	100	100
	FN	17	0	0	0	0	0
0.01	**FP**	9	0	88	100	100	100
	FN	71	17	0	0	0	0
0.02	**FP**	97	66	8	0	0	0
	FN	0	0	0	74	100	100
0.04	**FP**	0	0	0	0	1	98
	FN	100	96	53	4	1	0
0.08	**FP**	0	0	0	0	0	0
	FN	100	100	95	37	12	2
0.16	**FP**	0	0	0	0	0	0
	FN	100	100	90	30	11	5

Table-2.2 Results Case A: Only for 100 sequences. Number of false positive and false negative cases based on the level of divergence.

In the second scenario, we modified the training datasets by incorporating all levels of divergence at once to produce a single CNN (Fig-2.5). This increased the size of the training dataset and, at the same time, introduced more variability by including the full range of divergence levels. Fig-2.11 showed that this approach drastically improved the accuracy of the predictions when compared to the six CNN shown in Fig-2.9. However, at the lowest level of divergence, represented by a mean branch length of 0.005, the CNN was not able to perform very well (accuracy of 0.8; Fig-2.11) and its performance was reduced when compared to a CNN trained specifically on the data with this level of divergence. This was again due to an increase in the number of false negatives cases (Table-2.3) and it is possible that the number of false negatives is related to the level of divergence (Fig-2.10). Even though the CNN was trained on mixed levels of divergence, the $f_{i,j}$ value for low level of divergence is not big enough to be detected by the CNN. Nevertheless, the single CNN trained on mixed levels of divergence reached an accuracy close to 1 for any other levels of divergence, with a clear decrease in the number of false negative results (Table-2.3). The single CNN further did not return any false positive results, independently of the level of divergence used to simulate the testing datasets. This suggests that, in terms of performance, it is advantageous to mix datasets of varying levels of divergence. This has already been shown in typical used of CNN (Su et al., 2019; Alzubaidi et al., 2021) and is evident in our specific application. We have included all the possible variations that the pattern of coevolution may have, depending on the level of divergence, and we have increased the size of our training dataset. Such technic of data augmentation may also increase the accuracy of the CNN, and further work should be performed to better understand its effect. However, compared with the results described before (see Fig-2.9 and Fig-2.11) as well as other studies (Su et al., 2019), it seems clear that including all the possible variations will have the tendency to avoid overfitting and help to generalize the CNN. Moreover, this pattern is also found for the cases with 150 and 50 sequences (see Supp.Fig-2.6; Supp.Fig-2.7).

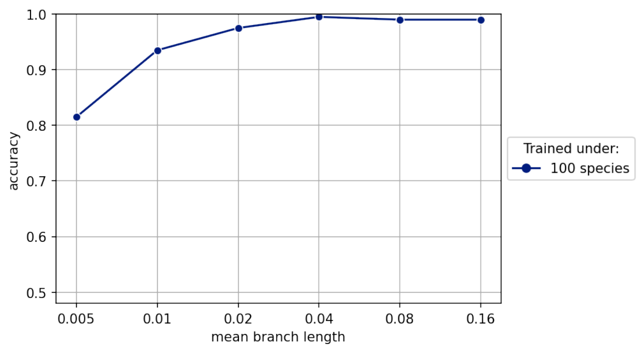


Fig-2.11 For 100 sequences. The y-axis represents the accuracy of the CNN for case A in the second scenario. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue color represents the trained CNN. Each point corresponds to the accuracy of trained CNN tested under the different levels of divergence.

	Mean Branch Length					
	0.005	**0.01**	**0.02**	**0.04**	**0.08**	**0.16**
FP	0	0	0	0	0	0
FN	37	13	5	1	2	2

Table-2.3 Case A-b Only for 100 sequences. Number of false positive and false negative cases based on the level of divergence.

3.2. Case B: effect of number of sequences

We assessed the performance of our CNN when the number of sequences differed between the training and testing datasets. This is important as the genomic data that the CNN is aimed for will vary regarding the number of sequences in each MSA and the number of sites. We examined the behavior of our CNN under 3 different scenarios by simulating datasets with either 50, 100 or 150 sequences (Fig-2.12). We trained all six levels of divergence separately for each of these number of sequences and we applied each CNN on testing datasets with the same level of divergence than the training datasets while varying the number of sequences (Fig-2.6). We showed that the CNN trained on the same level of divergence for the same number of sequences as the testing dataset has a high accuracy to predict coevolution (Fig-2.12). However, when a specifically trained CNN was applied to testing datasets under the same level of divergence with either larger or smaller number of sequences than it was trained on, the accuracy dropped (Fig-2.12). If we looked at the 6 different CNNs trained with 50 sequences, the drop in accuracy was more extreme at larger levels of divergence than at low levels of divergence where it only slightly decreased. An unexpected pattern was the accuracy found for the trained CNN with 50 sequences and a level of sequence divergence of 0.04, which had a reduced accuracy for 100 and 150 sequences compared to the other CNN (Fig-2.12 A). We trained this CNN a second time and tested it under different datasets randomly selected, but the results were similar. The issue could be the size of the training dataset, which was not very large (400 samples included), but this would need to be analyzed in more details to fully understand this pattern. For the six CNN trained on datasets containing 100 sequences (Fig-2.12 B), we observed a similar pattern as described above when the testing datasets were increased to contain 150 sequences. However, if the number of sequences is smaller in the testing than in the training datasets, the accuracy dropped independently of the level of divergence. In accordance with these results, we observed that for the 6 CNNs trained with 150 sequences, the accuracy gradually dropped when decreasing the number of sequences irrespective of the level of sequence divergence (Fig-2.12 C).

These results suggest that the number of sequences used to train the CNN had an impact on its performance, although this performance varied depending on the difference in number of sequences between the training and testing datasets and the level of divergence. For instance, when the CNN was trained with a smaller number of sequences (here 50) and was tested on a dataset with larger number of sequences, the accuracy obtained was good (between 0.8 and 0.9) at low levels of divergence. This can be explained by the fact that, at low levels of divergence (in our simulations, a mean branch length lower than 0.02), there are few changes occurring through the sites and branches. The F matrices contained therefore mostly zeros (Supp.Fig-2.5) and the consequence of padding these matrices to reach the desired larger size did not drastically alter the information. The CNN was therefore still able to detect the pattern of coevolution. However, at higher levels of divergence, padding introduced areas in the F matrices that were very dissimilar to the real part derived from the MSA and phylogenetic trees. This mismatch lead to a poor accuracy because it is highly unlikely that many zeros could occur at higher levels of divergence. In contrast, when the CNN was trained with datasets containing 150 sequences, padding the smaller testing datasets completely altered the expected data structure even at low divergence levels. The effect was stronger for the smaller datasets with 50 sequences than for the intermediate one with 100 sequences (Fig-2.11). This is probably due to the fact, that even if there were almost no changes at low levels of divergence, there were still some changes that may happen throughout the 298 branches. This pattern was not available with testing dataset with lower number of sequences because they were padded (Fig-2.7) leading to an inability of the CNN to recognize the coevolution pattern. Even though padding the smaller matrices with zeros should not be an issue for the CNN (Hashemi, 2019), we see that because the CNN was not trained including all possible scenarios (50, 100 and 150 sequences), the CNN is not able to classify correctly the samples with lower number of sequences that have been padded.

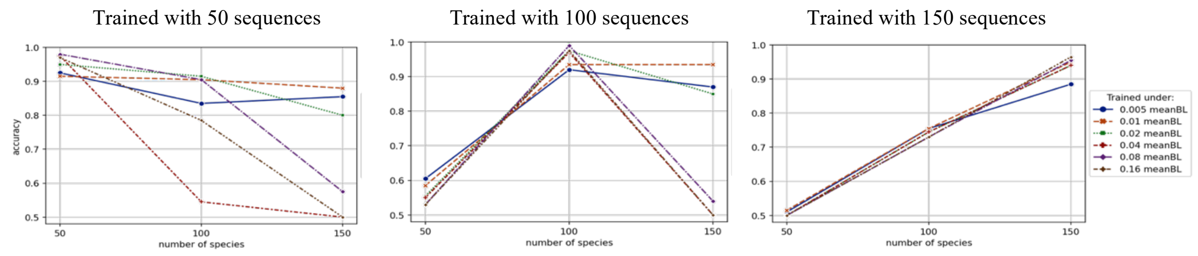


Fig-2.12 The y-axis represents the accuracy of the trained CNN on each tested dataset. The x-axis represents the different number of sequences which it was tested the CNN. The blue, orange, green, red, purple and brown colors represent the testing dataset for the mean branch length of 0.005, 0.01, 0.02, 0.04, 0.08 and 0.16 respectively.

3.3. Case C: effect of including all levels of divergence and number of sequences

In the last case, we modified the training datasets by incorporating all levels of divergence and all number of sequences at once to produce a single CNN (Fig-2.6). This model combined the two other cases previously described. The size of the training dataset was also increased and it included all the diversity of dataset sizes and levels of divergence. Fig-2.12 showed the accuracy of this single CNN when testing on datasets with different levels of divergence and for each number of sequences. We obtained a similar trend in accuracy irrespective of the number of sequences included in the testing datasets. Concerning the level of divergence, we again saw a lower accuracy at the lowest level of divergence compared to higher levels (Fig-2.12). For instance, the CNN was only able to reach an accuracy of 0.8 for the testing dataset that included 100 sequences and a mean branch length of 0.005, although this value slightly decreased for 150 and 50 sequences. The accuracy went up to 0.95 when increasing the mean branch length to 0.01 before reaching a value of almost 1.0 with increasing levels of divergence. When the level of divergence is large enough (0.02 and above), the number of sequences in the datasets did not alter the accuracy obtained (Fig-2.13). When we looked in more details at what affected the accuracy (Table-2.4), we observed that there was a high number of false negatives at low divergences, while, at high divergence, the trained model was able to correctly discriminate almost all datasets leading to very few false positives or negatives.

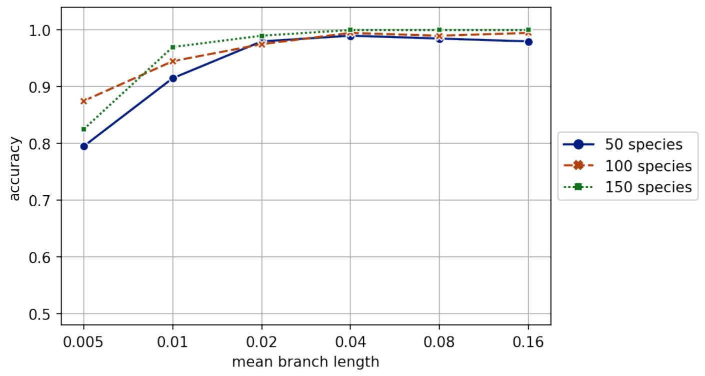


Fig-2.13 Accuracy across the levels of divergence for the testing datasets of 50, 100 and 150 sequences. The y-axis represents the accuracy of the trained CNN on each tested dataset. The x-axis represents the different levels of divergence expressed as mean branch length. The blue, orange and green colors represent the testing dataset for 50, 100 and 150 sequences respectively.

Number of sequences		Mean branch length					
		0.005	0.01	0.02	0.04	0.08	0.16
50	FP	0	0	0	1	1	0
	FN	41	17	4	1	2	4
100	FP	0	0	1	1	0	0
	FN	25	11	4	0	2	1
150	FP	0	0	1	0	0	0
	FN	35	6	1	0	0	0

Total	FP	0	0	1	2	1	0
	FN	101	34	9	1	4	5

Table-2.4 Results from Case C. Number of false positive and false negative cases based on the level of divergence and in the number of sequences.

We wanted to better understand the false cases that we have, hence the reasons of these results. The pattern of coevolution in our data is accounting for multiple co-changes between pair of sites, the more co-changes they are between a pair of sites, stronger the signal of coevolution is. Given that we don't know which pairs of sites are under coevolution from the predictions of our Coev-Asymmetric-CNN output, we looked at each site and counted how many times a type of change has occurred ($f_{i,j}$) and per F matrix we take its maximum $f_{i,j}$. Then, we looked at the distribution of the maximum number of changes for the F matrices where there is coevolution, and we highlighted where the false negatives cases are (Fig-2.14 left for 100 sequences). We saw that at levels with medium to high divergence the false negative cases tend to be the one where the F matrices have a lower maximum number of occurrences than expected at its divergence level. However, at very low levels, there was one false positive case for level of divergence of 0.01 and there are 6 cases for level of divergence of 0.005 where the maximum number of occurrences are above the average expected at that level of divergence, and a deeper study needs to be done. Nevertheless, this pattern is also similar for 150 and 50 sequences (Supp.Fig-2.10; Supp.Fig-2.11).

Moreover, we also looked at the distribution of the maximum number of occurrences for the F matrices where there is no coevolution, and we highlighted where the false positive cases are (Fig-2.14 right for 100 sequences). We see that at level of divergence 0.04, the only false positive case is where its F matrix has a higher maximum number of occurrences than expected at its divergence level. For level of divergence 0.02, a deeper study needs to be done to better understand why the CNN is doing that prediction. Nevertheless, we can see a tendency for the false positive cases, including 150 and 50 sequences (Supp.Fig-2.10; Supp.Fig-2.11) where its F matrices have a maximum number of occurrences above the mean maximum number of occurrences expected at its level of divergence.

These results suggests that the number of occurrences may have an impact on the performance of the CNN, where a low number of occurrences may contribute to be a false negative case, and a high number of occurrences may contribute to be a false positive case.

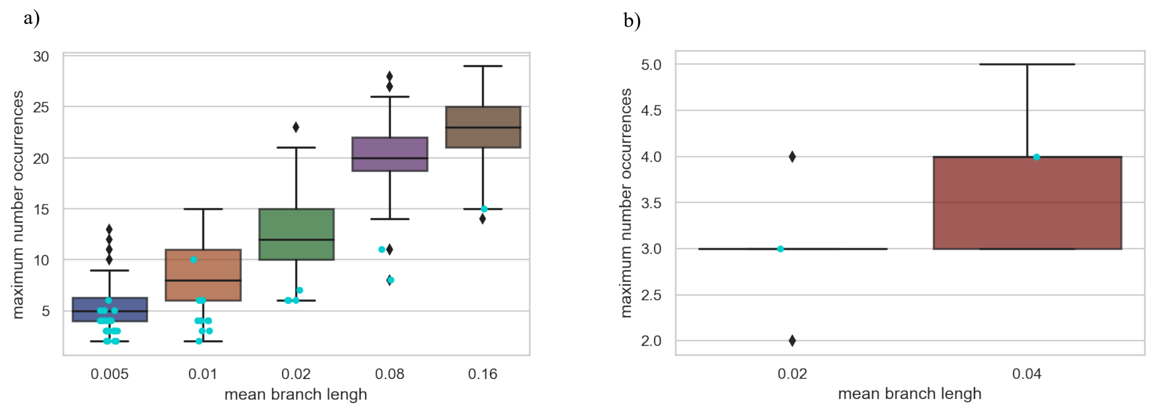


Fig-2.14 For 100 sequences and for case C. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of average branch length from 0.005 to 0.16. a) shows the distribution of the F matrices predicted as True Positive at each level of divergence. Each point in blue represents one F matrix predicted as False Negative. b) shows the distribution of the F matrices predicted as True Negative at each level of divergence. Each point in blue represents one F matrix predicted as False Positive. There are only shown the levels of divergence where there is a False Negative or a False Positive.

As a final step, we looked at the activation maps of our trained CNN when classifying F matrices from our testing dataset in Case C. First, for each F matrix predicted as true positive (F matrix simulated under coevolution and predicted under coevolution) we followed the pipeline described in Fig-2.3 and we selected activation value for the top X sites simulated under coevolution (each F matrix was simulated with a different number of coevolving sites, following a poison distribution See Simulations in Materials and methods). In Fig-2.15 we see that for 100 sequences (similar pattern found for 50 and 150 sequences) the true of coevolving sites are on the top X activated sites from the CNN.

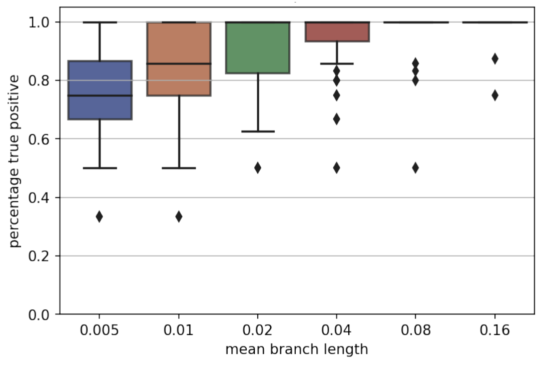


Fig-2.15 For 100 sequences and for case C. The y axis represents the percentage of top X sites being under true coevolution. The x axis represents the different levels of mean branch length from 0.005 to 0.16. The plot shows the distribution of the percentage of true coevolving sites being in the top X of sites for each F matrix.

Further to this, for each F matrix we followed the pipeline described in Fig-2.3 and we selected the highest averaged activation value. In Fig-2.16 we see that for 100 sequences (similar pattern found for 50 and 150 sequences) the true of coevolving sites are on the top X activated sites from the CNN. In accordance with the results that we also showed in **Chapter 1**.

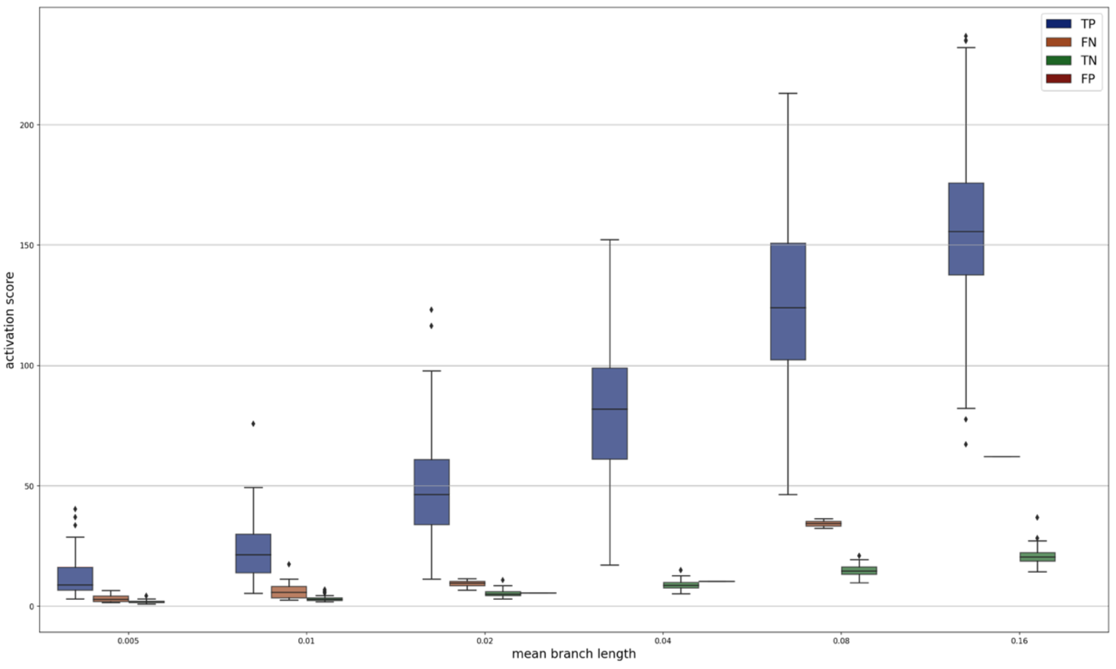


Fig-2.16 For 100 sequences and for case C. The y axis represents the maximum site' activation score per F matrix. The x axis represents the different levels of mean branch length from 0.005 to 0.16. The colors blue, orange, green and brown represents respectively the True Positive, False Negative, True Negative and False Positive predicted F matrix cases.

4. CONCLUSIONS

In this work, we have presented Coev-Asymmetric-CNN, a machine learning model based on convolutional neural networks to predict if there is any signal of coevolution given an MSA and its phylogenetic tree. Reconstructing the ancestral state for each site at each branch of the tree, to see if there is a co-occurring pattern has been already investigated by (Dutheil, 2012). Therefore, we went a step further and mapped these types of changes in a matrix and counted the occurrences of these changes per site. Taking this information as an input, we trained the CNN over a labeled benchmarking dataset to classify the signature of coevolution, which have a characteristic pattern manifested in the matrix.

Because divergence has an impact on predicting coevolution, see **Chapter 1**, we simulated the benchmarking dataset under the Coev and LG evolutionary models, with different divergence scenarios; contrary to other studies where the phylogeny is simulated in the absent of evolution (Marmier et al., 2019).

We verify, as expected (Su et al., 2019; Alzubaidi et al., 2021), that the performance of our CNN improves when the training dataset is more diverse, including the different levels of divergence simulated and the several number of sequences. The results are very robust at high divergence levels, while at low divergence we find difficulties as was already shown by methods like plmDCA, Comap and Coev in **Chapter 1**. It would now be interesting to test on a real dataset and see the performance of the CNN mixing more disperse features, varying more the level of divergence, the number of sequences, the topology of the trees and the sequence length.

We also showed that padding had an effect in the performance of the CNN. Alternatively, we could try to use other type of padding to see how the CNN performances. Instead of padding to left or to the bottom of the matrix, it would be interesting to further investigate the effect of padding all around the matrix.

Last but not less, it might be interesting to look into RNNs. While the CNNs are good capturing the pattern of coevolution, this problem could be seen as a time series problem. The insight gained with our data transformation from the phylogenetic tree with the MSA, might help to develop other machine learning models able to detect the signature of coevolution too.

5. SUPPLEMENNTARY DATA

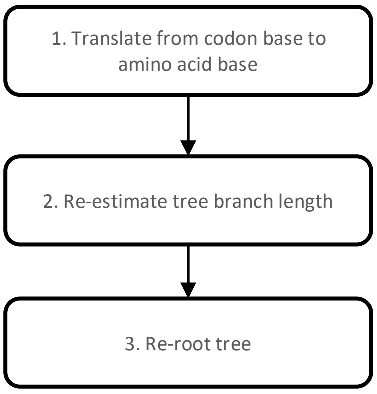
5.1. Simulated Data

We simulated a training dataset based on the Selectome Database of bony vertebrates. This curated database is codon and in consequence we converted it from nucleotides to amino acid base re-estimating also the branches of the phylogenetic trees. Because we missed a benchmarking dataset to train our network, we reutilized the phylogenetic trees to simulate alignments and have a simulated dataset closer to reality.

5.1.1. Pipeline from nucleotides to amino acid

First, we took the dataset Euteleostomi (vertebrates) from the positive selection Selectome database. Because it is codon-based, we translated the DNA sequence to amino acid base. We re-estimated the branch tree length for each tree using phyML (Guindon & Gascuel, 2003). The substitution model was set to LG, bootstrap was not computed, the number of relative substitution rate categories was set by default to 4. The gamma distribution was set to *e* to get the maximum likelihood estimate, and the branch length and rate parameters were optimized.

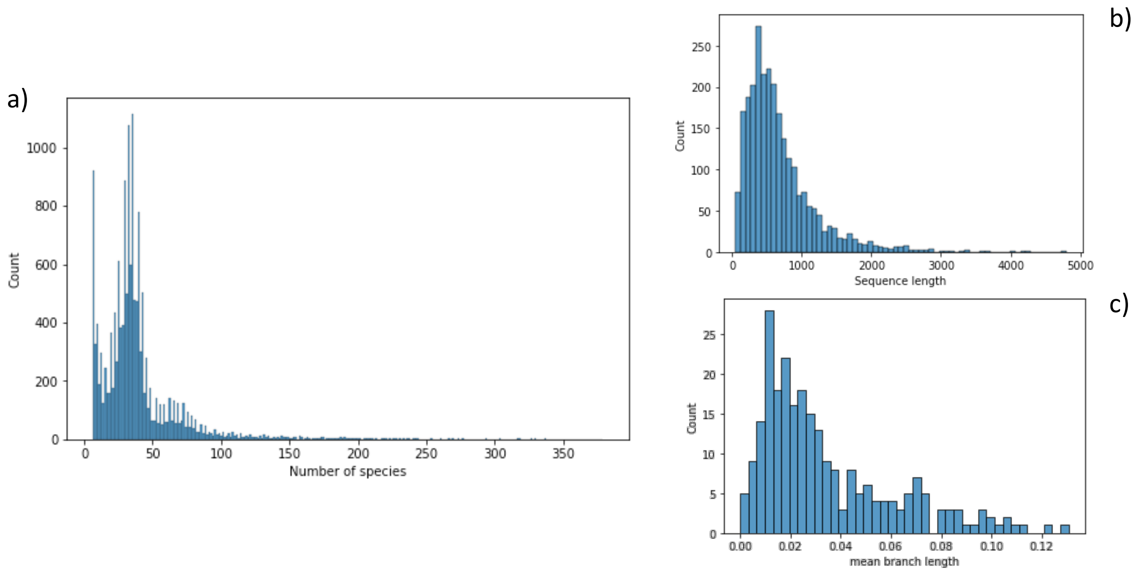
Finally, we re-rooted the tree to maintain the tree topology.


1. Translate from codon base to amino acid base
2. Re-estimate tree branch length
3. Re-root tree

Supp.Fig-2.1 Pipeline to convert the alignment from codon base to amino acid and to re-estimate the branch length of the tree

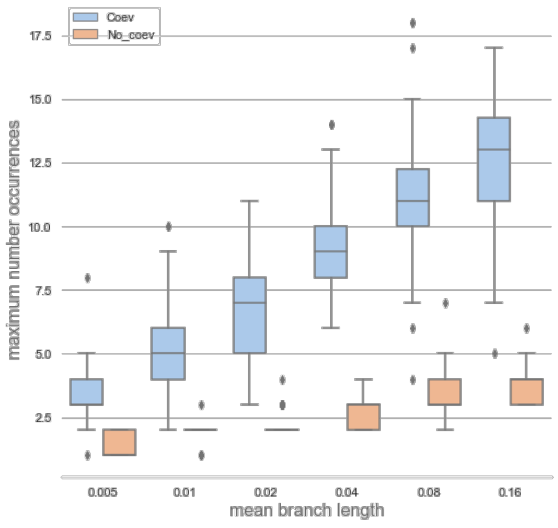
5.1.2. Selected training dataset

We filtered the amino acid dataset to have only samples with a number of sequences between 50 and 200 sequencies and a sequence length in the range of 50 and 200 amino acids too. In total there are 252 samples

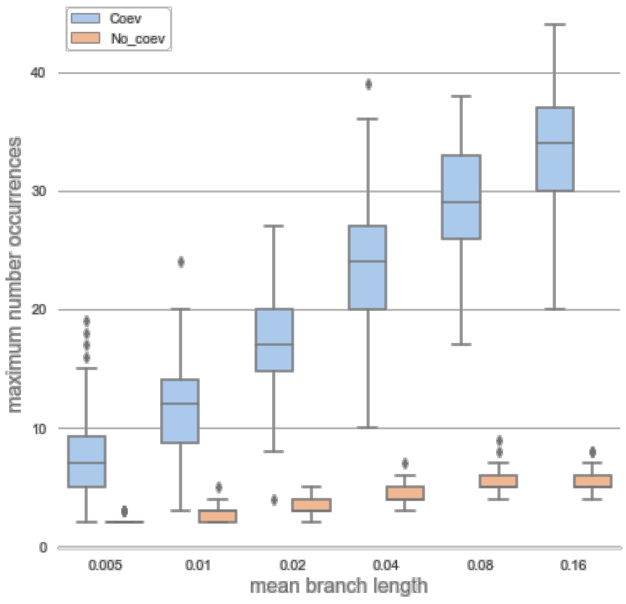


Supp.Fig-2.2 a) number of sequences in the bony vertebrate's dataset. b) sequence length distribution for samples with a number of sequences between 50 and 200. c) the average branch length distribution for the samples with a number of sequences between 50 and 200.

5.2. Distribution maximum number of occurrences for 50 and 150

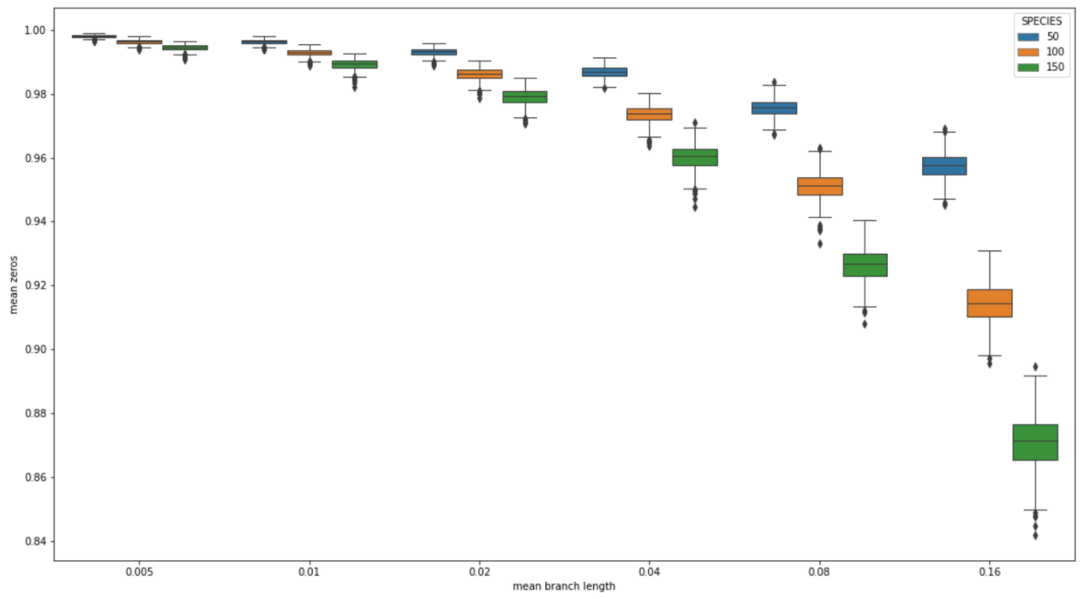


Supp.Fig-2.3 Distribution maximum number of occurrences for 50 sequences. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of divergence, from 0.005 to 0.16. At each level of divergence there is the distribution of the maximum number of occurrences for each site under coevolution in blue and for each site under non coevolution inn orange.



Supp.Fig-2.4 Distribution maximum number of occurrences for 150 sequences. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of divergence, from 0.005 to 0.16. At each level of divergence there is the distribution of the maximum number of occurrences for each site under coevolution in blue and for each site under non coevolution inn orange.

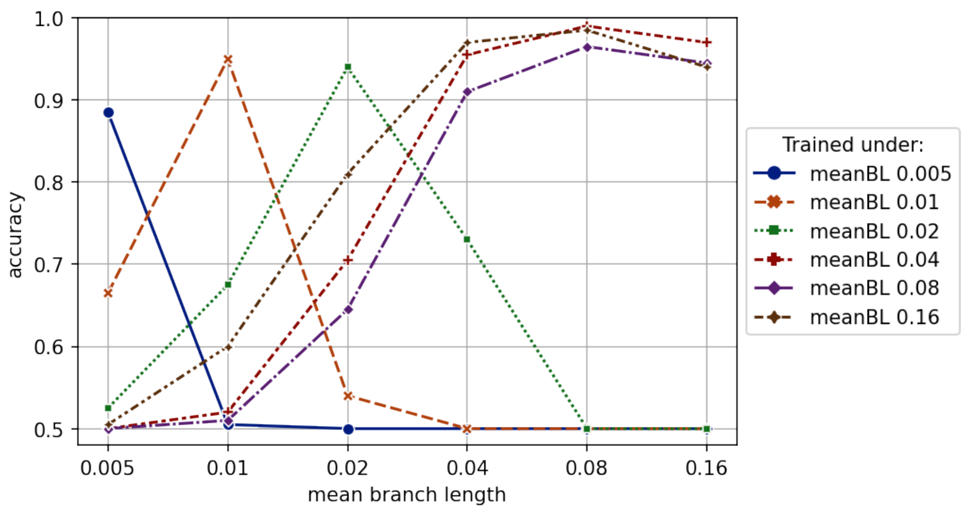
5.3. Distribution of zeros



Supp.Fig-2.5 Boxplots showing the mean of zeros (y-axis) for each F matrix with 50, 100 and 150 sequences (blue, orange and green), at each level of divergence (x-axis).

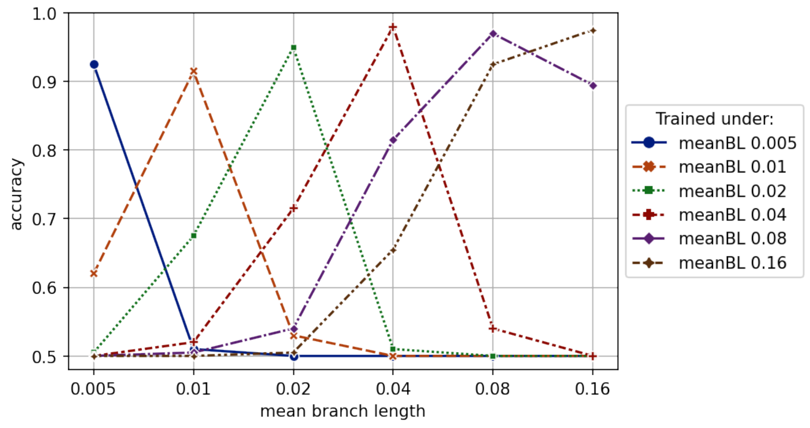
5.4. Case A

5.4.1. 150 sequences



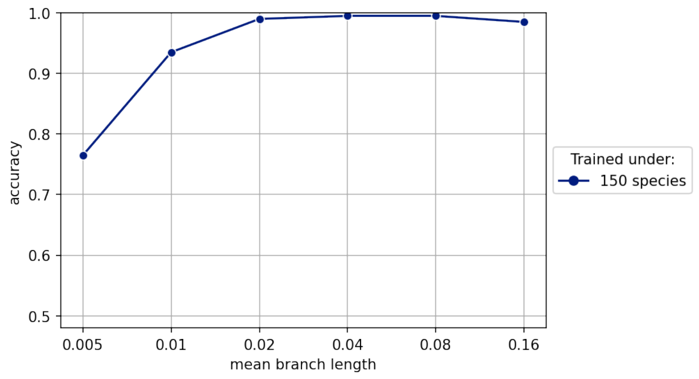
Supp.Fig-2.6 Case A. For 150 sequences. The y-axis represents the accuracy of the trained CNN on each tested dataset. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue, orange, green, red, purple and brown colors represent the trained CNN for the divergence level of 0.005, 0.01, 0.02, 0.04, 0.08 and 0.16 respectively. Each point corresponds to the accuracy of each trained CNN tested under the different levels of divergence.

5.4.2. <u>50 sequences</u>



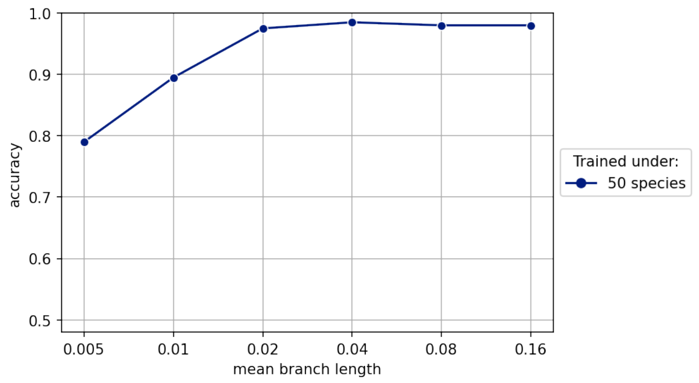
Supp.Fig-2.7 Case A. For 50 sequences. The y-axis represents the accuracy of the trained CNN on each tested dataset. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue, orange, green, red, purple and brown colors represent the trained CNN for the divergence level of 0.005, 0.01, 0.02, 0.04, 0.08 and 0.16 respectively. Each point corresponds to the accuracy of each trained CNN tested under the different levels of divergence.

5.4.3. <u>Sub-case for 150 sequences</u>



Supp.Fig-2.8 Sub-case a. For 150 sequences. The y-axis represents the accuracy of the CNN for case A in the second scenario. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue color represents the trained CNN. Each point corresponds to the accuracy of trained CNN tested under the different levels of divergence.

5.4.3. <u>Sub-case for 50 sequences</u>

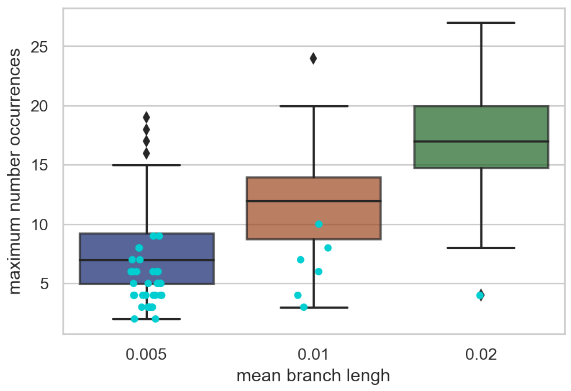


Supp.Fig-2.9 Sub-case a. For 50 sequences. The y-axis represents the accuracy of the CNN for case A in the second scenario. The x-axis represents the different levels of divergence from 0.005 to 0.16. The blue color represents the trained CNN. Each point corresponds to the accuracy of trained CNN tested under the different levels of divergence.

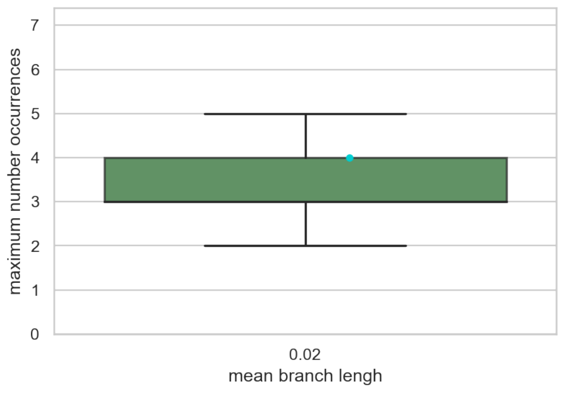
5.5. Distribution false positive

5.5.1. 150 sequences

a)



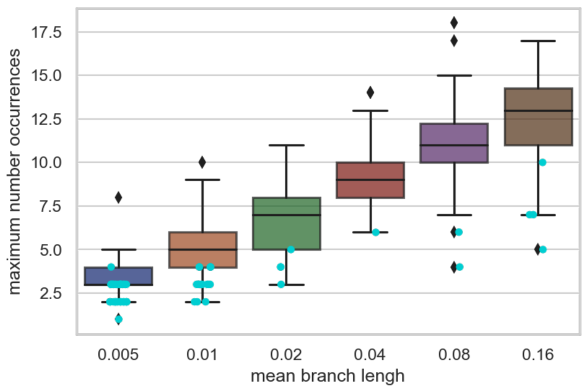
b)



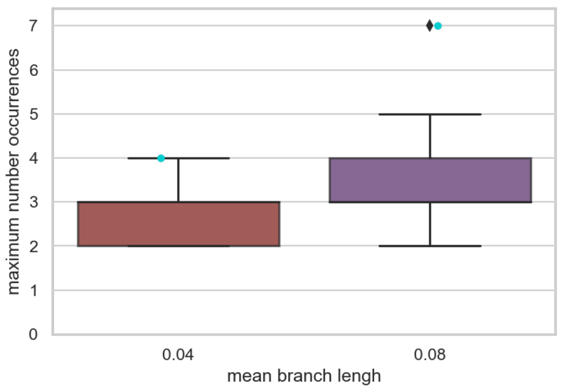
Supp.Fig-2.10 For 150 sequences and for case C. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of average branch length from 0.005 to 0.16. a) shows the distribution of the F matrices predicted as True Positive at each level of divergence. Each point in blue represents one F matrix predicted as False Negative. b) shows the distribution of the F matrices predicted as True Negative at each level of divergence. Each point in blue represents one F matrix predicted as False Positive. There are only shown the levels of divergence where there is a False Negative or a False Positive.

5.5.2. 50 sequences

a)



b)



Supp.Fig-2.11 For 50 sequences and for case C. The y axis represents the maximum number of occurrences per F matrix. The x axis represents the different levels of average branch length from 0.005 to 0.16. a) shows the distribution of the F matrices predicted as True Positive at each level of divergence. Each point in blue represents one F matrix predicted as False Negative. b) shows the distribution of the F matrices predicted as True Negative at each level of divergence. Each point in blue represents one F matrix predicted as False Positive. There are only shown the levels of divergence where there is a False Negative or a False Positive.

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, *22*(10), 1533–1545. https://doi.org/10.1109/TASLP.2014.2339736

Ackerman, S. H., Tillier, E. R., & Gatti, D. L. (2012). Accurate Simulation and Detection of Coevolution Signals in Multiple Sequence Alignments. *PLoS ONE*, *7*(10). https://doi.org/10.1371/journal.pone.0047108

Adhikari, B., Hou, J., & Cheng, J. (2018). Protein contact prediction by integrating deep multiple sequence alignments, coevolution and machine learning. *Proteins: Structure, Function, and Bioinformatics*, *86*, 84–96.

Adrion, J. R., Galloway, J. G., & Kern, A. D. (2019). Inferring the landscape of recombination using recurrent neural networks. *BioRxiv*, 662247. https://www.biorxiv.org/content/10.1101/662247v2

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data 2021 8:1*, *8*(1), 1–74. https://doi.org/10.1186/S40537-021-00444-8

Blischak, P. D., Barker, M. S., & Gutenkunst, R. N. (2021). Chromosome-scale inference of hybrid speciation and admixture with convolutional neural networks. *Molecular Ecology Resources*, *June 2020*, 1–13. https://doi.org/10.1111/1755-0998.13355

De Juan, D., Pazos, F., & Valencia, A. (2013). Emerging methods in protein co-evolution. *Nature Reviews Genetics*, *14*(4), 249–261. https://doi.org/10.1038/nrg3414

Dib, L., Salamin, N., & Gfeller, D. (2018). Polymorphic sites preferentially avoid co-evolving residues in MHC class I proteins. *PLoS Computational Biology*, *14*(5), 1–19. https://doi.org/10.1371/journal.pcbi.1006188

Dib, L., Silvestro, D., & Salamin, N. (2014). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dunn, S. D., Wahl, L. M., & Gloor, G. B. (2008). Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, *24*(3), 333–340. https://doi.org/10.1093/bioinformatics/btm604

Dutheil, J., Pupko, T., Jean-Marie, A., & Galtier, N. (2005). A model-based approach for detecting coevolving positions in a molecule. *Molecular Biology and Evolution*, *22*(9), 1919–1928. https://doi.org/10.1093/molbev/msi183

Dutheil, J. Y. (2012). Detecting coevolving positions in a molecule: Why and how to account for phylogeny. *Briefings in Bioinformatics*, *13*(2), 228–243. https://doi.org/10.1093/bib/bbr048

Ekeberg, M., Lövkvist, C., Lan, Y., Weigt, M., & Aurell, E. (2012). *Improved contact prediction in proteins: Using pseudo-likelihoods to infer Potts models*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Guindon, S., & Gascuel, O. (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, *52*(5), 696–704. https://doi.org/10.1080/10635150390235520

Hashemi, M. (2019). Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. *Journal of Big Data*, *6*(1), 1–13. https://doi.org/10.1186/S40537-019-0263-7/FIGURES/4

Heo, J., Lim, J. H., Lee, H. R., Jang, J. Y., Shin, Y. S., Kim, D., Lim, J. Y., Park, Y. M., Koh, Y. W., Ahn, S.-H., Chung, E.-J., Lee, D. Y., Seok, J., & Kim, C.-H. (2022). Deep learning model for tongue cancer diagnosis using endoscopic images. *Scientific Reports*, *12*(1), 1–10. https://doi.org/10.1038/s41598-022-10287-9

Hopf, T. A., Morinaga, S., Ihara, S., Touhara, K., Marks, D. S., & Benton, R. (2015). Amino acid coevolution reveals three-dimensional structure and functional domains of insect odorant receptors. *Nature Communications*, *6*(1), 1–7.

Jones, D. T., Buchan, D. W. A., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, *28*(2), 184–190. https://doi.org/10.1093/bioinformatics/btr638

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*. https://doi.org/10.1038/s41586-021-03819-2

Källberg, M., Wang, H., Wang, S., Peng, J., Wang, Z., Lu, H., & Xu, J. (2012). Template-based protein structure modeling using the RaptorX web server. *Nature Protocols 2012 7:8*, *7*(8), 1511–1522. https://doi.org/10.1038/nprot.2012.085

Kamisetty, H., Ovchinnikov, S., & Baker, D. (2013). Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era. *Proceedings of the National Academy of Sciences*, *110*(39), 15674–15679.

Kandathil, S. M., Greener, J. G., & Jones, D. T. (2019). Prediction of interresidue contacts with DeepMetaPSICOV in CASP13. *Proteins: Structure, Function and Bioinformatics*, *87*(12), 1092–1099. https://doi.org/10.1002/prot.25779

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Králik, M., & Ladányi, L. (2021). Canny Edge Detector Algorithm Optimization Using 2D Spatial Separable Convolution. *Acta Electrotechnica et Informatica*, *21*(4), 36–43. https://doi.org/doi:10.2478/aei-2021-0006

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Le, S. Q., & Gascuel, O. (2008). An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution*, *25*(7), 1307–1320. https://doi.org/10.1093/MOLBEV/MSN067

Lecun, Y., Bottou, L., Bengio, Y., & Ha, P. (1998). LeNet. *Proceedings of the IEEE, November*, 1–46.

Liu, J., Li, M., Luo, Y., Yang, S., Li, W., & Bi, Y. (2021). Alzheimer's disease detection using depthwise separable convolutional neural networks. *Computer Methods and Programs in Biomedicine*, *203*, 106032. https://doi.org/10.1016/j.cmpb.2021.106032

Liu, Z.-K., Zhang, L.-H., Liu, B., Zhang, Z.-Y., Guo, G.-C., Ding, D.-S., & Shi, B.-S. (2022). *Deep learning enhanced Rydberg multifrequency microwave recognition*. *2022*. https://doi.org/10.1038/s41467-022-29686-7

Liu, Z., Ren, M., Niu, Z., Wang, G., & Liu, X. (2020). DeepED: A Deep Learning Framework for Estimating Evolutionary Distances. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12396 LNCS*, 325–336. https://doi.org/10.1007/978-3-030-61609-0_26

Lo, J. E., Kang, E. Y. C., Chen, Y. N., Hsieh, Y. T., Wang, N. K., Chen, T. C., Chen, K. J., Wu, W. C., Hwang, Y. S., Lo, F. S., & Lai, C. C. (2021). Data Homogeneity Effect in Deep Learning-Based Prediction of Type 1 Diabetic Retinopathy. *Journal of Diabetes Research*, *2021*. https://doi.org/10.1155/2021/2751695

Lo, S. Y., Hang, H. M., Chan, S. W., & Li, J. J. (2018). Efficient Dense Modules of Asymmetric

Convolution for Real-Time Semantic Segmentation. *1st ACM International Conference on Multimedia in Asia, MMAsia 2019*. https://doi.org/10.48550/arxiv.1809.06323

Malinverni, D., Lopez, A. J., De Los Rios, P., Hummer, G., & Barducci, A. (2017). Modeling Hsp70/Hsp40 interaction by multi-scale molecular simulations and coevolutionary sequence analysis. *ELife*, *6*, 1–20. https://doi.org/10.7554/eLife.23471

Manning, T., Wassan, J. T., Palu, C., Wang, H., Browne, F., Zheng, H., Kelly, B., & Walsh, P. (2019). Phylogeny-Aware Deep 1-Dimensional Convolutional Neural Network for the Classification of Metagenomes. *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018*, 1826–1831. https://doi.org/10.1109/BIBM.2018.8621543

Marmier, G., Weigt, M., & Bitbol, A. F. (2019). Phylogenetic correlations can suffice to infer protein partners from sequences. *PLoS Computational Biology*, *15*(10), 1–24. https://doi.org/10.1371/journal.pcbi.1007179

Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., Zecchina, R., Onuchic, J. N., Hwa, T., & Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, *108*(49), E1293–E1301. https://doi.org/10.1073/pnas.1111471108

Muscat, M., Croce, G., Sarti, E., & Weigt, M. (2020). FilterDCA: Interpretable supervised contact prediction using inter-domain coevolution. *PLoS Computational Biology*, *16*(10), 1–19. https://doi.org/10.1371/journal.pcbi.1007621

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*(NeurIPS).

Proux, E., Studer, R. A., Moretti, S., & Robinson-Rechavi, M. (2009). Selectome: a database of positive selection. *Nucleic Acids Research*, *37*(suppl_1), D404–D407. https://doi.org/10.1093/NAR/GKN768

Qin, C., & Colwell, L. J. (2018). Power law tails in phylogenetic systems. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(4), 690–695. https://doi.org/10.1073/pnas.1711913115

Reiman, D., Metwally, A. A., Sun, J., & Dai, Y. (2020). PopPhy-CNN: A Phylogenetic Tree Embedded Architecture for Convolutional Neural Networks to Predict Host Phenotype from Metagenomic Data. *IEEE Journal of Biomedical and Health Informatics*, *24*(10), 2993–3001. https://doi.org/10.1109/JBHI.2020.2993761

Revell, L. J. (2012). phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, *3*(2), 217–223. https://doi.org/10.1111/j.2041-210X.2011.00169.x

Schliep, K. P. (2011). phangorn: phylogenetic analysis in R. *Bioinformatics*, *27*(4), 592–593. https://doi.org/10.1093/BIOINFORMATICS/BTQ706

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., & Bridgland, A. (2019). Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins: Structure, Function, and Bioinformatics*, *87*(12), 1141–1148.

Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Su, L., Fan, P., & Zhao, T. (2019). Deep learning the high variability and randomness inside multimode fibers. *Optics Express, Vol. 27, Issue 15, Pp. 20241-20258*, *27*(15), 20241–20258. https://doi.org/10.1364/OE.27.020241

Uguzzoni, G., Lovis, S. J., Oteri, F., Schug, A., Szurmant, H., & Weigt, M. (2017). Large-scale identification of coevolution signals across homo-oligomeric protein interfaces by direct coupling analysis. *Proceedings of the National Academy of Sciences of the United States of America*, *114*(13), E2662–E2671. https://doi.org/10.1073/pnas.1615068114

Wen, F., Zhang, Z., He, T., & Lee, C. (2021). AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove. *Nature Communications 2021 12:1*, *12*(1), 1–13. https://doi.org/10.1038/s41467-021-25637-w

Yin, R., Luusua, E., Dabrowski, J., Zhang, Y., & Kwoh, C. K. (2020). Tempel: Time-series mutation prediction of influenza A viruses via attention-based recurrent neural networks. *Bioinformatics*, *36*(9), 2697–2704. https://doi.org/10.1093/bioinformatics/btaa050

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zerihun, Mehari B., & Schug, A. (2017). Biomolecular coevolution and its applications: Going from structure prediction toward signaling, epistasis, and function. *Biochemical Society Transactions*, *45*(6), 1253–1261. https://doi.org/10.1042/BST20170063

Zerihun, Mehari Bayou, & Schug, A. (2018). RNA Structure Prediction Guided by Coevolutionary Information. *Biophysical Journal*, *114*(3), 436a.

Zhang, X., & LeCun, Y. (2015). *Text Understanding from Scratch*. 1–9. http://arxiv.org/abs/1502.01710

Zizka, A., Andermann, T., & Silvestro, D. (2022). IUCNN – Deep learning approaches to approximate species' extinction risk. *Diversity and Distributions*, *28*(2), 227–241. https://doi.org/10.1111/ddi.13450

Zou, Z., Zhang, H., Guan, Y., Zhang, J., & Liu, L. (2020). Deep Residual Neural Networks Resolve Quartet Molecular Phylogenies. *Molecular Biology and Evolution*, *37*(5), 1495–1507. https://doi.org/10.1093/MOLBEV/MSZ307

Chapter 3

Coev-Asymmetric-CNN: A Convolutional Neural Network for predicting coevolution accounting for phylogeny

*"As one Google Translate engineer put it:
when you go from 10,000 training examples
to 10 billion training examples,
it all starts to work.
Data trumps everything."*

Garry Kasparov

Coev-Asymmetric-CNN: A Convolutional Neural Network for predicting coevolution accounting for phylogeny

ABSTRACT

Here we present Coev-Asymmetric-CNN, a Convolutional Neural Network (CNN) method to infer coevolution with a user-friendly pipeline. We demonstrated the success of the CNN after training it under a controlled simulated dataset. We further demonstrated that taking into account the phylogenetic tree, of the protein under study, improves the comprehension of the pairs identified under coevolution. Using the bony vertebrate Selectome genomic dataset, we show that Coev-Asymmetric-CNN is able to detect coevolution in 217 proteins. Additionally, we identified the limitations of Coev-Asymmetric-CNN when testing it on a real genomic dataset. The Coev-Asymmetric-CNN code is available at https://github.com/phylolab/Coev-Asymmetric-CNN.

INTRODUCTION

In recent years, machine learning (ML) approaches such as deep learning methods have gained popularity due to the huge growth and development in the field of big data (Najafabadi et al., 2015; Kremer et al., 2017). Deep learning methods are highly accurate to analyze big datasets and find complex patterns in the data (Lecun et al., 2015). They have proved to be useful tools for large-scale analysis in many different fields such as physics (Liu et al., 2022), language recognition (Wen et al., 2021) or medicine (Heo et al., 2022). The exponential growth of sequenced genomes nowadays makes it possible to have databases with thousands of proteins (Moretti et al., 2014; Mistry et al., 2021), which we can analyze to better understand biological processes.

One of the important biological processes that we can study with genomic data is coevolution, which can provide essential information about the structure of a protein (Sutto et al., 2015), its function (Fares & Travers, 2006), and also the interaction between proteins (Malinverni et al., 2017). Indeed, there are many available methods to infer coevolution. Methods only considering a multiple sequence alignment (MSA) to infer coevolution like: DCA, PSICOV, GREMLIN, (Jones et al., 2012; Kamisetty et al., 2013; Ekeberg et al., 2014); and methods like CoMap, CAPS or Coev, that study coevolution considering a MSA and its associated phylogenetic tree (Dutheil et al., 2005; Fares & Travers, 2006; Dib et al., 2014). Nevertheless, these methods are often limited by the number of sequences needed to be accurate about their prediction and the large computing time that is needed to run the analysis. By taking advantage of ML methods, we could potentially analyze and study coevolution in a scalable and efficient way and apply these analyses to genomic scale data.

In **Chapter 2** we developed a scalable and efficient ML method based on Convolutional Neural Networks that we called Coev-Asymmetric-CNN. It enabled us to infer the signal of coevolution while integrating the evolutionary history of the sequences. With a given labeled dataset, it is possible to train the Coev-Asymmetric-CNN to predict if coevolution has left a signature in the amino acid patterns or not. Even though, to our knowledge, there is not a benchmarking dataset with labeled data to be used to train a ML model, it is possible to simulate thousands of datasets with and without coevolution (Baker et al., 2018) given the contribution of evolutionary models such as LG (Le & Gascuel, 2008), and coevolutionary models such as Coev (Dib et al., 2014). Thus, we can simulate as much labeled data as needed by varying the number of sequences, or the level of divergence, for example (like we did in **Chapter 1** and

Chapter 2), and adapt the training dataset depending on the type of protein family under study. After the training process, we can predict coevolution rapidly on databases with thousands of proteins.

Although CNNs are highly successful for image classification/detection, their application to detect the signature of coevolution is novel and no tools are readily available to apply our new model. We implemented Coev-Asymmetric-CNN using two asymmetrical 1D convolutional layers. We have shown in **Chapter 2** the characteristic pattern of the signature of coevolution, where there are co-occurrent changes happening for different rows at the same columns. This type of data structure is possible to detect making use of asymmetric spatial separable convolutions (Lo et al., 2018; Králik & Ladányi, 202). The standard 2D kernel used for the CNN is split across its two spatial axes, which are, in our case, the rows, representing the sites of the MSA, and the columns, representing the branches of the phylogenetic tree. Instead of applying a direct convolution of a 2D kernel on the input matrix F, we performed two separate convolutions. First, we convolved the matrix F of size *L x B* with a vector of size *L x 1*, resulting in a vector of size *B x 1*. This filter allowed the detection of sites that shared a similar pattern of substitution. Second, we applied the convolution of a vector *1 x B* on the result of the previous convolution to detect the branches having a similar pattern of substitution. The resulting scalar will give an estimation of the weight of coevolution signal corresponding to the two 1D filters in the input matrix F (check **Chapter 2**: Materials and Methods for more details).

The development of libraries and API (Chollet & others, 2015; Abadi et al., 2016; Team et al., 2016; Paszke et al., 2019) dedicated to ML algorithms, make it easier to implement, to test and to run CNN algorithms. Moreover, profiting from these tools, we can design and implement our own CNN to depart from the standard image classifiers. However, it is still not straightforward how to prepare the data, implement the CNN model and output the results with visualizations. Here we present a user-friendly pipeline described in a Jupyter notebook to predict coevolution on MSAs and their phylogenetic trees. The implementation is available on Github[2] and the pipeline can also be executed through the command line with a Python module. Our implementation contains three main steps:

1) Data input preparation:
    a. Transformation of the phylogenetic tree and the alignment into a matrix M_{LxB} accounting for the changes in the ancestral state reconstruction

[2] https://github.com/phylolab/Coev-Asymmetric-CNN

b. Conversion of the M_{LxB} matrices to F_{LxB} matrices, considering the number of occurrences per each type of change

2) Training and validation of the CNN-Asymmetric-Coev with a given labeled input dataset
3) Testing the CNN-Asymmetric-Coev on a given labeled input dataset, and output the results

Finally, we illustrated the use of the Coev-Asymmetric-CNN on the positive selection Selectome Database. The goal is to show the steps to follow to analyze and explore some characteristics of the data that can help to predict which protein families have been evolving through coevolution. We analyzed a total of 252 protein families and we detected 217 proteins under coevolution. We showed that Coev-Asymmetric-CNN can run in an efficient computing time. In addition, we studied these results to better understand the signature of coevolution.

2. MATERIALS AND METHODS

A typical workflow to execute Coev-Asymmetric-CNN contains three steps: 1) the preparation of the input data, 2) the training and validation of the model, and 3) the prediction and visualization of the testing datasets. We present here each of these steps and give the implementation details that were used to run the model.

2.1. Input preparation

Our Coev-Asymmetric-CNN uses as an input a matrix derived from substitutions occurring for each site of a MSA along each branch of its phylogenetic tree. We assume that the matrix and the tree are in *fasta* and *newick* format, respectively, which are standard formats used for such data. The workflow (Fig-3.1) to generate the input data needed for the CNN has two parts. The first part is done in R, while the second is implemented in Python.

First, we need to generate a matrix M_{LxB} (L = sequence length, B = number of branches), whose values $m_{i,n}$ are the types of substitutions occurring for site *i* on branch *n*. Even though we assume that the input MSA is amino-acid based, the script can be modified to process nucleotides passing as an argument the *fasta* type (nucleotides or amino acids). For amino acid datasets, which contain 20 different states, there are a total of 380 possible substitution types, while nucleotide datasets have 4 different states and a total of 12 possible substitution types.

Each substitution type is represented by the $m_{i,n}$ values, which can include the value 0 to indicate no change (see **Chapter 2** for more details). The order of the branches is created automatically by the *phytools* R package (Revell, 2012) when reading the phylogenetic tree. This has, however, no impact on the CNN model as branches can be reordered without affecting the coevolution pattern. At the end, we generate a matrix M_{LxB} that will be used, after another transformation step (see below), as an input feature to the CNN (Fig-3.1). In most datasets, the majority of the sites of the alignment do not change over most branches, especially for lowly evolving amino acid/nucleotide sequences. The M_{LxB} matrix is therefore usually sparse.

Once the M_{LxB} matrix is created, we transform it into a frequency matrix F_{LxB} by counting the number of times each substitution type occurred at a given site. The values $f_{i,n}$ of the F matrix represent the number of times the type of change encoded in $m_{i,n}$ has occurred for site i through the phylogenetic tree (see **Chapter 2** for more details).

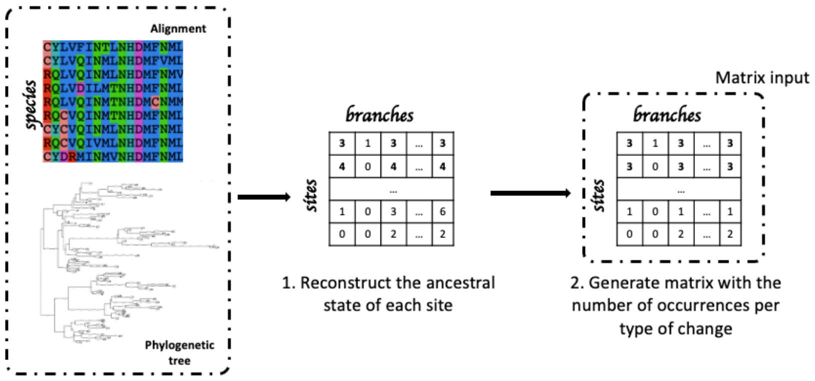

Alignment
species
Phylogenetic tree
branches
sites
1. Reconstruct the ancestral state of each site
Matrix input
branches
sites
2. Generate matrix with the number of occurrences per type of change

Fig-3.1 The pipeline for data generation. 1) The user provides a MSA and a phylogenetic tree 2) Ancestral state reconstruction for each site is done and it is mapped into a matrix of changes (M_{LxB}) where the branches are the columns and the sites are the rows. 3) F_{LxB} matrix is generated with the number of occurrences per type of change from M_{LxB}.

The dataset containing the MSAs and their phylogenetic trees must be already prepared, following the directory hierarchy shown in Fig-3.2, to generate the F matrices. The full dataset (both for training and testing the CNN) must contain two types of classes: a set of MSAs with their corresponding phylogenetic trees with signature of coevolution, labeled as '*COEV*'; and a set of MSAs with their corresponding phylogenetic trees with no signature of coevolution, labeled as '*NO_COEV*'. To simulate a reliable dataset with labeled data we recommend following the steps in **Chapter 2**.

```
cnnData
|-- test
|   |-- COEV
|   `-- NO_COEV
`-- train
    |-- COEV
    `-- NO_COEV
```

Fig-3.2 Folder structure where the parent directory (cnnData) contains 2 folders: test and train. Each of them has 2 sub-folders: COEV and NO_COEV, with the MSAs and the phylogenetic trees.

2.2. CNN-Asymmetric-Coev training and validation

We implemented the CNN-Asymmetric-Coev model based on one input layer of size *LxB* with a single channel, two asymmetric convolutional layers (the first one *Lx1* and the second one *1xB*), and a fully connected layer with two output values, one value for each class (i.e. presence or absence of a coevolution signal; Fig-3.3).

After each convolution layer, we used a standard rectified linear activation function (ReLU; Schmidhuber, 2015). Other functions can also be used, but it was shown that the *ReLU* allowed the model to learn faster and reach better performance in most CNN applications (Goodfellow et al., 2016). The results from the second convolution were then converted into a 1-dimensional array and fed into the fully connected network. We further used the regularization method *Dropout* (Srivastava et al., 2014) to reduce overfitting and improve the generalization in the model. This is done during training, where some nodes of the fully connected network are randomly ignored. Finally, we used a *LogSoftMax* activation function as the output layer of our CNN to obtain a vector of two values describing the probabilities for the input data to be under coevolution or not.

We did not use pooling in our model due to the characteristics of our input data. Co-substitutions occurring across sites at several branches have a positional dependency in the F_{LxB} matrix representing the process of coevolution. If we applied a pooling layer, then the positional dependency would be lost and the input matrix labeled as '*COEV'* would no longer represent the coevolution pattern that our CNN is supposed to learn.

Moreover, we did not normalize the input data because of the sparseness contained in the input matrices (see above *Input preparation*). A normalization step would make the values in the F matrix very small and, thus, very similar to each other. It would render the signal of coevolution fuzzy and impossible to detect by the CNN.

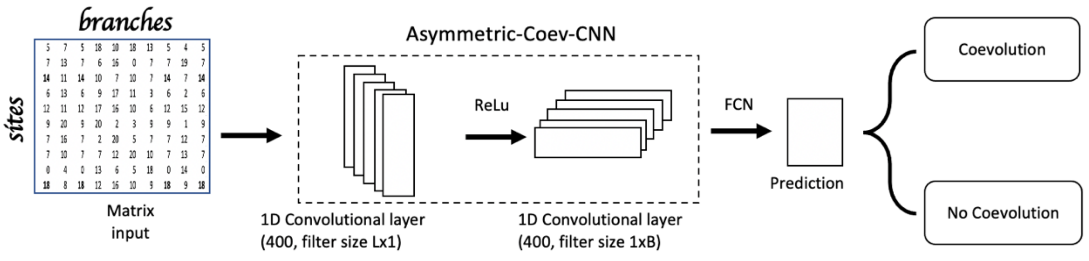

branches
sites
Matrix input
Asymmetric-Coev-CNN
ReLu
1D Convolutional layer (400, filter size Lx1)
1D Convolutional layer (400, filter size 1xB)
FCN
Prediction
Coevolution
No Coevolution

Fig-3.3 CNN-Asymmetric-Coev network architecture. The matrix F_{LxB} is the input to the Asymmetric-Coev-CNN. It contains two 1D convolution layers, each of them with a ReLu activation function. Finally, a fully connected network layer provides the output: there is signature or coevolution or not.

2.3. CNN-Asymmetric-Coev predicting and visualizing coevolution

Once trained, CNN-Asymmetric-Coev can be applied to any input matrix F_{LxB}, of the same size as it was trained for, to predict if there is any signature of coevolution. The prediction label is returned for each input matrix. If the prediction label returned is 'COEV' (there is signature of coevolution), then it will also return by default the 4 most probable sites under coevolution (it can be modified to return more sites). Moreover, it is possible to plot, in an image, the rows from the F_{LxB} matrix corresponding to these coevolving sites, to better visualize the branches where the co-substitutions are happening.

2.4. Implementation

There are many libraries available to implement deep learning models. The most popular are based in Python such as: Keras (Chollet & others, 2015), TensorFlow (Abadi et al., 2016) or PyTorch (Paszke et al., 2019). We implemented CNN-Asymmetric-Coev model in PyTorch (Paszke et al., 2019) because its coding style is more imperative and dynamic, which makes the code more readable and understandable. Furthermore, we also find that PyTorch also allows low-level coding, giving more freedom to write custom CNN models, contrary to Keras. Moreover, because of our big training datasets, PyTorch and TensorFlow are better in terms of performance. Finally, we wanted to have a prototype, and we found PyTorch's less complex framework ideal.

We created our own CNN model, a PyTorch object (named *coevClassifier*) based on the class *torch.nn.Module* that is used for all neural network modules. The instantiation of this class requires two arguments specified by the user: the size of the two 1D filters: *ROW_SIZE* for the length of the sequence alignment, and *COL_SIZE* for the total number of branches. Both convolutional layers are implemented using the PyTorch function *Conv2d*. The first layer is a

1D convolution layer and it is created with 1-dimensional 400 filters of size *ROW_SIZE x 1* (by default it is 400 but it could be modified). The second convolutional layer is another 1D convolution and it is created with 1-dimensional 400 filters of size *1 x COL_SIZE* (also, by default it is 400 filters, but it could be modified). We incorporated a bias parameter in both layers. Moreover, the stride of the convolution is set to 1, which means that the 1D filters will be applied to each row and column in turn. We further set the padding to 0. Finally, after each convolutional layer, we are utilizing the *ReLU* function, already implemented in Pytorch, converting all the negative values to 0. The two layers of fully connected networks were implemented using the PyTorch function *Linear*, which applies a linear transformation to the data. The first fully connected neural network takes as an input the number of filters used in the previous convolutions and transforms them into an output size of 100 nodes (or neurons). Between these two layers, we used the PyTorch function *dropout* during training, giving a probability of 0.25 for a node to be set to zero. The output was then passed as input to the second fully connected network, which returned an output consisting of two values. We applied the function *log_softmax* implemented in PyTorch to the final output and selected the maximum value as the final result.

During training, we used the optimizer function *Adam* from PyTorch (Kingma & Ba, 2015) with a standard learning rate of 0.008. It is possible to use other optimizers (e.g. Adadelta or RMSprop; Zeiler, 2012; Schaul et al., 2013), but *Adam* can better handle the sparse features of our data and it worked well with standard parameters. Our CNN aimed at classifying our input data into coevolving and non-coevolving datasets. We thus selected a loss function that is appropriate for this type of problems and used the PyTorch function *CrossEntropyLoss*. Moreover, we utilized mini-batches of size 64 by default, meaning that during training it loads in memory 64 samples of the training dataset. However, this value can be modified by the user. It is recommended to use a power of 2 for the mini-batch size (64, 128, 256 or 512). If the data set is bigger, then a good option could be to increase the mini-batch size. Nevertheless, the bigger the size of the mini-batches, the more data that is loaded in memory.

Furthermore, the weights of the convolutional and fully connected layers were initialized through the default Kaiming Uniform method (He et al., 2015), which is the most appropriate way to initialize the weights when the ReLU activation function is used.

Finally, the training of the CNN was implemented with the possibility of using an early-stop to avoid overfitting. The training loop stops when the performance on the validation dataset (validation loss) starts to degrade consecutively during 2 epochs. This early-stop step can also be decided by the user.

The current version of the software can be downloaded and run from GitHub (https://github.com/phylolab/Coev-Asymmetric-CNN). The readme file on the GitHub repository provides a list with all the packages needed to be installed before running the software.

2.5. Data visualization an activation map

After the CNN has been trained, we can input a F matrix to our CNN to obtain a prediction score defining if the MSA contained any signal of coevolution or not. Due to the fact that it is the second convolutional layer the one detecting which sites are under coevolution (it is the one having the filters of size *1 x L,* and going row by row detecting which pattern is common between them), we proposed a pipeline to identify the most probable sites to be under coevolution (Fig-3.4), using the activation maps from the second convolutional layer.

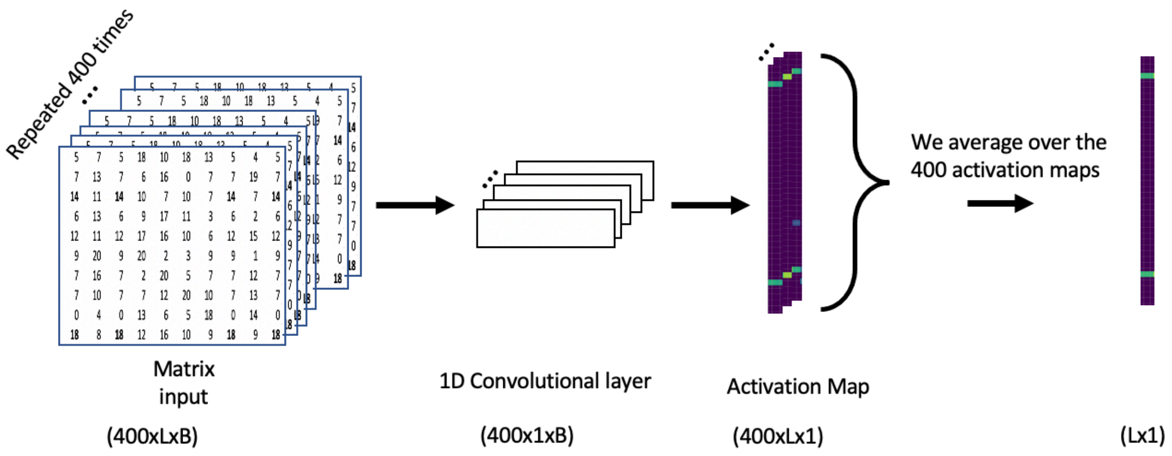


Fig-3.4 Pipeline to rank the sites to get the top most probable sites under coevolution. First we have the input matrix of size 400xLxB, (the 2D F matrix repeated 400 times), then, we applied the 400 learnt filters of size 1xB to each dimension of our input. After that, we get the activation map of size 400xLx1 (each of the 400 dimensions correspond to one specific filter from the previous step). Finally, we average over the 400 activation maps to get the average activation map of size Lx1. This last output is a vector where the L values correspond to the activation value of the L sites.

First, we repeated the input F matrix as many times as the number of filters used in the second convolutional layer. In our case, there were 400 filters in the second convolutional layer and the input F matrix was repeated 400 times to have a matrix of size *400 x L x B*. We then applied the second convolutional layer to this new input matrix (F matrix repeated 400 times) to obtain the activation maps of size *400 x L x 1*. Each of the 400 activation maps (also known as feature maps) corresponds to the activation of the different sites from the input F matrix. We averaged the 400 activation maps, with size *L x 1*, to obtain a single final averaged activation map of size *L x 1*. Finally, to identify the sites that were responsible for the strong activations and thus the signal of coevolution, we ranked the L sites based on the averaged activation value and selected those with higher values to represent the sites more likely to be under coevolution.

There are other methods available to visualize the activation maps like Grad-CAM (Selvaraju et al., 2017). However, given our CNN architecture, with 2 asymmetrical 1D filters, the activation maps returned by Grad-Cam only provide useful information for the first convolutional filter. Thus, we can detect which branches are "activated", but not which sites..

2.6. Data input format

To run the Coev-Asymmetric-CNN, there are two options: load all the training dataset in memory and run the CNN, or load in memory only the batches which the CNN is going to train. For example, in **Chapter 2**, the total size of case C was approx. 4GB (14,400 F matrices in format *.csv*). Thus, we provided two different types of data format to store the input data depending on the memory limitations that the users may have. First, it is possible to use the standard *npy* format, which is a binary file that can be manipulated through the Numpy library (Harris et al., 2020). The input data will have to include 6 files in total: 3 files to train the CNN and 3 files to test it. In each of these two cases, the 3 files contain i) all the F_{LxB} matrices (*input.npy*), ii) the labels associated with each matrix (0 for no coevolution and 1 for coevolution) (*label.npy*), iii) the name of the files containing each F_{LxB} matrix (*name.npy*). The latter is important to be able to retrieve the MSA and phylogenetic tree that were used to create the F matrices. Second, we provide the *H5* format, which is a Hierarchical Data Format (HDF) able to store multidimensional arrays with a structure (Koranne, 2011). The input data will be contained in a single file that includes the 6 datatypes described for the *npy* format above. It acts like a dictionary, and it is possible to access any of the following elements: training_input F_{LxB} matrices, training_label, training_names; and testing_input F_{LxB} matrices, testing_label, testing_names.

We have done a small benchmarking comparing the behavior between these two types of formats, checking the memory occupancy and the time to train the CNN during one epoch (there are 64 F_{LxB} matrices in each epoch) *(*Table-3.1). We see that if we have a memory limitation, it is better to use the *H5* format to train the CNN. When the number of filters is low, and consequently the number of parameters to train the network is small, loading all the data with the *npy* format is faster than with the *H5* format. When increasing the number of filters, and consequently the number of parameters to be trained, the time to train the network, during one epoch, is practically the same between the two formats but the memory footprint is much more reduced with the *npy* format.

Dataset size	Data type	Filters	Time (seconds)	Memory
~ 6 GB ($Matrix_{199x396}$)	h5	100	66	< 700 MB
		200	174	< 1 GB
		300	335	~ 1GB
	npy	100	40	~ 6.8 GB
		200	152	~ 7 GB
		300	314	~ 7 GB
~ 2.5 GB ($Matrix_{199x396}$)	h5	100	27	< 700 MB
		200	70	< 1 GB
		300	137	~ 1 GB
	npy	100	16	~ 3 GB
		200	61	~ 3 GB
		300	125	< 4 GB

Table-3.1 Comparative table between h5 and npy formats varying the size of the dataset and the number of filters. Bigger dataset and a higher number of filters mean that it will take more time to train the model and more memory to allocate.

3. PIPELINE EXECUTION EXAMPLE

To illustrate how to proceed to run Coev-Asymmetric-CNN, here we describe the steps to follow: 1) input preparation, 2) model training and validation, and 3) prediction and visualization.

3.1. Input preparation

1. Execute the script *run_ancestral_changes_matrix.sh* with the following inputs:
   - a *fasta* MSA
   - a *newick* phylogenetic tree
   - a folder to save the matrix M_{LxB} as a *csv* file

```
bash run_ancestral_changes_matrix.sh my_fasta_file_1.fasta my_tree_file_1.nwk ~/cnnData/train/COEV/
```

This step must be done for each MSA and its phylogenetic tree, the more samples/simulations we have, the better. To have a balanced dataset, we should have a similar number of matrices in *COEV* and *NO_COEV* folders.

2. Execute the script *generate_cnn_datasets.py* with the following inputs:

- path: the path to the training and testing label dataset (with the hierarchy of Fig-3.2)
- output: the path to store the file(s) with all the training and testing label data (by default ./)
- format: h5 or npy (by default h5)
- row: the number of rows (sites) of our final matrices the value must be the highest number of rows through all the matrices (by default 199)
- col: the number of columns (branches) of our final matrices the value must be the highest number of columns through all the matrices (by default 396)

```
python generate_cnn_datasets.py -path ~/cnnData/ -output ~/cnnFiles/ -format h5 -row 199 -col 396
```

The program will go through all the M_{LxB} matrices files, first in the *~/cnnData/train/* folder and later in the *~/cnnData/test/*, transforming all the M_{LxB} matrices in the F_{LxB} and saving them into a new file(s).
See Supplementary Data for more details about the output.

3.2. Model training and validation

3. Execute the script *run_CNN.py*. It takes as an input a configuration file (Supp.Fig-3.1), which contains all the needed variables to train the CNN.

```
python train_CNN.py config_train.txt
```

The program trains a Coev-Asymmetric-CNN model with the training dataset provided in the configuration file.

It outputs the accuracy of the CNN for the testing dataset and saves the trained model as a PyTorch "*.pt*" file.

3.3. Prediction and visualization

4. Execute the script *test_CNN.py*. It takes as an input a configuration file (Supp.Fig-3.2), which contains all the needed variables to test the CNN.

```
python test_CNN.py config_test.txt
```

The program loads the Coev-Asymmetric-CNN model from the configuration file to infer coevolution on a given testing dataset.

It outputs a csv file (Supp.Fig-3.3) which contains information about the predictions. Moreover, it outputs a pdf file (Supp.Fig-3.4), which each page is a file predicted with coevolution signal and an image of the most probable 2 pairs being under coevolution.

3.5. Illustrating our pipeline on an empirical data

We wanted to apply Coev-Asymmetric-CNN to a real dataset to infer coevolution. We selected the bony vertebrate Selectome dataset (Proux et al., 2009), which is a positive selection database with curated data containing more than 15,000 gene families of vertebrates with their phylogenetic trees and corresponding alignments. Given the lack of a labeled dataset with and without coevolution to train our model, the first step we did was to simulate a training labeled dataset (MSAs with their corresponding phylogenetic trees). Then, we transformed the simulated data into a 2D matrix representing the number of substitutions occurring at each site along each branch of the tree. The 2D matrices are then used as the training dataset input to the CNN architecture. Afterward, we used the trained CNN to predict coevolution on the real bony vertebrate Selectome dataset.

The predictions of coevolution required first to train Coev-Asymmetric-CNN on a labeled dataset. However, there is little understanding of the minimum size that such a training dataset should have to obtain a good performance for CNN methods, and some studies suggest to take increasingly bigger subsets of the data, calculate the error and see how the model performances when increasing the training dataset (Cho et al., 2015; Balki et al., 2019).

Therefore, we simulated three datasets going from a small, medium, and large number of samples included in the training (~2,000, ~4,000 and ~8,000 respectively). We used the phylogenetic trees available for the datasets from the Selectome database to simulate new MSA as described below.

3.5.1. Simulations

First, we filtered the Selectome database to have a more heterogeneous dataset with only alignments and phylogenetic trees containing between 50 and 200 sequences whose sequence length was longer than 50 but smaller than 200 amino-acids. Our final dataset contained 252 MSAs with their phylogenetic trees.

We simulated new MSAs by resampling with replacement the 252 phylogenetic trees. The distributions of the number of sequences, the mean branch lengths of the phylogenetic trees and the alpha parameter for the Gamma distribution used to model site heterogeneity were given in Supp.Fig-3.5. For each tree sampled, we simulated a new MSA by varying the sequence length and the alpha value by randomly selecting these two parameters from their respective distributions. The MSA were randomly assigned to one of the two labeled categories defined as either evolving under coevolution, using the LG (Le & Gascuel, 2008) and Coev models (Dib et al., 2014) or under the independent LG model alone. The details of the simulations can be found in **Chapter 2** and we followed here strictly the same procedure.

Next, we transformed each MSA and its phylogenetic tree into a 2D matrix, F_{LxB}, as described in **Chapter 2**. Considering that each matrix has a different number of sites and branches, we padded all of them with zeros until all had the size of the biggest matrix (199 rows x 396 columns).

3.5.2. Training and testing dataset

We studied the performance of our Coev-Asymmetric-CNN while varying the number of samples used to train our network. The large dataset contained a total of 8,476 simulated samples (MSA and trees) and half of the dataset was simulated under the signature of coevolution and the other half was simulated under no signature of coevolution. We split the dataset into different training datasets as follows: the first case, referred to as the small dataset, contained 1,008 samples, the second case, referred to as the medium dataset, contained 3,296 samples, while the third case, referred to as the big dataset, contained 8,072 samples.

To test the performance of each training case we randomly took from the full dataset 562 samples, not included during the training phase. There are 291 samples simulated under coevolution and 271 under the independent model. Moreover, this testing dataset was common to test the performance of the three training scenarios.

Moreover, the performance of the model for each scenario was done calculating the recall and precision over the testing dataset. Where the recall is defined as the number of true positive predictions divided by the sum of the true positive and the false negative predictions; and the precision is defined as the number of true positive predictions divided by the total number of predicted positive cases.

recall = true positives/(true positives + false negatives)

precision = true positives/(true positives + false positives)

3.5.3. <u>Results from the simulated dataset</u>

We tested the Coev-Asymmetric-CNN performance under the different simulated datasets. We observed that increasing the size of the training dataset increased the accuracy of the CNN model (Table-3.2). These results were in agreement with the literature (Hestness et al., 2017; Sun et al., 2017), where it was also shown that the size of the training data had an effect on the accuracy of the model. In general, the more data present in the training dataset, the better was the accuracy of the CNN model. In our case, when the training dataset size increases by 8 times, the accuracy of the model increases from 0.79 of accuracy with the smallest dataset to 0.87 for the bigger one. Moreover, we showed that the ratio of false negative cases decreased when increasing the size of the training dataset size (Table-3.2). Concerning the false positive cases, our simulations showed that they were stable across the different sizes of training datasets with only a difference of 2 among them. Overall, our results suggested that we need a bigger training dataset.

Training Dataset	Training size	Testing size	Accuracy	TP	TN	FP	FN
Small	1008	562	0.79	192	254	17	99
Medium	3296	562	0.83	213	255	16	78
Big	8072	562	0.87	237	253	18	54

Table-3.2 Results after training the Coev-Asymmetric-CNN with small, medium and big. Testing results analysis showing the number of true positive, true negative, false positive and false negative cases.

Based on the results obtained with the *Big dataset*, we looked at better understanding the behavior of the CNN. Beside the prediction of coevolution done at the MSA level, it is also possible to identify the pairs of sites that activated the CNN and led to a positive prediction. First, we looked at the activation score given by Coev-Asymmetric-CNN to the top sites predicted as coevolution for each F matrix (to know more details about how the activation score is calculated, read the "Visualization and activation maps" in the Implementation section). We observed that the activation score is in general higher for the true positive cases (Fig-3.5). The mean activation score for the true positives sites is 44.55 with a minimum value of 1.3 and a maximum of 337.6. The true negative cases have a mean value of 2.2, with a range between 0.4 and 46.6. We also observed that the false negative had in average a lower activation score than the false positive cases (Fig-3.5). Still, it is difficult to understand the false positive cases only based on the activation score. These false positive cases have the highest activation score lower than 21 (lower than the mean value for the true positive cases), and they have a mean activation score of 8.8. Nevertheless, these results suggest that even though the activation score it is not enough to understand how the classification it is done, it may be an indicator for the CNN to decide if there is coevolution.

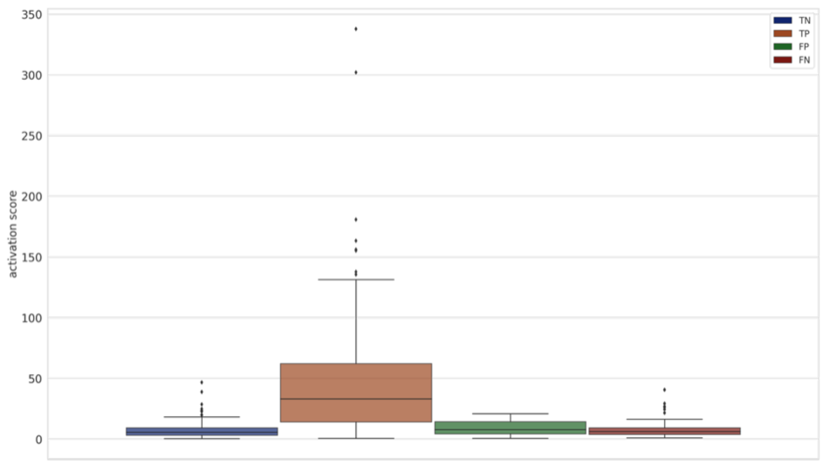


Fig-3.5 Boxplots of the highest activation score per alignment. The y-axis represents the maximum site' activation score per F matrix. The colors blue, orange, green and brown represent respectively the True Positive, False Negative, True Negative and False Positive predicted F matrix cases.

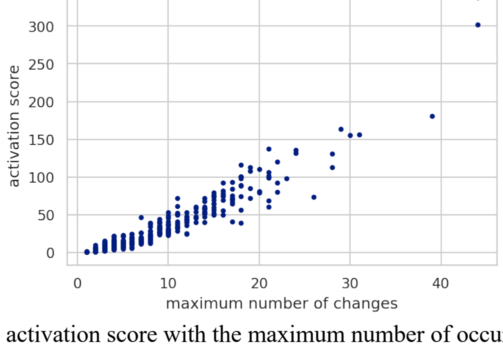


Fig-3.6 Positive correlation between activation score with the maximum number of occurrences per F matrix. Each point is a F matrix. The y-axis represents the activation score of the top-1 site given by Coev-Asymmetric-CNN. The x axis is the maximum number of occurrences in a given F matrix

In addition, given the results from **Chapter 2**, where we showed that the maximum number of changes had an effect on detecting the signature of coevolution, we looked at the correlation between the maximum number of changes and the activation score Fig-3.6. We observed that these two variables are positively correlated and the higher the maximum number of changes, the higher the activation score is. This also supports the results from **Chapter 2**, and it suggests that the maximum number of changes per F matrix may be having an impact on the activation score and, therefore, on the way Coev-Asymmetric-CNN is learning to detect the signature of coevolution.

Finally, we looked at the recall and precision (see *3.5.2. Training and testing dataset*) of our Coev-Asymmetric-CNN considering the maximum number of occurrences per F matrix. In Fig-3.7, we see that the recall is equal to 1 when the maximum number of occurrences are bigger than 15. For the precision it is when the maximum number of occurrences are bigger than 8, meaning that above 8 number of occurrences, there are not false positive cases, only coevolving simulations have 8 or more number of occurrences. Once again, these results are consistent with the previous results, suggesting that there is a bias in the simulations and that coevolving simulations tend to have more number of occurrences happening than the simulations done under the independent model. Thus, actually, may lead to false positive cases, i.e., when the number of occurrences happening for one site is high, but it is not co-occurring with any other site.

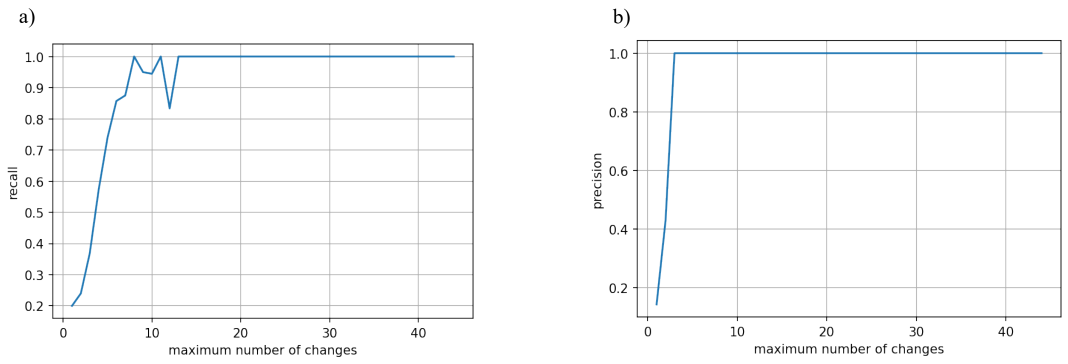


Fig-3.7 Recall (a) and precision (b) scores based on the maximum number of changes for each F matrix. Both values (recall and precision) increase with the maximum number of changes.

3.5.4. Predictions on the real dataset

After training Coev-Asymmetric-CNN and seeing that we still had a good overall accuracy, we predicted coevolution on the real bony vertebrate Selectome dataset. It took 40 seconds to go through the 252 proteins. We showed that the CNN, trained with the *Big dataset*, predicted 217 proteins as being under coevolution.

The large number of MSA predicted under coevolution (86% of the data) was unexpected. When comparing with another method (CoMap) that uses the phylogenetic information to estimate pairs of coevolving sites, we found that all 252 datasets analyzed had at least one pair of sites significant. It shows that our method is probably more able to tease apart real coevolution signals from noise coming from the pattern of amino-acid frequencies found in the MSA, especially at low sequence divergence (see **Chapter 1**). To better understand the factors involved in this result, we showed in Fig-3.8 the distribution of maximum number of changes in both the real and simulated datasets. The distribution is similar in both datasets. Nevertheless, trying to understand a bit better the behavior of our Coev-Asymmetric-CNN, and knowing from **Chapter 1** and **Chapter 2** that divergence had an effect on predicting coevolution, we looked at the divergence of our data. In Fig-3.9, we observed that on the simulated dataset, we have the same range of mean branch lengths for the simulations of either coevolution or no coevolution. In contrast, in the real protein dataset, all the samples predicted to be under no coevolution have a very low mean branch length, while the samples predicted to be coevolving had a wide range of mean branch length (Fig-3.9). The conclusions that we get from Fig-3.8 and Fig-3.9, is that on real data, the maximum number of occurrences are directly linked with the mean branch length. Somehow, this is something expected because the lower the mean branch length the less changes are expected to occur.

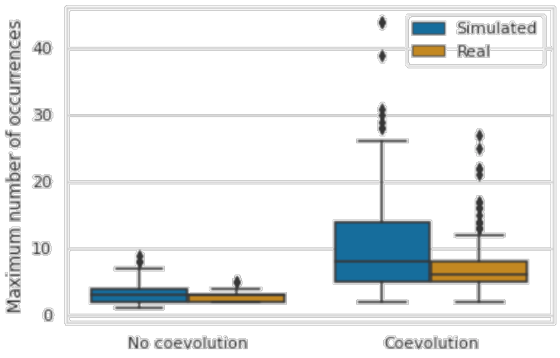


Fig-3.8 Boxplots of the maximum number of occurrences of the cases predicted by Coev-Asymmetric-CNN under coevolution and without coevolution on simulated data (blue) and Selectome dataset (orange)

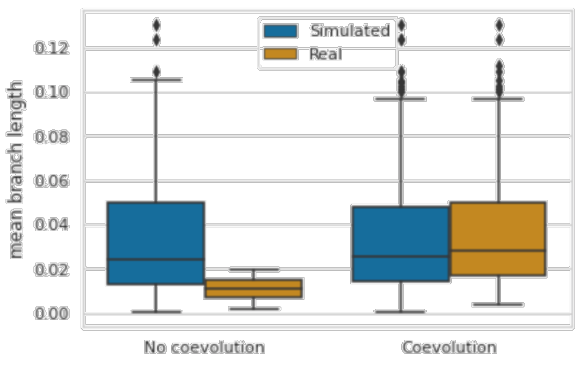


Fig-3.9 Boxplots of the mean branch length of the cases predicted by Coev-Asymmetric-CNN under coevolution and without coevolution on simulated data (blue) and Selectome dataset (orange)

4. CONCLUSIONS

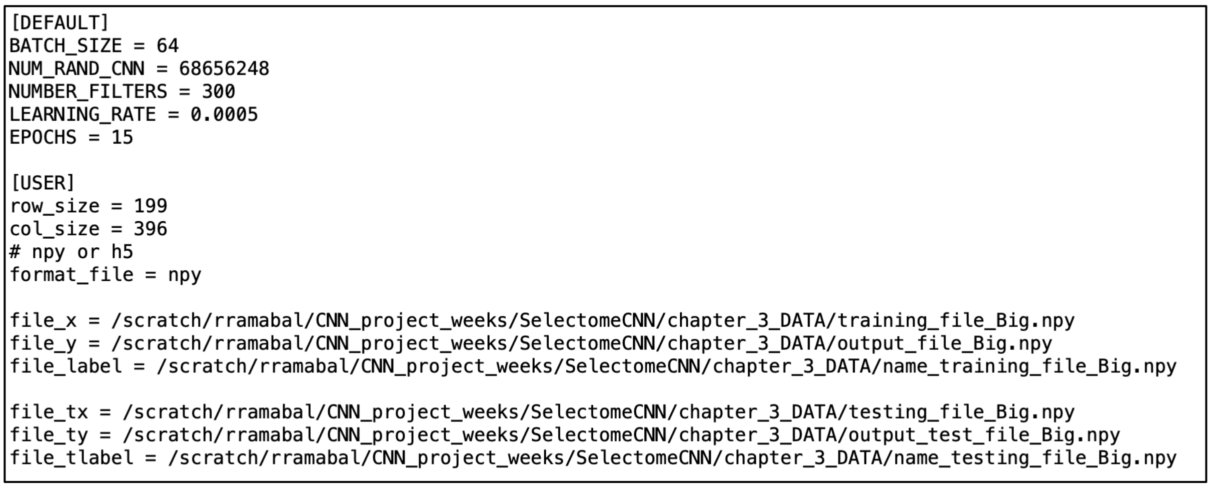
Coevolution is an important evolutionary phenomenon that occurs to maintain proteins' function and structure. Nevertheless, understanding coevolution and detecting its signature is not an easy task. In this study we presented Coev-Asymmetric-CNN, a user-friendly implementation of a deep learning method to infer the signature of coevolution. It provides two main features: 1) a way to transform a given MSA with its phylogenetic tree into a type of data easy to input to a CNN; and 2) a CNN model for predicting the signature of coevolution.

Given that the good accuracy of the model is based on the quality of the training process, the training dataset is decisive. We observed that a big dataset with enough variability and closer to the type of real genomic data is needed. One way to improve the training process is to have a training dataset with as much variability as the real testing dataset. For this, some features that could be useful to take into account would be: including more trees with different topologies, varying the gamma values per alignment, and adding bigger or lower number of sequences are some of the features to take into account.

Moreover, we trained our Coev-Asymmetric-CNN with a big dataset (~8.000 samples) and we analyzed 252 protein families from the bony vertebrate Selectome database. The results indicate that there are 217 protein families under coevolution. Also, we showed that Coev-Asymmetric-CNN is probably bias by the maximum number of changes occurring per site, per alignment. Thus, there is still a need for a more deeply investigation of these families.

5. SUPPLEMENTARY DATA

5.1. Configuration files



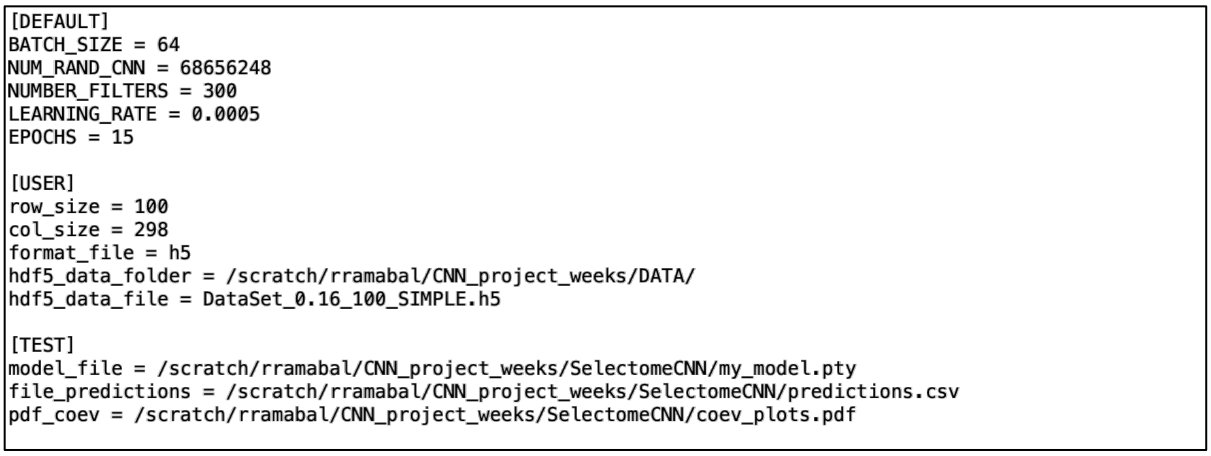
```
[DEFAULT]
BATCH_SIZE = 64
NUM_RAND_CNN = 68656248
NUMBER_FILTERS = 300
LEARNING_RATE = 0.0005
EPOCHS = 15

[USER]
row_size = 199
col_size = 396
# npy or h5
format_file = npy

file_x = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/training_file_Big.npy
file_y = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/output_file_Big.npy
file_label = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/name_training_file_Big.npy

file_tx = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/testing_file_Big.npy
file_ty = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/output_test_file_Big.npy
file_tlabel = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/chapter_3_DATA/name_testing_file_Big.npy
```

Supp.Fig-3.1 Configuration file to train the CNN



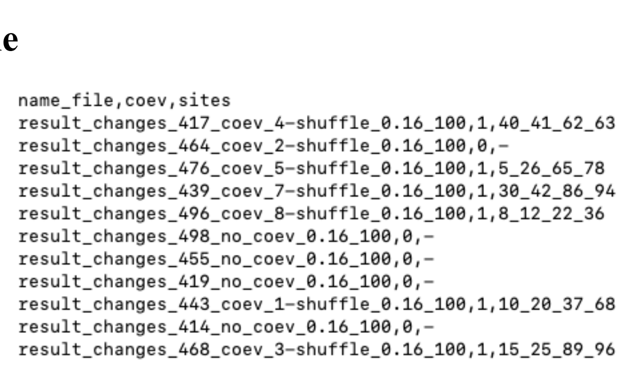
```
[DEFAULT]
BATCH_SIZE = 64
NUM_RAND_CNN = 68656248
NUMBER_FILTERS = 300
LEARNING_RATE = 0.0005
EPOCHS = 15

[USER]
row_size = 100
col_size = 298
format_file = h5
hdf5_data_folder = /scratch/rramabal/CNN_project_weeks/DATA/
hdf5_data_file = DataSet_0.16_100_SIMPLE.h5

[TEST]
model_file = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/my_model.pty
file_predictions = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/predictions.csv
pdf_coev = /scratch/rramabal/CNN_project_weeks/SelectomeCNN/coev_plots.pdf
```

Supp.Fig-3.2 Configuration file for predictions and visualization

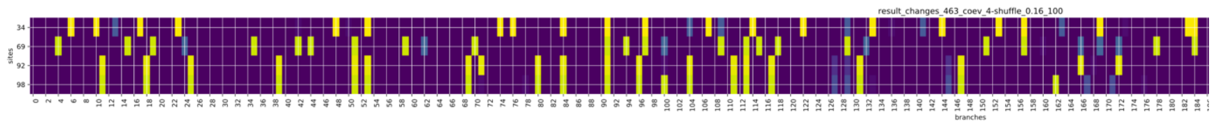
5.2. Output csv file



```
name_file,coev,sites
result_changes_417_coev_4-shuffle_0.16_100,1,40_41_62_63
result_changes_464_coev_2-shuffle_0.16_100,0,-
result_changes_476_coev_5-shuffle_0.16_100,1,5_26_65_78
result_changes_439_coev_7-shuffle_0.16_100,1,30_42_86_94
result_changes_496_coev_8-shuffle_0.16_100,1,8_12_22_36
result_changes_498_no_coev_0.16_100,0,-
result_changes_455_no_coev_0.16_100,0,-
result_changes_419_no_coev_0.16_100,0,-
result_changes_443_coev_1-shuffle_0.16_100,1,10_20_37_68
result_changes_414_no_coev_0.16_100,0,-
result_changes_468_coev_3-shuffle_0.16_100,1,15_25_89_96
```

Supp.Fig-3.3 Output csv file with the result from the predictions.

5.3. Output pdf file



Supp.Fig-3.4 Output pdf file with the sites under coevolution with the branches where the changes occur.

5.4. Output format

The output files vary depending on the type of format.

If the format is *.npy*, then we will have 6 outputs:

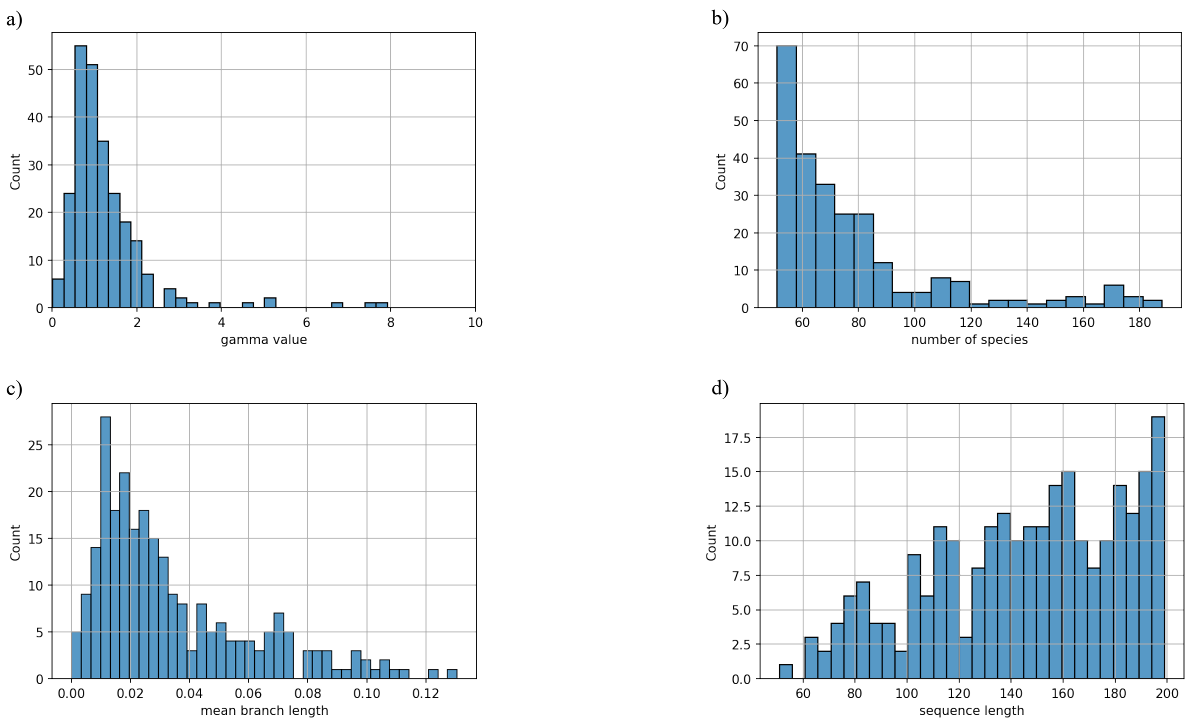
- *training_file.npy* Containing all the F_{LxB} matrices for training
- *name_training_file.npy* Containing all the names corresponding to the F_{LxB} matrices for training
- *label_training_file.npy* Containing the values 1 or 0 corresponding to the F_{LxB} matrices for testing, depending on if the matrix was on the COEV or NO_COEV folder respectively

- *testing_file.npy* Containing all the F_{LxB} matrices for testing
- *name_testing_file.npy* Containing all the names corresponding to the F_{LxB} matric
- es for testing
- *label_testing_file.npy* Containing the values 1 or 0 corresponding to the F_{LxB} matrices for testing, depending on if the matrix was on the COEV or NO_COEV folder respectively

On other hand, the *.h5* format will contain a single file: *name_output.h5.*
It has a hierarchical structure, completely hidden for the user, separated as well on training and testing:

- training_dataset
- filenames_training_dataset
- label_training_dataset
- testing_dataset
- filenames_testing_dataset
- label_testing_dataset

5.5. Distribution features Selectome dataset



Supp.Fig-3.5 Analysis from the bony vertebrate Selectome filtered dataset. Histograms showing the distribution of the gamma value (a), number of species (b), mean branch length (c) and sequence length (d).

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Research, G. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. https://doi.org/10.48550/arxiv.1603.04467

Baker, R. E., Peña, J. M., Jayamohan, J., & Jérusalem, A. (2018). Mechanistic models versus machine learning, a fight worth fighting for the biological community? *Biology Letters*, *14*(5). https://doi.org/10.1098/RSBL.2017.0660

Balki, I., Amirabadi, A., Levman, J., Martel, A. L., Emersic, Z., Meden, B., Garcia-Pedrero, A., Ramirez, S. C., Kong, D., Moody, A. R., & Tyrrell, P. N. (2019). Sample-Size Determination Methodologies for Machine Learning in Medical Imaging Research: A Systematic Review. *Canadian Association of Radiologists Journal*, *70*(4), 344–353. https://doi.org/10.1016/j.carj.2019.06.002

Cho, J., Lee, K., Shin, E., Choy, G., & Do, S. (2015). *How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?* https://doi.org/10.48550/arxiv.1511.06348

Chollet, F., & others. (2015). *Keras*. GitHub. https://github.com/fchollet/keras

Dib, L., Silvestro, D., & Salamin, N. (2014a). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dib, L., Silvestro, D., & Salamin, N. (2014b). Evolutionary footprint of coevolving positions in genes. *Bioinformatics*, *30*(9), 1241–1249. https://doi.org/10.1093/bioinformatics/btu012

Dutheil, J., Pupko, T., Jean-Marie, A., & Galtier, N. (2005). A model-based approach for detecting coevolving positions in a molecule. *Molecular Biology and Evolution*, *22*(9), 1919–1928. https://doi.org/10.1093/molbev/msi183

Ekeberg, M., Hartonen, T., & Aurell, E. (2014). Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences. *Journal of Computational Physics*, *276*, 341–356. https://doi.org/10.1016/j.jcp.2014.07.024

Fares, M. A., & Travers, S. A. A. (2006). A novel method for detecting intramolecular coevolution: Adding a further dimension to selective constraints analyses. *Genetics*, *173*(1), 9–23. https://doi.org/10.1534/genetics.105.053249

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature 2020 585:7825*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. https://doi.org/10.3389/fpsyg.2013.00124

Heo, J., Lim, J. H., Lee, H. R., Jang, J. Y., Shin, Y. S., Kim, D., Lim, J. Y., Park, Y. M., Koh, Y. W., Ahn, S.-H., Chung, E.-J., Lee, D. Y., Seok, J., & Kim, C.-H. (2022). Deep learning model for tongue cancer diagnosis using endoscopic images. *Scientific Reports*, *12*(1), 1–10. https://doi.org/10.1038/s41598-022-10287-9

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., & Zhou, Y. (2017). *Deep Learning Scaling is Predictable, Empirically.*

http://arxiv.org/abs/1712.00409

Jones, D. T., Buchan, D. W. A., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, *28*(2), 184–190. https://doi.org/10.1093/bioinformatics/btr638

Kamisetty, H., Ovchinnikov, S., & Baker, D. (2013). Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(39), 15674–15679. https://doi.org/10.1073/PNAS.1314045110/-/DCSUPPLEMENTAL/SD02.XLS

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Koranne, S. (2011). Hierarchical data format 5: HDF5. In *Handbook of Open Source Tools* (pp. 191–200). Springer.

Králik, M., & Ladányi, L. (2021). Canny Edge Detector Algorithm Optimization Using 2D Spatial Separable Convolution. *Acta Electrotechnica et Informatica*, *21*(4), 36–43. https://doi.org/doi:10.2478/aei-2021-0006

Kremer, J., Stensbo-Smidt, K., Gieseke, F., Pedersen, K. S., & Igel, C. (2017). Big Universe, Big Data: Machine Learning and Image Analysis for Astronomy. *IEEE Intelligent Systems*, *32*(2), 16–22. https://doi.org/10.1109/MIS.2017.40

Le, S. Q., & Gascuel, O. (2008). An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution*, *25*(7), 1307–1320. https://doi.org/10.1093/MOLBEV/MSN067

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature 2015 521:7553*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, Z.-K., Zhang, L.-H., Liu, B., Zhang, Z.-Y., Guo, G.-C., Ding, D.-S., & Shi, B.-S. (2022). *Deep learning enhanced Rydberg multifrequency microwave recognition. 2022.* https://doi.org/10.1038/s41467-022-29686-7

Lo, S. Y., Hang, H. M., Chan, S. W., & Li, J. J. (2018). Efficient Dense Modules of Asymmetric Convolution for Real-Time Semantic Segmentation. *1st ACM International Conference on Multimedia in Asia, MMAsia 2019.* https://doi.org/10.48550/arxiv.1809.06323

Malinverni, D., Lopez, A. J., De Los Rios, P., Hummer, G., & Barducci, A. (2017). Modeling

Hsp70/Hsp40 interaction by multi-scale molecular simulations and coevolutionary sequence analysis. *ELife*, *6*, 1–20. https://doi.org/10.7554/eLife.23471

Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., & Bateman, A. (2021). Pfam: The protein families database in 2021. *Nucleic Acids Research*, *49*(D1), D412–D419. https://doi.org/10.1093/NAR/GKAA913

Moretti, S., Laurenczy, B., Gharib, W. H., Castella, B., Kuzniar, A., Schabauer, H., Studer, R. A., Valle, M., Salamin, N., Stockinger, H., & Robinson-Rechavi, M. (2014). Selectome update: quality control and computational improvements to a database of positive selection. *Nucleic Acids Research*, *42*(D1), D917–D921. https://doi.org/10.1093/NAR/GKT1065

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1–21. https://doi.org/10.1186/S40537-014-0007-7/METRICS

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*(NeurIPS).

Proux, E., Studer, R. A., Moretti, S., & Robinson-Rechavi, M. (2009). Selectome: a database of positive selection. *Nucleic Acids Research*, *37*(suppl_1), D404–D407. https://doi.org/10.1093/NAR/GKN768

Revell, L. J. (2012). phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, *3*(2), 217–223. https://doi.org/10.1111/j.2041-210X.2011.00169.x

Schaul, T., Zhang, S., & LeCun, Y. (2013). No more pesky learning rates. *30th International Conference on Machine Learning, ICML 2013*, *PART 2*, 1380–1388.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple

Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 843–852. https://doi.org/10.1109/ICCV.2017.97

Sutto, L., Marsili, S., Valencia, A., & Gervasio, F. L. (2015). From residue coevolution to protein conformational ensembles and functional dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, *112*(44), 13567–13572. https://doi.org/10.1073/PNAS.1508584112/-/DCSUPPLEMENTAL/PNAS.1508584112.SAPP.PDF

Team, T. T. D., Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., … Zhang, Y. (2016). *Theano: A Python framework for fast computation of mathematical expressions*. https://doi.org/10.48550/arxiv.1605.02688

Wen, F., Zhang, Z., He, T., & Lee, C. (2021). AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove. *Nature Communications 2021 12:1*, *12*(1), 1–13. https://doi.org/10.1038/s41467-021-25637-w

Zeiler, M. D. (2012). *ADADELTA: An Adaptive Learning Rate Method*. http://arxiv.org/abs/1212.5701

DISCUSSION AND PERSPECTIVES

Machine Learning techniques are highly successful in many areas of science. In life science, the Machine Learning algorithm AlphaFold is one of the most impressive advances for its capabilities to predict the 3D structure of proteins from their amino acid sequence (Jumper et al., 2021). However, we are still far from comprehending the full set of mechanisms underlying these structures. One crucial piece of information used to better predict the folding of proteins are coevolving pairs of sites. Moreover, coevolving pairs of sites are involved in maintaining or improving not only structural interactions but also functional ones. In this sense, it is possible to profit from powerful Machine Learning techniques to improve our knowledge of how to infer coevolving pairs of sites and to get a step closer to understanding these biological mechanisms. In my Ph.D. thesis, I studied the limitations of some of the current methods inferring coevolution. For this, I simulated different genomic datasets with varying levels of evolutionary divergence (**Chapter 1**). I then developed a new Deep Learning model (Coev-Asymmetric-CNN) to predict coevolution, which I trained under different scenarios before testing its performance (**Chapter 2**). Moreover, I described its implementation using PyTorch and the pipeline to run Coev-Asymmetric-CNN on real datasets (**Chapter 3**). Finally, I tested its performance predicting coevolution on a real dataset from the Selectome database.

In **Chapter 1**, I benchmarked some of the current methods to infer sites that show signatures of coevolution (plmDCA, CoMap and Coev). I simulated pairs of sites under coevolution using the evolutionary model of Coev, and sites without coevolution, under the independent model of evolution LG. Finally, I studied the limitations of these methods under different levels of divergence. The results that I obtained showed that divergence has an impact on the way these methods infer coevolution and that the interpretation of the results from these methods are challenging. Even though plmDCA is outstanding compared to the evolutionary methods of Coev and CoMap, I highlighted the fact that ranking each pair of sites based on one score was not a reliable measure to infer which sites are under coevolution. The score given by plmDCA has a range that varies depending on the MSA under study, and I have shown that divergence has an impact on the value of these scores. Because plmDCA is only providing a score for each pair of sites of the MSA, it would be interesting to study how it could be possible to add a parameter to determine if the score is significant or not. In contrast, methods like Coev and CoMap, provide more than one score to predict which pairs of sites are coevolving, therefore enriching their predictions. I think that these abilities provide more robust information about each pair of sites such as the evolutionary rate or if the correlation value is significant or not. Nevertheless, it was surprising to find that even though the coevolving sites were simulated under the Coev model of co-evolution, these evolutionary methods did not perform as well as plmDCA. This may be due to the fact that the co-evolutionary model used (Coev) is adding noise when simulating coevolving pairs of sites. As shown in **Chapter 3**, Coev can simulate coevolving pairs of sites that are not under coevolution because they both change on the same but single branch of the phylogenetic tree. Thus, these changes cannot be considered as coevolution from an evolutionary perspective (based on the definition by Dutheil, 2012), because they can easily occur through chance alone. Future coevolutionary model simulators should take into account the minimum number of mutations co-occurring between pairs of sites. Additionally, other studies are simulating coevolving pairs of sites; for example, in Marmier et al., 2019, they generated controlled synthetic "sequences" as strings of values 0 or 1. They considered perfect binary trees and follow a Poisson distribution to draw the number of mutations per branch to evolve the sequences (0 and 1) along the phylogenetic tree. However, they do not base their simulations on any of the evolutionary models available in the literature, making this approach very simplistic compared with the true biological process.

Overall, while it is true that plmDCA had a good performance, I observed that it is also affected by divergence. As mentioned previously, since it only provides a score as an output for each predicted pair of sites, it is not sufficient to know how many sites are truly under coevolution. This is why I have highlighted the importance of having more knowledge about the data, and the pairs inferred under coevolution (or non-coevolution) like the Coev or CoMap methods. If the phylogenetic tree is included to infer coevolution, it is possible to estimate the level of divergence of the protein family as well as where and how many co-occurrences are happening, and the site-specific substitution rate (Dutheil, 2012). This additional data would provide a better understanding of the evolutionary process behind these models' predictions.

In **Chapter 2**, I developed a supervised machine learning method using Convolutional Neural Networks (CNN) to infer coevolution in a 2D matrix that contains the coevolutionary pattern from a multiple sequence alignment (MSA) and its phylogenetic tree. I created a training labeled dataset by simulating phylogenetic trees and MSAs under coevolution and non-coevolution. I transformed the data mapping changes into the branches of the phylogenetic tree through ancestral state reconstruction, generating 2D matrices containing the evolutionary information. After studying and discovering the pattern of the signature of coevolution I implemented a convolutional neural network using two 1D asymmetric convolutions.

One of the drawbacks of using machine learning is that it requires training labeled datasets. For coevolution, as for most applications in molecular and evolutionary biology, there are no labeled datasets available. This is why simulating coevolving data can help, as shown in other studies in which they simulated data to have a bigger and more variable dataset (Amini et al., 2022). To have a genomic dataset close enough to a real one, I based my simulations (MSAs with their phylogenetic trees) on the positive selection database Selectome.

Moreover, it is important to recognize that by simulating the genomic data and reconstructing the ancestral state for each site, we only obtain an approximation, which could bias the 2D matrix reconstruction. Nevertheless, the main concept behind creating a 2D matrix is to account for the number of changes and to make it easier to detect which branches are in common for any pairs of sites. Based on the study from Dutheil (Dutheil, 2012), we can consider that the higher the number of co-occurring changes between a pair of sites, the stronger the evolutionary pressure favoring the maintenance of mutations at both sites, and therefore, the stronger the signal of coevolution between this pair of sites. However, to my knowledge, there are no studies providing how many co-occurring changes are needed in a pair of sites to be considered a case of coevolution. Furthermore, how coevolution occurs and why this biological process is happening, are questions still under consideration. Furthermore, this new approach to generate 2D matrices, based on the MSA and a phylogenetic tree (**Chapter 2** and **Chapter 3**), highlights the signature of coevolution by increasing the values of the changes depending on the number of times the type of change has occurred. Therefore, it is easier to detect when a pair of sites have co-occurring changes. This work led to an innovation in the type of input data available to infer coevolution. Moreover, this type of data is suitable as an input for a machine learning user-friendly software tool (**Chapter 2** and **Chapter 3**) based on 1D asymmetrical convolutions, Coev-Asymmetrical-CNN.

Additionally, in **Chapter 2**, I realized that the 2D matrices and therefore, the performance of Coev-Asymmetric-CNN, are also affected by divergence along the tree, similar to what has been observed in **Chapter 1** for plmDCA, CoMap and Coev methods. Thus, I tested the CNN performance under different levels of divergence, and although the method has an accuracy up to 90% when all the levels of divergence are mixed, the method performed poorer under low levels of divergence. I observed that at a low level of divergence, the number of changes under coevolution and the independent model (non-coevolution) is very low in the 2D matrix obtained after transforming the simulated genomic data. On the contrary, when there is a high level of divergence, the number of changes occurring under coevolution or the independent model is higher. This is expected because at a low level of divergence there are more conserved sites and fewer changes occur; and for a high divergence level, there are sites that are more variable because more changes occur. To reiterate, including the phylogenetic tree allows us to better analyze these results (i.e., looking at the number of changes occurring or at the level of divergence) and to better understand the limitations of the method and the input data.

While the performance of the method is sufficient, I realized that it is a tedious process to obtain the input 2D matrix, and the divergence is still affecting the performance of the CNN. Hence, future studies should advance towards improving the model by evaluating other machine learning models. I suggest investigating Depthwise Separable Convolutional Neural Networks (Howard et al., 2017), where instead of having a 1-dimensional matrix, the input matrix could be extended to have 380 channels, each of them to map a type of change which would allow us to keep better track of the changes. The tradeoff is that it will increase the amount of memory needed to train the network. Moreover, Recurrent Neural Networks seem like a good approach to detect the signature of coevolution. The pattern is repeated through multiple branches (the columns), and if there are 2 sites under coevolution (the rows), they have a similar pattern always in the same branches. The matrix's rows could be seen as an input signal for the RNN, where it could learn to detect whenever there are similar signals in a 2D matrix. Nevertheless, my Coev-Asymmetric-CNN approach also permits extending the type of classification done. Since divergence is affecting the performance of the model, it is possible to have more than a binary classification and more types of classes based on the type of divergence.

In **Chapter 3**, I described the implementation of Coev-Asymmetric-CNN and I proposed a pipeline to run the model to detect coevolution using any genomic data. I made the code accessible to other users with user-friendly options to run it. I also illustrated the use of the model by analyzing a subset of the database of bony vertebrates provided in Selectome. Overall, in this chapter, I discovered the limitations of Coev-Asymmetric-CNN when applied to real genomic data.

Once more, there is the need to simulate the genomic data to be able to train the model to detect coevolution in a given genomic dataset. A good approach to follow would be to simulate the genomic data based on the dataset where coevolution wants to be detected. With the big dataset that I simulated in **Chapter 3**, it took approximately 3 hours to train the network and less than 1 second to predict if a new 2D matrix contains any signature of coevolution. In this sense, I am confident that it is a good approach to analyze big databases with thousands of proteins to have a quick overview of which proteins may have a signature of coevolution.

However, the more variable the data, the more the data that is needed to train the network, as discussed in **Chapter 3**. Considering the variability in terms of the number of sequences, sequence length, gaps contained in alignments and divergence, among other features, I believe it is worth further investigating the possibility to train a network based on these features. For instance, it would be possible to train three different CNNs to detect coevolution, each one for a different number of sequences: low, medium and high. In this scenario, alignments with a similar number of sequences will be clustered, therefore, the matrices will have similar sizes, avoiding the issues related to huge padding at the left of the matrix as shown in **Chapter 2**.

Nevertheless, while Coev-Asymmetric-CNN is accurate and it detects the signature of coevolution under simulated data, it is necessary to understand the genomic data in the study before drawing robust conclusions if two sites are coevolving. When I predicted 217 proteins under coevolution out of the 252 proteins filtered from the bony vertebrates Selectome dataset, I realized that understanding the reasons for these predictions are necessary. Incorporating the phylogenetic tree to infer coevolution provides extra information to better understand the results from the CNN. I observed that the maximum number of occurrences per site was having an impact on the way the CNN was identifying the signature of coevolution. In addition, the maximum number of occurrences was directly correlated with the level of divergence. Consequently, taking into account the phylogenetic tree in the analysis helps to better understand these results, which suggests that the CNN was biased by the maximum number of occurrences.

In conclusion, during the process of this thesis, I developed Coev-Asymmetrical-CNN, the first Deep Learning method to detect the signature of coevolution, while highlighting the relevance of taking into account the evolutionary history of the protein in the study. Overall, I provided experimental evidence of the potential of machine learning models to detect the signature of coevolution, which may be easy to extrapolate to other evolutionary processes such as positive selection. The results I obtained integrate the advances in machine learning models with the transformation of sequence data and their phylogenetic tree, which brings us a step closer to understanding how coevolution contributes to the evolution of molecular structure and function, and ultimately, to the creation of the biodiversity that we have on Earth.

REFERENCES

Amini, A., Wang, T.-H., Gilitschenski, I., Schwarting, W., Liu, Z., Han, S., Karaman, S., & Rus, D. (2022). Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. *2022 International Conference on Robotics and Automation (ICRA)*, 2419–2426.

Dutheil, J. Y. (2012). Detecting coevolving positions in a molecule: Why and how to account for phylogeny. *Briefings in Bioinformatics*, *13*(2), 228–243. https://doi.org/10.1093/bib/bbr048

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. http://arxiv.org/abs/1704.04861

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*. https://doi.org/10.1038/s41586-021-03819-2

Marmier, G., Weigt, M., & Bitbol, A. F. (2019). Phylogenetic correlations can suffice to infer protein partners from sequences. *PLoS Computational Biology*, *15*(10), 1–24. https://doi.org/10.1371/journal.pcbi.1007179

ANNEXES

Annex 1: **Master First-step project of Karim Saied**

Data extraction and machine learning in features involved in coevolution

December 2017

Abstract

Coevolution is a fundamental process observed at different levels in nature. At molecular level, several methods are used to detect coevolving pairs. Once done, their distribution is generally investigated in order to highlight their location within and between proteins, which provide useful structural and functional information. In this study, coevolving positions pairs in several human genes were first used to assess the relationship between coevolution and structural features in the genes products. The impact of gene conservation was also evaluated. The results showed that beta strands tend to contain a higher amount of coevolving positions compared to other secondary structures. Moreover, relatively variable proteins seem to display more related pairs than conserved proteins. As a second step, by using machine learning, another approach to investigate coevolution was introduced. As a matter of fact, predictions about coevolution scores and the distributions of positions pairs among the secondary structures have been attempted. Unfortunately, the dataset used by the machine learning algorithm suffered from a lack of information as only two features were used to predict a third one. As a consequence, none of the predictions displayed a sufficient accuracy.

Annex 2: Master First-step project of Léonard Jequier

Automatically classifying clownfish pictures datasets at the species level

December 2018

Abstract

Amphiprion species, commonly called clownfishes, are a valuable evolutionary model. Researchers are interested in computationally studying morphological variation between clownfishes species in order link theses variations to their molecular bases. To achieve this, they need as much data as possible, consisting of pictures of clownfish specimen labeled with the species name. Currently, labelling the picture is done manually and require a lot of time and expertise. The aim of this project is to create a program capable of providing species label to a sets of clownfish pictures. To do so, the best currently available image recognition technology was used: convolutional neural networks. Among the created programs, one can distinguish between pictures of A. clarkii and A. perideraion with an accuracy of 96%. Another can distinguish A. ocellaris and A. frenatus in addition to the two species mentioned before with an accuracy of 89%.