RAPPORT DE STAGE

Deep Learning pour la phylogénie et l'alignement de séquences

Master DAC

PHO Vinh-Son

Encadrants : Olivier Gascuel - Mathilde Carpentier

Table des matières

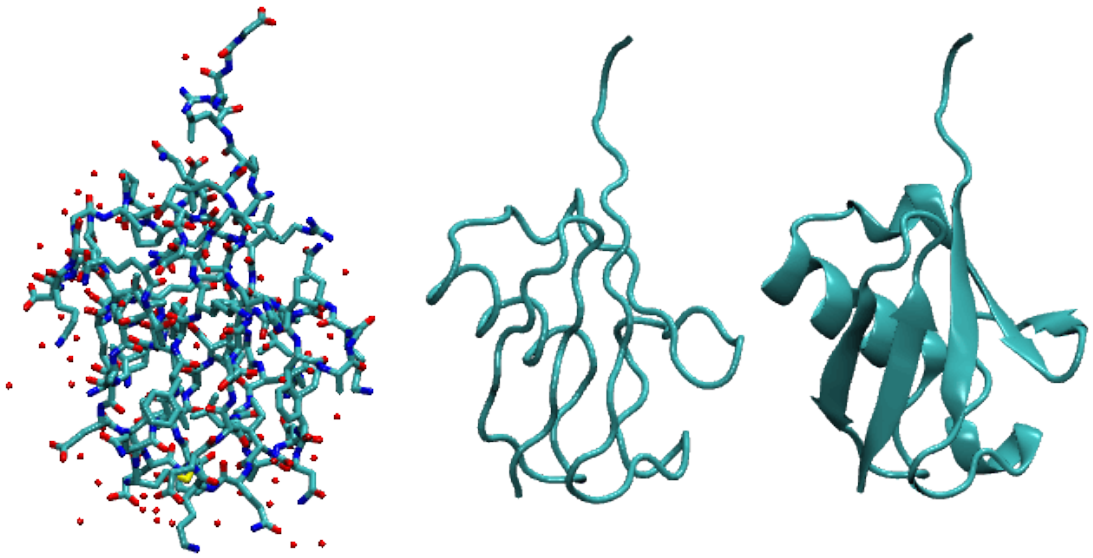


Figure 1: Différentes représentations 3D de la protéine Ubiquitine, de gauche à droite : Licorice, Tube et NewCartoon

1 Introduction

1.1 Contexte biologique

1.1.1 Protéines

Les protéines sont responsables de presque toutes les tâches de la vie d'une cellule, telles que l'organisation interne, la gestion des déchets, le transport de molécules... Par exemple, l'Ubiquitine est une protéine présente chez la plupart des eucaryotes dont l'humain. Elle remplie ses diverses fonctions en se conjuguant à d'autres protéines. On peut la représenter par sa séquence d'acides aminés : "MQIFVKTLTGKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAGKQLEDGRTLSDYNIQRESTLHLVLRLRGG" ou bien par sa structure (voir figure 1).

Dans le cadre de ce stage, nous ne nous intéressons qu'à leur séquence d'acides aminés. Il est quand même important de noter que l'arrangement dans l'espace de la protéine aux échelles atomiques et moléculaires contient beaucoup d'information sur la fonction biologique de cette protéine.

Une séquence d'acides aminées caractérise la protéine à laquelle il correspond. Il y a au total 20 acides aminés standards, tous encodés par un ou plusieurs triplets de nucléotides, élément constitutif de l'ADN. Cette séquence est donc codée par l'ADN qui va accumuler des mutations au fil du temps. Ces mutations dans l'ADN peuvent transparaître dans les protéines qu'il code. Ce sont ces mutations dans les protéines qui vont nous intéresser. Elles

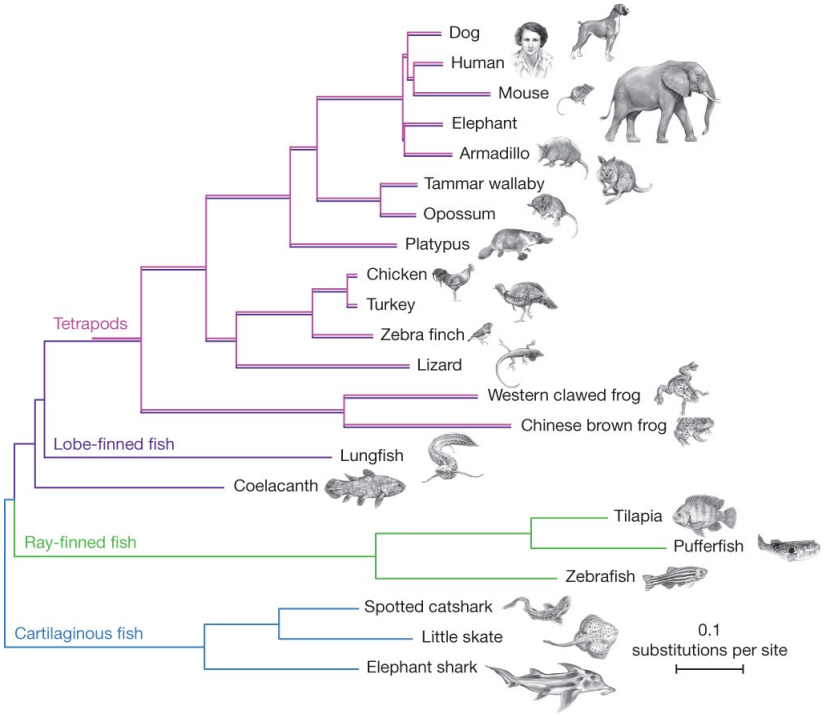


Figure 2: Exemple d'arbre phylogénétique (Amemiya et al. 2013)

apparaissent de 3 manières distinctes :

1. le remplacement d'un acide aminé par un autre dans la séquence, appelé substitution
2. l'insertion d'un bloc d'acides aminés dans la séquence
3. la délétion d'un bloc d'acides aminés dans la séquence

1.1.2 Phylogénie

La phylogénie est l'étude des liens de parentés entre différents taxons, ces taxons peuvent être des êtres vivants, des virus ou même des protéines. Les protéines étant créées par les êtres vivants, étudier les protéines permets à terme de mieux comprendre la phylogénie des différents taxons qui est représenté par un arbre phylogénétique (voir figure 2). Si l'on utilise des protéines pour réaliser cette étude, une étape importante est d'aligner ces séquences, c'est-à-dire créer une matrice (voir table 1) ou chaque ligne correspond à une séquence et chaque colonne correspond à un ensemble d'acides aminés descendant d'un même acide aminé ancestral. Comme les séquences ne sont pas de la même longueur à cause des insertions et délétions, il faut donc identifier où elles ont eu lieu. À partir d'un alignement, il est usuel de créer un profil, c'est-à-dire une séquence contenant, pour chaque colonne de l'alignement, les fréquences d'apparition des 20 acides aminés dans cette colonne.

L'ADN codant les protéines ne se conservant pas très bien, nous n'avons accès qu'à des

seq_1	.	.	.	N	A	L	E	S	G	H	I	A	G	A	A	I	D	V	F	.	.
seq_2	A	L	C	D	A	L	A	S	.	.	.	A	G	A	A	I	D	V	F	P	T
seq_3	A	L	A	Q	A	L	K	D	.	.	.	.	.	.	A	I	D	V	F	P	V

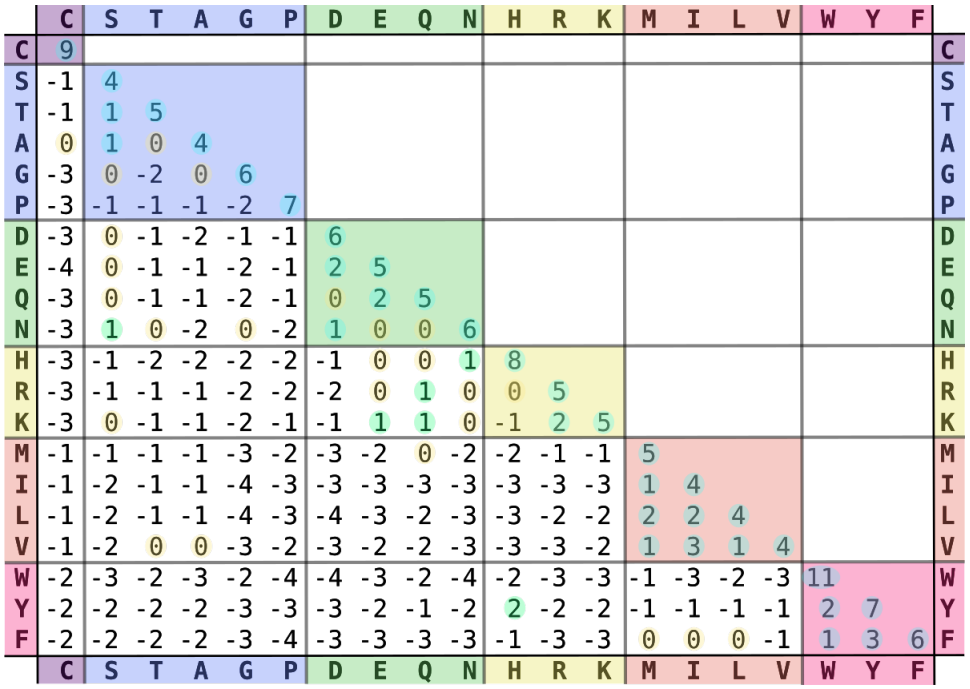
Table 1: Exemple d'alignement de séquence, les insertions et délétions sont notées par un "."

protéines récentes, âgées au plus d'1 à 2 millions d'années. Nous faisons alors l'hypothèse de la "time reversibility", c'est-à-dire que l'on va supposer que pour tout couple de protéines homologues, c'est-à-dire descendantes d'une même protéine ancestrale, on peut supposer l'une ou l'autre comme ancestrale sans que cela impacte les résultats. Plus formellement, étant donné deux acides aminés X et Y, on suppose $P(X)P(X \to Y) = P(Y)P(Y \to X)$, avec $\to$ notant la substitution d'un acide aminé vers un autre.

1.2 État de l'art

Depuis l'invention du séquençage de l'ADN en 1970 par Sanger, les progrès technologiques du domaine sont tels que les quantités de séquences biologiques ont augmentés de manière exponentielle. Ce volume de données permet l'utilisation de nouvelles techniques telles que le *Deep Learning*. La plupart des approches de *Deep Learning* se basent sur les réseaux de neurones artificiels tels que les Transformers ou les *Convolutional Neural Networks*. Ces réseaux de neurones, inspirés par le fonctionnement des neurones biologiques, sont un ensemble d'unités connectées entre elles appelées neurones artificiels. Chaque connexion, telle que les synapses du cerveau, peut transmettre un signal. Chaque neurone reçoit un signal, le transforme, puis le transmet aux autres neurones qui lui sont connectés. Les signaux sont des nombres réels et les transformations sont des compositions de fonctions linéaires avec d'autres fonctions non-linéaires appelées activations. On entraîne les poids des transformations du réseau en traitant des exemples et en calculant l'erreur entre la sortie produite et la sortie attendue.
Les Transformers [1] qui nécessitent une quantité très grande d'exemples pour bien "apprendre" leur poids sont de plus en plus utilisés en biologie grâce aux milliards de séquences biologiques produites ces dernières années. On peut citer notamment Alphafold[2] qui est utilisé pour prédire la structure (le repliement des protéines) à partir de la séquence d'acides aminés. D'autres approches similaires de plus en plus utilisée pour l'étude des protéines sont les *Protein Language Models*[3][4][5], qui sont l'application des *Large Language Models* au monde des protéines. Ces Modèles permettent de créer pour chaque acide aminé de la séquence un vecteur décrivant le contexte local et global de cet acide aminé dans la séquence. Ces vecteurs sont à leur tour très utiles, car ils permettent d'identifier la fonction biologique ou la structure des protéines données en entrée.

Pour l'instant, personne ne s'est posé la question de l'application du *Deep Learning* pour estimer des probabilités de substitutions. De fait, les meilleurs modèles de nos jours sont très simples et ne comportent que quelques centaines de paramètres. Parmi ces modèles, on peut citer notamment les matrices de substitutions ou celles de taux de transitions.



	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

Figure 3: Matrice BLOSUM62, coloriée selon la classification des acides aminés de Dayhoff

Les matrices de substitutions, telles que Point Accepted Mutation ou BLOcks SUbstitution Matrix[6], figure 3, donnent pour chaque paire d'acide aminé (X, Y) un score, plus ce score est élevé, plus la substitution est fréquente. La matrice BLOSUM est calculée empiriquement à partir d'alignement avec la formule suivante :

$$BLOSUM(X,Y) = 2\log_2(\frac{P(X \to Y)}{P(X)P(Y)})$$

Ces matrices de substitution sont notamment utilisées pour l'alignement de séquences.

Un inconvénient des matrices précédentes est qu'elles ne prennent pas en compte une information essentielle, la distance évolutive entre deux séquences. Les matrices de transitions, pour utiliser cette information, se placent dans le cadre d'un processus stochastique. Elles supposent que les sites d'une séquence évoluent indépendamment et que le processus des substitutions est continu et homogène au cours du temps. Elles modélisent alors les substitutions comme une chaîne de Markov à temps continu à 20 états correspondant chacun à un acide aminé standard.
Puisque nous faisons l'hypothèse de la "time reversibility", on peut voir la distance évolutive t comme une durée. On note $M_{ij}(t) = Pr(X_t = j | X_0 = i)$ la probabilité qu'à l'instant t on observe l'acide aminé j, en partant de l'acide aminé i au temps 0.
En supposant $\frac{M(t)}{dt} = M(t) \cdot Q$, où Q est la matrice de taux de transition, on obtient :

$$M(t) = e^{Qt}$$

La matrice de taux de transition Q est calculée de façon à utiliser la distance évolutive t entre les 2 séquences. Cette distance évolutive est définie comme le nombre moyen de substitutions par site (position) entre les 2 séquences. Cette distance est néanmoins impossible à calculer, car pour un site donné, si l'on observe un acide aminé A sur la première séquence et un acide aminé D sur la deuxième, on ne peut pas savoir s'il y a eu une substitution ($A \to D$ ou $D \to A$) ou plusieurs ($A \to C \to D$ par exemple). Les substitutions de A vers C et de C vers D sont des substitutions cachées. Nous pouvons quand même obtenir une estimation par maximum de vraisemblance de cette distance avec le logiciel FastME [7], entre autres. Il existe de nombreuses matrices de taux de transitions WAG[8], JTT[9], LG[10]... Ces matrices sont estimées par maximum de vraisemblances sur divers jeux de données. Un de leurs défauts est qu'elles ne prennent pas du tout en compte le contexte biologique au sein de la protéine pour estimer la probabilité de substitution. Cette information est pourtant très importante pour les substitutions. Par exemple, si les acides aminés voisins de X et Y sont hydrophobes, alors X et Y sont probablement hydrophobes. C'est pourquoi nous voulons évaluer le gain d'information possible en créant un modèle prenant en compte ce contexte.

1.3 Objectif

L'objectif de ce stage est d'améliorer les méthodes d'estimation des probabilités de substitution, c'est-à-dire étant donnés deux acides aminés (X, Y) et Z des informations contextuelles, estimer $P(X \to Y|X, Z)$. Par la suite, nous prendrons pour informations contextuelles Z, les acides aminés voisins de X et Y, une similarité entre les protéines contenant X et Y, mais aussi les fréquences d'apparition des acides aminés dans ces protéines, mais aussi dans l'ensemble des séquences qui leur sont alignées. Par exemple (sur la figure 2), si l'on prend un acide aminé d'une séquence provenant d'un éléphant pour X, et l'acide aminé de la séquence homologue chez le tatou (armadillo) pour Y, nous voulons estimer la probabilité que l'acide aminé X présent chez les éléphants se substitue en l'acide aminé Y présent chez le tatou (armadillo). Ce stage vise aussi à identifier quelles informations contextuelles sont réellement utiles pour l'estimation, étant donné que les méthodes actuelles ne prennent peu ou pas en compte ces informations. Enfin, nous voulons appliquer ces nouvelles méthodes pour l'alignement de séquences.

2 Données

Pour créer un modèle empirique d'estimation de probabilité de substitution, nous avons besoin d'un certain type de données : des alignements de séquences.

2.1 Pfam

Pfam[11] est une banque de données de référence pour les alignements de séquences, elle contient plus de 19632 alignements, soit plus d'un million de séquences seed et 61 millions

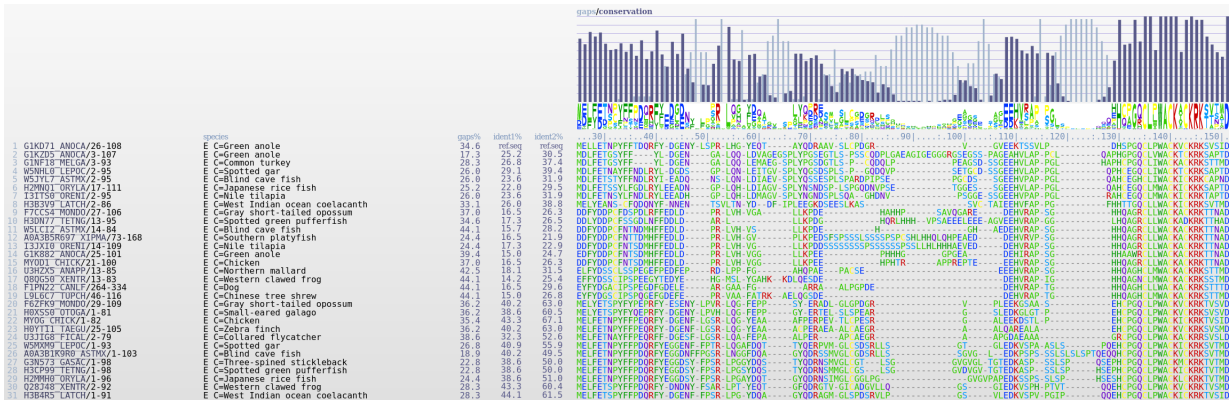


Figure 4: Visualisation d'un alignement de protéines

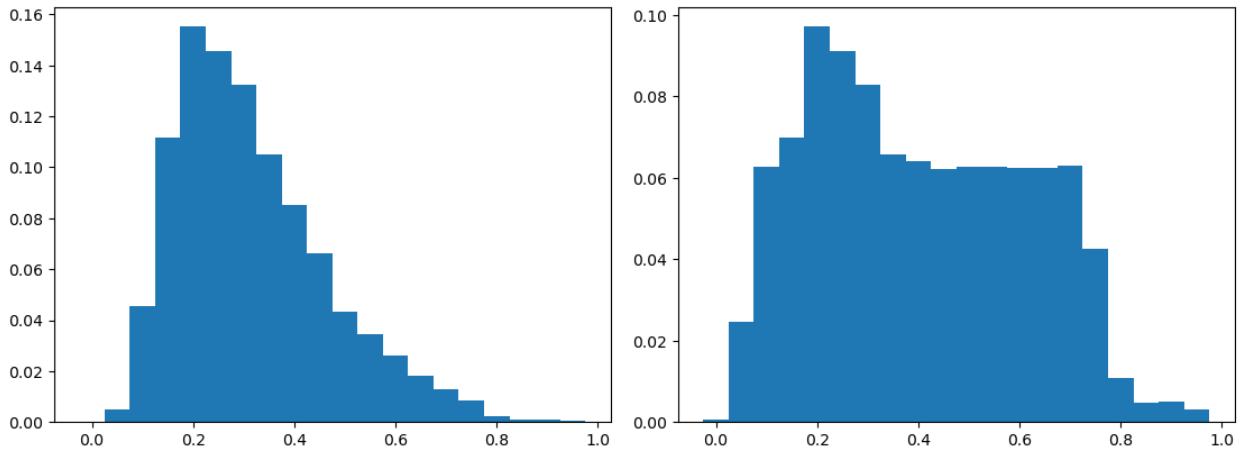
de séquences au total. Ces séquences sont des protéines, c'est-à-dire des séquences d'acides aminés. Le vocabulaire de ces séquences ne contient que 22 caractères : les 20 lettres représentant les 20 acides aminés standards, le caractère "inconnu", le caractère de l'*indel* (usuellement "." ou "-") indiquant une insertion ou une délétion a eu lieu par rapport à l'alignement. Chaque élément de la séquence peut donc être encodé par un vecteur one-hot à 21 dimensions, le mot inconnu étant représenté par le vecteur nul.

Pfam contient 19632 fichiers, soit un par alignement. Un alignement (aussi appelé famille) est un ensemble de séquences présentant de fortes similarités structurelles et/ou fonctionnelles, on suppose que toutes ces séquences descendent d'une même séquence ancestrale. On organise l'alignement en une matrice (voir figure 4), où chaque ligne correspond à une protéine et chaque colonne correspond à un ensemble d'acide aminés descendants d'un même acide aminé ancestral.

Bien évidemment, pour tester les performances de nos modèles, nous séparons notre jeu de données en un jeu d'apprentissage, un de validation et un pour le test. Nous faisons le choix de réaliser cette séparation par famille. Cette séparation permet d'assurer, grâce au score obtenu sur le jeu de données de test, que les modèles appris ont bien réussi à généraliser pour de nouvelles familles et qu'ils seront effectivement utiles pour de futures utilisations.

2.2 Prétraitements

Pfam contient pour chaque famille 2 alignements, l'alignement "seed", vérifié manuellement, et les alignements "complet" qui sont générés automatiquement autour de l'alignement "seed" en cherchant dans une banque de séquence. Nous choisissons de ne garder que les alignements "seed" qui sont généralement considérés comme étant de meilleure qualité. Nous avons quand même un grand nombre d'exemples, si l'on compte l'ensemble des paires d'acides aminés alignés, nous en avons de l'ordre de 77 milliards. Comme nous ne comptons pas tous les voir, nous proposons de tirer aléatoirement un certain nombre d'exemples à chaque batch.



(a) Histogramme de fréquence du PID du dataset d'origine (b) Histogramme de fréquence du PID du dataset biaisé

Figure 5: Distribution du PID dans le jeu de données avant et après ajout de biais

Pour le tirage des données, dans un premier temps, pour éviter la sur-représentation de certaines familles dans la distribution des données, nous proposons de biaiser l'échantillonnage en tirant, pour chaque famille, un nombre d'exemples proportionnel à la longueur de l'alignement et proportionnel au nombre de membres de la famille. Ce tirage nous permet d'éviter que les familles petites en longueur et en nombre de membres soient sur-représentées dans la distribution des données.

Dans un second temps, nous voulons nous assurer que les modèles produits soient utiles pour de futures applications. Nous voulons donc qu'ils soient entraînés sur des données proches des données qui seront utilisées en application. En général, les protéines qui seront utilisées en application sont relativement proches. Pour quantifier la similarité entre deux séquences s_1 et s_2 nous allons utiliser le Pourcentage d'IDentité (noté PID) qui est calculé de la manière suivante :

$$PID(s_1, s_2) = N_{s_1,s_2}/Len(s_1, s_2)$$

Où N_{s_1,s_2} est le nombre de positions où les deux acides aminés sont identiques et différents de l'*indel* et $Len(s_1, s_2)$ est le nombre de colonnes dans l'alignement où ni s_1 ni s_2 ne sont des *indel*.

Quand on observe la distribution du PID (figure 5a) dans Pfam, nous observons que la majorité des paires de séquences ont un PID assez faible (< 0.5). Pour remédier à cela, nous biaisons de nouveau le tirage en augmentant la fréquence de tirage (au sein d'une famille) des tranches de PID peu présentes dans le jeu de données. Nous arrivons donc à la distribution suivante (figure 5b) où l'on tire beaucoup plus de paire de séquences avec un PID relativement élevé ($0.4 <$PID< 0.7). Ce sont ces paires des séquences qui nous intéressent le plus pour de futures applications.

Enfin, nous choisissons de retirer les protéines membranaires du jeu de données. Ces protéines existent dans la membrane cellulaire et n'ont donc pas la même composition que les autres

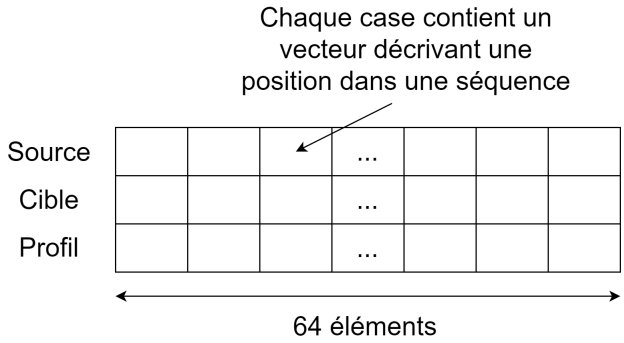


Figure 6: Données d'entrée

protéines. Ces protéines sont généralement plus hydrophobes que les autres protéines et très différentes de ces dernières. Nous avons donc retiré du dataset les familles portant une annotation GO [12] associée à la membrane cellulaire.

3 Approches

Pour évaluer le gain d'information possible pour l'estimation de probabilité de substitution en ajoutant de l'information contextuelle, nous allons procéder par étude d'ablation. Dans un premier temps, nous souhaitons réaliser le meilleur modèle possible, et ensuite, nous retirerons certaines informations contextuelles.

En plus de l'acide aminé d'origine X, nous donnons aux modèles comme informations contextuelles, 30 acides aminés à gauche et à droite de X et de l'acide aminé à prédire Y. Les acides aminés de ces 2 séquences sont encodées en One-hot. Nous considérons une troisième séquence notée $profile$ contenant, pour chacune des 61 positions considérées, un vecteur composé des fréquences d'apparition des acides aminés dans la colonne de l'alignement correspondant à la position considérée. Nous ajoutons à chacune de ces 3 séquences 3 éléments : le PID, les fréquences d'apparition des acides aminés sur cette séquence et les fréquences d'apparition sur l'alignement entier. Ces fréquences ne prennent pas en compte de l'acide aminé Y que l'on veut prédire. L'entrée du réseau est donc un tenseur de dimensions [3, 64, 21] (voir figure 6).

Par la suite, nous notons les séquences contenant X, respectivement Y, src respectivement tgt

Tous les modèles présentés ont été implémentés avec PyTorch, et ont été entraînés par la rétro-propagation du gradient de l'erreur cross-entropique entre la prédiction $\hat{Y}$ et la vérité Y pour plus de détails, voir section B.

3.1 Perceptron multicouche

Pour nous assurer que les réseaux de neurones étaient bien adaptés à la tâche à réaliser, nous avons utilisé en première approche un perceptron multicouche sur l'aplatissement (transformation en un vecteur unidimensionnel) des 2 séquences X et Y encodées en one-hot. Nous avons testé plusieurs architectures, la plus performante avait pour dimensions des couches cachées [512, 256, 32] et avait comme fonction d'activation des GELU [13].
Cette approche nous a présenté un gain conséquent par rapport à l'état de l'art (table 2). Cependant, elle n'est pas du tout adaptée à nos données séquentielles. En effet, elle a été faite pour traiter des vecteurs et non des séquences. De plus, elle n'est pas faite pour extraire l'information issue des relations entre éléments d'une séquence.

3.2 Attention

Pour mieux prendre en compte la nature des données, nous choisissons d'utiliser un mécanisme d'attention[1] déjà très utilisé en génomique [3][4][5]. Cette approche permet d'une part de mieux prendre en compte les interactions entre les acides aminés et d'autre part de mieux prendre en compte la structure des données. La self-attention étant une opération d'ensemble, nous rajoutons à chaque élément de chaque séquence l'information de sa position relative par rapport à la position considérée, encodée de la façon suivante : En notant $pos(aa)$ la position de l'acide aminé aa dans la séquence, $PosEncoding(aa)$, l'encoding positionnel de l'acide aminé aa:

$$PosEncoding(aa) = \text{Affine}(\frac{pos(aa) - pos(X)}{30})$$

Avec Affine: $\mathbb{R} \to \mathbb{R}^d$ une transformation affine apprise par le réseau et d la dimension d'embedding. Enfin, nous ajoutons $PosEncoding$ à l'embedding. Nous avons essayé d'utiliser un encoding positionnel plus complexe, tel que celui utilisé dans l'article présentant le mécanisme d'attention, mais cet encoding n'a pas donné de meilleurs résultats, en plus d'être plus éloigné de la réalité biologique.

Le score d'attention est calculé de la façon suivante :

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

où Q, K et V sont des matrices contenant des projections linéaires des éléments de la séquence d'entrée, et d_k et le nombre de dimensions de la projection de Q et K. Nous utilisons l'attention multi-têtes consistant à paralléliser plusieurs calculs d'attention avant de combiner linéairement les scores obtenus.

En empilant plusieurs couches d'attention multi-têtes, nous pouvons créer un modèle similaire en architecture à BERT.

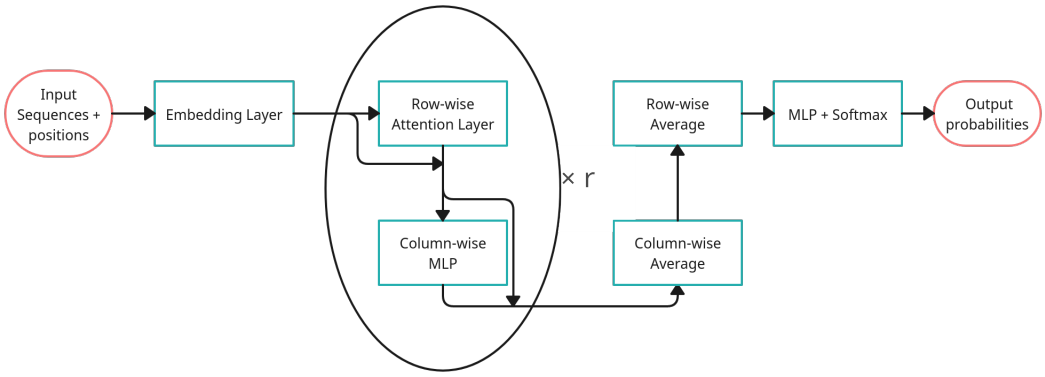


Figure 7: Architecture du Transformer utilisant l'attention semi-axiale

3.3 Attention semi-axiale

Pour bien prendre en compte l'information d'alignement, nous avons adapté l'attention axiale introduite par Alphafold[2].

Étant donné une famille de protéines alignées (à voir comme une matrice), l'attention axiale consiste à appliquer une couche d'attention dite "row-wise" sur les lignes de la matrice, puis une couche d'attention dite "column-wise" sur les colonnes de la matrice et enfin un MLP "element-wise" sur chaque élément de la matrice. L'opération "row-wise" permet d'extraire l'information contenue dans les interactions entre acides aminés d'une même séquence. l'opération "column-wise" permets quant à elle d'extraire l'information contenue dans les colonnes de l'alignement.

Dans notre cas, nous avons 3 séquences alignées (*src*, *tgt*, profil) jouant 3 rôles différents. Appliquer une couche d'attention sur une séquence de 3 éléments n'ayant pas vraiment de sens, nous remplaçons l'attention "column-wise" et le MLP "element-wise" de l'attention axiale par un MLP "column-wise".
Notre nouvelle couche, appelée attention semi-axiale, consiste d'une couche d'attention "row-wise" suivie d'un MLP "column-wise". Cette approche possède 2 avantages principaux par rapport à l'attention axiale, elle est beaucoup plus rapide et permet au modèle de distinguer les éléments des 2 séquences *src* et *tgt*.

Pour combler le trou dans la séquence *tgt* laissé par l'absence de Y nous rajoutons $f(X)$ avec f linéaire. Pour obtenir un vecteur à la sortie des couches d'attention, nous effectuons une moyenne pondérée (poids appris) sur les 2 séquences puis une moyenne pondérée sur les éléments de la séquence. Enfin, pour l'estimation de probabilité, nous utilisons une couche linéaire suivie d'un softmax.

Pour ce Transformer semi-axial (voir figure 7), nous avons testé différents embedding, un embedding one-hot et un embedding à {32, 64, 128, 256, 512} dimensions. Nous observons les meilleures performances avec un embedding à 128 dimensions, 12 modules d'attention semi-axiale. Chaque module utilise 4 têtes d'attention à 64 dimensions chacune, un MLP

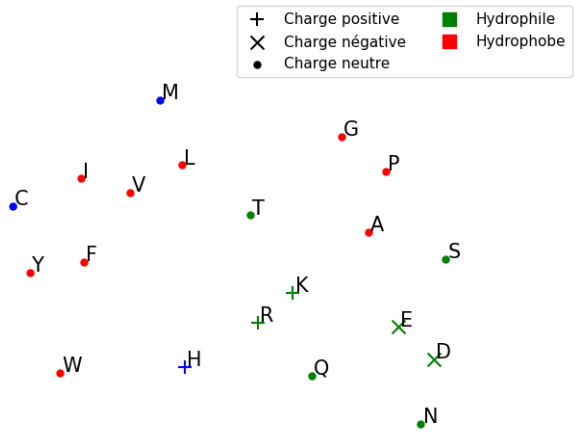


Figure 8: Visualisation des Embeddings par t-SNE[14]

"column-wise" avec un "widening factor" de 2, et pour la classification finale, une couche linéaire suivie d'un softmax.
Étant donné une entrée de taille $[s, n, d]$ composée de s séquence alignée, de longueur n, avec un embedding par position de d dimensions, les n entrées du MLP "column-wise" est un tenseur de dimension $[s, d]$ que l'on aplatit en vecteur de dimension $[s \times d]$. Si l'on utilise un widening factor de 2, alors la dimension cachée du MLP est $s \times d \times 2$. On réarrange les n sorties de tailles $[s \times d]$ en un tenseur de la taille d'origine $[s, n, d]$. Comme nous utilisons un widening factor de 2 et un embedding de taille 128, la dimension cachée de ces MLP est 512 pour 2 séquences d'entrée et 768 pour 3 séquences.

Comme nos données se composent de 22 caractères (20 acides aminés, *indel* et "inconnu"). Nous encodons ces caractères avec un encoding One-Hot à 21 dimensions (le mot "inconnu" étant encodé par un vecteur nul). Nous concaténons à ce vecteur l'encoding positionnel discuté précédemment. Enfin, pour créer un embedding, nous appliquons une couche linéaire, cela nous permet de créer un embedding aussi pour les profils qui sont des fonctions de densité discrète.

On peut remarquer sur la figure 8 que les embeddings appris semble fortement corrélés à des propriétés physiques et chimiques des acides aminés. De fait, nous pouvons remarquer que les acides aminés similaire selon la classification de Dayhoff ou la matrice BLOSUM62 (figure 3) ont des embeddings similaires.

3.4 Méthodes interprétables

3.4.1 PID et probabilités conditionnelles

Le PID est la fréquence d'acides aminés alignés et identiques dans les 2 séquences. Il correspond à $P(X = Y)$. En le combinant naïvement aux fréquences conditionnelles : $P(X \to Y)$.

On peut arriver à un modèle f très simple :

$$f(X, PID) = \begin{cases} PID & \text{si } X = Y \\ (1 - PID) \times \frac{P(X \to Y)}{1 - P(X \to X)} & \text{sinon.} \end{cases}$$

À partir de cette formule, on peut créer deux modèles : un dont la matrice de fréquences conditionnelles est calculée sur tout l'ensemble d'entraînement et le PID est calculé sur la paire de séquences alignées. Un second va calculer une estimation du PID sur les 2 séquences alignées avec réseaux de neurones et apprendre une matrice de fréquences conditionnelle par rétro-propagation du gradient.

3.4.2 Estimation de distance et matrice de taux de transitions

Pour créer un modèle plus interprétable, on peut apprendre une matrice à taux de transition Q de taille (20× 20) et estimer une distance évolutive t. Finalement, les probabilités de substitutions sont obtenues avec la formule suivante :

$$P(X \to Y|X) = X \cdot e^{Qt}$$

Nous choisissons de réaliser l'estimation de distance avec un modèle basé sur l'attention semi-axiale.

4 Application : Alignement de séquences

Pour aligner 2 séquences *src* et *tgt* non-alignés, nous utilisons l'algorithme Needleman-Wunsch [15] qui consiste à remplir une matrice de taille $N \times M$ avec
$N, M = length(src), length(tgt)$ avec la formule de récurrence suivante.

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + w(src[i], tgt[j]) & \text{(cas 1)} \\ S_{i-1,j} - g & \text{(cas 2)} \\ S_{i,j-1} - g & \text{(cas 3)} \end{cases}$$

Une fois la matrice d'alignement calculée, il faut alors trouver le chemin emprunté par les max allant de $S_{N-1,M-1}$ à $S_{0,0}$. Pour la valeur $S_{i,j}$, le cas 1 correspond à ce que la position i de la séquence *src* soit alignée avec la position j de la séquence *tgt*. Le cas 2 correspond à ce que la position j de *tgt* soit aligné à un *indel*. Et le cas 3 correspond à ce que la position i de *src* soit aligné à un *indel*.

Cet algorithme a besoin d'un système de score, c'est-à-dire un score $w(a, b)$ nous indiquant la propension d'un acide aminé à muter en un autre (étant donné le contexte ou non) et d'une pénalité g pour l'ouverture ou l'extension d'un gap. La matrice BLOSUM62, habituellement utilisée comme score $w(a, b)$ pour cet algorithme, utilise la formule :

$$BLOSUM(X, Y) = \lambda \log \frac{P(X \to Y)}{P(X)P(Y)}$$

Nous proposons donc d'utiliser le score suivant :

$$Score(X,Y) = \lambda \log \frac{P(X \to Y)}{P(X)P(Y)}$$
$$Score(X,Y) = \lambda \log \frac{P(X \to Y|X)}{P(Y)}$$
$$Score(X,Y) = \lambda \log \frac{f(X)[Y]}{P(Y)}$$

Où $f(X)[Y]$ est la prédiction $P(X \to Y|X)$ de notre modèle estimateur, et λ un facteur constant permettant d'ajuster les scores par rapport aux pénalités.
Nous choisissons d'avoir deux pénalités de gap différentes :

$$g = \begin{cases} g_{op} := 10 & \text{(pour l'ouverture d'un gap)} \\ g_{ext} := 1 & \text{(pour l'extension d'un gap)} \end{cases}$$

Ce choix, assez standard pour aligner des séquences, nous oblige à modifier légèrement l'algorithme de Needleman-Wunsch pour pouvoir utiliser une pénalité d'ouverture et une d'extension. Nous aussi avons essayé d'utiliser différentes valeurs pour des pénalités $g_{op} = g_{ext}$, mais elles donnent de moins bons résultats pour cet algorithme avec la matrice BLOSUM62.

4.1 Mesures de similarité

Pour aligner 2 séquences, savoir si deux séquences sont proches ou éloignés est une information importante. Le PID jusqu'à présent était une très bonne estimation de la similarité entre 2 protéines, mais elle suppose le fait d'avoir accès aux 2 séquences alignées. Nous avons donc besoin d'une mesure de similarité/dissimilarité fonctionnant sur des séquences de tailles différentes, non alignées et pouvant présenter des trous dont nous n'avons pas conscience.

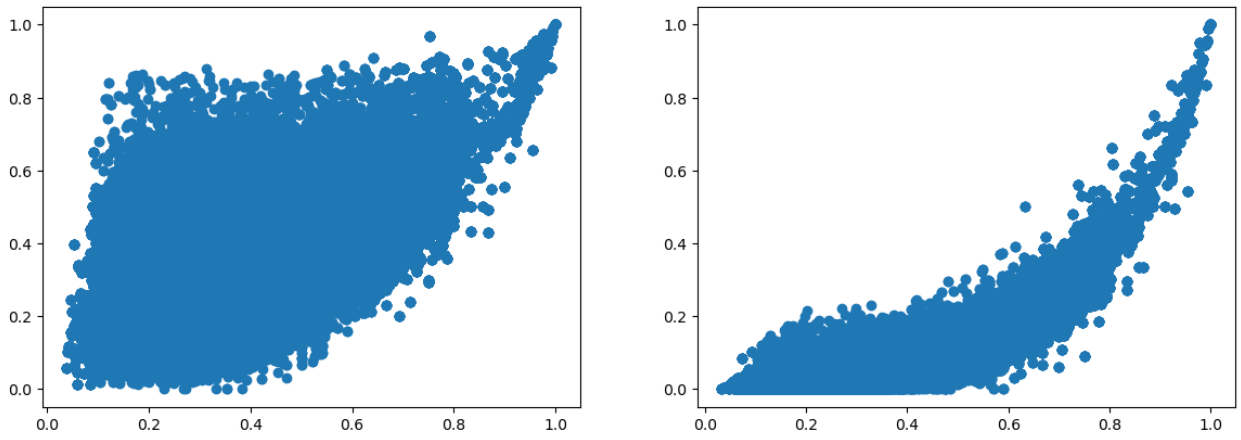
4.1.1 Similarité de Jaccard

Une façon de calculer une mesure de similarité entre deux séquences est de comparer leur ensemble de k-mer (k-uplets) avec la similarité de Jaccard.

$$JAC(src,tgt) = \frac{|kmer(src) \cap kmer(tgt)|}{|kmer(src) \cup kmer(tgt)|}$$

avec $kmer(seq)$ dénotant l'ensemble des k-mer (k-uplets) de la séquence seq. La méthode MUSCLE [16] propose une variante de la similarité de Jaccard, encore plus corrélée au PID.

$$JAC(src,tgt) = \sum_{\gamma} \frac{\min[count(\gamma,src), count(\gamma,tgt)]}{\min[length(src), length(tgt)] - k + 1}$$



(a) Similarité 2-mer

(b) Similarité 3-mer

(c) Similarité 4-mer

(d) Similarité 6-mer

Figure 9: Similarité basée sur les k-mer en fonction du PID pour différentes valeurs de k

Où γ est un k-mer, $count(\gamma, seq)$ compte le nombre d'occurrences de γ dans la séquence seq.

Ces mesures de similarité ont plusieurs problèmes (figure 9) :

1. pour k < 3 : les mesures ne sont pas du tout corrélées avec le PID.
2. pour k > 2 : les mesures sont corrélées avec le PID si les séquences sont similaires (PID > 0.6)
3. pour k > 5 : les mesures sont presque nulles pour des séquences non-similaires (PID < 0.7)

Ces problèmes nous empêchent donc d'utiliser ces mesures pour de l'alignement, étant donné que les séquences que l'on veut aligner peuvent être très différentes (PID < 0.5).

4.1.2 Cross-corrélation

La cross-corrélation est une approche utilisée pour la reconnaissance de motifs dans des images ou des signaux. C'est d'ailleurs cette opération (et non la convolution) qui est implémentée généralement dans les *Convolutional Neural Networks*.

La méthode MAFFT [17] propose, pour comparer deux séquences d'acides aminés non-alignées, d'effectuer la cross-corrélation sur les listes des polarités et volumes des acides aminés de ces deux séquences (sans *indel*):

$$CCORR(src, tgt, k) = \sum_i f(src[i]) \times f(tgt[i+k])$$

Où $f(a)$ correspond à une propriété physique ou chimique de l'acide aminé a.

Étant donné que les embeddings appris (voir section 3.3) sont corrélés à des propriétés physiques et chimiques des acides aminés. nous proposons de calculer les mesures de similarité suivantes :

$$CCORR(src, tgt, k, e) = \sum_i Embedding(src[i])[e] \times Embedding(tgt[i+k])[e]$$

Avec $Embedding(aa)[e]$ la e-ième dimension de l'embedding appris de l'acide aminé aa. Nous choisissons de normaliser cet embedding, pour que les scores de cross-corrélation soit en général compris dans $[-1, 1]$.

Et nous proposons de ne garder que $CCORR(src, tgt, k^*, e)$ avec :

$$k^* = \arg\max_k \sum_e CCORR(src, tgt, k, e)$$

Notre score de similarité est donc :

$$SIM(src, tgt, e) = CCORR(src, tgt, k^*, e)$$

Il est intéressant de noter que la cross-corrélation est très similaire à la convolution. Donc, étant donné que $N = lenght(src) \approx length(tgt)$, le calcul de la cross-corrélation à une complexité temporelle en $O(N^2)$. En utilisant l'algorithme de la transformée de Fourier rapide (FFT) et le théorème de la convolution, on peut réduire cette complexité à $O(N \log N)$. En pratique, cette réduction de la complexité se traduit par une réduction du temps de calcul quand les calculs sont réalisés sur le CPU, mais à une augmentation du temps de calcul sur GPU. Cette augmentation est probablement due au fait que l'implémentation de l'algorithme de calcul de la cross-corrélation en $O(N^2)$ (`torch.nn.functional.conv1d`) est très optimisé sur GPU, par rapport à l'alternative en $O(N \log N)$.

Nous allons par la suite remplacer le PID utilisé auparavant dans l'apprentissage des modèles par ces 128 scores de cross-corrélation $SIM(src, tgt, e)$, $e \in [0, 127]$.

5 Résultats

5.1 Estimation des probabilités de substitution

Nous choisissons d'utiliser le score de Brier, une généralisation de l'erreur des moindres carrés aux vecteurs de probabilités, pour évaluer les performances de nos modèles.

$$Brier(\hat{Y}, Y) = \frac{1}{N} \sum_{i}^{N} \sum_{k}^{C} (\hat{Y}[i][k] - Y[i][k])^2$$

Avec $\hat{Y}$ les probabilités produites par l'estimateur et Y encodé en One-hot, N le nombre d'exemples et C le nombre de classes. Plus le score de Brier est faible, plus les prédictions sont bonnes. Ce score, par rapport à l'erreur cross-entropique, présente l'avantage d'être borné, en effet $Brier(\hat{Y}, Y) \in [0, 2]$.

Modèle	Score de Brier	Données d'entrée du modèle
Matrice LG et distance de FastME (état de l'art)	0.76	X et distance évolutive
PID et probabilités conditionnelles	0.74	X et PID
Matrice de taux de transition et distance locale estimée par réseau de neurones	0.71	src et tgt et PID
Probabilités conditionnelles et PID local estimé par réseau de neurones	0.71	src et tgt et PID
Perceptron multicouche	0.71	src et tgt et PID
Attention (sur séquence tgt)	0.92	tgt
Attention (sur séquence src)	0.72	src et PID
Attention (sur séquence src) avec cross-corrélation	0.69	src et similarité de cross-corrélation
Colonne du profil	0.64	$profile$
Attention semi-axiale (sur séquences src et tgt)	0.69	src, tgt et PID
Attention semi-axiale (sur séquences src, tgt et $profile$)	0.57	src, tgt, $profile$ et PID

Table 2: Score de Brier des modèles testés

Les scores d'évaluation des modèles sont calculés sur 900 000 exemples de notre jeu de données de Test. Ces scores sont présents dans la table 2. Ces modèles sont brièvement décrits ci-dessous par ordre d'apparition dans la table 2.

La matrice LG a été calculée sur 3,412 alignements de Pfam (version de mai 2007) par maximum de vraisemblance, elle contient 210 paramètres. Ces alignements ont été prétraités pour ne contenir qu'un nombre modéré de taxons. Par conséquent, les séquences sur lesquelles ce modèle a été entraîné sont beaucoup plus proches que celles que nous utilisons. C'est

le seul modèle que nous n'avons pas développé. Les matrices à taux de transition font une forte hypothèse, que nous pouvons modéliser ces substitutions avec des chaînes des Markov à temps continu.

Le modèle de la matrice de probabilités conditionnelles et PID (voir section 3.4.1) a été appris en comptant chaque substitution de chaque position de chaque paire de séquences de chaque alignement de l'ensemble d'entraînement.

La matrice de taux de transition couplée à l'estimation de distance évolutive par réseau de neurones (voir section 3.4.2) consiste en une matrice de taux de transition apprise (400 paramètres), et d'un réseau de neurones servant uniquement à estimer la distance évolutive locale entre les séquences *src* et *tgt* pour la position considérée. Cette distance est donc locale et non globale comme la distance estimée par FastME. Ce réseau de neurones (4 millions de paramètres) suit l'architecture des modèles d'attention semi-axiales avec 128 dimensions d'embedding, 6 couches d'attention semi-axiale (les MLP utilisent un widening factor de 2), 4 têtes d'attention chacune de 64 dimensions. Ce modèle suppose que les sites d'une séquence évoluent indépendamment et que le processus des substitutions est continu au cours du temps, mais n'est pas homogène le long de la séquence.

Le modèle des probabilités conditionnelles avec le PID estimé localement par réseaux de neurones (voir section 3.4.1), consiste en une matrice (400 paramètres) et d'un réseau de neurones servant uniquement à estimer un PID local entre les 2 séquences *src* et *tgt* pour la position considérée. De même, cette PID estimée est locale contrairement à la PID observée sur l'entièreté de *src* et *tgt*. Le réseau (4 millions de paramètres) suit l'architecture des modèles d'attention semi-axiales avec 128 dimensions d'embedding, 6 couches d'attention semi-axiale (les MLP utilisent un widening factor de 2), 4 têtes d'attention chacune de 32 dimensions.

Le perceptron multicouche (voir section 3.1) prend en entrée l'aplatissement des 2 séquences (30 acides aminés à gauche et à droite de l'acide aminé considéré), il fait l'hypothèse que les interactions entre sites d'une séquence dépendent de leurs positions relatives.

Les modèles basés sur l'attention sur une seule séquence *src* ou *tgt* utilisent une architecture similaire à BERT (3 millions de paramètres) avec 128 dimensions d'embedding, 6 couches d'attention semi-axiale (les MLP utilisent un widening factor de 4), 4 têtes d'attention chacune de 64 dimensions. Ces modèles font l'hypothèse que les interactions entre acides aminés d'une même protéine sont importantes pour estimer les substitutions possibles. Le modèle sur la séquence *tgt* n'utilise en entrée que les 30 acides aminés à gauche et à droite de l'acide aminé considéré et les fréquences d'apparition des acides aminés dans la séquence *tgt* et dans l'alignement. Les 2 modèles sur la séquence *src* utilisent en entrée que les 30 acides aminés à gauche et à droite de l'acide aminé considéré et les fréquences d'apparition des acides aminés dans la séquence *tgt* et dans l'alignement. Un des deux modèles prend en entrée la PID et l'autre prend en entrée les 128 similarités basées sur la cross-corrélation.

Le modèle utilisant la colonne du profil (voir section 1.1.2) retourne simplement les fréquences

d'apparition des acides aminés de la colonne dans l'alignement de l'acide aminé considéré.

Les modèles basés sur l'attention semi-axiale (voir section 3.3) (4 millions de paramètres) utilisent 128 dimensions d'embedding, 6 couches d'attention semi-axiale (les MLP utilisent un widening factor de 2), 4 têtes d'attention chacune de 64 dimensions. L'un utilise en entrée les 2 séquences *tgt* et *src* (30 acides aminés à gauche et à droite de l'acide aminé considéré) ainsi que les fréquences d'apparition des acides aminés dans les séquences *tgt* et *src* et dans l'alignement. L'autre utilise ces mêmes entrées auxquelles on ajoute les fréquences d'apparition des acides aminés dans les 61 colonnes considérés de l'alignement. Ces modèles, en plus de faire l'hypothèse que les interactions entre acides aminés d'une même protéine sont primordiales, supposent que l'information d'alignement est aussi importante pour estimer les substitutions possibles.

5.2 Alignement de séquences

Pour évaluer la qualité des alignements, nous utilisons le jeu de données PREFAB[16] qui contient des paires de séquences à aligner et un alignement de référence pour chacune de ces paires.

La mesure usuelle de la qualité d'un alignement par rapport à un alignement de référence est le score du "développeur" noté f_D [18] qui est défini comme le nombre de paires d'acides aminés correctement alignées par rapport à la référence divisé par le nombre total de paires alignées dans l'alignement de référence :

$$f_D = \frac{|P_{ref} \cap P_{test}|}{|P_{ref}|}$$

Où P_{ref} désigne l'ensemble des paires d'acides aminées alignées selon l'alignement de référence et P_{test} désigne l'ensemble des paires d'acides aminées alignées selon l'alignement à tester. On a donc $f_D \in [0, 1]$, 0 si l'alignement test est de mauvaise qualité par rapport à l'alignement de référence, 1 si l'alignement test est identique à l'alignement de référence.

Avec l'algorithme Needleman-Wunsch et en utilisant une pénalité de 10 pour la création d'un gap et 1 pour l'extension d'un gap, nous les résultats de la table 3.

Système de score utilisé	Score f_D
Matrice de BLOSUM62	0.503
Attention (sur *src*) avec cross-corrélation, $\lambda = 2$	0.523
Attention (sur *src*) avec cross-corrélation, $\lambda = 2.2$	0.535
Attention (sur *src*) avec cross-corrélation, $\lambda = 2.5$	0.541
Attention (sur *src*) avec cross-corrélation, $\lambda = 2.8$	0.547
Attention (sur *src*) avec cross-corrélation, $\lambda = 3$	0.551
Attention (sur *src*) avec cross-corrélation, $\lambda = 3.3$	0.552

Table 3: Score f_D des modèles testés pour l'alignement

L'utilisation de la matrice BLOSUM62 comme système de score pour l'algorithme de Needleman-Wunsch est une des méthodes les plus utilisées pour aligner des paires de séquences. Nous pouvons voir qu'en utilisant un système de score basé sur un réseau de neurones nous permet d'améliorer significativement le score f_D par rapport à la matrice BLOSUM62. Ces alignements de meilleure qualité ont un coût, sur ce jeu de test, notre méthode est en moyenne 18 fois plus lente, pour le même code de l'algorithme de Needleman-Wunsch.

6 Discussion

Nous pouvons voir sur la table 2 que le profil de l'alignement est l'information la plus importante pour estimer les probabilités de substitution. L'estimateur qui retourne simplement les fréquences d'apparition des acides aminés dans la colonne considérée de l'alignement obtient un score de Brier de 0.64.

Notre meilleur estimateur, utilisant l'attention semi-axiale sur les 3 séquences (src, tgt et $profile$), arrive à un score de 0.57. Mais si l'on enlève l'information du profil, les performances tombent à 0.69, ce qui est moins bien que le score obtenu en ayant seulement l'information du profil (0.64). Cette information semble cruciale, elle permet de savoir quels acides aminés sont probables d'exister à chaque position de la séquence.

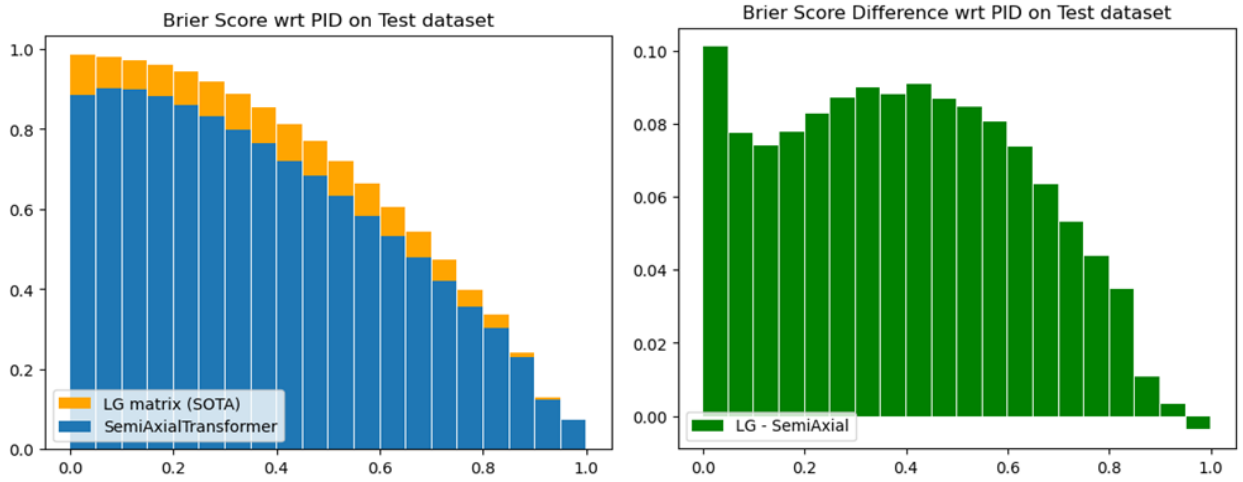
Étant donné une information en entrée similaire : séquences src et tgt. Nous remarquons que notre modèle utilisant l'attention semi-axiale est bien plus performant que l'état de l'art, à savoir la matrice LG muni de distance estimés par FastME. Nous pouvons naturellement nous poser la question suivante : Pour quelles séquences, l'attention semi-axiale est-elle plus performante que la matrice LG ? Nous observons (voir figure 10) que le gain de performance se trouve sur les séquences relativement éloignées, c'est-à-dire avec un PID < 0.8. De plus, nous arrivons à d'assez bons résultats avec notre matrice à taux de transition couplée avec un réseau de neurones estimant la distance évolutive entre les 2 protéines. Cela suggère que le gain de performances s'explique principalement par une meilleure estimation de la similarité entre les protéines.

Le gain d'information obtenu en donnant l'information du profil suggère 2 choses :

1. Les hypothèses des matrices de taux de transitions (les sites d'une séquence évoluent indépendamment et le processus des substitutions est continu et homogène au cours du temps) sont fausses.
2. Nos modèles n'arrivent pas très bien à modéliser les dépendances entre l'évolution des sites d'une séquence sans l'information du profil.

C'est deuxième point suggère que pour mieux estimer les probabilités de substitutions, il faudrait avoir plus d'information sur les interactions entre acides aminés d'une même séquence, cette information pourrait par exemple être la structure de la protéine.

Un autre point intéressant à relever est que l'utilisation de cross-corrélation sur les embed-



(a) Scores de Brier de la matrice LG et du modèle d'attention semi-axiale par tranche de PID

(b) Différence entre les scores de Brier de la matrice LG et du modèle d'attention semi-axiale par tranche de PID

Figure 10: Comparaison des performances entre le modèle d'attention semi-axiale et la matrice LG par tranche de PID

dings permet un gain significatif par rapport à l'utilisation du PID, nous passons en effet d'un score de Brier de 0.72 à 0.69, cela souligne encore une fois l'importance de la qualité de la mesure de similarité utilisée.

Nous pensons qu'une façon d'améliorer encore les résultats serait d'entraîner des modèles sur tous les alignements présents dans Pfam, pas seulement les alignements "seed". Ce gain de volume d'information se ferait au détriment de la qualité des données, mais nous permettrait d'implémenter des modèles encore plus gros en limitant les risques de sur-apprentissage. La manière d'entraîner de nos modèles avait ses avantages, mais aussi ses défauts. D'une part, se concentrer sur une fenêtre autour d'un acide aminé auquel nous voulons prédire la substitution, nous permet de simplifier le problème et de nous comparer facilement à l'état de l'art. Mais d'autre part, cette approche ralentissait l'apprentissage et ne nous permettait pas de prendre en compte l'entièreté des séquences que nous voulons traiter. Une autre approche possible aurait été d'entraîner nos modèles avec du "Masked Language Modeling", où étant donné la séquence *src* entière et la séquence *tgt* masquée partiellement, nous voulons prédire les acides aminés masqués de la séquence *tgt*.

Pour l'alignement, nous pouvons voir sur la table 3 que notre modèle est meilleur que la matrice BLOSUM62 pour l'algorithme Needleman-Wunsch. Malheureusement, comme le calcul de score est fait position par position et non pour toute une séquence d'un coup comme cela aurait pu être fait avec un apprentissage de type "Masked Language Modeling", l'alignement de deux protéines homologues est en moyenne 18 fois plus lent avec notre modèle qu'avec la matrice BLOSUM62. Nous notons quand même que l'utilisation d'un réseau de neurones pour de l'alignement est assez prometteuse.

7 Conclusion

Dans un premier temps, nous avons développé une nouvelle architecture de réseau de neurones spécialisée pour traiter des séquences alignées. Nous avons utilisé cette architecture pour modéliser les substitutions d'acides aminés dans des protéines. Les modèles que nous avons développés obtiennent de meilleurs scores que l'état de l'art. Ces gains de performances semblent surtout venir d'une meilleure estimation de la similarité/distance évolutive entre les protéines. Néanmoins, étant donné les résultats obtenus avec le profil des alignements, il semble encore très réaliste d'améliorer les performances de nos modèles. Par la suite, il serait intéressant d'étudier les gains de performances possibles en ajoutant l'information des positions dans l'espace des acides aminés ou en améliorant la mesure de similarité utilisée.

Dans un second temps, nous avons utilisé nos modèles pour une application bio-informatique concrète : l'alignement de paires de protéines. Les alignements retournés par notre méthode sont meilleurs que les alignements que l'on peut obtenir en utilisant des méthodes très répandues telles que l'algorithme de Needleman-Wunsch couplé à la matrice de BLOSUM62. L'utilisation de Deep Learning semble très prometteuse pour réaliser de l'alignement de séquence. Il serait intéressant de continuer ce travail en adaptant l'algorithme de Needleman-Wunsch pour des réseaux de neurones ou en généralisant notre méthode d'alignement à de l'alignement multiple.

A Matériel Informatique

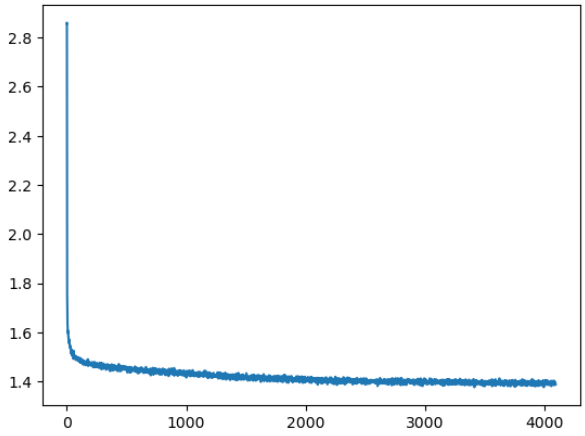
Nous avons utilisé un ordinateur sous Linux Ubuntu 20.04 avec les composantes suivantes :
GPU : NVIDIA RTX A5000, 24GB VRAM
CPU : Intel Xeon W-2265 Processor, 3.50 GHz
RAM : 128GB

Nous avons implémenté les modèles ainsi que les différentes fonctions pour le chargement des données et l'alignement en Python 3.8.16 avec PyTorch 1.11 et CUDA 11.1.

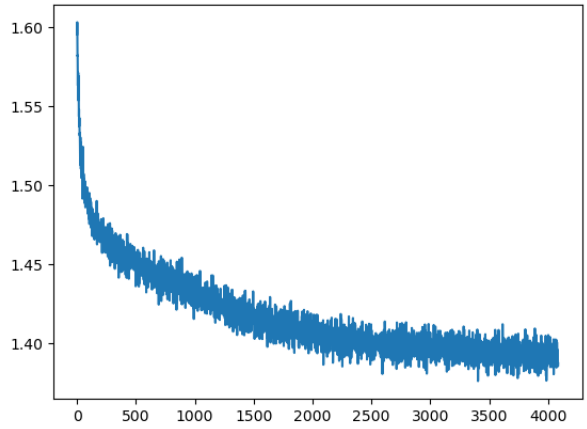
B Entraînement des Modèles

Nous avons entraîné tous les modèles basés sur l'attention sur 4096 "époques", pour chaque époque, le modèle voit 90000 exemples (paires d'acides aminés X,Y) à traiter. Nous utilisons l'optimiseur AdamW, avec un weigth decay de 0.01. Nous utilisons aussi un scheduler de learning rate "CosineAnnealing" avec une learning rate allant de $1e^{-4}$ à l'époque 0 à $1e^{-5}$ à l'époque 2047. De l'époque 2048 à 4095, la learning rate est constante à $1e^{-5}$. Nous utilisons comme régularisation supplémentaire du DropPath, c'est-à-dire ignorer les sorties de certains modules tirés aléatoirement à chaque batch pendant l'apprentissage.

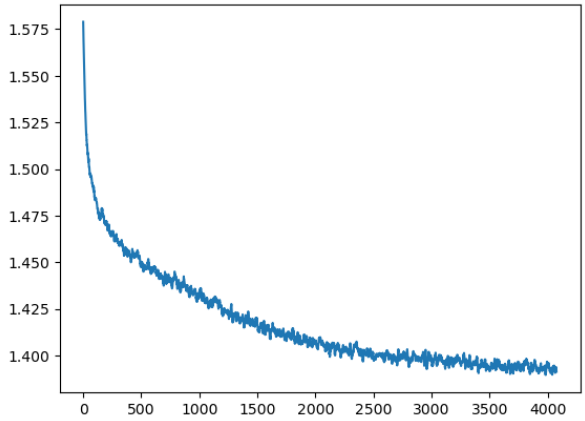
L'entraînement des modèles les plus complexes nous a pris entre 3 pour les modèles utilisant la PID et 6 jours pour les modèles utilisant la cross-corrélation qui est relativement longue à calculer. Par exemple, voici sur la figure 11 les courbes d'apprentissage du modèle basé sur l'attention semi-axiale sur les 3 séquences (src, tgt et $profile$). Nous pouvons voir que l'apprentissage est très rapide sur les 100 premières époques. Nous pouvons voir sur la figure 11d que le modèle arrive aux meilleures performances aux alentours de l'époque 3000, ensuite sur les 1000 dernières époques, les performances ni s'améliorer ni se dégrader.



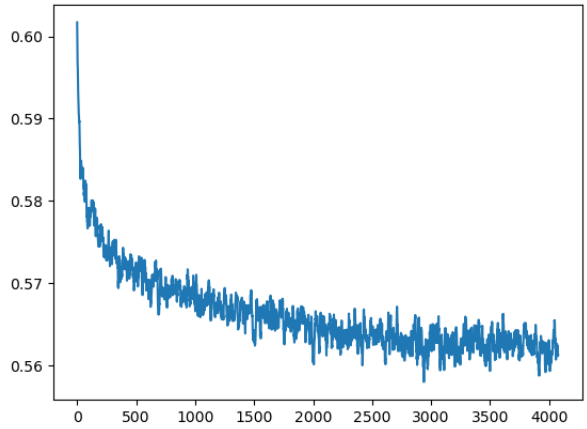
(a) Erreur cross-entropique par rapport au nombre d'époques



(b) Erreur cross-entropique par rapport au nombre d'époques, à partir de l'époque 10



(c) Erreur cross-entropique par rapport au nombre d'époques, à partir de l'époque 10, lissée avec une moyenne glissante de taille 10



(d) Score de Brier sur l'ensemble de validation par rapport au nombre d'époques, à partir de l'époque 10, lissée avec une moyenne glissante de taille 10

Figure 11: Courbes d'apprentissage du modèle utilisant l'attention semi-axiale sur les 3 séquences (src, tgt et $profile$)

References

[1] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://doi.org/10.1038/s41586-021-03819-2.

[3] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (2021), e2016239118. DOI: 10.1073/pnas.2016239118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2016239118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2016239118.

[4] Nadav Brandes et al. "ProteinBERT: a universal deep-learning model of protein sequence and function". In: *Bioinformatics* 38.8 (Feb. 2022), pp. 2102–2110. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btac020. eprint: https://academic.oup.com/bioinformatics/article-pdf/38/8/2102/49009610/btac020.pdf. URL: https://doi.org/10.1093/bioinformatics/btac020.

[5] Ahmed Elnaggar et al. "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning". en. In: *IEEE Trans Pattern Anal Mach Intell* 44.10 (Sept. 2022), pp. 7112–7127.

[6] S Henikoff and J G Henikoff. "Amino acid substitution matrices from protein blocks". en. In: *Proc Natl Acad Sci U S A* 89.22 (Nov. 1992), pp. 10915–10919.

[7] Vincent Lefort, Richard Desper, and Olivier Gascuel. "FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program". In: *Molecular Biology and Evolution* 32.10 (June 2015), pp. 2798–2800. ISSN: 0737-4038. DOI: 10.1093/molbev/msv150. eprint: https://academic.oup.com/mbe/article-pdf/32/10/2798/13166892/msv150.pdf. URL: https://doi.org/10.1093/molbev/msv150.

[8] S Whelan and N Goldman. "A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach". en. In: *Mol Biol Evol* 18.5 (May 2001), pp. 691–699.

[9] D T Jones, W R Taylor, and J M Thornton. "The rapid generation of mutation data matrices from protein sequences". en. In: *Comput Appl Biosci* 8.3 (June 1992), pp. 275–282.

[10] Si Quang Le and Olivier Gascuel. "An Improved General Amino Acid Replacement Matrix". In: *Molecular Biology and Evolution* 25.7 (Mar. 2008), pp. 1307–1320. ISSN: 0737-4038. DOI: 10.1093/molbev/msn067. eprint: https://academic.oup.com/mbe/article-pdf/25/7/1307/3520981/msn067.pdf. URL: https://doi.org/10.1093/molbev/msn067.

[11] Jaina Mistry et al. "Pfam: The protein families database in 2021". In: *Nucleic Acids Research* 49.D1 (Oct. 2020), pp. D412–D419. ISSN: 0305-1048. DOI: 10.1093/nar/

gkaa913. eprint: https://academic.oup.com/nar/article-pdf/49/D1/D412/35363969/gkaa913.pdf. URL: https://doi.org/10.1093/nar/gkaa913.

[12] Michael Ashburner et al. "Gene Ontology: tool for the unification of biology". In: *Nature Genetics* 25.1 (May 2000), pp. 25–29. ISSN: 1546-1718. DOI: 10.1038/75556. URL: https://doi.org/10.1038/75556.

[13] Dan Hendrycks and Kevin Gimpel. "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units". In: *CoRR* abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415.

[14] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[15] Saul B. Needleman and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. ISSN: 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL: https://www.sciencedirect.com/science/article/pii/0022283670900574.

[16] Robert C. Edgar. "MUSCLE: multiple sequence alignment with high accuracy and high throughput". In: *Nucleic Acids Research* 32.5 (Mar. 2004), pp. 1792–1797. ISSN: 0305-1048. DOI: 10.1093/nar/gkh340. eprint: https://academic.oup.com/nar/article-pdf/32/5/1792/7055030/gkh340.pdf. URL: https://doi.org/10.1093/nar/gkh340.

[17] Kazutaka Katoh et al. "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform". In: *Nucleic Acids Research* 30.14 (July 2002), pp. 3059–3066. ISSN: 0305-1048. DOI: 10.1093/nar/gkf436. eprint: https://academic.oup.com/nar/article-pdf/30/14/3059/9488148/gkf436.pdf. URL: https://doi.org/10.1093/nar/gkf436.

[18] J. Sauder, Jonathan Arthur, and Roland Dunbrack. "Large-scale comparison of protein sequence alignment algorithms with structure alignments†". In: *Proteins: Structure, Function, and Bioinformatics* 40 (July 2000), pp. 6–22. DOI: 10.1002/(SICI)1097-0134(20000701)40:1%3C6::AID-PROT30%3E3.0.CO;2-7.

[19] Alex Mitchell et al. "The InterPro protein families database: the classification resource after 15 years". en. In: *Nucleic Acids Res* 43.Database issue (Nov. 2014), pp. D213–21.

[20] Luca Nesterenko, Bastien Boussau, and Laurent Jacob. "Phyloformer: towards fast and accurate phylogeny estimation with self-attention networks". In: *bioRxiv* (2022). DOI: 10.1101/2022.06.24.496975. eprint: https://www.biorxiv.org/content/early/2022/06/28/2022.06.24.496975.full.pdf. URL: https://www.biorxiv.org/content/early/2022/06/28/2022.06.24.496975.

[21] Brian Ondov et al. "Mash: Fast genome and metagenome distance estimation using MinHash". In: *Genome Biology* 17 (June 2016). DOI: 10.1186/s13059-016-0997-x.