



HAL
open science

Modèle Transformer pour l'interprétabilité et les prédictions multi-niveaux des fonctions des protéines à partir de leurs séquences

Nicolas Buton

► **To cite this version:**

Nicolas Buton. Modèle Transformer pour l'interprétabilité et les prédictions multi-niveaux des fonctions des protéines à partir de leurs séquences. Bioinformatics [q-bio.QM]. Université de Rennes, 2023. English. NNT : 2023URENS040 . tel-04347632

HAL Id: tel-04347632

<https://theses.hal.science/tel-04347632>

Submitted on 15 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601

*Mathématiques, Télécommunications, Informatique, Signal, Systèmes,
Électronique*

Spécialité : *Informatique*

Par

Nicolas Buton

Transformer models for interpretable and multilevel prediction of protein functions from sequences

Thèse présentée et soutenue à Inria Rennes, le « date »

Unité de recherche : IRISA

Rapporteurs avant soutenance :

Nataliya SOKOLOVSKA Professeure des universités, laboratoire LCQB, Paris, France
Blaise HANCZAR Professeur des universités, Université Paris-Saclay, Université d'Evry, laboratoire IBISC, France

Composition du Jury :

Président :	Nataliya SOKOLOVSKA	Professeure des universités, laboratoire LCQB, Paris, France
Examineurs :	Tatiana GALOCHKINA	Maîtresse de conférences, Université Paris Cité, laboratoire BIGR, France
	Blaise HANCZAR	Professeur des universités, Université Paris-Saclay/Evry, laboratoire IBISC, France
	Yann Le Cunff	Maître de conférences, Université de Rennes, France
	François COSTE	Chargé de recherche Inria, Rennes, France
	Nataliya SOKOLOVSKA	Professeure des universités, laboratoire LCQB, Paris, France
Dir. de thèse :	Olivier Dameron	Professeur des universités, Université de Rennes, France
Encadr. de thèse :	Yann Le Cunff	Maître de conférences Université de Rennes, France
Encadr. de thèse :	François COSTE	Chargé de recherche Inria, Rennes, France

REMERCIEMENTS

Je tiens tout d'abord à remercier Tatiana Galochkina, Blaise Hanczar et Nataliya Sokolovska pour avoir accepté d'être rapporteur · rice de ce manuscrit. Je les remercie pour leurs remarques intéressantes et qui ont pu contribuer à l'amélioration de celui-ci. Je tiens à remercier mon directeur de thèse Olivier Dameron pour ses conseils sur la Gene Ontology et pour m'avoir encouragé à partager mes recherches. Mes remerciements vont ensuite à mes deux encadrants Yann Le Cunff et François Coste pour m'avoir soutenu, aidé dans mes réflexions et apporté leur expertise tout au long de ma thèse. Je remercie Symbiose, composé des équipes DYLISS, GENSCALE et GenOuest, pour l'ambiance de travail qui a vraiment été agréable et dynamique. Je remercie toutes les personnes bienveillantes qui travaillent au sein de ces équipes. Merci à l'assistante d'équipe Marie Le Roïc pour ses multiples aides. Merci à tous les doctorants, ingénieurs, chercheurs. Merci à Victor pour son expertise LaTeX, ses réflexions sur l'éthique de la recherche. Je remercie Garance, Lucas, Matthieu, Victor pour la super expérience qu'a été le festival de courts métrages Science en courts.

Je remercie ma famille et plus particulièrement mes parents pour m'avoir soutenu tout au long de mes études. Je tiens à exprimer ma profonde gratitude envers Ornella pour les innombrables heures que nous avons passées ensemble, pour ses conseils éclairés lors de mes entraînements oraux, son expertise précieuse en matière de rédaction, ainsi que pour son soutien et son écoute attentive. Je remercie mes ami · e · s, Zoé pour nos discussions scientifique passionantes. Merci à Marion pour son soutien, ses partages d'expériences sur nos sujets de thèse et sa présence à mes différentes présentations. Plus généralement merci à tous mes amis: Yoann, Guillaume, Simon, Tristan, Alexia, Antoine, Flavien, Marion, Charlotte, Coline, Chloé, Gwenole, Léa, Corentin, Clarisse, Maxime, Morgane pour le terreau de réflexions intéressantes et les bons moments passés ensemble. Enfin, merci à mes colocs, Léo pour m'avoir incité à perfectionner mes pratiques de code à la suite de longues heures de discussions et débats et Glen pour avoir partagé l'expérience du

doctorat.

TABLE OF CONTENTS

List of figures	vii
List of tables	xi
Résumé de la thèse – Summary in french	xiv
Introduction	1
1 Background	7
1.1 Protein: from sequence to function	8
1.1.1 Amino acids	8
1.1.2 Primary structure - a chain	12
1.1.3 Secondary structure	13
1.1.4 Tertiary structure	15
1.1.5 Quaternary structure	17
1.1.6 Functions	18
1.2 Automatic Function Prediction (AFP)	22
1.2.1 Classical bio-informatics methods	22
1.2.2 Machine learning methods	24
1.2.3 Transformer architecture and two phases training	25
2 Predicting enzymatic function of protein sequences with attention	49
2.1 Introduction	49
2.2 Methods	50
2.2.1 Enzyme class prediction task	50
2.2.2 The model	52
2.2.3 Interpretability	53

TABLE OF CONTENTS

2.3	Results	57
2.3.1	Enzyme class prediction	57
2.3.2	Interpretability	59
2.4	Discussion	62
2.5	Conclusion and perspective	63
3	Using prior information for hierarchical multi-label prediction	67
3.1	Introduction	67
3.2	Gene Ontology (GO) analysis	68
3.2.1	Gene Ontology terms	68
3.2.2	Gene Ontology annotations	72
3.3	Performance metric for evaluating Automatic Function Prediction tools	79
3.4	Integrating Gene Ontology information in the labelling	82
3.4.1	Experiment description	84
3.4.2	Common setup	85
3.4.3	Sample size estimation - Power analysis	87
3.4.4	True path rule propagation	88
3.4.5	Ontology to avoid some false negative from the CWA	92
3.5	Integrate Gene Ontology information in the structure of the space	93
3.5.1	Positive and negative relations and sampling	96
3.5.2	Common architecture	99
3.5.3	Output probability	103
3.5.4	Loss functions	106
3.5.5	Metric choice	108
3.5.6	Performance and coherence of the different models	109
3.5.7	Discussion	112
	Conclusion and perspective	115
3.6	Conclusion	115
3.7	Perspective	116
3.7.1	On integrating the Gene Ontology information	116
3.7.2	Research idea for Transformer	118

Bibliography

123

LIST OF FIGURES

1.1	From DNA sequence to protein functions	7
1.2	9
1.3	Venn diagram of the physicochemical property of the 20 common amino acids. The cysteine is separated into two states, one with a disulfide bond (S-S) and one without (S-H). From the Taylor classification [Tay86]	11
1.4	Peptide chain formations. Two separated generic amino acids on the left and the resulting chain on the right with a peptide bond. This reaction forms water as a by-product. Figure from [22a].	12
1.5	Dihedral angles and peptide plane in a generic protein. From [Gol+16]. . .	13
1.6	The Ramachandran plot, as depicted in reference [22b], illustrates the observed angles of backbones, revealing two distinct clusters of points that correspond to the most frequently occurring secondary structures, namely the alpha-helix and beta-sheet	14
1.7	α -helix and beta sheet from [23c]	15
1.8	PDB structure of Human Aldose Reductase in complex with NADP+ and the inhibitor IDD594 at 0.66 Angstrom (PDB Id: 1US0)	16
1.9	Tropocollagen molecule: three left-handed procollagens (red, green, blue) join to form a right-handed triple helical tropocollagen. Figure from [23b]	17
1.10	Ancestor chart for GO:0098726 (Symmetric division of skeletal) from QuickGO website	19
1.11	Enzyme Commission number hiearachy	21
1.12	Multiplication of a transpose embedding matrix with a one-hot encoded residue to obtain the embedding of this specific residue	32
1.13	Long range contact in proteins	34
1.14	High-level Transformer encoder architecture	35

LIST OF FIGURES

1.15	Given a sequence of four residues, the Lysine (K) residue at position 3 gives the highest attention to the Leucine (L) residue at position 4 (left). This attention row can be found on line 3 of the attention map on the right. . .	37
1.16	Detailed view of the self-attention mechanism with one head. The blue trapeze represents linear projection. Residue positions are framed.	38
1.17	The feed-forward layer present in transformer network. The blue trapeze represents linear projection. The non-linearity after FF1 is not represented here.	39
1.18	Overview of one complete Transformer encoder layer. The red dotted line represents how the information can flow for the first embedding. Everywhere in the self-attention and only between itself in the feed-forward layer.	40
1.19	Two phase training for deep neural network (Transformer in this Figure). .	42
1.20	An example of pre-training for one protein. The first step is to mask/corrupt some tokens. The second step is to run the corrupted sequence through the network. And lastly, use the embedding to predict the original token. . . .	46
1.21	An example of fine-tuning on a classification task at the protein level. First, a special token named [CLS] is prepended to the sequence. Then the sequence is run through the model. And finally, the embedding of the [CLS] is used to predict the class of the protein.	47
2.1	Summary of the main types of interpretability methods	54
2.2	Aggregation methods AttnAgg1A1A and AttnAgg2A1A. AttnAgg2A1A averages all attention maps and then average over the columns to obtain a row vector.	56
2.3	Top: Precision / Recall curve for different interpretability methods. The attention aggregation group is represented by the AttnAgg1A1A method. Bottom: f1 score for top- k residue with the highest residue importance scores.	64

-
- 2.4 3D and 1D positions of most important residues highlighted by our interpretability method AttnAgg2A1A on N_h(3)-dependent nad(+) synthetase, one of the best example of catalytic site retrieval, and Aldehyde dehydrogenase (FAD-independent), the worst example of catalytic site retrieval. The 5% most important residues for our interpretability method are highlighted in red and catalytic sites identified in M-CSA database are represented by blue spheres. 65
- 3.1 Global pipeline step showcasing re-implementation and method unification, encompassing diverse components. Starting with data preprocessing, which leverages Gene Ontology information and annotations, followed by space and model selection. The training process entails the careful selection of loss functions and optimizers. Finally, performance evaluation and model coherence assessment are conducted using metrics, ensuring a comprehensive evaluation of the resulting model. 69
- 3.2 Graph distance of all GO terms to the root term of their respective namespace 73
- 3.3 An example of the "biosynthetic process" annotation for a protein and a potential propagation with the True Path Rule in the labeling process, which also annotated "organic substance metabolic process", "metabolic process" and "biological process". 83

LIST OF FIGURES

- 3.4 Illustration of the three experiments conducted to incorporate Gene Ontology (GO) information into the labeling process: (a) Comparison of two models: the first one can predict annotations at all levels and the second can only predict leaf-level annotations. This experiment employs a dataset consistently containing the most precise GO annotation for each protein. (b) contrast between the two models: the first model predicts all GO terms, while the second model predicts only the GO terms associated with a protein's most precise annotation. The first model considers the most precise annotation and all its ancestors as True, with the remaining annotations marked as False. The second model is trained using the most precise annotations as True and others as False. (c) Comparison between two models capable of predicting all GO terms. In the first model, there is no signal for the most precise annotations when they are unavailable. Conversely, the second model assigns all negative annotations if a protein is not annotated at the most precise level. 85
- 3.5 The t-values, plotted against the number of samples, were computed for the central distribution under the null hypothesis (H_0) to achieve an alpha value of 0.05. Additionally, the t-value for the offset t-distribution was determined, considering a difference of 0.01 with a desired power of 0.8. . . 89
- 3.6 The two t-distribution for the optimal value of n_1 equal 8.835. The blue area corresponds to 2.5% of the total area because it is a two-tailed test and the orange area that represent 80% of the total area. 90
- 3.7 Comparison of the implications of utilizing the True Path Rule for annotation propagation versus training solely on leaf GO terms without considering intermediate GO terms. The figure illustrates the difference in F_{\max} measure between the two variants. 92
- 3.8 Comparison of the difference in F_{\max} between the Open World Assumption (OWA) approach and the Closed World Assumption (CWA) approach during training. 94

3.9	Illustration of the different embedding types in the Poincaré Ball and Euclidean space with toy classes. (a) Hyperbolic Model - Distance-Based Embedding: classes (can be GO terms) are represented by points within the space. (b) Hyperbolic Model - Cone-Based Embedding: classes are represented by cones in the space. (c) Hyperbolic Model - Hyperplane-Based Embedding: classes are represented by hyperplanes in the space. (d) Euclidean Model - Point-Based Embedding: classes are represented by points within a euclidean space.	97
3.10	Comprehensive depiction of the universal architecture shared among diverse models, highlighting essential specificities when required. The classification layer comprises the output computation and the logits function. It can be noted that to compute the output and the logits both information from protein embedding and the GO embedding is needed	100
3.11	Example of a convex cone in the Poincare ball representing a GO term, and a projection of a protein. Protein has a high probability to be of the class of the drew GO cone because the protein angle is less than the opening angle of the GO cone.	104
3.12	Mean WF_{\max} metric across different embedding dimensions	110
3.13	<i>PosEdgeRecover</i> with respect to the dimension of embedding for each namespace	111
3.14	<i>IPMneg</i> with respect to the dimension of embedding for each namespace .	112
3.15	<i>IPMpos</i> with respect to the dimension of embedding for each namespace .	113
3.16	PosEdgeRecover Metric Evolution during Training for Euclidean_LR, HMI, and HMIHLR Models	114
3.17	WF_{\max} metric across different embedding dimensions for each namespace .	114
3.18	S_{\min} metric across different embedding dimensions for each namespace . . .	114

LIST OF TABLES

1.1	10
1.2	Summary of different order for each type of data. Comparison of the number of proteins available for each dataset	22
2.1	Number proteins sequences in each Enzyme Commission (EC) first level class in each dataset. The dataset named EC40 is from [Str+20] and the dataset named ECPred40 is derived from [Dal+18]. SwissProt_2021_04 is directly from UniprotKB/Swissprot without filtering.	50
2.2	Hyper-parameters used for the fine-tuning of the three different versions of EnzBert. When different a "-" is used.	53
2.3	Comparison with UDSMProt of the prediction quality at the two levels of EC40 test set.	58
2.4	Comparison with ECPred of the prediction quality at the five levels of ECPred40 test set.	59
2.5	Evaluation of best interpretability method of each category with respect to the M-CSA dataset. Precision-recall Gain Area Under the Curve (PRG-AUC) and the max F-Gain metrics is reported, and we also report the mean execution time for each method (for one protein, in second on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz).	61
3.1	The different relations types and between which namespace they operate with their respective number of edges from obo format go in the gene ontology website from April 2023	71
3.2	Number of GO terms, leaves, and intermediate terms for each sub-ontology: MF, BP, and CC	72

3.4	All the different relation qualifier order by number of annotation available in the EBI UniProt GAF file	75
3.3	Number of intermediate annotations, leaf annotations, and the total number of annotations for each namespace	76
3.5	Number of annotations for each evidence code in the EBI Uniprot GAF file only for protein	78
3.6	Top 20 of ECO code with their number of annotations. Some annotations can have multiple ECO codes because they are propagated to the root. . .	79
3.7	Top 10 of ECO code with their number of annotations only annotation recommended for training: ECO:0000269(EXP), ECO:0000305(IC), ECO:0006056(HTP), ECO:0000304(TAS)	80
3.8	Number of replicas necessary to find an effect as small as the one specified in the "effect of" with different estimated standard deviations.	91
3.9	Number of positive relations depending on the strategy. Only direct edges present in the GO graph, which are non-redundant or edges from the transitive closure that are redundant.	99
3.10	Number of negative relations depending on the strategy. Relations are here considered as not directed.	99

RÉSUMÉ DE LA THÈSE – SUMMARY IN FRENCH

Le nombre de séquences de protéines disponibles dans les bases de données de bio-informatique augmente rapidement. Par exemple, la base de connaissances de référence UniProtKB a connu une croissance substantielle au cours de cette thèse : entre octobre 2020 et mai 2023, le nombre de séquences disponibles dans UniProtKB est passé de 196 millions à plus de 249 millions de séquences. Cependant, les annotations fonctionnelles obtenues par des données expérimentales sont difficiles, coûteuses et chronophages à obtenir. En conséquence, seule une quantité limitée a été annotée de cette façon. Par exemple, les séquences manuellement vérifiées dans UniProtKB, réunis dans UniprotKB/Swiss-Prot, ne représentent que 0,25% de toutes les entrées. Comblent cet écart entre les séquences seules et les séquences possédant des annotations fonctionnelles est un défi majeur. Ainsi, l'annotation fonctionnelle automatique des séquences protéiques est un domaine en pleine expansion. Cela principalement grâce aux méthodes basées sur l'alignement de séquences et, plus récemment, aux approches d'apprentissage automatique et d'apprentissage profond. Ces avancées méthodologiques puisent souvent leur inspiration dans le traitement automatique du langage naturel (TAL), en effet, même si les séquences biologiques possèdent des particularités, les défis méthodologiques étaient souvent en partie communs. Récemment, des progrès importants ont été observés dans le domaine du TAL, grâce à l'émergence d'un nouveau type de modèles et de techniques d'entraînement très performantes. Deux développements notables dans ce domaine sont l'architecture d'apprentissage profond Transformer et l'approche d'entraînement en deux phases. Cette architecture Transformer permet de mieux modéliser les dépendances longue distance, un aspect crucial lorsqu'il s'agit de modéliser les protéines. De plus, la méthodologie d'entraînement en deux phases a gagné en importance. Elle comprend une première phase non supervisée, plus précisément autosupervisée, qui utilise des séquences non annotées, suivie d'une

phase d'ajustement avec des données annotées cette fois-ci, adaptée à une tâche spécifique. Ces avancées ont commencé à être utilisées dans le traitement des séquences protéiques, mais, principalement pour la prédiction de structure secondaire, de contacts, de stabilité ou pour la prédiction de fonctions spécifiques telle que la fluorescence. Cependant, au début de l'adoption de l'architecture Transformer, il n'y avait pas de publications de recherche se penchant spécifiquement sur la prédiction de fonctions génériques. Le premier objectif de cette thèse a été d'étudier l'impact de l'architecture Transformer, ses mécanismes d'attention et sa capacité à modéliser les dépendances longue distance, ainsi que l'entraînement en deux phases, dans le domaine de la prédiction automatique de fonction (AFP). Afin d'accomplir cela, il a été nécessaire d'établir un environnement contrôlé en utilisant des enzymes. Les enzymes ont été choisies comme cible, car elles représentent une part importante de toutes les protéines et sont bien caractérisées grâce à la hiérarchie définie par la nomenclature Enzyme Commission (EC). Elles jouent un rôle crucial dans l'accélération des réactions chimiques des êtres vivants. L'environnement contrôlé consistait donc dans la prédiction des classes d'enzymes monofonctionnelles. En complément, une nouvelle méthode d'interprétabilité dérivée de cette architecture a été développée et comparée à des méthodes classiques. L'objectif principal de la deuxième partie de la thèse était d'étudier trois approches pour intégrer l'information de la Gene Ontology dans les modèles de prédiction. De plus, la deuxième partie de la thèse s'est étendue à la prédiction multilabels. La nature multilabels peut être divisée en deux aspects. Le premier aspect concerne la multifonctionnalité, car les protéines peuvent avoir plusieurs fonctions distinctes. Le deuxième aspect concerne la prédiction de fonctions à différents niveaux de spécificité, que l'on peut nommer multi-niveaux. Par la suite, une description détaillée des deux principales contributions réalisées au cours de la thèse sera faite.

Le premier objectif de la thèse a été la prédiction de la fonction des enzymes avec une architecture Transformer. Les enzymes constituent une proportion importante de l'ensemble des protéines identifiées. Par exemple, en juin 2023, elles représentaient 48% des protéines dans la base de données UniProtKB/Swiss-Prot. De plus, les enzymes jouent un rôle crucial dans l'évaluation de l'efficacité des modèles de prédiction automatique de fonction. Cette tâche est un terrain d'essai pour le développement de méthodes prédictives pour deux raisons. Premièrement, les fonctions des enzymes sont bien définies puisqu'elles

sont décrites et organisées par l'Enzyme Commission (EC), qui fournit des labels précis pour entraîner les modèles d'apprentissage automatique. Deuxièmement, la quantité de séquences enzymatiques annotées disponibles est suffisante pour permettre l'entraînement et l'évaluation. Dans cette tâche, différents modèles ont été développés pour prédire les fonctions enzymatiques. Certains modèles reposent sur l'alignement de séquences, où les séquences protéiques sont alignées pour identifier les similarités et les variations. En trouvant la séquence la plus proche avec une annotation fonctionnelle connue, l'annotation peut être transférée à la nouvelle séquence. Cependant, ces méthodes reposent uniquement sur la similarité des séquences, attribuant un poids égal aux différents résidus mutés à travers diverses familles de protéines. En conséquence, elles échouent à identifier les résidus cruciaux qui jouent des rôles spécifiques dans les fonctions protéiques. Ces méthodes rencontrent donc des difficultés pour identifier de manière précise les protéines qui ont des fonctions conservées depuis leur dernier ancêtre commun, mais présentent des variations significatives de leurs séquences. D'autres approches utilisent des techniques d'apprentissage automatique classiques avec des caractéristiques pré-extraites telles que les proportions de chaque acide aminé, la masse moléculaire, le point isoélectrique et les propriétés physico-chimiques. Cependant, en raison de cette extraction préalable des caractéristiques et le modèle n'ayant jamais accès aux séquences brutes, ces modèles peuvent manquer des signaux importants qui pourraient contribuer à des prédictions plus précises. Les avancées récentes en apprentissage profond ont été appliquées à la prédiction des enzymes, avec des modèles utilisant directement les séquences brutes. Cependant, les architectures d'apprentissage profond utilisées telles que les réseaux de neurones convolutifs (CNN) et les réseaux Long Short-Term Memory (LSTM) présentent des limites et ont été dépassées par les modèles Transformer dans les tâches de traitement automatique du langage naturel (TAL). Les Transformers ont donné des résultats encourageants dans diverses tâches biologiques, notamment la prédiction de contacts et la prédiction de la structure secondaire. La thèse contribue au domaine en appliquant le modèle Transformer à la prédiction de classes enzymatiques, ce qui a donné d'excellents résultats. En plus de l'annotation fonctionnelle, nous avons également exploré l'application du mécanisme d'attention du Transformer en tant que méthode d'interprétabilité. Cela permet une compréhension plus complète de la connexion entre l'entrée, qui est la séquence protéique et la sortie, qui est

la prédiction fonctionnelle produite par le modèle. À partir du mécanisme d'attention, des scores d'importances pour les résidus peuvent être obtenus. Une comparaison approfondie entre les méthodes d'interprétabilité classiques et l'approche d'interprétabilité basée sur l'attention du Transformer a été réalisée, démontrant de meilleurs résultats pour cette dernière.

La deuxième partie de cette thèse a été consacrée à la prédiction de toutes les fonctions de toutes les protéines en utilisant le vocabulaire de la Gene Ontology (GO), tout en essayant d'exploiter sa structure. Contrairement à la hiérarchie EC, qui se conforme à une relation strictement hiérarchique, le GO permet des relations plus complexes, car certains termes peuvent avoir plusieurs parents dans le graphe. De plus, notre recherche visait à relever les défis posés par les scénarios multilabels, qui englobent (1) des prédictions à plusieurs niveaux, comprenant à la fois des termes GO génériques et spécifiques, ainsi que (2) des cas de multifonctionnalité, reconnaissant que certaines protéines peuvent posséder plusieurs fonctions distinctes. Plusieurs approches existent dans la littérature pour intégrer les ontologies. Dans ce cadre, trois façons d'utiliser les informations de la Gene Ontology ont été testées. La première méthode, largement utilisée, consiste à propager les labels d'une fonction spécifique pour annoter toutes ses fonctions parentes, qui sont intrinsèquement plus génériques. Cependant, à notre connaissance, les avantages de cette technique de propagation n'ont pas été examinés en détail et quantifiés dans les études précédentes. Les effets d'une telle propagation ont ici été étudiés. La deuxième méthode prend en compte le fait que nous sommes dans l'hypothèse du monde ouvert pour intégrer l'information de la Gene Ontology dans certaines annotations. Cette hypothèse suppose que nous ne possédons pas toutes les annotations pour chaque protéine et l'absence d'annotations n'est pas forcément une annotation négative, contrairement à l'hypothèse du monde clos. L'approche dominante dans la plupart des études utilisent l'hypothèse du monde clos, qui conduit souvent à des annotations faussement négatives lorsque les protéines ne possèdent pas d'annotations spécifiques. Sous cette hypothèse, lorsque les protéines sont partiellement annotées, toutes les fonctions précises sont étiquetées comme fausses, malgré une forte probabilité qu'au moins l'une d'entre elles soit correcte. Pour remédier à ce problème, nous avons exploré l'utilisation de la Gene Ontology pour marquer tous les descendants des annotations les plus précises avec un nouveau label "incertain". La

troisième approche d'intégration des connaissances ontologiques utilise des plongements associés à chaque terme de la GO. À ce sujet, nous avons comparé diverses techniques de plongement dans un cadre unifié de prédiction des fonctions des protéines en utilisant des métriques classiques du domaine. Ces techniques utilisent des plongements hyperboliques, car incorporer un graphe orienté acyclique, telle que la Gene Ontology, dans un espace euclidien tout en préservant la métrique des graphes soulève certains défis. Cependant, l'espace hyperbolique offre une solution plus adaptée en raison de la croissance exponentielle des distances correspondant à la croissance exponentielle des nœuds à différentes profondeurs dans le DAG. Plus précisément, trois modèles hyperboliques ont été testés. Le premier modèle testé utilise la distance hyperbolique, faisant apparaître un conflit entre la nature symétrique de la distance et la relation dirigée entre les termes GO. Pour résoudre ce problème, des cônes hyperboliques ont été proposés dans un deuxième modèle, où les termes GO sont représentés par un sommet et un angle d'ouverture dépendant de la distance par rapport au centre. Mais, la conversion des angles en probabilités présente des défis, qui sont évités grâce à une autre méthode, consistant à utiliser des hyperplans hyperboliques, qui peuvent facilement générer des probabilités. Une configuration standardisée a été utilisée pour comparer ces modèles, en adoptant des métriques classiques pour la prédiction des fonctions des protéines, telles que le maximum F1-score (Fmax) et la distance minimum sémantique (Smin), ainsi que leurs variantes pondérées. Alors que certains articles évaluent des modèles pour les prédictions de liens, notre travail s'est concentré sur la prédiction des fonctions des protéines.

Les résultats de la thèse sont les suivants. Dans la première partie, un modèle état de l'art appelé EnzBert a été présenté, il utilise les séquences pour prédire l'annotation fonctionnelle des enzymes à partir de l'architecture du Transformer et bénéficie du mécanisme d'attention pour capturer les caractéristiques importantes des séquences enzymatiques. Notre travail présente également une méthode d'interprétabilité simple, mais efficace basée sur des cartes d'attention, qui offre des perspectives sur la relation entre les séquences enzymatiques et l'annotation fonctionnelle. Ces résultats permettent une meilleure compréhension de la manière dont les classes enzymatiques sont déterminées à partir de leurs séquences. La méthode d'interprétabilité peut aider à découvrir les caractéristiques et les motifs importants dans les séquences d'enzymes, ce qui pourrait éventuellement contribuer

à de futures recherches sur l'optimisation des enzymes.

Dans la deuxième partie, le champ d'application a été élargi au-delà des enzymes pour inclure toutes les protéines en utilisant le vocabulaire de la Gene Ontology. En ce qui concerne l'intégration de l'information des relations de la Gene Ontology (GO) dans les annotations, l'étude a révélé que l'utilisation courante de la "True Path Rule" améliorait les performances. Cependant, lors de l'utilisation de GO pour atténuer la propagation des faux négatifs dans le cadre de l'hypothèse du monde clos, aucune augmentation significative de performance n'a été observée. De plus, l'utilisation de GO pour contraindre les plongements des termes de la GO n'a pas affecté les performances, mais a conduit à une meilleure cohérence des prédictions. Enfin, l'étude a comparé les modèles hyperboliques aux modèles euclidiens et a confirmé que les modèles hyperboliques étaient plus performants dans de petites dimensions, mais pas dans de plus grandes dimensions. Il est important de noter que ces résultats sont préliminaires et que des recherches supplémentaires sont nécessaires pour mieux comprendre les forces et les faiblesses des modèles hyperboliques.

INTRODUCTION

The number of protein sequences available in bioinformatics databases is growing at a rapid pace. For instance, the reference protein knowledge base UniProtKB has experienced substantial growth during this thesis: Between October 2020 and May 2023, the number of available sequences in UniProtKB increased from 196 million to over 250 million sequences. However, functional annotation supported by experimental data is a difficult, expensive, and time-consuming task. As a result, only a limited amount has been manually annotated. For example, reviewed sequences in UniProtKB named UniprotKB/Swiss-prot represent only 0.25% of the total entries. One of the major questions of interest is to bridge the gap between these sequences and the functions of the corresponding proteins. Hence, automatic functional annotation of protein sequences has been a growing field over the past years mainly driven by sequence alignment-based methods and, more recently, machine learning and deep learning approaches. Interestingly, these methodological advancements have frequently drawn inspiration from Natural Language Processing (NLP), despite the distinct characteristics of biological sequences and natural language sequences. Nevertheless, there are also many shared aspects between the two domains. Recently, substantial progress has been witnessed in the field of NLP, thanks to the emergence of highly efficient models and training techniques. These advancements have exhibited remarkable performance on various NLP benchmarks. Two noteworthy developments in this field are the Transformer architecture and the two-stage training approach. The Transformer architecture, which incorporates attention mechanisms, has proven to be particularly influential. This architecture enables the modeling of long-range dependencies, a crucial aspect when dealing with proteins. Furthermore, the two-stage training methodology has gained prominence. It involves an initial unsupervised stage that utilizes raw text data, followed by a fine-tuning stage tailored to a specific task. These advancement has started to emerge in the treatment of protein sequences. However, they were at first mostly used for secondary structure prediction, contact prediction, stability, or

specific functions like fluorescence. The prediction of generic functions was not initially a focus. The primary objective of this thesis was to investigate the potential transformative impact of the Transformer architecture, with its attention mechanisms and ability to model long-range dependencies, on the field of Automatic Function Prediction (AFP). Additionally, a novel interpretability method derived from this architecture was developed and compared against classical methods. The initial part of the thesis is dedicated to testing the applicability of the Transformer architecture in a controlled environment. To achieve this, enzymes were chosen as the target since they represent a significant portion of all proteins and are well-characterized thanks to Enzyme Commission (EC) numbers. Enzymes play a crucial role in accelerating chemical reactions. The primary focus in the initial part of the thesis is on predicting monofunctional enzyme classes. Additionally, the thesis assesses the interpretability method derived from the Transformer architecture and compares it to other conventional interpretability methods. The primary objective of the second part of the research was to investigate three approaches for integrating Gene Ontology into the prediction models. Furthermore, the research expanded to encompass a multi-label prediction task for two specific reasons. Firstly, proteins can possess multiple functions, necessitating the consideration of their multi-functionality. Secondly, the prediction of functions across different specificity levels was pursued, emphasizing a multi-level approach. The following will provide a detailed description of the two main contributions made during the thesis.

The first objective of the thesis has been the prediction of enzyme function with a specific deep-learning architecture named the Transformer. Enzymes constitute a significant proportion of all identified proteins. For instance, as of June 2023, they accounted for 48% of all proteins in UniProtKB/SwissProt. Moreover, enzymes play a crucial role in assessing the effectiveness of automatic function prediction models. This task is a testbed for the development of predictive methods for two reasons. First, the enzyme's functions are well-defined according to experimental evidence. These functions are standardized by Enzyme Commission (EC) which provides well-defined targets to train machine learning models. Secondly, the amount of annotated enzymatic sequences available is sufficient to enable training and independent large-scale evaluations. In this task, various models have been developed for predicting enzymatic functions. Some models rely on sequence

alignment, where protein sequences are aligned to identify similarities and variations. By finding the closest sequence with a known functional annotation, the annotation can be transferred to the new sequence. However, these methods solely rely on sequence similarity, assigning equal weight to different mutated residues across diverse protein families. As a result, they fail to identify the crucial residues that play specific roles in protein functions. Consequently, these methods face challenges in accurately identifying remote equivalents, which refer to proteins that have conserved functions since their last common ancestor but exhibit important sequence variations. Other approaches employ classical machine learning techniques with pre-computed features such as amino acid proportions, molecular weight, isoelectric point, and physicochemical properties. However, due to the extraction of features from raw sequences, these models may miss important signals that could contribute to more accurate predictions. Recent advances in deep learning have been applied to enzyme prediction, with models directly using raw sequence information. However, the deep learning architectures used such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have limitations and have been surpassed by Transformer models in Natural Language Processing (NLP) tasks. Transformers have started to demonstrate encouraging results in various biological tasks including contact prediction and secondary structure prediction. One of the contributions of the thesis to the field lies in the application of the Transformer model to enzyme class prediction, which has resulted in excellent outcomes. In addition to functional annotation, we also explore the application of the Transformer's attention mechanism as a method of interpretability. This enables a more comprehensive comprehension of the connection between the input, which is the protein sequence, and the output, which is the functional prediction produced by the model. From attention, importance scores for individual residues can be derived. A comprehensive comparison between classical interpretability methods and the Transformer's attention-based interpretability approach was conducted, demonstrating better results for the latter.

The second part of this thesis was dedicated to the prediction of all functions of all proteins using the vocabulary of Gene Ontology (GO) and leveraging its inherent structure. In contrast to the EC hierarchy, which adheres to a strict hierarchical relationship, the GO permits more intricate relationships, as child terms can have multiple parents.

Furthermore, our research aimed to tackle the challenges posed by multi-label scenarios, which encompass (1) predictions across multiple levels, including both generic and specific GO terms, as well as (2) multi-function cases, acknowledging that certain proteins can possess multiple distinct functions. There are several approaches available for integrating ontology knowledge into the framework. Three ways of using the Gene Ontology information have been tested. The first and widely used method involves propagating the labels from a specific function to annotate all its parent functions, which are intrinsically more generic. However, to our knowledge, the advantages of this propagation technique have not been thoroughly examined and quantified in previous studies. Therefore, an investigation of the effects of such propagation was conducted. The second way use the Closed World Assumption (CWA) to take into account gene ontology. The prevailing approach in most of the literature is to use the CWA, which often leads to False Negative annotations when proteins lack specific annotations. Under this assumption, when proteins are incompletely annotated, all precise functions are labeled as false, even though it is very likely that at least one of them is correct. To address this issue, we explored the usage of the Gene Ontology to mark all descendants of the most precise annotations with a new label "uncertain". The third approach to integrating ontology knowledge is using embeddings. On this topic, we compared various techniques for incorporating GO embeddings within a unified framework for protein function prediction using classical domain metrics. These techniques use hyperbolic embedding because embedding tree-like structures, such as the Gene Ontology, in Euclidean space while preserving metric properties poses challenges. However, hyperbolic space offers a more manageable solution due to the exponential growth of distances aligning with the exponential growth of nodes at different depths in the tree. More precisely three hyperbolic models were tested. The first tested model uses hyperbolic distance, but it has been observed that the symmetric nature of distance conflicts with the directed relationship between GO terms. To address this, hyperbolic cones were proposed in a second model, where GO terms are represented by an apex and an aperture angle dependent on the distance to the center. Converting angles to probabilities presents challenges, and alternative methods include using hyperbolic hyperplanes, which can easily output probabilities. A standardized setup was used to compare these models, adopting classical metrics for AFP, such as F_{max} and S_{min} ,

and their weighted variants. While some papers evaluate models for link predictions, our study focused on protein function prediction.

The thesis is structured as follows. The first chapter provides background information, introducing proteins and the necessary machine learning concepts. The second chapter focuses on enzyme class prediction and explores interpretability aspects specifically related to enzymes. Finally, the last chapter delves into the examination of three distinct methods for integrating Gene Ontology into the prediction model.

BACKGROUND

As said before, the work done here is in conjunction with two themes: protein and deep learning. This first chapter will present sufficient biological and deep learning knowledge to understand the different contributions. Proteins play a critical role in maintaining the proper functioning of cells, tissues, and organs. The first section takes us from the building blocks of proteins to their functions, passing through their diverse conformations.

The second section is about protein modelization and automatic function prediction (AFP). First, the classical bio-informatic tool used to predict protein function for new sequences is presented. Next the basis of neural networks and deep learning is introduced. How the common method uses embedding to encode sequence information and which supervised and unsupervised loss are used. Finally, the Transformer neural network architecture is described in detail, which would be of great importance for this thesis.

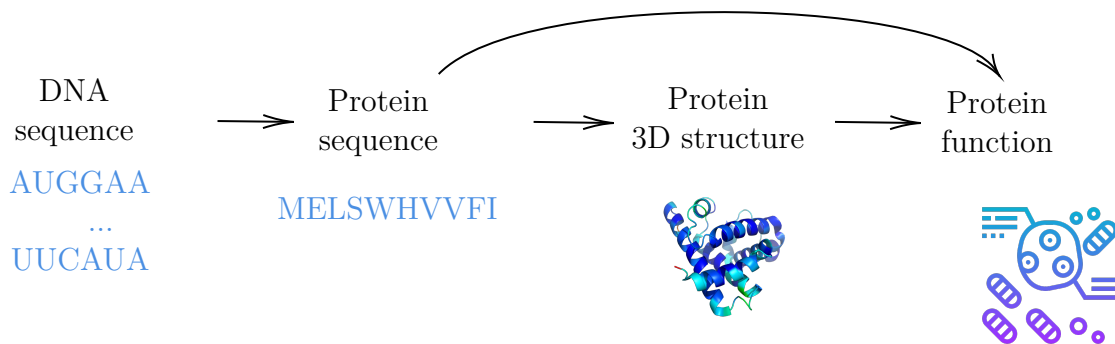


Figure 1.1 – From DNA sequence to protein functions

1.1 Protein: from sequence to function

In this section, the role of amino acids as the fundamental building blocks of proteins will be elucidated. Providing a comprehensive understanding of their crucial role as the basic components of proteins. First, we will describe amino acids and their physicochemical properties. Then their assembly into chains, two, three, and fourth-dimensional structure, to finally arrive at their functions. The process of protein assembly and folding, which involves complex three-dimensional structures, is described, shedding light on how proteins achieve their functional shapes. Furthermore, elaborates on the diverse functions that proteins express in various biological processes.

1.1.1 Amino acids

Only a subset of amino acids are the basic component of proteins. The next section will present the different types of amino acids and their different properties. Amino acids are organic compounds that have two functional groups (see fig 1.2). The first functional group is a carboxyl group ($\text{C}(=\text{O})\text{OH}$). The second is an amine functional group, it is a nitrogen atom with a pair of valence electrons that are not shared with another atom. The first carbon atom attached to the carbonyl group is named the C- α . The proteinogenic amino acids are α -amino acids because the amino group is also linked to C- α , then we can also call this "the central carbon" atom of the amino acids. Attached to this carbon there is a side chain R that differs among all amino acids. Amino acids have two optical isomers (enantiomers), two molecules that are mirror images of each other. But as observed in figure 1.2, the side chain is toward us and the hydrogen is behind. This is a lévogyre amino acid (L-Amino acids) like all Proteinogenic ones. Finally, proteinogenic amino acids are L- α -amino acids that are used as building blocks for protein. In the following, we will use the term amino acids only to refer to proteinogenic amino acids.

The International Union of Pure and Applied Chemistry (IUPAC) defined twenty different common amino acids plus two special ones present only in a few species (Pyrrolysine and Selenocysteine)[ZG07]. They are represented in Table 1.1, with their name, a one-letter code, and a three-letter code. Four other are also used to take into account uncertainty when we describe the sequence of amino acids. There are some amino acids that

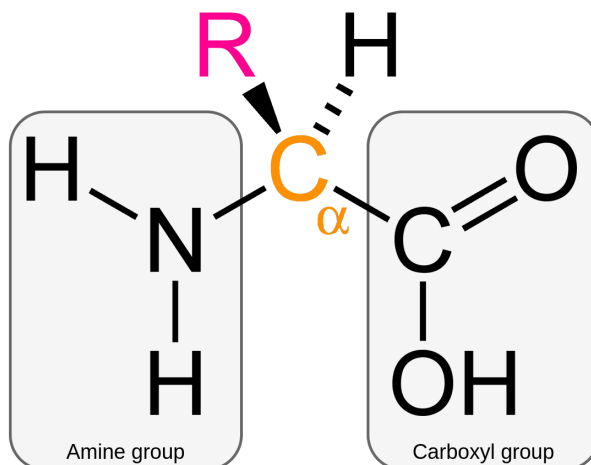


Figure 1.2 – Structure of an L- α -amino acids, modified from [23a]. R represents the side chain of the amino acid.

are difficult to distinguish, asparagine and aspartate (Asx, B), Glutamine and glutamate (Glx, Z), and leucine and isoleucine (Xle, J). Moreover, when no clues are available to identify the amino acid we have a code for unknown (Xaa, X).

The difference in the side chain (R) that defines amino acids causes different physico-chemical properties between them. For instance, we will present some of the most common properties. Some amino acids have charged side chains that allow them to interact with other charged molecules. These charged amino acids can form important electrostatic contacts called salt bridges which are for instance important for protein structure. Other amino acids have polar, uncharged side chains that allow them to form hydrogen bonds with other amino acids and water, they are hydrophilic. Meanwhile, other amino acids have hydrophobic side chains that repel water. These amino acids are one of the driving forces of protein folding. Another group of amino acids, Aromatic amino acids has an aromatic functional group. It has an alternating cyclic structure with double-bond characteristics. They can participate in various types of interactions with other amino acids, such as pi-stacking, hydrogen bonding, and van der Waals interactions. Contrary to aliphatic amino acids that have an aliphatic non-aromatic side chain. These amino acids are generally hydrophobic and are often found in the interior of proteins where they


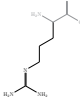


















Name	Three letter code	One letter code	Formula
alanine	ala	A	
arginine	arg	R	
asparagine	asn	N	
aspartic acid	asp	D	
cysteine	cys	C	
glutamine	gln	Q	
glutamic acid	glu	E	
glycine	gly	G	
histidine	his	H	
isoleucine	ile	I	
leucine	leu	L	
lysine	lys	K	
methionine	met	M	
phenylalanine	phe	F	
proline	pro	P	
serine	ser	S	
threonine	thr	T	
tryptophan	trp	W	
tyrosine	tyr	Y	
valine	val	V	
selenocysteine	Sec	10	U
pyrrolysine	Pyl		O
asparagine or aspartate	Asx		B
glutamine or glutamate	Glx		Z
leucine or isoleucine	Xle		J
unknown	Xaa		X

Table 1.1 – Table of the 20 (classic)+2 (special)+3 (uncertain)+1 (unknown)=26 different

can interact with other hydrophobic amino acids. All of these properties and the amino acids associated can be represented with the help of a Venn Diagram (Figure 1.3).

There are also some amino acids that have interesting properties. Glycine is very flexible because it has no side chain. Cysteines can form covalent bonds with other cysteines (disulfide bonds), which are important for protein structure and stability. Proline is very rigid because its side chain joins back onto the main part of the amino acid. They are the only ones to possess a secondary amine instead of a primary one. The details of all the amino acids formula can be observed in figure 1.1.

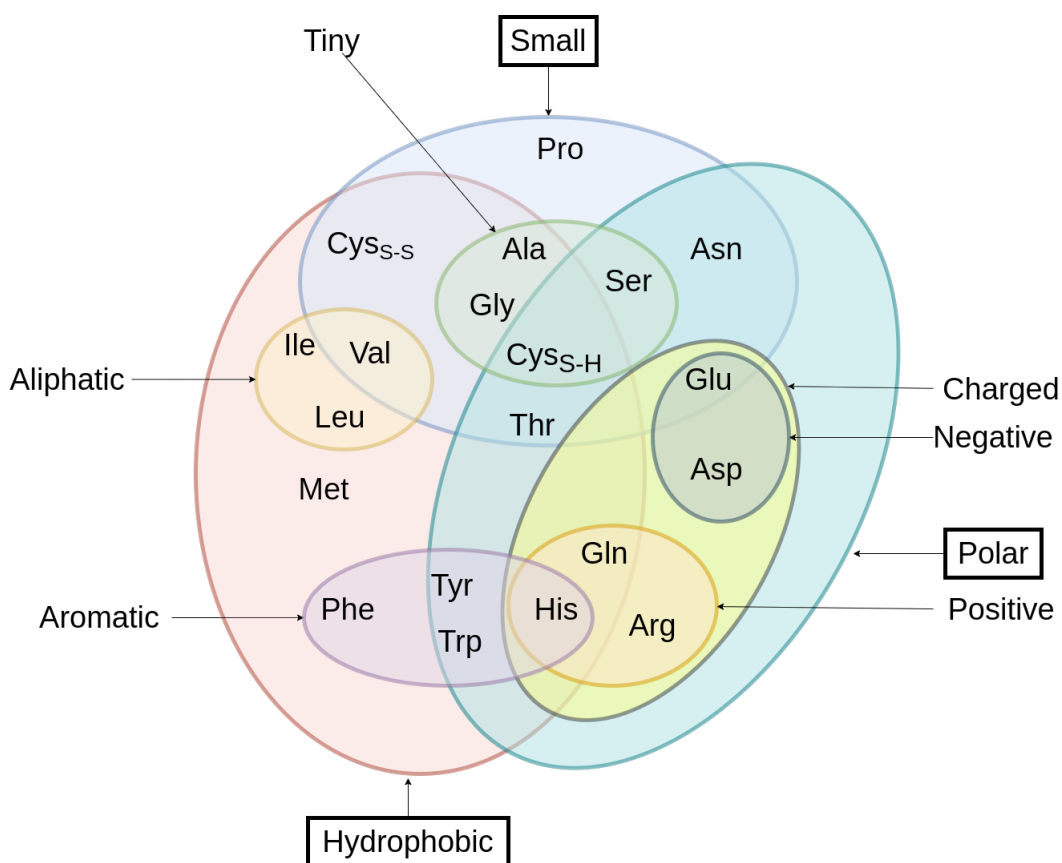


Figure 1.3 – Venn diagram of the physicochemical property of the 20 common amino acids. The cysteine is separated into two states, one with a disulfide bond (S-S) and one without (S-H). From the Taylor classification [Tay86]

Although amino acids serve as the fundamental units of proteins, they don't carry

out their biological function by themselves. The subsequent phase involves linking these amino acids together to form lengthy polymer chains.

1.1.2 Primary structure - a chain

The polymer chain is formed by combining amino acids with a condensation reaction. It consists of cutting OH from the carboxyl group of the first amino acid and cutting one hydrogen from the nitrogen of the other amino acids. Then linking the carbon of the carboxyl group to the nitrogen, as shown in Figure 1.4. The different amino acids at the different defined positions of a chain are called residues. In addition, the term "peptide" is usually used for chains with fewer than 20 amino acids, and "polypeptide" for chains longer than that.

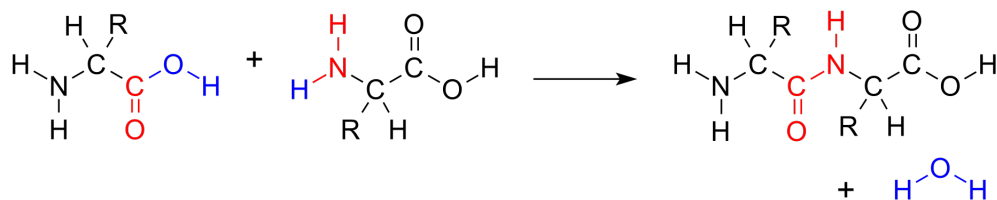


Figure 1.4 – Peptide chain formations. Two separated generic amino acids on the left and the resulting chain on the right with a peptide bond. This reaction forms water as a by-product. Figure from [22a].

The chain without side chains forms the backbone of the protein. More precisely the chain Nitrogen, α -Carbon, Carbon (=OH) as represented in Figure 1.5. On this backbone, dihedral angles can be defined. They are the angle between two peptide planes. ω (omega) is the angle in the chain $C_\alpha-C'-N-C_\alpha$, ϕ (phi) is the angle in the chain $C'-N-C_\alpha-C'$, and ψ (psi) is the angle in the chain $N-C_\alpha-C'-N$. These angles are very useful to describe the 3D structure and to discover patterns. These patterns will allow us to define some secondary structures (more in the upcoming section 1.1.3)

Traditionally, the protein sequence is written from its N-terminus to its C-terminus to avoid having both possibilities. The N-terminus is the free NH_2 at the end of the protein. The C-terminus is the carbon of the carboxyl group at the end of the protein.

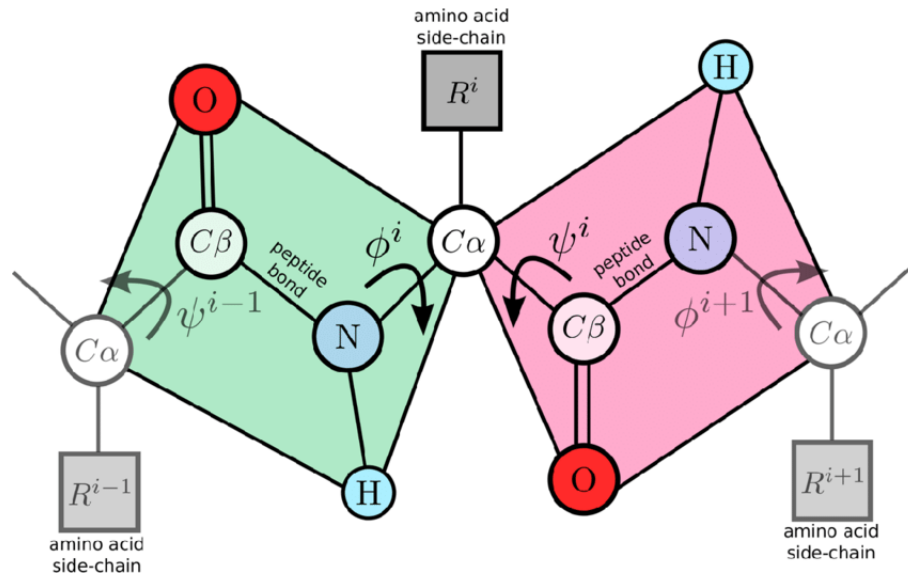


Figure 1.5 – Dihedral angles and peptide plane in a generic protein. From [Gol+16].

The order in which amino acids are strung together is defined in the DNA (Deoxyribonucleic Acid). The central dogma of molecular biology defines the information flow from DNA to proteins. The DNA information is initially transcribed into an mRNA before being translated by ribosomes into a polypeptide sequence. The translation is defined by the genetic code that associates each triple DNA base (codon) to one amino acid.

Thanks to advances in sequencing, many protein sequences are easily available. For example, the UniProt database [The19] contains about 246 million sequences in March 2023.

1.1.3 Secondary structure

The secondary structure is a local spatial conformation of the backbone. Multiple approaches exist to characterize secondary structure. One approach consists in observing the regular pattern of backbone dihedral angles in a particular region of the Ramachandran plot. This plot, depicted in Figure 1.6, is created by graphing the two important dihedral angles (ψ and ϕ) in the residue chain. On the left of this plot, two major patterns can be identified, α -helix (α in the figure), and β -sheet (β in the figure). α -helix is a type of

right-hand helix. It is formed when the N-H group of one residue forms hydrogen bonds with the C=O group of the fourth residue following it. β -sheet are arranged in multiple strands, typically of length 3 to 10. These strands are linked by hydrogen bonds like for α -helix. The difference is that this bonding is between different strands. An example of these two secondary structures can be found in Figure 1.7. But to assign consistent secondary structures a more rigorous scheme was needed.

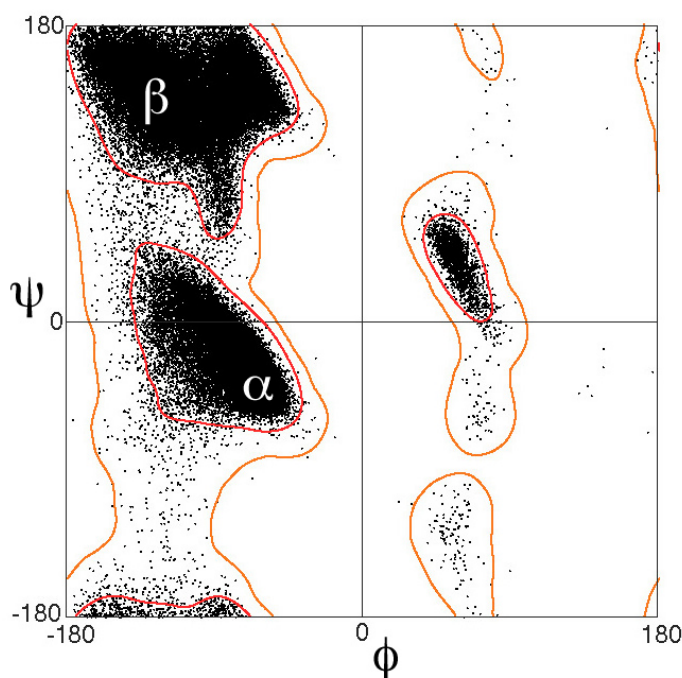


Figure 1.6 – The Ramachandran plot, as depicted in reference [22b], illustrates the observed angles of backbones, revealing two distinct clusters of points that correspond to the most frequently occurring secondary structures, namely the alpha-helix and beta-sheet

Multiple classification schemes were created, the two most important are DSSP and SST. The Dictionary of Protein Secondary Structure (DSSP) [KS83] is based on hydrogen bonding patterns. Whereas SST (Secondary STructure assignment)[KLA12] is based on the Shannon information criterion of Minimum Message Length (MML) inference. The task of finding a secondary structure class is viewed as a compression task. The method considers secondary structure assignments as potential explanations for the available co-

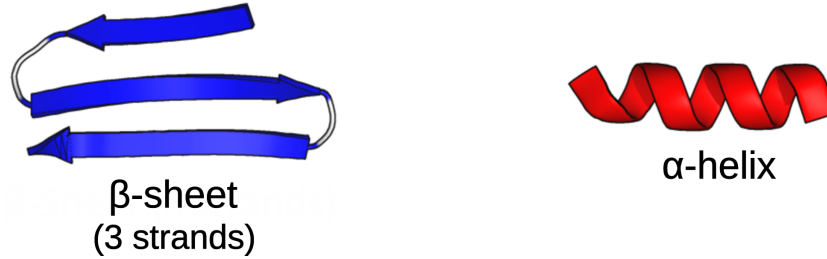


Figure 1.7 – α -helix and beta sheet from [23c]

ordinate data. Its goal is to find the most probable combination of a hypothesis and the data. The method compares all assignments against a default null hypothesis, and any assignment that fails to surpass it is deemed unsatisfactory. From this method, twelve secondary structure classes was defined.

Some tools exist with good performances to predict secondary structure (85% in the current tools from [Jia+17]). These local conformations are not enough to fully characterize the protein, thus we need to have a global 3D structure.

1.1.4 Tertiary structure

The tertiary structure is the global shape of the chains. It is defined by all the 3-dimensional coordinates of all atoms that compose the chain. A very important database for structure is the protein databank (PDB)[Ber+00]. As of April 2023, about two hundred and three thousand structures are experimentally defined in this database. An example of protein structure from this database is shown in Figure 1.8.

The tertiary structure for some proteins can involve the folding of multiple domains. Domains are regions of a protein that have distinct functions and can operate autonomously from the rest of the protein. Domains have an independently stable structure and function. Moreover analyzing the 3D structure, some regular patterns that connect some particular secondary structure can be found. These patterns are called structural motifs. Structural motifs are conserved 3-D structures found in different proteins, such as the helix-turn-helix motif. Motifs are typically shorter than domains and do not possess standalone structure.

Multiple databases try to unify groups of proteins that are similar in terms of struc-

ture. These groups are called protein families but the exact definition of a family is context-dependent. For the Pfam database [Fin+08], the protein family is a group of evolutionarily related proteins. Also exists a broader category, the superfamily, that defined distantly related proteins whose relatedness is not detectable by sequence similarity, but only from shared structural features. A database such as Structural Classification of Proteins extended (SCOPe)[Cha+22] is based on structure.

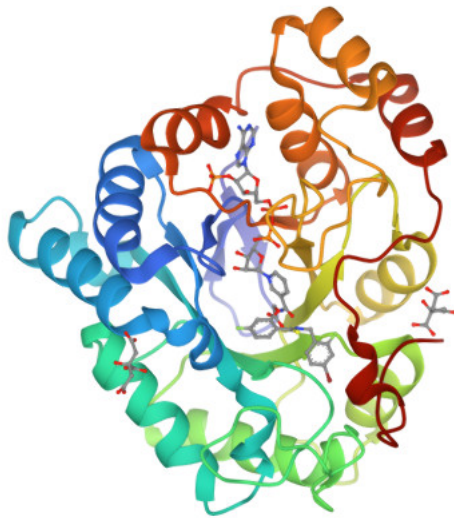


Figure 1.8 – PDB structure of Human Aldose Reductase in complex with NADP+ and the inhibitor IDD594 at 0.66 Angstrom (PDB Id: 1US0)

Anfinsen’s dogma states that the three-dimensional structure of a protein is determined solely by its amino acid sequence, with the native conformation being thermodynamically stable within its physiological environment. While this principle can be utilized as a theoretical framework, it is important to note that it is not without limitations. For example, some proteins need others (chaperone protein) to fold. Some have a closed and open conformation, and some are dynamic ([Lég+22]).

It can be seen that the number of experimentally determined proteins structure is multiple orders of magnitude lower 10^5 (in the PDB) compared to available protein se-

quences 10^9 (in BFD). To bridge this gap in knowledge, automatic structural prediction tools such as AlphaFold2 have been developed. AlphaFold2[Jum+21] is a state-of-the-art model for protein structure prediction, trained on the PDB database.

Some proteins can be functional as one chain but others need to assemble in multiple chains to do their function

1.1.5 Quaternary structure

Oligomers can be assemblies of multiple polypeptide chains that can be connected through various types of bonds, such as covalent bonds, hydrogen bonds, and ionic bonds. If the polypeptide chains are not covalently bonded, they are referred to as multimers. A homo-oligomer is composed of identical polypeptide chains, while a hetero-oligomer consists of different ones.

An example of an oligomer is collagen, which is the most abundant protein in mammals. Collagen is a fibrous protein that plays a vital role in connective tissues found in animals, including skin, bones, tendons, cartilage, and blood vessels. It is renowned for its tensile strength, providing structural support to tissues and organs. Collagen oligomers consist of three identical protein chains (refer to Figure 1.9).

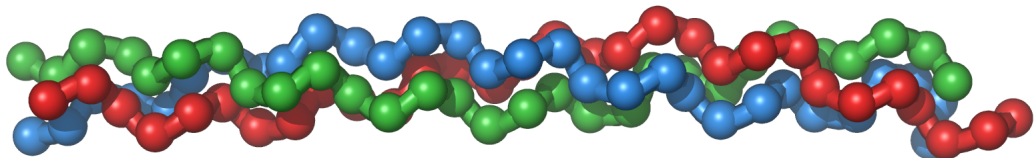


Figure 1.9 – Tropocollagen molecule: three left-handed procollagens (red, green, blue) join to form a right-handed triple helical tropocollagen. Figure from [23b]

But polypeptide chains can also be assembled with other bio-molecules to form one unit that can perform a function and can be called a complex. The different bio-molecules can for instance be RNA, DNA or other protein chain. For example, the ribosome complex is a large molecular machine composed of multiple protein subunits and RNA molecules. It plays a crucial role in translating messenger RNA (mRNA) into protein. The ribosome

is composed of two subunits, a large subunit and a small subunit, each of which consists of multiple proteins and RNA molecules (rRNA). These subunits come together to form a functional ribosome complex.

The most crucial aspect of protein analysis is understanding its function. Protein function plays a vital role in various fields, including medicine, disease comprehension, studying biological systems, and investigating protein interactions with other molecules. While protein structure can provide hints about its function, determining function solely based on the structure is often complex. Certain structural features, such as binding pockets facilitating molecule interactions or shape-dependent receptors, offer insights into protein function, but the relationship is not always straightforward.

1.1.6 Functions

A biological function pertains to the precise role or purpose fulfilled by a protein within a living organism. These functions intricately rely on the unique structure and properties of proteins. For instance, proteins can form functional pockets that facilitate the binding of molecules, thereby accelerating reactions. Additionally, the shape and structure of proteins greatly influence the functionality of receptors, emphasizing their interdependence.

One possible way to classify biological function is in 8 broad categories ([Urr+16]): enzymatic functions, defense proteins, storage proteins, transport proteins, hormonal proteins, receptor proteins, contractile and mobile proteins, and structural proteins. Enzymatic functions involve accelerating chemical reactions, such as kinases that facilitate phosphorylation. Defense proteins play a crucial role in the immune system, protecting against intruders, such as antibodies that recognize and neutralize pathogens. Storage proteins serve as reservoirs for amino acids, with casein being an example. Transport proteins facilitate the movement of bio-molecules within the body, such as hemoglobin which transports oxygen in the blood. Hormonal proteins help coordinate the activities of an organism, for example, insulin helps regulate glucose metabolism. Receptor proteins are responsible for cellular responses to chemical stimuli, such as those found in neurons. Contractile proteins enable movement, as seen in muscular tissue. Finally, structural proteins provide the framework for large tissues, with collagen being a prominent example.

The lack of specificity in these general categories fails to capture the wide range of protein functions, which is why ontologies for protein functions were developed.

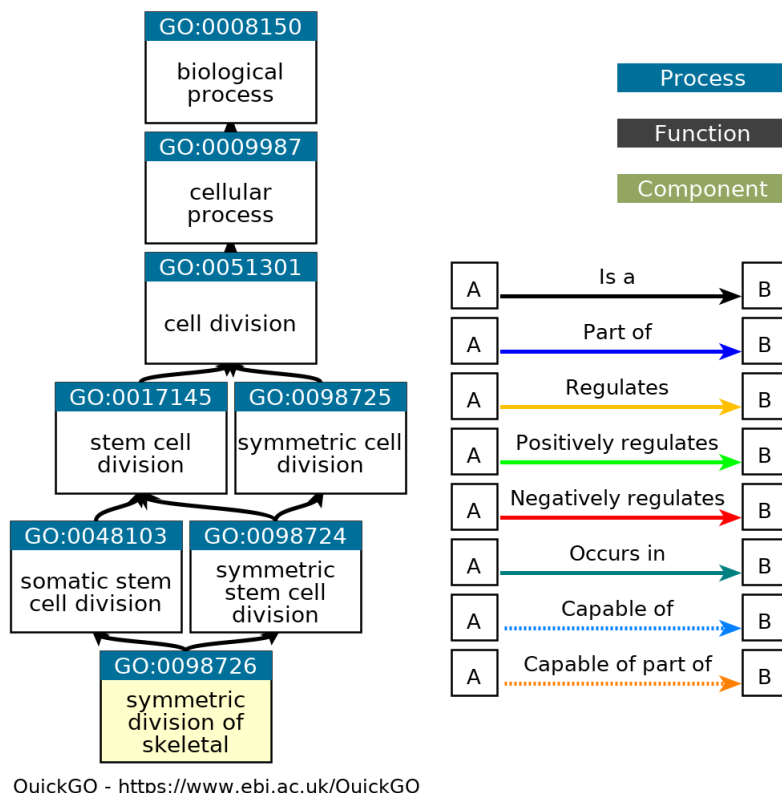


Figure 1.10 – Ancestor chart for GO:0098726 (Symmetric division of skeletal) from QuickGO website

Function ontology

The main functional vocabulary to annotate proteins function is Gene Ontology (GO). GO concepts are designed to describe an aspect of the function that a gene evolves to perform (a selected effect function). But for example, some molecular functions are not linked to biological processes and then are only candidate functions. Gene ontology encompasses three main categories to describe the functions of gene products. The Gene Ontology Consortium (GOC) uses the gene product term to refer most of the time to protein and less often functional RNA like in the ribosome or other bio-molecules. The first category of the GO, Molecular Function (MF), includes two subcategories: biochemical activity

and components in a larger system process. This entails describing the specific activities or roles that a gene product performs within a biochemical context or its contribution to a larger biological process within the system. The second category, Cellular Component (CC), encompasses the localization of gene products when they perform their functions. It is divided into two subcategories: relative to cellular compartment/structure (e.g. cytoplasm, mitochondrion) and relative to common/stable molecular complexes they are part of (e.g. the ribosome). This category provides information on where the gene product is located within the cell or its association with stable molecular complexes. The third and most complex category is Biological Process (BP), which is further divided into three subcategories. One category describes the gene product's involvement in biological processes, including its role as a constituent of a specific biological program. A second is on its regulatory function in controlling a biological program, and the last is on its upstream and essential contribution to initiating a biological program.

Overall, gene ontology provides a comprehensive framework for describing the functions of gene products, encompassing their biochemical activities, cellular localization, and involvement in biological processes. It is an evolving system that aids in the systematic organization and interpretation of gene function data in a standardized manner. An example can be seen in Figure 1.10.

Previously, other experts developed a hierarchy only for enzyme functions, the Enzyme commission number. Enzyme Commission (EC) numbers provide well-defined targets to train machine learning models ([KL94]). This nomenclature is composed of 4 levels. The first level provides the main class of the enzyme, which is encoded by a number between 1 and 7. The second and third digits correspond respectively to the subclass and sub-subclass of the enzyme, and the last one represents specific metabolites and co-factors involved, which basically provides the actual reaction catalyzed or a restricted set of very similar reactions. Therefore, an enzyme characterized to the fourth level is considered to be completely characterized. The hierarchy can be seen in Figure 1.11. Some mapping between the Gene Ontology term and Enzyme Commission number terms is possible. But in general, it is not a one-to-one relationship. Some GO terms correspond to multiple EC and vice versa.

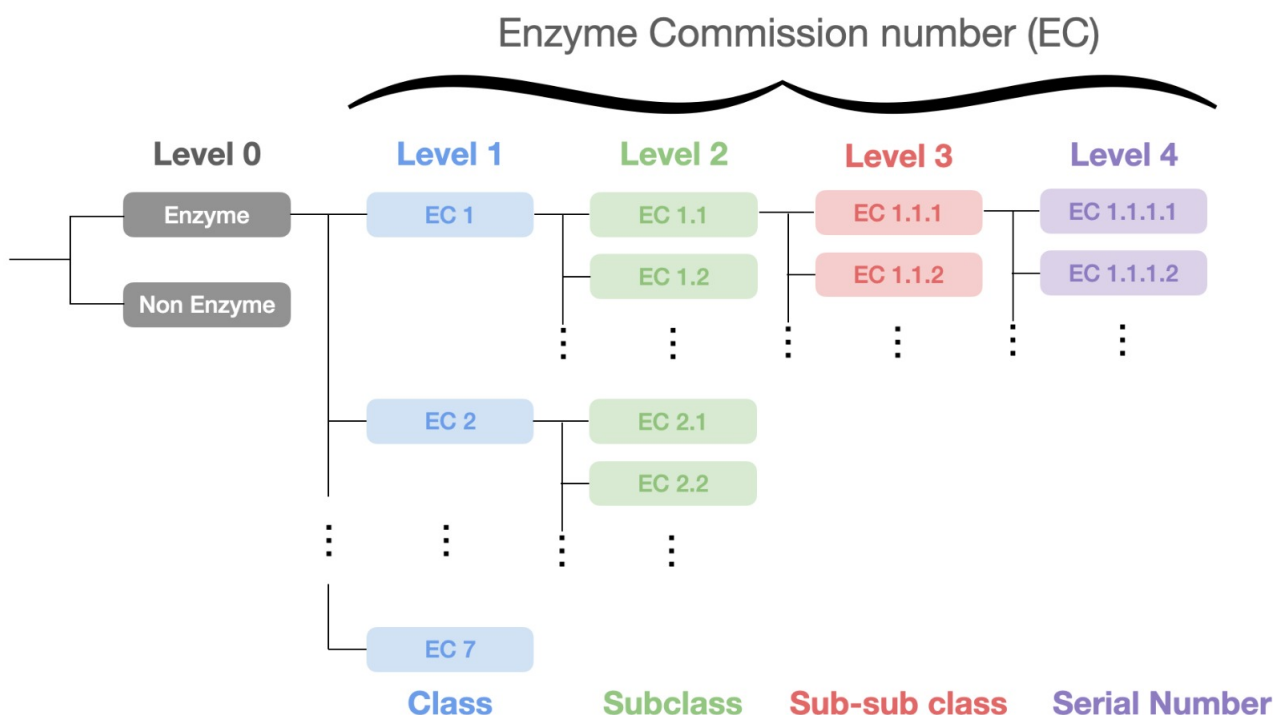


Figure 1.11 – Enzyme Commission number hierarchy

The essential

Proteins, which are essential molecules in living organisms, can be described using a sequence of amino acid letters. The arrangement of amino acids in a protein's sequence determines its quasi-unique 3D structure, which can give valuable insight into its function. The function of proteins is often described using the Gene Ontology, and more specifically for enzymes by the Enzyme Commission number. These ontologies and hierarchical structures play a crucial role in unifying and organizing the functional vocabulary associated with protein.

We have seen that protein can be described at multiple levels with different specificities.

Database	Data type	Number of proteins
Big Fantastic Database (BFD) [SS18]	sequence	2.5×10^9
UniProt [The19]	sequence	2.5×10^8
UniProt/SwissProt [The19]	sequence+function annotations	5.7×10^5
PDB[Ber+00]	sequence+structure	2.0×10^5

Table 1.2 – Summary of different order for each type of data. Comparison of the number of proteins available for each dataset

The different quantities of data available for some of the levels are presented in table 1.2. It can be observed a big gap between sequence-only databases and sequences with function annotations. In order to close this gap we need a model that links sequence to function. The next section will describe how the different levels and their specificity can be taken into account to model proteins. And how to link protein sequences to protein functions.

1.2 Automatic Function Prediction (AFP)

In this section, we will begin by discussing classical bioinformatics methods commonly used for predicting protein function. However, due to the limitations of these methods, which often rely on sequence similarity we will present some machine learning methods. But as they rely on pre-extracted engineered features that can be limiting, we will delve into deep learning models, and present more specifically the Transformer architecture.

1.2.1 Classical bio-informatics methods

This section will briefly present classical bio-informatics methods used for protein function prediction. The sequence which represents the amino acid composition and the specific order in which they are arranged within a protein chain holds valuable information about its functional characteristics. The central hypothesis of most of the classical tools is based on homology transfer. It suggests that if two proteins exhibit a significant sequence similarity, they likely originate from a shared ancestor and might possess similar functions. Classical tools such as the Basic Local Alignment Search Tool (BLAST), can be used

to identify similar sequences with functional annotations. While BLAST may be less sensitive than an exact solution, it is significantly faster. BLAST operates in three main steps. Firstly, the query sequence is divided into k-mers, which are then stored, along with similar ones, in a dictionary. Secondly, all sequences in a database (such as UniProt) are searched using the k-mer dictionary to identify matches. When a match is found, BLAST attempts to extend the k-mer from both sides until the extension decreases the matching score. The matching score is typically computed using the Blosum62 matrix. Finally, the third step involves computing the e-value, which represents the expected value of finding a match by chance alone. One approach that has been employed in conjunction with BLAST is the utilization of a K-Nearest Neighbor (KNN) algorithm, resulting in Blast-KNN. This combined approach has been utilized in predicting the enzyme class of novel protein sequences, as demonstrated in [Dal+18], where it was incorporated into an ensemble model along with other methods.

Another very popular model the profile Hidden Markov Model (pHMM) is a widely used method to infer functions from homologous sequences. It is based on the first-order Markov assumption, which means that the probability of the next state only depends on the current state. A pHMM consists of a finite set of states, each with a set of probabilities associated with it. There are three types of states: match states for conserved residues, insert states for extra residues, and delete states for missing residues. To build a pHMM, one needs to specify the state transition probabilities and the emission probabilities. State transition probabilities give the probability of moving from one position in the sequence to another based on the current position, while emission probabilities describe the chance that a particular amino acid is at this position. A pHMM can be constructed from a multiple sequence alignment (MSA) by estimating the transition and emission probabilities from the alignment data. This model can capture the features of a protein family that share the same function. Then, the model can be used to calculate the probability that a new sequence belongs to each functional family. However, it should be noted that these methods may struggle to capture distant sequences that have retained their structures and functions throughout evolution. To overcome these limitations, some methods have leveraged machine learning techniques.

1.2.2 Machine learning methods

This section presents how machine learning is applied to automatic function prediction, with a focus on enzyme function prediction as our initial domain of application.

Different methods for protein sequence analysis rely on classical machine learning approaches that utilize manual feature vectors as input. For instance, SPMap, as introduced in the study by [Dal+18], creates clusters of subsequences and assigns probabilities to the likelihood of a query subsequence matching with the subsequence clusters. This allows for the construction of a C-dimensional vector, where C represents the number of clusters created during training and an SVM is subsequently applied to classify the sequence into the appropriate EC class. Similarly, another method called Pepstats-SVM, also discussed in [Dal+18], extracts a feature vector comprising 37 characteristics of the protein, such as molecular weight, isoelectric point, and physicochemical properties. This feature vector is then used as input for an SVM-based classification approach. These methods demonstrate the utilization of manual feature vectors in combination with machine learning algorithms for protein sequence analysis.

One advantage of manual feature extraction methods is that they can simplify the modeling process by focusing on relevant information. Additionally, these methods can help mitigate the risk of overfitting when working with a limited amount of manually extracted properties. However, these methods may be limited by the input information fed into the SVM or other classifier models, as the manually extracted features may not be exhaustive. To address this limitation, an alternative approach is to use raw sequence information directly and infer intermediate representations through representation learning using deep learning techniques. This allows for the extraction of more comprehensive and nuanced features, potentially leading to improved performance in protein sequence analysis tasks.

In particular, Strodthoff et al. [Str+20] developed one of the state-of-the-art prediction tools based on sequences only. They used a deep learning model to automatically infer the relevant internal vectorized representations of the sequences (the “sequence embedding”), from which the EC prediction is derived.

In terms of architecture, they chose to base their language model on a ASGD Weight-Dropped Long Short-Term Memory (AWD-LSTM) neural network architecture [MKS17].

Yet, although architectures based on LSTM enable to better handle sequences by partially solving the vanishing gradient problem, they still seem to struggle to properly account for long-distance interactions which are known to be relevant in protein structures. In Natural Language Processing (NLP) applications, LSTM architectures are being superseded by a non-recurrent architecture: the Transformers. Transformers primarily use the *attention* mechanism [Vas+17] and seem to better account for long-range interactions as witnessed for instance by the success of BERT [Dev+19] or T5 [Raf+20] on NLP benchmarks.

In this work, we propose to evaluate the interest of Transformers for protein functional annotation based on sequences. The potential of Transformers applied to protein sequences has already been identified by several seminal studies: in their early paper calling for the development and proper assessment of better protein modeling methods, Rao et al. [Rao+19] introduced five tasks. On this benchmark (TAPE), they compared classical methods using alignment-based features (Netsurfp2.0, RaptorX and DeepSF for secondary structure, contact prediction and remote homology respectively) and multiple deep learning architectures, namely CNN (convolutional neural networks), LSTM and Transformers. While Transformers outperformed the other methods on several tasks (fluorescence and stability), they were still lacking on others: homology prediction (LSTM performing better) and structure prediction (methods based on alignment-based features performing better). More recently, [Eln+21a] have conducted a high-performance computational study of Transformers on large protein sequence databases to obtain on par, or state-of-the-art results on secondary structure prediction, protein localization, and membrane-bound vs water-soluble tasks. Besides, [Riv+21] also reported better performance of Transformers compared to LSTM on contact prediction. But, to the best of our knowledge, no previous work tested Transformers on a functional annotation task that can be evaluated as accurately as enzyme functional annotation.

1.2.3 Transformer architecture and two phases training

The fundamental concepts of neural networks including their architecture and training principles will be covered first. Specifically, we will explore the importance of loss functions for both self-supervised and supervised training, along with different training objectives

and different architectures. Lastly, we will delve into a comprehensive exploration of the Transformer neural network, highlighting its potential applications in protein function prediction.

Deep learning involves the use of neural networks. If you are already familiar with neural networks, you may skip the next two sections where we will discuss the basic principles of neural networks.

1.2.3.1 The problem of protein function prediction from sequences

The objective of this thesis is to tackle the classification problem of predicting the functional class of a protein based on its input sequence. The functional class can be determined through either an EC number if limited to enzymes, or Gene Ontology terms, encompassing all proteins. The primary task is to develop a predictive model capable of accurately assigning the correct functional class to a given protein sequence.

Two datasets will be used, the first one, denoted as D_u , contains only raw sequences and has the form $\{S_0, ; S_{N1}\}$, where $N1 \in \mathbb{N}$ is the number of sequences in the first phase. An input sequence S_i is defined as $S_i = \{r_0, ; r_L\}$, where L is the length of the sequence. The second one, denoted as D_s , contains sequences and their functional annotations and has the form $\{(S_0, y_0), ; (S_{N2}, y_{N2})\}$, where y_i is the functional annotation for the sequence i .

1.2.3.2 Basic of deep learning neural network

In order to facilitate a clear understanding of deep learning neural networks, it is essential to establish a set of notations that will be used throughout this discussion. The following notations will be employed:

a : Scalar

\mathbf{a} : Vector

\mathbf{A} : Matrix

\mathbf{A} : Tensor

$;$: means parametrized by

The artificial neuron serves as the fundamental building block of neural networks.

It can be conceptualized as a parametrized function that takes an input vector $\mathbf{x} \in \mathbb{R}^N$, where $N \in \mathbb{N}$ represents the input dimension, and produces a scalar output. This function relies on a weight vector $\mathbf{w} \in \mathbb{R}^N$, a bias term $b \in \mathbb{R}$, and a non-linear activation function ϕ . The output can be expressed as:

$$f(\mathbf{x}; \mathbf{w}, b) = \phi \left(\sum_{i=1}^N w_i x_i + b \right) \quad (1.1)$$

The weights and bias are the learnable parameters of the neural network, often denoted as θ . When we extend to M neurons, we can rewrite the previous equation using matrix multiplication by introducing a weight matrix. The output dimension of our function becomes $M \in \mathbb{N}$, the input is represented as $\mathbf{x} \in \mathbb{R}^{N \times 1}$, and the weight vector is replaced by a weight matrix $\mathbf{W} \in \mathbb{R}^{N \times M}$, where each row corresponds to a weight vector \mathbf{w} . The bias term is now $\mathbf{b} \in \mathbb{R}^M$, and the function's signature becomes $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$:

$$F(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \phi(\mathbf{W}^T \mathbf{x} + \mathbf{b}) \quad (1.2)$$

Each function F is commonly referred to as a layer. By composing these layers, we can construct a multi-layer neural network that performs more complex computations. The inclusion of non-linear activation functions is crucial, as without them, stacking linear transformations would result in a network comprised solely of linear units. Therefore, we incorporate activation functions that are non-linear and differentiable (We will see why later on). Examples of such activation functions include ReLU (Rectified Linear Unit), GELU (Gaussian Error Linear Unit), Sigmoid, and others:

$$\phi(x) \in \{\text{ReLU}, \text{GELU}, \text{Sigmoid}, \text{others}\}$$

$$\text{ReLU}(x) = \max(0, x) \quad (1.3)$$

$$\text{GeLU}(x) = x \cdot C(x) \approx 0.5 \cdot x \cdot \left(1 + \text{Tanh} \left(\sqrt{\frac{2}{\pi}} \cdot (x + 0.044715 \cdot x^3) \right) \right) \quad (1.4)$$

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \quad (1.5)$$

Here, $C(x)$ represents the Cumulative Distribution Function for the Gaussian distri-

bution.

The combination of a linear transformation followed by a non-linear activation function defines a feed-forward layer. When multiple fully connected layers are stacked together, we obtain a Multi-Layer Perceptron (MLP). These MLPs can approximate any continuous function [Cyb]. The term "fully connected" originates from the fact that each neuron in a layer is connected to all the inputs from the previous layer. This connectivity pattern is known as the architecture of the neural network.

1.2.3.3 Probabilistic framework for classification

In order to train our neural network, we adopt a probabilistic framework where the network will output probabilities. To achieve this, we utilize specific activation functions that enable probability-based outputs. For a Bernoulli conditional distribution, we employ the sigmoid function (equation 1.6), while for the categorical distribution (multinoulli), we use the softmax function (equation 1.7). The softmax function is an extension of the sigmoid function designed to handle multi-class problems.

The sigmoid function, denoted as $\sigma(x)$ (equation 1.6), transforms the input x into a probability value between 0 and 1. It accomplishes this by applying the mathematical operation of exponentiation and division. Specifically, the sigmoid function maps the input x to a value between 0 and 1 using the formula:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (1.6)$$

This function is well-suited for binary classification tasks where the goal is to assign an input to one of two classes. On the other hand, the softmax function, denoted as $\text{softmax}(x)$ (equation 1.7), is employed when dealing with multi-class problems. It takes a vector of inputs x and transforms it into a probability distribution over multiple classes. The softmax function calculates the exponential of each element in the input vector and normalizes the resulting values by dividing them with the sum of all exponentials. This ensures that the output probabilities sum up to 1. The softmax function is particularly

useful when we need to assign an input to one of multiple mutually exclusive classes.

$$\text{softmax}(x) = \frac{\exp(x)}{\sum_{i=1}^C \exp(x_i)} \quad (1.7)$$

In the context of neural networks, optimizing the parameters is crucial to make the network perform well on its intended task. This process involves finding the values of the parameters that minimize the difference between the network's predictions and the desired outputs. To achieve this, we need to define an optimization problem, which essentially means formulating the task of finding the best parameters as a mathematical problem.

The principle of maximum likelihood is a widely used approach in neural network optimization. It aims to find the parameter values that maximize the likelihood of observing the given training data, given the model. In other words, we want to find the parameters that make the neural network's predictions align as closely as possible with the actual outputs in the training data. By maximizing the likelihood, we increase the probability of obtaining the observed data under the current model.

On the other hand, Bayesian approaches, which take a probabilistic viewpoint, might prefer the maximum a posteriori (MAP) estimation. MAP estimation combines the likelihood of the observed data with a prior probability distribution over the parameters. The prior represents our beliefs about the parameter values before observing the data. By incorporating the prior, we can incorporate existing knowledge or assumptions into the parameter estimation process.

Interestingly, when a uniform prior distribution is assumed for the MAP estimation, the maximum likelihood estimation and the MAP estimation become equivalent. In other words, if we have no specific prior knowledge and assume that all parameter values are equally likely, the MAP estimation reduces to the maximum likelihood estimation. This equivalence simplifies the problem and allows us to focus solely on maximizing the likelihood, without considering the additional step of incorporating a prior distribution.

Bayesian

For the Bayesian, the random variable is the parameters of the model and the dataset is fixed. In this case $P(X; \theta) = P(X|\theta)$

With the function annotations $Y = \{y_0, y_1, \dots, y_n\}$ and the input sequences $S = \{S_0, S_1, \dots, S_n\}$ sample independently and from the same distribution (i.i.d). The maximum likelihood function gives us the equation 1.8. To avoid multiplying the probability, which can cause numerical overflow, the log is taken. This doesn't change the optimization problem. This allows us to have a sum instead of a product.

$$\theta^* = \operatorname{argmax}_{\theta} P(X|S; \theta) = \operatorname{argmax}_{\theta} \sum_{i=1}^{|X|} \log(P(x_i|s_i; \theta)) \quad (1.8)$$

$$\theta^* = \operatorname{argmin}_{\theta} - \sum_{i=1}^{|X|} \log(P(x_i|y_i; \theta)) \quad (1.9)$$

This optimization problem is equivalent to minimizing the negative log-likelihood (NLL) (equation 1.9) or the KL divergence between the empirical distribution defined by the training set and the distribution defined by the model. And it also corresponds to the minimization of the cross-entropy. More detail on these equivalence in [Cou16].

Our aim is to minimize the loss function across all examples in the training dataset. One approach to achieve this is by leveraging the gradient, specifically the gradient of the loss with respect to the parameters. The gradient is a vector of partial derivatives of the cost function with respect to each parameter. Back-propagation ([RHW86]), which involves applying the chain rule of partial derivatives in a neural network, is used to compute the gradient. The term "back-propagation" refers to the fact that the error is propagated backward from the end of the network through the different layers. This is in contrast to forward propagation, where predictions are computed from inputs.

Finding an exact expression for the gradient to minimize the cross-entropy is complex, as neural network functions are generally non-convex and have multiple local minima. Instead, iterative optimization algorithms that use optimizers are employed to modify the

parameters of the neural network. One commonly used optimizer is stochastic gradient descent (SGD - [RM51]). SGD involves taking a step in the opposite direction of the gradient, allowing us to update the parameters in a way that reduces the loss. The size of the step, known as the learning rate, determines the magnitude of the parameter updates. The parameters are updated by subtracting the gradient multiplied by the learning rate, resulting in a decrease in the cost function. The term "stochastic" stems from the fact that the gradient is estimated from just a sample of the dataset, known as a minibatch, rather than the entire dataset. This process is repeated multiple times, and when all examples in the training set have been processed, we say that one epoch of training has been completed.

Other optimization algorithms, such as Adam [KB17], also exist. Adam incorporates momentum into the weight updates, allowing for faster convergence during training.

Regarding the input layer, a popular approach for handling categorical variables like sequences is to use embeddings, which convert each token in the sequence into a continuous real vector. The upcoming section will provide more detailed information on this technique.

1.2.3.4 Embedding

The protein sequence consists of an ordered list of residues. In our neural network, the term "token" is adopted as the fundamental unit of description. In Natural Language Processing (NLP), a token can represent a letter, a part of a word, a whole word, or even multiple words. In the context of proteins, the most common approach is to consider each residue as a token. However, alternative choices for tokenization are discussed in section 3.7.2. Additionally, for specific tasks, special tokens such as [CLS] or [MASK] can be defined, with their roles and functions explained in section 1.2.3.6. The vocabulary of the model (denoted as V) is defined by the complete list of tokens.

As mentioned previously, the popular approach involves using real-valued vectors to represent tokens. These vectors, referred to as embeddings, have a fixed size denoted as E , which is a hyperparameter of the model. The embedding matrix, consisting of all the vectors corresponding to the tokens in the vocabulary, plays a crucial role. This matrix has dimensions of $|V|$ rows and E columns (see the transpose version on the left of Figure

1.12). Like other parameters in the network, the embedding matrix is initialized with random values. The embedding is then trained to refine the representations during the training process. The embedding matrix is an essential component as it captures the semantic relationships and representations of the tokens in a continuous vector space.

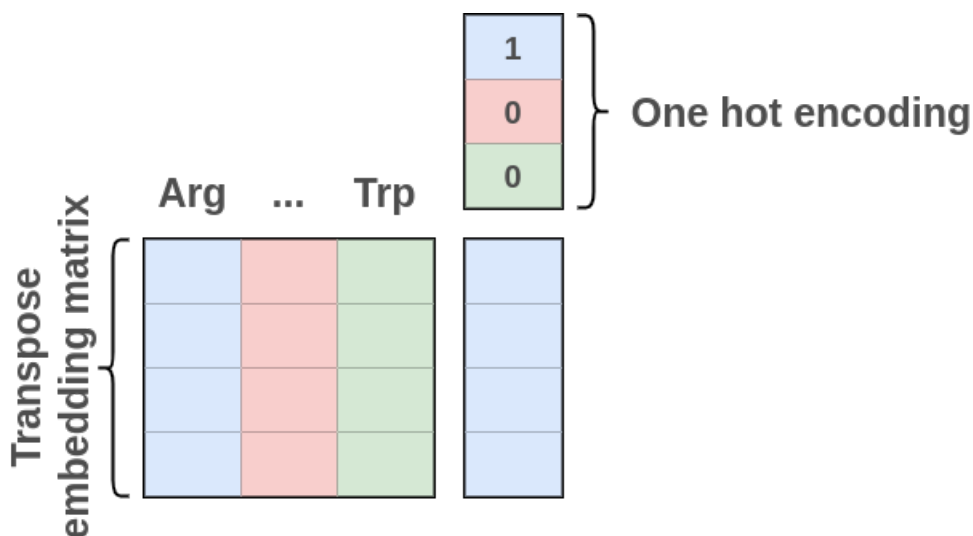


Figure 1.12 – Multiplication of a transpose embedding matrix with a one-hot encoded residue to obtain the embedding of this specific residue

The figure shown in Figure 1.12 illustrates the process of obtaining the embedding of a specific residue by multiplying a one-hot encoded residue vector with the transposed embedding matrix.

In the forward pass of our model, each residue is represented as a one-hot vector. This vector contains zeros in all positions except for a single one at the index corresponding to the residue. By multiplying this one-hot vector with the transposed embedding matrix, we can obtain the embedding representation of the residue. Alternatively, this operation can be viewed as selecting the corresponding row from the embedding matrix, which is often the implementation approach.

In the context of a Transformer, the embeddings of the tokens undergo refinement across the layers of the neural network and serve as the fundamental units for the network’s reasoning. Therefore, the selection of the token scale plays a crucial role. It involves

balancing various factors, including the vocabulary size, the information embedded within the initial token embeddings, and the diversity of tasks that the network needs to perform. Currently, each token corresponds to a single residue, as no other strategy has been found to be more effective. However, further detailed information and discussions on this topic can be found in part 3.7.

At the end of the Transformer, we obtain L output embeddings, where L is the length of the input sequence. The exact computation needed to refine the representation is detailed in the following.

1.2.3.5 Transformer architecture

The architecture of a neural network is determined by the interconnections between its neurons and the flow of information from the input to the output. This architectural design dictates the type and format of data that can be processed by the neural network. The connection structure of a neural network also incorporates some inductive biases, which may not always be explicitly defined or well understood. These biases imply that certain types of patterns are more readily extractable by the network, while others may be more intricate or challenging to extract. For instance, in the case of a convolutional neural network (CNN) layer, where information flows locally within each layer, extracting local dependencies becomes relatively straightforward. However, capturing long-range dependencies can become more difficult or even impossible. As the number of layers in a CNN grows, the receptive field expands, allowing the network to capture relationships between elements that are increasingly distant from each other.

Different architectures may excel at modeling different types of relationships. Hence, choosing the appropriate architecture is crucial in ensuring the optimal performance of the neural network. Proteins exhibit properties at multiple scales, ranging from their sequence to their tertiary and even quaternary structure (seen in 1.1), with long-distance contacts playing a crucial role. While architectures like Recurrent Neural Networks (RNNs such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU)) and CNNs are commonly used ([Str+20],[Bil+19]), they have limitations in effectively capturing long-range contacts (See Figure 1.13). In contrast, Transformer-based architectures have shown to be very effective in this regard, by for example retrieving long-range contact(>24 AA)

in an unsupervised way in their inner representation ([Rao+20]). Therefore, selecting the appropriate architecture, such as the Transformer, is crucial for accurately modeling long-range contacts in proteins.

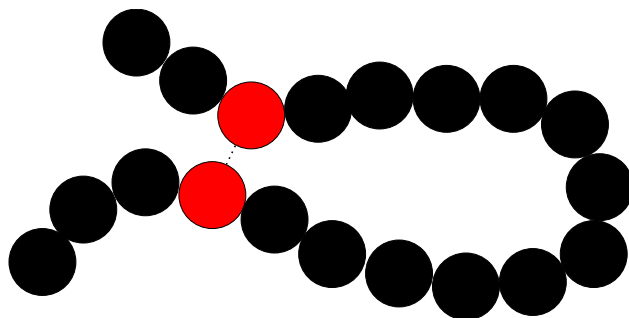


Figure 1.13 – Long range contact in proteins

The Transformer architecture is designed in a way that treats distant residues in a protein sequence as equally relevant to close ones, enabling seamless communication between them. The original Transformer architecture is composed of two parts, an encoder part, and a decoder part. First, we present the Transformer-Encoder. The global architecture of the Transformer-encoder is composed of two main blocks. The first is the attention block and the second is the feed-forward block. A global view can be seen in Figure 1.14. An important part of the Transformer is the attention mechanism. This mechanism is the specific part within the Transformer architecture where the exchange of information between the embeddings of different tokens (residues in our case) takes place.

1.2.3.5.1 The attention mechanism

In this section, we present the Transformer self-attention mechanism. Let L denote the sequence size, E represent the embedding size, and $x \in \mathbb{R}^{E \times 1}$ be the input. The first step involves creating the key (K), query (Q), and value (V) matrices. These matrices are obtained through three separate linear projections, which take the current embedding x as input and are computed as follows:

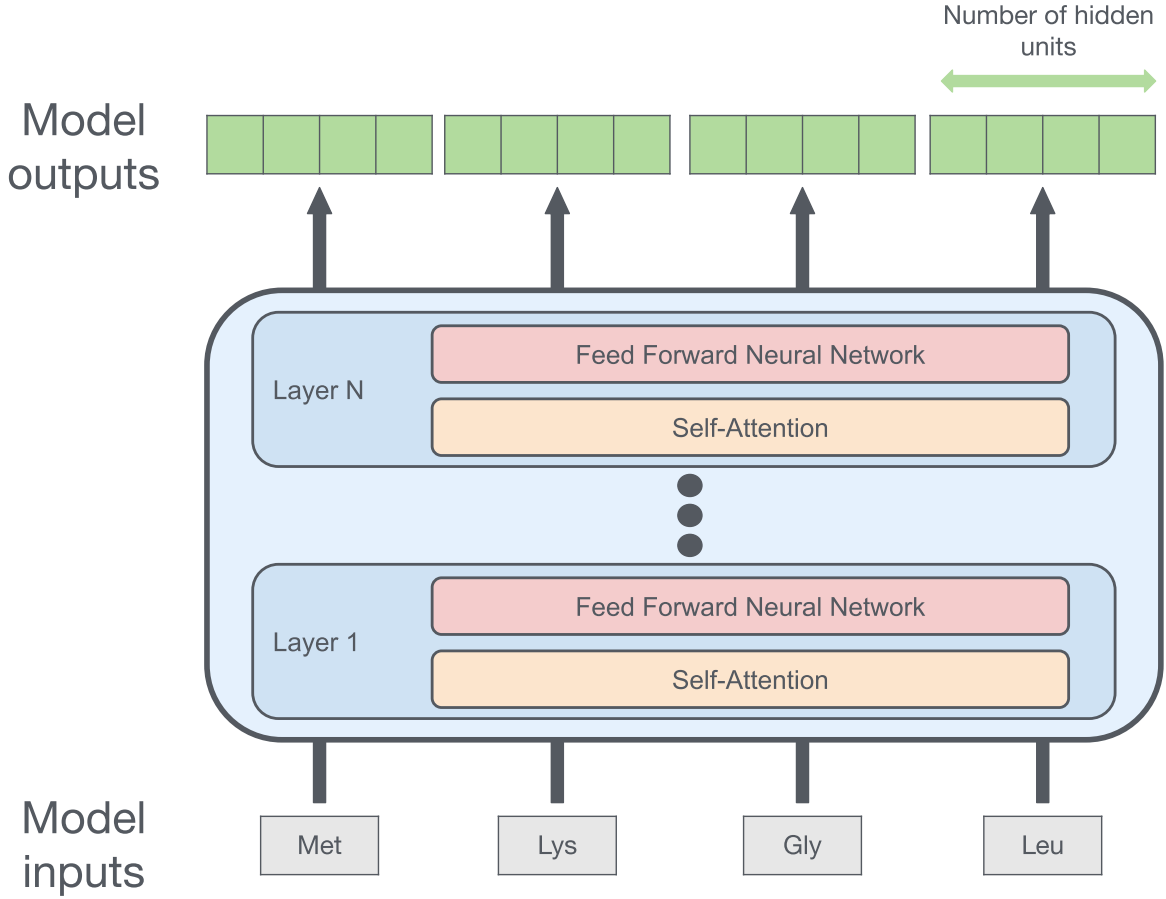


Figure 1.14 – High-level Transformer encoder architecture

$$Q = \mathbf{W}^{\mathbf{q}\top} x \quad (1.10)$$

$$K = \mathbf{W}^{\mathbf{k}\top} x \quad (1.11)$$

$$V = \mathbf{W}^{\mathbf{v}\top} x \quad (1.12)$$

The linear projections are parameterized by $W^q \in \mathbb{R}^{E \times E}$, $W^k \in \mathbb{R}^{E \times E}$, and $W^v \in \mathbb{R}^{E \times E}$, which represent the weights trained during the training process.

Next, we compute the attention map (equation 1.13), which is an $L \times L$ matrix (refer

to Figure 1.15). Each row of the attention map represents a distribution over all tokens, obtained through the softmax operation.

$$\text{Attention_map} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{E}}\right) \quad (1.13)$$

$$(1.14)$$

This attention mechanism captures the importance of different tokens for each token in the sequence. The attention map, computed as the softmax of the scaled dot product between the query (Q) and key (K) matrices, assigns weights to each token, indicating its relevance to other tokens in the sequence. Multiplying the attention map by the value (V) matrix allows us to combine the information from tokens that hold significant importance as represented by the row distribution. This process results in the attention representation, which summarizes the aggregated knowledge from relevant tokens, providing a comprehensive context for each token in the sequence.

$$\text{Attention} = \text{Attention_map} \times V \quad (1.15)$$

The entire attention mechanism is illustrated in Figure 1.16. It consists of the key, query, and value projections, followed by the computation of the attention map, and the multiplication of the attention map with the values, which collectively form an attention head. Typically, multiple attention heads are used in parallel within each layer. In such cases, the projected dimension is equal to the embedding dimension E divided by the number of heads. The outputs of all the attention heads are then concatenated and projected back to the original dimension E (as shown in Equation 1.17), using another set of weights denoted as W^O . These weights are responsible for combining information from different attention heads. We define the output of the attention mechanism as the intermediate embedding (IE).

At the current layer N , the attention mechanism takes as input the current embedding \mathbb{E}^N and produces as output the intermediate embedding IE^{N+1} .

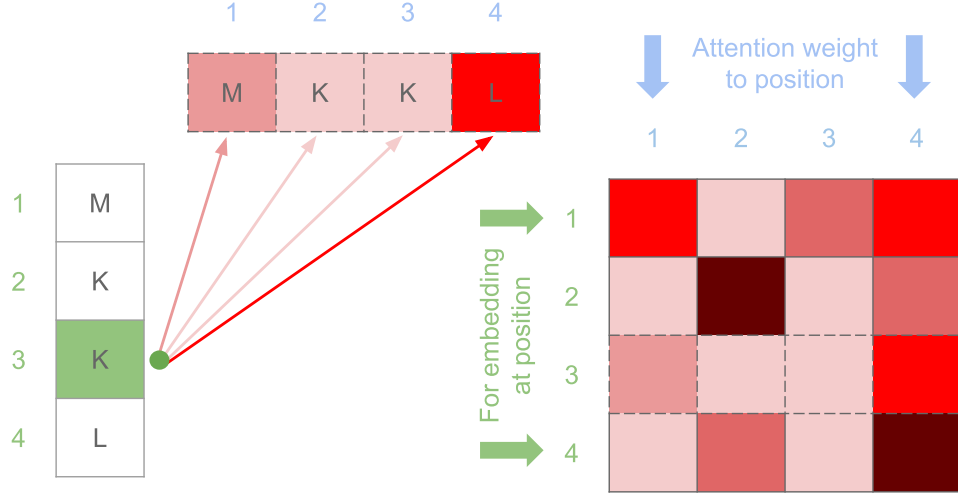


Figure 1.15 – Given a sequence of four residues, the Lysine (K) residue at position 3 gives the highest attention to the Leucine (L) residue at position 4 (left). This attention row can be found on line 3 of the attention map on the right.

(1.16)

$$IE = \text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \quad (1.17)$$

$$\text{head}_i = \text{Attention}(\mathbf{W}^q x, \mathbf{W}^k x, \mathbf{W}^v x) \quad (1.18)$$

1.2.3.5.2 The Feed-forward block

After the self-attention layer, the intermediate representation (IE) is transformed independently for each token. The embedding is projected to a higher dimension (generally of dimension $4 \times E$), then a non-linearity is applied and it is projected back to the original dimension E (See equation 1.19 and Figure 1.17).

With an embedding size of $E \in \mathbb{N}$ and an intermediate size of $4E$. $\mathbf{x} \in \mathbb{R}^{E \times 1}$, $\mathbf{W1} \in \mathbb{R}^{E \times 4E}$, $\mathbf{b} \in \mathbb{R}^{4E}$, $\mathbf{W2} \in \mathbb{R}^{4E \times E}$, $\mathbf{b} \in \mathbb{R}^E$ and a $ReLU(1.3)$ activation function, but this can be $GeLU$ (1.4) or others.

$$\text{Transformer_feed_forward}(\mathbf{x}) = \mathbf{W2}^\top(\text{ReLU}(\mathbf{W1}^\top \mathbf{x} + \mathbf{b1})) + \mathbf{b2} \quad (1.19)$$

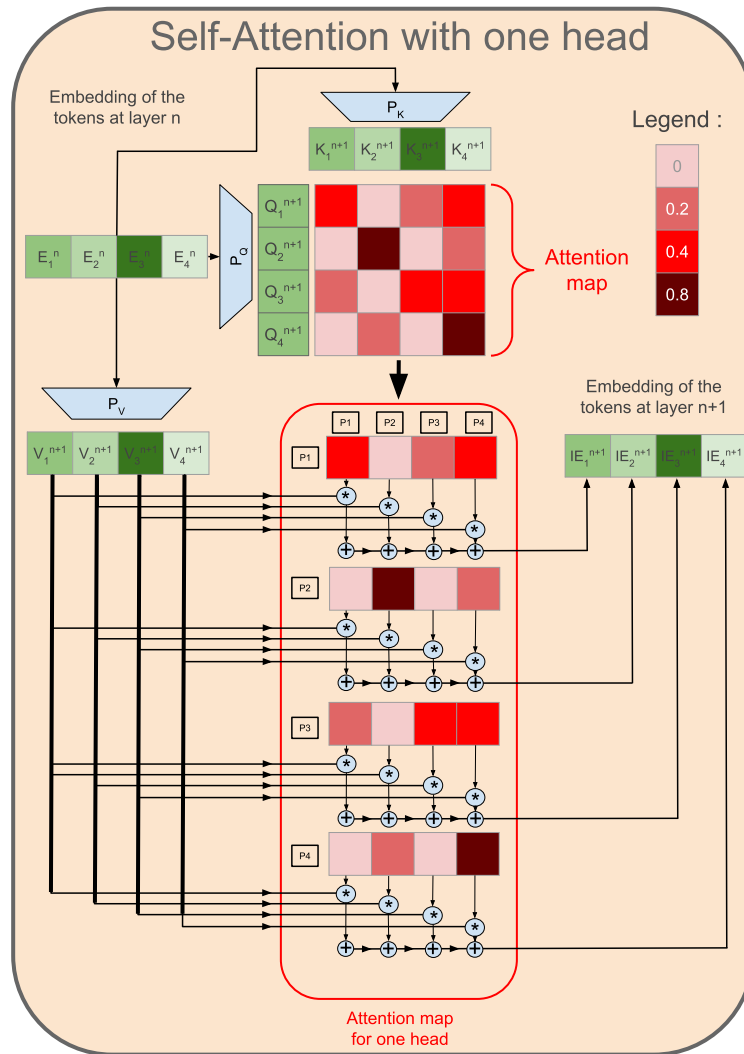


Figure 1.16 – Detailed view of the self-attention mechanism with one head. The blue trapeze represents linear projection. Residue positions are framed.

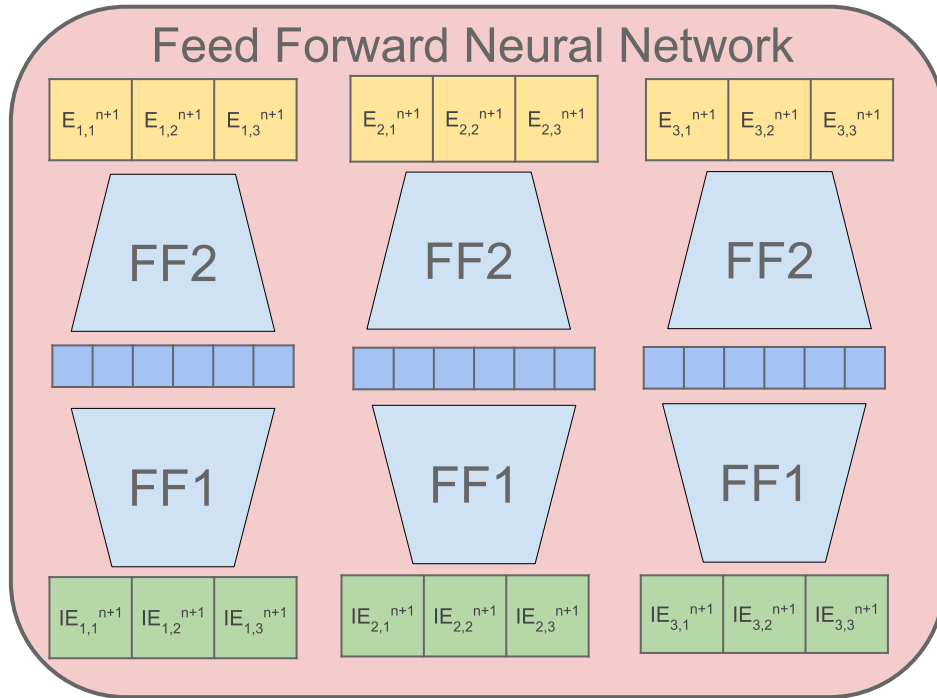


Figure 1.17 – The feed-forward layer present in transformer network. The blue trapeze represents linear projection. The non-linearity after FF1 is not represented here.

1.2.3.5.3 Global architecture

In addition to the attention layer and the feed-forward layer, there is a normalization layer and skip connection in the Transformer architecture (See Figure 1.18). The layer normalization[BKH16] has been proven to accelerate the training. The skip connection allows us to train deeper networks in avoiding the vanishing gradient problem. To the best of our knowledge, this was first used in this context (Deeper network) in ResNet[He+15].

1.2.3.5.4 One missing information: Position encoding

As can be seen from all the different layers in the network, the transformer network is position invariant. Thus, the information can be added in different ways. The first method is absolute position encoding. It consists in adding a vector to the token encoding at the start of the network. The classical way to create this vector is to take different sinusoidal

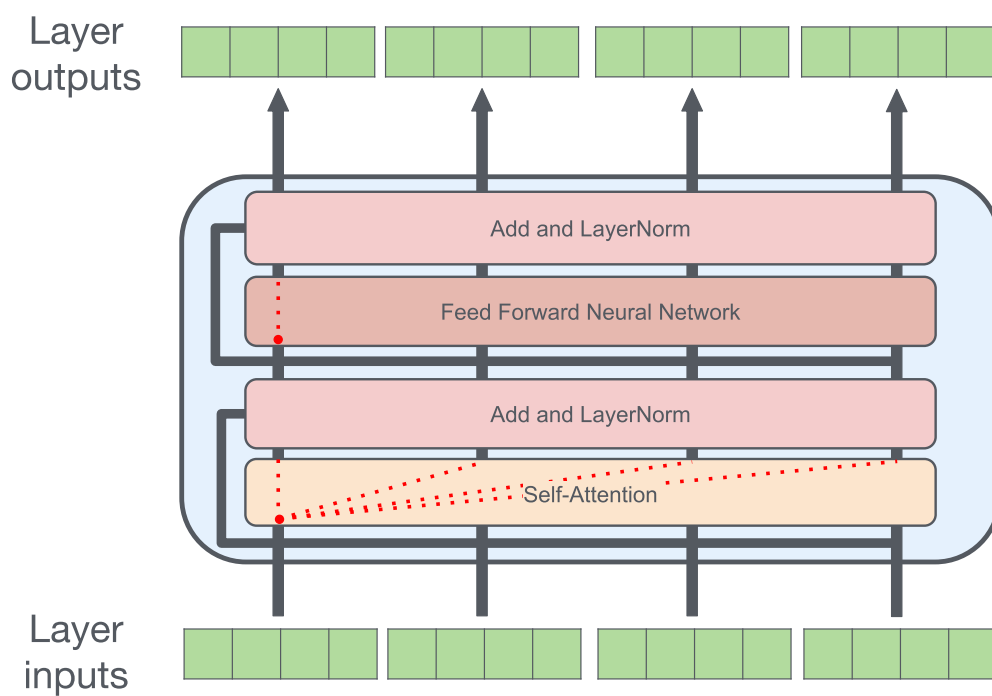


Figure 1.18 – Overview of one complete Transformer encoder layer. The red dotted line represents how the information can flow for the first embedding. Everywhere in the self-attention and only between itself in the feed-forward layer.

functions([Dev+19]), one for each dimension in the embedding with decreasing frequencies along the dimension. Another way is to use a learned positional embedding.

Sometimes the relative position is the most important information. For example, in protein, if we encode two proteins with the same domain inside, but not at the same place, we want the attention to be the same inside the domain, and this is not guaranteed with absolute position encoding. One way to solve this is relative position encoding. Multiple mechanisms exist to do relative position encoding. The simplest version is the one proposed by [Raf+20]. It consists in adding learned scalars at each position in the attention maps, addressed by $i-j$, with i the row number and j the column number. If the distance between res i and res j is greater than a threshold (often 128), only one scalar is learned for all distances greater than this threshold. More information can be found in [KHL21].

Key Takeaways

Transformer is a type of neural network architecture that is best suited to capture long-range relationships than other traditional neural network architectures like RNNs and CNNs. The central part of the Transformer architecture is the attention mechanism that allows the model to focus on relevant parts of the input data. It calculates the importance or weight of different residues/tokens in the input based on their relevance to the current context. By using attention, the model can selectively attend to and incorporate information from different parts of the input, improving its ability to capture meaningful relationships.

In the following, we will explore the specific information that these embeddings should encapsulate and the essential features they need to encode.

1.2.3.6 Training objectives

Pre-training involves training a deep neural network (Transformer in our case) on a large dataset of protein sequences before fine-tuning it on specific protein-related tasks,

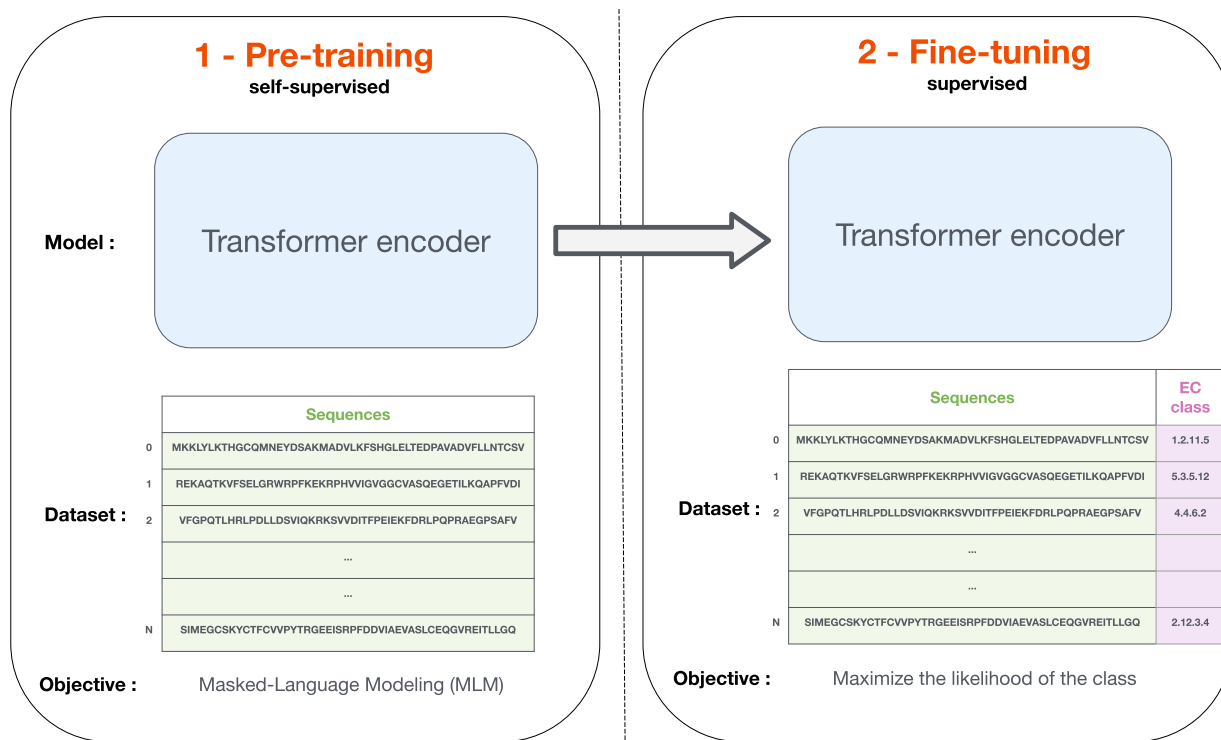


Figure 1.19 – Two phase training for deep neural network (Transformer in this Figure).

such as protein folding, protein-protein interaction prediction, and protein function prediction (Illustrated in Figure 1.19). One of the key advantages of pre-training in protein language models is the ability to learn meaningful representations of proteins by capturing their structural, functional, and evolutionary characteristics. Moreover, some protein functions have very few representatives. And pre-training mitigates overfitting. Pre-training only uses protein sequences that are abundant without the need for a label.

1.2.3.6.1 Pre-training

Multiple pre-training tasks exist, but one that significantly revolutionized token embedding is word2vec[Mik+13]. This model introduced highly efficient algorithms for training embeddings using massive text corpora. Two approaches, namely CBOW and Skip-

gram, were proposed to model word embeddings in natural language processing. These approaches are part of the known as word2vec, which aims to capture word representations in a distributed vector space. In CBOW, the model predicts a target word based on its surrounding context words. The context words are used to provide contextual information for predicting the target word. On the other hand, Skip-gram operates in the opposite direction by predicting the context words given a target word. It tries to optimize the model to correctly identify the context words that are most likely to appear in proximity to the target word. Both CBOW and Skip-gram have proven to be effective methods for learning high-quality word embeddings, with applications in various NLP tasks[Mik+13]. Another model with a similar objective to word2vec is GloVe ([PSM14]), which takes a different approach by considering global context matrices instead of relying solely on the local context within a specific document. GloVe computes these global context matrices and then aims to factorize them to obtain word embeddings. However, these embeddings are static and do not take into account the context or position of the token. As a result, these models struggle to differentiate between homonyms, such as the word "bank" in the contexts of "river bank" and "going to the bank.". In contrast to natural language processing (NLP), where the vocabulary is often extensive (consisting for example of 30,000 tokens in [Dev+19]) and composed of word chunks, the protein domain presents a different scenario. In protein analysis, the token represents a residue, resulting in a considerably smaller vocabulary (approximately 20 tokens). Additionally, the position of a residue within a protein carries significant importance, making it challenging to have useful embeddings if all embeddings for a given amino acid are identical, regardless of their position. To address these challenges, there is a need for context-dependent embeddings in the protein domain. This requirement extends to both local and global contexts. Local context refers to patterns specific to a residue's immediate surroundings, such as secondary structure. On the other hand, global context involves long-range contacts that play a vital role in defining the three-dimensional structure of the protein. To obtain this contextualized embedding two major approach exists. The first one is autoregressive modeling, where the neural network tries to predict the next word based on all the previous ones, as in Generative Pre-Training (GPT)[Rad+]. However, this method is limited because it can only model context from left to right. Some models try to do bidirectional

prediction, such as Bidirectional Recurrent Neural Networks[SP97], ELMO[Pet+18] and ULMFIT[HR18]. But they still cannot consider the context from both directions simultaneously. The final embedding is the concatenation of the two contexts. Unlike the previous methods that concatenate the left and right contexts, autoencoding modeling can capture the context from both directions simultaneously by using masked language modeling (MLM) as the main training task. This approach was popularized by the Bidirectional Encoder Representations from Transformers (BERT) [Dev+19]. The bi-directional context has been shown to be important for protein modeling in [Eln+21a](Section 4.4 last paragraph). The following paragraph will explain in detail how masked language modeling (MLM) works.

1.2.3.6.2 Masked Language Modeling (MLM)

Masked language modeling (MLM) is a task that involves creating a modified version of an input sequence by randomly masking or corrupting a predetermined percentage of its tokens. The goal of MLM is to train a neural network to predict the masked or corrupted tokens, effectively denoising the input sequence to retrieve the original sequence.

By doing so, the model will improve its final embedding to have relevant biological information easily accessible.

The masked language modeling technique, as described in [Dev+19], involves randomly selecting 15% of the tokens from the input sequence. Among these selected tokens, 80% are replaced with a special mask token, 10% are substituted with random values, and the remaining 10% are left unchanged. The model then utilizes the embeddings of the masked tokens to predict their original values, thus denoising the sequence and restoring it to its original state. The denoising process is visualized in Figure 1.20. It should be noted that these specific proportions have been widely adopted in many models that use the masked language modeling technique, but there may exist other variations that could be more effective for different tasks or datasets.

The T5 study, conducted by [Raf+20], investigated the effect of various corruption rates on performance in natural language processing (NLP) tasks. In addition to the standard 15% corruption rate, they experimented with rates of 10%, 25%, and 50%. In-

terestingly, they observed no significant performance difference between the lower rates (10%, 15%, and 25%), suggesting that these rates produce comparable results. However, when the corruption rate was increased to 50%, they noted a significant drop in performance. Furthermore, the researchers found that replacing spans of tokens instead of individual tokens gave a slight advantage, especially for spans that were less than 10 tokens in length.

Masked language modeling (MLM) is a self-supervised task that requires only sequences as input. However, generating meaningful representations of protein residues through MLM typically demands significant computing power. It is hypothesized that the representations generated by MLM are derived from the model's ability to capture patterns and features within protein sequences that may be indicative of their structural characteristics. These patterns may include identifying specific amino acid motifs or sequence patterns that correspond to secondary structure elements like alpha-helices or beta-sheets. Additionally, the model may be able to capture evolutionary information by identifying regions in protein sequences that are conserved across evolution, which may correspond to critical structural elements for protein function. However, further research is needed to confirm these hypotheses and to fully understand the effectiveness of MLM in capturing structural and evolutionary information in protein-related tasks. Following pre-training, the next step is fine-tuning, which involves adapting the pre-trained model for a specific task.

1.2.3.6.3 Fine-tuning

One common method for fine-tuning a pre-trained protein language model for classification tasks involves adding a special [CLS] token to the beginning of the protein sequence and using its embedding as a representation of the entire sequence. However, other approaches also exist, such as taking the mean embedding of all the residues in the sequence or using other aggregation methods to generate a condensed representation of the sequence. Finally, this embedding is fed into a feedforward classification layer, which makes predictions about the class of the protein. The weights of the classification layer are initialized randomly, and then the model is fine-tuned on a labeled dataset through backpropagation, adjusting the weights to minimize the classification loss. This fine-tuning process can be repeated

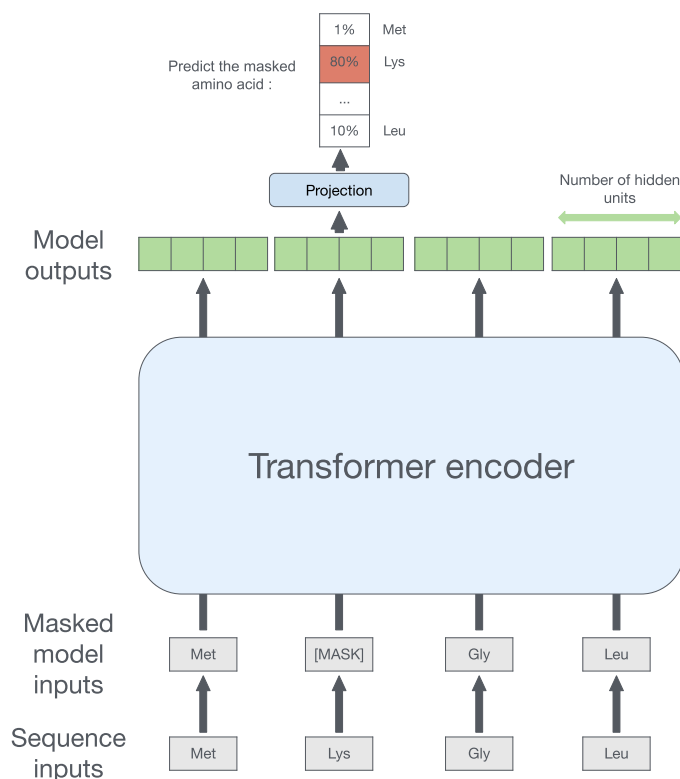


Figure 1.20 – An example of pre-training for one protein. The first step is to mask/corrupt some tokens. The second step is to run the corrupted sequence through the network. And lastly, use the embedding to predict the original token.

for multiple classification tasks, each with its own classification layer. The pre-trained weights of the model can be frozen or fine-tuned for each task. The goal of fine-tuning is to adapt the pre-trained model to specific protein classification tasks, leveraging the model’s ability to capture complex patterns and features in protein sequences. An illustration of the fine-tuning process is provided in Figure 1.21.

The challenge of developing effective Automatic Function Prediction (AFP) tools for protein-related tasks has been addressed using a myriad of techniques. By leveraging pre-training, models can extract general patterns and features from vast, unlabeled datasets, while fine-tuning enables adaptation of these pre-trained models to specific classification tasks.

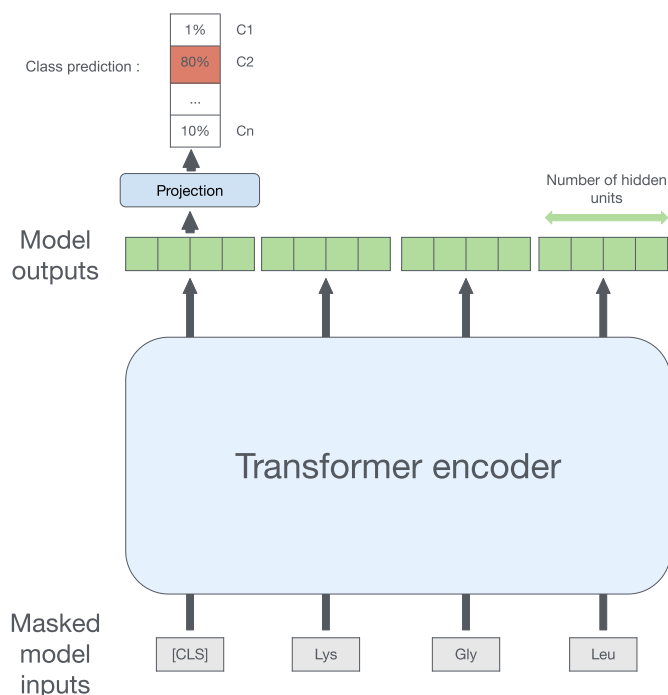


Figure 1.21 – An example of fine-tuning on a classification task at the protein level. First, a special token named [CLS] is prepended to the sequence. Then the sequence is run through the model. And finally, the embedding of the [CLS] is used to predict the class of the protein.

The success of pre-training and fine-tuning is contingent upon several factors, such as the availability of a substantial quantity of unlabeled sequence data, the selection of appropriate pre-training and fine-tuning methodologies, and the configuration of model architecture and hyperparameters. With natural language processing as inspiration this has given rise to advanced pre-trained protein language models [Eln+21a], which have demonstrated promising outcomes in a range of protein-related tasks, including secondary structure and contact prediction.

Whilst these models have not been directly applied to AFP tasks at the beginning of this thesis. The subsequent chapter will investigate the potential of these strategies for addressing the challenges inherent to AFP, shedding light on their efficacy and areas for further exploration.

PREDICTING ENZYMATIC FUNCTION OF PROTEIN SEQUENCES WITH ATTENTION

2.1 Introduction

The first objective of this thesis is to examine the application of attention-based models, such as the Transformer, in the Automatic Function Prediction (AFP) domain, with a specific emphasis on predicting enzyme function. Enzyme function prediction serves as a controlled environment for studying the effectiveness of attention mechanisms in AFP tasks. The functional prediction of enzymes, which account for around half of all known proteins [The21], is an important application of functional annotation of protein sequences. This task is a testbed for the development of predictive methods for several reasons. First, the enzyme's functions are well-defined according to experimental evidence. These functions are then standardized by Enzyme Commission (EC) numbers which provide well-defined targets to train machine learning models [KL94]. Secondly, the amount of annotated enzymatic sequences available is sufficient to enable training and independent large-scale evaluations.

In addition to functional annotation, we also chose to evaluate the benefits of the Transformers' attention mechanism as an interpretability method. That is a way to better understand the link between the input (a protein sequence in our case) and the output (a functional prediction) found by the model. In a word, for a given residue, attention provides an importance score showing which other residues are most related to it to perform a given prediction task. As such, attention naturally highlights key relations within the sequence [CGW21]. Previous work [Vig+20] has related protein attention of

Transformers (TAPE - [Rao+19], ProTrans - [Eln+21a]) and the presence of binding sites. Yet, while these studies show the potential of attention-based methods, whether they can be used and be better than classical interpretability methods in the context of biological sequences is still an open question. Tackling this issue requires developing a common setup to allow for an exhaustive comparison between methods: this thesis presents such a setup in the context of enzymes' functional annotation.

To summarize, this section is about : (i) We show that using Transformer neural networks enables us to achieve state-of-the-art results for predicting the enzymatic function of a protein from its sequence only; (ii) We present a simple attention-based interpretability method that outperforms classical generic ones in terms of coherence with prior biological knowledge.

2.2 Methods

2.2.1 Enzyme class prediction task

2.2.1.1 Task description and datasets

Dataset	Not an enzyme	Oxidoreductases	Transferases	Hydrolases	Lyases	Isomerases	Ligases	Translocase	Total
EC40 training	0	14 574	34 031	25 329	8 792	5 399	11 861	0	99 986
EC40 validation	0	365	873	746	182	115	225	0	2 506
EC40 testing	0	376	887	677	147	156	263	0	2 506
ECPred40 training	22 797	21 380	62 124	33 362	16 926	10 926	24 045	0	191 560
ECPred40 validation	5 502	3 014	8 097	3 947	2 008	1 366	2 536	0	26 470
ECPred40 testing	499	17	201	162	19	20	6	0	924
SwissProt_2021_04	302 753	26 278	80 485	40 913	23 253	14 453	26 374	8 455	522 964

Table 2.1 – Number proteins sequences in each Enzyme Commission (EC) first level class in each dataset. The dataset named EC40 is from [Str+20] and the dataset named ECPred40 is derived from [Dal+18]. SwissProt_2021_04 is directly from UniprotKB/Swissprot without filtering.

The task of interest in this thesis consists in predicting Enzyme Commission (EC) numbers at a specific level. This nomenclature is composed of 4 levels. The first level

provides the main class of the enzyme, which is encoded by a number between 1 and 7. The second and third digits correspond respectively to the subclass and sub-subclass of the enzyme, and the last one represents specific metabolites and co-factors involved, which basically provides the actual reaction catalyzed or a restricted set of very similar reactions. Therefore, an enzyme characterized to the fourth level is considered to be completely characterized.

Two datasets, EC40 and ECPred40, were studied in this thesis, in order to have the most direct comparison with state-of-the-art models. The first dataset, EC40 from [Str+20], goes up to EC level two. It ensures that i) sequences from the testing set share less than 40% sequence identity with any sequence used for training the models and ii) sequences in the testing set share less than 40% sequence identity between themselves.

We developed ourselves the second dataset, ECPred40, which goes up to EC level four. It is inspired by Dalkiran et al. [Dal+18] who trained their model on Swiss-Prot release 2017_3 and tested it on enzymes annotated after this release. Yet, the authors did not consider the degree of homology in sequences between training and testing datasets. Therefore, we took inspiration from [Str+20] to rebuild the testing set according to their strict procedure. More specifically, we first retrieved the newly created annotations between the release Swiss-Prot (release: 2017_3) and the release Swiss-Prot (release: 2021_04). Then, we clustered these novel annotations and the training set with MMseqs2 ([SS17]) at 40% identity threshold: if two sequences shared 40% or more identity, they were assigned to the same cluster. To build the testing set, all clusters with at least one sequence from the training set were discarded and a representative of each remaining cluster was included in the testing set, ensuring less than 40% sequence identities between the sequences from the testing set and the other sequences used (training/validation). Concerning the training and validation sets, [Dal+18] applied the same process, keeping only one representative sequence per cluster. By contrast, we choose to keep all the sequences for training, as did [Str+20].

We also labeled as "non-enzymes" all sequences having neither GO annotation with catalytic activities, nor any EC number annotation. This is required to perform the prediction of whether a protein is an enzyme or not, referred to hereafter as "level 0" prediction. For a fair comparison, we only kept EC class that ECPred was capable of predicting

(628 EC classes at level 4). We also filtered on size because ECPred cannot predict on sequences less than 40 residues long. Only non-fragment sequences were kept. Finally, we set a max size of 1024 residues because of memory constraints. We named this custom dataset ECPred40.

As a final remark, one can notice in table 2.1 that the translocase class is absent. This is due to the fact that models we compare our Transformer to only consider Swiss-Prot releases before 2018, which is when translocase class was first introduced.

2.2.1.2 Evaluation procedure and metrics for prediction performance

For the comparison with UDSMProt [Str+20], we used their EC40 dataset and computed the accuracy at level 2 as they did. We also computed the macro-f1 score for our model to have a basis for comparison that is not biased toward more common classes. Macro means that the average is computed on each enzyme class of the different metrics, which allows sampling biases to be mitigated if some enzyme classes are more represented than others. For the comparison with ECPred [Dal+18], we also needed to evaluate the enzyme / non-enzyme discrimination. We have done two evaluations based on the new EC40Pred testing set. The first one focused on the enzyme vs non-enzyme classification task. For the second, we only considered the enzymes in the testing set and evaluated the predictions at levels 1 to 4. Macro measures (precision, recall, f1) were used.

2.2.2 The model

2.2.2.1 EnzBert

Our model, named EnzBert, is based on a Transformer architecture. To avoid some unnecessary computation, we started from an already pre-trained Transformer, the ProtBert-BFD model variant from [Eln+21a].

Then, we fine-tuned this model for the EC class prediction task, to obtain EnzBert. In order to do this, we classically appended a special [CLS] token to all input token sequences. The embedding of the [CLS] token is meant to represent the whole sequence, from which we can derive the functional annotation (as in [Dev+19]). We did not freeze the first layers when performing the fine-tuning part, by contrast with [Eln+21a]. We

Parameter	EnzBert _{EC40-ECPred40-SwissProt}
Dropout on [CLS]	0.2
Batch Size	2
Accumulation step	16
Learning rate	1.0×10^{-5}
Optimizer	Adam($\beta_1=0.9$, $\beta_2=0.999$)
Lr scheduler	Lr(epoch)=0.8*Lr(epoch-1)
Number epochs	5-15-15

Table 2.2 – Hyper-parameters used for the fine-tuning of the three different versions of EnzBert. When different a "-" is used.

used a normalization layer to speed up the training process, followed by a linear layer to project the [CLS] final embedding to the desired number of classes. For fine-tuning, we did not consider the EC hierarchy : all classes were treated as being independent, allowing us to use a cross-entropy loss function.

The code used for this thesis is written in PyTorch and is available at <https://gitlab.inria.fr/nbuton/tfpc>, the hyper-parameters are shown in table 2.2 .

Two models were trained: EnzBert_{EC40} and EnzBert_{ECPred40} fine-tuned respectively on EC40 and ECPred40. We also trained EnzBert_{SwissProt} on all SwissProt (release dump 2021_04) in order to compare interpretability methods.

Due to the presence of the over-represented "non-enzyme" class (more example of "non-enzymes" than examples of other level 4 EC) and EC classes with very few examples, we chose to balance the dataset during training with a weighted random sampler (WeightedRandomSampler class from PyTorch) to train EnzBert_{SwissProt}. The chosen weights were the inverse of the occurrence for each class.

2.2.3 Interpretability

In order to compare our custom attention-based interpretability method, we first re-implemented classical methods on the same common setup for proper comparison.

2.2.3.1 Descriptions of classical interpretability methods

As shown in figure 2.1, we considered methods that are model-specific (e.g. requires the model to be differentiable) or model-agnostic, such as LIME ([RSG16]). We also included class-specific (e.g. TGradCam see [CGW21]) and class-agnostic methods (e.g. Attention last layer, named Raw attention in [AZ20]). Note that, while being class-specific in theory, generic Gradient-based methods tend to behave like class agnostic methods ([CGW21]), meaning that the feature importance does not change much between classes. Providing an exhaustive review of gradient-based interpretability methods is beyond the scope, as many variants exist (e.g. Gradient time input - [Shr+17], Integrated Gradient - [STY17], ...).

Finally, some methods rely on the attention maps which are specific to Transformers (e.g. Attention Last Layer and Rollout, see [AZ20])

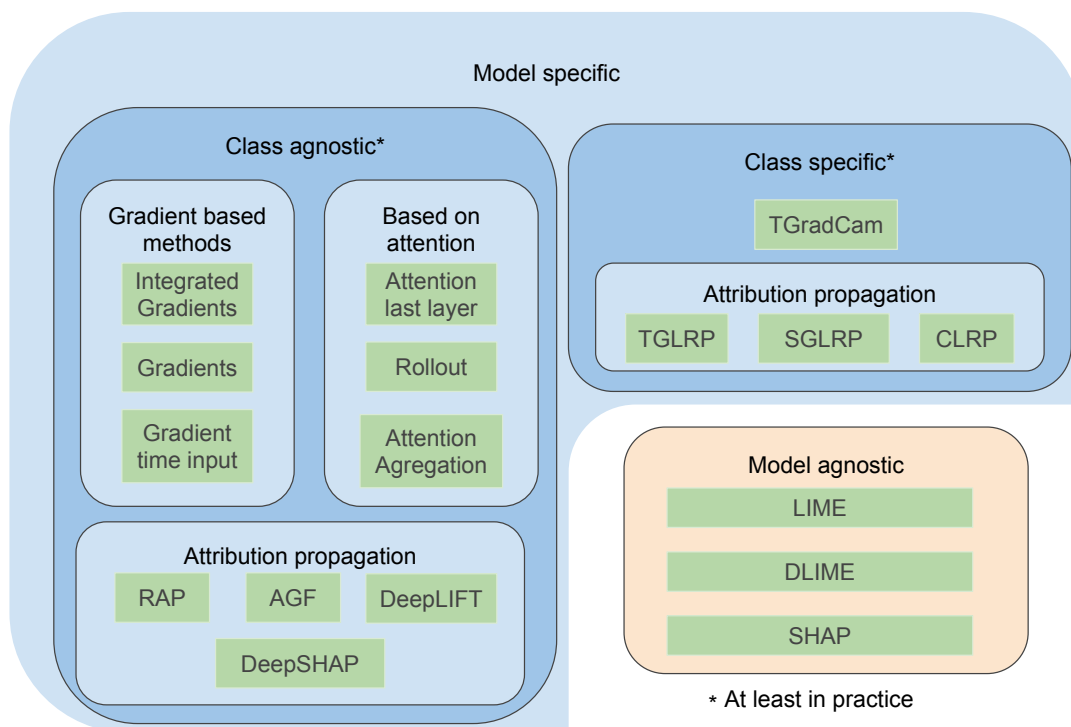


Figure 2.1 – Summary of the main types of interpretability methods

2.2.3.2 Attention aggregation interpretability methods

All attention maps in the Transformer can be stored in a tensor of shape $(L \times H, N, N)$, with L the number of layers in the Transformer, H the number of head per layer and N the length of the sequences. The first dimension is the dimension of all heads for all layers, the second dimension is the attention received by a given residue from the others and the last dimension is the attention given by a residue to all the others.

In order to obtain a vector of size N containing the "aggregated attention" for each residue, we could collapse the T tensor along the second dimension (to obtain a matrix of shape $(L \times H, N)$) and then again along the first dimension of this matrix, to obtain a vector of size N (as illustrated in figure 2.2), as well as other orders of aggregation, e.g. starting the aggregation on the first dimension of the tensor. In terms of aggregation, we explored two possibilities: the average and the maximum. Overall, combining along the various dimensions and choosing an aggregation method provides 13 different possibilities, after removing duplicates (16 before). We will name these variants AttnAgg followed by the first and second dimensions collapsed, each annotated by the function used for pooling (A: Average and M: Max), e.g. AttnAgg1A1A as shown in figure 2.2.

2.2.3.3 Catalytic residues as gold label

One key question is whether the residue importance for the classification task can be related to known features of the amino acids in the sequence. To explore this aspect, we used the Mechanism and Catalytic Site Atlas (M-CSA) ([Rib+18]) database. It documents numerous known enzyme catalytic residues and reaction mechanisms. All information in this database has been manually curated and is supported by research papers. It describes 992 enzymes, with an average length of 439 amino acids and on average 4.9 catalytic residue's annotation per enzyme.

These catalytic residues will be used as a proxy to evaluate the different interpretability methods and will be referred to as "gold labels" in the following sections.

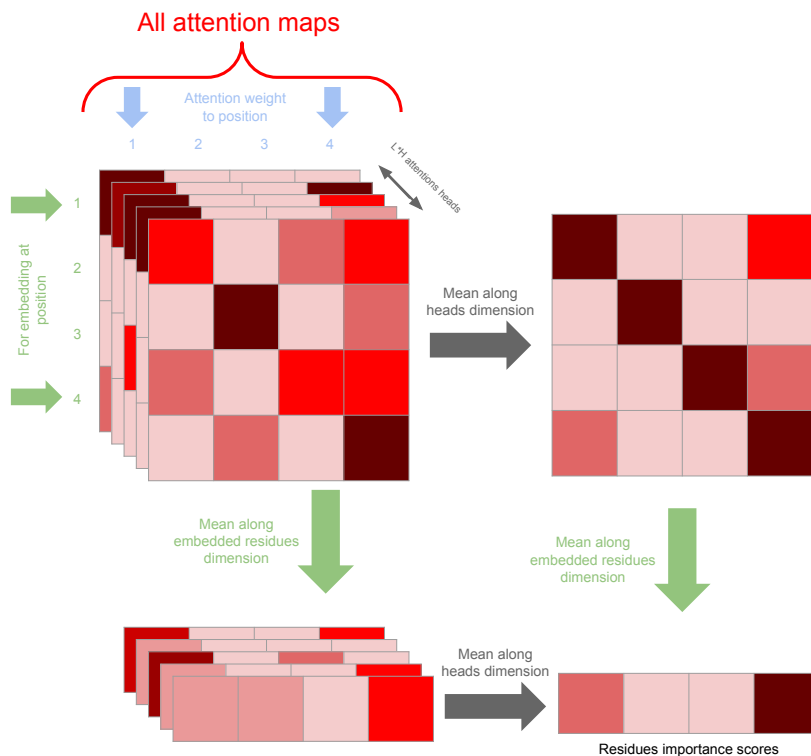


Figure 2.2 – Aggregation methods AttnAgg1A1A and AttnAgg2A1A. AttnAgg2A1A averages all attention maps and then average over the columns to obtain a row vector.

2.2.3.4 Evaluation and metrics for interpretability

There is no clear consensus about which metric to use to evaluate interpretability with gold labels [DeY+20]. In this thesis, we use mainly two types of curves, from which we derived comparison metrics. First, we took inspiration from [CGW21]. For a given interpretability method, for each enzyme, we selected the residues with the top-k importance scores given by the interpretability method. We then crossed this features' importance list with the gold labels: each important residue with a gold label is considered a true positive. This allowed us to compute an f1 score for a given sequence. We then averaged the f1 scores over all sequences and repeated this process from k=1 to 50 top-k scores. Second, from the same importance list, we also derived a precision-recall curve. Such a curve is often used in the context of unbalanced classes, which is the case here as catalytic

residues only represent 1.16% of all residues.

We then derived two metrics to quantitatively compare models. First, we considered the Precision-Recall Gain Area Under the Curve (PRG-AUC - [FK15]). Second, we computed a maximum F-Gain score as follows. First, for each sequence, we deleted the score on the CLS token and rescaled the tokens' importance score (either min-max scaling, normalization, division by L1 or L2 norms, or no scaling). Then, we created a set with all the (rescaled) scores of all the tokens from the testing set sequences. All tokens whose score is higher than a given threshold are considered "important". An F1 score is then computed, crossing important tokens and known catalytic annotations. The maximum F1 score consists in keeping the highest F1 score achieved by varying the threshold. Finally, the F-Gain, precision gain and recall gain scores are derived from these F1, precision and recall scores accounting for the performances of an always-positive classifier [FK15].

We also designed a 'baseline' method to check whether the distribution of scores was enough to provide high interpretability metrics. To do so, we shuffled the importance score of the tokens and applied the same metrics (PRG-AUC and maximum F-Gain score) to compare to our (non-shuffled) results.

Computation time for each interpretability method was measured on a CPU as some interpretability methods (e.g. TGLRP) need more RAM than available in the GPUs at our disposal. All the models and the interpretability annotation are available on gitlab <https://gitlab.inria.fr/nbuton/tfpc>.

2.3 Results

2.3.1 Enzyme class prediction

Table 2.3 summarizes the class prediction quality of EnzBert_{EC40} on EC40 test set, as well as UDSMProt [Str+20], the best-known enzyme predictor at level 2 using only sequences. On this dataset with less than 40% identity between training and testing sequences, the results show : i) an increase in accuracy from 87% for the previous state-of-the-art model, UDSMProt, to 97% for EnzBert_{EC40} concerning the prediction of the 6 possible classes at level 1 and ii) an increase from 84% to 95% accuracy for the prediction

of the 51 possible classes at level 2. Macro-f1, which allows all classes to be considered equally and avoids skewing the evaluation towards the most common classes, reaches here respectively 96% and 89% at levels 1 and 2 for $\text{EnzBert}_{\text{EC40}}$.

When it comes to finer levels of predictions, the LSTM-based network from UDSMProt cannot be used as it is trained to predict up to level 2 classes only. The current state-of-the-art for level 4 predictions is ECPred [Dal+18]. Table 2.4 shows the prediction performances of $\text{EnzBert}_{\text{ECPred40}}$ and ECPred at the different EC levels, from level 0 (enzyme vs non-enzyme discrimination task) to level 4 (628 classes), on ECPred40 test set. $\text{EnzBert}_{\text{ECPred40}}$ significantly improves predictions over ECPred at each level, except at level 1 where ECPred favours recall and our model favours precision. At level 4, the finer and more challenging level of prediction, our model improves both macro-precision and macro-recall with respect to ECPred, resulting in an increase of the macro-f1 score from 40.7% to 55.2%.

Model	Level	Macro-f1	Macro-precision	Macro-recall	Accuracy	Number of classes
UDSMProt	1	-	-	-	0.87	6
EnzBert_{EC40}	1	0.96	0.96	0.96	0.97	6
UDSMProt	2	-	-	-	0.84	51
EnzBert_{EC40}	2	0.89	0.93	0.87	0.95	51

Table 2.3 – Comparison with UDSMProt of the prediction quality at the two levels of EC40 test set.

In our approach, we focused on state-of-the-art models UDSMProt and ECPred. Other tools exist like EzyPred [SC07], EFICAz [KS12] and DEEPred [Li+18], but the experimentation by [Dal+18] suggests that ECPred outperforms them. Moreover, [Str+20] presented experiments showing that UDSMprot outperforms DEEPred at level 2 and is on par with ECPred at level 1 (we do not have information on level 2 comparison). These results suggest that Transformers outperforms state-of-the-art tools for the prediction of the enzymatic class of proteins from their sequence. In the next section, we focus on the interpretability offered by their attention mechanism.

Model	Level	Macro-f1	Macro-precision	Macro-recall	Accuracy	Number of classes
ECPred	0	0.769	0.784	0.781	0.769	2
EnzBert_{ECPred40}	0	0.837	0.874	0.831	0.845	2
ECPred	1	0.728	0.691	0.841	0.824	6
EnzBert _{ECPred40}	1	0.604	0.784	0.582	0.813	6
ECPred	2	0.492	0.468	0.579	0.759	51
EnzBert_{ECPred40}	2	0.629	0.676	0.672	0.781	51
ECPred	3	0.496	0.491	0.549	0.727	132
EnzBert_{ECPred40}	3	0.609	0.625	0.652	0.749	132
ECPred	4	0.407	0.431	0.412	0.636	628
EnzBert_{ECPred40}	4	0.552	0.576	0.562	0.687	628

Table 2.4 – Comparison with ECPred of the prediction quality at the five levels of ECPred40 test set.

2.3.2 Interpretability

The Transformers’ attention provides a built-in interpretation mechanism, but its complexity in the case of multiple attention heads makes it difficult to use directly. We have thus proposed and tested different attention aggregation methods. Among 13 different possibilities for attention aggregation described in section 2.2.3.2, the best performing was ‘AttnAgg2A1A’ (which is mathematically equivalent to ‘AttnAgg1A1A’), described in figure 2.2. It consists in i) taking the average over all attention maps and then ii) averaging over the vertical dimension of this average attention map. Exploring other aggregation orders shows that averaging over the attention a given residue is paying to all others leads to poor performance (all PRG-AUC < 90 %). By contrast, all methods focusing on the attention received by a given residue perform well (PRG-AUC > 90 %), with the exception of AttnAgg1M1M (PRG-AUC 66.25 %). Note that this method chains two max pooling operations instead of averaging, which might result in a loss of information. As shown in figure 2.3, top, the best attention aggregation method outperforms all other interpretability methods, for all levels of recall. In the low recall regime, ‘AttnAgg2A1A’ shows at least twice the precision of all other methods.

Figure 2.3 top, shows the f1 score with respect to the k most important residues identified by each interpretability method (see section 2.2.3). AttnAgg2A1A, outperforms all other interpretability methods for all numbers of residues between 1 and 50. As an illustration, the highest point in figure 2.3, bottom, for Attention aggregation corresponds to 2 residues: this means that testing whether the two most important residues are catalytic sites results in an f1 score of 35%. The same procedure with the two most important residues identified by LIME, for instance, results in an f1 score of 19%. Finally, in table 2.5, we reported the Precision-recall Gain Area Under the Curve (PRG-AUC), the max F-Gain metrics for each method. AttnAgg2A1A outperforms the other methods on both prediction metrics.

Considering the execution time, three groups appear and AttnAgg2A1A belongs to the fastest one. The first group contains methods nearly as quick as classical prediction, all being under 5 seconds to run: Gradient, Gradient time input, attention last layer and attention aggregation. A second group consists of methods around ten times slower, with TGLRP, GradCam, and integrated gradient. Finally, LIME method is about 5000 times slower than prediction times, which corresponds to the number of sequences needed to estimate the local estimator of LIME. For instance, evaluating the importance of residues in 1000 proteins of length 439, the average length in the M-CSA database, would take about 197 days with LIME and less than an hour with our approach on CPU.

2.3.2.1 Visualization of important residues

Figure 2.4 illustrates cases of best and worst agreement between the known annotation for a given residue to be a catalytic site and the corresponding importance score for the same residue computed with our best interpretability method, namely AttnAgg1A1A.

For the best case, seven proteins exhibit a PRG-AUC of one, which means that all their catalytic residues have the highest importance scores. Among those, we chose to present the nh(3)-dependent nad(+) synthetase enzyme in figure 2.4 since it has the highest number of catalytic sites (two) and the highest annotation quality of 5/5 in uniprot. The two highest importance scores correspond to the two catalytic residues: GLU162 and ASP50. The two next highest importance scores correspond to two residues ASP158, HIS159 which are near the first catalytic site. We then searched in Swiss-Prot (the manually reviewed

Method type	PRG-AUC(x100)	max F-Gain(%)	Time(s)
Random	42.54 ± 4.37	69.85 ± 1.04	–
Grad	75.01	81.27	4.64
Grad X input	63.62	78.66	7.74
Integrated grad	76.41	81.70	2.48×10^2
Attn last layer	87.80	85.62	2.87
Attn agg	98.02	96.05	3.72
Rollout	66.08	76.77	2.95
TGLRP	90.92	88.56	4.05×10^1
TGradCam	81.00	76.77	4.35×10^1
LIME	93.46	91.44	1.73×10^4

Table 2.5 – Evaluation of best interpretability method of each category with respect to the M-CSA dataset. Precision-recall Gain Area Under the Curve (PRG-AUC) and the max F-Gain metrics is reported, and we also report the mean execution time for each method (for one protein, in second on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz).

part of UniProtKB) for potential annotations in the other residues within the top 10 importance scores. We found that four were associated with binding sites: GLY48, GLY47, SER46 and THR157. Overall, catalytic site and binding site annotations were significantly enriched in the top 10 highest importance score (6/10 compared to 21/271 for the whole enzyme, p-value of 0.00003).

The worst agreement between importance and presence of catalytic sites was found for the Aldehyde dehydrogenase (FAD-independent) enzyme. The sole catalytic residue, GLUE869, is not highlighted by our interpretability method, that is, it does not belong to the top 5% residues in terms of importance scores (position 887 over 907 residues). In this case, residue importance scores seem to focus more on the binding sites of the protein. Indeed, 2 binding sites documented in Swissprot are found in the top 10 residues in terms of importance score: CYS103 and CYS45. In this case as well, we observe a significant enrichment of binding and catalytic sites within the top 10 highest importance scores (2/10 compared to 10/907 for the whole enzyme, p-value of 0.00516).

2.4 Discussion

Our experimentation shows that the use of attention has the potential to significantly outperform state-of-the-art approaches for the prediction of the enzymatic class based on sequences. This provides another example, outside the field of natural language processing, of the interest of Transformers over LSTM-based neural networks. This study also demonstrates the difficulty of estimating the performances of models. Indeed, although we took special care to evaluate the approaches on published and functional benchmarks — the EC40 benchmark from [Str+20] and the new benchmark ECPred40 up to level 4 inspired by the time-based evaluation by [Dal+18] — a significant difference in the expected performances can be observed at level 1 and 2 with respect to the dataset used, despite the fact that they are based on the same identity threshold of 40% (see tables 2.3 and 2.4). As an additional experiment (data not shown), we trained an EnzBert model to predict the level 2 classes (without the non-enzyme class) on ECPred40 with exactly the same procedure as for EC40. The gap in classification performance was still important. This highlights how critical the choice of the dataset is and that further studies will probably be needed to better estimate the actual predictive power of enzyme classification methods in practice.

Our experimentation also shows that the attention of Transformers provides a built-in interpretable mechanism pointing to important residues of enzymes, thanks to our simple AttnAgg2A1A aggregation of multi-head attentions. It is surprising that this simple linear aggregation retrieves enzymatic sites better than state-of-the-art attention-based interpretability methods. For instance, the best of these later methods, which has also been efficient in text interpretability [CGW21], is TGLRP. Based on attention maps and gradient computations, TGLRP can take into account non-linearities and might focus on more subtle signals. The characterization of its results and their comparison to the important non-catalytic sites found by our method remains an open issue. More generally, the study focused here on the safest, but limited, proxy for estimating the value of the different interpretability methods and it would be interesting to study other important residue features, as suggested by the analysis of figure 2.4.

2.5 Conclusion and perspective

We provide a state-of-the-art model *EnzBert* that only uses sequences to predict enzymes' functional annotation. This model benefits from the attention mechanism through the use of the Transformer architecture. We also propose a simple yet successful interpretability method that only relies on attention maps. The resulting insights on enzymes' sequences can help further research to better understand how enzyme classes are derived from the protein sequence and even help for further steps, for example regarding enzyme's optimization.

Finally, in our model, enzymatic classes are considered independent from one another. This means that we do not yet exploit to the fullest the underlying hierarchy structure of EC numbers. Integrating meaningful prior knowledge into deep neural network architectures remains challenging but we believe that the field of automatic functional annotation might particularly benefit from such approaches.

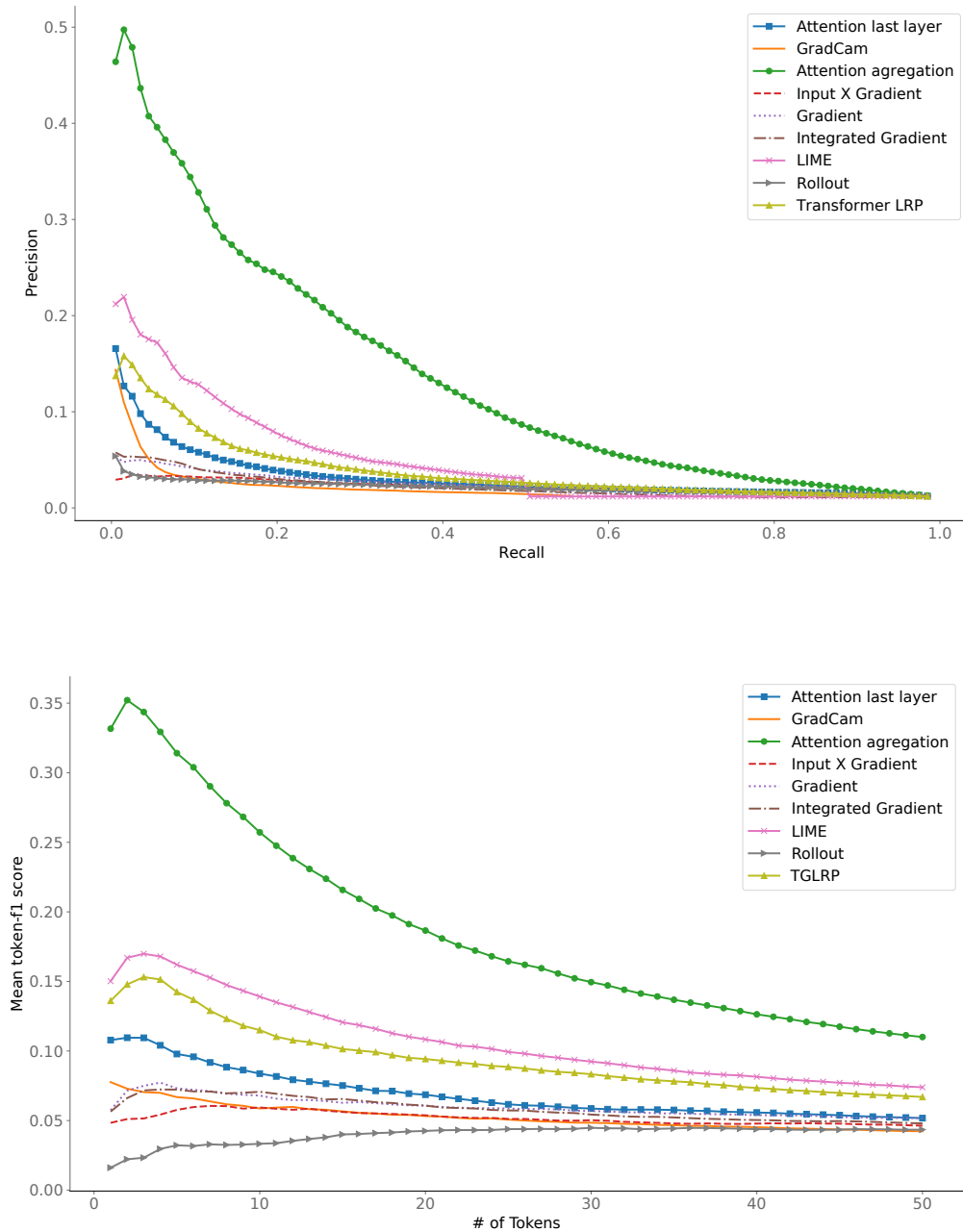
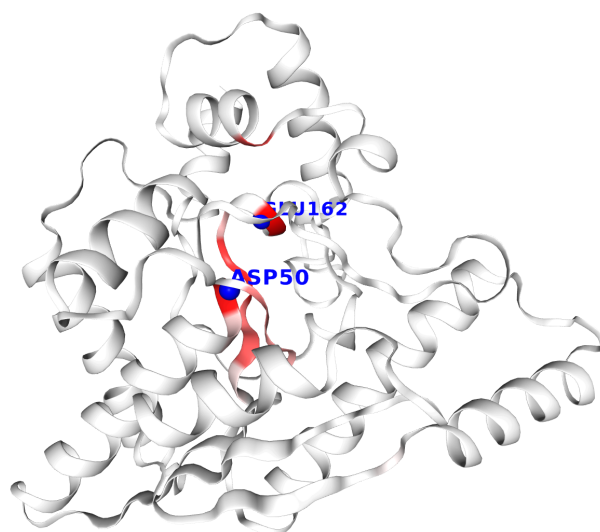


Figure 2.3 – Top: Precision / Recall curve for different interpretability methods. The attention aggregation group is represented by the AttnAgg1A1A method. Bottom: f1 score for top-*k* residue with the highest residue importance scores.

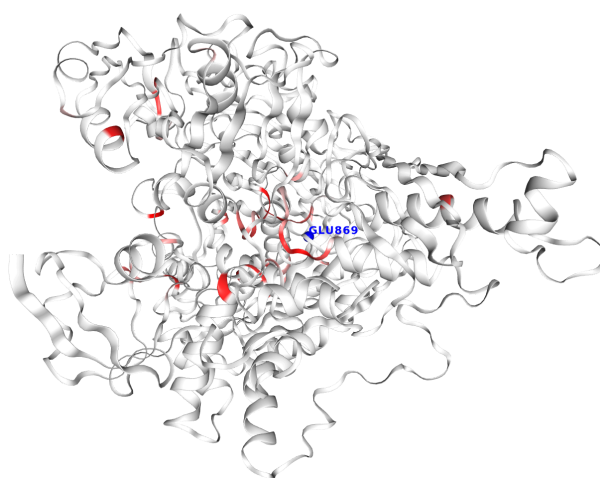


```

0 MSMQEKIMRE LHVKPSIDPK QEIEDRVNFL KQYVKKTGAK GFVLGI[REDACTED] [REDACTED]STLAGRLAQ LAVESIREEG GDAQFIAVRL PHGTQQDEDD AQLALKFIKP
1 DKSWKFDIKS TVSAFSDQYQ QETGDQLTDF NKGNVKARTR MIAQYAIIGG EGLLV[REDACTED] [REDACTED]AVTGFFT KYGDGGADLL PLTGLTKRQG RTLLKELGAP
2 ERLYLKEPTA DLLDEKPOQS DETELGIS[REDACTED] EIDDYLEGKE VSAKVSEALE KRYSMTEHKK QVPASMFDDW WK

```

(a) Nh(3)-dependent nad(+) synthetase



```

0 MIQKVITVNG IEQNLFVDAE ALLSDVLRQQ LGLTGKVKG [REDACTED]EQ[REDACTED]GAE[REDACTED]SV ILDGKVVRA VTKMKRVADG AQITTIIEGVG QPENLHPLQK AWWLHGGAAQ
1 GF[REDACTED]SPGFIVS AKGLLDTNAD PSREDVRDWF QKHRNACR[REDACTED]T GYKPLVDAMV DAAAVINGKK PETDLEFKMP ADGRIWGSKY PRPTAVAKVT GTLD[REDACTED]GADLG
2 LKMPAGTLHL AMVQAKV[REDACTED]A NIKGIDTSEA LTM[REDACTED]GVHSVI THKDVKGKNR ITGLITFPTN KGDGMDRPII CDEKVFQY[REDACTED]C[REDACTED]LVCADSE AM[REDACTED]RAAAEKV
3 KVDLEELPAY MSGPAAAED AIEIHPGTPN VYFEQPIVKG EDTGPIFAS [REDACTED]DVTVEGDFYV GRQF[REDACTED]NP[REDACTED]IEP DVAFAYMGDD GKCYIHSKS[REDACTED]GVHLHLYMIA
4 PGVGLPEPQL VLVANP[REDACTED]I [REDACTED]YKFSPTSE ALVAVAAMAT [REDACTED]RPVHLRYNY QQQQYTGKR SPWEMNVKFA AKK[REDACTED]GTLLAM ESDWLVDHGP YSEFGDLLTL
5 RGAQFIGAGY NIPNIRGLGR TVATNHVWGS AF[REDACTED]GYGAPQS MFASECLMDM L[REDACTED]EKL[REDACTED]MDPL ELRYKNAYRP GDTNPTGQEP EVFSLPDMID QLRPKYQAAAL
6 EKAQKESTAT HKKGV[REDACTED]ISIG VYGSGLDGPD ASEAWAELNA DGTITVHTAW E[REDACTED]GQ[REDACTED]ADIG CVGTAHEALR PMGVAPEKIK FTWPNTATTP NSGPGSGSRQ
7 QVMTGNAIRV [REDACTED]CENLLKACE KPGGGYYTYD ELKAADKPTK ITGNMTASGA THCDAVTGLG KPFVVMYMGV FMAEVTVDVA TGQTTVDGMT LMDLGLSLCN
8 QLATDGIYQ GLAQIGLAL SEDFEDIK[REDACTED]ATLVGAGFPF IKQIPDKLDI VVYVNHPRPDG PFGASGVGEL PLTSPHA[REDACTED]I NAIKSATGVR IYRLPAYPEK
9 VLEALKA _ _

```

(b) Aldehyde dehydrogenase (FAD-independent)

Figure 2.4 – 3D and 1D positions of most important residues highlighted by our interpretability method AttnAgg2A1A on Nh(3)-dependent nad(+) synthetase, one of the best example of catalytic site retrieval, and Aldehyde dehydrogenase (FAD-independent), the worst example of catalytic site retrieval. The 5% most important residues for our interpretability method are highlighted in red and catalytic sites identified in M-CSA database are represented by blue spheres.

USING PRIOR INFORMATION FOR HIERARCHICAL MULTI-LABEL PREDICTION

3.1 Introduction

The second part of this thesis was dedicated to the prediction of all functions using the vocabulary of Gene Ontology (GO) and leveraging the Gene Ontology structure to improve prediction quality in AFP. In contrast to the EC hierarchy, which adheres to a strict hierarchical relationship, the GO is an ontology that permits more intricate relationships, as child terms can have multiple parents. Furthermore, our research aimed to tackle the challenges posed by multi-label scenarios, which encompass predictions across (i) multiple levels, including both generic and specific GO terms, as well as (ii) multi-function cases, acknowledging that certain proteins can possess multiple distinct functions.

The primary objective of this research was to integrate Gene Ontology information into the predictions' framework as a priori information. There exist multiple ways to integrate Gene Ontology information, one example is into neural network architecture itself ([Bou+21],[KKH18],[Ma+18],[Fio+18]). But these poses a challenge in determining the optimal order for making predictions, namely, whether to prioritize more specific predictions or more generic ones first. In this work, two other categories of methods were tested for this purpose. The first involved integrating Gene Ontology information in the conversion of annotations to label vectors (labeling process), while the second focused on integrating it into the structure of the prediction space in our neural networks. Figure

3.1 provides a comprehensive overview of the developed framework, which enables these comparisons. We will describe specific terms in the sections later on.

Before delving into the integration process and using the associated annotations, the first two sections present and analyze Gene Ontology and its annotation properties. Additionally, these sections introduce commonly used performance metrics for automatic function predictions. The subsequent two sections discuss the two categories for integrating this information.

3.2 Gene Ontology (GO) analysis

In this section, we conducted thorough analyzes to explore the GO and its annotations. Our objective was to gain a comprehensive understanding of the data and handle it accurately. By delving into the ontology and annotations, we aimed to extract meaningful insights and navigate the hierarchical structure. We also examined the distribution of annotations across sub-ontologies.

3.2.1 Gene Ontology terms

The Gene Ontology (GO) consists of a controlled vocabulary comprising GO terms and relations between them. The GO terms are categorized into three namespaces: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC). These namespaces have already been presented in section 1.1.6. Two file formats are available to download the gene ontology. The Open Biological and Biomedical Ontologies (OBO), that is human readable and machine-readable. And the OWL (Web Ontology Language) format is not directly human-readable but allows easier specification of more complex ontology. Additionally, the Gene Ontology Consortium (GOC), proposes three different versions of the GO with varying complexity (`go-basic/go/go-plus`). The first one is called `go-basic`, which exclusively includes relations between GO terms within the same namespace. It retains relations that go toward the root of the ontology, specifically `is_a`, `part_of`, `regulates`, `negatively_regulates`, and `positively_regulates`. The `go` version encompasses two additional relations, namely `has_part`, which can be directed to-

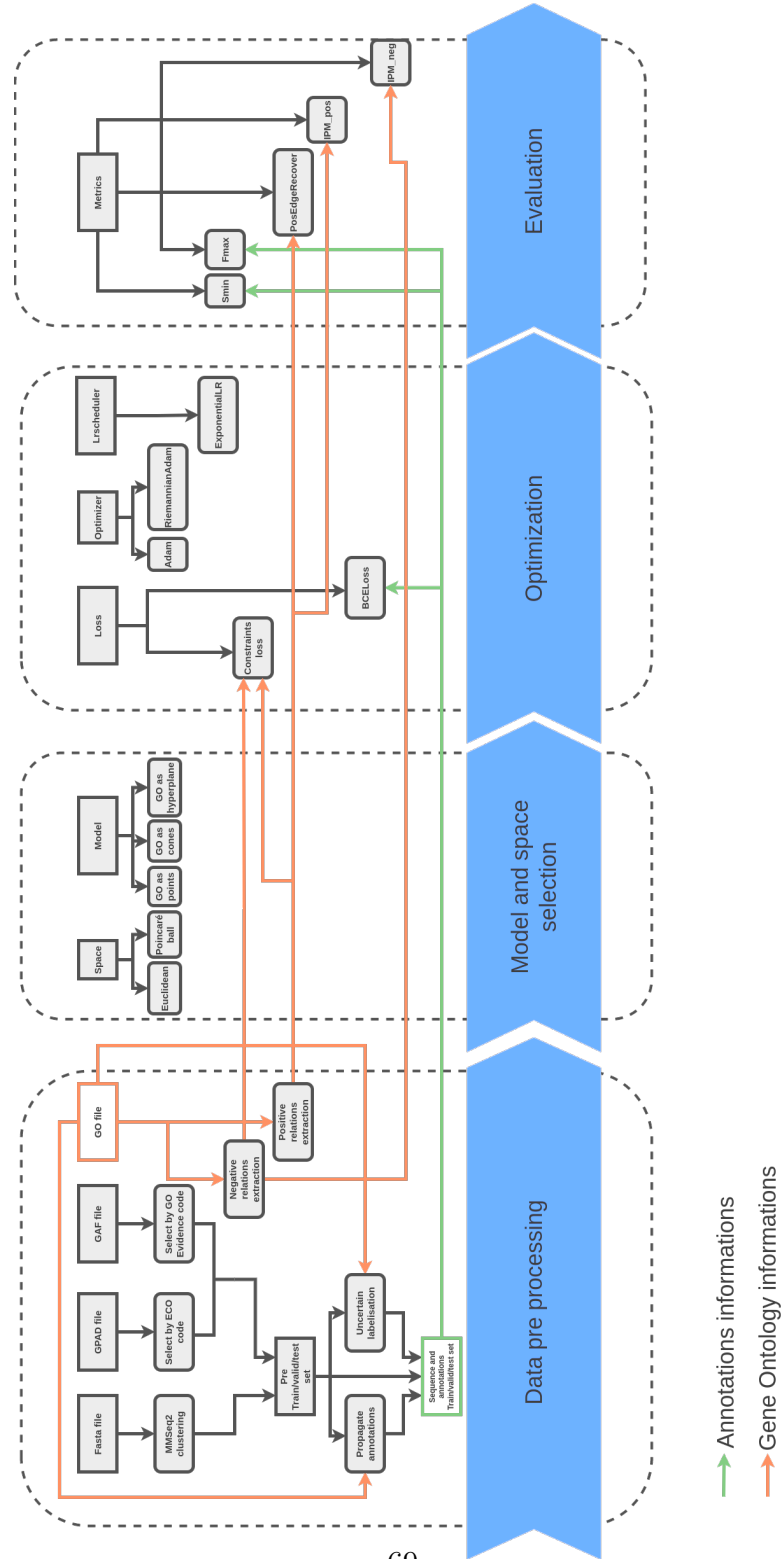


Figure 3.1 – Global pipeline step showcasing re-implementation and method unification, encompassing diverse components. Starting with data preprocessing, which leverages Gene Ontology information and annotations, followed by space and model selection. The training process entails the careful selection of loss functions and optimizers. Finally, performance evaluation and model coherence assessment are conducted using metrics, ensuring a comprehensive evaluation of the resulting model.

wards or away from the root, and `occurs_in`. On the other hand, the `go-plus` version incorporates additional cross-database relations and represents a fully axiomatized version¹. For the purpose of this work, the "go" version is utilized. The relationships are defined by the OBO Relations Ontology (RO), which consists of a collection of OWL relations designed to be employed across various biological ontologies. The following section presents the eight primary relation types available between GO terms.

The most significant relation type is `is_a`, representing a subtype relationship. For instance, *lyase activity* `is_a` *catalytic activity*. The next relationship, `part_of`, denotes a part-whole relationship where A is necessarily `part_of` B. For example, *mitochondria* `part_of` *cytoplasm*. The `regulates` relation is utilized when one process directly influences the manifestation of another process or quality. For instance, *signaling receptor regulator activity* `regulates` *signaling receptor activity*. The `positively_regulates` and `negatively_regulates` relations are employed when there is a consistent effect. The `occurs_in` relation is used for biological processes or molecular functions that occur within a cellular component. Lastly, `has_part` represents a part-whole relationship from the perspective of the parent term.

From the GO OBO file (version of April 2023) the number of relations between each specific namespace is extracted and shown in table 3.1. As seen, the `is_a` relation is only between GO terms of the same namespace and it is the most common. `Regulates` can only be between MF and BP.

In the latest version of Gene Ontology, there are certain obsolete GO terms included for historical reasons, which are marked with a specific tag. Out of the total terms in the latest version, approximately 4,404 terms are classified as obsolete. This leaves us with a current count of 43,093 active GO terms. However, it is important to note that the distribution of these terms across the different namespaces is not uniform. A breakdown of the total GO terms in each namespace can be found in Table 3.2. Specifically, the Gene Ontology (GO) terms in the Biological Process (BP) namespace make up 27,789 terms, while the Molecular Function (MF) and Cellular Component (CC) namespaces consist of 11,261 and 4,043 terms respectively.

1. An axiomatized version means that the relationships, constraints, and logical rules governing the ontology are explicitly defined and specified using description logic in the case of the GO.

Relation name	From namespace	To namespace	Number of relations
is_a	Biological process	Biological process	53 271
is_a	Molecular Function	Molecular Function	13 495
is_a	Cellular component	Cellular component	4719
part_of	Biological process	Biological process	5247
part_of	Cellular component	Cellular component	2190
part_of	Molecular Function	Biological process	1070
part_of	Molecular Function	Molecular Function	11
regulates	Biological process	Biological process	3198
regulates	Biological process	Molecular Function	298
regulates	Molecular Function	Molecular Function	30
regulates	Molecular Function	Biological process	2
negatively_regulates	Biological process	Biological process	2766
negatively_regulates	Biological process	Molecular Function	267
negatively_regulates	Molecular Function	Molecular Function	42
positively_regulates	Biological process	Biological process	2752
positively_regulates	Biological process	Molecular Function	275
positively_regulates	Molecular Function	Molecular Function	27
has_part	Biological process	Biological process	225
has_part	Molecular Function	Molecular Function	202
has_part	Cellular component	Cellular component	178
has_part	Biological process	Molecular Function	173
occurs_in	Biological process	Cellular component	150
occurs_in	Molecular Function	Cellular component	42
happens_during	Biological process	Biological process	8

Table 3.1 – The different relations types and between which namespace they operate with their respective number of edges from obo format go in the gene ontology website from April 2023

It is worth mentioning that the distribution of intermediate and leaf GO terms differs significantly among the three sub-ontologies. As illustrated in Table 3.2, the number of GO terms in the BP and MF namespaces exhibit a considerable difference; however, this gap narrows significantly when only the number of leaf terms is considered. Figure 3.2 shows a histogram of the graph distance between the different GO terms and the root of their namespace. The leaf annotation can be at varying distances from the root, some are at a distance 2 of the root and others at a distance 11. Moreover, in the biological process, the proportion of intermediate GO term compared to leaf GO term is a lot bigger than in the other ontology. It can be observed that the CC namespace tends to have leaves terms closer to the root than the two others. Having gained knowledge of gene ontology terms, the next step is to explore its application in protein annotation.

Namespace	Number of GO terms	Number of leafs	Number of intermediate terms
Biological Process	27 789	14 687	13 102
Molecular Function	11 261	9216	2045
Cellular Component	4043	3169	874

Table 3.2 – Number of GO terms, leaves, and intermediate terms for each sub-ontology: MF, BP, and CC

3.2.2 Gene Ontology annotations

3.2.2.1 Introduction

Gene Ontology (GO) annotations are created by trained individuals called curators who possess expertise in GO terms and follow established guidelines to extract precise and reliable information from scientific literature [PG17]. These curators employ their knowledge of GO and employ best practices to ensure the accuracy and quality of the annotations. The GOA (Gene Ontology Annotations) database, as described in [Hun+15], is the most comprehensive repository for such annotations. It includes annotations for proteins, non-coding RNA, and complexes, with associated UniProtKB IDs. The database incorporates both manual and automatic assertions. In the latest version (version 215)

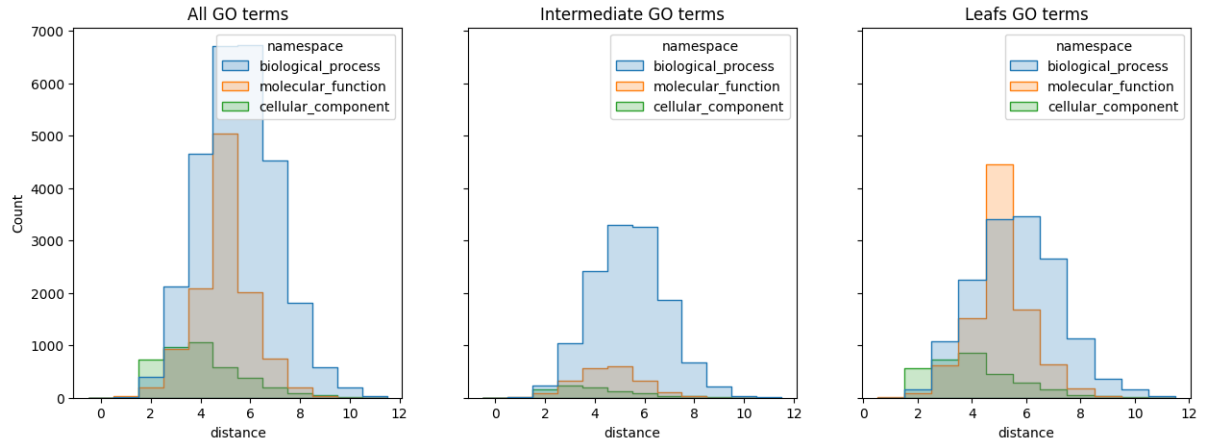


Figure 3.2 – Graph distance of all GO terms to the root term of their respective namespace

of Uniprot-GOA², manual annotations from 60 different annotator groups can be found. Notable examples include the Mouse Genome Informatics (MGI), which provides informatics resources for laboratory mouse and related human biology data, and the Human Protein Atlas, a Swedish-based program launched in 2003 with the objective of mapping all human proteins in cells, tissues, and organs using various omics technologies such as antibody-based imaging, mass spectrometry-based proteomics, transcriptomics, and systems biology. Additionally, multiple automatic annotations are derived from diverse tools, such as phylogenetic-based propagation of functional annotations with PANTHER [Tho+03], logical inference, and sequence similarity. This raises the question of whether some of these automatic annotations are of sufficient quality to be included in the training dataset for our models. The database can be accessed in two file formats: GAF (Gene Association File) and GPAD (Gene Product Association Data). The GAF format begins with the GAF version, followed by a header that defines the 17 columns³. Subsequently, the file consists of tab-separated entries for each annotation. Each annotation represents a specific gene product linked to a particular GO term. On the other hand, the GPAD format closely resembles GAF, with the exception of the available columns. Some in-

2. https://www.ebi.ac.uk/GOA/uniprot_release

3. Detailed in <http://geneontology.org/docs/go-annotation-file-gaf-format-2.2/>

formation that is present in the GPAD format is obtained from another Gene Product Information (GPI) file. The GPAD format consists of 12 columns.

3.2.2.2 Annotation description

An annotation represents a connection between a gene product (specifically proteins in our context) and a Gene Ontology (GO) term. Experimental annotations typically occur at the most specific level, without annotating all ancestral terms. However, certain annotations may exist at a more general level, even if the most specific term is already known, if they differ in other characteristics. For example, the Gene Association File (GAF) contains 17 columns of annotation, including the **Evidence Code**, the **DB ref code**, and the relation type (**Qualifier**). Similar to relationships between GO terms, various relation types can exist between gene products and GO terms. Table 3.4 displays the counts of edges for various relation types. The four most common relations between a Gene Product and a GO term are **enable**, **involved_in**, **located_in** and **part_of**⁴. These counts come from the EBI UniProt GAF annotations, available at ftp.ebi.ac.uk/pub/databases/GO/goa/UNIPROT/goa_uniprot_all.gaf.gz. All subsequent analyses use annotations from version 215 (generated on March 15, 2023 by EBI). While there is a higher number of Gene Ontology (GO) terms associated with the Biological Process (BP) namespace, the majority of annotations are actually found within the Molecular Function (MF) namespace, as indicated in Table 3.3. Additionally, Table 3.3 reveals that the majority of annotations are at the intermediate level rather than at the leaf level. This suggests that the proteins being annotated are not fully characterized at the most specific level. It is important to note that this cannot be solely attributed to the presence of more intermediate terms. In this dataset of annotations, the annotations are made at the most specific level whenever possible.

Relation type	Number of annotations
enables	487 872 818

4. It is important to note that these relation types also exist in the context of GO terms connected with other GO terms, but in this context, we are specifically referring to the Gene Product connecting to a GO term

involved_in	291 789 613
located_in	235 166 271
part_of	16 612 961
is_active_in	953 611
acts_upstream_of_or_within	150 083
contributes_to	33 884
colocalizes_with	8582
acts_upstream_of	5908
NOT involved_in	4578
NOT enables	4207
NOT located_in	1691
acts_upstream_of_or_within_positive_effect	507
acts_upstream_of_positive_effect	437
NOT is_active_in	347
NOT acts_upstream_of_or_within	327
acts_upstream_of_negative_effect	214
NOT part_of	141
acts_upstream_of_or_within_negative_effect	122
NOT colocalizes_with	51
NOT contributes_to	27
NOT acts_upstream_of_or_within_negative_effect	4
NOT acts_upstream_of_or_within_positive_effect	3
NOT acts_upstream_of	1

Table 3.4: All the different relation qualifier order by number of annotation available in the EBI UniProt GAF file

3.2.2.3 GO evidence code and ECO

Another important column in the GAF file is the evidence code. An evidence code is a classification or label used to indicate the type of supporting evidence for a gene function annotation, such as experimental, phylogenetic, computational, author statement, cura-

Namespace	Nb intermediate annotations	Nb leaf annotations	Nb total
Biological Process (BP)	253 328 798	38 622 999	291 951 797
Molecular Function (MF)	328 923 540	158 987 396	487 910 936
Cellular Component (CC)	243 338 345	9 405 310	252 743 655

Table 3.3 – Number of intermediate annotations, leaf annotations, and the total number of annotations for each namespace

tor statement, or electronically derived evidence. The GOC defines 26 evidence code, but only 24 appears in the annotations. The number of annotations for each evidence code is represented in Table 3.5. The most common evidence code by a large margin is **Inferred from Electronic Annotation (IEA)**. But this controlled vocabulary is not very fine-grained. It exists another controlled vocabulary with an ontology for the evidence code, the ECO vocabulary. It is more fine-grained with 2106 terms in April 2023. This is only available in the GPAD+GPI file format and not in the GAF file format. Some mapping exists between these two vocabularies but necessitates other information like the DB_ref columns. The count for the top 20 ECO codes is shown in Table 3.6. The term **evidence** is the most common since all the terms are annotated under this root term. Then the "evidence used in the automatic assertion" reflects the "IEA" from the GO evidence code. But then the next evidence code can give us some information on which type of electronic inference gives the most annotations. For example, nearly half a billion come from "sequence similarity evidence" and about 33 million from "evidence-based on logical inference from automatic annotation used in the automatic assertion". The ECO codes in the top 20 list primarily correspond to automatic assertions, but to identify experimental assertions, one can use the recommended CAFA (Critical Assessment of Functional Annotation)⁵ GO evidence codes (old evidence code) and convert them to their corresponding ECO codes (new ontology) using a mapping. According to CAFA guidelines, the recommended experimental codes for training purposes are: "EXP", "IDA", "IPI", "IMP", "IGI", "IEP", "TAS", "IC", "HTP", "HDA", "HMP", "HGI", and "HEP". These codes can be further mapped to the ECO (Evidence and Conclusion Ontology) to provide minimal recommen-

5. Which organizes the most important challenge to evaluate AFP tools

dations, which include: ECO:0000269 (EXP), ECO:0000305 (IC), ECO:0006056 (HTP), and ECO:0000304 (TAS). By considering these mapped ECO codes and their child ECO codes, new evidence codes can be discovered that were not recommended by CAFA. This reveals the problem of coherence, which may not have been apparent with the original evidence codes but becomes visible with the ECO ontology. For example, the ND GO evidence code corresponds to ECO:0000307.

In Table 3.7, the counts are presented for the top 10 ECO codes specifically related to manual annotations, as recommended by CAFA (Critical Assessment of Functional Annotation) GO evidence codes. The ECO code that is most frequently associated with annotations is "direct assay evidence used in the manual assertion". Additionally, the ECO code "physical interaction evidence used in the manual assertion", which is a subtype of the direct assay, and "experimental phenotypic evidence used in the manual assertion" also rank high in terms of annotation counts.

As previously mentioned, CAFA recommends using experimental evidence codes for training, which cover annotations for approximately 239,553 different gene products. However, if manual evidence codes are included, this number increases to approximately 1,288,698 different gene products annotated. Manual evidence codes typically involve transferring annotations with the assistance of automated tools and validation by an expert biologist who utilizes their background knowledge to verify the annotations.

Evidence code	Number of annotations
Inferred from Electronic Annotation (IEA)	1 026 752 240
Inferred from Biological aspect of Ancestor (IBA)	3 488 350
Inferred from Sequence Orthology (ISO)	507 279
Inferred from Direct Assay (IDA)	392 955
Inferred from Physical Interaction (IPI)	368 318
Inferred from Sequence or structural Similarity (ISS)	306 002
Inferred from Mutant Phenotype (IMP)	237 290
No biological Data available (ND)	219 580
Traceable Author Statement (TAS)	133 125
High Throughput Direct Assay (HDA)	52 151

Inferred from Genetic Interaction (IGI)	43 493
Non-traceable Author Statement (NAS)	29 858
Inferred from Expression Pattern (IEP)	27 258
Inferred from Sequence Alignment (ISA)	11 488
Reviewed Computational Analysis (RCA)	11 295
Inferred from Sequence Model (ISM)	8448
Inferred by Curator (IC)	7275
Experiment (EXP)	5104
High Throughput Mutant Phenotype (HMP)	2510
High Throughput Expression Pattern (HEP)	1117
Inferred from Key Residues (IKR)	662
Inferred from Genomic Context (IGC)	525
High Throughput Genetic Interaction (HGI)	64
High Throughput Experiment (HTP)	1

Table 3.5: Number of annotations for each evidence code in the EBI Uniprot GAF file only for protein

The GAF (Gene Association File) and GPAD (Gene Product Association Data) files do not directly provide the protein sequences. Therefore, a mapping process is required to associate the annotations with the corresponding protein sequences. One might consider using the "DB Object ID" field to retrieve the sequence from UniProtKB. However, this ID does not provide information about the specific isoform. To address this, the "Gene Product Form ID" column is utilized. This column contains the UniProt ID followed by a hyphen and the specified isoform (e.g., P12345-2). Within the file, there are 4,373 protein sequences associated with a UniProt ID that has multiple known isoforms, out of a total of 228,804 proteins.

With a clearer understanding of Gene Ontology and its annotations, as well as their sources, it is now possible to delve into the evaluation of models that automatically assign protein function annotations.

3.3. Performance metric for evaluating Automatic Function Prediction tools

ECO code	Name	# of annotations
ECO:0000000	evidence	1 065 496 679
ECO:0000501	evidence used in automatic assertion	1 059 547 960
ECO:0007672	computational evidence	564 412 826
ECO:0007669	computational evidence used in automatic assertion	563 587 316
ECO:0000361	inferential evidence	500 676 129
ECO:0007832	inferential evidence used in automatic assertion	500 437 416
ECO:0000041	similarity evidence	497 756 919
ECO:0000044	sequence similarity evidence	494 154 660
ECO:0000249	sequence similarity evidence used in automatic assertion	493 329 150
ECO:0000251	similarity evidence used in automatic assertion	493 329 150
ECO:0000202	match to sequence model evidence	483 898 485
ECO:0000256	match to sequence model evidence used in automatic assertion	483 890 022
ECO:0000205	curator inference	466 742 704
ECO:0007322	curator inference used in automatic assertion	466 503 991
ECO:0000362	computational inference	33 933 425
ECO:0000363	computational inference used in automatic assertion	33 933 425
ECO:0000366	evidence based on logical inference from automatic annotation used in automatic assertion	33 795 111
ECO:0000201	sequence orthology evidence	9 939 027
ECO:0000265	sequence orthology evidence used in automatic assertion	9 439 128
ECO:0000352	evidence used in manual assertion	5 948 719

Table 3.6 – Top 20 of ECO code with their number of annotations. Some annotations can have multiple ECO codes because they are propagated to the root.

3.3 Performance metric for evaluating Automatic Function Prediction tools

AFP (Automatic Function Prediction) tools typically provide a confidence score, often in the form of a probability, for each association between proteins and Gene Ontology (GO) terms. The metrics presented in this section come from the recommendation of the CAFA challenge ([Zho+19]), which evaluates the performance of different models for the AFP task. The F_{\max} metric is commonly used to evaluate these tools ([Zho+19]). It involves identifying the optimal threshold that yields the highest scores for the harmonic

ECO code	Name	# of annotations
ECO:0000314	direct assay evidence used in manual assertion	809 152
ECO:0000353	physical interaction evidence used in manual assertion	368 721
ECO:0007634	experimental phenotypic evidence used in manual assertion	284 643
ECO:0000315	mutant phenotype evidence used in manual assertion	240 471
ECO:0000307	no evidence data found used in manual assertion	210 964
ECO:0007005	high throughput direct assay evidence used in manual assertion	52 151
ECO:0000316	genetic interaction evidence used in manual assertion	43 650
ECO:0000270	expression pattern evidence used in manual assertion	27 444
ECO:0007746	biological system reconstruction evidence used in manual assertion	20 544
ECO:0005547	biological system reconstruction evidence based on inference from background scientific knowledge used in manual assertion	16 027

Table 3.7 – Top 10 of ECO code with their number of annotations only annotation recommended for training: ECO:0000269(EXP), ECO:0000305(IC), ECO:0006056(HTP), ECO:0000304(TAS)

mean of recall and precision (See equation 3.3). The recall (rc) represents the proportion of true predictions retrieved from all positive instances, while precision (pr) indicates the proportion of true positive predictions from all positive predictions.

$$pr(\tau) = \frac{1}{N} \sum_{i=1}^N \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in P_i(\tau))} \quad (3.1)$$

$$rc(\tau) = \frac{1}{N} \sum_{i=1}^N \frac{\sum_f \mathbb{1}(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f \mathbb{1}(f \in T_i)} \quad (3.2)$$

$$F_{\max} = \max_{\tau} \left\{ \frac{2 \cdot pr(\tau) \cdot rc(\tau)}{pr(\tau) + rc(\tau)} \right\} \quad (3.3)$$

Let N represent the number of proteins. The symbol $\mathbb{1}$ denotes the indicator function, which takes a value of one when a condition is true and zero when it is false. The set

$P_i(\tau)$ represents the GO terms predicted for protein i with a probability greater than τ . Similarly, T_i represents the set of true GO terms for protein i . The symbol f iterates over all possible GO terms. The variable τ ranges from 0 to 1 and represents the threshold for the model’s output probability.

However, the F_{\max} metric may not be the most suitable for evaluating classes organized as an ontology. This is because some Gene Ontology (GO) terms are highly generic, making them easier to predict accurately, while others are more specific and pose greater challenges for accurate predictions. To address this issue, one approach is to calculate the information accretion (IA), also known as information content (IC), for each Gene Ontology (GO) term ([CR13]). IA represents the negative logarithm of the probability of observing a specific GO term given the presence of its parent GO terms (See equation 3.6). This measure helps quantify the amount of information contained within each GO term, considering its relationship with higher-level terms in the ontology.

$$Parent_{GO} = \{p | \exists (GO, is_a, p) \in Edges\} \quad (3.4)$$

$$Pr(GO | Parent_{GO}) = \frac{Occ(GO)}{\min_{p \in Parent_{GO}} (Occ(p))} \quad (3.5)$$

$$IC(GO) = -\log(Pr(GO | Parent_{GO})) \quad (3.6)$$

By quantifying the information content, it becomes possible to establish a meaningful metric known as the minimum semantic distance (S_{\min}). This metric aims to evaluate and rank classification models based on their semantic distance. Unlike existing metrics, S_{\min} introduces two information-theoretic concepts: remaining uncertainty and misinformation. The concept of remaining uncertainty measures the information about a protein’s true annotation that is not provided by the predicted annotation (See equation 3.7). On the other hand, misinformation quantifies the total information content of incorrect paths in the predicted graph (See equation 3.8). To calculate these values, we incorporate the information accretion associated with specific nodes in the graph. To determine the S_{\min} metric, we identify the threshold that minimizes the square root of the squared remaining uncertainty (ru) and misinformation (mi), as shown in Equation 3.9. By considering both

the remaining uncertainty after prediction and the potential for misinformation in the predictions, this metric provides a comprehensive evaluation of classification models.

$$ru(\tau) = \frac{1}{N} \sum_{i=1}^N \sum_f ic(f) \cdot \mathbb{1}(f \notin P_i(\tau) \wedge f \in T_i) \quad (3.7)$$

$$mi(\tau) = \frac{1}{N} \sum_{i=1}^N \sum_f ic(f) \cdot \mathbb{1}(f \in P_i(\tau) \wedge f \notin T_i) \quad (3.8)$$

$$S_{\min} = \min_{\tau} \left\{ \sqrt{ru(\tau)^2 + mi(\tau)^2} \right\} \quad (3.9)$$

It is also possible to integrate information accretion into classical F_{max} metric by weighting the GO class in the F_{max} metric (See 3.12).

$$wpr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_f ic(f) \cdot \mathbb{1}(f \in P_i(\tau) \wedge T_i(\tau))}{\sum_f ic(f) \cdot \mathbb{1}(f \in P_i(\tau))} \quad (3.10)$$

$$wrc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \frac{\sum_f ic(f) \cdot \mathbb{1}(f \in P_i(\tau) \wedge T_i(\tau))}{\sum_f ic(f) \cdot \mathbb{1}(f \in T_i(\tau))} \quad (3.11)$$

$$WF_{\max} = \max_{\tau} \left\{ \frac{2 \cdot wpr(\tau) \cdot wrc(\tau)}{wpr(\tau) + wrc(\tau)} \right\} \quad (3.12)$$

Most often the different metrics are computed per namespace (Molecular Function, Biological Process and Cellular Component). To have a unique performance number it is also possible to do the arithmetic mean of the different weighted- F_{max} (WF_{max}) of all the different namespaces. This is the chosen metric for the CAFA5 challenge.

3.4 Integrating Gene Ontology information in the labelling

We focus here on the prediction of classes within an ontology structure and the potential benefits of using ontology to enhance the labeling process that uses annotations. There are several approaches available for integrating ontology knowledge into the anno-

tations. The initial and widely used method involves propagating the labels from a specific function to annotate all its more generic parent functions, known as the True Path Rule (TPR), an example is shown in Figure 3.3. However, to our knowledge, the advantages of this propagation technique have not been thoroughly examined and quantified in previous studies. Therefore, this thesis aims to investigate the effects of such propagation in two distinct scenarios. The first scenario examines the situation where the training set consistently provides the most specific GO term. Subsequently, a more realistic scenario is considered, where some annotations are complete while others are not.

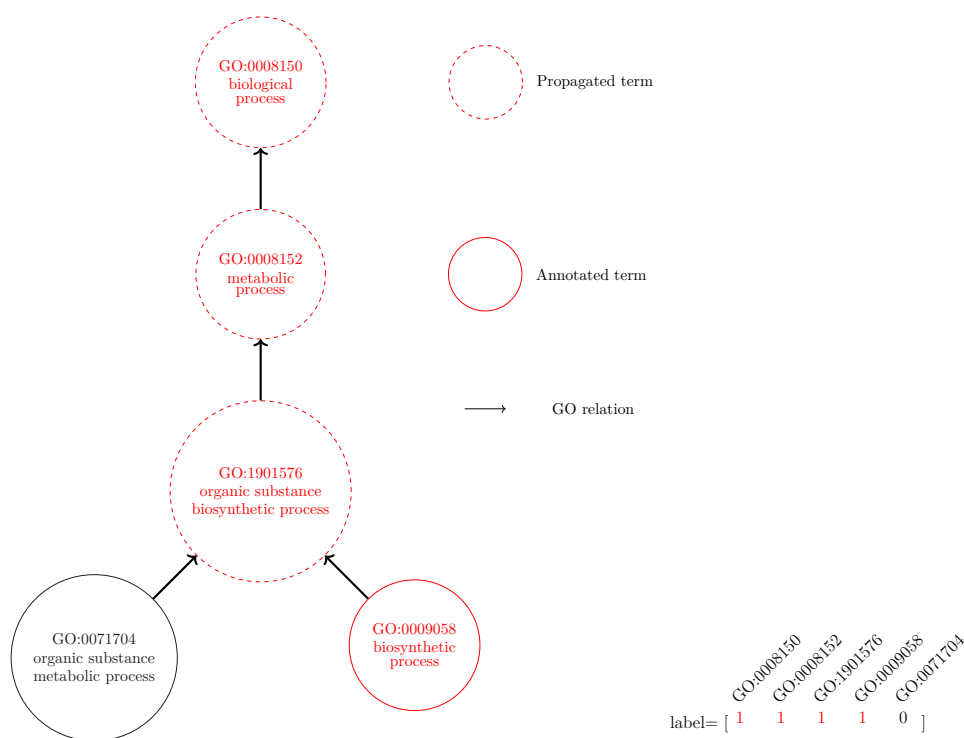


Figure 3.3 – An example of the "biosynthetic process" annotation for a protein and a potential propagation with the True Path Rule in the labeling process, which also annotated "organic substance metabolic process", "metabolic process" and "biological process".

An alternative approach for integrating ontology knowledge involves considering the Open World Assumption (OWA). When a protein lacks a specific GO annotation, it can be interpreted in two ways: either the protein truly does not possess that function, or there

has been no experimental evidence supporting its association with that function. Bearing this in mind, if a protein is annotated to an intermediate term rather than a leaf term, it is expected that the majority of its descendants are false but should contain at least one correct annotation (except in cases where the GO term describing the function does not exist yet). Consequently, it becomes possible to label all these subgraphs as uncertain and abstain from providing a signal during training for such examples. This approach contrasts with the traditional Closed World Assumption (CWA), where a true GO term is more likely to be falsely labeled as false (False Negative).

3.4.1 Experiment description

A first experiment will investigate this, two models will be compared: one trained to predict both leaf and intermediate classes, leveraging the available ontology information, and another model trained to predict only leaf classes without accessing the ontology. The evaluation will be performed solely on the leaf classes to assess any differences in performance between the two approaches. The training set will consist of only the protein where all annotations are complete to the leaf GO term.

The second question pertains to scenarios where incomplete annotations are present. In such cases, the investigation focuses on the potential improvement offered by a model trained with true classes propagated to the root, utilizing the True Path Rules. This will be compared to a model that only has access to the true class of the most specific annotation available for each instance.

The final question stems from the observation that when annotations are not provided at the leaf level, there is a high probability that at least some leaf annotations are correct. To address this, we explore the approach of avoiding the most probable false negatives and excluding a significant number of true negatives by providing no information on them. The objective is to assess whether this approach yields improved results compared to other methods.

All three experiments are described in Figure 3.4 and they share a common setup and similar strategies in determining the required number of replications to detect an effect. The following two sub-subsections will outline the common setup for the three

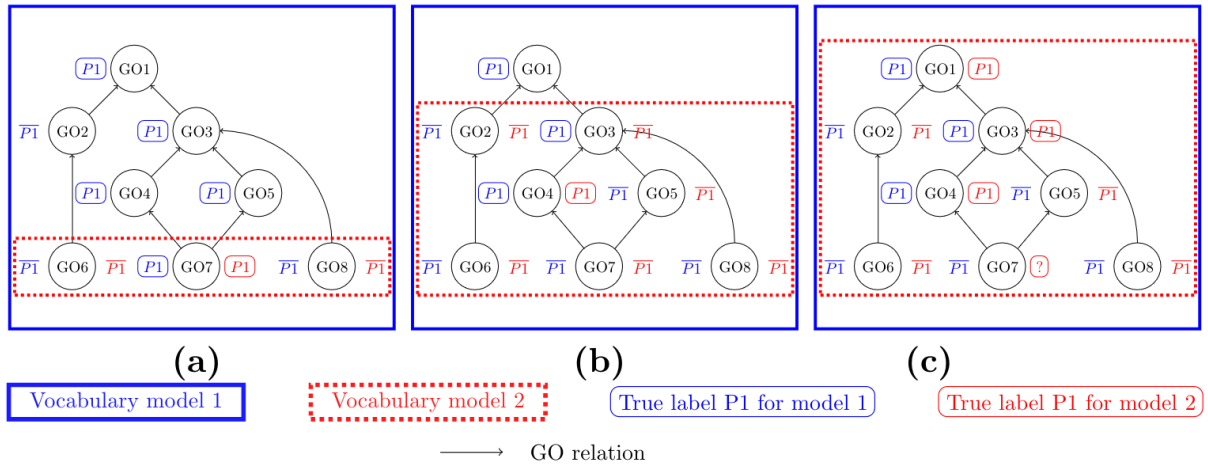


Figure 3.4 – Illustration of the three experiments conducted to incorporate Gene Ontology (GO) information into the labelling process: (a) Comparison of two models: the first one can predict annotations at all levels and the second can only predict leaf-level annotations. This experiment employs a dataset consistently containing the most precise GO annotation for each protein. (b) contrast between the two models: the first model predicts all GO terms, while the second model predicts only the GO terms associated with a protein’s most precise annotation. The first model considers the most precise annotation and all its ancestors as True, with the remaining annotations marked as False. The second model is trained using the most precise annotations as True and others as False. (c) Comparison between two models capable of predicting all GO terms. In the first model, there is no signal for the most precise annotations when they are unavailable. Conversely, the second model assigns all negative annotations if a protein is not annotated at the most precise level.

experiments. Firstly, the model and dataset will be described, followed by an analysis of the sample size necessary to detect a significant effect.

3.4.2 Common setup

First, a common setup is described to test the impact of these different approaches. In the following, our input will be the mean embedding of the residues of the proteins. This embedding is computed with the pre-trained model Prot-T5([Eln+21a]). Prot-T5 is a language model with an encoder and decoder architecture, trained on two different

model sizes. We specifically opted for the T5-XL variant, which has 3 billion parameters, instead of the T5-XXL with 11 billion parameters. Notably, the T5-XXL model did not exhibit superior performance according to [Eln+21b].

The training of Prot-T5 involved two datasets: BFD and UniRef50. The model employed BERT’s denoising objective to corrupt and reconstruct individual tokens. Notably, the encoder outperformed the decoder in various benchmarks, as reported in [Eln+21b]. Consequently, we discarded the decoder during inference to reduce the model size. Additionally, the model was trained using half-precision mode, as it did not exhibit diminished performance ([Eln+21b]).

Unlike our experiment on enzymes, we froze the weights of this part of the model and did not fine-tune it. This decision allowed us to utilize the larger and improved Prot-T5 model instead of the previous ProtBert model. To prevent potential out-of-memory errors, we truncated the input sequence size to 2048 residues.

After obtaining the mean embedding from Prot-T5, we performed further computations using a two-layer Multi-Layer Perceptron (MLP) with a GeLU (1.4) activation function. First, we begin by projecting the Prot-T5 mean embedding into a new embedding of size 1024. This projection serves two purposes: it enables the model to selectively emphasize pertinent features for function prediction. Subsequently, we apply another projection to map the output to the total number of available GO classes (47497). The classical Binary Cross Entropy (BCE) loss has been used with an Adam optimizer with a starting learning rate of 4.0e-3. An exponential learning rate scheduler was also used with a gamma of 0.9 and an end learning rate of 1e-5. The gamma is the multiplication factor used to multiply the current learning rate at each step. These values were obtained by testing different learning rates (between 1.0e-3 and 1.0e-5) with the HyperOptSearch class of the Ray Tune library that uses the Tree-structured Parzen Estimators algorithm. Ray Tune is a Python library that enables experiment execution and hyperparameter tuning of machine learning frameworks using advanced algorithms like PBT, HyperBand/ASHA, and integration with multiple hyperparameter optimization tools. And a batch size of 32 was used which was the maximum possible with the available GPU RAM.

The separation between the training and testing set was done like for EnzBert with mmseqs2 ([SS17]) at a 40% identity threshold. In order to verify if the following observed

effect was not due to chance in the selected test sequences, multiple replications for each parameter tested were done. To know in advance how much replication will be needed the first step will be to do a power analysis(See Section 3.4.3).

To construct the dataset, the annotations from version 215 of UniProt-GOA was used and all relations type between protein and GO terms are taken.

Datasets

Two distinct datasets were created for our analysis. The first dataset called the **Leaf Only Dataset**, consisted of proteins with comprehensive Gene Ontology (GO) term annotations that extended to the leaf level of the GO graph. In this dataset, all proteins had annotations for all the relevant GO terms, providing a complete and exhaustive set of annotations. The second dataset, known as the **AllDataset**, included annotations for all the proteins, regardless of whether the GO term annotations were at the leaf level or not. This dataset encompassed a broader range of annotations, including those that were not limited to the leaf nodes of the GO graph.

The F_{\max} metric was used because all evaluation was done on the leaf thus it is less of a concern to not weight by information content. No separation was done between the different namespaces here.

3.4.3 Sample size estimation - Power analysis

In this research study, we will conduct multiple experiments, with each experiment consisting of two variants. Each variant will be tested N times with shuffling of the train/test each time. Subsequently, we will compare the mean performance differences between the two variants. To carry out this analysis, we will employ a paired mean comparison using either a student test (parametric) or a Wilcoxon test(non-parametric), depending on the nature of the data distribution.

However, to ensure the validity of our experiments, it is crucial to determine the appropriate number of repetitions for each variant in order to detect a specific effect size. To address this requirement, we will conduct a power analysis, which will guide us in determining the optimal sample size necessary for our research.

In order to conduct a power analysis, conventional values will be utilized. These values

include a significance level (α) set at 0.05, which represents the probability of the study rejecting the null hypothesis assuming it is true. Additionally, a statistical power ($1-\beta$) of 0.8 will be employed, which signifies the probability of correctly rejecting the null hypothesis, also known as a "true positive.". More information on power analysis in [CG12].

This will only be used as a first guess because for simplicity a parametric t-test will be supposed for this sample size estimation. More precisely a paired t-test, where the standard deviation from previous experiments is estimated to be 0.008. Moreover, a minimum difference in the two means of 0.01 will be set, this seems reasonable for the variation of F_{\max} measure that is wanted (The impact of this value will be tested).

To find this sample size (N) two curves will be plotted. Because a paired test is supposed we will find n_1 that is equal to $\frac{N}{2}$. The first curve represents the t-value necessary to have an area of 0.975 ($1 - \frac{\alpha}{2}$) between $-\infty$ and the t-value for all tested n_1 . The second curve represents the t-value necessary for an area of 0.2 (β) between $-\infty$ and this t-value, from which the minimum difference between F_{\max} multiplied by $\sqrt{n_1}$ is appended. The resulting plot is shown in Figure 3.5, where a value of n_1 of 8.835 is found, which gives us a value of 17.67 for N. A representation of the two t-distribution is in Figure 3.6.

To assess the effects of the estimated standard deviation from the previous experiment and the minimum difference in F_{\max} between the two setups, a range of values was tested. The values explored ranged from 0.5 to 1.5 times the original estimated standard deviation and from a difference of 0.01 to 0.05. The estimated number of pair of samples for each combination is presented in Table 3.8, providing insights into the impact of these variables on the experimental design. Due to time constraints, a value of n_1 (number of paired samples) equal to 8 was selected for the experiment. This decision resulted in a total of 16 experiments and training runs for each experiment. This cover all values in our Table 3.8 except the 3 most extreme values in the top-right corner of the table with high standard deviation and very little effect.

3.4.4 True path rule propagation

In this subsection, the effect of the true path rule (TPR) will be explored. The true path rule consists in propagating a GO annotation to all his parent's more generic GO terms.

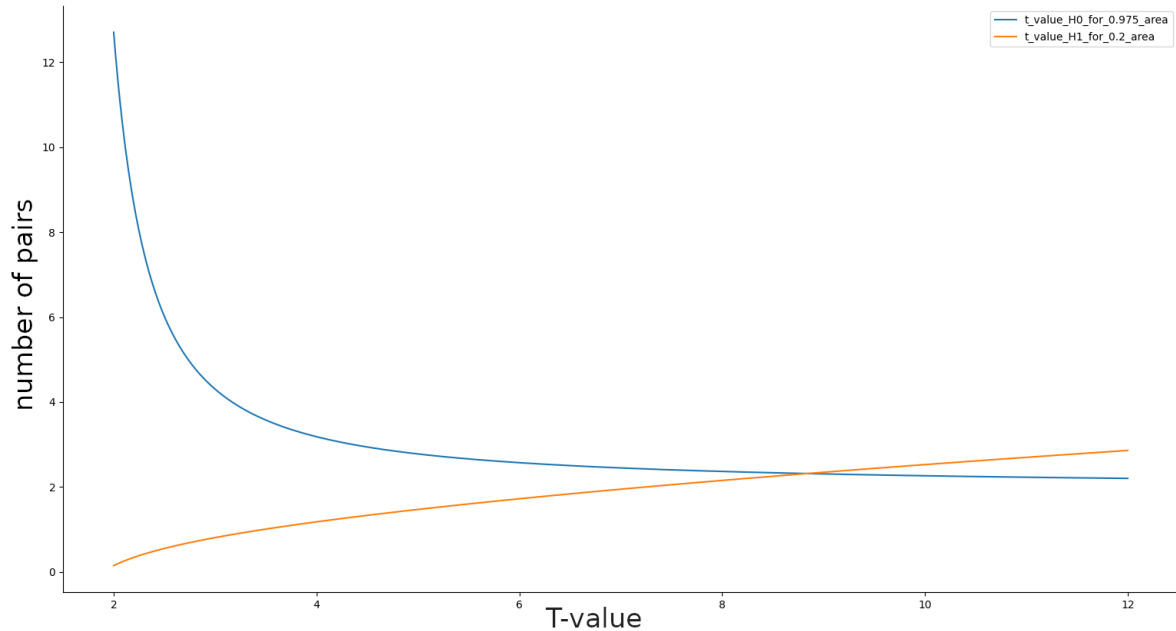


Figure 3.5 – The t-values, plotted against the number of samples, were computed for the central distribution under the null hypothesis (H0) to achieve an alpha value of 0.05. Additionally, the t-value for the offset t-distribution was determined, considering a difference of 0.01 with a desired power of 0.8.

For example, if a protein A has the annotation NAD⁺ kinase activity (GO:0003951), with the TPR the protein A has also the annotations kinase activity(GO:0016301) and all other ancestors.

Most of the models in the literature use the True Path Rule. Which consists of propagated annotations to all these ancestors to the root. But as seen before multiple relations exist between GO terms. In the literature, some only consider relations `is_a` ([KKH18]), some `is_a` and `part_of` ([KH19]), and others all relations except `has_part`⁶ [EMS22]. In the following, the simpler case will be considered with only the transitivity of the `is_a` relation.

6. Because it is in the reverse order and even if inverse could be interpreted as part of some and not necessarily part of.

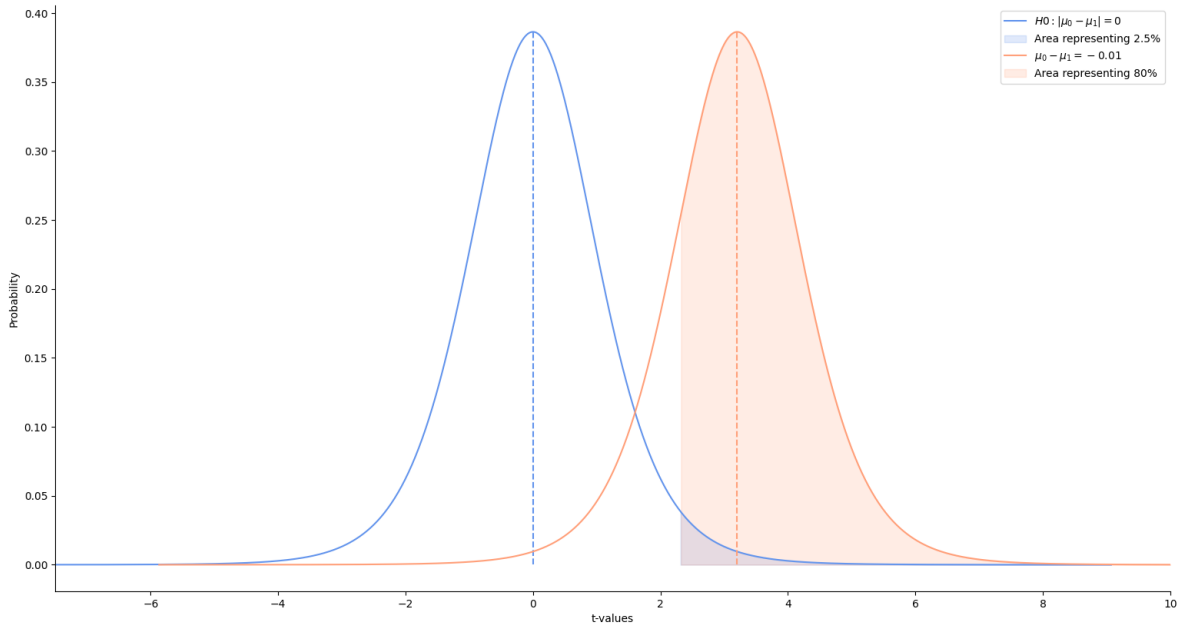


Figure 3.6 – The two t-distribution for the optimal value of n_1 equal 8.835. The blue area corresponds to 2.5% of the total area because it is a two-tailed test and the orange area that represent 80% of the total area.

3.4.4.1 True Path Rule with complete available annotations

The number of replications was quite low then a non-parametric version of the paired t-test, the Wilcoxon test, was performed. This choice provides less power to the statistical test but is more conservative to avoid detecting an effect that doesn't exist. The comparison was between a model that can predict all the annotations and use the TPR to propagate annotation and a model that only predicts leaf GO term (then without the TPR). The resulting p-value was 0.0078, which is less than the significance level of 0.05. Thus, we can reject the null hypothesis (H_0) that the true mean difference is zero, indicating a significant difference in F_{\max} between the two setups. The mean difference observed between the F_{\max} values was 0.011 in favor of the model that uses the TPR. The complete boxplot, without linking the paired samples, is shown on the left of Figure 3.7.

This result was not straightforward as it could have gone either way. On one hand, the

	Estimated std			
	0.005	0.007	0.010	0.013
Effect of 0.010	3.780	6.540	10.710	16.330
0.015	2.880	4.150	5.980	8.440
0.020	2.520	3.300	4.340	5.720
0.030	2.200	2.630	3.150	3.780
0.050	1.960	2.200	2.450	2.730

Table 3.8 – Number of replicas necessary to find an effect as small as the one specified in the "effect of" with different estimated standard deviations.

result could have been better due to the inclusion of ancestral information, which helps establish connections between different leaf GO classes. On the other hand, the result could have been worse because predicting more classes (including intermediate ones) might have come at the expense of sacrificing accuracy in predicting the important leaf classes.

3.4.4.2 True Path Rule with partial annotations

The previous setup does not accurately reflect the reality of Gene Ontology annotations. In practice, only around 10% of manually annotated proteins have all their functions specified up to the most detailed GO leaf. Therefore, the next setup includes both complete and incomplete annotations to investigate whether the propagation approach is still beneficial in this scenario and to assess its overall impact.

In this setup, the use of TPR was also statistically better (Wilcoxon test, $p\text{-value}=0.0078<0.05$). Like the other experiment, the difference in the mean is significant with an observed mean difference of 0.013. This difference is the same order of magnitude as the previous experiment.

The small difference observed in the results can be surprising, as we would expect the model without annotation propagation to introduce false negative labels by not propagating certain annotations. However, this may be explained by the assumption that there is a strong correlation between the patterns of the leaf class and its parent class. Although we do not have direct evidence of this correlation, it is assumed to exist based on the hierarchical nature of the Gene Ontology. The model may be able to learn and capture the patterns of the parent class from the available data, even without explicit annotation

propagation. The observed mean difference boxplot can be observed on the right of Figure 3.7.

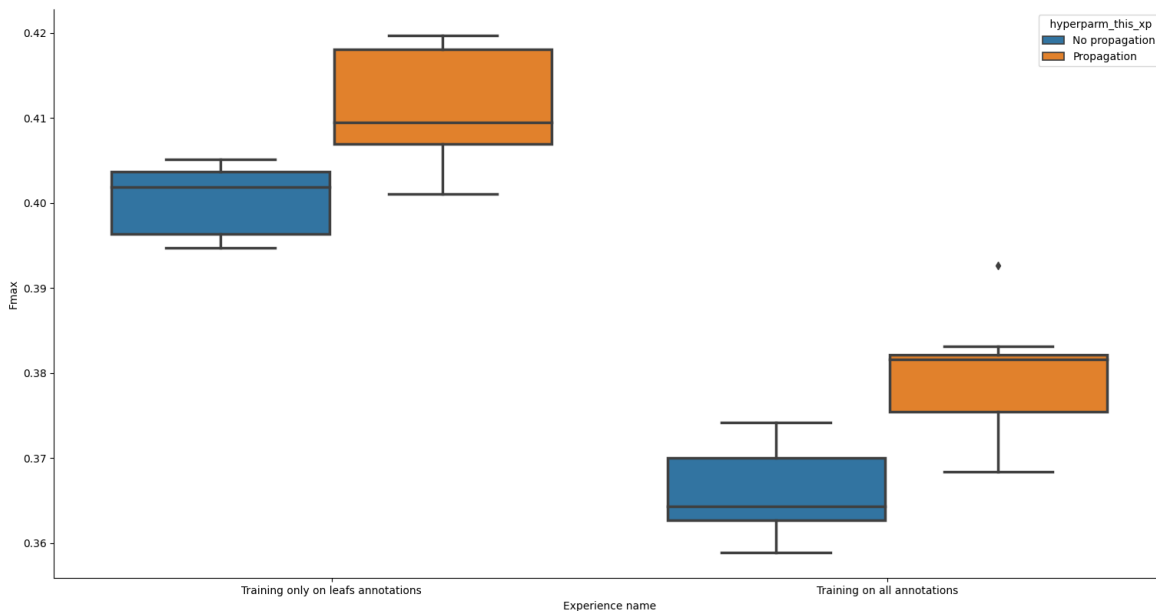


Figure 3.7 – Comparison of the implications of utilizing the True Path Rule for annotation propagation versus training solely on leaf GO terms without considering intermediate GO terms. The figure illustrates the difference in F_{\max} measure between the two variants.

In conclusion, both experiments provide further confirmation of the usefulness of the True Path Rules.

3.4.5 Ontology to avoid some false negative from the CWA

The available annotations typically provide information at the most specific level of the Gene Ontology (GO) hierarchy. However, there are cases where experimental evidence cannot conclusively determine the precise GO term, resulting in annotations made at intermediate GO terms. In such situations, it is often the case that the protein’s function corresponds to one of the descendants of the intermediate GO term, except for missing GO terms for specific functions.

During training under the Closed World Assumption (CWA), all the descendants of

the intermediate GO terms are labeled as not present. However, at least one of these descendants is likely positive, resulting in a False Negative in the "Target label". On the other hand, the advantage of CWA is that in cases where there are more than two descendants, the descendants generally have a higher probability of being negative than positive.

To address the issue of False Negatives, one approach is to consider all the descendants of the most specific GO terms as uncertain and assign them an uncertain label. During training, these uncertain classes can be handled by excluding them from the Binary Cross Entropy loss calculation. This means that no explicit information is provided for these classes, neither indicating correctness with a probability of one nor incorrectness with a probability of zero. By doing so, False Negatives can be avoided while still accounting for the uncertainty associated with these classes.

As depicted in Figure 3.8, the disparity in Fmax between the two settings seems minimal, in fact, the difference was found to be not significant (Paired Wilcoxon test, p-value=0.148>0.05). This could be attributed to a lower effect size than initially anticipated, or to the fact that the Fmax metric on incomplete test data may not be well adapted, despite some work([Jia+14]) saying that under realistic assumption incomplete knowledge on the evaluation has not a big effect.

3.5 Integrate Gene Ontology information in the structure of the space

In this section, we will explore how graph node embedding can be used as a complementary approach to label propagation for incorporating Gene Ontology information. This method involves assigning an embedding to each node, where each node represents a functional class (GO term) within the Gene Ontology. The embedding of each GO term enables the organization of the space. Additionally, the inclusion constraint derived from the Gene Ontology can be utilized to impose constraints on the GO term embeddings.

Various techniques for incorporating GO embedding were examined here and compared within a unified framework, focusing on protein function prediction and employing

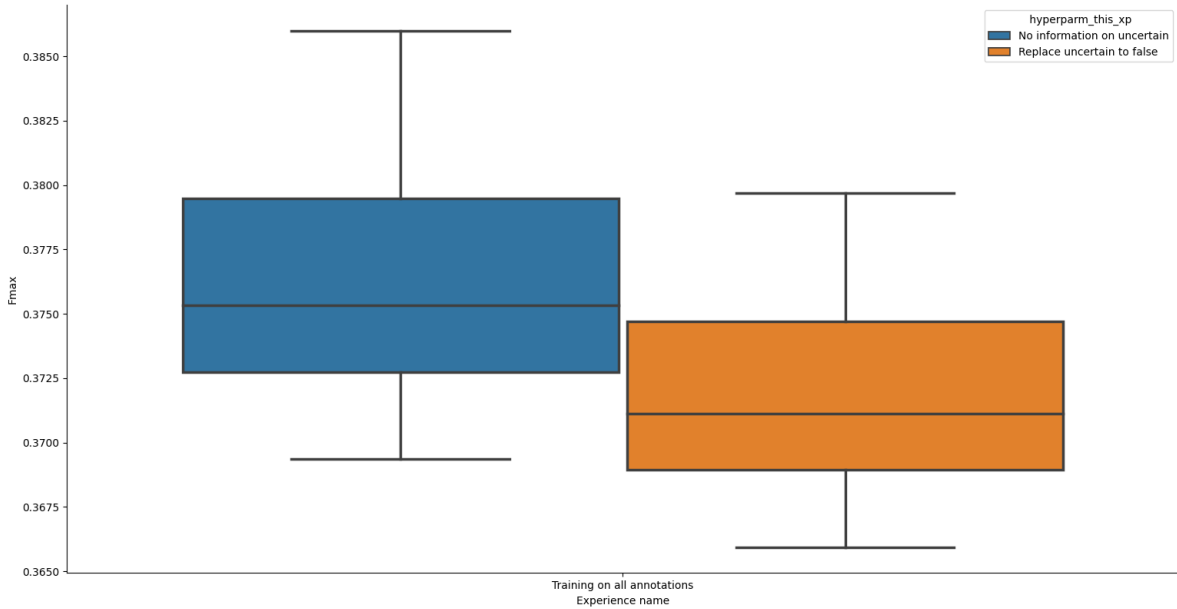


Figure 3.8 – Comparison of the difference in F_{\max} between the Open World Assumption (OWA) approach and the Closed World Assumption (CWA) approach during training.

classical domain metrics. These techniques will be presented in the following. It is worth noting that the Gene Ontology can be represented as a Directed Acyclic Graph (DAG). However, even simpler structures such as trees can present challenges when embedding them in euclidean space while preserving their metric properties. Research by De Sa et al. [De +18] has shown that arbitrary tree structures cannot be embedded with arbitrarily low distortion in Euclidean space. Interestingly, this task becomes considerably more manageable in hyperbolic space with just two dimensions, as the exponential growth of distances aligns with the exponential growth of nodes at varying depths in the tree. This highlights the suitability of hyperbolic space for accurately representing tree-like structures. Thus, the first basic model that we can think of is by encoding GO term by points on the manifold and using the hyperbolic distance to bring closer linked GO terms [NK]. But some other work ([GBH18b]) point out that the distance is symmetric, whereas the relation between GO terms is directed. To address this problem they proposed the introduction of hyperbolic cones, wherein a GO term is represented by an apex and an angle of aperture that only depend on the distance to the center. To capture the relationship,

the distance between the apex was not considered, but rather, the inclusion of the apex of the child GO term inside the cones of the parent GO term. Nevertheless, cones present inherent challenges, such as the non-trivial conversion of angles into probabilities associated with different functions. We tested one possible conversion method (detailed below), but other studies opted to directly employ hyperbolic hyperplanes for defining relations. One advantage is that hyperbolic hyperplanes can easily be used to output probability as presented in [Xio+]. Finally, a common setup was used to compare these different models. From the previous literature ([GBH18b],[NK],[Xio+]) multiple ways to sample negative and positive relations in the GO graph are possible. To have a common testbed we choose one specific sample strategy from the different possibilities (see section 3.5.1). In contrast to the metrics utilized in previous studies that introduced different hyperbolic models, the classical metrics for Automatic Function Prediction (AFP) [Zho+19], such as F_{\max} and S_{\min} , and their weighted variants were adopted. It is important to note that some papers ([Xio+],[GBH18b]) evaluate their models for link predictions, where the goal is to identify missing relations when not all relations are available. However, this particular evaluation goes beyond the scope of this PhD. Several examples of embedding can be found in Figure 3.9.

Remarks on the chronological order of experiments:

To clarify my motivation for delving into imperfect methods, it is essential to outline the chronological progression of my experiments. Initially, in the initial phase of my research on this topic, I was unaware of a recent paper that introduced a method for incorporating relation constraints into hyperplanes. While exploring hyperbolic entailment cones, I made several attempts to convert angles into probabilities but encountered significant challenges. Concurrently, I discovered Hyperbolic Logistic Regression (HLR), an extension of logistic regression tailored for hyperbolic space. I aimed to integrate relation constraints into this model but faced numerical instabilities when running the model. Fortunately, I subsequently stumbled upon a recent paper ([Xio+]) presenting a simpler approach to achieving my desired objective. The paper proposed a clever idea, demonstrating the definition of hyperplanes as the intersection between an external ball and the Poincaré ball. This innovative insight greatly facilitated the formulation of loss constraints, surpassing the complexity of my earlier attempts. Armed with this newfound understanding, I pro-

ceeded to re-implement the various methods within a unified framework. This allowed me to establish connections between the different techniques while comprehensively evaluating their respective strengths and weaknesses.

3.5.1 Positive and negative relations and sampling

In order to train different types of neural network models, a graph loss specific to each model is required. However, all models necessitate positive and negative annotations. The positive annotations are extracted from the `is_a` and `part_of` edges in the Gene Ontology (GO) graph. Some approaches ([GBH18a]) also involve computing the transitive closure⁷ of the graph using the selected relations, which introduces correct edges but also redundant ones. In this work, the transitive closure was not computed and the set of possible relations was named E_p (see equation 3.18).

Sampling negative annotations can be more complicated. Negative relations are not directed, unlike positive relations. All possible relations that are not positive can be considered negative, meaning that any pair (i, j) or (j, i) that does not exist in the GO graph, is considered negative. However, this approach is not correct for the classic graph due to the transitivity property. Even if we consider only relations that are not in the transitive closure, there will still be multiple redundant negative relations. To address this redundancy, we adopt the technique proposed by [1]. The technique can be described as follows: we iterate through all GO terms in the graph, for each GO term, we retrieve all his children, compute the list of descendants for each child, and then consider all pairs of children. If two children have no descendants in common, they are annotated as a negative pair. This set of negative annotations will be named $N_{sibling}$ for sibling negative and formally computed as in equation 3.16. This procedure enables us to mine negative annotations without requiring additional expert knowledge, except for the ontology that we already possess.

However, if we aim to model one protein as one point in this space, it can introduce certain problems. Proteins can have multiple functions, so even if the ontology indicates that two GO terms are separate, a protein may possess both functions. Therefore, an

7. The transitive closure adds all relations between GO that can be deduced by the transitivity property of for example `is_a` and `part_of`

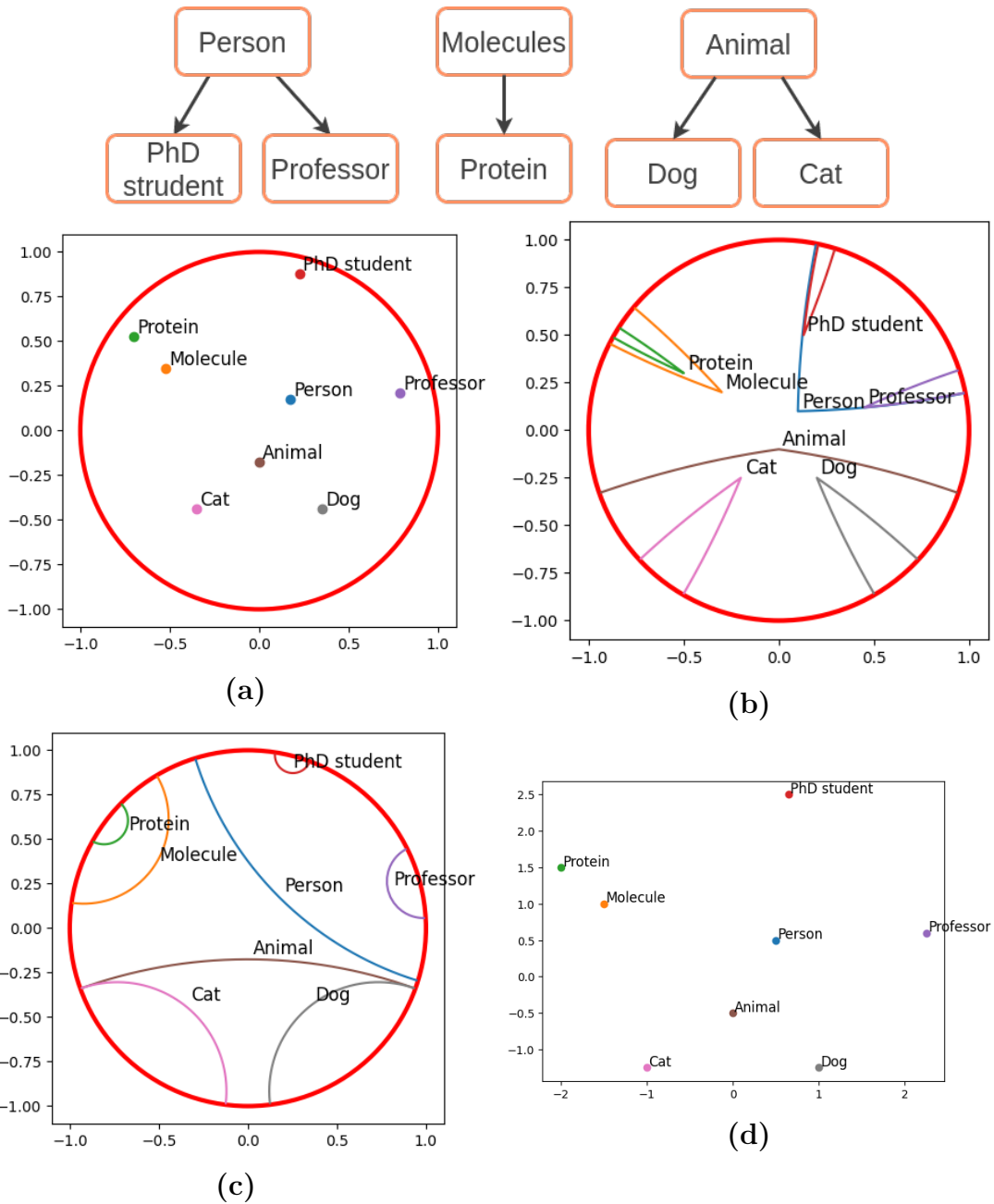


Figure 3.9 – Illustration of the different embedding types in the Poincaré Ball and Euclidean space with toy classes. (a) Hyperbolic Model - Distance-Based Embedding: classes (can be GO terms) are represented by points within the space. (b) Hyperbolic Model - Cone-Based Embedding: classes are represented by cones in the space. (c) Hyperbolic Model - Hyperplane-Based Embedding: classes are represented by hyperplanes in the space. (d) Euclidean Model - Point-Based Embedding: classes are represented by points within a euclidean space.

additional step is necessary to eliminate from the set of negative annotations the functions that occur in the same proteins. To accomplish this, we compute the GO terms that co-occur in the dataset, and when such co-occurrence is detected, the corresponding negative relations are excluded from the negative annotations. The negative set of coocurrence was named $N_{cooccur}$ (see equation 3.17) and the resulting negative set from the two constraints was named E_n (see equation 3.19).

$$G = (V, E) \quad (3.13)$$

$$TC = (V_c, E_c) \quad (3.14)$$

$$S = \{(i, j), \forall i, j \in \{0 \dots |V|\} | \exists p \in V, (p, i) \in E, (p, j) \in E\} \quad (3.15)$$

$$N_{sibling} = \{(i, j), \forall i, j \in S | \nexists p \in V, (i, p) \in E_c, (j, p) \in E_c\} \quad (3.16)$$

$$N_{cooccur} = \{(i, j) \forall i, j \in \{0 \dots |V|\} | \nexists x \in P, (x, y_i) \in \mathcal{D}_p, (x, y_j) \in \mathcal{D}_p\} \quad (3.17)$$

$$E_p = E \quad (3.18)$$

$$E_n = N_{sibling} \cap N_{cooccur} \quad (3.19)$$

With x_i a protein and y_i a GO term, TC is the transitive closure of the graph G , S the set of siblings. \mathcal{D} is the dataset which is a set of all couples (x, g) where x is a protein and g is a GO term, and \bar{S} is the complement of set S . \mathcal{D}_p is the dataset with the addition of the propagated annotations by using the TPR with the relations `is_a` and `part_of`. In this definition, both directions were taken (i, j) and (j, i) because some methods like cones need negative in both directions because the loss is not symmetric.

As shown in Table 3.10, the number of negative annotations significantly varies depending on the strategy employed. This discrepancy impacts the way we sample the relations.

The subsequent question arises: How do we efficiently sample the relations? The efficiency depends on the number of negative annotations. In the case of a very large graph, with approximately 47,514 nodes and 2,257,580,196 possible relations, of which around 99% is negative, we need to consider whether rejection sampling is more efficient. Furthermore, the choice of relations to the sample plays a role. Are true negatives with siblings

more efficient? Should we sample in both directions for unifying the framework? For instance, while the sphere exclusion loss is symmetric, the cones exclusion loss is not (As we will see in subsection 3.5.4).

Strategy	Number of relations
Direct edges	75 552
Edges from transitive closure	575 775

Table 3.9 – Number of positive relations depending on the strategy. Only direct edges present in the GO graph, which are non-redundant or edges from the transitive closure that are redundant.

Strategy	Number of relations
All edges that are not present in the transitive closure	1 128 166 809
Only GO terms that do not co-occur in annotations	1 112 046 151
Only siblings that are not related	926 420
No co-occur and siblings not related	870 954

Table 3.10 – Number of negative relations depending on the strategy. Relations are here considered as not directed.

3.5.2 Common architecture

In this section, the different hyperbolic models and the baseline euclidean model will be presented. The objective is to have all models as equivalent as possible, with the same number of parameters. First, a manifold is an n-dimensional topological space that can be linearly approximated to an n-dimensional real space at any point. Riemannian manifolds are a couple comprising a differentiable manifold and a metric tensor, here named $(\mathbb{D}^n, g_x^{\mathbb{D}})$. Hyperbolic space is a Riemannian manifold with constant negative curvature. More specifically all the following neural network model will operate on the Poincare Ball, that have a constant negative curvature of -1. In the Poincaré ball model, hyperbolic space is represented as a unit ball in Euclidean space. The interior of the ball represents the entire hyperbolic space, while the boundary of the ball represents infinity. Straight lines are

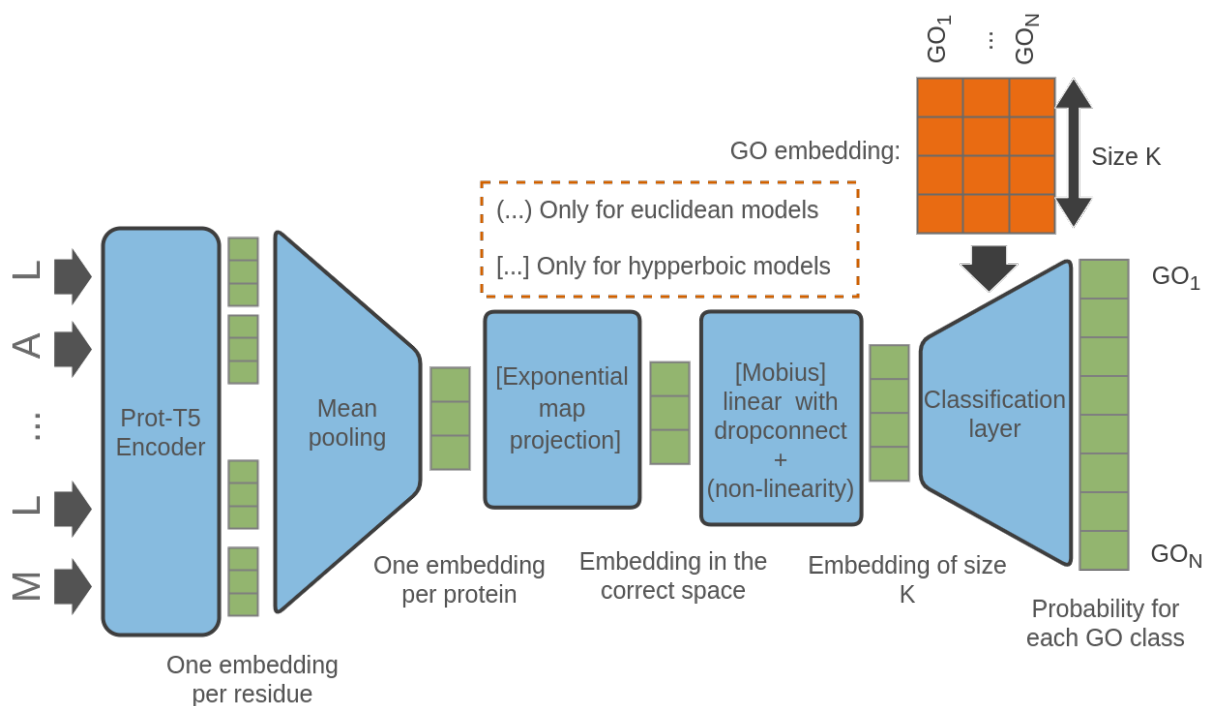


Figure 3.10 – Comprehensive depiction of the universal architecture shared among diverse models, highlighting essential specificities when required. The classification layer comprises the output computation and the logits function. It can be noted that to compute the output and the logits both information from protein embedding and the GO embedding is needed

represented as geodesics, which are curves that minimize the distance between two points. Geodesics in the Poincaré ball model are arcs of circles or straight lines that intersect the boundary of the ball at right angles. The equation for the distance between two points is presented in equation 3.22, where $\|\cdot\|$ is the euclidean norm (Also called L2-norm), and \cosh^{-1} is the inverse hyperbolic cosine also named arccosh, and $g_x^{\mathbb{D}}$ is the Poincare ball metric.

$$\mathbb{D}^n = \{x \in \mathbb{R} : \|x\| < 1\} \quad (3.20)$$

$$d_{\mathbb{D}}(x, y) = \cosh^{-1} \left(1 + 2 \frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)} \right) \quad (3.21)$$

$$g_x^{\mathbb{D}} = \lambda_x^2 g^E, \quad \text{where} \quad \lambda_x := \frac{2}{1 - \|x\|^2}, \text{ and} \quad g^E = I_n \quad (3.22)$$

Another important concept is the concept of tangent space. At any point on a hyperbolic manifold (a surface or space with hyperbolic geometry), the tangent space is a flat, Euclidean space that "tangentially touches" the manifold at that point. It is a linear approximation of the hyperbolic space near that point, capturing the local behavior of the hyperbolic geometry. Some computations are common to all the hyperbolic models and have also a close counterpart for the baseline which is in a euclidean space.

Now we will describe the computation done in all the model variants. Firstly, the input to the neural network is the mean embedding along the sequence dimension obtained from Prot-T5 ([Eln+21a]), a protein embedding model. To incorporate this input into a hyperbolic neural network, it is projected onto the Poincare ball by using the exponential map (this operator allows to project vectors from the euclidean tangent space to the manifold). [Xio+] choose to normalize only if the vector norm is more than one to be on the Poincare ball, but we choose the exponential map to have always the same computation and not have "truncate" the norm only in some cases. This projection results in obtaining one embedding per protein.

$$M \otimes x = \tanh \left(\frac{\|Mx\|}{\|x\|} \tanh^{-1}(\|x\|) \right) \frac{Mx}{\|Mx\|} \quad (3.23)$$

$$x \oplus y = \frac{(1 + 2\langle x, y \rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x, y \rangle + \|x\|^2\|y\|^2} \quad (3.24)$$

For better lisibility, like in the other equations $\|\cdot\|$ is the euclidean norm ($\|\cdot\|_2$).

Next, the representation is transformed from the Prot-T5 embedding size (which is 1024) to the GO embedding dimension using a linear layer followed by a non-linearity (GELU here) for the euclidean model or a möbius linear layer ([GBH18b]) for hyperbolic models without non-linearity here because as stated by [SMH21] the Möbius linear is by construction non-linear. The möbius linear is the equivalent of the linear layer but for the Poincaré ball model, with the use of the gyrovector matrix multiplication (see equation 3.23) and the gyrovector addition (see equation 3.24) as defined by [GBH18b]. This maps the input embeddings to the desired GO embedding dimension, facilitating further computations.

To regularize the neural network during training, DropConnect ([Wan+]) is utilized instead of traditional dropout. DropConnect is a regularization technique that extends the concept of Dropout. While Dropout randomly sets a subset of activations to zero within each layer, DropConnect randomly sets a subset of weights to zero. This means that each unit in a layer receives input from a random subset of units in the previous layer. By applying regularization to the entire connectivity structure of a fully connected neural network layer, DropConnect facilitates the learning of robust representations by relying on different subsets of connections. This regularization technique helps mitigate overfitting and enhances the generalization performance of the network. We derived DropConnect formulation for the classical euclidean model and adapted it for the möbius linear in equations 3.26.

$$\text{DropConnect}_{\text{Euclidean}}(x, M, W, b) = (M \times W) \times x + b \quad (3.25)$$

$$\text{DropConnect}_{\text{Poincaré}}(x, M, W, b) = (M \times W) \otimes x \oplus b \quad (3.26)$$

where M is a binary matrix encoding the connection information and $M_{ij} \sim \text{Bernoulli}(p)$, W is the weight matrix, B is the bias vector and x is the input.

In parallel, for hyperbolic models GO terms are represented by hyperbolic vectors of size GO embedding dim. These vectors will be used for the incorporation of semantic relationships between the GO terms into the neural network. Finally, the embeddings of the GO terms and protein embeddings are utilized to compute probabilities for the different GO classes. The specific computation for probability estimation varies depending on the type of method employed in the neural network. For the euclidean model, a simple linear layer is used to project the number of GO classes. By following this common computation process, the neural networks in this study leverage Prot-T5 embeddings to make predictions and estimate probabilities for various GO classes. A global overview of the common architecture can be viewed in Figure 3.10

3.5.3 Output probability

In this section, the different ways to obtain probabilities from each model will be presented. First, let's define the output of all the neural networks as $\text{Proba}(p, l) = \sigma(O(p, l))$, with p the protein embedding, l a GO term embedding and $O \in \{O_d, O_c, O_h\}$, the three different output O will be presented bellow.

First, the simplest hyperbolic model computes h as the distance between the protein embedding and the GO term embedding as described in equation 3.27.

$$O_d(p, l) = d_{\mathbb{D}}(p, l) \quad (3.27)$$

In hyperbolic entailment cones, the cones are defined by one point, and then the opening angle of the cones is only defined by its distance to the origin o , as shown in equation 3.28. This allows satisfying the four following properties: Axis symmetry, rotation

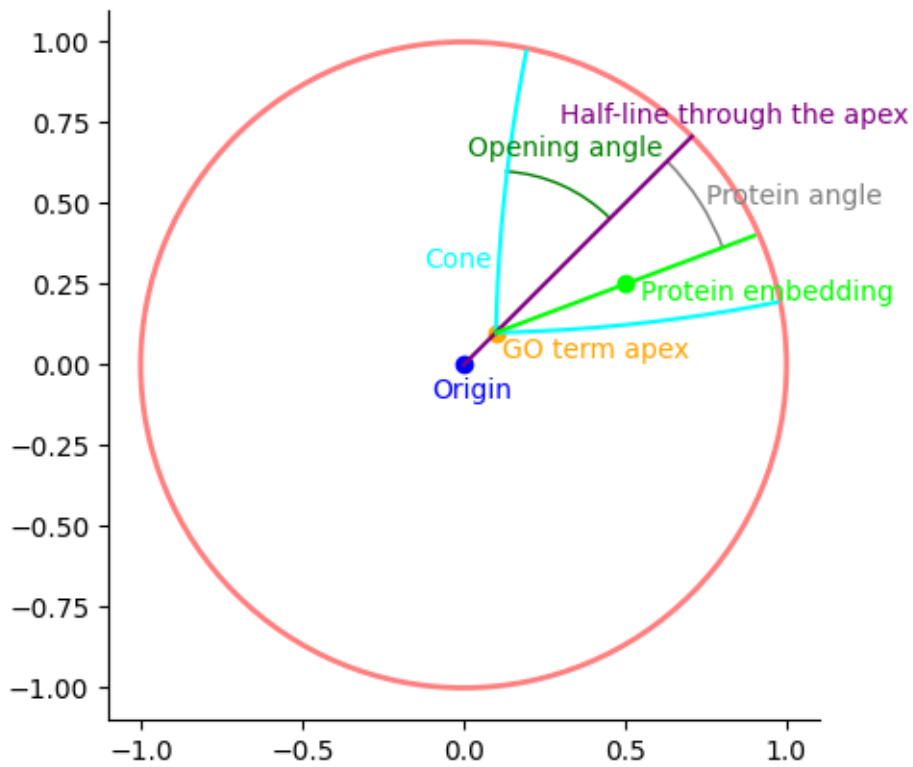


Figure 3.11 – Example of a convex cone in the Poincaré ball representing a GO term, and a projection of a protein. Protein has a high probability to be of the class of the drew GO cone because the protein angle is less than the opening angle of the GO cone.

invariance, continuous cone aperture function and transitivity of nested angular cones, more detail can be found in [GBH18a].

$$\psi(l) = \arcsin(K(1 - \|l\|^2)/\|l\|) \quad (3.28)$$

With K a hyperparameter that will define on which part of the ball the cones are defined, because the domain of this function is $\mathbb{D}^n \setminus B^n(o, \epsilon)$, with epsilon:

$$\epsilon = \frac{2K}{1 + \sqrt{1 + 4K^2}} \quad (3.29)$$

For the hyperbolic entailment cones([GBH18a]), the original hyperbolic entailment cones model doesn't have a way to compute this probability and was only used for the discovery of new GO relations. Then we proposed the following equation:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \exp(x) - 1, & \text{if } x \leq 0 \end{cases} \quad (3.30)$$

$$\Xi(l, p) == \pi - \text{angle}(olp) \quad (3.31)$$

$$O_c(p, l) = \text{ELU}(w_m + 1) * (\psi(l) - \Xi(l, p)) + b \quad (3.32)$$

Where l is the label embedding of a GO term, p is the protein embedding, o represents the origin and $\Xi(l, p)$ represents the angle between the half line of the cone and the half line that passes through the apex of the cone and the protein embedding. The half line of the cone is derived from the line that passes through the origin of the Poincare ball and the apex of the cones, this is shown in Figure 3.11. And $\psi(l)$ represents the opening angle of the cone. This equation uses the fact that if the cones are larger, then they should represent a generic concept and then contribute to a bigger probability to have this GO term.

For the hyperplanes' classifier, the output O_h represents the signed distance between

the hyperplane and a given point. Various parametrizations of hyperplanes have been proposed in previous works ([GBH18b],[SMH21]). One common approach involves selecting a point on the hyperplane and a vector normal to the tangent space at that point. However, this method leads to multiple ways of defining the same hyperplane. To address this issue and ensure uniqueness, Xiong et al. ([Xio+]) suggested setting the point as a unique point on the hyperplane, specifically the center of the hyperplane, which corresponds to the point nearest to the origin of the Poincaré ball. By adopting this definition, protein embeddings can be classified using the following equations:

$$O_h(p, l) = dh(p, l) \tag{3.33}$$

$$dh(p, l) = \sin^{-1} \left\{ \frac{2|\langle (-l) \oplus p, c \rangle|}{(1 + \|(-l) \oplus p\|^2)\|c\|} \right\} \tag{3.34}$$

3.5.4 Loss functions

The optimization problem in this study can be described as follows. The loss function used in neural network optimization consists of two parts: one part for assessing performance and another part for ensuring coherence in the embedding space. In this work, coherence is defined as the extent to which the model prediction aligns with the constraints derived from the Gene Ontology graph. The first loss function utilized is BinaryCrossEntropy. This loss function is commonly employed in binary classification tasks and measures the dissimilarity between predicted and true class labels. It does not explicitly consider the relationships between different Gene Ontology (GO) terms. The second loss function that we will name the constraints loss, is employed in the optimization process and depends on the specific hyperbolic model chosen. The choice of a hyperbolic model affects how the coherence of the embedding space is enforced and can vary between different approaches. A positive(\mathcal{L}_{pr}) and negative(\mathcal{L}_{nr}) relation loss is defined. Then the final objective is to minimize these relations loss and the BCELoss at the same time, with a λ parameter to control the relative importance of these two objectives, this can be seen

in equation 3.35.

$$\min_{\theta, \mathcal{C}} \sum_{(x_n, l_n) \in \mathcal{D}_p} BCELoss(x_n, l_n) + \lambda \left(\sum_{(l_i, l_j) \in E_P} \mathcal{L}_{pr}(l_i, l_j) + \sum_{(l_i, l_j) \in E_N} \mathcal{L}_{nr}(l_i, l_j) \right) \quad (3.35)$$

To optimize the neural network and minimize the loss function, the Riemannian Adam optimizer is used. The Riemannian Adam optimizer is implemented by the Geopt library([KKK20]), which provides functionality for optimization on Riemannian manifolds. This optimizer is tailored to work effectively with hyperbolic models and enables efficient gradient-based updates of the model parameters. By utilizing this optimization framework with the combined loss function and the Riemannian Adam optimizer, the neural network is trained to simultaneously optimize the performance metrics and ensure coherence in the embedding space. Now the details on the loss of the positive(\mathcal{L}_{pr}) and negative(\mathcal{L}_{nr}) relation for each model will be described.

The positive and negative loss for Hyperbolic distance is the following:

$$\mathcal{L}_{pr}(l_i, l_j) = d(l_i, l_j) \quad (3.36)$$

$$\mathcal{L}_{nr}(l_i, l_j) = -d(l_i, l_j) \quad (3.37)$$

The positive and negative loss for Hyperbolic cones is the following:

$$E(l_i, l_j) := \max(0, \Xi(l_i, l_j) - \psi(l_i)) \quad (3.38)$$

$$\mathcal{L}_{pr}(l_i, l_j) = E(l_i, l_j) \mathcal{L}_{nr}(l_i, l_j) = \max(0, \gamma - E(l_i, l_j)) \quad (3.39)$$

With l_i the apex of the cone i and $\psi(l_i)$ is the aperture.

The hyperplane can be viewed as the intersection between the boundary of an n-ball in \mathbb{R}^n and the boundary of the Poincare ball. The following function allows us to compute what would be the center and radius of this n-ball from the center of the hyperplane.

$$origin(l_i) = \frac{1 + \|\mathbf{l}_i\|^2}{2\|\mathbf{l}_i\|} \quad (3.40)$$

$$radius(l_i) = \frac{1 - \|\mathbf{l}_i\|^2}{2\|\mathbf{l}_i\|} \quad (3.41)$$

With these functions, the loss of the positive and negative relation can be defined as follows:

$$\mathcal{L}_{nr}(l_i, l_j) = \max \{0, radius(l_j) + radius(l_i) - \|origin(l_i) - origin(l_j)\|\} \quad (3.42)$$

$$\mathcal{L}_{pr}(l_i, l_j) == \max \{0, \|origin(l_i) - origin(l_j)\| + radius(l_i) - radius(l_j)\} \quad (3.43)$$

In order to maintain consistency with the original nomenclature, the hyperbolic hyper-plane models incorporating the constraint loss were designated as HMI+HLR. Conversely, the identical model lacking the relation constraints was referred to as HLR (Hyperbolic Logistic Regression). Furthermore, to denote the baseline model operating in Euclidean space, a straightforward designation of "Euclidean model" was adopted.

3.5.5 Metric choice

Two groups of metrics were employed in this study. The first group was used to assess the performance of predictions, while the second group was used to evaluate the coherence of the predictions. For the performance metrics, all the metrics presented in the "Common Metrics" section were computed. However, for the initial analysis and to simplify the interpretation of the results, we drew inspiration from CAFA5 and calculated the average of the maximum weighted F1 score for each namespace. These metrics were weighted by information accretion (IA). Weighting the metrics by IA provides a measure of the importance of each namespace in the evaluation.

Regarding the coherence metrics, three metrics were utilized. The term coherence was used to reference if the model output was correctly ordered as the Gene Ontology implies. Two metrics were employed for assessing positive relations, with (see equation 3.46) and without (see equation 3.44) a threshold. The threshold-free metric provides the assurance

that the prediction will maintain coherence for positive relations, regardless of the specific threshold value. Additionally, only a thresholded metric was applied for negative relations (see equation 3.47). This decision was made because converting negative relations to order on probability was not straightforward. Thus, the metric for negative relations was only calculated using a specific threshold. By employing these two groups of metrics, we were able to comprehensively evaluate both the performance and coherence of the predictions.

$$\text{PosEdgeRecover} = \frac{1}{|\mathcal{P}| |E_p|} \sum_{p \in \mathcal{P}} \sum_{(l_i, l_j) \in E_p} \mathbb{1}(O(p, l_j) > O(p, l_i)) \quad (3.44)$$

With $h(x)$ the output of our neural network, E_p the positive edges, E_n the negative edges, and \mathcal{P} the set of all proteins.

$$f(p, l) = \begin{cases} \top, & \text{if } O(p, l) \geq \tau \\ \perp, & \text{else} \end{cases} \quad (3.45)$$

$$\text{IPM}_{\text{pos}} = \frac{1 \times 10^6}{|\mathcal{P}| |E_p|} \sum_{p \in \mathcal{P}} \sum_{(l_i, l_j) \in E_p} \mathbb{1}(f(p, l_i) \wedge \neg f(p, l_j)) \quad (3.46)$$

$$\text{IPM}_{\text{neg}} = \frac{1 \times 10^6}{|\mathcal{P}| |E_n|} \sum_{p \in \mathcal{P}} \sum_{(l_i, l_j) \in E_n} \mathbb{1}(f(p, l_i) \wedge f(p, l_j)) \quad (3.47)$$

With τ a threshold define as hyperparameters for the metric. \top which is True, \perp which is false, and \neg which is not.

All the code is available at <https://gitlab.com/rootNico/tgoa> and uses the Pytorch library([Pas+]) for classical deep learning and the Geopt library([KKK20]) for deep learning in the context of hyperbolic models.

3.5.6 Performance and coherence of the different models

In this experiment, we evaluated the performance of three distinct models, namely Euclidean_LR, HLR (Hyperbolic hyperplane), and *HLR+HMI* (Hyperbolic hyperplane with constraint loss), across three different embedding dimensions: 32, 256, and 512. A

lambda of $1.0e-3$ in the loss function which controls the constraints' loss importance was chosen from some small previous experiments. The best hyperparameters found in previous experiments for the euclidean models were kept. And Figure 3.12 illustrates that for example for the lowest dimension of 32, the two hyperbolic models outperform other models in terms of the WF_{max} metric. However, integrating constraints on the space does not seem to provide any advantage in performance. Moreover, as the embedding dimension increases to 256 and 512, the Euclidean model becomes the best-performing one.

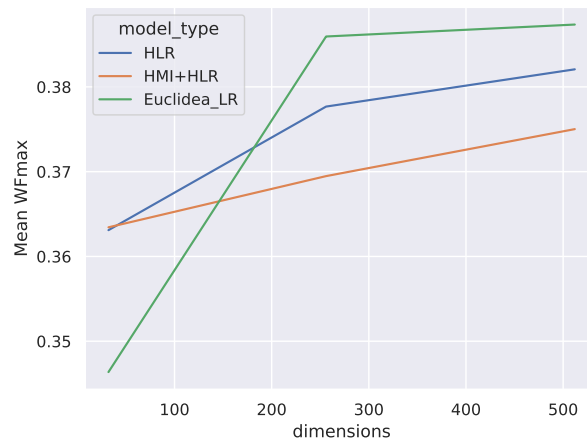


Figure 3.12 – Mean WF_{max} metric across different embedding dimensions

When assessing the coherence of predictions, the HLR+HMI model stands out as the top performer across all dimensions. As depicted in Figure 3.13, the HLR+HMI model correctly orders the probabilities, with the probability of the more general Gene Ontology (GO) terms (parent GO terms) being greater than the probability of specific GO terms (child GO terms) for a majority of instances (99.25% to 99.28%). On the other hand, the HLR and Euclidean models achieve lower PosEdgeRecover in this regard, with values between 96.57% and 96.70%, and between 78.48% and 91.41%, respectively.

Regarding incorrect predictions with a threshold of 0.01, Figures 3.15 and 3.14 demonstrate a maximum of about 3 errors per million evaluated positive relations and less than 6 per million evaluated negative relations for the HLR+HMI model. In comparison, the Euclidean model exhibits significantly more errors, reaching up to 152 for positive relations and 18 for negative relations.

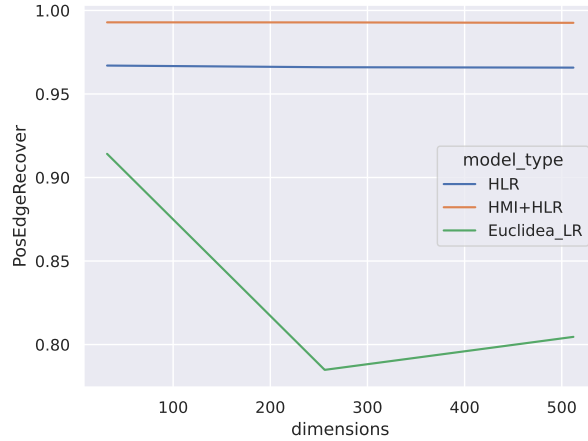


Figure 3.13 – *PosEdgeRecover* with respect to the dimension of embedding for each namespace

Additionally, when analyzing Figure 3.16, which presents the evolution of the *PosEdgeRecover* metric on the validation set during the training of the three models with a dimension of 512, it becomes apparent that the *PosEdgeRecover* performance deteriorates throughout the training process for the Euclidean models. On the contrary, the two hyperbolic models exhibit an initial improvement during training, followed by reaching a stable performance level. This pattern of performance evolution holds true across different dimensions and is indicative of the distinct behavior of the Euclidean models compared to the hyperbolic models.

Overall, these findings highlight the advantages of the HLR+HMI model in terms of coherence and robustness regarding lower-dimensional embeddings, making it a promising approach for further investigations.

A comprehensive analysis of the performance metric for each namespace individually is presented in Figure 3.17 and Figure 3.18. These detailed visualizations enable us to observe that the performance gap in the cellular component namespace is particularly prominent at the lowest dimension compared to other namespaces. However, it is important to note that further replication of the experiment is necessary to validate these findings conclusively.

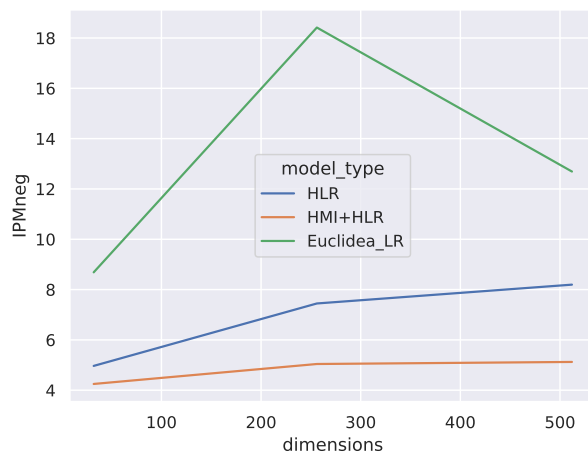


Figure 3.14 – $IPMneg$ with respect to the dimension of embedding for each namespace

3.5.7 Discussion

The current work presented in this study is still ongoing, with several areas left to explore. One potential avenue for further investigation is the use of multiple embeddings per protein. This approach could offer the advantage of having a single function per embedding, creating a functional space that corresponds specifically to the Gene Ontology (GO). However, generating a varying number of embeddings for different proteins would significantly increase the complexity of training the model.

Another aspect worth considering is the limitation of having only one hidden layer in the current model. This design choice may restrict the expressivity of the model and could potentially be addressed by incorporating additional hidden layers.

To ensure the reliability and generalizability of the findings, it is crucial to replicate the experiment and conduct statistically significant tests. Further, experiments with larger sample sizes would strengthen the validity of the results and provide more robust conclusions.

There is room for improvement by refining the search for optimal hyperparameters. For instance, the learning rate was chosen to prevent divergence but may be too small, resulting in slow training. Due to time resources, the model may not have been trained until convergence, particularly for the largest model dimension of 512. This could explain why

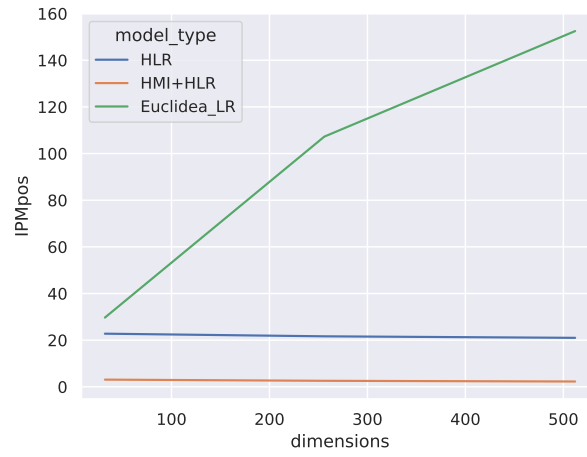


Figure 3.15 – IPM_{pos} with respect to the dimension of embedding for each namespace the hyperbolic models have worse performance than the euclidean one in this dimension.

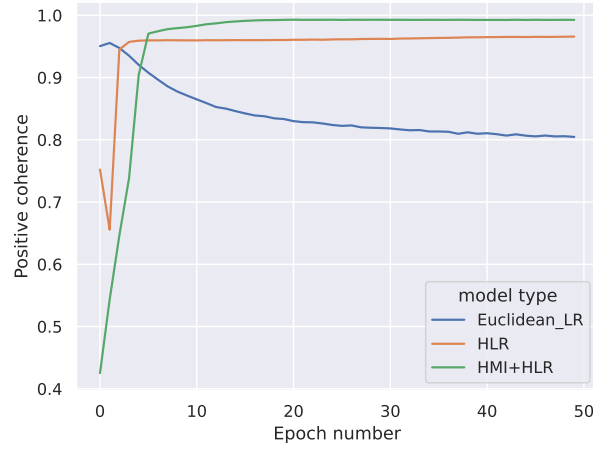


Figure 3.16 – PosEdgeRecover Metric Evolution during Training for Euclidean_LR, HMI, and HMIHLR Models

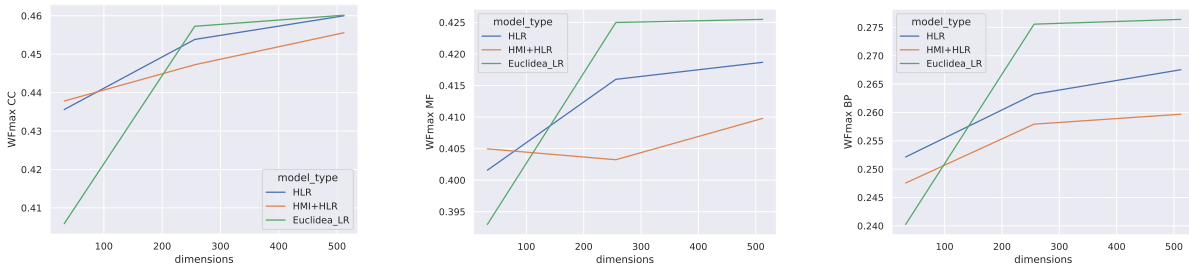


Figure 3.17 – WF_{\max} metric across different embedding dimensions for each namespace

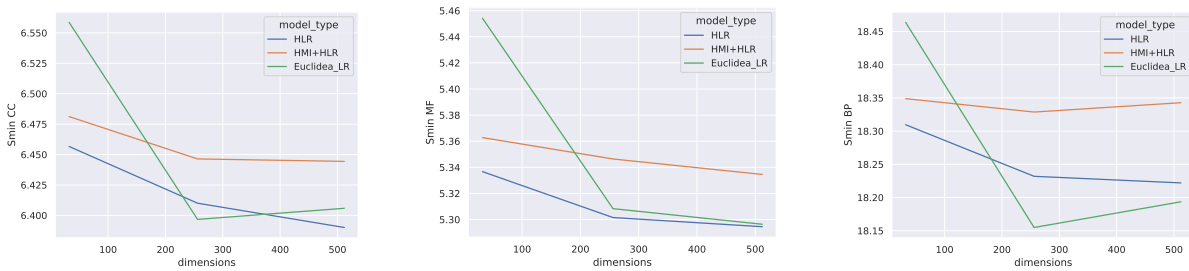


Figure 3.18 – S_{\min} metric across different embedding dimensions for each namespace

CONCLUSION AND PERSPECTIVE

3.6 Conclusion

In the first part of this thesis, a state-of-the-art⁸ model called EnzBert was presented, which uses sequences to predict the functional annotation of enzymes. This confirms the interest of the Transformer of Automatic Function Prediction (AFP). EnzBert uses the Transformer architecture and benefits from the attention mechanism to capture important features such as catalytic residues in the enzyme sequences. This achievement is made possible by the introduction of a simple yet effective interpretability method based on attention maps, which provides insights into the relationship between enzyme sequences and functional annotation. The findings of the research have implications for further understanding how enzyme classes are derived from protein sequences. The interpretability method can assist in uncovering important features and patterns in enzyme sequences, which can contribute to future research in enzyme optimization and related areas.

In the second part of this thesis, the scope was expanded beyond enzymes to include all proteins and the gene ontology vocabulary was used for this broader protein category. In terms of integrating Gene Ontology (GO) into annotations, the study found that the common usage of the True Path Rule did improve performance. However, when using GO to mitigate false negative propagation within the context of the Closed World Assumption (CWA), there was no significant increase in performance observed. Additionally, using GO to constrain the representation of the protein embedding space did not affect performance, but it did lead to improved coherence of predictions. Furthermore, the study compared hyperbolic models to Euclidean models and discovered that hyperbolic models outperformed Euclidean models in small dimensions, but not in larger dimensions. It is important to note that these findings are preliminary, and further research is required to gain a better understanding of the strengths and weaknesses of hyperbolic models.

8. At submission time

3.7 Perspective

Expanding on the findings and implications discussed in this thesis, the perspective section aims to broaden the scope of the investigation and explore new research avenues, beginning with research related to the integration of Gene Ontology (GO) information.

3.7.1 On integrating the Gene Ontology information

This research idea focuses on investigating various aspects related to Gene Ontology (GO), including the impact of different negative relation definitions, extending constraints to other logical constraints arising from relations like `regulate` and `has_part`, and assessing the logical correctness of downloaded GO annotations.

One avenue of exploration involves studying the impact of different negative relation definitions in GO annotations. Negative relations play a crucial role in capturing the absence or negation of specific relationships between entities. By analyzing different definitions and their effects on the quality and performance of GO annotations, researchers can gain insights into the significance of negative relations and their influence on downstream applications such as gene function prediction.

Extending constraints relations to other logical constraints that arise from relations like `regulate` and `has_part` is another aspect worth investigating. The existing constraints in GO annotations provide valuable information for ensuring consistency and coherence. By considering additional logical constraints stemming from other relations, such as the regulation of biological processes or the composition of biological entities, the expressiveness and comprehensiveness of the ontology could be enhanced. This extension could improve the accuracy and utility of GO annotations, enabling more sophisticated analyses and interpretations.

Assessing the logical correctness of downloaded GO annotations with respect to the rules in the Gene Ontology OWL file is another important research direction. GO annotations are typically obtained from databases and are subject to potential errors or inconsistencies. By systematically evaluating the logical correctness of downloaded annotations and comparing them to the established rules and axioms in the GO ontology, researchers can identify discrepancies, highlight potential issues, and contribute to the

refinement and improvement of the annotation curation process.

By pursuing research in these areas, a deeper understanding of negative relations, extended constraints, and the logical correctness of GO annotations can be achieved. This research can contribute to the advancement of knowledge in the field of biological ontologies, improve the quality of GO annotations, and facilitate more reliable and accurate analysis of gene functions and biological processes.

One avenue of exploration involves testing the integration of other data sources in addition to manually reviewed annotations during training. This could include incorporating annotations based on phylogeny(ECO:0000318) or logical inference (ECO:0000364), potentially with different weights assigned to each type of data. By evaluating the impact of integrating diverse data sources, researchers can assess the benefits and challenges associated with incorporating different types of information into the training pipeline. This investigation can provide insights into the potential improvements in gene function prediction and the challenges in reconciling and effectively leveraging heterogeneous data.

Another research direction is to test and compare the different embeddings generated by various pre-trained networks such as Prot-T5, ESM1, and ESM2. Each network offers distinct strategies for representing biological sequences, and comparing their performance can provide insights into their strengths and weaknesses for different downstream tasks in gene function prediction. Such a comparative analysis can inform researchers about the most suitable embedding choice based on the specific requirements of their application. Additionally, exploring ensemble approaches that combine multiple embeddings could be beneficial for harnessing the complementary information provided by different pre-trained networks.

Instead of relying on simple mean pooling of the pre-trained network embeddings, researchers can explore advanced pooling strategies to capture more informative representations from sequence data. This can involve training a one-layer Transformer or a simplified transformer to intelligently pool the embeddings from individual residues into a single representation. By considering more sophisticated pooling techniques, higher-order dependencies could potentially be captured and richer features could be extracted from the sequence data. This exploration can lead to improved performance in gene function prediction tasks by better preserving the nuanced characteristics of the sequences.

Investigating the impact of fine-tuning the entire architecture, rather than just updating the embeddings, is another important research direction. Fine-tuning involves training not only the last layer but also updating the weights of the lower layers during the fine-tuning process. By examining the effects of comprehensive fine-tuning, researchers can assess whether it leads to better generalization and improved performance in gene function prediction tasks. This exploration can provide insights into the extent to which the lower layers of the network contribute to capturing meaningful representations and understanding the hierarchical structure of biological data.

Finally, there is potential merit in exploring the utilization of the N-k layer, rather than the last layer, in the network architecture. [Val+23] suggest the N-3 layer may yield more meaningful representations for biological sequences, researchers can investigate whether leveraging this layer leads to enhanced performance in gene function prediction or related tasks. This exploration can contribute to understanding the optimal layer for capturing informative features in biological sequences and guide the design of more effective neural network architectures.

3.7.2 Research idea for Transformer

Several research directions can be explored in the context of protein language modeling (pLM). This section aims to summarize, unify, and expand upon some fascinating research avenues predominantly investigated in natural language processing (NLP).

Tokenization

Despite its effectiveness, the current Transformer architecture suffers from various limitations. Primarily, the Transformer always employs the same computation units based on the initial tokenization. Consequently, most pLM solely relies on residues as the unit of computation. However, it may seem peculiar not to have embeddings for an entire domain, as this could potentially entail excessive and redundant computations for not sharing information across multiple tokens that could be consolidated into a single token for certain calculations. Numerous solutions can be considered to address this issue, such as dynamic tokenization, which could change the tokenization across different layers of the Transformer and subsequently remap it to the residues when predicting at the residue level. A

similar possible approach that has been explored is the Perceiver-IO ([Jae+22]), which utilizes fictive tokens independent of the real tokens, effectively tackling this challenge.

Pre-training Techniques

To enhance pre-training techniques, incorporating biological knowledge can be advantageous when selecting which residues to mask. By leveraging available information, such as identifying long-range contacts between residues, the masking process can be guided to focus on more biologically relevant positions. This approach has the potential to improve training time by prioritizing the masking of residues that are likely to have significant interactions. For instance, selectively masking a residue when it is in long-range contact with another residue can facilitate the model’s understanding of important molecular interactions during pre-training.

Architectural Enhancements

One potential research direction is exploring the possibility of reducing the number of parameters in the Transformer model by externalizing a portion of the memory. By storing relevant information externally and querying it only when necessary, to optimize the model’s performance while enhancing interpretability. This concept aligns with the Transformer MSA([Rao+21]) paradigm, where the query step can be externalized using tools like mmseq2. Additionally, some NLP applications have used memory-based Transformers, where a separate Transformer creates the memory, and the Transformer embedding is used for querying, resulting in a more flexible querying process. However, the effectiveness of this method compared to mmseq2 in discovering relevant sequences remains uncertain. Despite this uncertainty, investigating the sequences retrieved by such a network and comparing them to those retrieved by mmseq2 could be a compelling avenue of research, shedding light on potential differences and insights gained from each approach.

The attention mechanism is commonly used to integrate information from different residue embeddings. However, alternative methods like MLP-mixer([Tol+21]), KMDPA (from [Rah+21]), Mask attention network (SAN, DMAN, FFN) of [Fan+21] offer additional ways to combine information. Further research is required to gain a deeper understanding of the inductive biases inherent in these various token-mixing approaches.

An important consideration is striking a balance between generality and computational cost. Ideally, we aim to develop a module that is applicable to diverse tasks, yet

mindful of the finite computational resources available. The architectural inductive bias plays a crucial role in achieving efficiency. A good bias is one that the network would have naturally learned if provided with unlimited computational resources. Exploring specific subclasses of functions that are well-suited for particular tasks can potentially lead to improved efficiency. This research direction also intersects with areas such as meta-learning and neural architecture search, which could provide valuable insights and connections. By investigating the inductive biases of different token mixing methods, we can better understand their effectiveness and potential for enhancing the overall performance and efficiency of deep learning models. Furthermore, an alternative approach to address the computational time and incorporate observed biases into the architecture is through connection pruning. By selectively removing certain connections, the model’s computational requirements can be reduced. However, it is important to note that our current hardware resources, such as GPUs or TPUs, may not be well-suited for efficiently performing sparse matrix multiplications required in sparse models. Therefore, finding a balance between model sparsity and hardware limitations becomes a crucial challenge in designing efficient architectures. Exploring techniques that strike a suitable trade-off, considering the computational constraints of our hardware, can be a valuable avenue for further investigation.

Parameters sharing

Some progress has been made in exploring the concept of sharing computational units, drawing parallels to how programmers utilize functions multiple times. For example, Dynamic inference with a neural interpreter has demonstrated the ability to identify reusable functions, leading to parameter sharing. This idea has been also applied in models like ALBERT ([Lan+20]), particularly within the Transformer context. Surprisingly, using the same set of parameters across all layers in ALBERT has not shown a catastrophic effect that might have been anticipated, even on the domain of the protein (ProtAlbert [Eln+21b]).

There also exists a paper on an Adaptive Computation Time ([Gra17]) algorithm. The ACT algorithm is a deterministic and differentiable method that allows recurrent neural networks to dynamically determine the number of computational steps needed between input and output, resulting in improved performance across various tasks, including language modeling, by adapting computation to the complexity of the data. In the same

idea, the Universal Transformer incorporates a dynamic per-position halting mechanism ([Deh+19]), which stops updating the embeddings of tokens that have already reached their desired state, effectively reducing unnecessary computations and optimizing efficiency.

These findings raise important questions regarding the adequacy of parameter count when scaling computation. It prompts us to consider how many parameters are truly necessary when we increase computational resources. Conversely, another intriguing direction involves increasing the number of parameters without a corresponding increase in computational requirements. Approaches like Swich-Transformer ([FZS21]) employ expert models to expand the parameter space, presenting an alternative perspective on model capacity. These lines of inquiry open up exciting possibilities for further research and investigation into the optimal balance between parameter count, computation, and model performance. It is essential to explore these trade-offs to gain a deeper understanding of the dynamics between model size, computational efficiency, and the associated gains in performance.

BIBLIOGRAPHY

- [22a] *Peptide bond*, en, Page Version ID: 1125449075, Dec. 2022, URL: https://en.wikipedia.org/w/index.php?title=Peptide_bond&oldid=1125449075 (visited on 03/28/2023).
- [22b] *Ramachandran plot*, en, Page Version ID: 1109592042, Sept. 2022, URL: https://en.wikipedia.org/w/index.php?title=Ramachandran_plot&oldid=1109592042 (visited on 03/28/2023).
- [23a] *Acide aminé*, fr, Page Version ID: 201965361, Mar. 2023, URL: https://fr.wikipedia.org/w/index.php?title=Acide_amin%C3%A9&oldid=201965361 (visited on 03/23/2023).
- [23b] *Collagen*, en, Page Version ID: 1147973436, Apr. 2023, URL: <https://en.wikipedia.org/w/index.php?title=Collagen&oldid=1147973436> (visited on 04/11/2023).
- [23c] *Protein structure*, en, Page Version ID: 1135729401, Jan. 2023, URL: https://en.wikipedia.org/w/index.php?title=Protein_structure&oldid=1135729401 (visited on 04/11/2023).
- [AZ20] Samira Abnar and Willem Zuidema, « Quantifying Attention Flow in Transformers », en, *in: arXiv:2005.00928 [cs]* (May 2020), arXiv: 2005.00928, URL: <http://arxiv.org/abs/2005.00928> (visited on 08/23/2021).
- [Ber+00] Helen M. Berman et al., « The Protein Data Bank », *in: Nucleic Acids Research 28.1* (Jan. 2000), pp. 235–242, ISSN: 0305-1048, DOI: 10.1093/nar/28.1.235, URL: <https://doi.org/10.1093/nar/28.1.235> (visited on 04/11/2023).

-
- [Bil+19] Maxwell L. Bileschi et al., « Using Deep Learning to Annotate the Protein Universe », en, in: *bioRxiv* (May 2019), Publisher: Cold Spring Harbor Laboratory Section: New Results, p. 626507, DOI: 10.1101/626507, URL: <https://www.biorxiv.org/content/10.1101/626507v3> (visited on 06/17/2021).
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton, *Layer Normalization*, arXiv:1607.06450 [cs, stat], July 2016, DOI: 10.48550/arXiv.1607.06450, URL: <http://arxiv.org/abs/1607.06450> (visited on 05/05/2023).
- [Bou+21] Victoria Bourgeais et al., « Deep GONet: self-explainable deep neural network based on Gene Ontology for phenotype prediction from gene expression data », in: *BMC Bioinformatics* 22.10 (Sept. 2021), p. 455, ISSN: 1471-2105, DOI: 10.1186/s12859-021-04370-7, (visited on 05/30/2023).
- [CG12] Carroll Croarkin and Will Guthrie, *NIST/SEMATECH e-Handbook of Statistical Methods*, NIST/SEMATECH e-Handbook of Statistical Methods, Apr. 1, 2012, URL: <https://www.itl.nist.gov/div898/handbook/index.htm> (visited on 07/16/2023).
- [CGW21] Hila Chefer, Shir Gur, and Lior Wolf, « Transformer Interpretability Beyond Attention Visualization », en, in: *arXiv:2012.09838 [cs]* (Apr. 2021), arXiv: 2012.09838, URL: <http://arxiv.org/abs/2012.09838> (visited on 08/23/2021).
- [Cha+22] John-Marc Chandonia et al., « SCOPe: improvements to the structural classification of proteins – extended database to facilitate variant interpretation and machine learning », in: *Nucleic Acids Research* 50.D1 (Jan. 2022), pp. D553–D559, ISSN: 0305-1048, DOI: 10.1093/nar/gkab1054, URL: <https://doi.org/10.1093/nar/gkab1054> (visited on 04/11/2023).
- [Cou16] Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, URL: <https://www.deeplearningbook.org/> (visited on 05/05/2023).

-
- [CR13] Wyatt T. Clark and Predrag Radivojac, « Information-theoretic evaluation of predicted ontological annotations », *in: Bioinformatics* 29.13 (July 2013), pp. i53–i61, ISSN: 1367-4803, DOI: 10.1093/bioinformatics/btt228, URL: <https://doi.org/10.1093/bioinformatics/btt228> (visited on 10/06/2022).
- [Cyb] G Cybenkot, « Approximation by superpositions of a sigmoidal function », en, *in: ()*.
- [Dal+18] Alperen Dalkiran et al., « ECPred: a tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature », en, *in: BMC Bioinformatics* 19.1 (Dec. 2018), p. 334, ISSN: 1471-2105, DOI: 10.1186/s12859-018-2368-y, URL: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2368-y> (visited on 07/18/2021).
- [De +18] Christopher De Sa et al., *Representation Tradeoffs for Hyperbolic Embeddings*, en, arXiv:1804.03329 [cs, stat], Apr. 2018, URL: <http://arxiv.org/abs/1804.03329> (visited on 06/20/2023).
- [Deh+19] Mostafa Dehghani et al., *Universal Transformers*, arXiv:1807.03819 [cs, stat], Mar. 2019, DOI: 10.48550/arXiv.1807.03819, URL: <http://arxiv.org/abs/1807.03819> (visited on 07/17/2023).
- [Dev+19] Jacob Devlin et al., « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding », en, *in: arXiv:1810.04805 [cs]* (May 2019), arXiv: 1810.04805, URL: <http://arxiv.org/abs/1810.04805> (visited on 12/22/2021).
- [DeY+20] Jay DeYoung et al., *ERASER: A Benchmark to Evaluate Rationalized NLP Models*, en, Number: arXiv:1911.03429 arXiv:1911.03429 [cs], Apr. 2020, URL: <http://arxiv.org/abs/1911.03429> (visited on 05/25/2022).
- [Eln+21a] Ahmed Elnaggar et al., « ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing », en, *in: IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1, ISSN: 0162-8828, 2160-9292, 1939-3539, DOI: 10.1109/

-
- TPAMI . 2021 . 3095381, URL: <https://ieeexplore.ieee.org/document/9477085/> (visited on 07/18/2021).
- [Eln+21b] Ahmed Elnaggar et al., « ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing », en, in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1, ISSN: 0162-8828, 2160-9292, 1939-3539, DOI: 10.1109/TPAMI.2021.3095381, URL: <https://ieeexplore.ieee.org/document/9477085/> (visited on 07/18/2021).
- [EMS22] Alejandro A Edera, Diego H Milone, and Georgina Stegmayer, « Anc2vec: embedding gene ontology terms by preserving ancestors relationships », in: *Briefings in Bioinformatics* 23.2 (Mar. 2022), bbac003, ISSN: 1477-4054, DOI: 10.1093/bib/bbac003, URL: <https://doi.org/10.1093/bib/bbac003> (visited on 04/07/2023).
- [Fan+21] Zhihao Fan et al., « Mask Attention Networks: Rethinking and Strengthen Transformer », en, in: *arXiv:2103.13597 [cs]* (Mar. 2021), arXiv: 2103.13597, URL: <http://arxiv.org/abs/2103.13597> (visited on 03/17/2022).
- [Fin+08] Robert D. Finn et al., « The Pfam protein families database », in: *Nucleic Acids Research* 36.Database issue (Jan. 2008), pp. D281–D288, ISSN: 0305-1048, DOI: 10.1093/nar/gkm960, URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2238907/> (visited on 04/11/2023).
- [Fio+18] Diego Fioravanti et al., « Phylogenetic convolutional neural networks in metagenomics », en, in: *BMC Bioinformatics* 19.2 (Mar. 2018), p. 49, ISSN: 1471-2105, DOI: 10.1186/s12859-018-2033-5, URL: <https://doi.org/10.1186/s12859-018-2033-5> (visited on 07/17/2023).
- [FK15] Peter A. Flach and Meelis Kull, « Precision-Recall-Gain curves: PR analysis done right », in: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’15, Cambridge, MA, USA: MIT Press, Dec. 2015, pp. 838–846, (visited on 06/13/2022).

-
- [FZS21] William Fedus, Barret Zoph, and Noam Shazeer, « Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity », en, *in: arXiv:2101.03961 [cs]* (Jan. 2021), arXiv: 2101.03961, URL: <http://arxiv.org/abs/2101.03961> (visited on 08/23/2021).
- [GBH18a] Octavian-Eugen Ganeva, Gary Bécigneul, and Thomas Hofmann, *Hyperbolic Entailment Cones for Learning Hierarchical Embeddings*, en, Number: arXiv:1804.01882 arXiv:1804.01882 [cs, stat], June 2018, URL: <http://arxiv.org/abs/1804.01882> (visited on 06/24/2022).
- [GBH18b] Octavian-Eugen Ganeva, Gary Bécigneul, and Thomas Hofmann, *Hyperbolic Neural Networks*, en, Number: arXiv:1805.09112 arXiv:1805.09112 [cs, stat], June 2018, URL: <http://arxiv.org/abs/1805.09112> (visited on 06/22/2022).
- [Gol+16] Michael Golden et al., « A Generative Angular Model of Protein Structure Evolution », *in: Molecular biology and evolution* 34 (Dec. 2016), DOI: 10.1093/molbev/msx137.
- [Gra17] Alex Graves, *Adaptive Computation Time for Recurrent Neural Networks*, arXiv:1603.08983 [cs], Feb. 2017, DOI: 10.48550/arXiv.1603.08983, URL: <http://arxiv.org/abs/1603.08983> (visited on 07/17/2023).
- [He+15] Kaiming He et al., *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015, DOI: 10.48550/arXiv.1512.03385, URL: <http://arxiv.org/abs/1512.03385> (visited on 05/05/2023).
- [HR18] Jeremy Howard and Sebastian Ruder, *Universal Language Model Fine-tuning for Text Classification*, en, arXiv:1801.06146 [cs, stat], May 2018, URL: <http://arxiv.org/abs/1801.06146> (visited on 04/28/2023).
- [Hun+15] Rachael P. Huntley et al., « The GOA database: gene Ontology annotation updates for 2015 », eng, *in: Nucleic Acids Research* 43.Database issue (Jan. 2015), pp. D1057–1063, ISSN: 1362-4962, DOI: 10.1093/nar/gku1113.

-
- [Jae+22] Andrew Jaegle et al., *Perceiver IO: A General Architecture for Structured Inputs & Outputs*, arXiv:2107.14795 [cs, eess], Mar. 2022, DOI: 10.48550/arXiv.2107.14795, URL: <http://arxiv.org/abs/2107.14795> (visited on 07/17/2023).
- [Jia+14] Yuxiang Jiang et al., « The impact of incomplete knowledge on the evaluation of protein function prediction: a structured-output learning perspective », *in: Bioinformatics* 30.17 (Sept. 2014), pp. i609–i616, ISSN: 1367-4803, DOI: 10.1093/bioinformatics/btu472, URL: <https://doi.org/10.1093/bioinformatics/btu472> (visited on 05/02/2023).
- [Jia+17] Qian Jiang et al., « Protein secondary structure prediction: A survey of the state of the art », *en, in: Journal of Molecular Graphics and Modelling* 76 (Sept. 2017), pp. 379–402, ISSN: 1093-3263, DOI: 10.1016/j.jm gm.2017.07.015, URL: <https://www.sciencedirect.com/science/article/pii/S1093326317304217> (visited on 04/11/2023).
- [Jum+21] John Jumper et al., « Highly accurate protein structure prediction with AlphaFold », *en, in: Nature* 596.7873 (Aug. 2021), pp. 583–589, ISSN: 0028-0836, 1476-4687, DOI: 10.1038/s41586-021-03819-2, URL: <https://www.nature.com/articles/s41586-021-03819-2> (visited on 08/26/2022).
- [KB17] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017, DOI: 10.48550/arXiv.1412.6980, URL: <http://arxiv.org/abs/1412.6980> (visited on 05/05/2023).
- [KH19] Maxat Kulmanov and Robert Hoehndorf, « DeepGOPlus: improved protein function prediction from sequence », *en, in: Bioinformatics* (July 2019), ed. by Lenore Cowen, btz595, ISSN: 1367-4803, 1460-2059, DOI: 10.1093/bioinformatics/btz595, URL: <https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btz595/5539866> (visited on 07/18/2021).
- [KHL21] Guolin Ke, Di He, and Tie-Yan Liu, « Rethinking Positional Encoding in Language Pre-training », *en, in: arXiv:2006.15595 [cs]* (Mar. 2021), arXiv: 2006.15595, URL: <http://arxiv.org/abs/2006.15595> (visited on 03/14/2022).

-
- [KKH18] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf, « DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier », *in: Bioinformatics* 34.4 (Feb. 2018), pp. 660–668, ISSN: 1367-4803, DOI: 10.1093/bioinformatics/btx624, URL: <https://doi.org/10.1093/bioinformatics/btx624> (visited on 01/24/2023).
- [KKK20] Max Kochurov, Rasul Karimov, and Serge Kozlukov, *Geopt: Riemannian Optimization in PyTorch*, arXiv:2005.02819 [cs], July 2020, DOI: 10.48550/arXiv.2005.02819, URL: <http://arxiv.org/abs/2005.02819> (visited on 07/16/2023).
- [KL94] J. F. Kennedy and L. L. Lloyd, « Enzyme nomenclature — Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology: Academic Press Ltd, London, UK, 1992. xiii + 862 pp. Price £40.00. ISBN 0-12-227165-3 », *en, in: Carbohydrate Polymers* 23.4 (Jan. 1994), pp. 291–292, ISSN: 0144-8617, DOI: 10.1016/0144-8617(94)90194-5, URL: <https://www.sciencedirect.com/science/article/pii/0144861794901945> (visited on 10/28/2022).
- [KLA12] Arun S. Konagurthu, Arthur M. Lesk, and Lloyd Allison, « Minimum message length inference of secondary structure from protein coordinate data », *in: Bioinformatics* 28.12 (June 2012), pp. i97–i105, ISSN: 1367-4803, DOI: 10.1093/bioinformatics/bts223, URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3371855/> (visited on 04/11/2023).
- [KS12] Narendra Kumar and Jeffrey Skolnick, « EFICAz2.5: application of a high-precision enzyme function predictor to 396 proteomes », *eng, in: Bioinformatics (Oxford, England)* 28.20 (Oct. 2012), pp. 2687–2688, ISSN: 1367-4811, DOI: 10.1093/bioinformatics/bts510.
- [KS83] Wolfgang Kabsch and Christian Sander, « Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features », *en, in: Biopolymers* 22.12 (1983), *_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bip.360221211*, pp. 2577–2637, ISSN: 1097-0282, DOI: 10.1002/bip.360221211, URL: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/bip.360221211>

[//onlinelibrary.wiley.com/doi/abs/10.1002/bip.360221211](https://onlinelibrary.wiley.com/doi/abs/10.1002/bip.360221211) (visited on 04/11/2023).

- [Lan+20] Zhenzhong Lan et al., « ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS », en, *in*: (2020), p. 17.
- [Lég+22] Corentin Léger et al., « Dynamics and structural changes of calmodulin upon interaction with the antagonist calmidazolium », en, *in*: *BMC Biology* 20.1 (Aug. 2022), p. 176, ISSN: 1741-7007, DOI: 10.1186/s12915-022-01381-5, URL: <https://bmcbiol.biomedcentral.com/articles/10.1186/s12915-022-01381-5> (visited on 04/11/2023).
- [Li+18] Yu Li et al., « DEEPre: sequence-based enzyme EC number prediction by deep learning », en, *in*: *Bioinformatics* 34.5 (Mar. 2018), ed. by John Hancock, pp. 760–769, ISSN: 1367-4803, 1460-2059, DOI: 10.1093/bioinformatics/btx680, URL: <https://academic.oup.com/bioinformatics/article/34/5/760/4562505> (visited on 07/18/2021).
- [Ma+18] Jianzhu Ma et al., « Using deep learning to model the hierarchical structure and function of a cell », *in*: *Nature Methods* 15.4 (Apr. 2018), Number: 4 Publisher: Nature Publishing Group, pp. 290–298, ISSN: 1548-7105, DOI: 10.1038/nmeth.4627, URL: <https://www.nature.com/articles/nmeth.4627> (visited on 11/09/2023).
- [Mik+13] Tomas Mikolov et al., *Efficient Estimation of Word Representations in Vector Space*, arXiv:1301.3781 [cs], Sept. 2013, DOI: 10.48550/arXiv.1301.3781, URL: <http://arxiv.org/abs/1301.3781> (visited on 05/05/2023).
- [MKS17] Stephen Merity, Nitish Shirish Keskar, and Richard Socher, « Regularizing and Optimizing LSTM Language Models », en, *in*: *arXiv:1708.02182 [cs]* (Aug. 2017), arXiv: 1708.02182, URL: <http://arxiv.org/abs/1708.02182> (visited on 03/24/2022).
- [NK] Maximillian Nickel and Douwe Kiela, « Poincaré Embeddings for Learning Hierarchical Representations », en, *in*: ().
- [Pas+] Adam Paszke et al., « Automatic differentiation in PyTorch », en, *in*: ().

-
- [Pet+18] Matthew E. Peters et al., *Deep contextualized word representations*, arXiv:1802.05365 [cs], Mar. 2018, DOI: 10.48550/arXiv.1802.05365, URL: <http://arxiv.org/abs/1802.05365> (visited on 05/05/2023).
- [PG17] Sylvain Poux and Pascale Gaudet, « Best Practices in Manual Annotation with the Gene Ontology », en, in: *The Gene Ontology Handbook*, ed. by Christophe Dessimoz and Nives Škunca, Methods in Molecular Biology, New York, NY: Springer, 2017, pp. 41–54, ISBN: 978-1-4939-3743-1, DOI: 10.1007/978-1-4939-3743-1_4, URL: https://doi.org/10.1007/978-1-4939-3743-1_4 (visited on 07/05/2023).
- [PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning, « Glove: Global Vectors for Word Representation », en, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543, DOI: 10.3115/v1/D14-1162, URL: <http://aclweb.org/anthology/D14-1162> (visited on 07/12/2023).
- [Rad+] Alec Radford et al., « Improving Language Understanding by Generative Pre-Training », en, in: ().
- [Raf+20] Colin Raffel et al., « Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer », en, in: *arXiv:1910.10683 [cs, stat]* (July 2020), arXiv: 1910.10683, URL: <http://arxiv.org/abs/1910.10683> (visited on 03/21/2022).
- [Rah+21] Nasim Rahaman et al., « Dynamic Inference with Neural Interpreters », en, in: *arXiv:2110.06399 [cs]* (Oct. 2021), arXiv: 2110.06399, URL: <http://arxiv.org/abs/2110.06399> (visited on 03/11/2022).
- [Rao+19] Roshan Rao et al., *Evaluating Protein Transfer Learning with TAPE*, en, preprint, Synthetic Biology, June 2019, DOI: 10.1101/676825, URL: <http://biorxiv.org/lookup/doi/10.1101/676825> (visited on 06/17/2021).

-
- [Rao+20] Roshan Rao et al., *Transformer protein language models are unsupervised structure learners*, en, preprint, Synthetic Biology, Dec. 2020, DOI: 10.1101/2020.12.15.422761, URL: <http://biorxiv.org/lookup/doi/10.1101/2020.12.15.422761> (visited on 07/18/2021).
- [Rao+21] Roshan Rao et al., « MSA Transformer », en, *in: bioRxiv* (Feb. 2021), Publisher: Cold Spring Harbor Laboratory Section: New Results, p. 2021.02.12.430858, DOI: 10.1101/2021.02.12.430858, URL: <https://www.biorxiv.org/content/10.1101/2021.02.12.430858v1> (visited on 06/17/2021).
- [RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, « Learning representations by back-propagating errors », en, *in: Nature* 323.6088 (Oct. 1986), Number: 6088 Publisher: Nature Publishing Group, pp. 533–536, ISSN: 1476-4687, DOI: 10.1038/323533a0, URL: <https://www.nature.com/articles/323533a0> (visited on 05/05/2023).
- [Rib+18] António J M Ribeiro et al., « Mechanism and Catalytic Site Atlas (M-CSA): a database of enzyme reaction mechanisms and active sites », *in: Nucleic Acids Research* 46.D1 (Jan. 2018), pp. D618–D623, ISSN: 0305-1048, DOI: 10.1093/nar/gkx1012, URL: <https://doi.org/10.1093/nar/gkx1012> (visited on 12/08/2021).
- [Riv+21] Alexander Rives et al., « Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences », en, *in: Proceedings of the National Academy of Sciences* 118.15 (Apr. 2021), e2016239118, ISSN: 0027-8424, 1091-6490, DOI: 10.1073/pnas.2016239118, URL: <http://www.pnas.org/lookup/doi/10.1073/pnas.2016239118> (visited on 12/22/2021).
- [RM51] Herbert Robbins and Sutton Monroe, « A Stochastic Approximation Method », *in: The Annals of Mathematical Statistics* 22.3 (Sept. 1951), Publisher: Institute of Mathematical Statistics, pp. 400–407, ISSN: 0003-4851, 2168-8990, DOI: 10.1214/aoms/1177729586, URL: <https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-3/>

-
- A-Stochastic-Approximation-Method/10.1214/aoms/1177729586.full (visited on 05/05/2023).
- [RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, « "Why Should I Trust You?": Explaining the Predictions of Any Classifier », *in: arXiv:1602.04938 [cs, stat]* (Aug. 2016), arXiv: 1602.04938, URL: <http://arxiv.org/abs/1602.04938> (visited on 12/22/2021).
- [SC07] Hong-Bin Shen and Kuo-Chen Chou, « EzyPred: A top-down approach for predicting enzyme functional classes and subclasses », *en, in: Biochemical and Biophysical Research Communications* 364.1 (Dec. 2007), pp. 53–59, ISSN: 0006-291X, DOI: 10.1016/j.bbrc.2007.09.098, URL: <https://www.sciencedirect.com/science/article/pii/S0006291X07020864> (visited on 09/02/2022).
- [Shr+17] Avanti Shrikumar et al., « Not Just a Black Box: Learning Important Features Through Propagating Activation Differences », *in: arXiv:1605.01713 [cs]* (Apr. 2017), arXiv: 1605.01713, URL: <http://arxiv.org/abs/1605.01713> (visited on 12/22/2021).
- [SMH21] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada, *Hyperbolic Neural Networks++*, *en, arXiv:2006.08210 [cs, stat]*, Mar. 2021, URL: <http://arxiv.org/abs/2006.08210> (visited on 03/10/2023).
- [SP97] M. Schuster and K.K. Paliwal, « Bidirectional recurrent neural networks », *in: IEEE Transactions on Signal Processing* 45.11 (Nov. 1997), Conference Name: IEEE Transactions on Signal Processing, pp. 2673–2681, ISSN: 1941-0476, DOI: 10.1109/78.650093.
- [SS17] Martin Steinegger and Johannes Söding, « MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets », *en, in: Nature Biotechnology* 35.11 (Nov. 2017), Bandiera_abtest: a Cg_type: Nature Research Journals Number: 11 Primary_atype: Correspondence Publisher: Nature Publishing Group Subject_term: Environmental microbiology;Functional clustering;Sequence annotation;Software Subject_term_id: environmental-microbiology;functional-clustering;sequence-annotation;software, pp. 1026–1028,

ISSN: 1546-1696, DOI: 10.1038/nbt.3988, URL: <https://www.nature.com/articles/nbt.3988> (visited on 12/22/2021).

- [SS18] Martin Steinegger and Johannes Söding, « Clustering huge protein sequence sets in linear time », en, *in: Nature Communications* 9.1 (Dec. 2018), p. 2542, ISSN: 2041-1723, DOI: 10.1038/s41467-018-04964-5, URL: <http://www.nature.com/articles/s41467-018-04964-5> (visited on 07/18/2021).
- [Str+20] Nils Strodthoff et al., « UDSMProt: universal deep sequence models for protein classification », *in: Bioinformatics* 36.8 (Apr. 2020), pp. 2401–2409, ISSN: 1367-4803, DOI: 10.1093/bioinformatics/btaa003, URL: <https://doi.org/10.1093/bioinformatics/btaa003> (visited on 06/17/2021).
- [STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan, « Axiomatic Attribution for Deep Networks », *in: arXiv:1703.01365 [cs]* (June 2017), arXiv: 1703.01365, URL: <http://arxiv.org/abs/1703.01365> (visited on 12/22/2021).
- [Tay86] W. R. Taylor, « The classification of amino acid conservation », eng, *in: Journal of Theoretical Biology* 119.2 (Mar. 1986), pp. 205–218, ISSN: 0022-5193, DOI: 10.1016/s0022-5193(86)80075-3.
- [The19] The UniProt Consortium, « UniProt: a worldwide hub of protein knowledge », en, *in: Nucleic Acids Research* 47.D1 (Jan. 2019), pp. D506–D515, ISSN: 0305-1048, 1362-4962, DOI: 10.1093/nar/gky1049, URL: <https://academic.oup.com/nar/article/47/D1/D506/5160987> (visited on 07/18/2021).
- [The21] The UniProt Consortium, « UniProt: the universal protein knowledgebase in 2021 », *in: Nucleic Acids Research* 49.D1 (Jan. 2021), pp. D480–D489, ISSN: 0305-1048, DOI: 10.1093/nar/gkaa1100, URL: <https://doi.org/10.1093/nar/gkaa1100> (visited on 12/22/2021).
- [Tho+03] Paul D. Thomas et al., « PANTHER: A Library of Protein Families and Subfamilies Indexed by Function », *in: Genome Research* 13.9 (Sept. 2003), pp. 2129–2141, ISSN: 1088-9051, DOI: 10.1101/gr.772403, URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC403709/> (visited on 07/05/2023).

-
- [Tol+21] Ilya Tolstikhin et al., *MLP-Mixer: An all-MLP Architecture for Vision*, arXiv:2105.01601 [cs], June 2021, DOI: 10.48550/arXiv.2105.01601, URL: <http://arxiv.org/abs/2105.01601> (visited on 07/17/2023).
- [Urr+16] Lisa A. Urry et al., *Campbell Biology*, Pearson, 2016.
- [Val+23] Lucrezia Valeriani et al., *The geometry of hidden representations of large transformer models*, en, arXiv:2302.00294 [cs, stat], Feb. 2023, URL: <http://arxiv.org/abs/2302.00294> (visited on 04/20/2023).
- [Vas+17] Ashish Vaswani et al., « Attention Is All You Need », *in: arXiv:1706.03762 [cs]* (Dec. 2017), arXiv: 1706.03762, URL: <http://arxiv.org/abs/1706.03762> (visited on 12/22/2021).
- [Vig+20] Jesse Vig et al., *BERTology Meets Biology: Interpreting Attention in Protein Language Models*, en, preprint, Bioinformatics, June 2020, DOI: 10.1101/2020.06.26.174417, URL: <http://biorxiv.org/lookup/doi/10.1101/2020.06.26.174417> (visited on 06/18/2021).
- [Wan+] Li Wan et al., « Regularization of Neural Networks using DropConnect », en, *in: ()*.
- [Xio+] Bo Xiong et al., « Hyperbolic Embedding Inference for Structured Multi-Label Prediction », en, *in: ()*.
- [ZG07] Yan Zhang and Vadim N. Gladyshev, « High content of proteins containing 21st and 22nd amino acids, selenocysteine and pyrrolysine, in a symbiotic deltaproteobacterium of gutless worm *Olavius algarvensis* », *in: Nucleic Acids Research* 35.15 (Aug. 2007), pp. 4952–4963, ISSN: 0305-1048, DOI: 10.1093/nar/gkm514, URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1976440/> (visited on 07/05/2023).
- [Zho+19] Naihui Zhou et al., « The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens », *in: Genome Biology* 20.1 (Nov. 2019), p. 244, ISSN: 1474-760X, DOI: 10.1186/s13059-019-1835-8, URL: <https://doi.org/10.1186/s13059-019-1835-8> (visited on 02/09/2022).

APPENDIX

GO Annotation File (GAF) fields:

Column	Content	Required?	Cardinality	Example
1	DB	required	1	UniProtKB
2	DB Object ID	required	1	P12345
3	DB Object Symbol	required	1	PHO3
4	Qualifier	required	1 or 2	NOT involved_in
5	GO ID	required	1	GO:0003993
6	DB:Reference (DB:Reference)	required	1 or greater	PMID:2676709
7	Evidence Code	required	1	IMP
8	With (or) From	optional	0 or greater	GO:0000346
9	Aspect	required	1	F
10	DB Object Name	optional	0 or 1	Toll-like receptor 4
11	DB Object Synonym (Synonym)	optional	0 or greater	hToll
12	DB Object Type	required	1	protein
13	Taxon(taxon)	required	1 or 2	taxon:9606
14	Date	required	1	20090118
15	Assigned By	required	1	SGD
16	Annotation Extension	optional	0 or greater	part_of(CL:0000576)
17	Gene Product Form ID	optional	0 or 1	UniProtKB:P12345-2

Gene Product Association Data (GPAD) fields:

Column	Content	Required?	Cardinality	Example
1	DB	required	1	SGD
2	DB Object ID	required	1	P12345
3	Qualifier	required	1 or greater	enables
4	GO ID	required	1	GO:0019104
5	DB:Reference(s) (DB:Reference)	required	1 or greater	PMID:20727966
6	Evidence Code	required	1	ECO:0000021
7	With (or) From	optional	0 or greater	Ensembl:ENSRNOP00000010579
8	Interacting taxon ID	optional	0 or 1	4896
9	Date	required	1	20130529
10	Assigned by	required	1	PomBase
11	Annotation Extension	optional	0 or greater	occurs_in(GO:0005739)
12	Annotation Properties	optional	0 or greater	annotation_identifier = 2113431320

Titre : Modèle Transformer pour l'interprétabilité et les prédictions multi-niveaux des fonctions des protéines à partir de leurs séquences

Mot clés : Annotation fonctionnelle automatique, Apprentissage profond, Transformer, Enzymes, Gene Ontology

Résumé : L'annotation automatique des séquences protéiques est en plein essor pour gérer l'augmentation des séquences non annotées expérimentalement. Premièrement nous avons étudié l'application du Transformer à la prédiction des fonctions enzymatiques. Le modèle EnzBert améliore le macro-f1 de 41% à 54% comparé au précédent état de l'art. De plus une comparaison des méthodes d'interprétabilité montre qu'une approche basée sur l'attention obtient un score F-Gain de 96,05%, surpassant les méthodes classiques (91,44%). Deuxièmement l'intégration de la Gene Ontology dans les modèles de prédic-

tion de fonctions a été explorée. Deux approches ont été testées : l'intégration dans le processus de labellisation et l'utilisation de plongements hyperboliques. Les résultats obtenus confirment à la fois l'efficacité de la propagation des labels selon la hiérarchie GO et la supériorité des plongements hyperboliques (mean WFmax : 0.36) par rapport au modèle euclidien (0.34) en petite dimension (32). Ils maintiennent une plus grande cohérence avec la Gene Ontology (relations correctement ordonnées : 99.25%-99.28% vs. 78.48%-91.41% pour modèle euclidien).

Title: Transformers models for interpretable and multilevel prediction of protein functions from sequences

Keywords: Automatic functional annotation, Deep learning, Transformer, Enzymes Gene Ontology

Abstract: Automatic annotation of protein sequences is on the rise to manage the increasing number of experimentally unannotated sequences. First, we investigated the application of the Transformer for enzymatic function prediction. The EnzBert model improves macro-F1 from 41% to 54% compared to the previous state-of-the-art. Furthermore, a comparison of interpretability methods shows that an attention-based approach achieves an F-Gain score of 96.05%, surpassing classical methods (91.44%). Second, the integration of Gene

Ontology into function prediction models was explored. Two approaches were tested: integration in the labeling process and the use of hyperbolic embeddings. The results confirm both the effectiveness of the True Path Rule and the superiority of hyperbolic embeddings (mean WFmax: 0.36) compared to the Euclidean model (0.34) in low dimensions (32). They maintain greater consistency with the Gene Ontology (correctly ordered relations: 99.25%-99.28% vs. 78.48%-91.41% for the Euclidean model).