Rapport de projet long de Master Bioinformatique à l'Université de Paris 2022-2023.

Présenté par :

BELAKTIB Anas

Predicting protein-carbohydrate binding sites using protein embeddings

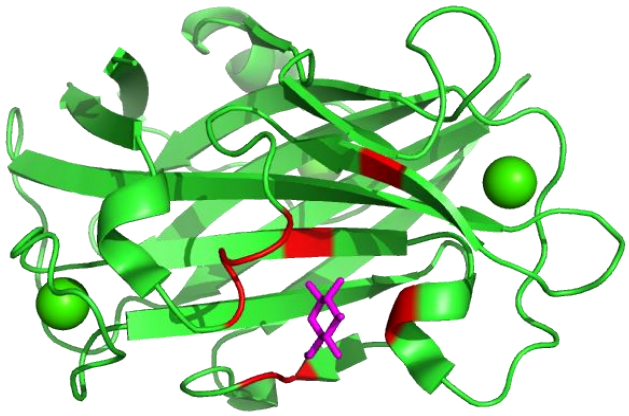

Table des matières

Introduction :

Les interactions entre les protéines et les glucides jouent un rôle important dans différents processus biologiques, notamment l'embryogenèse [1], la réponse immunitaire [2], le trafic de protéines [3], l'absorption de toxines bactériennes [4] et l'infection virale [5]. Leur rôle a été récemment mis en avant en raison de l'importance de la modélisation des glucides dans la compréhension de la réplication du virus SARS-CoV-2.
En effet, dans le contexte de la conception d'un vaccin, il est essentiel de prendre en compte toutes les stratégies développées par les virus pour échapper à la réponse immunitaire de l'hôte. Dans ce cadre, de nombreux virus utilisent un bouclier de glycanes pour masquer les épitopes immunogènes ciblés par les anticorps neutralisants [6] [7].

Le bouclier de glucide joue un rôle critique en cachant la surface de la protéine S de la reconnaissance moléculaire. Cependant, pour fonctionner efficacement, le pic doit reconnaître et se lier aux récepteurs ACE2 qui constituent la principale voie d'infection des cellules hôtes. [8,9].
Les glucides sont capables de former des glycanes complexes et ramifiés à partir d'unités monosaccharides, ils font donc partie des classes de ligands les plus polyvalents. De nos jours, les informations relatives aux glucides sont mal annotées [10] dans la Protein Data Bank (PDB) [11], la flexibilité et la faible énergie des interactions protéine-glucide rend l'étude des interactions glucide-protéine complexes.
Les outils de prédiction modélisant les interactions glucide protéine existe, mais manque de précision à cause de telle complexité.
Pour y remédier, nous avons récupéré des structures expérimentales de complexes liés aux protéines et aux glucides dans une base de données existante et créé notre propre base de données de sites de liaison entre protéines et glucides. À travers ce projet, notre équipe conçoit différentes architectures d'apprentissage profond pour prédire l'emplacement des sites de liaison des glucides sur la surface d'une protéine.

Figure 1 : Fixation du Galactose sur 1DIW en rouge les acides aminées fixant le sucre

Matériels et méthodes :

Jeu de données :

Le jeu de données initial est constitué de 5172 protéines [12] dont la séquence est de taille variable entre 14 et 1528 acides aminés. Pour chacun des acides aminés, nous avons 321 features ainsi qu'une classe 0 ou 1 correspondant à la fixation ou non des glucides qu'on essaye de prédire. Le jeu de données va être divisé en 3 pour ainsi obtenir un jeu d'apprentissage, un jeu de validation et un jeu test.
Pour éviter la redondance des ensembles d'apprentissage, de validation et de test, nous avons utilisé la classification ECOD [13]. Dans la classification ECOD, chaque domaine protéique est affecté à cinq groupes, indiquant leur relation évolutive à différents niveaux. Pour éliminer la redondance entre l'ensemble de formation-validation et l'ensemble de test, nous avons conçu l'ensemble de test de manière qu'il contienne des données provenant d'un seul domaine à différents niveaux H (troisième niveau ECOD, homologie). Nous avons sélectionné des fenêtres glissantes à partir de 178 domaines, chacun à un niveau H différent, soit 36 425 fenêtres coulissantes. Nous avons ensuite exclu les données des domaines au même niveau H du reste de l'ensemble de données (607 380 fenêtres). Pour sélectionner et optimiser notre modèle, nous avons utilisé une stratégie de validation croisée plate. De même, pour des raisons de robustesse, nous avons divisé l'ensemble d'apprentissage et l'ensemble de validation de manière qu'ils contiennent des fenêtres glissantes de niveau X strictement différent (deuxième niveau ECOD, homologie possible) et que l'ensemble de validation ait une taille d'environ 15 % de l'ensemble d'apprentissage. Finalement, l'ensemble d'apprentissage contient 2.951.657 fenêtres, l'ensemble de validation 345.882.

Encodage des données : Transformers Evolutionary Scale Modeling

Les Transformers sont des modèles d'apprentissage profond initialement développés pour le traitement du texte en utilisant un mécanisme d'attention. Les réseaux de neurones Transformers ont pour but de prédire une séquence de longueur variable en fonction d'une autre séquence de longueur variable. Les mots constituant ces phrases possèdent une distribution de probabilité d'apparition issue d'entrainement sur des corpus de texte, on parle de modèle de langage. Ces systèmes sont très utiles pour faire de la traduction, car ces réseaux de neurones permettent de tenir compte du contexte d'un mot grâce au mécanisme d'attention. Les modèle de langage ont pu être adaptés à des langages biologiques comme les séquences d'acide aminé.
Facebook AI Research a développé des Transformers qui sont des modèles de langage protéique permettant de prédire différentes tâches. Notamment, ils ont développé les modèles pour le traitement des données protéiques. Dans ce cas, les séquences des acides aminés sont considérées comme des phrases constituées de mots.

Evolutionary Scale Modeling (ESM) est un des Transformers développé par Facebook AI Research [14]. Parmi eux, ESM-2 surpasse tous les modèles de langage protéique monoséquence testés dans une gamme de tâches de prédiction de structure. Il existe différente version de ce Transformers, parmi lesquelles le nombre de couches, le nombre de paramètres et le nombre de dimensions de sortie varie. Pour ce projet, nous utilisons la version Esm2_t6_8M_UR50D du Transformers.
La version Esm2_t6_8M_UR50D à la particularité d'être la plus petite en termes de paramètre que l'on retrouve avec 8M de paramètres. Ces paramètres sont divisés en 6 couches et permettent de générer pour chaque acide aminé constituant une séquence protéique un total de 320 features correspondant à diverses informations sur la séquence. Une feature supplémentaire est ajoutée nous précisant s'il s'agit d'un vrai résidu ou si la fenêtre glissante dépasse de la protéine. À la sortie de ce Transformers, nos séquences qui possédaient une dimension se retrouvent à 321 dimensions, elles sont dites "embeded".

Fenêtres glissantes :

Étant donné que pour ce genre de problème, le résultat de la classe peut être dû aux résidus précédents ou suivants le résidu que l'on veut prédire. Notre modèle va prendre en entrer une fenêtre glissante. Cette fenêtre glissante comprendra les features des six résidus précédents le résidu à prédire, les features du résidu à prédire et les features des six résidus suivants. S'il s'agit d'un résidu, au bord de notre séquence, alors les features auront comme valeurs 0.

Implémentation du programme :

Le programme a été implémenté sous :

- python 3.8.12
- tensorflow-gpu-2.6
- keras 2.6.0
- numpy 1.19.5
- pandas 1.3.4
- h5py 3.1.0
- scikit-learn 1.0.1

Compression du jeu de donnée :

Notre jeu de données est très volumineux (~100Gb), et par sa taille, nous ne pouvons pas le charger en mémoire sous la forme de vecteur pour l'implémenter dans notre réseau de neurones. Pour stocker toute notre base de données dans un seul gros fichier tout en ayant accès rapidement à des sous-éléments de la base de données, nous avons compressé notre jeu de données grâce au module h5py sous la forme d'un fichier binaire h5 contenant toutes les informations de nos séquences.

Modèles

Pendant ce projet, nous avons développé deux modèles de réseau neuronal convolutif (CNN). Ces réseaux de neurones ont été élaborés afin de pouvoir prendre en entrée 321 features d'une séquence d'une certaine longueur. Et prédire en sortie la probabilité de fixation ou non des glucides le long de nos protéines sous la forme de classe binaire 0 ou 1.

Tableau 1 : Tableau comparant nos deux modèles

	CNN	CNN simple
Nombre de paramètres	4.8M	93k
Durée d'une époque	14 mins	10 mins

Le premier modèle est plus complexe que le second, il compte 4,8 millions de paramètres. Ce réseau de neurones est constitué de 3 couches de convolution 1D de taille de filtre 100, 120 et 144 et de taille de fenêtre de convolution de 3 ainsi qu'une couche de dropout d'une valeur de 0,3 afin de régulariser l'apprentissage. Il est suivi d'une couche Flatten qui permet d'aplatir nos dimensions et d'un réseau de couche entièrement connecté permettant de donner en sortie la bonne taille à nos vecteurs prédits. La dernière couche de dense possède une activation softmax dans le but d'obtenir des résultats sous la forme de probabilité.

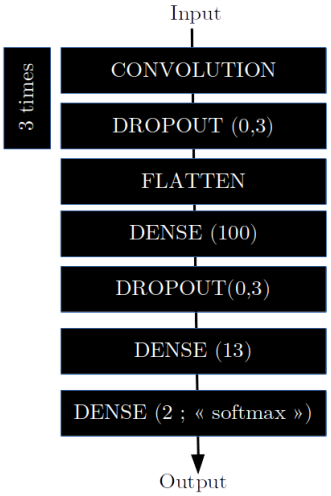

Figure 2 : Schéma de conception du modèle CNN

Le deuxième modèle est un modèle inspiré du premier, mais est plus simplifié. Ce réseau est constitué d'une couche de convolution 1D de taille de filtre 144 avec une taille de fenêtre de convolution de 3, une couche de dropout d'une valeur de 0,3 afin de régulariser l'apprentissage. Et du réseau de couche entièrement connecté suivi de la couche d'activation softmax. Les modèles ont été lancés avec un batch_size de taille 64, pendant 20 époques.

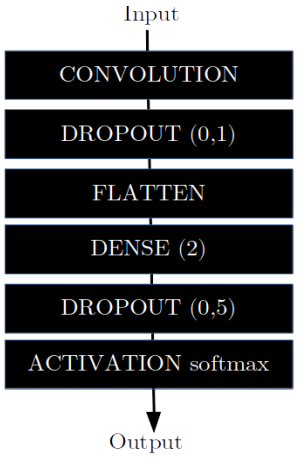

Figure 3 : Schéma de conception du modèle CNN simplifié

Optimisation des hyper paramètres :

Afin d'avoir un réseau de neurones optimal, nous avons conservé la structure présente obtenue grâce à Keras mais avons affiné les hyperparamètres tels que le nombre d'époques où le taux d'apprentissage. Le nombre d'époques a été paramétré dans le but de se stopper automatiquement dès lors ou notre modèle ne s'améliore plus du tout sur une marge de 10 époques sur le jeu de validation. Le taux d'apprentissage permet d'affiner l'apprentissage en jouant sur la vitesse de descente du gradient à chaque itération.

Déséquilibre des classes :

Ce problème survient lorsque la fréquence de la classe cible est fortement déséquilibrée, c'est-à-dire que l'occurrence d'une des classes est très élevée par rapport aux autres classes présentes. Le modèle aura alors tendance à attribuer la classe la plus fréquente et cette fausse prédiction sera camouflée par la présence de nombreuses vraies prédictions. Dans notre cas, l'événement rare est la fixation de sucres sur une protéine, nous avons énormément plus de région sans fixation qu'avec. Nous avons utilisé un algorithme issu de la bibliothèque python Scikit-learn pour récupérer le poids de chacune des classes qui utilise la formule suivante : $\frac{n\ samples}{(n\_classes * np.bincount(y))}$

Le ratio du déséquilibre des classes est le suivant [0 : 0,537 1 : 7,233].

Matrices de confusion :

Dans le but de vérifier les résultats de prédictions, le modèle est appliqué aux valeurs d'entrée du jeu de données, d'apprentissage et de validation. Les prédictions sont comparées aux valeurs réelles de fixation de glucide ou non sous la forme de matrices de confusions. Les matrices de confusion sont soit brutes, correspondant aux nombres de fois où chaque classe a été attribué. Ou elles sont normalisées sur l'axe "true", on obtient ainsi des matrices dans

lesquelles false positif + true positif = 1 et false negatif + true negatif =1. Cette normalisation permet de comparer nos performances dans chaque classe. Les matrices de confusion ont été réalisées via un algorithme de la bibliothèque python Scikit-learn [15].

Métriques :

Nos modèles ont été évalués par différentes métriques données par Keras.

- Accuracy (threshold=0.5) :

Calcule la fréquence à laquelle les prédictions correspondent aux classes initiales.
Cette métrique crée deux variables locales, total et count, qui sont utilisées pour calculer la fréquence à laquelle la prédiction correspond à la classe réelle en fixant un treshold qui correspond au seuil que doit avoir la probabilité pour être attribué à une classe.
Accuracy = $\frac{count}{total}$.

- Precision :

Calcule la précision des prédictions par rapport aux classes initiales.
La métrique crée deux variables locales, true_positives et false_positives.
Precision = $\frac{true\_positives}{true\_positives + false\_positives}$

- tf.keras.metrics.Recall :

Calcule le rappel des prédictions par rapport aux classes initiales.
Cette métrique crée deux variables locales, true_positives et false_negatives.
Recall = $\frac{true\_positives}{true\_positives + false\_negatives}$

- Aire sous la courbe (Area Under the Curve) :

L'AUC est une aire calculée à partir des courbes ROC (Receiver operating characteristic) qui représente le taux de vrai positif en fonction du taux de faux positif. Il s'agit en réalité d'une approximation en utilisant des sommes finies approchant des intégrales (somme de Riemann). Cette aire est une mesure de qualité des classifications binaires. Cette valeur est prédite en même temps que les prédictions sont réalisées. La valeur AUROC d'un modèle parfait est de 1.

- Precision Recall :

La courbe précision-rappel montre le compromis entre la précision et le rappel pour différents seuils. Une aire élevée sous la courbe représente à la fois un rappel et une précision élevés, où une précision élevée correspond à un faible taux de faux positifs, et un rappel élevé correspond à un faible taux de faux négatifs. Des scores élevés pour les deux montrent que le classificateur renvoie des résultats exacts (précision élevée), ainsi qu'une majorité de tous les résultats positifs (rappel élevé). Tout comme l'AUROC, la valeur AUPR d'un modèle parfait est de 1 tandis que celle d'un modèle aléatoire est de 0,5 pour les deux valeurs.

Résultats :

Commenté [1]: il faut dire qq mots sur le contenu de jeu de données: combien de résidus de chaque classe, idéalement distribution de chaque classe par type d'acid aminée par exemple. Une très breve analyse exploratoire

Il y a plusieurs variables permettant d'évaluer nos modèles. On remarque que le modèle a appris le long des différentes époques, lors de la phase d'apprentissage du jeu de donnée avec une réduction de la courbe Loss. Les différentes performances évaluées n'ont fait que s'améliorer et se sont stabilisés au bout de 15 époques ce qui a entrainé l'arrêt de notre apprentissage.
Pour ce qui est de la partie apprentissage, on se retrouve finalement avec une valeur AuROC de 0,5912. On constate à partir de là que notre modèle apprend et nous donne des classes qui ne sont pas aléatoires, un modèle aléatoire à une valeur AuROC de 0,5. Notre précision est très faible, avec une valeur de 0,07, ce qui signifie que notre modèle a tendance à prédire des résultats positifs qui ne le sont pas. Notre modèle arrive à prédire quelques fois la classe 1 qui est très rare, ce qui est une avancée importante.

Notre rappel est meilleur que notre précision avec une valeur de 0,57 pour notre dernière époque, ce qui signifie que les faux négatifs sont moins retrouvés dans notre modèle que les faux positifs. Nous avons une accuracy estimé de 0.562, ce qui est un bon début ayant des classes déséquilibré un modèle aléatoire aurait une très bonne accuracy juste en prédisant la classe 0 a l'ensemble de nos sorties.

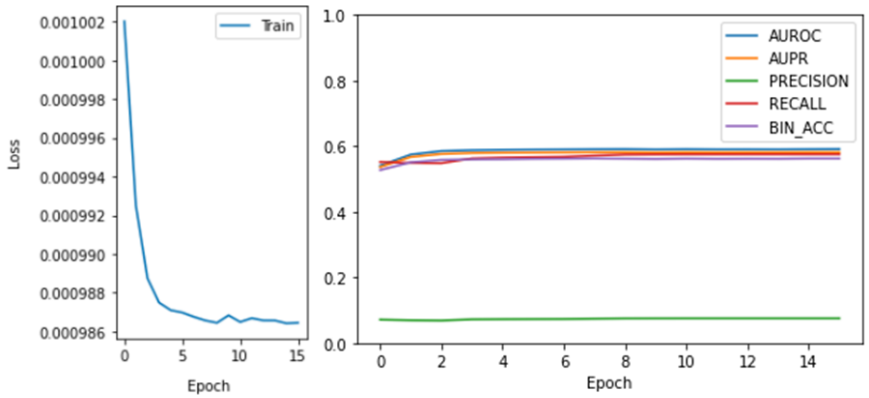

Figure 4 : Graphiques des performances du modèle CNN en fonction des époques lors de la phase d'apprentissage

Nous avons chargé le modèle obtenu afin de produire des matrices de confusion en reprenant le jeu de donnée d'apprentissage. La matrice de confusion quantitative nous montre que l'évènement le plus souvent retrouvé est le vrai négatif, ici, il s'agit de l'absence de fixation de sucre sur notre protéine. Ce résultat est attendu, car on savait que la fixation d'un sucre est un événement rare, donc l'absence de cette fixation est attendu un très grand nombre de fois dans notre jeu de donnée. L'événement vrai positif, ici, est rare (133937), on l'observe moins de fois que l'événement faux positif (1.5e+06). Ce qui signifie que notre modèle à tendance à attribuer des résultats positifs qui doivent être négatif.
Afin de comprendre ces probabilités, nous avons dû normaliser sur l'axe des vrais évènements, on se retrouve alors avec une matrice différente et plus explicative.

On constate de ce fait que la probabilité de donner une bonne prédiction est toujours supérieur à la probabilité de donner une fausse prédiction. Et qu'on a effectivement, 58 % de probabilité d'avoir de vrai positif et 52 % de probabilité d'avoir de vrais négatifs. Notre modèle a bien appris certaine information du jeu de donnée, mais il est actuellement incapable de réattribuer correctement toutes les classes.

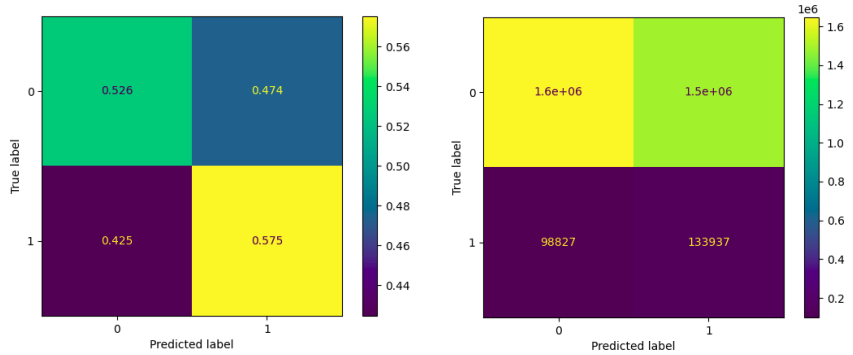

Figure 5 : Matrice de confusion brut et normalisé du modèle CNN lors de la phase d'apprentissage

Le modèle a été ensuite utilisé sur un jeu de donnée de validation. Les performances du jeu de validation sont proches de celle du jeu de donnée d'apprentissage. Notre courbe de loss ne réduit pas, elle augmente légèrement, ce qui va avec notre accuracy (0.49) qui diminue légèrement, mais n'évolue pas énormément. Toutes les performances sur le jeu de validation se voient légèrement réduites sauf notre rappel qui est légèrement accru sur ce jeu de donnée avec une valeur de 0,58 pour notre dernière époque, ce qui signifie que les faux négatifs sont moins retrouvés dans notre modèle. Notre modèle n'a pas été capable de prédire de meilleurs résultats. Mais il s'agit là d'un bon début, notre modèle ne prédit pas juste aléatoirement, il a retenu certains motifs du jeu de donnée d'apprentissage.

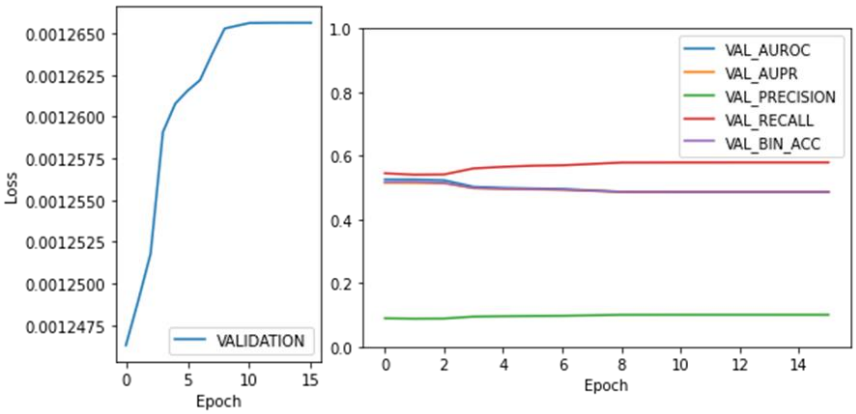

Figure 6 : Graphiques des performances du modèle CNN en fonction des époques lors de la phase de validation

En observant les matrice de confusion, on constate que ce jeu de donnée présente moins d'individu que le jeu d'apprentissage et que l'événement positif est rare. On remarque

également que cet événement rare lorsqu'il est observé aux mêmes probabilités d'être vraie que sur le jeu de donnée d'apprentissage 58%.

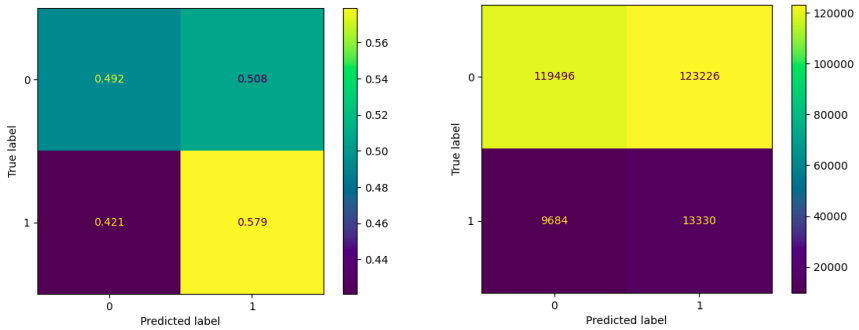

Figure 7 : Matrice de confusion brut et normalisé du modèle CNN lors de la phase de validation

Après avoir testé différents hyper paramètres pour notre modèle complexe, nous n'avons pas réussi à obtenir des performances accrues, nous avons donc décidé de fabriquer un autre modèle plus simple. Le but étant d'apprendre juste une fonctionnalité globale des protéines sur la fixation des sucres ou non. Ce nouveau modèle, a bien appris le long des différentes époques, on voit notre courbe loss diminué le fil des époques.

Pour ce qui est de la partie apprentissage, on se retrouve finalement avec une valeur AuROC de 0,67, on constate à partir de là que notre modèle apprend mieux et nous donne des classes. Notre précision est bonne, avec une valeur de 0,65, ce qui signifie que notre nouveau modèle prédit beaucoup moins de faux positif. Notre rappel reste inchangé avec une valeur de 0,48 pour notre dernière époque.
Ce qui nous donne une accuracy estimé de 0,61.

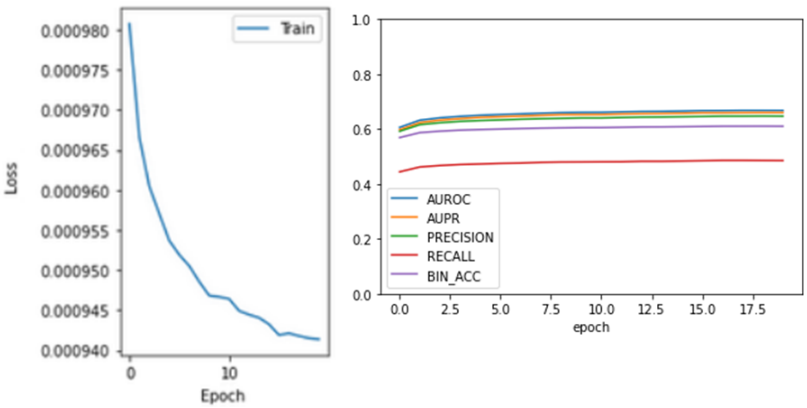

Figure 8 : Graphiques des performances du modèle CNN simplifié en fonction des époques lors de la phase d'apprentissage

Les matrices de confusion sur le jeu de donnée d'apprentissage confirment l'amélioration de nos modèles. En terme quantitatif, nous avons encore la majorité de nos résultats bien prédits, les groupes vrai positif et vrai négatif compte plus d'individu que les groupes faux positifs et

faux négatif. On le remarque encore plus une fois la normalisation réalisée. Nous avons la probabilité de donner une bonne prédiction est toujours supérieur à la probabilité de donner une fausse prédiction. On a effectivement, 59 % de probabilité d'avoir de vrai positif et 65 % de probabilité d'avoir de vrais négatifs. Notre modèle a mieux appris certaine le jeu de donnée, cette fois-ci, mais il reste incapable de réattribuer correctement toutes les classes.

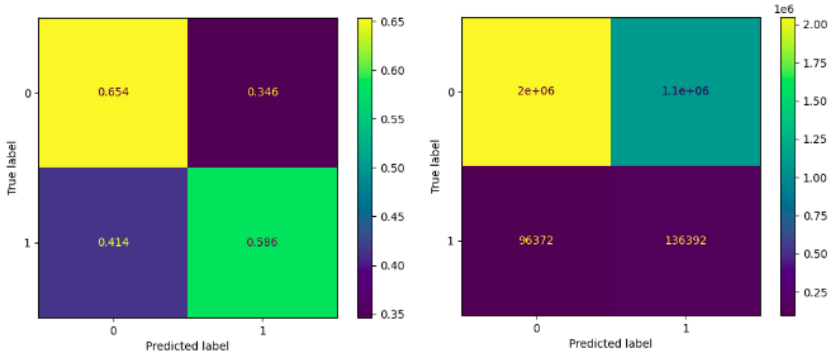

Figure 9 : Matrice de confusion brut et normalisé du modèle CNN simplifiée lors de la phase d'apprentissage

Le modèle a été repris pour prédire sur le jeu de donnée de validation. Notre courbe de loss décroit même si elle n'est pas continue, on observe une décroissance au cours des époques. La loss de la validation n'est pas stable au bout de 20 époques, elle aurait pu encore décroitre si l'entrainement avait continu. Les performances du jeu de validation sont proches de celle du jeu de donnée d'apprentissage. Nous nous retrouvons avec de bonne performance de validation, une accuracy de 0,61, une précision de 0,55, un recall de 0,55.
Ces résultats montrent qu'un modèle moins complexe peu donner de meilleurs résultats à notre problématique. Nous avons ainsi pu corriger nos problèmes de précision rencontrée précédemment. Il aurait été intéressant d'étudier un tel modèle avec un nombre d'époques plus conséquent afin de voir à quel point un modèle simple est capable d'être juste.

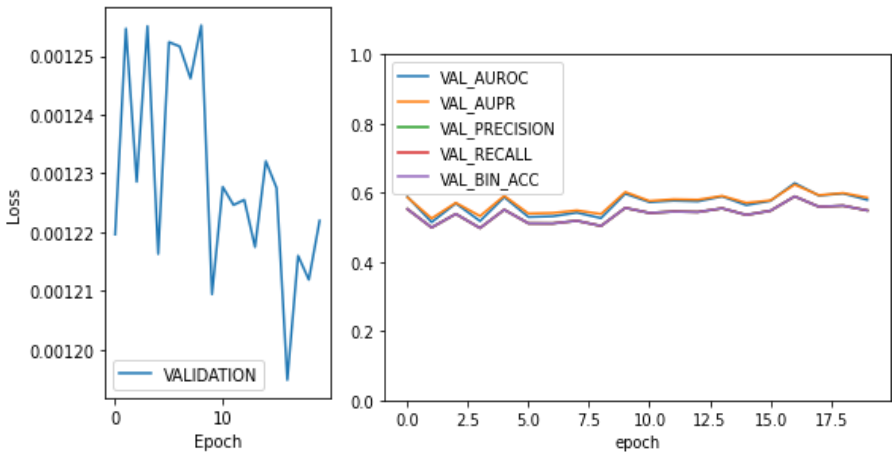

Figure 10 : Graphiques des performances du modèle CNN simplifié en fonction des époques lors de la phase de validation

Les matrices de confusion du jeu de donnée de validation montrent des résultats satisfaisants. La majorité des prédictions sont justes de manière quantitative, les classes les plus distribuées sont les bonnes. Nous avons 52% de probabilité d'avoir de vrai positif et 56% de probabilité d'avoir de vrai négatif. On améliore avec ce modèle la prédiction de vrai négatif, mais on diminue celle de vrai positif.

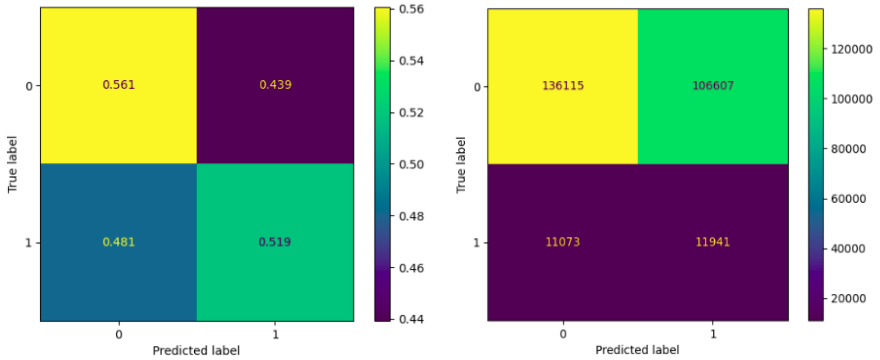

Figure 11 : Matrice de confusion brut et normalisé du modèle CNN simplifiée lors de la phase de validation

Ce modèle simplifié est plus rapide avec des performances qui sont satisfaisantes, mais la loss montre que le modèle peut encore s'améliorer. Donc le modèle simplifié est pour nous l'approche la plus prometteuse. De plus, nous avons passé pas mal de temps à modifier les hyperparamètres du modèle complexe, mais n'avons pas eu le temps de faire les mêmes optimisations sur le modèle simplifié. On a alors comparé les performances de notre modèle complexe qu'on a amélioré au mieux avec un modèle simple dont l'optimisation reste encore à faire.

Perspectives

Le modèle le plus prometteur est finalement le modèle CNN simplifié. Il a perdu un peu de performance au cours des dernières époques, mais il a montré qu'il pouvait avoir de bons résultats. Nous notons notamment les performances de la seizième époque qui se révèle être celle dont les résultats sont les meilleurs, dépassants tous les résultats de notre modèle CNN, même optimisé. La courbe Loss montre une décroissance discontinue qui nous montre une possibilité d'amélioration dans les époques futures.

On pourrait prolonger le nombre d'époques en conservant le modèle CNN simplifié afin de voir les performances évoluées dans un temps plus long. Nous pourrions aussi tester différent hyper paramètres comme nous l'avons fait pour le modèle CNN, ce qui pourrait nous faire augmenter nos performances.

Nous avons aussi mis en place des réseaux de neurones plus complexe tel qu'un Google Inception et un model comprenant plusieurs couches GRU. Ces modèles sont beaucoup plus complexes que les précédents étudiés. Nous n'avons pas eu le temps de les tester avec notre jeu de donnée, nous les avons juste compilés. Cependant, notre étude nous a montré qu'un modèle plus complexe en termes de paramètres et qui nécessite plus de ressource ne donnent pas forcément de meilleurs résultats.

La dernière étape manquante serait d'appliquer notre meilleur modèle au jeu de donnée test.

Références

(1) Onuma Y., Tateno H., Tsuji S., Hirabayashi J., Ito Y., Asashima M.. A lectin-based glycomic approach to identify characteristic features of xenopus embryogenesis. PLoS One. 2013; 8:e56581

(2) Maverakis E., Kim K., Shimoda M., Gershwin M.E., Patel F., Wilken R., Raychaudhuri S., Ruhaak L.R., Lebrilla C.B.. Glycans in the immune system and the altered glycan theory of autoimmunity: a critical review. J. Autoimmun. 2015; 57:1–13.

(3) Hauri H.-P., Nufer O., Breuza L., Tekaya H.B., Liang L.. Lectins and protein traffic early in the secretory pathway. Biochem. Soc. Symp. 2002; 69:73–82.

(4) Zuverink M., Barbieri J.T.. Protein toxins that utilize gangliosides as host receptors. Prog. Mol. Biol. Transl. Sci. 2018; 156:325–354.

(5) Chen L., Li F.. Structural analysis of the evolutionary origins of influenza virus hemagglutinin and other viral lectins. J. Virol. 2013; 87:4118–4120.

(6) Watanabe Y.; Allen J. D.; Wrapp D.; McLellan J. S.; Crispin M. Site-Specific Glycan Analysis of the SARS-CoV-2 Spike. Science 2020, eabb9983. 10.1126/science.abb9983.

(7) Casalino, L.; Gaieb, Z.; Goldsmith, J. A.; Hjorth, C. K.; Dommer, A. C.; Harbison, A. M.; Fogarty, C. A.; Barros, E. P.; Taylor, B. C.; McLellan, J. S.; Fadda, E.; Amaro, R. E. Beyond Shielding: The Roles of Glycans in the SARS-CoV-2 Spike Protein. *ACS Cent. Sci.* **2020**, *6* (10), 1722–1734. https://doi.org/10.1021/acscentsci.0c01056.

(8) Sztain, T.; Ahn, S.-H.; Bogetti, A. T.; Casalino, L.; Goldsmith, J. A.; Seitz, E.; McCool, R. S.; Kearns, F. L.; Acosta-Reyes, F.; Maji, S.; Mashayekhi, G.; McCammon, J. A.; Ourmazd, A.; Frank, J.; McLellan, J. S.; Chong, L. T.; Amaro, R. E. A Glycan Gate Controls Opening of the SARS-CoV-2 Spike Protein. *Nat. Chem.* **2021**, *13* (10), 963–968. https://doi.org/10.1038/s41557-021-00758-3.

(9) Yan R.; Zhang Y.; Li Y.; Xia L.; Guo Y.; Zhou Q. Structural Basis for the Recognition of SARS-CoV-2 by Full-Length Human ACE2. Science (Washington, DC, U. S.) 2020, 367 (6485), 1444–1448. 10.1126/science.abb2762.

(10) Lütteke T., Frank M., von der Lieth C.-W.. Data mining the protein data bank: automatic detection and assignment of carbohydrate structures. Carbohydr. Res. 2004; 339:1015–1020.

(11) Burley S.K., Berman H.M., Bhikadiya C., Bi C., Chen L., Di Costanzo L., Christie C., Dalenberg K., Duarte J.M., Dutta S. et al... RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic Acids Res. 2019; 47:D464–D474.

(12) Copoiu, L.; Torres, P. H. M.; Ascher, D. B.; Blundell, T. L.; Malhotra, S. ProCarbDB: A Database of Carbohydrate-Binding Proteins. *Nucleic Acids Res.* **2020**, *48* (D1), D368–D375. https://doi.org/10.1093/nar/gkz860.

(13) Cheng H, Schaeffer RD, Liao Y, Kinch LN, Pei J, Shi S, Kim BH, Grishin NV. ECOD: an evolutionary classification of protein domains. PLoS Comput Biol. 2014 Dec 4;10(12):e1003926. doi: 10.1371/journal.pcbi.1003926

(14) https://github.com/facebookresearch/esm

(15) https://scikit-learn.org/dev/index.html