Projet Deep Learning :
Classification of ligand-binding pockets in proteins

Ce projet est basé sur l'article: Pu L., Govindaraj R.G., Lemoine J.M., Wu H.C. & Brylinski M. (2018). DeepDrug3D: Classification of ligand-binding pockets in proteins with a convolutional neural network. *PLoS Comput Biol, 15(2)*.

I. Introduction

Il existe une multitude de type de protéine différentes. L'être humain en fabriquerait plus 100 000 sortes différentes parmi lesquelles ont retrouve des stéroïdes, des métalloprotéines, ... Il est fréquent que les protéines se lient avec des ligand. Un ligand est une molécule qui se lie de manière réversible à une macromolécule ciblée comme un protéine. Il peut jouer un rôle dans la stabilisation de la molécule, jouer un rôle de catalyseur ou encore transmettre un signal. Les protéines ont des formes bien spécifique que l'on pourrait comparer à des "poches" et le ligand vient si encastrer. Comprendre comme un ligand interagit avec une protéine est un enjeux scientifique majeur. En effet, il est important de comprendre ces interactions afin de pouvoir étudier les fonction moléculaire de certaines protéines mais également pour comprendre les liens évolutifs entre elles. D'un points de vu technologie, l'étude des ces interactions permet également de développer des enzymes et des substrats de synthèse utiles dans le développement de nouvelles médications. Cependant, malgré la technologie grandissante, il reste compliqué d'étudier avec précision ces interactions. Les protéines peuvent parfois présenter de grandes similarité structurales mais pourtant correspondre à des ligands différents. La vitesse de production de données étant en augmentation il devient important de pouvoir discriminer une protéine en fonction de son ligand.

Dans l'article sur lequel ce travail est basé et inspiré ce travail, les chercheurs ont tout d'abord commencé par modélisé la structure moléculaires de différentes protéines. Pour ce faire, ils ont modélisé une sphère de rayon 15 Å de voxels espacé de 1 Å autour de la protéine. Les voxels sont au fur et à mesure éliminés de la sphère: en premier les voxels superposant la protéine; Ensuite les voxels trop éloignés de la protéines. A chaque voxels restant est attribué un potentiel d'interaction électrostatique. Il existe 14 interactions possibles. La poche ainsi modélisé est alors placé dans un cube de 32x32x32. Pour ce projet, les données qui sont utilisés ont déjà été mises sous cette forme.

On dispose ici d'un jeu de données contenant 596 poche de hèmes, 1553 poche de nucléotides, 69 poches de stéroïdes et 1946 poches contrôles. Les hèmes sont des cofacteurs contenant un atome de métal, souvent du fer, servant à accueillir un gaz diatomique (par exemple du dioxygène O2). Parmi les protéines contenant des hèmes on retrouve notamment les hémoglobines ou encore le cytochrome C par exemple. Le groupe contrôle est constitué de protéines et de poches qui ne sont pas dans les groupes hèmes, nucléotides ou stéroïdes. Pour chaque poche il y a une protéine associé.

Le but de se projet est donc de créer un réseau de neurones capables de discriminer la nature d'une partie de la structure d'une protéines entre hème, nucléotide, stéroïde ou autre en fonction des potentiels d'interaction électrostatiques entre la protéine et son ligand.

II. Matériels et méthodes

Tout les fichiers et documents sont disponibles sur GitHub à cette adresse (https://github.com/nicolassilva/DeepLearningProject_DeepDrig3D). Ce projet a été réalisé sous un environnement Conda avec Python 3.7. Sur cet environnement on été également installés Numpy, Random, Keras and Tensorflow. Sur le git, est également disponible un fichier .yml avec lequel la création d'un environnement Conda identique à celui dans lequel le projet a été réalisé est possible.

La première étape du programme est de créer un jeu d'apprentissage pour le réseau. Les données de voxels sont bien données sous formats binaire ce qui facilite l'importation des data par le programme python. Cependant, il est nécessaire ici de créer un array au format binaire qui permet d'indiquer a quel groupe (hème, nucléotide, stéroïde ou contrôle) appartient la poche d'intérêt. Pour la partie apprentissage, il est important de réduire le jeu de données car l'analyse de la totalité du jeu de données surpasse la mémoire d'un ordinateur portable classique. Comme il y a beaucoup moins de stéroïdes que des autres classes j'ai dupliqué la totalité des stéroïdes N fois. N correspondant à la centaine des autres classes sélectionnées (par exemple pour 300 hèmes, nucléotides et contrôles, alors N=3). Cela évite que les stéroïdes ne se retrouvent noyés dans la masse de données des autres classes.

On définit ensuite la dimension du canal en "Channel first". On peut définir un canal de deux manières en prenant comme exemple ici les données: la première manière est de donné d'abord les dimensions du cube autour de la poche et ensuite le type d'interaction obtenant ainsi des données aux dimensions 32x32x32x14 ("Channel last"); la seconde manière est de donné d'abord le type d'interaction et ensuite les dimensions du cube autour de la poche obtenant ainsi des données aux dimensions 14x32x32x32 ("Channel first").

Plusieurs réseaux on été testés ici:

- Le premier réseau est composé d'une couche de convolution 3D constitué de 16 neurones et d'une taille de kernel de 2x2x2. Il y a ensuite un drop_out de 20% suivie d'un maxPooling de 2x2x2 et d'encore un drop_out de 20%.
- Le second réseau est composé de deux couches de convolution 3D constitué de 64 neurones et d'une taille de kernel de 2x2x2 chacune. Il y a ensuite un drop_out de 20% suivie d'un maxPooling de 2x2x2 et d'encore un drop_out de 20%.
- Le second réseau est composé de deux couches de convolution 3D constitué de 4 neurones et d'une taille de kernel de 2x2x2 pour la première et de 16 neurones et d'un kernel_size de 2x2x2 pour la seconde. Il y a ensuite un drop_out de 20% suivie d'un maxPooling de 2x2x2 et d'encore un drop_out de 20%.

Tous les modèles implémentés finissent par une couche flatten et une couche dense de 4 neurones avec une activation softmax. Ils sont effectués avec un batch size de 16, un validation split de 30% et un nombre d'epochs de 300. Afin d'éviter le sur-apprentissage les modèles ont un earlyStopping avec une patience de 10. Afin d'évaluer le modèle, un jeu de données Test est créé sur le même principe que le jeu de donnée d'apprentissage et on regarde les performance du réseau de neurones sur ce jeu de données la.

III. Résultats

IV. Conclusion