Université Paris Diderot 2021-2022

Université de Paris

Master 2 - Bioinformatique

Projet Kaggle : OpenVaccine : COVID-19 mRNA Vaccine Degradation Prediction

---

Auteurs :
Hippolyte MIZINIAK - Nicolas ROBILLARD - Taroumta DJIRAÏBE

Encadrants :
Jean-Christophe GELLY
Frédéric GUYON

28 octobre 2021

1 Introduction

Les vaccins à ARNm représentent une technologie émergente. Cette émergence a été mis en avant dans le cadre de la crise sanitaire du covid-19 que nous vivons depuis 2 ans maintenant. La prolifération rapide de ce virus ainsi que les cas graves entrainant soit la mort, soit les complications respiratoires, ont rendu urgent la création d'un vaccin afin d'enrayer la prolifération de ce virus. Le vaccin à ARNm s'est tout de suite imposé comme la solution la plus efficace et la plus rapide devant l'ampleur du problème. En effet les vaccins "traditionnels" utilisent du matériel viral (fragment de protéines du virus) ainsi qu'un vecteur de production de protéines virales, pour permettre une grande reproduction des protéines virales, toutes ces étapes représentent un coût financier ainsi que du temps.

Dans le cas du vaccin à ARNm beaucoup d'étapes sont simplifiées, en effet, l'antigène de la protéine virale est directement produit par les cellules de l'individu après injection de l'ARNm, l'autre avantage est que l'ARNm se réplique et se dégrade tout seul.

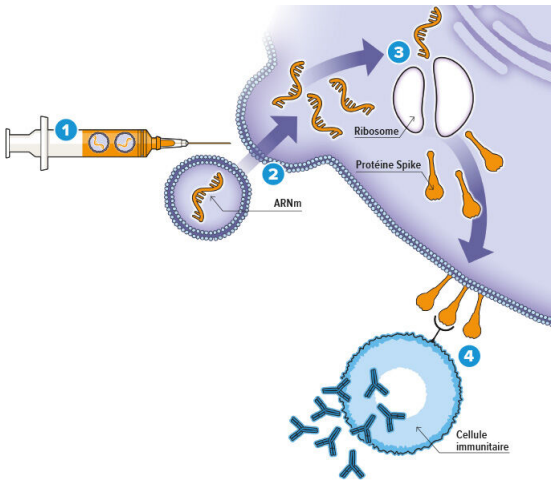

FIGURE 1 – *Schéma de la technologie du vaccin à ARNm.*

2 Objectif

L'objectif de ce projet va être de créer un modèle capable de prédire le taux de dégradation de chaque base (celles ayant les plus faibles interactions) de chaque séquence d'ARNm fournit par Kaggle dans les jeux de données, afin de les comparer entre elles et de sélectionner les modèles les plus pertinents pour la prédiction. Les trois principaux descripteurs qui vont nous aider à faire ces prédictions sont les séquences en elles-mêmes, l'appariement des bases et enfin la prédiction des boucles. Nous allons essayer de déterminer quelles sont les ARNm éligibles pour un vaccin ceux ayant le plus faible taux de dégradations.

3 Matériels & Méthodes

3.1 Données Brutes

Les données fournies par l'équipe de Stanford comportent 3029 séquences d'une longueur de 107 bases chacune. Les mesures ne peuvent être effectuer que sur les 68 premières bases. Ces séquences ont été séparés en deux sets de données "train" et "test".

- "train.json" : correspond aux données d'apprentissage
- "test.json" : correspond à l'ensemble de test
- "sample_submission.csv" : le fichier de soumission

Les données "test" publiques sont au nombre de 629 elles ont été traitées au préalable selon des critères spécifiques dont un étant le rapport signal/bruit supérieur à 1. "SN filter" est une indication sur le filtrage des séquences. Les données "train" correspondent aux 2400 autres séquences. Quant aux données tests privées elles sont au nombre de 3005 avec une longueur de 130 bases on ne s'intéressera là qu'au 91 première bases.

Les données de séquence ont toutes été analysées sous 5 "targets" fournies par Kaggle sous forme d'un tableau à virgule flottante comportant les 68 premières bases de la séquence :

- "reactivity" : Utilisé pour déterminer la structure secondaire probable de l'échantillon d'ARN.
- "deg_pH10 " : probabilité de dégradation de l'appariement des bases/liaisons après incubation sans Magnésium à un pH élevé (pH = 10).
- "deg_Mg_pH10" : probabilité de dégradation de l'appariement des bases/liaisons après incubation avec Magnésium à un pH élevé (pH = 10).
- "deg_50C" : probabilité de dégradation de l'appariement des bases/liaisons après incubation sans Magnésium à haute température (50 degrés Celsius).
- "deg_Mg_50C" : probabilité de dégradation de l'appariement des bases/liaisons après incubation avec Magnésium à haute température (50 degrés Celsius).

Les prédictions sont faites en tenant compte d'un encodage spécifique des ARNm :

- Nous nous situons sur des séquences ARNm. Les bases prises en compte sont donc A, U, G et C.
- La structure des séquences est représentée de la manière suivante ex : "(....)" signifie que la base 0 est appariée avec la base 5 et que les bases 1 à 4 ne sont pas appariées (la première parenthèse est à la position 0 et la dernière à la position 5.
- "predicted_loop_type" correspond au type de structure de boucles attribuées "BEHIMSX" où chaque caractère représente un type de boucle.

3.2 Traitement des données

Avant de pouvoir tester un quelconque modèle il a été nécessaire de nettoyer les données. Toutes les séquences n'ayant pas un "SN filter" égale à 1 ont été enlevées du jeu de données. Pour un SN filter égal à 1 le rapport signal sur bruit est supérieur à 1.

3.3 Modèles

$$MCRMSE = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{N}\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2}$$

La soumission des modèles a pu se faire grâce à la méthode MCRMSE (Mean Columnwise Root Mean Squared Error). Elle représente la moyenne de toutes les valeurs de RMSE (Root Mean Square Error) pour chaque colonne (le RMSE représente l'erreur quadratique moyenne, il s'agit d'une règle de notation quadratique qui mesure également l'ampleur moyenne de l'erreur. C'est la racine carrée de la moyenne des différences carrées entre la prédiction et l'observation réelle). Le MCRMSE peut également s'écrire de la manière suivante :

$$MCRMSE = \frac{1}{m}\sum_{j=1}^{m} RMSE_j$$

Nous avons pris le parti dans ce projet de tester plusieurs modèles afin d'observer lequel correspondait le mieux, quels étaient les meilleurs résultats par rapport à notre type de données. Les modèles que nous avons testés sont le modèle RNN, Dense, GRU et LSTM.
Les modèles GRU (Gated Recurrent Unit) et LSTM (Longue Short-term Memory) sont des modèles dont l'avantage est une facilité à gérer la mémoire et notamment grâce à leur système de portes.
La nature séquentielle des données que nous utilisons tend à choisir un modèle plus adapté à ce type. Dans les modèles adaptés, on trouve notamment les Recurent Neural Network (RNN), ainsi que plusieurs variants de ces modèles, comme les GRU, les LSTM ainsi que les BERT. Le module Keras fourni plusieurs fonctions pour ces modèles, on testera donc certains de ces modèles pour en déterminer le plus efficace pour nos données.

4 Résultats & Discussion

4.1 Evaluation sur les données Train

La visualisation de la précision des modèles reste peu fiable car elle est basée sur les mêmes données que l'apprentissage. Elle nous permettra donc d'évaluer provisoirement le modèle mais une deuxième évaluation devra être réalisée avec les données tests.

4.1.1 Modèle Dense

Nous avons procédé à un apprentissage sur 100 epochs avec une activation soft max pour les 2 premières couches et une activation linéaire pour la dernière le tout en optimisation "adam". On peut observer assez vite l'apparition d'un plateau sur la loss(**fig 3**). L'accuracy(précision) (**fig 4**) stagne à différents paliers avec de grandes variations, on peut donc s'accorder à dire que ce modèle ne convient pas vraiment à nos données train. Nous avons essayé de complexifier le modèle mais cela n'a pas amélioré le rendu au niveau de la loss et de l'accuracy.

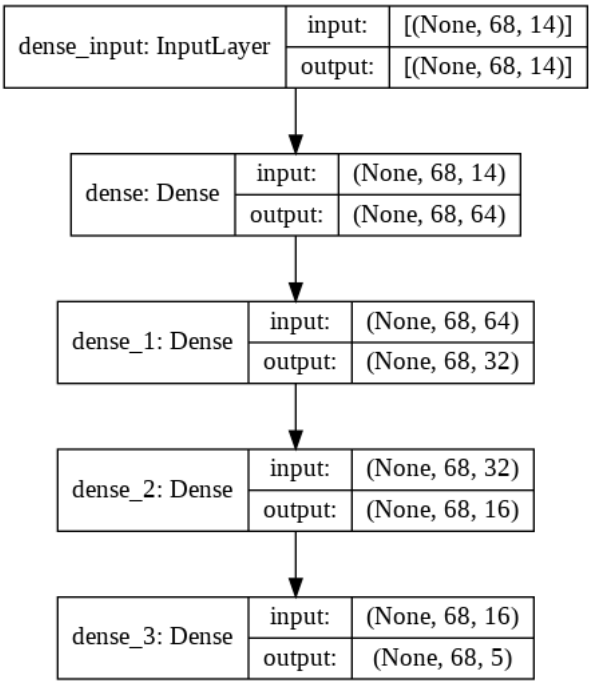

FIGURE 2 – *Diagramme du modèle Dense utilisé*

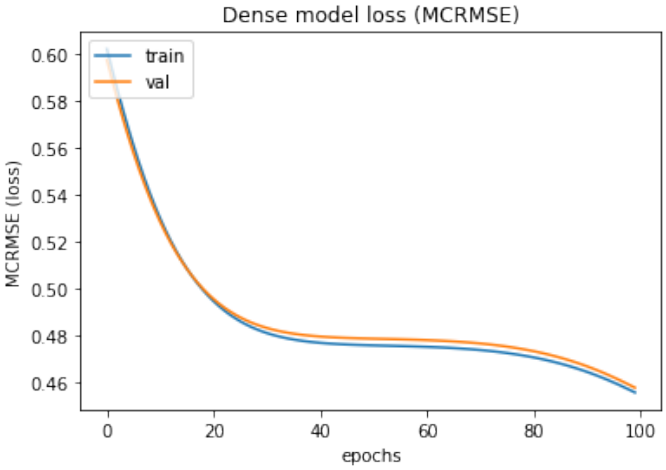

FIGURE 3 – Loss pour le modèle dense

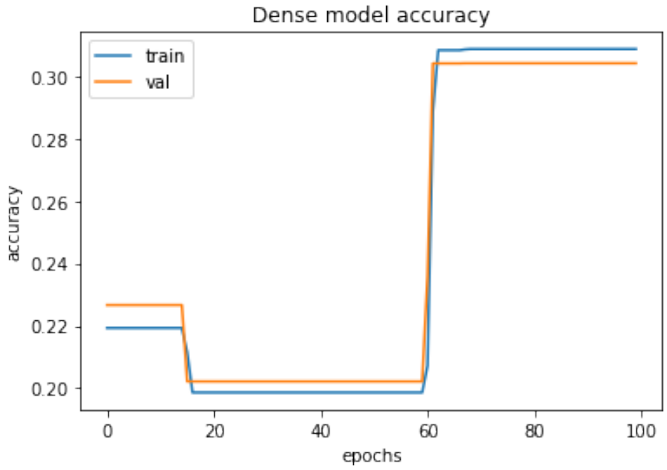

FIGURE 4 – Accuracy pour dense

4.1.2 Modèle RNN

Dans un premier temps nous avons testé le modèle RNN modèle basique fournit par la bibliothèque Keras de python. Nous avons utilisé 5 couches les 4 premières étant des couches RNN simple et la

dernière une couche dense. L'entraînement a été fait sur 50 epochs.

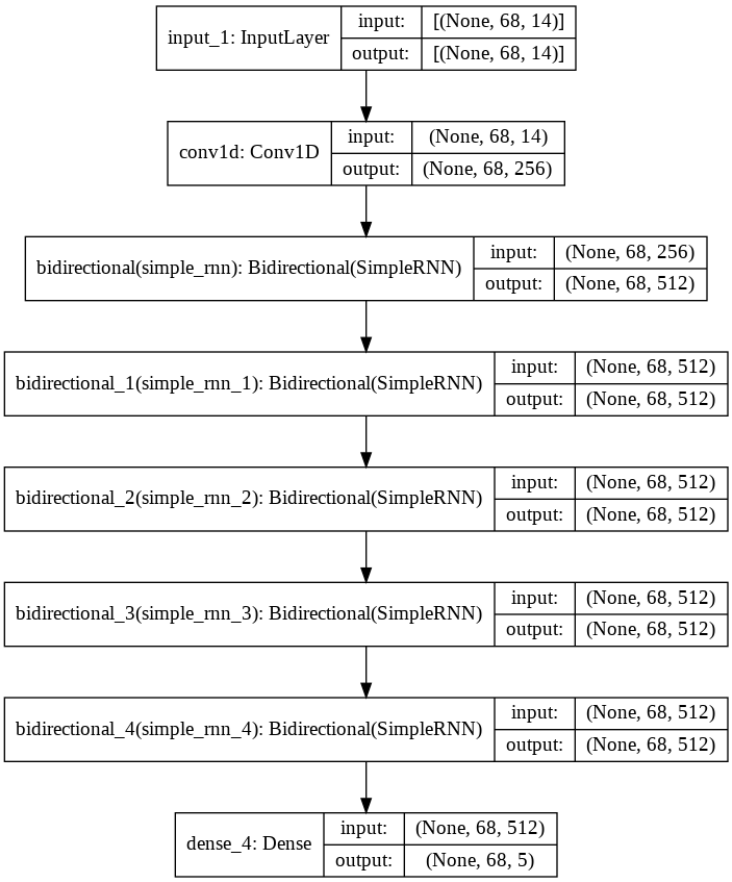

FIGURE 5 – *Diagramme du modèle RNN utilisé*

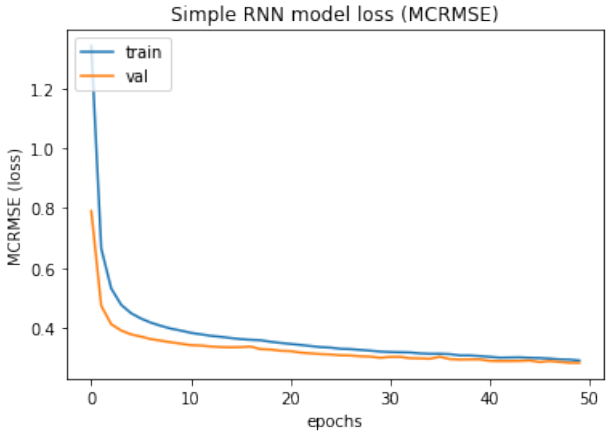

FIGURE 6 – Loss pour RNN

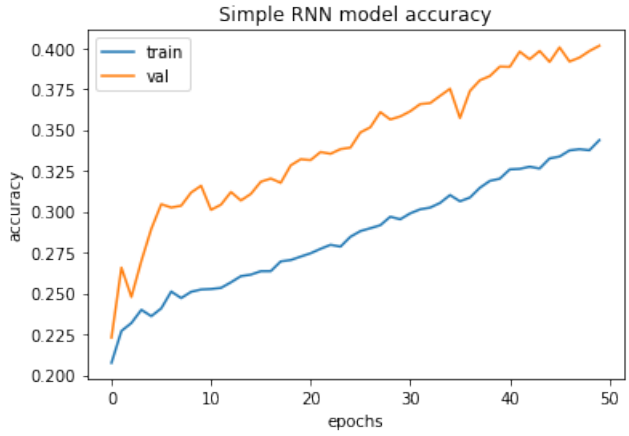

FIGURE 7 – Accuracy pour RNN

Le RNN simple n'apprend pas de manière optimale car il n'est pas fiable pour les dépendances à long terme à cause de la disparition du gradient, dans cette optique les solutions possibles sont l'utilisation de RNN plus complexes comme LSTM et GRU.

4.1.3 Modèle LSTM

Le LSTM *(Long Short Term Memory)* est un réseau de neurone réccurent ayant pour particularité de de résoudre plus facilement les problèmes liées au vanishing and exploding gradient (disparition du gradient), que l'on rencontre dans notre cas avec le RNN simple. Cette architecture vise à diviser le signal entre les paramètres important à court (hidden state) et long terme (cell state), et de propagé les informations pertienante à long terme tout en gardant celle à court terme. Dans notre projet, nous testons ce type de réseau de manière bidirectionnelle pour améliorer l'apprentisage et la prédiction du modèle vis à vis d'un RNN simple.

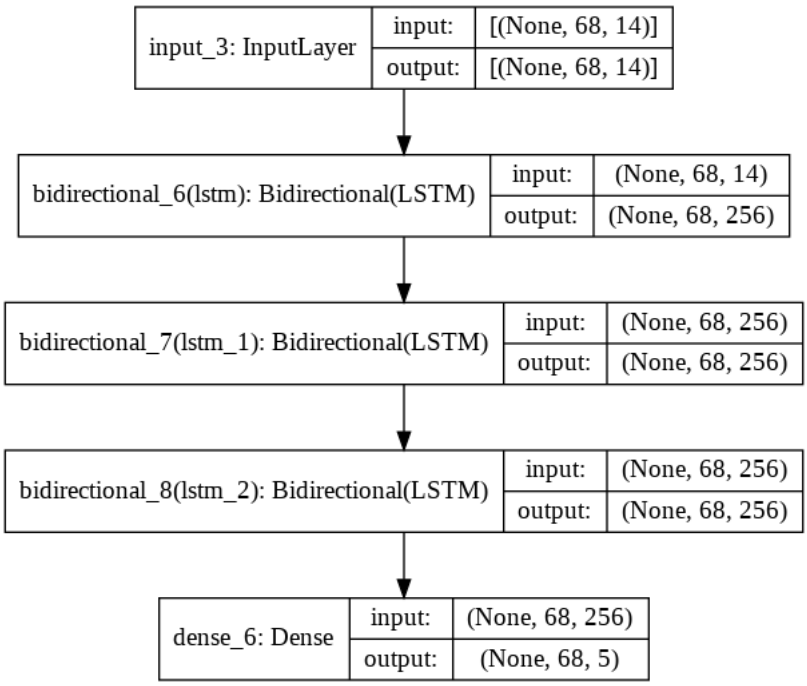

FIGURE 8 – *Diagramme du modèle LSTM utilisé*

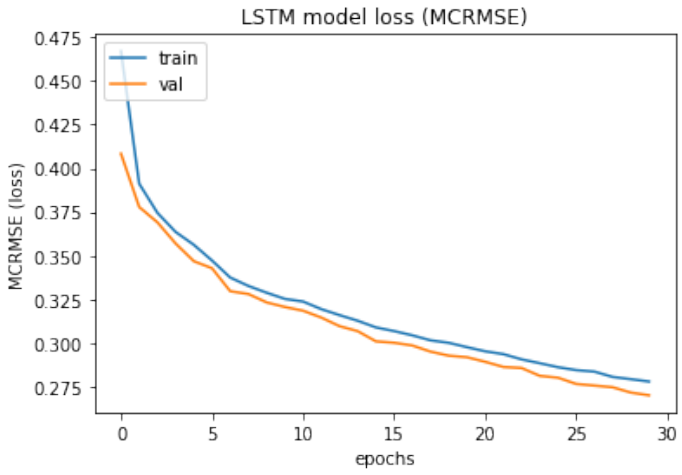

FIGURE 9 – Loss pour LSTM

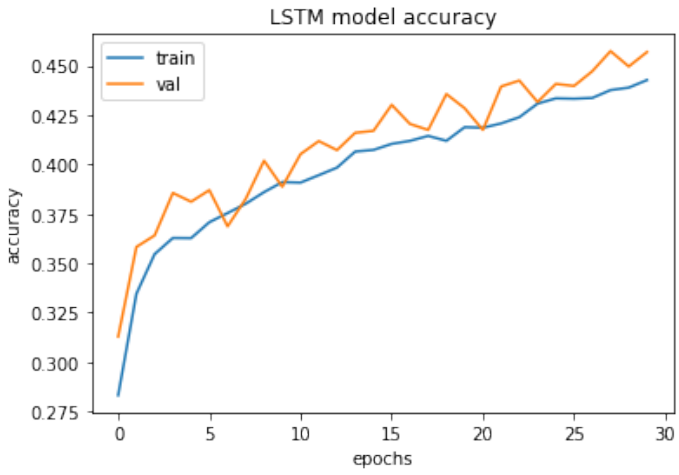

FIGURE 10 – Accuracy pour LSTM

On observe après 50 époques que le modèle LSTM à 3 couches bidirectionnelles, le modèle à mieux appris que le RNN simple, avec une MCRMSE qui diminue d'avantage et une précision sur les données d'entraînement plus importante.

4.1.4 Modèle GRU

Le modèle GRU est aussi un réseau de type RNN plus complexe : ce modèle de neurone vise également à résoudre les problèmes de la disparition du gradient. Il utilise un système de réinitialisation et de mise à jour des signaux. Celui-ci prend en compte moins de paraètre que le LSTM et donc reste plus efficace sur l'apprentisage que ce dernier. On test dans notre projet ce modèle pour savoir s'il améliore l'efficacité de notre modèle.

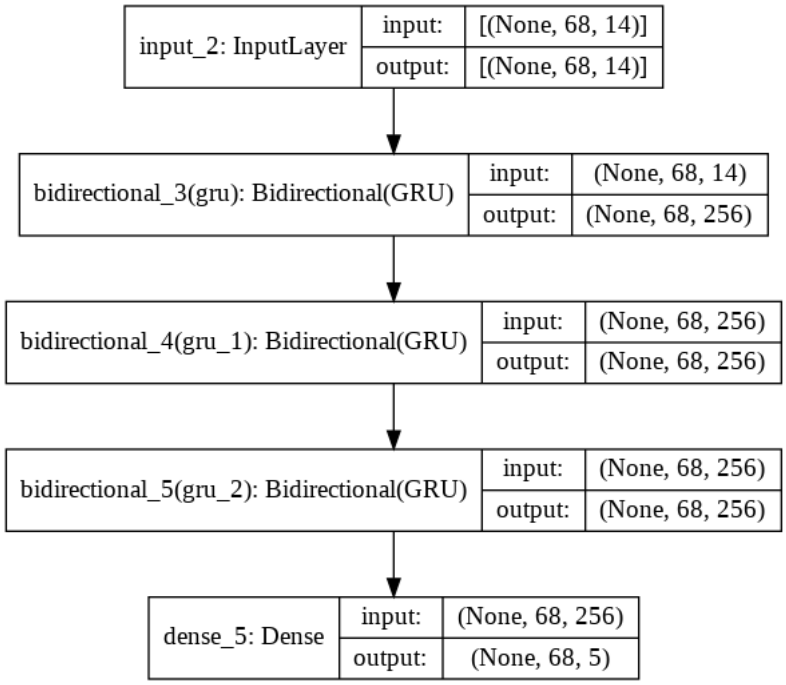

FIGURE 11 – *Diagramme du modèle GRU utilisé*

Dans les même conditions, on observe que le modèle GRU à une MCRMSE et une précision semblable à celles du LSTM. Cependant, le fait que celui-ci soit plus simple que le précédant nous encourage à l'utiliser pour nos données tests (principe du rasoir de d'Ockham).

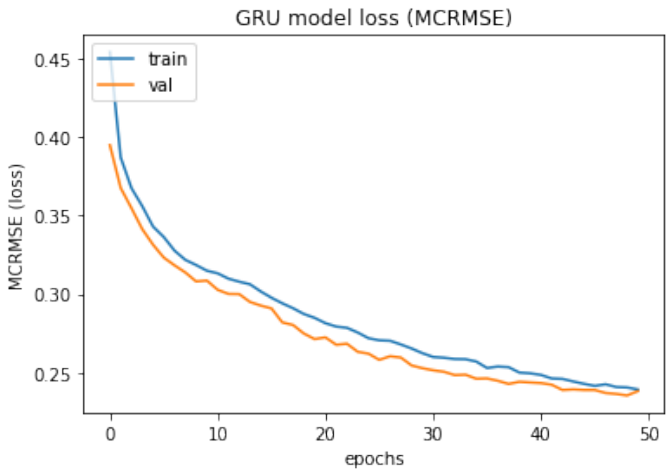

FIGURE 12 – Loss pour le GRU

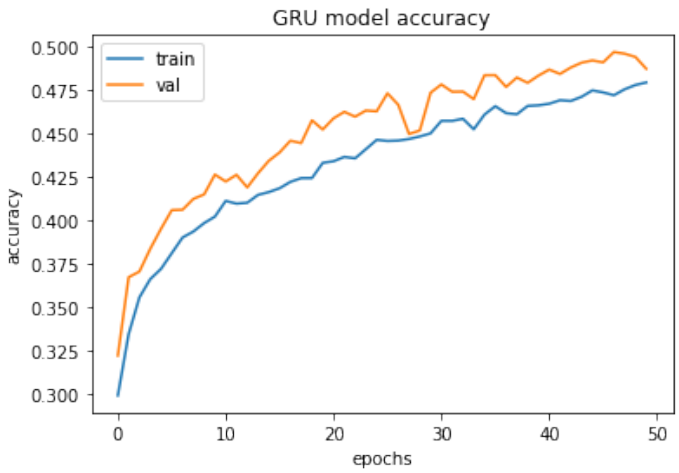

FIGURE 13 – Accuracy pour GRU

4.2 Evaluation sur les données Test

Nous avons pris le parti de choisir le modèle GRU pour les données test, les resultats etaient assez semblables au modèle LSTM mais avec un meilleur apprentissage tout de même. Nous avons recalculé le SN filter sur les données test publiques et nous avons établi que nos prédictions sembles acceptables.

4.3 Les données Test publiques

— Le test effectué sur la valeur minimale par rapport aux 5 conditions qui doivent être supérieures à -0,5 est bon.

— Le test sur le rapport signal/bruit moyen pour que les 5 conditions soient supérieures à 1,0 est aussi bons. (Le signal / bruit est défini comme moyenne (valeur de mesure supérieure à 68 nts) / moyenne (erreur statistique de la valeur de mesure supérieure à 68 nts)). On peut dire que le modèle semble plutot fiable, mais on ne peut pas être à 100% sûr. (On pourrait eventuellement effectuer des modeles plus complexes avec une meilleur loss pour tendre vers une plus grande fiabilité)

On ne sait pas si ça passe le SN_Filer sur les données Privates, donc on ne peut pas checker. (On peut essayer de calculer le signal_noise pour voir la proportions des privates et voir combien passent, mais ça ne donne aucune indication réelle)

5 Conclusion & Perspective

En conclusion, les modèles utilisés pour prédire le taux de dégradation de chaque base de chaque séquence d'ARNm ne sont pas tous équivalants en terme de fiabilité. Le modèle qui semble être le plus pertinent pour la prédiction est le module GRU. Les résultats ont été stocké dans un fichier "Submission.csv". Nous avons pu déterminer les ARNm éligibles pour un vaccin, c'est à dire, ayant les taux de dégradations les plus faibles.

Nos expériences ont été limité par les données qui ont été fourni car on ne pouvait utiliser que les 68 premières bases pour les données du Train et du Test publique, et les 91 premières bases pour les données du Test Privé. Il aurait été préférable de faire les tests sur l'ensemble des données.