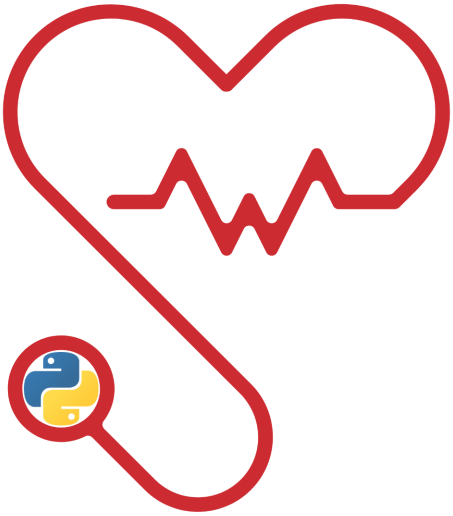
École Nationale Supérieur d'Ingénieurs Sud-Alsace – Mulhouse

Rapport de Projet – Apprentissage



Da Silva Jérémy
& Roguet William
Année 2022–2023

Responsables :
M. Devanne Maxime
& M. Hassenforder Michel
Date : 12 juin 2023

Da Silva Jérémy & Roguet William – 2A Informatique & Réseaux
https://github.com/WolfGang1710/ECG_Classification (*Repo privé*)

Table des matières

1 Présentation du projet

Le service de cardiologie d'un hôpital nous a contacté pour mettre un place une analyse de signaux ECG. En particulier pour la détection d'infarctus.
Des données nous on été fournies disponible ici :

— https://maxime-devanne.com/datasets/ECG200/ECG200_TRAIN.tsv

— https://maxime-devanne.com/datasets/ECG200/ECG200_TEST.tsv

2 Cahier des charges

— Construire et comparer les performances d'au moins deux modèles de Deep Learning dont un **Convolutional Neural Network** (CNN) et un **Recurrent Neural Network** (RNN)

— Un README expliquant le projet

3 Classification

3.1 Convolutional Neural Network

3.1.1 Explication du code

Nous avons utilisé les spécificités suivantes :

— Le modèle contient deux couches 1D de 5 filtres chacune ;

— Nous avons choisi un taux d'apprentissage de 0.01 pour éviter le sur-apprentissage ;

— Nous avons choisi un `batch_size` (taille de lot) de 128 pour éviter que le modèle ne se focalise uniquement sur des détaille propre aux données d'entraînement.

3.1.2 Résumé

Résumé du modèle `CNN` (donné directement par `Keras` de `Tensorflow`) :

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 96, 1)]           0

 conv1d (Conv1D)             (None, 96, 5)             20

 max_pooling1d (MaxPooling1D  (None, 48, 5)            0
 )

 conv1d_1 (Conv1D)           (None, 46, 5)             80

 max_pooling1d_1 (MaxPooling  (None, 23, 5)            0
 1D)

 flatten (Flatten)           (None, 115)               0

 dense (Dense)               (None, 2)                 232

=================================================================
Total params: 332
Trainable params: 332
Non-trainable params: 0
_________________________________________________________________
```

3.1.3 Résultats

Nous obtenons les résultats suivants pour le `CNN` :

- L'accuracy sur l'ensemble du train est : 0.82
- L'accuracy sur l'ensemble du test est : 0.80

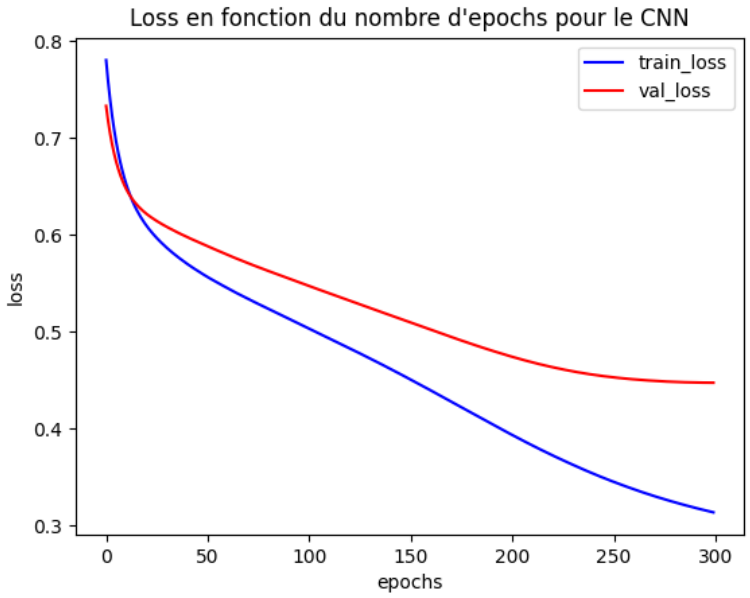


FIGURE 1 – Loss en fonction du nombre d'epochs pour le CNN

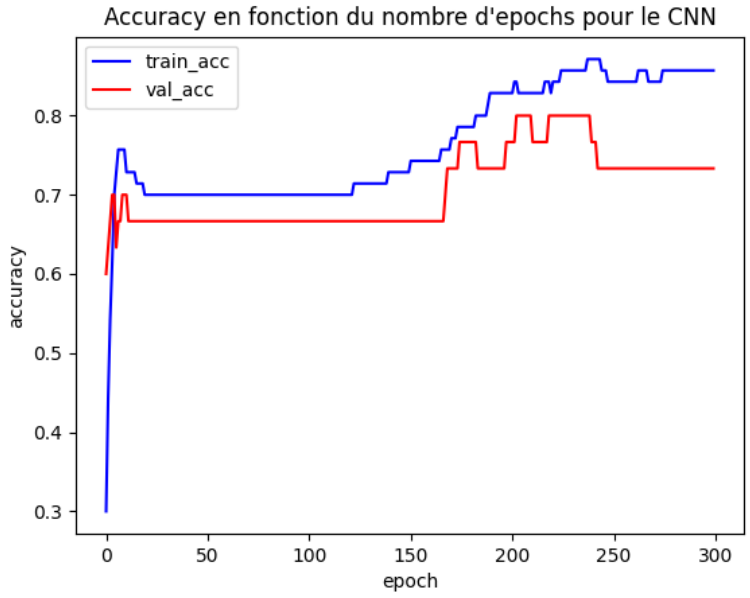


FIGURE 2 – Accuracy en fonction du nombre d'epochs pour le CNN

3.1.4 Interprétation

On constate que la précision diverge de 2 % entre l'ensemble de train et de test.
L'ensemble de test nous sert à évaluer la capacité du modèle à généraliser ce qu'il a appris à de nouvelles données inconnues.
Néanmoins, dans le cas général, une précision de 80 % est acceptable.

3.2 Recurrent Neural Network

3.2.1 Explication du code

Nous avons utilisé les spécificités suivantes :

- Le modèle `RNN` est composé de deux couches 1D. Nous avons choisi un nombre de neurones égale à 5 afin d'avoir un modèle avec une complexité modérée pour chaque couche ;
- `batch_size` : 64.
- Nombre d'époques : 300 (comme le `CNN`).

3.2.2 Résumé

Résumé du modèle `RNN` (donné directement par `Keras` de `Tensorflow`) :

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 96, 1)]           0

 simple_rnn (SimpleRNN)      (None, 96, 5)             35

 simple_rnn_1 (SimpleRNN)    (None, 5)                 55

 dense_1 (Dense)             (None, 2)                 12

=================================================================
Total params: 102
Trainable params: 102
Non-trainable params: 0
_________________________________________________________________
```

3.2.3 Résultats

Nous obtenons les résultats suivants pour le `RNN` :

- L'accuracy sur l'ensemble du train est : 0.88
- L'accuracy sur l'ensemble du test est : 0.73

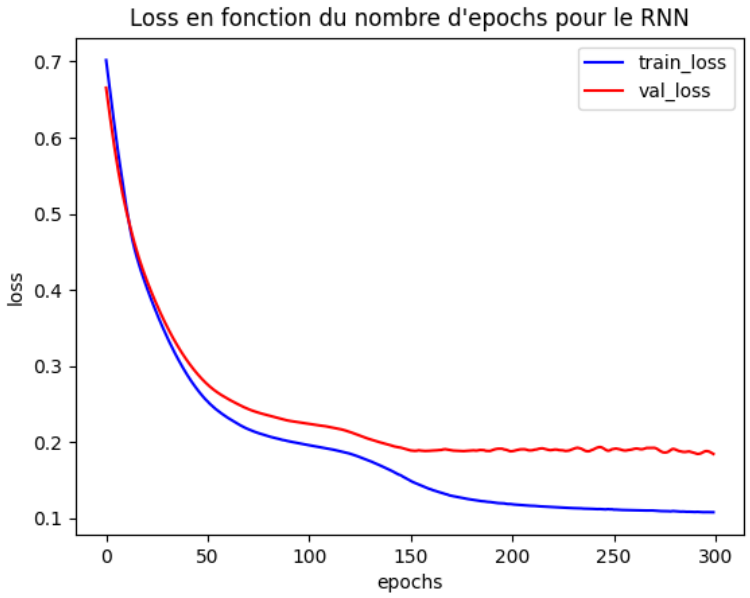


FIGURE 3 – Loss en fonction du nombre d'epochs pour le RNN

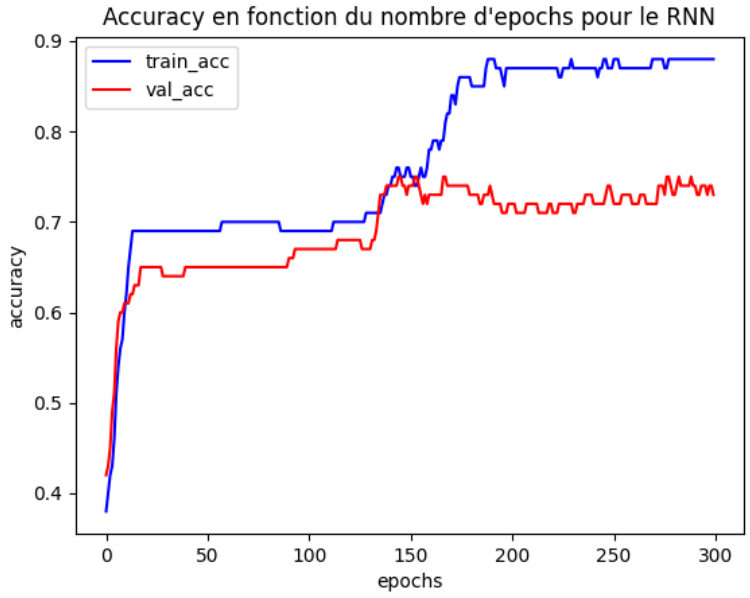


FIGURE 4 – Accuracy en fonction du nombre d'epochs pour le RNN

3.2.4 Interprétation

On constate que la précision diverge de 15 % entre l'ensemble de train et de test ; c'est écart est très important.
On peut alors supposer la présence d'un "sur-entraînement", c'est-à-dire que le modèle a appris des caractéristiques spécifiques au jeu d'entraînement.
Dans le cas général, une précision de 73 % est acceptable.

4 Retours sur les résultats

Rappels. Nous souhaitons rappeler que nous avons été contactés par un hôpital, par conséquence, les exigences sont élevées. En effet, nous parlons ici d'analyse d'ECG pour prévenir l'apparition d'infarctus. Par conséquent, la précision sur des données alors inconnues du modèle doit être grande. Nous pourrions fixer un seuil à 90 % voire 95 %.

4.1 CNN

La précision du `CNN` est confortable avec un écart de seulement 2 %. Néanmoins elle le dépasse pas les 80 %.
Nous pouvons donc y voir un risque pour les patients du service de cardiologie : il se peut que dans 20 % des cas nous passions à côté d'un infarctus. Ceci ne nous semble pas acceptable, car encore une fois, nous parlons ici de vie humaine.

4.2 RNN

L'écart sur la précision est considérable. En effet, il y a eu un sur-apprentissage.
Nous le constatons car la précision sur l'ensemble de données inconnues par le modèle est 15 % inférieure à la précision sur le jeu d'entraînement.
Pour le monde hospitalier, cela présente un risque important de passer à côté d'un infarctus.
Il faudrait modifier les paramètres du modèle afin d'obtenir de meilleures résultats.

5 Choix des paramètres

Pour arriver aux résultats qui vous sont présentés dans ce rapport, nous avons déterminé les paramètres du `CNN` et du `RNN` de manière empirique, après plusieurs tests.

CNN. Nous avons joué sur les hyper-paramètres tels que le taux d'apprentissage, le nombre d'époques et la taille du batch. Nous avons également commencé avec une couche de convolution puis nous sommes ensuite passés à deux. Nous avons également joué sur le nombres de filtres.

RNN. Nous avons également joué sur les hyper-paramètres comme la taille du batch, le nombre d'époque et de neurones.

6 Mesures des résultats

Nous avons choisi de regarder la précision de nos modèles, c'est-à-dire la capacité du système à rejeter les solutions non-pertinente. Mais ce n'est pas la seule façon de valider un modèle. En effet, nous aurions pu mesurer la capacité de nos modèles à donner toutes les solutions pertinentes, appelée "rappel", pour aboutir à une moyenne harmonique de la précision et du rappel, appelée "F-mesure". La F-mesure mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres.
Également nous aurions pu mettre en place une matrice de confusion pour observer sous un autre angle les résultats des modèles.
Nous avons également constaté que nos résultats dépendant de l'utilisation des différentes ressources de nos ordinateurs. Ainsi, les derniers résultats en date ont été obtenus dans les conditions suivantes :

- Applications ouvertes :
  - Mozilla Firefox ;
  - PyCharm Professional ;
  - Texmaker.
- Wi-Fi désactivé ;
- Aucune manipulation durant le déroulement du programme.