Introduction

L’analyse de la fonction cardiaque est essentielle en cardiologie clinique pour le diagnostic des maladies, la prise en charge des patients et la prise de décisions thérapeutiques. L’idée du challenge est se concentrer sur quatre pathologies cardiaques (Infarctus du myocarde ; Cardiomyopathie dilatée ; Cardiomyopathie hypertrophique ; Ventricule droit anormal) qui peuvent passer inaperçues initialement, mais qui peuvent se révéler mortelles. C’es très important de déceler ces problèmes le plut tôt possible pour orienter le traitement et prévenir les complications.

Pour ce qui est du dataset, il y a 150 sujets séparé en 100 sujets d’entraînement et 50 sujets de test. Pour chaque sujet, nous avons à disposition 2 images IRM : la première à la fin de la diastole et la seconde à la fin de la systole. Ce sont des images 3D qui contiennent le cœur et les structures adjacentes. Nous disposons également pour chaque image une segmentation en 4 parties :

0 – Background

1 – Right ventricle cavity

2 – Myocardium

3 – Left ventricle cavity

Pour arriver à classifier ces patients en fonction de leurs maladies, le challenge va se dérouler en plusieurs étapes. Dans un premier temps, il faut améliorer la segmentation des images test car la segmentation fournie ne distingue pas l’arrière-plan et le ventricule gauche. Dans un second temps, il va falloir extraire des features de nos images, notamment grâce aux segmentations. Enfin, il va falloir créer un modèle avec lequel on pourra entrainer nos données et prédire les pathologies des patients test. Pour faciliter l’accès aux données, j’ai créé un dictionnaire pour train et test, avec les images et les segmentations sous forme de tableau python.

Segmentation :

https://www.kaggle.com/competitions/ima2

Une image contenant texte, capture d’écran

Description générée automatiquementLa segmentation est une partie très importante car c’est avec cela que l’on va pouvoir extraire des features de nos images. Sur l’image, on peut voir les quatre zones segmentées. Cependant sur les segmentations des données de test, la zone bleue est considérée comme de l’arrière-plan.

Enfin de segmenter cette partie-là, j’ai remarqué que sur toutes les images, le ventricule gauche est entouré du myocarde. Après quelques recherches, je me suis penché vers la méthode de skimage :

segmentation.flood\_fill()

Cette méthode prend en paramètre la segmentation, et un point intérieur à une forme. Comme son nom l’indique, on va verser de l’eau dans la forme et l’eau va se propager en inondation, tout en restant à l’intérieur de l’objet.

Pour trouver le centre de l’objet, j’ai utilisé la méthode de ndimage :

center1, center2 = ndimage.measurements.center\_of\_mass(data\_ES\_seg[:,:,i]==2)

Où le ’2’ correspond au myocarde, car on veut trouver le centre du cercle vert.

Extraction des features :

Grâce au différents documents fournis, j’ai pu trouver différentes inspirations pour les features à extraire.

Dans un premier temps, les volumes des différentes parties est très important, et notamment la comparaison entre après la diastole et après la systole. Pour cela, il suffisait de prendre : np.sum(np.where(data\_ED\_seg==1, 1, 0))/BSA

Où le BSA (Body Surface Area) permet de normaliser par une combinaison du poids et de la taille. On fait cela pour les 3 zones différentes, dans les deux images. On peut ensuite forcer une feature en donnant le ratio entre les deux.

Ensuite j’ai choisi de calculer l’aire de la sphère de chacune des parties. Pour cela j’ai utilisé la méthode de skimage :

measure.find\_contours()

J’ai choisi de ne pas aller plus loin dans la recherche des features basées sur la forme des images. Je me suis ensuite concentré sur l’intensité des pixels de l’IRM. J’ai donc extrait le niveau de gris moyen ainsi que la variance de chaque zone.

Modèles de machine learning :

Tout d’abord, je voudrais préciser que j’ai séparé mes 100 sujets en sujets d’entraînement et sujet de validation pour faciliter la vision de la pertinence de chaque modèle. Pour ce qui est du modèle de machine learning, je me suis concentré sur deux d’entre eux.

Le premier est le « RandomForestClassifier » (cf class RandomForest:). C’est une méthode basée sur une classification en arbres, mais tout de même assez complexes car il y a plusieurs hyperparamètres, comme le nombre d’arbres, le critère de sélection d’une feature, la profondeur max etc. J’ai décidé de coupler ce modèle avec un PCA. Ayant beaucoup de features, et n’étant pas à 100% sûr de la pertinence des mes features, j’ai entrainé mon modèle pour qu’il trouve la meilleure projection de mes features. Avec ce modèle, j’obtiens des bons scores de validation, autour de 0.88

Le deuxième est le multi-layer perceptron (MLP), qui est un modèle basé sur les réseaux de neurones. Encore plus complexe que le RandomForest, il était d’autant plus difficile de trouver les bons hyperparamètres (le learning\_rate, le nombre d’époques, le nombre de couches, le nombre de composantes par couches etc.). Afin d’utiliser au maximum les ressources Google Colab que nous avions à disposition, j’ai tenté de faire une cross validation couplé d’une GridSearch afin de trouver les meilleurs hyperparamètres pour mon modèle. Pour cela, j’ai utilisé KerasClassifier de la librairie scikeros.wrappers, mais sans succès. J’ai rencontré de multiples erreurs que je n’ai pas réussi à résoudre. J’ai donc opté pour une recherche plus manuelle des hyperparamètres. Comme on entrainre sur peu de données, il faut faire attention à pas trop overfitter notre modèle. Il ne faut donc pas prendre trop de couches, ni trop de composantes par couches. Concernant le learning rate et le nombre d’époques, je suis parti sur un très petit learning rate (10e-4) et il faut donc beaucoup d’époques pour tomber sur le minimum de notre fonction convexe (j’ai choisi 400). Avec ce modèle, j’obtiens un score correct de 0.76

Conclusion :

Afin de faciliter le test des différents modèles, faire des classes m’a paru une bonne idée, et permet d’alléger un peu le code qui devient vite lourd.

Après analyse des deux modèles que je viens de présenter, j’ai préféré partir sur le RandomForestClassifier. C’est un modèle plus simple, mais comme nous n’avons que 100 données d’entraînement, il faut faire très attention à l’overfitting. A mon avis, c’est donc mieux de choisir un modèle moins complexe, qui se prêtera mieux à nos données. A l’inverse, le MLP est, je pense, trop complexe, et il y a donc de l’overfitting sur les données d’entrainement, ce qui dégrade le score sur les données de validation.

Avec le RandomForestClassifier couplé avec un PCA, j’obtiens un score de 0.82857 sur le Kaggle, ce qui est plutôt un bon score même s’il pourrait être largement amélioré. Pour cela, une piste d’amélioration serait d’étudier plus en détail la pertinence des features avec les papiers de recherche afin d’uniquement sélectionner les features qui jouent un vrai rôle de la diagnostiquassions des maladies cardiaques.