Stage

Licence 3 MIASHS - IDS (2021-2022)

Julien Bastian

Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2

Stage réalisé au sein du Laboratoire Eric
Encadrant : Guillaume Metzler
Tuteur enseignant : Stéphane Chrétien

Résumé

L'objectif de ce travail est d'explorer des méthodes d'analyses de données fonctionnelles avec une application pour la classification de battements de coeur. Dans ce cadre nous étudierons d'abord les réseaux de neurones convolutifs. Nous nous appuierons alors sur un travail existant qui vise à construire un réseau transférable et proposerons une amélioration de la capacité de généralisation de celui-ci. Puis, nous nous intéresserons aux méthodes d'analyses des signaux en nous concentrant sur la transformation en ondelettes continues. Avec l'étude d'une publication qui les emploie pour la classification d'arythmie.

Table des matières

1 Introduction

Les électrocardiogrammes (ECG) sont un outil important dans de nombreux domaines biomédicaux. Ils permettent de mesurer et d'enregistrer l'activité électrique du coeur pour en faire l'analyse. On pourra ainsi les utiliser pour detecter des maladies cardiaques mais aussi pour faire de la reconnaissance d'émotion ou encore de l'identification biométrique.
Pour réaliser un ECG, on utilise plusieurs électrodes qui vont mesurer les impulsions electriques correspondantes à chaque battement. Le nombre et le placement de ces électrodes permet au maximum l'extraction de 12 mesures, appelées dérivations qui enregistrent toutes le même signal mais selon des angles différents offrant donc la possibilité de localiser les éventuelles anomalies.

Dans ce travail il s'agira d'explorer des méthodes de détection d'anomalies dans les battements de coeur. En effet, le developpemment de techniques solides et rigoureuses de classifications de ces anomalies est un outil de première importance dans le cadre de la prévention de maladies cardiovasculaires qui sont la première cause de morts à l'échelle mondiale [1]. D'autant que les méthodes explorées ici ont vocations à être utilisables sur un grand nombre de données, remplaçant l'oeil expert du médecin. La figure 1 donne un exemple des battements que nous étudierons, ici il s'agit d'un battement sain et d'un battement avec arythmie, i.e. une anomalie de la fréquence cardiaque, provenant d'un des jeux de données que nous allons employer.

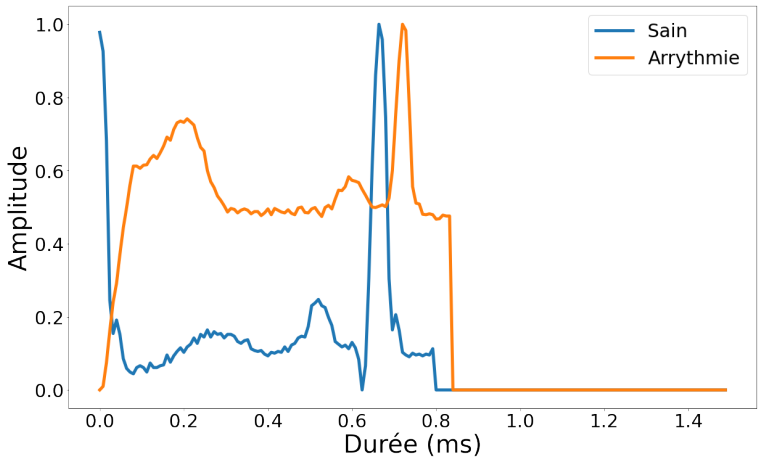

FIGURE 1 – Battements à différencier, en bleu un battement sain et en jaune un battement avec arrythmie.

Les données issues d'ECG sont des dites fonctionnelles. Chaque enregistrement peut être assimilé à une courbe bien que les valeurs extraites soient sous la forme de vecteurs, ce qui implique des techniques d'analyses propres. Ici, nous allons adopter plusieurs approches dans une perspective exploratoire. Nous aurons ainsi, à la fin de ce travail, une idée plus large des techniques d'analyses de données fonctionnelles et de leur application dans le cadre des ECG.
Nous étudierons d'abord les réseaux de neurones profonds et en particulier les réseaux de neurones convolutifs qui sont les plus adaptés à notre problème. Ces réseaux, lorsqu'ils sont bien construits, permettent une analyse solide et transférable grâce à leur haut niveau d'abstraction. Ce caractère transférable est particulièrement intéressant dans la mesure où il permet à une

1. D'aprés l'organisation mondiale de la santé, voir www.who.int

même architecture d'être appliquée à plusieurs jeux de données en ne requérant qu'une adaptation minime de sa construction. Dans ce cadre, le réseau comportera deux parties : d'abord il s'agit d'extraire des caractéristiques puis de réaliser une tache de classification sur les caractéristiques extraites. La figure 2 illustre schématiquement ce fonctionnement. Pour construire un réseau transférable il faudra donc être capable d'extraire une représentation rigoureuse du vecteurs d'entrée, afin d'appliquer les couches qui réalisent cette extraction à de nouvelles données.

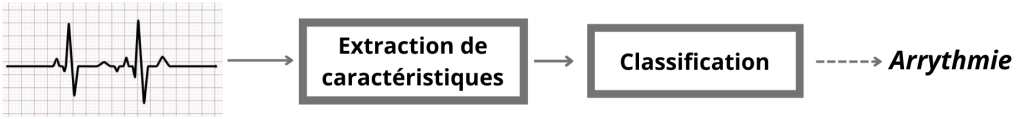

FIGURE 2 – Exemple des étape du réseau, on extrait d'abord des caractéristiques puis on réalise une tache de classification sur celles-ci.

Il existe plusieurs exemples d'utilisations de réseaux de neurones convolutifs pour la classification d'ECG, on citera par exemple [Weimann and Conrad, 2021] avec une approche similaire à celle que nous allons explorer. Dans notre cas nous commencerons par reproduire les résultats obtenus en [Kachuee et al., 2018], à la fois sur un premier jeu de données d'entraînement puis en transfert. Puis, nous proposerons une piste pour réduire la complexité de cette architecture sans sacrifier ni ses performances, ni sa transferabilité, voire en les améliorant.

Ensuite, nous aborderons notre sujet du point de vue de l'analyse de signaux en explorant la transformation en ondelettes continue. À l'image de la transformation de Fourier cet méthode nous permet d'obtenir une décomposition fréquentielle de notre fonction d'origine. Mais à la différence de celle-ci la transformation en ondelettes offre une bonne résolution à la fois en temps et en fréquence, plutôt que simplement une description fréquentielle. Ce qui est nécessaire pour analyser des ECG et des battements de coeur car ils ne forment pas des fonctions monotones. D'autant que les caractéristiques locales de ces signaux sont particulièrement précieuses, permettant par exemple de détecter les anomalies correspondant à une crise d'arythmie.
L'utilisation d'ondelettes pour l'analyse d'ECG est déjà répandu [Li, 2018], mais principalement dans le cadre de la transformation en ondelettes discrète. Il s'agira dans notre cas d'employer la transformation en ondelettes continue en reproduisant les résultats présentés en [Wang et al., 2021]. On transformera les battements de coeur qui se définissent dans le temps par leur transformée en ondelettes sur deux dimensions, le temps et les fréquences. Puis nous utiliserons un réseau de neurones convolutif pour la classification finale, ceux-ci étant particulièrement adaptés aux données bidimensionnelles.

2 Présentation des outils

On se placera dans le cadre d'une classification multi-classes, avec C le nombre de classe, où les données sont issues d'une distribution inconnue $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ où $\mathcal{X}$ représente la distribution des features et $\mathcal{Y}$ celle des labels. On considèrera un échantillon $S = \{\mathbf{x}_i, y_i\}_{i=1}^m$ de taille m où $\mathbf{x}_i \in \mathbb{R}^d$ est le vecteur de dimension d des features de l'individu i et $y_i \in \{1, \cdots, C\}$ son étiquette

ou *label*.
On note, les nombres en lettres minuscules : x, les vecteurs en gras : $\mathbf{x}$, avec $\mathbf{x} = \{\mathbf{x}_i, ..., \mathbf{x}_n\}$, et les matrices en majuscules : $\mathbf{X}$.

2.1 Réseaux de neurones convolutifs

Généralités sur les Réseaux de neurones

Pour comprendre les réseaux de neurones convolutifs (en anglais, convolutionnal neural networks, CNN) il est nécessaire d'avoir une vision de ce qu'est l'apprentissage profond à travers les réseaux de neurones profonds (en anglais, deep neural networks, DNN).

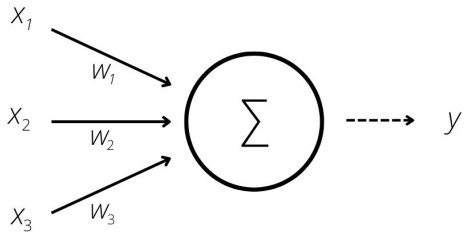

FIGURE 3 – Le perceptron

L'élément de base des réseaux de neurones est le perceptron, illustré par la figure 3, c'est une fonction qui prend en entrée un vecteur $\mathbf{x}$ et donne en sortie un réel y en effectuant une somme pondérée des éléments de $\mathbf{x}$ par un vecteurs de poids $\mathbf{w}$ et en ajoutant un biais b. On a donc une fonction linéaire, à laquelle on rajoute une fonction d'activation $g(\cdot)$ non linéaire, qui peut prendre plusieurs formes. C'est d'elle que dépend la sortie du neurone, son activation.

$$\Theta(\mathbf{x}, \mathbf{w}) = g(\sum_{i=1}^{n} x_i w_i + b) = z.$$

En Figure 4, quatre fonctions d'activation courantes avec leurs expressions : sigmoïde, tangente hyperbolique, Rectified Linear Unit (ReLU), Leaky ReLU. Aujourd'hui, la ReLU est une des fonctions d'activations les plus utilisées [Nair and Hinton, 2010].

Pour construire un réseau, on va agréger ces perceptrons en couches denses où tous les perceptrons seront connectés aux données de la couche précédente. Les poids de la couche sont donc regroupés sous la forme d'une matrice $\mathbf{W}$, avec les poids de chaque perceptron en lignes et on obtiendra en sortie un vecteur de réel $\mathbf{y}$ qui contient une valeur pour chaque perceptron. On peut résumer le traitement d'une couche dense par la formule suivante,

$$\Theta(\mathbf{x}, \mathbf{W}) = g(\mathbf{W}\mathbf{x}) = \mathbf{z}.$$

A partir de ces couches denses on crée des réseaux multicouches en les superposant (Figure 5 une illustration d'un réseau multicouches). Les valeurs prises en entrée par la couche i seront

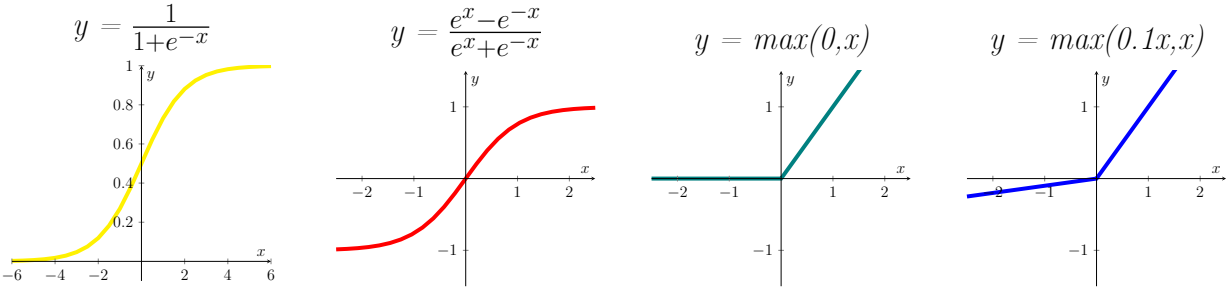

FIGURE 4 – Les principales fonctions d'activation, de gauche à droite : la sigmoïde, la tangente hyperbolique, la ReLU et la Leaky ReLU.

les valeurs de sortie de la couche précédentes, la couche $i-1$, ce qui revient à combiner les fonctions de toutes les couches. On obtient :

$$\Theta_i(\Theta_{i-1}(\mathbf{x})) = g_i(\mathbf{W}_i(g_{i-1}(\mathbf{W}_{i-1}\mathbf{x})) = \mathbf{z}_i.$$

Avec $\mathbf{z}_i$ le vecteur en sortie de la couche dense i. Il est nécessaire de comprendre cette imbrication de fonctions pour comprendre le fonctionnement de la descente de gradient qui est au coeur de l'apprentissage d'un DNN comme on le verra plus loin. En approfondissant ainsi notre architecture on la complexifie pour lui permettre de résoudre des problèmes plus exigeants, bien que le nombre de couches et de neurones doit être controlé pour éviter le surapprentissage.

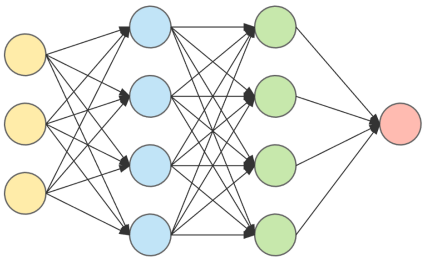

FIGURE 5 – Un réseau multicouches avec deux couches cachées entièrement composee 4 neurones chacune. [2]

Réseaux convolutifs

A ce stade, il manque encore une brique importante pour construire un CNN, les convolutions. Le produit de convolution est la fonction mathématique qui prend en entrée une matrice ou un vecteur et qui fait glisser sur ces données un masque de convolutions afin d'extraire une carte de caractéristiques(comme en Figure 6). Ainsi, le résultat de ces convolutions n'est pas un réel z comme pour le perceptron mais une matrice $\mathbf{Z}$ ou un vecteur $\mathbf{z}$.

En plus des convolutions on ajoute un opérateur de sous-échantillonnage (pooling en anglais), qui va résumer l'information contenue dans une carte de caractéristiques et ainsi réduire encore la dimension de nos données. Ce sont également des fenêtres glissantes. Il existe trois méthodes principales pour réaliser ce sous-échantillonnage : en récupérant la valeur maximale de la fenêtre

2. source : upgrad.com

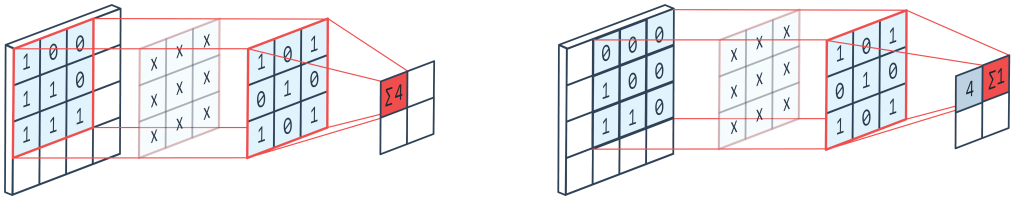

FIGURE 6 – Illustration d'une convolution, le masque de convolution se déplace sur les données d'entrée pour en extraire une carte de caractéristiques [3].

(max pooling), en récupérant la somme des valeurs (sum pooling) ou en encore en faisant la moyenne (average pooling). La plus répandue étant le max pooling.

L'utilisation de convolutions plutôt que de simples couches denses offre une meilleure conservation de l'information contenue dans les données à chaque étape. De plus, la nature spatiale des convolutions (le masque se "déplace" sur les données comme illustré par la Figure 6) permet d'extraire des informations locales sur les données ce qui est nécessaire pour faire de l'analyse de fonctions.
Ces caractéristiques font des CNN l'outil de choix pour faire de l'analyses de données fonctionnelles à l'aide de DNN.

Prédictions et entrainement du modéle

Il est important de noter qu'il sera nécessaire d'employer une opération particulière à la fin de notre modèle.Pour réaliser une classification multi-classes nous utiliserons la fonction softmax. Cette fonction prend en entrée un vecteur de dimension C, avec C classes possibles, et sort un vecteur de C réels qui somment à 1, *i.e.* un vecteur de probabilités $\mathbf{p}$ d'appartenance à chaque classe dont les éléments sont définis par :

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j}^{C-1} \exp(z_j)}, \quad \forall i = 1, \ldots, C.$$

Maintenant que nous pouvons construire notre réseau il reste encore à l'entrainer. En effet, sa performance dépend de la bonne définitions des poids w présentés précédements, que nous regroupons en une matrice $\mathbf{W}$. On utilisera de façon équivalente $\mathbf{W}$ et θ. pour signifier les poids/paramétres du réseau. On peut alors representer un réseau de la façon suivante,

$$\Theta(\mathbf{x}, \mathbf{W}) = \Theta(\mathbf{x}, \theta) = \hat{y}.$$

Où $\hat{y}$ designe le label prédit par le modéle, si l'on utilise la fonction softmax il correspond à la classe pour laquelle la probabilité d'appartenance est la plus élevée. Pour mesurer la performance de notre modèle on introduit une fonction de perte (ou fonction objectif) de la forme : L qui depend des descripteurs $\mathbf{x}$, du label associe y et des paramètres du modele θ. Il

3. source : peltarion.com

existe plusieurs fonctions de perte, on emploira ici l'entropie croisée, qui est la plus adapté dans les taches de classification multiclasse, dont la formule est la suivante :

$$L(\mathbf{x}, \mathbf{y}, \theta) = -\sum_{c=1}^{C} y_c \log(p_c)$$

Pour trouver les paramétres qui minimisent cette fonction de perte on utilise l'algorithme de descente de gradient. Il nous permettra de mettre à jour itérativement les paramètres selon un pas d'apprentissage η.

$$\theta = \theta - \eta \times \nabla_\theta L(\mathbf{x}, \mathbf{y}, \theta)$$

Il existe un certain nombre d'algorithme qui permettent d'optimiser la descente de gradient. Un des plus important, et que nous utiliserons, est l'Adapative Moment Estimation (Adam) [Kingma and Ba, 20 dont l'algorithme est détaillé ci-dessous. Il s'agit de calculer la moyenne et la variance du gradient. En considèrant $g_{t,i}$ comme étant le gradient de la fonction L au temps t par rapport au paramètre θ_i, on a l'algorithme 1. Malgré l'inconvénient de rajouter trois hyper-paramètres cet algorithme permet une franche amélioration de la descente de gradient. Il est donc largement utilisé lors de l'apprentissage de réseaux de neurones.

Algorithm 1: Adaptative Moment Estimaton (Adam). On utilisera quasiment toujours les valeurs des hyper-paramètres β_1, β_2 et ϵ conseillées par les auteurs : $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

Require: θ_0 : vecteur initial des paramétres
  $m_0 \leftarrow 0$
  $v_0 \leftarrow 0$
  $t \leftarrow 0$
  while θ_t pas convergé **do**
    $t \leftarrow t + 1$
    Calculs de la moyenne et de la variance du gradient
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    Normalisation de la moyenne et de la variance
    $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
    $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
    Mise à jour des paramètres
    $\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$
  end while

Après un certain nombre d'itérations de la descente de gradient sur des données d'entrainement, on considére que l'entrainement est terminé. Si le réseau offre des résultats satisfaisants sur de nouvelles données qu'il n'a jamais rencontré alors il est prêt à être employé à plus grande échelle.

2.2 Transformée en ondelettes

Généralités

La méthode la plus répandue en analyse du signal est la transformation de Fourier. Elle consiste à décomposer une fonction en les différentes fréquences qui la compose. On passe alors d'une représentation du signal dans le temps à une représentations en fréquences. Cependant, cette méthode présente un défaut majeur. Sa résolution temporelle est nulle, le résultat de la transformation de Fourier n'apporte aucune indication sur la localisation des variations dans la fonction initiale. Elle n'est pas adaptée à des signaux non-stationnaires comme nos données. Pour extraire des caractéristiques discriminantes des signaux à notre disposition il est nécessaire d'appliquer une transformation qui offre une bonne résolution dans le temps tout en extrayant les informations fréquentielles importantes. Une piste crédible pour cela est la transformation de Fourier à court terme. Elle consiste à opérer la transformation de Fourier sur plusieurs intervalles du signal. On obtient alors une décomposition en fréquence sur chaque intervalle de temps. On perd en résolution dans le domaine fréquentiel mais on en gagne dans le domaine temporel. Une autre piste est la transformation en ondelettes continues qui a une haute résolution en temps et fréquence (pour plus d'informations sur la comparaison entre transformation de Fourier et en ondelettes voir [Sifuzzaman et al., 2009] et [Lin, 2007]). La figure 7 donne une comparaison des résolutions temps-fréquence des trois méthodes évoquées.

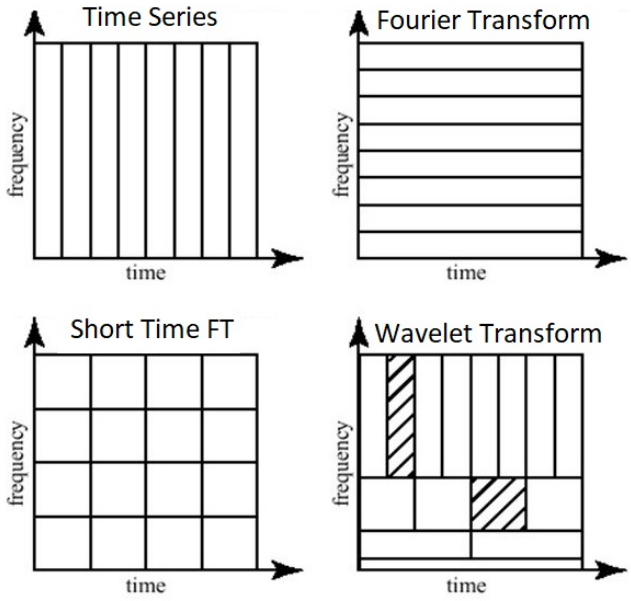

FIGURE 7 – Comparaison des résolutions en temps et fréquences entre la transformation de Fourier, la transformation de Fourier à court terme et la transformation en ondelettes. À basse fréquence la transformation en ondelettes a une excellente résolution fréquentielle mais faible résolution temporelle, à haute fréquence c'est l'inverse [4].

Pour ces qualités et parce qu'elle est déjà répandue pour l'analyse de données fonctionnelles nous allons étudier ici la transformation en ondelettes continues. Pour une présentation globale de l'usage des ondelettes pour la classification d'ECG voir [Li, 2018].

Les ondelettes

4. Illustration tirée de *A guide for using the Wavelet Transform in Machine Learning*.

On définit une ondelette comme une fonction Ψ qui dépend du temps t, qu'on appelle ondelette mère. Il en existe de nombreuses, la figure 8 en illustre deux des plus importantes, celle de Ricker [Liu et al., 2005] (aussi appelée *mexican hat*) et celle de Morlet [Lin and Qu, 2000].

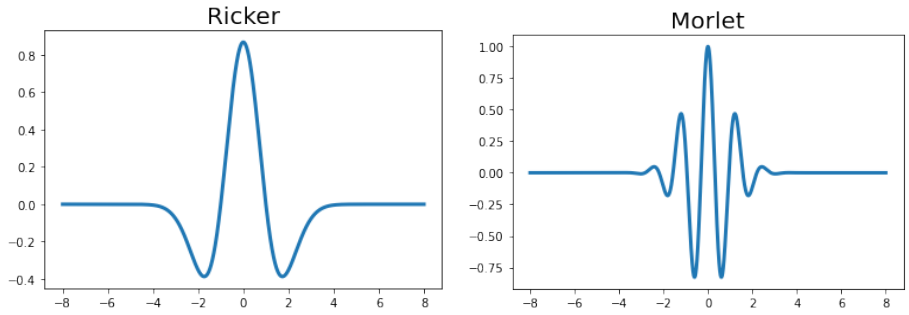

FIGURE 8 – À gauche l'ondelette de Ricker (ou *mexican hat*) et à droite celle de Morlet. Elles sont largement utilisées pour la transformation en ondelettes continue.

À partir de l'ondelette mère Ψ on peut former une famille $\psi_{s,\tau}$ telle que,

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right).$$

Cette famille correspond à la wavelet mère selon les paramètres τ et s qui contrôlent respectivement la translation et la dilatation. C'est grâce à qu'on réalise la transformation. Le lecteur désireux d'en savoir plus sur la transformation en ondelettes est invité à consulter les références [Lin, 2007] et *a guide for using the Wavelet Transform in Machine Learning*.

Transformée en ondelette continue

La transformée en ondelette continue désigne le résultat de la transformation d'une fonction d'origine par convolution avec une famille d'ondelettes $\psi_{s,\tau}$. Pour une fonction $f \in \mathbf{L}^2(\mathbb{R})$ on définit sa transformée g par :

$$g(s,\tau) = \int_{-\infty}^{+\infty} f(t)\psi_{s,\tau}(t)\mathrm{d}t.$$

La translation permet de parcourir l'ensemble du signal, de réaliser une convolution. La dilatation elle, va nous permettre de tester sa réponse par rapport à plusieurs échelles de l'ondelette. Échelles que l'on peut convertir en fréquence par la formule suivante $freq = \frac{C_f}{s}$, où $freq$ est la fréquence, C_f est la fréquence centrale de l'ondelette mère et s est le facteur de dilatation. On transforme donc le signal f, originalement en une dimension, en sa transformée g, en deux dimensions. Ces dimensions étant le temps et la fréquence. On passe alors d'une représentation temporelle de nos données à une représentation temps-fréquence, apportant un gain d'information important, en nous donnant accès à de nouvelles caractéristiques. La figure 9 donne un exemple de la représentation obtenue par transformation en ondelettes continues.

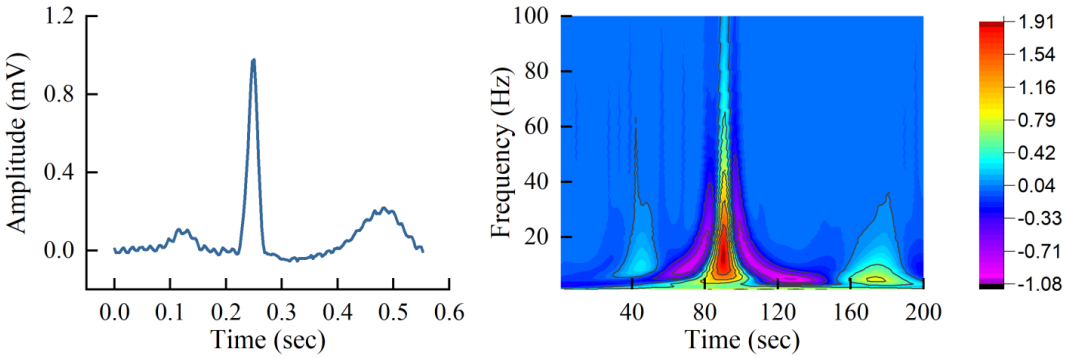

FIGURE 9 – Passage d'une représentation temporelle du signal à une représentation temps-fréquence. À gauche le signal d'origine à sa droite sa transformée continue [5].

La représentation ainsi extraite est le résultat de la convolution de l'ondelette sur le signal initial pour un ensemble de fréquences pertinent. On pourra ensuite l'utiliser dans une tâche de classification. Pour cela, plusieurs méthodes sont envisageables parmi lesquels les réseaux de neurones convolutifs qui sont tout à fait appropriés aux traitement de ces données à deux dimensions. C'est cette piste qui a été choisie, pour la classification d'ECG, par [Wang et al., 2021] et que nous allons étudier en 3.2 et 4.3.

Transformée en ondelettes discrètes et autres applications

Nous n'avons présenté ci-dessus que la transformée en ondelettes continues et son application pour extraire des caractéristiques du signal car c'est l'usage que nous allons en faire dans ce travail. Cependant, il est bon de rappeler que les ondelettes peuvent être utilisées dans d'autres cadres et en particulier grâce à la transformée en ondelettes discrètes. Deux cas d'usages sont très répandus, le débruitage et la compression de données. Ils sont rendus possible par le fait que la transformation discrète donne une représentation éparse des données original. Ce qui signifie que les caractéristiques importantes du signal seront concentrées en un nombre réduit de coefficients. Cette concentration de l'information permet un traitement efficace du bruit. Mais surtout la transformée discrète offre un résumé rigoureux du signal, permettant ainsi sa compression [Lewis and Knowles, 1992]. On citera par exemple le format JPEG2000 qui utilise la transformation en ondelettes.

3 Contribution

3.1 CNN

Il s'agit ici de reproduire le réseaux de neurones convolutif proposé en [Kachuee et al., 2018]. Le but est de construire un CNN transférable pour la classification de battements de coeur. On peut décomposer le réseau en deux parties. D'abord, des couches de convolutions qui vont apprendre une représentation des données, c'est là le caractère transférable du modèle. En effet, une fois l'entraînement terminé on gèlera les paramètres de cette partie pour l'utiliser sur des nouveaux jeux de données. Ensuite, on a un prédicteur qu'on devra adapter et entraîner

5. Illustration tirée de [Wang et al., 2021].

spécifiquement pour chaque nouvel ensemble de données.
On a donc une partie du réseau qui va apprendre une représentation des données, que l'on va transférer et une partie dont le rôle est de classifier les données à partir de la représentation apprise et que l'on devra adapter pour chaque usage.
L'intérêt de construire un réseau transférable est de pouvoir transférer des connaissances apprises sur un jeu de données à un autre. Ici, la représentation apprise sur le jeu de données initial pourra être utilisée sur de nouveaux enregistrement, même si l'ensemble étudié ne contient que peu de battements. Les cas d'usages de réseaux transférables sont nombreux et leur utilité n'est plus à prouver pour analyser de nouvelles données même lorsqu'elles sont peu nombreuses.

Architecture

L'architecture proposée dans l'article est illustrée en Figure 10. On a une répétition de cinq blocs. Chacun de ces blocs contient deux couches de convolutions à une dimension. A chaque couche on a 32 filtres avec des masques de convolution de taille 5. Après la deuxième couche de convolutions on a une shortcut connection, elle permet d'ajouter à la carte de caractéristique obtenue une carte de caractéristique antérieure, simplifiant ainsi l'optimisation du réseau [He et al., 2015]. Ici la carte de caractéristique qu'on ajoute est celle que le bloc prend en entrée. Enfin, on a une couche de sous-échantillonage, en l'occurrence de max-pooling, de taille 5 et de pas égal à 2.
A l'issue de ces 5 blocs on a un prédicteur composé de 2 couches denses de 32 neurones chacun et d'une couche softmax. Pour adapter ce prédicteur à chaque nouvel ensemble de données il faudra modifier la dimension du softmax pour correspondre aux nombres de classes à prédire et réinitialiser les paramètres des deux couches denses.

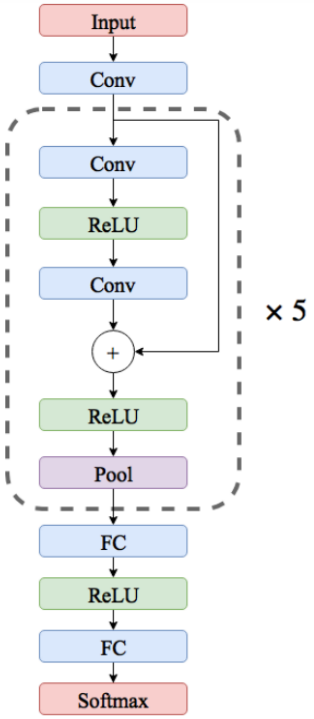

FIGURE 10 – Architecture du réseau, on a une répétition de cinq blocs qui vont apprendre une représentation des données puis le prédicteur avec des couches denses et une couche softmax [6].

Ma Proposition

La problématique à laquelle nous souhaitons apporter une réponse est celle des performances du modèle lorsqu'il est transféré à de nouvelles données. En effet, comme on le verra dans la partie 4.2 les performances en transfert, bien que bonnes peuvent être améliorées. Pour cela il est nécessaire d'augmenter la capacité de généralisation du réseau, c'est-à-dire éviter que la représentation apprise soit trop adapté aux données d'entraînement.

6. Illustration tirée de l'article d'origine [Kachuee et al., 2018].

Pour répondre à cette problématique nous allons introduire deux modifications dans l'architecture précédente. D'abord nous allons réduire sa profondeur en passant de 5 blocs à 3, réduisant le nombre de couche entraînables de 13 à 9. L'idée derrière cette modification est de limiter la spécialisation de chaque neurone. En effet, plus leur nombre est faible plus chacun devra apprendre des caractéristiques générale, ce qui va dans le sens d'améliorer la capacité du réseau à s'adapter à de nouvelles données.
De plus nous allons introduire du dropout. Cette technique qui consiste à désactiver aléatoirement certains neurones pendant l'apprentissage est reconnue pour apporter une amélioration significative à la capacité de généralisation d'un réseau [Srivastava et al., 2014]. Dans le cas présent il s'agira de désactiver 20% des neurones dans les couches de convolutions à chaque itération de l'entraînement.
La reproduction des résultats de [Kachuee et al., 2018] et leur comparaison avec l'architecture que nous proposons sont présentées dans la partie 4.2

3.2 Ondelettes

Pour notre exploration de la transformée en ondelettes nous allons reproduire les résultats de [Wang et al., 2021]. L'idée ici est de faire un usage joint des caractéristiques extraites par la transformation en ondelettes continue et de caractéristiques médicales connues sur les battements de coeur, pour réaliser notre classification. Ces caractéristiques médicales sont construites à partir du point d'amplitude maximum du battements (qu'on appelle R-peak) et des intervalles entre ces points (RR-intervalls). On en emploiera quatre :

- Previous-RR, l'intervalle entre le battement courant et le précédent.
- Post-RR, l'intervalle entre le battement courant et le suivant.
- Ratio-RR, le ratio entre previous-RR et post-RR.
- Local-RR, la moyenne des dix RR-intervals précédents le battement courant.

Ces quatre indicateurs sont reconnus pour être modifiés lors des crises d'arythmie et apporterons donc des informations pertinentes.
La tache de classification est réalisée par un réseau de neurones convolutifs. Plusieurs couches de convolutions vont extraire les caractéristiques de la transformée en ondelette puis on ajoutera les informations sur les RR-intervals avant le classifieur final. L'utilisation d'un réseau de neurones convolutifs permet de faire plein usage des deux dimensions des battements transformés.

Architecture

L'architecture présentée est illustrée par la figure 11. On a trois couches de convolutions équipées d'une ReLU comme fonction d'activation. Chacune est suivie par une couche de max-pooling. Leur rôle est d'extraire les caractéristiques pertinentes des matrices correspondant aux transformée en ondelettes données en entrée. À la suite de cette extraction on va concaténer les données relatives aux RR-intervals. Puis, la classification est réalisée par deux couches denses. La première contient 32 neurones et la seconde 4, i.e. le nombre de classes de nos données.
La reproduction des résultats présentés en [Wang et al., 2021] est disponible dans la partie 4.3.

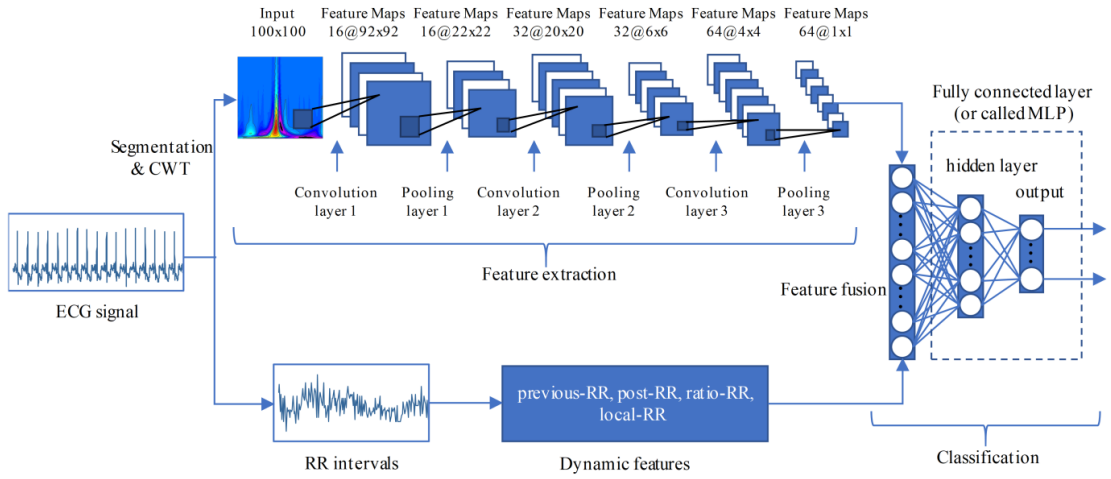

FIGURE 11 – Architecture du réseau, les couches de convolutions extraient les caractéristiques de la transformée en ondelette puis deux couches denses vont réaliser la classification.

4 Expériences

4.1 Données

Nous allons utiliser deux bases de données d'électrocardiogrammes étiquettés pour nos expériences. La première est la Physionet MIT-BIH Arrythmia database (qu'on abrégera en MIT-BIH). Elle contient les électrocardiogrammes de 47 sujets organisés en cinq classes avec une classe de battement sain et quatre classes d'arythmie. La seconde est la PTB Diagnostic (qu'on abrégera en PTB) database avec 290 sujets, 52 sont sains, 148 ont été enregistrés pendant une crise cardiaque et les 90 restant présentent sept maladies différentes. Pour laquelle on regroupera les enregistrements des crises cardiaques et des sept maladies dans la même classe "anormal". La figure 12 donne un exemple des données à classifier.
Les bases de données sont accessibles librement, MIT-BIH Arrythmia, PTB diagnostic.

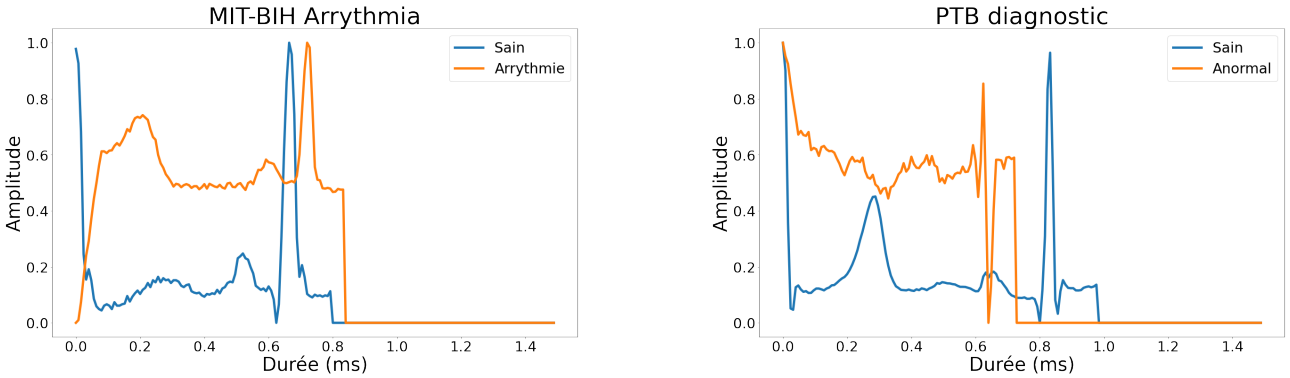

FIGURE 12 – Exemple de battements à classifier, ces battements ont été découpé à partir d'enregistrement d'ECG plus long. À gauche deux battements issus de la MIT-BIH Arrythmia database, avec en bleu un battement sain et en orange un battement avec arythmie. À droite deux battements provenant de la base PTB diagnostic, avec en bleu un battement sain et orange un battement anormal.

4.2 Réseaux de neurones convolutifs

4.2.1 Données et protocole expérimental

Préparation des données

À partir des jeux de données précédents on va découper les ECG en battements pour y opérer notre classification. Les étapes de cette préparation sont détaillées en [Kachuee et al., 2018]. Après ce découpage on va également augmenter nos données pour équilibrer le nombre de battements dans chaque classe, en les dilatant et amplifiant aléatoirement.
Pour la construction de nos jeux de données on prend 819 battements par classes de MIT-BIH et 500 par classes de PTB.

Reproduction et proposition

Les expériences ont été réalisé sur le langage python et le réseau à été construit avec la bibliothèque Tensorflow. Pour l'entraînement du réseau original, que nous souhaitons transférer, on utilise l'entropie croisée comme fonction objectif, des batch de taille 500, l'optimiseur Adam, avec les paramètres recommandés : $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, et on programme l'entraînement pour 75 epochs. Au transfert sur les données de PTB nous gelons les paramètres de la partie du réseau censée apprendre une représentation des battements (les blocs de convolutions) et nous remplaçons le prédicteur précédent par un nouveau, dont la dimension de sortie est adapté au nombre de classes et qui n'est pas encore entraîné. On utilisera les mêmes paramètres pour l'entraînement à l'exception des batch qui seront de taille 100.
Il est à noter que les résultats présentés pour la reproduction et la proposition sont issus d'une seule exécution du code.

4.2.2 Résultats

Les résultats obtenus à la reproduction sur MIT-BIH correspondent à ceux présentés dans l'article malgré une légère différence, probablement due à un écart dans les paramètres d'apprentissage et dans la méthode d'augmentation des données. Également, les résultats de notre proposition sont équivalents à ceux de la reproduction sur MIT-BIH. Ce qui est rassurant quand à la viabilité de nos modifications. Cependant, notre objectif est d'améliorer la capacité de transfert du modèle, c'est donc ses performances sur PTB qui nous intéressent le plus. La table 1 présente les résultats sur MIT-BIH.

Work	**Approche**	**Accuracy**(%)
Proposition	**CNN avec 3 blocs et dropout**	**96**
Kachuee et al.	CNN avec 5 blocs	96

TABLE 1 – Précision satisfaisante sur MIT-BIH. Notre proposition ne semble pas réduire les performances sur les données initiales.

En transfert, nos performances en reproduction correspondent à nouveau avec celles de l'article. Surtout on obtient une nette amélioration de notre proposition par rapports aux résultats originaux. L'accuracy ainsi que la precision et le recall sont meilleurs, comme présenté dans la table 2. De plus quand on compare l'évolution de l'accuracy et de la loss pendant l'entraînement entre notre proposition et la reproduction, on remarque une convergence plus rapide, bien que moins stable, pour le réseau avec dropout (figure 13). L'instabilité de l'évolution de ces indicateurs peut-être imputé à l'effet du dropout. En effet, en désactivant certains neurones lors de chaque traversée on perd une partie des informations apprises précédemment que les autres vont devoir s'adapter pour compenser. C'est d'ailleurs pour cette raison que le dropout permet d'améliorer la généralisation d'un modèle. Ainsi, d'une epoch à l'autre on peut avoir des neurones importants qui seront désactivés entraînant une baisse de performance par rapport à l'itération précédente. Ce qui induit la variabilité constatée.

Work	**Accuracy**(%)	**Precision**(%)	**Recall**(%)
Proposition	**99**	**99**	**99**
Kachuee et al.	94.5	94.5	94.5

TABLE 2 – Résultats sur PTB en transfert. On voit une amélioration des performances par rapport au modèle initial.

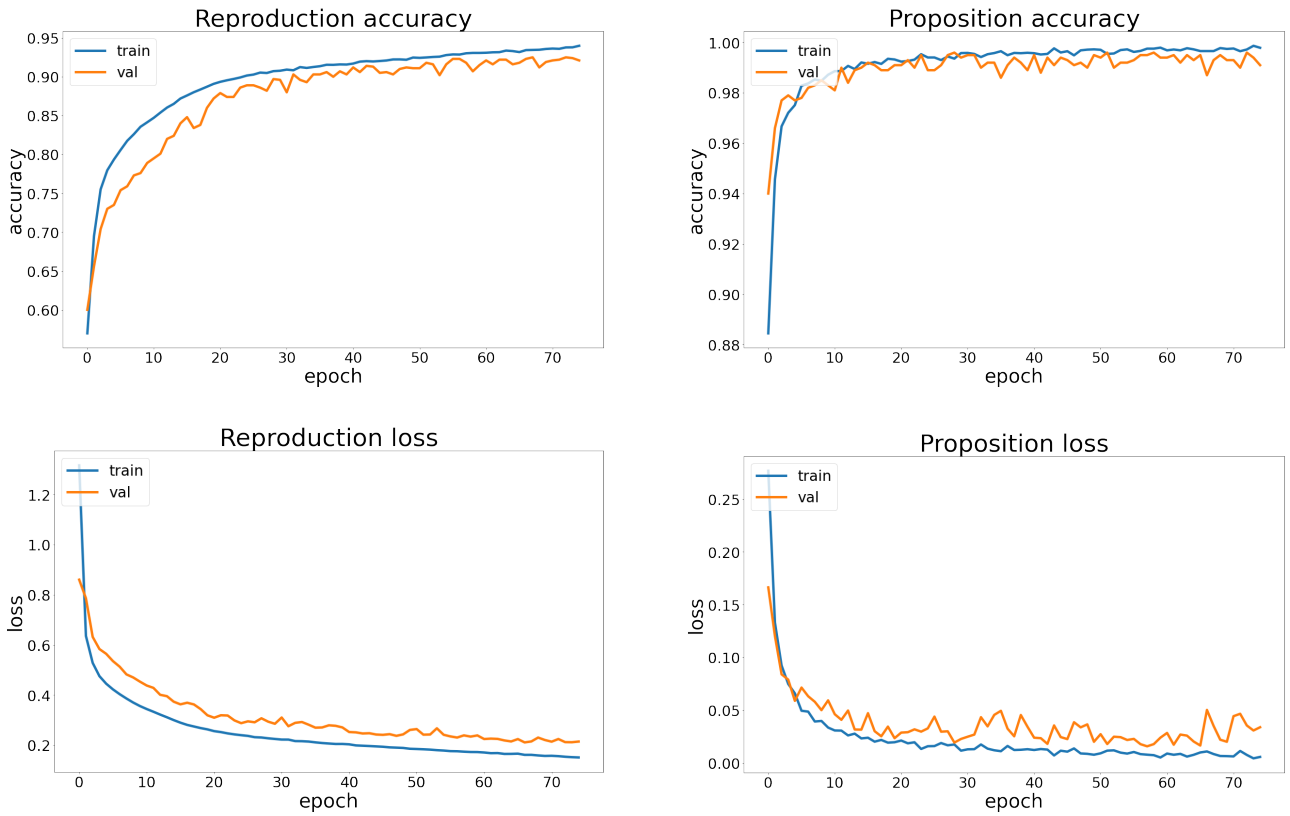

FIGURE 13 – Comparaison entre l'accuracy et loss de notre proposition et de la reproduction. À gauche la reproduction, à droite notre proposition. On a une meilleure convergence pour le modèle avec dropout, malgré une stabilité moindre.

Ces résultats vont dans le sens de l'affirmation que l'ajout de dropout améliore les capacités de transfert de notre réseau, ce qui correspond à nos attentes. De plus, en réduisant le nombre de blocs on réduit largement notre nombre de paramètres. On a donc à l'issue de ces expériences un modèle à la fois plus performant et moins complexe que celui présenté originalement. Il reste important de tester cette architecture sur d'autres données pour confirmer la qualité de son caractère transférable, mais l'ajout de dropout semble être une piste pertinente pour la construction de modèle qui ont vocation à transférer une représentation apprise.

4.3 Ondelettes

4.3.1 Données et protocole expérimental

Dans ce cadre expérimental nous n'utiliserons que les données de MIT-BIH, de plus au lieu d'utiliser les cinq classes disponibles nous n'utiliserons que les quatre premières. La cinquième est ignorée par manque de données et aussi pour son intérêt limités en application. On souhaite extraire à partir des enregistrements d'ECG les battements uniques, ainsi que les quatre caractéristiques des RR-intervals : previous-RR, post-RR, ratio-RR, local-RR. On commence par débruiter le signal en corrigeant le "baseline wandering" qui correspond à des variations dues à la respiration ou aux mouvements du patient mais qui n'apporte aucune information. Ensuite, on va segmenter les battements uniques et en extraire les caractéristiques des RR-intervals.
Une fois cette segmentation et ce débruitage réalisé on opère la transformation en ondelettes de nos battements. Quatre types d'ondelettes ont été testées dans l'article et c'est celle de Rickler qui a été retenue pour ses performances. On obtient alors une matrice 100x100 pour chaque battement.
Les méthodes et étapes de cette préparation sont présentées plus en détails en [Wang et al., 2021]. Le code a été écrit dans le langage python, la bibliothèque utilisée pour la transformation en ondelettes continues est pywavelets et celle utilisée pour la construction du réseau de neurones est pytorch.

4.3.2 Résultats

Les résultats obtenus sont tout à fait satisfaisant avec une accuracy de 97.5%. Cependant, la qualité de la prédiction varie très largement entre les classes, comme l'illustre la table 3. En effet, le modèle échoue complètement à reconnaître la quatrième classe, avec 1 seul valeur prédite correctement et 387 erreurs.

Classes	Precision(%)	Recall(%)
1	98	99
2	89	74
3	93	95
4	0.2	0.01

TABLE 3 – Performances de la méthodes sur les différentes classes. Les performances sont largement satisfaisantes sur les 3 premières mais pas sur la dernière.

L'échec de classification sur cette dernière catégorie ne suffit pas à invalider la qualité de la méthode. En effet, malgré cela on obtient des résultats satisfaisant en moyenne avec 97.5% d'accuracy, comme évoqué plus haut. De plus, cette classes est sous-représentée dans les données ce qui peut expliquer ces performances. Ces facteurs font que cette performance spécifique ne remet pas en cause l'intérêt de la transformation en ondelettes continue qui offre en moyenne de bons résultats. Nous confortant dans l'idée qu'elle est adaptée aux tâches de classification d'ECG.

La cause de ce manque de performance sur une classe est plus certainement à chercher ailleurs que dans l'usage des ondelettes. On peut par exemple remettre en cause l'usage des caractéristiques des RR-intervals, la méthode présentée en 4.2 offre de meilleurs résultats sur les mêmes données sans en faire usage. De plus, on peut imaginer l'usage d'un classifieur différents pour la prédiction finale. Si on souhaite rester dans le cadre d'un réseau de neurones, ceux-ci ayant prouvés leur efficacité sur la tâche que nous étudions, on pourrait employer une couche de softmax en sortie. On pourrait également explorer de nouvelles méthodes d'apprentissage machine, telles que les *support vector machine* comme en [Faziludeen and Sabiq, 2013].

L'usage de la transformation en ondelettes continue se montre donc pertinent pour la classification de battements de coeurs, malgré certains défauts dans les résultats dont il est difficile de définir les causes exactes. Cependant, les résultats restent en deçà de ceux obtenus plus haut avec l'usage simple d'un CNN avec pourtant une préparation plus lourde des données grâce aux ondelettes.

5 Conclusion et perspectives

Après les expériences que nous avons menées, à la fois dans le cadre d'un CNN transférable et sur la transformée en ondelettes continue couplé d'un réseau de neurone pour la classification, on remarque deux choses.
D'abord les deux méthodes offres des performances satisfaisantes, elles sont toutes les deux pertinentes dans le cadre de la classification de battements de coeur. L'approche finale présentant cependant certaines similarités. En effet, dans les deux cas il est fait usage d'un CNN, même si dans le premier cas il est construit dans l'objectif d'être transférable et qu'il est beaucoup plus profond que pour les ondelettes où il ne présente que quelques couches.
Ensuite, l'utilisation d'un CNN directement sur les données d'entrée, les battements de coeurs sans autres caractéristiques, atteint de meilleures performances sur presque tous les aspects avec le même jeu de données. On peut donc douter de la pertinence de la transformée en ondelettes continues dans le cadre de la classification d'ECG. En effet, elle suppose un traitement supplémentaire des battements avant de pouvoir les classifications. Alourdissant alors la procédure sans pour autant permettre de meilleures performances.

L'usage simple d'un CNN semble être la méthode la plus appropriée pour la classification de battements de coeurs. Comme nous l'avons vu pour avec la détection d'arythmie et d'anormalités. D'autant plus que cela ouvre la porte aux transfert de connaissances qui présente un intérêt tout particulier pour les usages médicaux. En effet, les données manquent parfois et l'idée d'apprendre une représentation généralisable peut être très profitable dans ce cas.
Par ailleurs, nous avons pu voir que l'ajout de dropout au sein d'un modèle peut améliorer sa

capacité de généralisation ce qui est le critère le plus important lors du transfert. C'est une piste qui mérite d'être approfondie et il serait intéressant d'utiliser notre proposition sur de nouvelles données pour confirmer ses qualités.

Une autre perspective pour de futures travaux sur la classification automatique de battements de coeur est la méthode des Signatures. Celle-ci offre des résultats encourageants pour l'analyse de données temporelles mais manque encore de cas d'usages en pratique. Or, sa capacité à extraire des caractéristiques pertinentes à été démontrée et son usage pourrait donc permettre un gain de performance. Pour une présentation détaillée de la méthode des Signature on lira [Chevyrev and Kormilitzin, 2016] et [Fermanian, 2021].

Références

[Chevyrev and Kormilitzin, 2016] Chevyrev, I. and Kormilitzin, A. (2016). A primer on the signature method in machine learning.

[Faziludeen and Sabiq, 2013] Faziludeen, S. and Sabiq, P. (2013). Ecg beat classification using wavelets and svm. In *2013 IEEE Conference on Information & Communication Technologies*, pages 815–818. IEEE.

[Fermanian, 2021] Fermanian, A. (2021). *Learning time-dependent data with the signature transform*. Theses, Sorbonne Université.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[Kachuee et al., 2018] Kachuee, M., Fazeli, S., and Sarrafzadeh, M. (2018). Ecg heartbeat classification : A deep transferable representation. In *2018 IEEE international conference on healthcare informatics (ICHI)*, pages 443–444. IEEE.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam : A method for stochastic optimization. cite arxiv :1412.6980Comment : Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[Lewis and Knowles, 1992] Lewis, A. S. and Knowles, G. (1992). Image compression using the 2-d wavelet transform. *IEEE Transactions on image Processing*, 1(2) :244–250.

[Li, 2018] Li, W. (2018). Wavelets for electrocardiogram : overview and taxonomy. *IEEE Access*, 7 :25627–25649.

[Lin and Qu, 2000] Lin, J. and Qu, L. (2000). Feature extraction based on morlet wavelet and its application for mechanical fault diagnosis. *Journal of sound and vibration*, 234(1) :135–148.

[Lin, 2007] Lin, P.-Y. (2007). An introduction to wavelet transform. *Graduate Institute of Communication Engineering National Taiwan University, Taipei, Taiwan, ROC*.

[Liu et al., 2005] Liu, J., Wu, Y., Han, D., and Li, X. (2005). *Time-Frequency decomposition based on Ricker wavelet*, pages 1937–1940.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

[Sifuzzaman et al., 2009] Sifuzzaman, M., Islam, M. R., and Ali, M. (2009). Application of wavelet transform and its advantages compared to fourier transform.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 :1929–1958.

[Wang et al., 2021] Wang, T., Lu, C., Sun, Y., Yang, M., Liu, C., and Ou, C. (2021). Automatic ecg classification using continuous wavelet transform and convolutional neural network. *Entropy*, 23 :119.

[Weimann and Conrad, 2021] Weimann, K. and Conrad, T. O. (2021). Transfer learning for ecg classification. *Scientific reports*, 11(1) :1–12.