Interprétation des modèles d'apprentissage profond d'aide au diagnostic appliqués aux données d'électrocardiogrammes

Rapport de stage

par

Alexis ALZURIA

Encadrant entreprise : Victoria BOURGEAIS

Encadrant universitaire : Laurent MASCARILLA

Table des matières

1 Lieu du stage – LaBRI

Le LaBRI (Laboratoire Bordelais de Recherche en Informatique) est un laboratoire de recherche en informatique situé à Bordeaux. Il est rattaché à plusieurs tutelles prestigieuses : le CNRS, l'Université de Bordeaux et Bordeaux INP, avec un partenariat actif avec Inria depuis 2002. Le laboratoire joue un important à la fois dans le développement de la recherche fondamentale et appliquée en informatique, dans la formation de haut niveau, notamment à travers l'encadrement de doctorants, et dans la valorisation des travaux scientifiques auprès du monde socio-économique.
Le LaBRI entretient une forte connexion avec les acteurs de son territoire. Le soutien apporté par la Région Nouvelle-Aquitaine, que ce soit en matière de financement d'équipements, de bourses de recherche ou de projets collaboratifs, participe activement à son dynamisme. L'activité du laboratoire se structure autour de plusieurs départements scientifiques spécialisés, couvrant l'ensemble des grands domaines de l'informatique.

1.1 Le département SeD et l'équipe BKB

C'est au sein du département Systèmes et Données (SeD) que mon stage a été réalisé. Ce département s'intéresse principalement aux systèmes informatiques et aux méthodes de traitement des données, en portant une attention particulière à leur efficacité, robustesse et accessibilité. Il regroupe plusieurs équipes qui abordent ces problématiques selon des angles variés, parfois en lien avec d'autres disciplines.
Mon travail s'est déroulé au sein de l'équipe BKB (Bench to Knowledge and Beyond), qui se concentre sur l'étude complète du cycle de vie des données, depuis leur acquisition jusqu'à la restitution de connaissances exploitables, en passant par leur modélisation, leur traitement et leur visualisation. L'équipe a la particularité de travailler en étroite collaboration avec des experts de domaines applicatifs, notamment dans les secteurs de la santé, de la biologie ou des sciences sociales, avec un intérêt marqué pour les approches d'intelligence artificielle explicable (XAI). Les travaux menés visent à produire des outils concrets, applicables à des problématiques réelles.

1.2 Conditions d'accueil et environnement de travail

Mon stage s'est déroulé entre le 3 mars et le 25 juillet 2025. J'ai été accueilli dans un environnement de travail agréable et stimulant. Un bureau individuel m'a été attribué, partagé avec une autre stagiaire dans un premier temps, puis avec un second stagiaire par la suite, ce qui a facilité les échanges et l'entraide au quotidien. L'intégration dans l'équipe a été rapide, grâce à la participation à des réunions de groupe régulières, au cours desquelles j'ai pu suivre les projets en cours et présenter l'avancement de mes travaux. J'ai également eu l'opportunité d'assister à plusieurs séminaires scientifiques proposés par le laboratoire, couvrant des sujets variés en intelligence artificielle et traitement de données.
Au-delà du cadre scientifique, le LaBRI propose une ambiance de travail conviviale et humaine, rythmée par des moments d'échanges informels, des événements collectifs, et des initiatives favorisant l'intégration des stagiaires et doctorants. Cette atmosphère chaleureuse a largement contribué à la qualité de mon expérience.

Cette expérience au sein du LaBRI, et plus particulièrement de l'équipe BKB, m'a permis de m'immerger dans un environnement de recherche orienté vers des problématiques concrètes, au croisement entre informatique fondamentale, sciences des données, et applications biomédicales.

Le stage qui m'a été confié s'inscrit pleinement dans cette dynamique, puisqu'il vise à répondre à une problématique actuelle et critique : l'interprétabilité des modèles d'apprentissage profond utilisés pour l'aide au diagnostic médical à partir de données ECG.

C'est dans ce contexte que s'ouvre ce rapport, dont l'introduction présentera tout d'abord les enjeux médicaux et technologiques liés à la prédiction des arythmies sévères, avant de développer les limites des approches actuelles, les apports récents de l'intelligence artificielle explicable, et les objectifs spécifiques du travail de recherche réalisé durant ces cinq mois de stage.

2 Introduciton

2.1 Contexte général

L'intelligence artificielle (IA), et plus particulièrement l'apprentissage profond (deep learning), connaît un essor fulgurant dans de nombreux domaines, allant de la reconnaissance d'images au traitement du langage naturel. Son potentiel en médecine est désormais reconnu, notamment pour le diagnostic automatisé à partir de données massives, telles que les signaux biologiques. Dans ce cadre, l'électrocardiogramme (ECG) constitue un support

privilégié, en raison de sa large utilisation clinique, de sa richesse informative, et de son accessibilité.
L'une des pathologies cardiovasculaires particulièrement préoccupantes est la Torsade de Pointes (TdP), une forme grave d'arythmie ventriculaire associée à un allongement de l'intervalle QT corrigé (QTc) sur l'ECG. Elle peut être d'origine congénitale (syndrome du QT long, cLQTS) ou médicamenteuse (diLQTS), et peut conduire à une mort subite si elle n'est pas détectée à temps. Le projet ANR DeepECG4U vise précisément à prédire précocement le risque de TdP à partir d'ECG, grâce à des modèles de deep learning entraînés sur des bases de données de patients.

2.2 Limites des approches existantes

Bien que les modèles profonds, tels que les architectures CNN ou CRNN, aient montré une excellente performance prédictive dans ce domaine, leur opacité décisionnelle demeure un frein majeur à leur adoption clinique. En effet, ces modèles sont considérés comme des « boîtes noires », incapables de justifier leurs décisions auprès des médecins utilisateurs. Ce manque d'interprétabilité pose des problèmes de confiance, de reproductibilité et de conformité réglementaire, notamment vis-à-vis du Règlement Général sur la Protection des Données (RGPD) et du AI Act de l'Union Européenne, qui impose une explicabilité des décisions algorithmiques en santé.
L'étudiant précédant ce stage a mené une première évaluation comparative du modèle DeepECG4U et de plusieurs architectures (genDenseNET+ft, CNN-1D-Attn, etc.), sur un nouveau jeu de données de patients exposés à des médicaments allongeant le QT. Un début d'approche explicative a été amorcé, via l'extraction de concepts à partir des activations du réseau. Toutefois, l'explication demeurait post-hoc, c'est-à-dire ajoutée après l'entraînement du modèle, ce qui limite sa précision et sa fiabilité.

2.3 Problématique

Comment concevoir un modèle de diagnostic à partir d'ECG à la fois performant et intrinsèquement explicable, en s'appuyant sur des concepts médicaux interprétables par les cliniciens ? Peut-on aller au-delà des méthodes d'explication post-hoc pour construire des modèles ante-hoc dont les décisions sont structurellement fondées sur des représentations humaines (par exemple, la présence d'ondes T anormales, de variations de QT, etc.) ?

2.4 État de l'art

2.4.1 Le défi de l'explicabilité en intelligence artificielle médicale

L'un des obstacles majeurs à l'intégration des modèles d'apprentissage profond dans les pratiques cliniques réside dans leur manque d'interprétabilité. Contrairement aux scores cliniques traditionnels fondés sur des critères transparents (ex. : CHADS2, QTc), les réseaux de neurones convolutifs (CNN), récurrents (RNN/GRU) ou à attention restent des modèles "boîtes noires". Ce manque de transparence empêche non seulement l'adoption par les médecins, mais entre aussi en conflict avec les exigences légales récentes comme le RGPD et l'AI Act de l'Union Européenne.

2.4.2 Approches post-hoc

Les approches post-hoc cherchent à éclairer les décisions d'un modèle déjà entraîné. On distingue plusieurs catégories :
Attribution locale

LIME, SHAP, Integrated Gradients, ou Grad-CAM tentent de dire quelles parties du signal (dans un ECG par ex.) ont influencé une prédiction.

Attribution conceptuelle

TCAV (Kim et al., 2018) identifie si certains concepts (ex. : "QT long", "onde T aplatie") sont présents dans les couches internes du réseau. Cette méthode repose sur des "activateurs conceptuels" définis par des exemples positifs/négatifs.

L'article Post-hoc vs ante-hoc explanations (Retzlaff et al., 2024) propose un cadre de comparaison systématique entre ces méthodes. Il met en évidence plusieurs limites importantes. Tout d'abord, les explications post-hoc peuvent être instables ou non fidèles au raisonnement réel du modèle. Egalement, les outils comme SHAP ou LIME peuvent produire des justifications plausibles mais incorrectes.
Ces approches sont souvent locales et non généralisables à d'autres cas cliniques.

2.4.3 Approches ante-hoc

À l'opposé, les méthodes ante-hoc visent à rendre les modèles intrinsèquement interprétables en les contraignant à raisonner explicitement en termes de concepts humains. C'est l'objectif des Concept Bottleneck Models (CBMs) introduits par Koh et al. (2020), qui reposent sur un pipeline en deux étapes. Le modèle prédit d'abord un vecteur de concepts médicaux interprétables à partir du signal (par ex. : "présence d'onde U", "écart QTc $>$ 500ms"). Ces concepts servent ensuite d'entrée à un modèle simple (souvent linéaire) pour produire la prédiction finale (ex. : "risque de TdP"). Ainsi, les CBMs permettent d'intervenir sur les concepts à test time (ex. : corriger la présence d'un artefact pour voir l'impact sur la prédiction), de produire des explications causales : "Si le patient n'avait pas de repolarisation tardive, le modèle n'aurait pas prédit un risque élevé".
Des variantes et extensions récentes renforcent encore cette approche :

- CBMs hybrides avec concepts supervisés + non supervisés (Sawada et al., 2022) pour améliorer la couverture du spectre clinique.
- Concept Activation Regions (CARs) (Crabbé et van der Schaar, 2022) : identifient les zones spatiotemporelles où chaque concept s'active, applicable aux signaux ECG.
- Frameworks génériques d'ante-hoc explainability (Sarkar et al., 2020) : apprentissage par contraintes conceptuelles et alignement latent.
- ConceptECGxAI (Bourgeais et al., 2024) : modèle post-hoc exploitant un dictionnaire de concepts cliniques ECG pour expliquer les prédictions d'un CNN sur le risque de TdP.

Enfin, dans le contexte du projet DeepECG4U, des premiers travaux exploratoires ont été menés pour identifier si le modèle activait des représentations cliniquement pertinentes (ex. : segments ST, repolarisation). Ces efforts initiaux manquaient toutefois d'un cadre rigoureux d'apprentissage supervisé par concepts, que ce stage ambitionne d'apporter.

2.5 Objectifs du stage

Ce stage s'inscrit dans une continuité scientifique directe avec le projet DeepECG4U et les stages précédents, tout en introduisant une nouvelle orientation méthodologique majeure : passer d'un paradigme post-hoc à une architecture conceptuelle ante-hoc, dans laquelle l'explicabilité est une propriété native du modèle.
Développer et évaluer un modèle d'apprentissage profond ante-hoc explicable basé sur des concepts cliniques interprétables, appliqué au diagnostic du risque de Torsade de Pointes à partir d'ECG.
Pour répondre à la problématique identifiée, le travail réalisé au cours de ce stage s'est articulé autour de plusieurs axes méthodologiques complémentaires. La première étape a consisté en une revue approfondie de la littérature scientifique, visant à identifier les approches existantes d'explicabilité, avec un accent particulier sur les méthodes conceptuelles dites ante-hoc. Cette phase a permis de comparer les architectures traditionnelles de type « boîte noire » avec les modèles à goulot d'étranglement conceptuel (Concept Bottleneck Models) proposés récemment, et d'évaluer leur applicabilité aux données de type signal 1D, comme les électrocardiogrammes.
Sur cette base, une sélection de méthodes pertinentes a été établie afin de concevoir un modèle explicable et interactif, spécifiquement adapté au contexte du projet DeepECG4U. L'implémentation a reposé sur l'adaptation du schéma CBM proposé par Koh et al. (2020), en définissant une couche intermédiaire correspondant à des concepts cliniques interprétables (tels que l'allongement du QTc, l'aplatissement de l'onde T, la distance J-Tpeak, etc.), annotés ou dérivés à partir des données disponibles. Plusieurs stratégies d'apprentissage ont été envisagées : un apprentissage en deux temps (séquentiel), une approche conjointe optimisant simultanément la prédiction des concepts et du diagnostic, ainsi qu'un mode hybride intégrant des concepts latents non annotés pour renforcer la capacité de représentation du modèle.
L'évaluation de ces modèles a été réalisée sur des jeux de données ECG publics et confidentiels, notamment ceux produits dans le cadre du projet ANR. Les performances ont été mesurées à l'aide des indicateurs classiques en classification binaire (accuracy, AUC-ROC, F1-score), mais également au regard de critères spécifiques à l'explicabilité : fidélité des concepts aux annotations, capacité d'intervention sur les représentations internes, robustesse face à la variabilité inter-patient. Le modèle ante-hoc a été systématiquement comparé à plusieurs approches post-hoc de référence, telles que SHAP, TCAV ou Grad-CAM, appliquées à des architectures convolutionnelles de type DenseNET ou CNN-1D-Attn, afin de mesurer la plus-value réelle de l'intégration des concepts dans l'apprentissage.
Enfin, une attention particulière a été portée à la documentation du travail, tant du point de vue scientifique que technique, dans une perspective de valorisation académique. Si les résultats obtenus s'avéraient concluants, ils pourraient faire l'objet d'une soumission sous forme d'article dans une conférence internationale dédiée à l'intelligence artificielle appliquée à la santé. À travers cette démarche, le stage s'inscrit pleinement dans une dynamique de recherche appliquée, à l'interface entre innovation technologique, exigence réglementaire, et pertinence clinique.

3 Données utilisées et concepts médicaux

3.1 Introduction au signal ECG

L'électrocardiogramme (ECG) est un signal biomédical permettant d'enregistrer l'activité électrique du cœur à l'aide d'électrodes placées sur la surface corporelle. Il constitue un outil fondamental du diagnostic cardiologique. Chaque battement du cœur produit une impulsion électrique qui se propage dans le muscle cardiaque et génère une onde mesurable. Un enregistrement ECG typique présente une séquence d'ondes caractéristiques : **onde P**, **complexe QRS**, **onde T**, et parfois une **onde U**.
Ces composantes correspondent respectivement à différentes phases de l'activité cardiaque :

Définitions des points clés du signal ECG :

- **Onset** : instant où une onde (P, QRS, T) commence.
- **Offset** : instant où l'onde se termine.
- `P_onset` : début de l'onde P.
- `P_offset` : fin de l'onde P.
- `QRS_onset` : début du complexe QRS.
- `T_offset` : fin de l'onde T.
- L'**onde P** reflète la dépolarisation des oreillettes.
- Le **complexe QRS** traduit la dépolarisation des ventricules.
- L'**onde T** représente la repolarisation ventriculaire.
- L'**intervalle QT**, entre `QRS_onset` et `T_offset`, mesure la durée entre le début de la dépolarisation et la fin de la repolarisation ventriculaire ; son allongement est associé à un risque de Torsade de Pointes (TdP), une arythmie potentiellement mortelle.

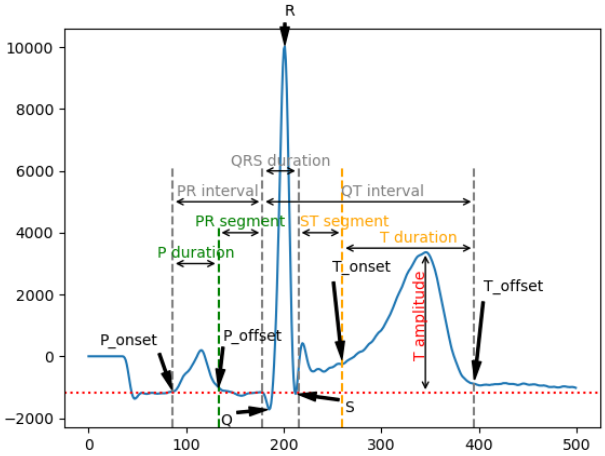


FIGURE 1 – ECG annoté montrant les onsets et offsets des ondes principales.

L'analyse du signal ECG permet ainsi de détecter des anomalies du rythme cardiaque, telles que les arythmies, les tachycardies, ou les syndromes de repolarisation anormale comme le *syndrome du QT long* (Long QT Syndrome, LQTS).

3.2 Le jeu de données GenerePol

Dans le cadre du projet ANR *DeepECG4U*, un jeu de données synthétique nommé **GenerePol** a été développé pour pallier le manque de bases de données ECG annotées en concepts. Ce jeu est généré de manière contrôlée à partir de modèles électrophysiologiques et d'annotations médicales réelles, de sorte à reproduire fidèlement des signaux ECG représentant des patients sous différents traitements médicamenteux susceptibles de modifier l'intervalle QT.
Chaque enregistrement du jeu GénérePol est composé de :

- un **signal ECG** unidimensionnel brut, d'une durée fixe (ex : 10 secondes, échantillonné à 500 Hz, soit 5000 points),
- un **vecteur de concepts annotés** pour chaque signal,
- un **label de classe** indiquant le type de traitement reçu (ou un diagnostic associé à un profil clinique),
- des **métadonnées** (sexe, âge, fréquence cardiaque, etc.) parfois incluses.

Ce jeu a été conçu pour évaluer à la fois la performance prédictive et la capacité des modèles à s'appuyer sur des caractéristiques médicalement pertinentes et compréhensibles. Contrairement à de nombreux jeux ECG open-source, GénérePol intègre explicitement la notion de **concepts médicaux** sous forme vectorielle, ce qui en fait un excellent candidat pour l'apprentissage avec des modèles explicables de type *Concept Bottleneck*.

3.3 Concepts médicaux annotés

Les concepts représentent des informations intermédiaires, compréhensibles par des experts humains, qui décrivent des aspects cliniques du signal. Dans notre cas, ils sont issus soit d'annotations automatiques extraites du signal ECG, soit de connaissances médicales.
Voici quelques exemples typiques de concepts utilisés :

- **Allongement du QT** : booléen (0/1) ou continu (valeur normalisée),
- **Présence d'une onde T anormale** (ex. : inversée, bifide),
- **Intervalle RR régulier ou non**,
- **Présence de pauses** (arrêt momentané de l'activité cardiaque),

— **Sexe féminin** : variable binaire souvent associée à un risque accru de TdP,
— **Type de médicament administré** : catégorisé selon son effet sur la repolarisation,
— **Mesures temporelles issues du signal ECG** :
  — `p-duration`, `pp-interval`, `pr-interval`, `pr-segment`,
  — `qrs-duration`, `qt-duration`, `rr-interval`,
  — `st-segment`, `stt-segment`, `t-duration`, `tp-interval`

3.4 Objectif lié aux concepts dans ce travail

L'objectif du stage est de s'assurer que les prédictions du modèle de diagnostic (i.e. prédiction de la classe finale) sont fondées sur une représentation sémantiquement cohérente : les concepts. Nous cherchons donc à :

— entraîner un modèle g capable de prédire les concepts à partir du signal brut,
— entraîner un modèle f prenant en entrée les concepts (réels ou prédits) pour produire une sortie diagnostique,
— évaluer la pertinence des concepts prédits, leur influence sur la classification, et leur valeur explicative.

Cela permet de faire le lien entre le signal physiologique mesuré, les connaissances médicales humaines, et la prédiction finale, dans une optique de transparence et d'acceptabilité réglementaire (notamment en lien avec l'AI Act européen).

3.5 Jeu de données et construction des concepts

Dans un premier temps, les expérimentations ont été menées sur les données du jeu de données MIT-BIH Arrhythmia, qui constitue une base de référence bien établie dans la littérature pour l'analyse des signaux ECG. Le **MIT-BIH Arrhythmia Database** est une base de données ECG, utilisée pour la détection automatique d'arythmies cardiaques. Elle contient 48 enregistrements ECG de 30 minutes (environ 650 000 samples), chacun annoté manuellement par des experts, avec des types de battements classifiés selon les standards de l'AAMI (Association for the Advancement of Medical Instrumentation). Les signaux ont été enregistrés à une fréquence d'échantillonnage de 360 Hz sur deux dérivations par patient, et incluent une variété de rythmes cardiaques, de morphologies de battements et d'arythmies.
L'objectif est de classifier les segments ECG comme **normaux** ou **anormaux**, à l'aide de signaux bruts et/ou de représentations par concepts physiologiques.
L'utilisation de MIT-BIH a nécessité un travail de construction des concepts approfondi. Une première approche a consisté à **définir manuellement les concepts** à partir d'observations cliniques et de connaissances en électrocardiographie, sans annotation explicite (durées PR, RR, QT, QRS.). Ce jeu de données a permis une première validation de l'architecture de modèle et des fonctions de perte.

Par la suite, le travail a été transféré sur le jeu de données **GenerePol**, spécifiquement recueilli et annoté dans un contexte clinique plus récent, en utilisant les mêmes méthodologies et définitions de concepts que pour le jeu de données MIT-BIH.

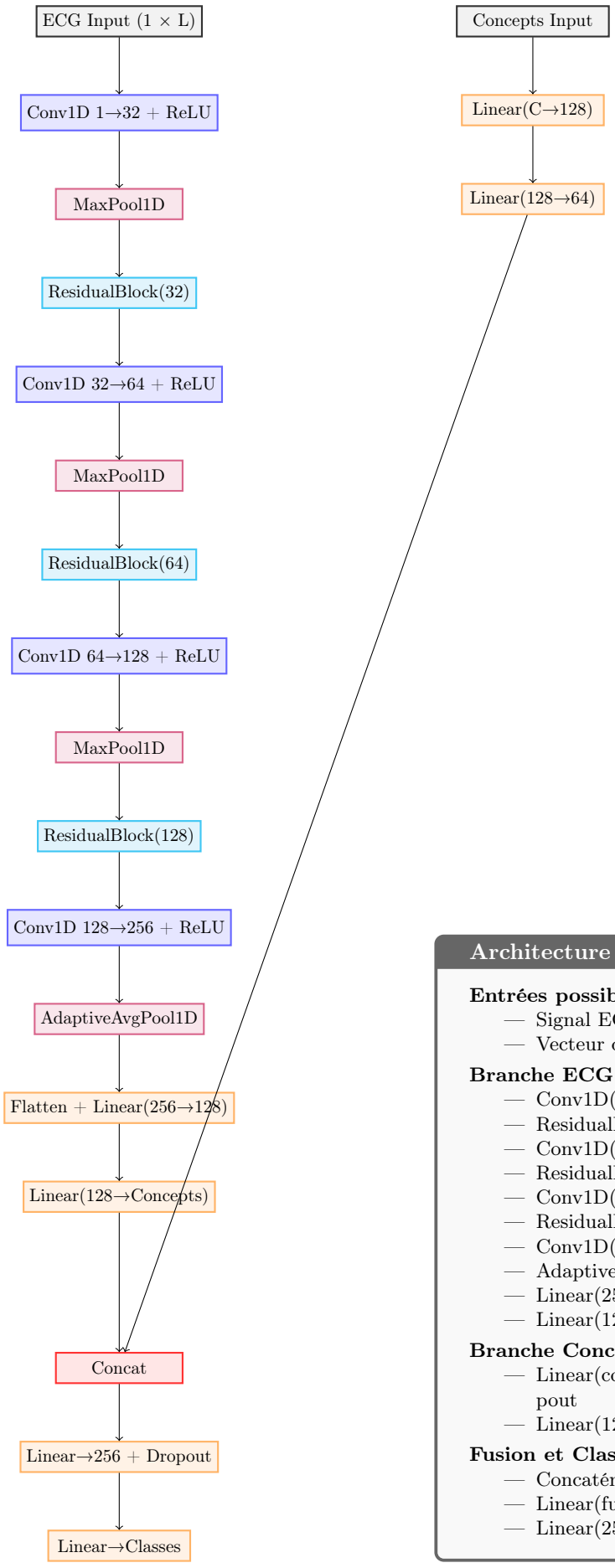
Ensuite, des expérimentations ont été réalisées à partir de **splits pré-définis** fournis avec le jeu de données, sans intervention manuelle dans la définition des concepts. Cela a permis d'évaluer les performances du modèle sur des partitions indépendantes, avant l'introduction d'annotations plus détaillées.
Dans un second temps, le jeu de données a été enrichi avec des **concepts annotés**, à la fois **binaires** (ex : présence ou absence d'un motif pathologique) et **continus** (ex : mesure quantitative d'un segment ou d'une variation de forme), permettant une évaluation plus précise et plus fine des capacités d'apprentissage conceptuel du modèle.

Cette progression dans la qualité et la granularité des concepts a permis d'étudier les performances du modèle dans différents scénarios réalistes, allant de la découverte autonome de concepts à l'utilisation de concepts experts continus ou discrets.

4 Modèle end-to-end avec supervision par concepts

4.1 Chargement et prétraitement des données ECG et concepts

4.2 Architecture et performances



Architecture

Entrées possibles :
- Signal ECG brut $(1, L)$
- Vecteur de concepts (*concept_input_size*)

Branche ECG (Conv1D + Résidus) :
- Conv1D(1, 32) + BN + ReLU + MaxPool1D(2)
- ResidualBlock(32)
- Conv1D(32, 64) + BN + ReLU + MaxPool1D(2)
- ResidualBlock(64)
- Conv1D(64, 128) + BN + ReLU + MaxPool1D(2)
- ResidualBlock(128)
- Conv1D(128, 256) + BN + ReLU
- AdaptiveAvgPool1D(1) → Flatten
- Linear(256→128) + BN + ReLU + Dropout(0.3)
- Linear(128→concept_input_size)

Branche Concepts (MLP) :
- Linear(concept_input_size→128) + BN + ReLU + Dropout
- Linear(128→64) + BN + ReLU

Fusion et Classification :
- Concaténation des branches actives
- Linear(fusion→256) + ReLU + Dropout
- Linear(256→num_classes)

FIGURE 2 – Architecture du modèle *ECGConceptCNN* combinant une branche ECG résiduelle et une branche conceptuelle dense

TABLE 1 – Performances de classification sur les jeux de données *Générépol* et *MIT-BIH* (moyenne ± écart-type)

Méthode	Train Accuracy	Validation Accuracy	Test Accuracy
Jeu de données : MIT-BIH			
ECG uniquement	0.9920 ± 0.0025	0.9850 ± 0.0030	0.9960 ± 0.0018
ECG + Concepts fenêtrés	0.9940 ± 0.0020	0.9870 ± 0.0028	0.9870 ± 0.0025
Jeu de données : Générépol			
ECG uniquement	0.9553 ± 0.0150	0.9245 ± 0.0082	0.8922 ± 0.0069
ECG + Concepts CSV	0.9496 ± 0.0045	0.9165 ± 0.0058	0.8842 ± 0.0033

5 Apprentissage basé sur des concepts : le Concept Bottleneck Model (CBM)

5.1 Chargement et prétraitement des données ECG et concepts - Concepts Continus

La classe `ECGConceptDataset` est une sous-classe de `torch.utils.data.Dataset` destinée à charger et préparer des données d'électrocardiogrammes (ECG) ainsi que des annotations conceptuelles associées pour un apprentissage supervisé. Cette classe gère notamment le filtrage des données selon un split, la normalisation des signaux ECG, le traitement des labels, et l'intégration des concepts continus extraits.

```
class ECGConceptDataset(Dataset):
    def __init__(self, csv_path, concepts_csv_path, split_path=None):
```

Le constructeur reçoit :

- `csv_path` : chemin vers un fichier CSV contenant les données ECG et leurs labels.
- `concepts_csv_path` : chemin vers un CSV contenant les valeurs des concepts associés aux ECG.
- `split_path` (optionnel) : chemin vers un fichier HDF5 spécifiant un sous-ensemble d'ECG à utiliser (ex : ensemble d'entraînement ou de validation).

Chargement des données ECG :

```
ecg_df = pd.read_csv(csv_path)
```

On charge les données ECG dans un `DataFrame` `ecg_df` avec pandas.

Application du split :

```
if split_path is not None:
    import h5py
    with h5py.File(split_path, "r") as f:
        split_ids = {f["uid"][i].decode("utf-8") for i in range(f["uid"].shape[0])}
    ecg_df = ecg_df[ecg_df["ECG-id"].isin(split_ids)]
```

Si un fichier de split est fourni, on récupère les identifiants `ECG-id` autorisés et on filtre le `DataFrame` pour ne conserver que ces ECG.

Extraction des colonnes du signal :

```
self.signal_columns = [col for col in ecg_df.columns if col not in ['ECG-id', 'class']]
self.signals = ecg_df[self.signal_columns].values.astype(np.float32)
```

On identifie toutes les colonnes contenant les données du signal ECG (en excluant les colonnes `"ECG-id"` et `"class"`) et on les convertit en tableau NumPy de type `float32`.

Normalisation des signaux :

```
self.signals = (self.signals - np.mean(self.signals, axis=1, keepdims=True)) / \
               (np.std(self.signals, axis=1, keepdims=True) + 1e-8)
```

On applique une normalisation *z-score* sur chaque signal ECG indépendamment, c'est-à-dire que l'on centre et réduit chaque vecteur signal en soustrayant sa moyenne et en divisant par son écart-type. On ajoute un terme très petit (1×10^{-8}) au dénominateur pour éviter la division par zéro.

Traitement des labels et identifiants :

```
self.ecg_ids = ecg_df['ECG-id'].values
ecg_df['class'] = ecg_df['class'].apply(lambda x: 1 if str(x).lower() == 'sotalol' else 0)
self.labels = ecg_df['class'].values.astype(np.int64)
```

On récupère les identifiants ECG dans `self.ecg_ids`. Ensuite, on transforme la colonne `class` en une variable binaire où la classe "sotalol" est codée par 1 et toutes les autres par 0. Cette colonne est convertie en un tableau entier 64 bits.

Chargement et normalisation des concepts :

```
self.concept_features = [
    'p-duration', 'pp-interval', 'pr-interval', 'pr-segment', 'qrs-duration',
    'qt-duration', 'rr-interval', 'st-segment', 'stt-segment', 't-duration',
    'tp-interval'
]
self.concepts_df = self._load_concepts(concepts_csv_path, self.concept_features)
self.available_concepts_ids = set(self.concepts_df.index)
```

On définit la liste des concepts d'intérêt. Ces concepts sont ensuite chargés depuis un CSV via la méthode interne `_load_concepts`, qui applique également une normalisation min-max. On conserve les identifiants ECG pour lesquels les concepts sont disponibles.

Méthode privée `_load_concepts` :

```
def _load_concepts(self, path, concept_features):
    df = pd.read_csv(path, header=[0, 1])
    df.columns = pd.MultiIndex.from_tuples(
        [('ECG-id', '')] + [(c1.strip(), c2.strip()) for c1, c2 in df.columns[1:]]
    )
    df.set_index(('ECG-id', ''), inplace=True)

    selected_columns = []
    for concept in concept_features:
        selected_columns.append((concept, 'mean'))

    concept_df = df[selected_columns].copy()
    concept_df.columns = [f"{c1}-{c2}" for c1, c2 in concept_df.columns]

    concept_df = (concept_df - concept_df.min()) / (concept_df.max() - concept_df.min() + 1e
        -8)

    return concept_df
```

Cette méthode charge un fichier CSV contenant des mesures statistiques des concepts (moyennes et éventuellement écarts-types). On sélectionne uniquement les colonnes des moyennes des concepts désirés, on renomme les colonnes pour simplifier, puis on applique une normalisation min-max sur chaque colonne.

Taille du dataset :

```
def __len__(self):
    return len(self.signals)
```

Cette méthode retourne le nombre total d'exemples disponibles, basé sur le nombre de signaux ECG chargés.

Accès à un élément du dataset :

```
def __getitem__(self, idx):
    signal = torch.tensor(self.signals[idx], dtype=torch.float32).unsqueeze(0)  # (1,
        signal_length)
    label = torch.tensor(self.labels[idx], dtype=torch.long)
    ecg_id = self.ecg_ids[idx]

```

```
    valid_flag = ecg_id in self.available_concepts_ids
    if valid_flag:
        concept_values = self.concepts_df.loc[ecg_id].fillna(0).values.astype(np.float32)
    else:
        concept_values = np.zeros(len(self.concepts_df.columns), dtype=np.float32)

    concept_tensor = torch.tensor(concept_values, dtype=torch.float32)

    return signal, concept_tensor, label, valid_flag
```

Pour un indice donné `idx`, cette méthode retourne :

- Le signal ECG sous forme d'un tenseur PyTorch de dimension $(1, \text{longueur\_signal})$, ajouté d'une dimension canal.
- Le vecteur des concepts correspondants, normalisé et converti en tenseur PyTorch.
- Le label de classification binaire.
- Un indicateur booléen `valid_flag` qui est vrai si les concepts sont disponibles pour cet ECG, sinon faux. Si les concepts ne sont pas disponibles, un vecteur nul est retourné.

5.2 Chargement et prétraitement des données ECG et concepts - Concepts Binaires

La classe `ECGConceptDatasetBinaire` est une variante de `ECGConceptDataset` (voir explication précédente) destinée à manipuler des concepts ECG **binaires**, extraits sous deux formes : `long` et `short`. Elle applique automatiquement une binarisation si les données originales sont continues.
Elle repose sur les mêmes principes de chargement, normalisation et filtrage des signaux ECG que `ECGConceptDataset`, et ne diffère principalement que par le traitement des concepts.

Méthode `_load_concepts`

```
def _load_concepts(self, path, concept_features):
    df = pd.read_csv(path, header=[0, 1])

    df.columns = pd.MultiIndex.from_tuples([(c1.strip(), c2.strip()) for c1, c2 in df.columns
        ])

    id_col = [col for col in df.columns if col[0] == 'ECG-id']
    if not id_col:
        raise KeyError("Impossible de trouver la colonne 'ECG-id' dans les niveaux de colonnes
            .")
    df.set_index(id_col[0], inplace=True)
```

Les colonnes sont chargées sous forme de multi-index pour accéder à des mesures distinctes (par ex. `p-duration, short` et `p-duration, long`). La colonne `ECG-id` est utilisée comme index.

```
    selected_columns = [
        (c1, metric)
        for c1 in concept_features
        for metric in ['long', 'short']
        if (c1, metric) in df.columns
    ]

    if not selected_columns:
        raise ValueError(f"Aucune colonne valide trouv e pour les concepts : {
            concept_features}")

    concept_df = df[selected_columns].copy()
    concept_df.columns = [f"{c1}-{c2}" for c1, c2 in concept_df.columns]
```

Les concepts sont extraits sous leurs deux versions (long/short) pour chaque caractéristique. On concatène le nom de la colonne comme `p-duration-long`, `qt-duration-short`, etc.

Binarisation des concepts

```
    is_binary = concept_df.dropna().isin([0, 1]).all().all()

    if is_binary:
        print("Colonnes d j  binaires     pas de binarisation appliqu e.")
        concept_df = concept_df.fillna(0).astype(np.float32)
    else:
        print("Donn es non binaires     binarisation par m diane appliqu e.")
```

```
        threshold = concept_df.median()
        concept_df = (concept_df >= threshold).astype(np.float32)

    return concept_df
```

— Si toutes les colonnes sont déjà binaires (valeurs 0 ou 1), elles sont simplement nettoyées des `NaN` et converties en `float32`.
— Sinon, une **binarisation par seuil médian** est appliquée : chaque valeur est comparée à la médiane de sa colonne et devient 1 si supérieure ou égale, 0 sinon. Cela permet de transformer des variables continues en indicateurs binaires sans supervision.

Méthode `__getitem__`

```
def __getitem__(self, idx):
    signal = torch.tensor(self.signals[idx], dtype=torch.float32).unsqueeze(0)  # (1, signal_length)
    label = torch.tensor(self.labels[idx], dtype=torch.long)
    ecg_id = self.ecg_ids[idx]

    valid_flag = ecg_id in self.available_concepts_ids
    if valid_flag:
        concept_values = self.concepts_df.loc[ecg_id].fillna(0).values.astype(np.float32)
    else:
        concept_values = np.zeros(len(self.concept_features) * 2, dtype=np.float32)  # x2: long + short

    concept_tensor = torch.tensor(concept_values, dtype=torch.float32)

    return signal, concept_tensor, label
```

La principale différence ici est la dimension des concepts binaires, qui est le double de celle du dataset de base, car chaque concept est décliné en deux versions : `long` et `short`. Si un ECG ne dispose pas d'annotations, un vecteur de zéros de longueur $2 \times$ nombre_de_concepts est retourné.

`ECGConceptDatasetBinaire` est conçu pour intégrer des **concepts binaires dérivés des ECG**. Il prend en charge :

— Le format `long/short` pour chaque concept ;
— La binarisation automatique par seuil médian si nécessaire ;
— Le traitement robuste des ECG sans annotation disponible.

Il est compatible avec les pipelines PyTorch standard et permet d'entraîner des modèles de classification utilisant des concepts interprétables, tout en conservant un signal normalisé.

5.3 Architecture et hyperparamètres

5.4 Variation de lambda et performances

6 Interprétabilité