Rapport de projet de session
GLO-7030 - Apprentissage par réseaux de neurones profonds

Rayane BADJI - 537 177 500, Fabrice GAGNON - 111 146 970,
Mathis REZZOUK - 537 172 964 et Loïs TRASSOUDAINE - 537 171 697

***Abstract*— L'analyse des séries temporelles, en particulier dans le contexte des données électroencéphalographiques (EEG), offre une perspective précieuse pour comprendre les variations des états cognitifs de différents sujets. En étudiant ces données aux travers des enregistrements temporels, les chercheurs peuvent observer les fluctuations des signaux cérébraux et identifier des motifs significatifs liés à divers états cognitifs. Malgré leur immense succès dans de nombreux domaines, les systèmes d'apprentissage automatique et d'apprentissage profond n'ont pas encore réussi à s'établir fermement dans des applications critiques en matière de santé. Une des principales raisons réside dans le fait que les modèles sont confrontés à des échantillons de longue portée générés par des systèmes d'enregistrements différents et soumis à du bruit ambiant propre à chaque patient. Ce projet a pour but d'explorer les performances d'un modèle type Transformers sur un problème de prédiction d'épilepsie effectué à partir de données EEG.**

I. INTRODUCTION

Les données EEG fournissent des informations cruciales sur l'activité électrique du cerveau, mais leur interprétation reste souvent complexe en raison de la nature hautement dimensionnelle et dynamique de ceux-ci [10]. Comprendre comment les modèles d'apprentissage automatique traitent et interprètent ces données peut aider les chercheurs et les cliniciens à mieux saisir les mécanismes sous-jacents des processus cognitifs et des troubles neurologiques. L'apprentissage automatique profond est devenu un outil incontournable dans l'analyse des données EEG, jouant un rôle vital dans l'expansion rapide des innovations du domaine de la santé. La disponibilité de jeux de données EEG abondants a ouvert des opportunités passionnantes pour les chercheurs d'appliquer des techniques d'apprentissage automatique, stimulant des approches innovantes dans l'analyse des données EEG [12]. Dans les dernières années, les Transformers sont devenus de plus en plus populaires dans les tâches de classification EEG et l'analyse de données de séries temporelles. Initialement conçus pour le traitement du langage naturel (NLP) et d'autres données séquentielles étendues, les Transformers gagnent en reconnaissance en raison de leur polyvalence et de leur puissance, comme illustré par ChatGPT. Ils utilisent l'auto-attention pour capturer les relations dans une séquence, ce qui les rend efficaces pour représenter les données. Cela permet un entraînement et une inférence plus rapides sur des ensembles de données plus importants avec des modèles complexes [2]. C'est dans ce cadre que nous avons décidé de faire un projet de session sur l'implémentation de cette architecture afin de prédire des crises d'épilepsies à partir d'enregistrements EEG.

II. ÉTAT DE L'ART

La nature multidimensionnelle des séries temporelles multivariées et la teneur élevée d'informations des enregistrements EEG favorise l'utilisation de modèles d'apprentissage profond. Dans l'état de l'art, le réseau de neurones à convolution (CNN) est une architecture d'apprentissage profond couramment utilisée à la fois en Vision par Ordinateur (CV) et en Interface Cerveau-Ordinateur (BCI). La force du CNN réside dans sa capacité à extraire des caractéristiques essentielles et des relations spatiales à partir des spectrogrammes EEG, qui représentent la fréquence de ces signaux dans le temps. Par exemple, un modèle de CNN a été utilisé pour classifier les étapes d'une crise d'épilepsie [11]. L'architecture CNN se compose de couches à convolution, de pooling et de couches complètement connectées, permettant une extraction efficace des caractéristiques et une compréhension des motifs. Les CNN sont bien adaptés à l'analyse EEG, car ils traitent les données de spectrogramme EEG comme des données semblables à des images. En parallèle, l'avènement des réseaux de neurones récurrents (RNN) conçus pour le traitement de données séquentielles a permis un grand bond dans le traitement de données temporelles telles que l'EEG. La variante Long Short-Term Memory (LSTM) est largement utilisée dans la classification EEG pour surmonter le défi du gradient qui disparaît et pour apprendre les dépendances à long terme présentes dans les données. Les LSTM intègrent des "portes" permettant de retenir les informations pertinentes sur de plus longues périodes. Les modèles RNN, en particulier les LSTM, sont souvent combinés avec d'autres architectures telles que les CNN pour constituer un pipeline puissant d'extraction de caractéristiques et de modélisation de séquences. Cette approche a été couronnée de succès dans la classification EEG [10]. Cependant, les modèles RNN sont souvent plus coûteux en termes de calcul par rapport à d'autres architectures comme les CNN. Dans cette lancée vient l'apparition des Transformers. Ce dernier se distingue de ses prédécesseurs (Recurrent Neural Network et Long Short Term Memory) par sa capacité à traiter les données temporelles sur de longues périodes de temps grâce à son mécanisme d'attention multi-têtes [2]. De plus, les mécanismes d'attention utilisés par les Transformers permettent de comprendre quels éléments de la séquence ont contribué à une prédiction donnée. Les architectures basées sur l'attention semblent être efficaces et réussissent à améliorer la capacité de généralisation des modèles de classification de signaux biologiques [9].

III. Présentation du jeu de données

A. La technique EEG

L'electroencephalographie est mise en place à l'aide d'un nombre fini d'électrodes positionnées de part et d'autre de la tête du patient. Une électrode capte le potentiel électrique d'une zone en particulier.

Les zones couvertes par les électrodes ont chacune une particularité, une activité précise à capter du cerveau.

- Lobe frontal : Activité associée à la planification, la prise de décision, le contrôle moteur. Les signaux EEG peuvent montrer des variations en fonction de la cognition, de la motricité volontaire et de l'attention.
- Lobe pariétal : Impliqué dans le traitement sensoriel, la perception spatiale et le traitement des informations sensorielles. Les signaux EEG peuvent refléter le traitement des stimuli sensoriels et la perception spatiale.
- Lobe temporal : Impliqué dans l'audition, la mémoire et le traitement des émotions. Les signaux EEG peuvent montrer des réponses spécifiques aux stimuli auditifs et aux processus de mémoire.
- Lobe occipital : Principalement associé à la vision et au traitement visuel. Les signaux EEG peuvent révéler des réponses spécifiques aux stimuli visuels et aux processus de traitement visuel.

B. Le jeu de données du MIT

Le jeu de donnée que nous avons utilisé a été recueilli dans le cadre du projet collaboratif entre Children's Hospital Boston (CHB) et le Massachusetts Institute of Technology (MIT) [1]. Ce jeu de donnée a été construit à partir des données de 22 patients (chb01, chb02, chb03,...), ayant entre 9 et 42 enregistrements stockés en .edf (European Data Format) selon le sujet. Dans la plupart des cas, les fichiers .edf contiennent exactement une heure de signaux EEG numérisés, bien que ceux appartenant au cas chb10 durent deux heures et ceux appartenant aux cas chb04, chb06, chb07, chb09 et chb23 durent quatre heures. De manière occasionnelle, les fichiers dans lesquels des crises sont enregistrées peuvent être plus courts. Les patients ont entre 3 et 22 ans.

Tous les signaux ont été échantillonnés à 256 Hz (256 points par seconde) avec une résolution de 16 bits. La plupart des fichiers contiennent 23 signaux EEG (24 ou 26 dans quelques cas). Le système international de positionnement et de nomenclature des électrodes EEG 10-20 a été utilisé pour ces enregistrements. Dans quelques enregistrements, d'autres signaux sont également enregistrés, tels qu'un signal ECG dans les 36 derniers fichiers appartenant au cas chb04 et un signal de stimulation du nerf vague (VNS) dans les 18 derniers fichiers appartenant au cas chb09. Dans certains cas, jusqu'à cinq signaux "factices" (nommés "-") ont été intercalés parmi les signaux EEG pour obtenir un format d'affichage facile à lire ; ces signaux factices peuvent être ignorés.

Les "Channels" réfèrent aux binômes de capteurs, aux électrodes qui vont être utilisées pour calculer la différence de potentiel entre les différentes zones d'intérêt du cerveau.

Enfin, le fichier RECORDS contient une liste des 664 fichiers .edf inclus dans cette collection, tandis que le fichier RECORDS-WITH-SEIZURES répertorie 129 de ces fichiers contenant une ou plusieurs crises. Le fichier SUBJECT-INFO contient le genre et l'âge de chaque sujet. (Le cas chb24 a été ajouté à cette collection en décembre 2010 et n'est pas actuellement inclus dans SUBJECT-INFO.)

Dans l'ensemble, ces enregistrements comprennent 198 crises au total ; le début et la fin de chaque crise sont annotés dans les fichiers d'annotations nommés chbnn-summary.txt contenus dans chaque dossier patient. De plus, ces fichiers contiennent des informations sur le montage utilisé pour chaque enregistrement, ainsi que le temps d'activité de chaque fichier patient.

IV. Approche proposée

A. Prétraitement des données

Comme expliqué dans la partie IV-C, l'architecture implémentée est un Transformer. Or, les Transformers sont des architectures profondes qui nécessitent beaucoup de données pour performer. De plus, le but du projet étant de détecter une crise d'épilepsie, il est donc plus pertinent de raccourcir la durée des segments pour se rapprocher de la durée réelle moyenne d'une crise d'épilepsie. À titre d'exemple, pour la majorité des patients, un segment dure une heure alors qu'une crise n'est qu'en moyenne d'une minute. L'idée est donc de couper les segments en sous-segments de durée égale. Cela aura l'avantage d'augmenter le nombre d'échantillons du Transformer et de les normaliser à la durée réelle d'une crise. De plus, cela permettra d'avoir plus d'informations, car certains segments possèdent au maximum six crises. Cependant, échantillonner les données aura pour conséquence d'augmenter le débalancement des classes. Ce dernier étant déjà à 4:1 sans segmentation des échantillons, un fichier épilepsie pour quatre fichiers neutre, en appliquant cette technique, on augmentera significativement le débalancement. À cet égard, il est à noter que l'architecture développée et présentée dans la partie IV-C travaille avec une attention mutuelle sur tous les tokens. Le pas de temps étant de 256 par minutes, les tokens sont aussi de courte durée. Pour des raisons computationnelles, nous devons de plus limiter le nombre de segments. Considérant ces contraintes, nous avons découpé les enregistrements en séquences de 10 secondes. Cela va avoir pour conséquence d'augmenter le débalancement.

Puisque les données sont segmentées, celles-ci doivent être étiquetées de nouveau. Pour ce faire, nous étiquetons comme crise tout segment contenant une portion de crise, même s'il s'agit d'une infime partie. Cela aura comme avantage de permettre d'identifier la crise en ne possédant qu'une partie de son schéma et augmentera donc la robustesse du classificateur. Pour faire cela, nous lisons chaque fichier chbnn-summary.tx avec la fonction read_summary, ce qui donne les périodes de crises.

Les données du projet sont très lourdes, celles-ci pèsent environ 42,6 Go. L'importation et le traitement sont donc

très lourds en charge mémoire. Pour diminuer cela, nous avons converti les données en float32 plutôt qu'en float64 à la création des jeux d'entrainements et de tests.

B. Extraction de caractéristiques

Comme précisé ci-dessus, pour être exploitées par notre modèle, nos données doivent être sous forme de tokens, et plus précisément sous la forme de vecteurs qui capturent les caractéristiques propres de chaque élément d'une donnée séquentielle. Un ensemble de tokens associé à une donnée doit en effet capturer l'ensemble des caractéristiques d'un segment. Rappelons qu'un patient, à quelques exceptions près, est toujours décris par 23 signaux tels que chaque signal est lié à une zone particulière du cerveau. Nous avons donc 23 données séquentielles pour chaque segment. Toutefois, nous voulons traiter un segment dans son entièreté et non chaque signal séparément (rappel : un segment représente une portion des signaux pendant un court laps de temps). Finalement, une ligne de nos données en fin de pré-traitement devra être (analogie : tout comme en NLP) une suite de tokens représentant chaque sous-partie (analogie : chaque mot) de notre segment de signaux (analogie : de notre texte). Vous pouvez retrouver ci-dessous une représentation de nos données Figure 1.

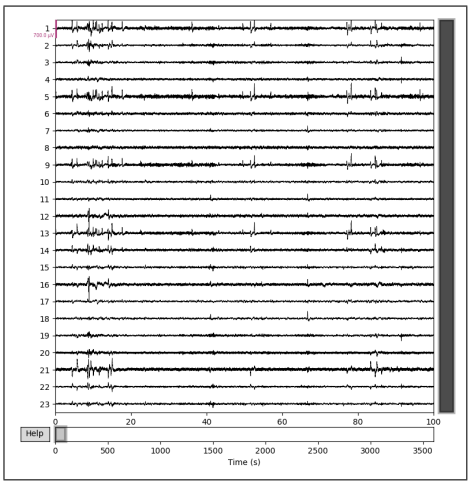

Fig. 1. Un enregistrement sans crise - Patient1_02

Comme nous pouvons le constater, un enregistrement a des valeurs pour chaque canal, chaque seconde. Un problème de dimension se pose. Une donnée dans notre cas est caractérisée par une matrice de valeurs : si nous prenons une sous-partie d'une durée de 1 seconde de notre segment de 10 secondes, nous avons 256 valeurs pour chaque channel (pas de temps) donc une matrice 23x256. Or un token doit être sous la forme d'un vecteur de valeurs numériques afin d'être manipulable par notre modèle. Afin de répondre à ce problème de dimension, nous ferons appel à l'extraction de caractéristiques par convolution. La méthode d'extraction de caractéristiques suivante a été directement inspirée par celle décrite dans l'article [9]. Une figure extraite de l'article est disponible Figure 2.

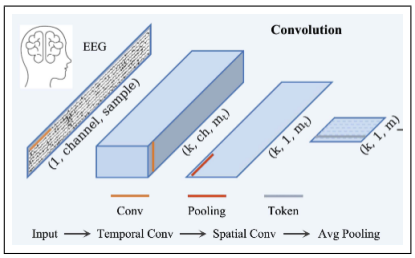

Fig. 2. Réduction de dimensionnalité et Extraction des caractéristiques

Le processus de convolution permet de réduire la dimension des données en un format plus approprié en extrayant les caractéristiques les plus importantes parmi les 23 canaux et ainsi transformer un segment temporel du signal en un vecteur de format (valeur, valeur, valeur, valeur).

Étapes de l'extraction par convolution :

- Transformation temporelle de l'entrée par convolutions : des convolutions de taille (1,25) avec un pas de (1,1) permettent de résumer les informations temporelles.
- Transformation spatiale : des convolutions de tailles (nombre_channel, 1) avec pas de (1,1). Cela permet de comprendre les relations entre les différents canaux.
- Introduction de non-linéarité : Ajout d'une composante de non-linéarité en introduisant une ELU.
- Average Pooling : permet de diminuer la complexité et les risques de sur-entraînement.
- Application du dropout : Fait en sorte que le token généré par une même donnée ne sera pas entièrement identique. Cela permet de générer du bruit et d'améliorer les performances de généralisation du système.

Ces étapes permettent d'obtenir des tokens de taille fixe qui résume les dimensions temporelles et spatiales des données en entrée.

C. Apprentissage profond avec architecture Transformer

L'approche que nous avons utilisée pour implémenter l'architecture Transformer se base sur celle présentée dans l'article EEGConformer: 'Convolutional Transformer for EEG Decoding and Visualization'[9]. L'analyse des signaux EEG à l'aide de cette architecture peut être bénéfique, car celle-ci permet de capturer les dépendances à long terme des signaux, ce que les méthodes conventionnelles utilisées jusqu'à présent dans ce domaine étaient incapables d'effectuer. La méthode EEG Conformer combine des couches de CNN pour apprendre les caractéristiques temporelles et spatiales locales, puis l'auto-attention pour encapsuler les caractéristiques temporelles globales. L'architecture globale est segmentée en trois modules principaux; un module de convolution, un module d'auto-attention et un classificateur entièrement connecté.

1) Module de convolution: Ce module suit les étapes décrites dans la section précédente.

2) Module d'auto attention: Le module d'auto attention est conçu pour apprendre les dépendances temporelles globales des EEG. Les jetons extraits du module précédent sont transformés en trois copies de même forme, selon le standard QKV, contenant l'information fréquentielle. Un produit scalaire est ensuite effectué entre Q et K pour évaluer la corrélation et un facteur de mis à l'échelle est appliqué afin d'éviter la perte du gradient. Une Softmax est ensuite appliqué pour obtenir la matrice de corrélation.
Une stratégie multi-tête est utilisée pour améliorer la diversité de représentation de chaque segment de données.

3) Classificateur: Finalement, deux couches pleinement connectées sont implémentées. La première effectue un global average pooling, permettant de calculer la moyenne sur la dimension des tokens et de réduire chaque ensemble de caractéristiques qui sont par la suite normalisés. La seconde applique séquentiellement des couches linéaires puis des fonctions d'activation qui permettent l'ajout de la non-linéarité au modèle.

D. Apprentissage profond avec une architecture LSTM

Afin de vérifier la performance de notre modèle de Transformer, nous allons comparer ce dernier à un modèle récurrent. Cela nous permet de confirmer si notre problème (problème binaire sur un grand jeu de données avec classes non balancées) nécessitera l'utilisation d'un modèle complexe impliquant un Transformer ou plutôt l'utilisation d'un simple modèle récurrent. Nous avons choisi de tester la performance de classification d'un réseau LSTM, car ce dernier, à l'inverse d'un RNN basique, est capable de capturer les relations à long terme entre les données. Cette capacité d'appariement des caractéristiques entre les différents tokens est aussi existante au sein des mécanismes d'attention, mais à plus grande échelle.

Comme nous pouvons le constater sur les signaux des figures 3 et 4, afin de capturer la crise, notre modèle doit prendre en considération la corrélation entre les différents canaux sans ordre spécifique. Sur les signaux provenant de deux crises différentes, il est assez flagrant de voir que les informations d'un canal à un autre n'est pas le même pour deux crises différentes. En effet, étant donné que le cerveau est un ensemble interconnecté, une zone (valeurs d'un canal) peut en affecter une autre, affecter les valeurs d'un autre.
Comparaison de deux crises :

- → Figure 3 : Patient 2, enregistrement 16, entre 130s et 212s
- → Figure 4 : Patient 3, enregistrement 2, entre 731s et 796s

Cette notion de relation globale des channels et par extension, des tokens est un aspect inhérent à nos données et c'est pour cela que nous testons directement des modèles capturant des dépendances à long terme.

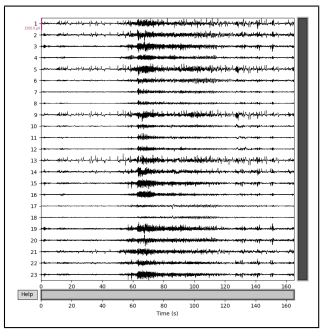

Fig. 3. Crise sur l'enregistrement 16 du patient 2

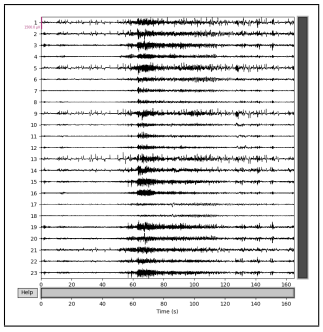

Fig. 4. Crise sur l'enregistrement 2 du patient 3

E. Entraînement de classes non balancées

Comme précisé dans la partie IV-A, les données ont un fort débalancement. De plus, dans le cas qui nous occupe, le groupe minoritaire est celui détectant les crises d'épilepsie et donc la classe que nous voulons classifier avec le plus de précision. Avec la segmentation choisie, le débalancement sera de l'ordre de 1000:1. Si aucune stratégie n'est adoptée et qu'une perte standard est utilisée, l'algorithme sera incapable d'apprendre la classe minoritaire. En effet, d'après les tests préliminaires, dans ce cas, le transformer apprend rapidement que la meilleure méthode pour diminuer une entropie croisée classique est de ne prédire que des 0. Cela donne effectivement une très bonne exactitude proche de 100%.

1) Métriques utilisées: Le premier constat à tirer de l'exemple précédent est que l'exactitude n'est pas une bonne métrique pour les données débalancées. C'est aussi ce qu'expose l'article [3]. L'article souligne que le f1-score est une très bonne métrique pour juger de la performance d'une classe débalancée. Il s'agit d'une alliance entre le rappel qui se concentre sur le nombre de données positives qui ont été correctement prédites et la précision qui donne le pourcentage de données prédites comme positives qui sont réellement positives. Il est à noter que, contrairement au rappel, la précision est sensible au débalancement dans la mesure où les faux positifs sont pris en compte. Néanmoins, dans notre cas, il est également important de ne pas avoir une quantité trop importante de segments détectés comme une crise pour ne pas dévaluer l'importance des segments positifs. Le f1-score nous permettra donc de juger

correctement de la performance de notre modèle.

2) Fonctions de perte testées: L'entropie croisée est une perte largement utilisée dans le cadre des réseaux de neurones. Il s'agit d'une perte qui permet de résoudre des problèmes d'optimisations déterministe et stochastique et permet d'estimer des événements rares [13]. Dans notre cas, il est fondamental de pouvoir prédire ce genre d'évènements. Cependant, la version classique de l'entropie croisée a été créée pour classifier en multi-classe et prend en entrée une fonction *logSoftmax* qui permet d'obtenir une distribution logarithmique de la probabilité d'appartenir à chaque classe. Notre projet est un cas de détection binaire, il serait donc plus adapté de travailler avec l'entropie croisée binaire qui est adapté à la classification binaire. Dont la formule est la suivante:

$$l_n = -w_n \left[y_n \cdot \log(\sigma(x_n)) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right] \tag{1}$$

pour n allant de 0 à N avec N: taille du batch, w_n: poids utilisé pour faire varier l'importance de la perte en fonction de l'époque, nous le gardons à 1, y_n: l'étiquette réelle, x_n: l'étiquette prédite.

On utilise une version de l'entropie croisée qui inclut directement une Sigmoid, qui est la fonction adaptée pour les sorties binaires.
Un autre version de l'entropie croisée, la perte focale, a été développée pour le besoin particulier de classes débalancées dans la détection d'objets denses [14]. L'équation donnée par l'article est la suivante 2.

$$FL(pt) = -\alpha(1 - pt)^{\gamma} \log(pt) \tag{2}$$

Où $pt = p$ si $y = 1$ et $1 - p$ sinon, avec p: étiquette prédite, y: étiquette réelle, α: permet de balancer les données.
Avec cette formule, on peut réécrire la formule de l'entropie croisée binaire en, $CE(pt) = -\log(pt)$ et on voit bien que la focal loss ajoute le facteur $(1 - pt)^{\gamma}$, qui permet de réduire l'importance des exemples simples, donc pour nous, de la classe négative désignant l'absence de crise.
Dans toutes les pertes dérivées de l'entropy croisée précédemment présentées, nous avons dû réaliser une adaptation de la formule pour prendre en compte les données débalancées. Pour cela, on met un poids plus important à la classe positive, en calculant le facteur de débalancement et en le donnant en paramètre aux fonctions.

3) Le sur-échantillonnage par Data Augmentation: Comme expliqué précédemment, nous avons une très faible quantité de données positives, par rapport aux données négatives. De plus, le transformer gagne à avoir le plus de données possible. Nous allons donc sur-échantillonner la classe positive, ce qui signifie que nous allons créer des variantes des données de la classe positive pour avoir de nouveaux exemples. L'article [15] traite d'un sujet assez similaire : la reconnaissance d'émotion à l'aide d'EEG. Les auteurs proposent une data augmentation avec l'algorithme Borderline-SMOTE. Cette approche est une variante de l'algorithme SMOTE qui crée des données synthétiques de la classe minoritaire en étudiant les voisins les plus proches au sein de cette classe. Borderline ajoute l'étude de la classe majoritaire. Borderline-SMOTE fait en sorte de revenir à un facteur de débalancement 1:1.
Cependant, nos données étaient trop éloignées, l'algorithme n'est pas parvenu à réaliser des données synthétiques à partir du peu d'échantillons dont nous disposions. Cette technique a donc été abandonnée.

L'autre méthode de sur-échantillonnage développé est celle de segmentation et reconstruction, comme décrite à la section B. de l'article [9]. Celle-ci prend en paramètre les données de nos classes positives et découpe chaque séquence en un nombre de séquences déterminées au préalable. Ensuite, les séquences sont mélangées aléatoirement puis reformées pour créer un nouvel enregistrement positif. Pour limiter la complexité de calcul lors de ces opérations, nous avons opté de découper chaque sous segments en sections de 20 pas de temps et de multiplier par 10 le nombre d'échantillons positifs.

4) Sous-échantillonnage: On réalise un sous-échantillonnage par un facteur 10 de la classe majoritaire pour tenter de rééquilibrer un peu les données. On passera d'un débalancement d'environ 1000:1 à 100:1.

V. Méthodologie et expérimentation

Les expérimentations et les résultats présentés dans la partie suivante ont été réalisés et obtenus sur un ordinateur sous Microsoft Windows 10, avec un processeur AMD Ryzen 7 1700 Eight-Core Processor, avec 16 processeurs logiques. La machine possède 16 Go de RAM et tourne avec un GPU 3060 12Gb. Pour utiliser le GPU, nous utilisons la version 11.8 de Cuda.

A. Transformer

1) Hyperparametres convolution: Pour la première convolution:

- in_channelles:1
- out_channel:40
- kernel_size:(1,25)
- stride:(1,1)

Pour la deuxième convolution:

- in_channelles:40
- out_channel:40
- kernel_size:(nb_channels,1)
- stride:(1,1)

Suivi de

- BatchNorm2d:40
- ELU
- AvgPool2d((1, 75), (1, 15))
- Dropout(0.5)

2) Hyperparametres Transformer:

- Nombre d'encodeurs: 6
- Taille embbeding: 40
- Nombre de têtes: 4
- Tailles des têtes: 512

3) Hyperparametres optimiseur et scheduler:

- AdamW: lr=0.001
- OneCycleLR: max_lr=0.003

4) Cas 1: Données prélevées de 9 patients, soit 01, 02, 03, 05, 06, 07, 08, 23, 24. Le nombre 0, donc la classe négative, a été diminué d'un facteur 10, car suivant des tests préliminaires, nous n'arrivions pas à des résultats concluants à cause du débalancement. Nous avons aussi appliqué l'augmentation et la reconstruction sur le jeu d'entrainement pour permettre au model de possiblement apprendre à reconnaitre une crise. Les résultats sont les suivants pour l'entrainement (Fig. 6), le test (Fig. 7) et la validation (Fig. 7) avec la FocalLoss et la Figure 5 donne l'évolution de la perte et de la précision.

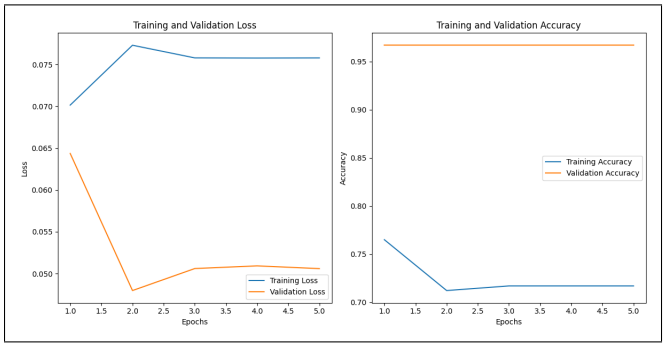

Fig. 5. Perte et précision sur 12 952 enregistrements

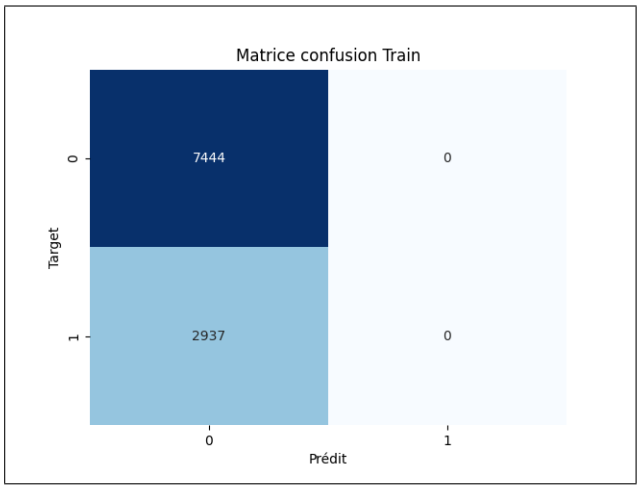

Fig. 6. Précision sur entrainement

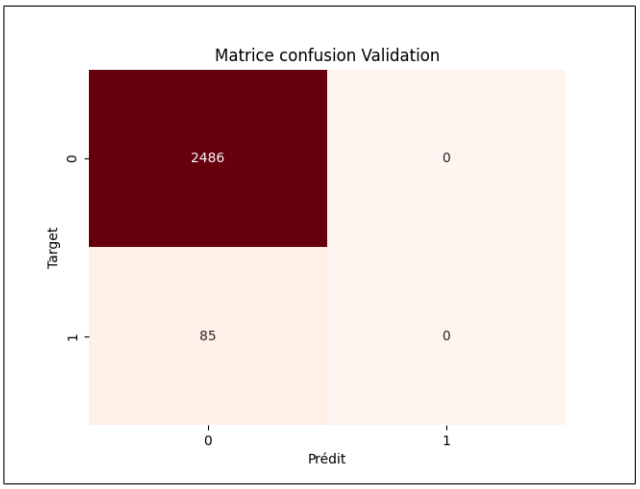

Fig. 7. Précision sur validation

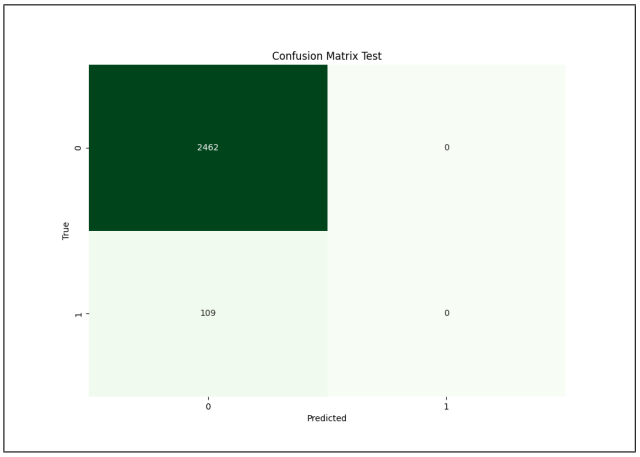

Fig. 8. Précision sur validation

Comme on peut le constater avec les matrices de confusion et l'historique d'entrainement, le réseau ne fait qu'apprendre à assigner des 0 aux enregistrements. Ceci était le cas aussi bien pour les données seulement augmenté que non modifiés. D'autres taux d'apprentissages ont été utilisés pour les tests, mais aucun ne menait à des résultats concluants.

B. LSTM

Notre modèle LSTM utilisera les mêmes tokens en entrée que notre modèle Transformer (mêmes hyperparamètres pour la convolution).

Hyperparamètres par défaut de notre modèle LSTM régularisé :

- 2 couches cachés;
- 128 neurones par couche cachée;
- dropout entre chaque couche de 0.5;
- weight-decay de 0.001 sur la dernière couche linéaire;

1) Cas 1 - Données sur 3 patients sans utilisation de la perte focale: Sur 3 patients, nous avons 40861 segments.

Notons que la proportion de crise parmi ces segments (proportion de segments avec label 1) est de 0.3%.
Afin de répondre aux limites computationnelles des `array` pour nos deux premiers cas, nous allons échantillonner 8000 segments parmi nos 3 patients. Cet échantillonnage est fait de telle sorte à ce que les proportions de 1 et de 0 ne soient pas modifiées, car nous souhaitons observer la capacité de classification de notre LSTM sans modification de la donnée. Pour 8000 segments, nous passons à une proportion de label un de 0.5%, ce qui est suffisamment proche de celle sur la donnée de trois patients.
Comme précisé dans le titre, pour cette expérimentation, nous n'allons pas utiliser la perte focale dans le cadre de l'entraînement. La fonction de perte utilisée sera la perte $BCEloss()$ qui est une entropie croisée binaire.

L'évolution de la perte et de l'exactitude de notre modèle sur cette portion de données peut être observée sur les figures 9 et 10.

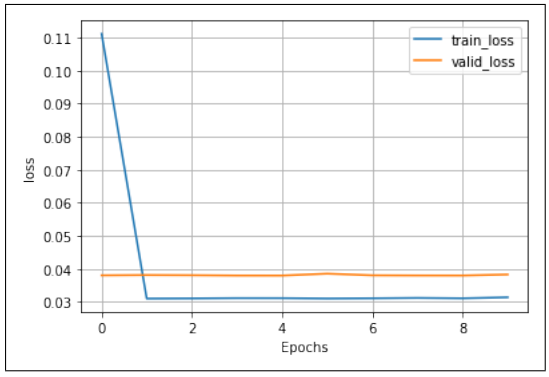

Fig. 9. Loss Cas 01 - LSTM sur 8000 enregistrements sans FL

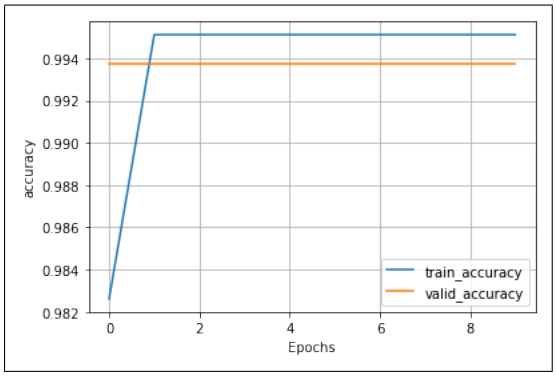

Fig. 10. Accuracy Cas 01 - LSTM sur 8000 enregistrements sans FL

Après une seule itération d'apprentissage (une époque), nous observons une nette diminution de la perte et une augmentation significative de l'exactitude sur nos données d'entraînement. Cette tendance peut être attribuée à la prédominance d'un grand nombre de zéros dans nos données. De plus, il est à noter que notre modèle a tendance à surapprendre, étant donné que la perte sur nos données de validation est restée constante tout au long des 20 itérations d'entraînement. Pour obtenir une évaluation plus approfondie des performances de notre modèle, tournons-nous vers la matrice de confusion. Cette dernière fournit des informations détaillées sur la répartition des prédictions correctes pour chaque classe de labels. Grâce à la matrice de confusion, nous pouvons observer le nombre de prédictions de la classe 1 sur la figure 11.

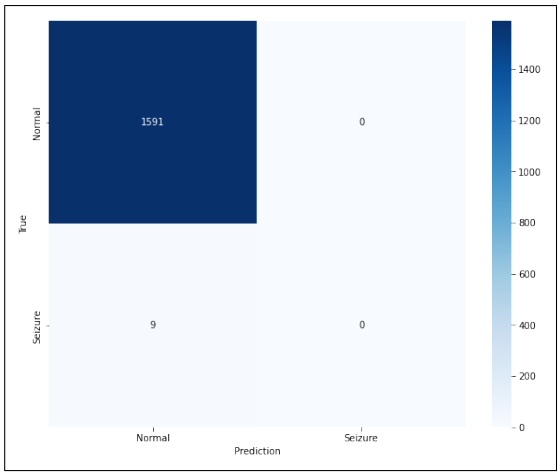

Fig. 11. Matrice de confusion Cas 01 - LSTM sur 8000 enregistrements sans FL

Il est assez facile d'observer à travers notre matrice que notre modèle n'a été en mesure de reconnaître aucune crise dans notre jeu de données de test. Le constat est le même que pour le Transformer.

2) Cas 2 - Données sur 3 patients avec utilisation de la perte Focale et régularisation: Testons désormais notre modèle avec la fonction de perte $FocalLoss()$ et un weight_decay. L'évolution de la perte de notre modèle sur cette portion de données peut être observée sur la figure 12. On peut observer que notre modèle ne se surajuste plus aux données d'entraînement étant donné que le loss n'est plus constant pour nos données de validation. Toutefois, on observe également un manque de stabilité en terme d'apprentissage.

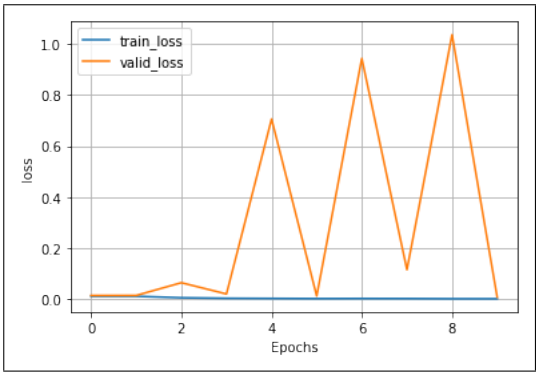

Fig. 12. Loss Cas 02 - LSTM sur 8000 enregistrements avec FL

Vous pouvez également observer sur la matrice de confusion (Fig. 13) que 6 segments comprenant une crise ont été

correctement labelisés. Passer de 0 à 6 nous montre que notre modèle a finalement par le biais de la régularisation (weight decay) et d'une perte adaptée au débalancement des classes, réussi à détecter les caractéristiques propres aux crises.

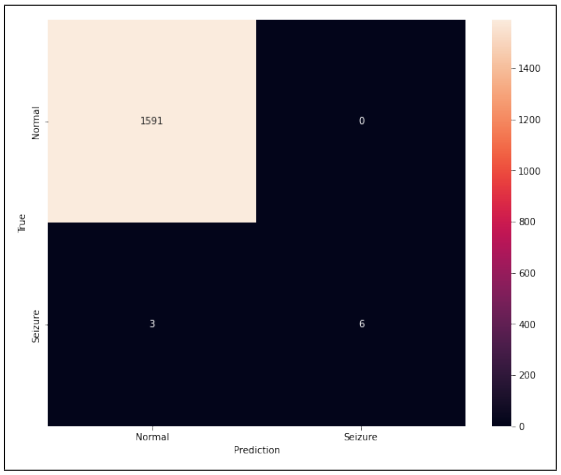

Fig. 13. Matrice de confusion Cas 02 - LSTM W/FL sur 8000 enregistrements

VI. Discussion des résultats

	Accuracy	F1-score	Temps d'execution
$LSTM_{8000}$ W/FL	99%	40%	2min
$Transformer$ W/FL	99%	0%	10 min

TABLE I
TABLEAU COMPARATIF DES ÉVALUATIONS DE CHAQUE MODÈLE

D'après le tableau I, seule l'architecture LSTM performe correctement.

En effet, bien que l'exactitude soit très importante pour le transformer, son F1-score est nulle. Cela est le reflet d'un jeu de données très balancé et d'une architecture ne prédisant que la classe majoritaire. Malgré tous nos efforts en terme de débalancement et avec toutes les données compatibles présentes dans notre jeu de données, nous ne sommes pas parvenus à changer ces résultats. Il est possible que le problème soit dû au fait que les transformers doivent être entraînés sur une quantité très importante de données. Or, bien que nous avons 42,6 Go de données, cela ne donne en réalité pas une très grande quantité de données aux environs de 40 000 segments. De plus, avec le downsampling nécessaire pour des raisons computationnelles et de débalancement, on retombe à un peu plus de 10 000 enregistrements, avec 2937 exemples positifs. Nous pensons que cela pourrait ne pas être une quantité suffisante pour permettre l'apprentissage d'un transformer.

Quant au LSTM, on a réussi à obtenir de meilleurs résultats que pour le transformer. Le LSTM est aussi plus rapide que le transformer, ce qui le rend meilleur en tout point, dans notre cas. La première version du LSTM utilisant la BCELoss (Cas 1) performe de manière similaire au Transformer en ne ressortant que la classe positive. Cependant, en introduisant la perte focale, qui est une perte créée dans l'optique des données débalancées, on obtient des résultats intéressants. On remarque d'abord que le modèle arrive à ne plus prédire que la classe majoritaire, il apprend réellement. On voit cet apprentissage grâce à l'évolution de la perte disponible Figure 12. Bien que la perte varie beaucoup, on voit qu'elle ne parvient pas à converger. On remarque un profil en dent de scies, qui montre une difficulté de stabilité dans le modèle. Cela peut être dû au fait que le modèle n'est pas assez robuste pour nos données. En effet, nos données sont extrêmement bruitées. Il est de plus possible que le peu de données positives empêchent le LSTM de correctement converger. Il est tout de même à conclure que le LSTM performe très bien et ne présente pas de surapprentissage, comme le montre les performances du tableau I et les bons résultats du f1-score en test.

VII. Conclusion

Dans le cadre de ce projet, nous avons implémenté deux architectures pour réaliser la détection de crises d'épilepsie à l'aide de signal EEG. Les deux architectures sont un Transformer et un LSTM. Le transformer n'a pas su donner des résultats satisfaisants, cependant le LSTM a permis d'effectuer une détection satisfaisante. Face au constat du maigre nombre de segments contenant des crises, nous avons dû mettre en place des stratégies pour le débalancement. Pour aller plus loin, on pourrait essayer de tester une sélection de channels sur les individus pour déterminer si des endroits du crâne sont déterminants dans la détection de crise. Retenter les architectures sur un jeu de données plus important pourrait permettre au Transformer de mieux performer et pourrait être intéressant à approfondir.

References

[1] Guttag, John. "CHB-MIT Scalp EEG Database" (version 1.0.0). PhysioNet (2010), https://doi.org/10.13026/C2K01R.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. Attention Is All You Need. (2023)

[3] Johnson, J.M., Khoshgoftaar, T.M. Survey on deep learning with class imbalance. J Big Data 6, 27 (2019). https://doi.org/10.1186/s40537-019-0192-5

[4] Wong S, Simmons A, Rivera-Villicana J, Barnett S, Sivathamboo S, Perucca P, Ge Z, Kwan P, Kuhlmann L, Vasa R, Mouzakis K, O'Brien TJ. EEG datasets for seizure detection and prediction- A review. Epilepsia Open. (2023)

[5] Jayant N Acharya, Abeer Hani, Janna Cheek, Partha Thirumala, Tammy N Tsuchida. Guideline 2: Guidelines for Standard Electrode Position Nomenclature. American Clinical Neurophysiology Society. (2016)

[6] Stevenson, N., Tapani, K., Lauronen, L. et al. A dataset of neonatal EEG recordings with seizure annotations. Sci Data 6, 190039 (2019). https://doi.org/10.1038/sdata.2019.39

[7] Zhu, Y. & Wang, M. Automated Seizure Detection using Transformer Models on Multi-ChannelEEGs. *2023 IEEE EMBS International Conference On Biomedical And Health Informatics (BHI) (IEEE BHI 2023)*. pp. 21 (2023,10)

[8] Wan, Z., Li, M., Liu, S., Huang, J., Tan, H. & Duan, W. EEGformer: A transformer–based brain activity classification method using EEG signal. *Frontiers In Neuroscience*. **17** (2023), https://doi.org/10.3389/fnins.2023.1148855

[9] Song, Y., Zheng, Q., Liu, B. & Gao, X. EEG Conformer: Convolutional Transformer for EEG Decoding and Visualization. *IEEE Transactions On Neural Systems And Rehabilitation Engineering*. **31** pp. 710-719 (2023)

[10] Bashivan, P., Rish, I., Yeasin, M. & Codella, N. Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks. (2016)

[11] Mao, W., Fathurrahman, H., Lee, Y. & Chang, T. EEG dataset classification using CNN method. *Journal Of Physics: Conference Series*. **1456**, 012017 (2020,1), https://dx.doi.org/10.1088/1742-6596/1456/1/012017

[12] Craik A, He Y, Contreras-Vidal JL. Deep learning for electroencephalogram (EEG) classification tasks: a review. J Neural Eng. 2019 Jun;16(3):031001. doi: 10.1088/1741-2552/ab0ab5. Epub 2019 Feb 26. PMID: 30808014.

[13] de Boer, PT., Kroese, D.P., Mannor, S. et al. A Tutorial on the Cross-Entropy Method. Ann Oper Res 134, 19–67 (2005). https://doi.org/10.1007/s10479-005-5724-z

[14] Tsung-Yi Lin. Focal Loss for Dense Object Detection. *ICCV*. (2017)

[15] Chen, Y., Chang, R. & Guo, J. Effects of Data Augmentation Method Borderline-SMOTE on Emotion Recognition of EEG Signals Based on Convolutional Neural Network. *IEEE Access*. **9** pp. 47491-47502 (2021)