ARN – Laboratoire #3

Quentin Surdez
ISCL, HEIG-VD
quentin.surdez@heig-vd.ch

REDACTED
ISCS, HEIG-VD
REDACTED

Introduction

Des données EEG sur plusieurs souris ont été reçues. Ces données possèdent plusieurs états auxquels sont associés des amplitudes ayant une fréquence donnée. Un *Multi-Layer Perceptron* (MLP) est mis en place pour classifier les données sans connaître l'état de la souris.

Ce rapport présente l'évolution du travail effectué. Un MLP binaire est d'abord implémenté pour classifier les données selon l'état d'endormissement – soit endormi ou réveillé – des souris. Dans un second temps, un MLP permettant de classifier les données selon 3 états – éveillé, sommeil REM et sommeil N-REM – est mis en place.

Les objectifs de ce travail sont les suivants:

- Explorer et sélectionner les features qui donneront le plus d'informations concernant l'état dans lequel la souris se trouve.
- Adresser le problème de déséquilibre du jeu de données de base qui a été relevé durant le premier travail.
- Développer et optimiser des architectures neuronales pour obtenir une classification correcte.
- Évaluer différentes stratégies d'optimisation et leur impact sur la performance du modèle.

Preprocessing

La première étape est celle du preprocessing lors de laquelle le jeu de donnes et chargé et les features d'intérêt sont sélectionnées. Il a été observé dans le laboratoire précédent que les features ayant le plus d'intérêt pour déterminer l'état dans lequel une souris se trouve sont les amplitudes les plus basses. Nous avons donc choisi de garder les amplitudes allant de 1 Hz à 25 Hz.

Après ce choix de features, les données sont normalisées. Cela permet de remettre sur une échelle commune les différentes plages de valeurs. Pour ce faire, le `StandardScaler` de la librairie `scikit-learn` est utilité.

Ensuite, les données sont validées grâce à la validation croisée. Le choix est porté sur une validation croisée à 3-fold. Les données de bases sont mélangées afin d'éviter tout biais, ce car la majeure partie des données se trouvent dans les amplitudes de basses valeurs.

Classification binaire

Les états sont encodés en deux classes: 1 pour éveillé et 0 pour les sommeils REM et N-REM. Ce choix est fait afin de pouvoir utiliser la fonction Sigmoïde – qui rend une valeur dans l'intervalle $[0, 1]$ – en sortie.

Résumé du modèle

Le modèle devisé est le suivant:

- Le vecteur d'input est composé de 25 élément pour les 25 features; il y a donc 25 neurones.

- Ces 25 neurones sont connectés de manière dense à une première couche cachée composée de 32 neurones et la fonction d'activation choisie est ReLU. Ce choix a été effectué de par sa simplicité et l'efficacité de la fonction. 32 neurones donnent un résultat.
- L'output est composé d'un seul neurone qui est soit actif, soit inactif. La fonction d'activation choisie est la fonction Sigmoïde puisque la valeur de sortie se situe entre 0 et 1.
- L'optimizer choisi est la descente de gradient stochastique avec un learning rate de 0.01 et un momentum de 0.9. D'autres valeurs ont été essayées sans changement majeur dans les résultats.
- La function de loss choisie est la binary cross entropy au vu du problème de catégorisation binaire.
- Le nombre d'epochs est de 100. On observe que moins d'epochs serait intéressant après plusieurs essais pour éviter l'overfitting.

Résultat

La matrice présentée à Fig. 1 a donné le meilleur score ainsi que le meilleur score F1 score associé. Le F1 score global est de **0.8928**.

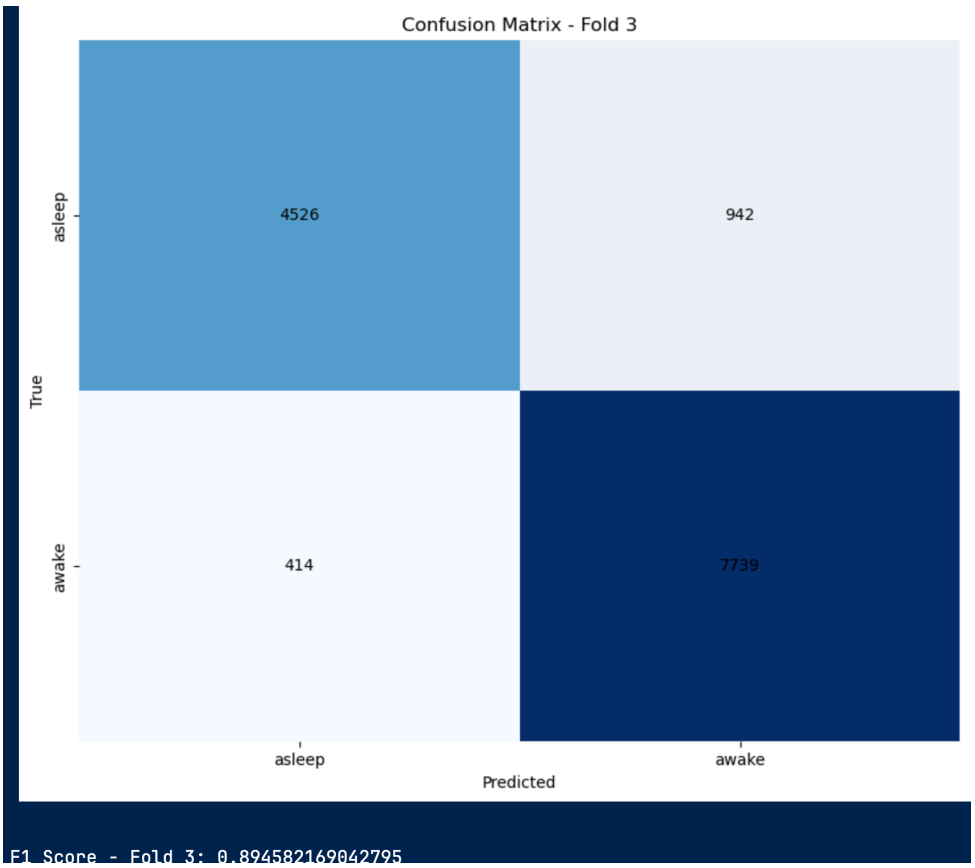

Fig. 1. – Matrice de confusion menant au F1 Score de 0.8928

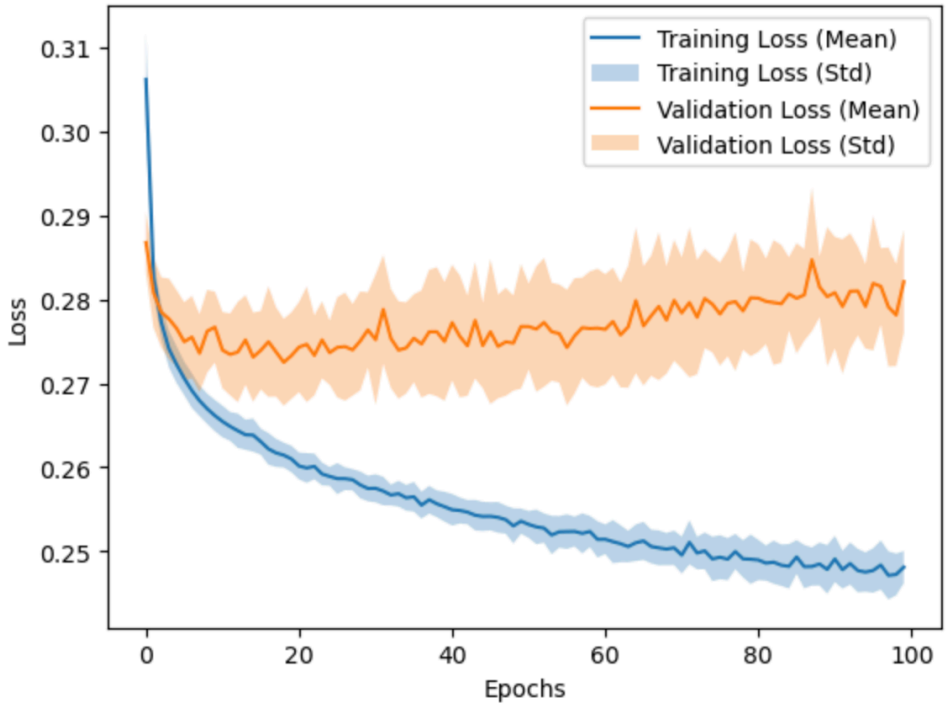

Fig. 2. – Loss par époque

La Fig. 2 montre que les courbes sont éloignées l'une de l'autre. La courbe de validation présente une pente positive à partir de 20 epochs alors que la courbe de training continue sur sa trajectoire négative même proche des 100 epochs. Cela peut indiquer que soit le dataset de training n'est pas représentatif de celui de validation, soit qu'il y a de l'overfitting sur les données de training.

Des amélioration potentielle à ce modèle seraient soit d'amener de la régulation avec de l'early stopping par exemple, soit de réduire le nombre d'epochs.

L'autre caractéristique principale de ces courbes sont les piques qui les composent. En utilisant la loss function MSE, ces piques disparaissent. Cela est indicatif que la binary cross validation nécessite plus d'architecture autour d'elle que de simples couches cachées. La solution pour réduire ces piques peut être de changer d'optimizer ou changer de loss function.

La matrice de confusion indique que le modèle a plus de peine à catégoriser la classe asleep que la classe awake. Cela ramène au problème de l'équilibrage du jeu de données qui avait été relevé lors de l'exploration des données.

Une amélioration potentielle à apporter à l'étape du preprocessing serait de faire de l'oversampling pour les classes qui n'ont que peu de représentation.

Le F1-score global est de 0.8928. Ce score est correct car il se situe très proche de 1.

Classification 3 classes

Les états sont maintenant encodés en trois classes qui sont présentes dans le dataset : éveillé, sommeil REM et sommeil N-REM. Un `LabelEncoder` est utilisé pour encoder les états. Ce dernier est mis à disposition par le module `preprocessing` de la librairie `scikit-learn`. Ensuite, la fonction `to_categorical` est appliquée afin d'avoir des vecteurs de 3 dimensions représentant chacune de nos classes.

Ce preprocessing supplémentaire permet une meilleure prise en main par la fonction de loss categorical crossentropy.

Résumé du modèle

Le modèle est similaire au modèle binaire. Ses caractéristiques sont les suivantes :

- Le vecteur d'input contient 25 éléments pour les 25 features et il y a donc 25 neurones.
- Ces 25 neurones sont connectés de manière dense à une première couche cachée composée de 16 neurones, la fonction d'activation choisie est ReLU. Nous l'avons choisi pour sa simplicité et son efficacité. 16 neurones donnent un bon résultat comparé à 32 dans ce cas-ci. On observe une baisse de l'efficacité corrélée à une hausse du nombre de neurones.
- Une deuxième couche cachée de 8 neurones. La fonction d'activation est toujours relu pour les mêmes raisons. 8 a été choisi car c'est la moitié de 16 et que les résultats avec ce nombre étaient concluants.
- La dernière couche possède 3 neurones, comme le nombre de classes. Ici, la fonction d'activation est softmax. Cette dernière est particulièrement adaptée lors des choix entre plusieurs classes comme elle ramène les résultats entre 0 et 1.
- L'optimizer descente de gradient stochastique est toujours utilisé.
- La loss function categorical cross entropy est utilisée puisqu'elle est devisée pour des problèmes de catégorisation.
- Le nombre d'epochs est de 100. On observe que moins d'epochs serait intéressant après plusieurs essais pour éviter l'overfitting.

Résultat

La matrice présentée dans la Fig. 3 donne le meilleur score F1 . Le F1 score global est de **0.7742**.

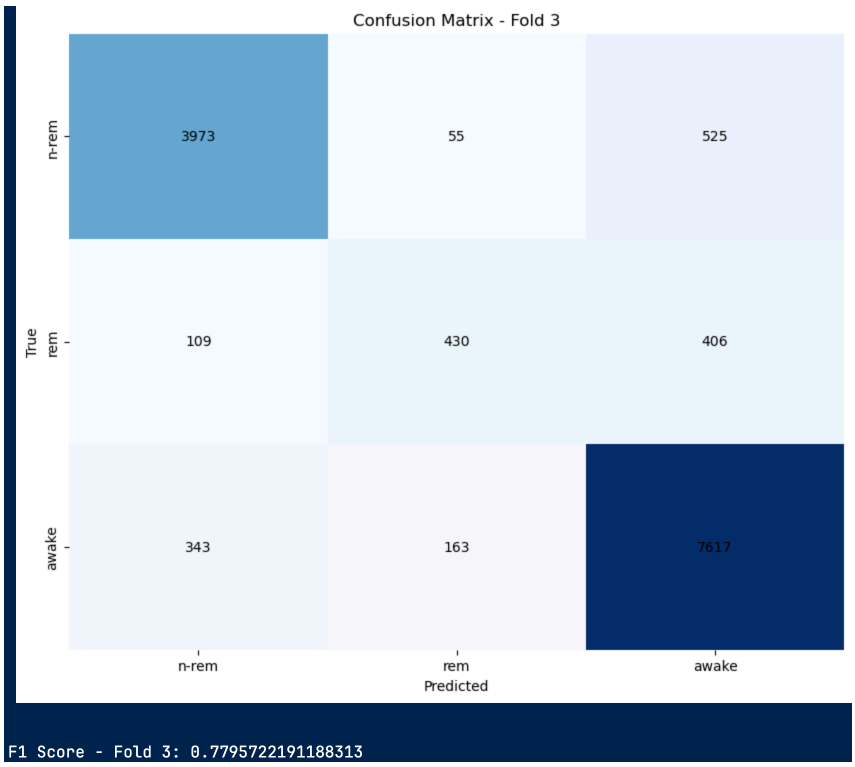

Fig. 3. – Matrice de confusion tri-classe

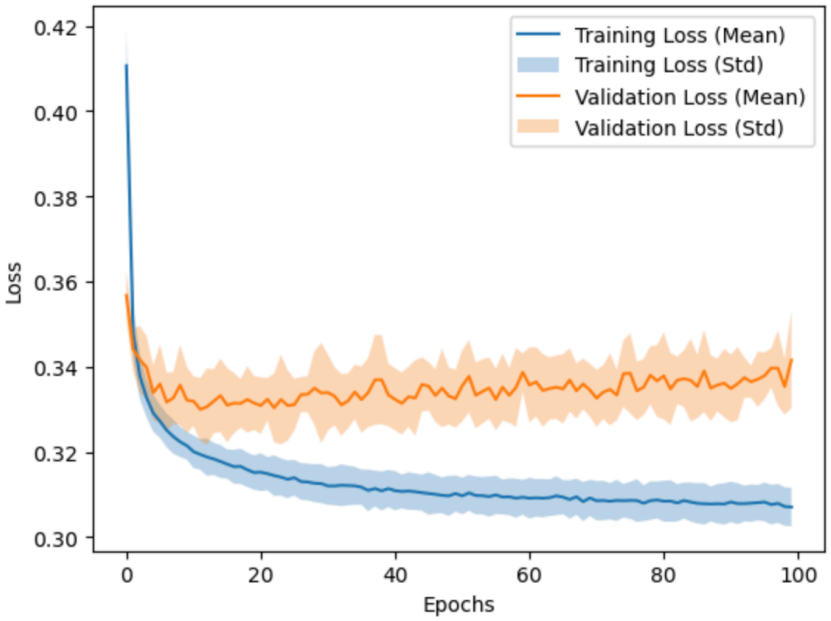

Fig. 4. – Loss par époque

Le graphique présenté à la figure Fig. 4 présente l'évolution de la loss au fil des époques.

Les observations sont particulièrement semblables au graphe de la catégorisation binaire. En effet, le même optimizer ainsi qu'une variante de la fonction de loss cross entropy ont été utilisé dans les deux cas.

Les conclusions restent les mêmes: insérer du dropout pour amoindrir l'overfitting et oversampler les classes sous-représentées.

Le F1 score global est de 0.7742, ce qui est une valeur satisfaisante. En effet, on ajoute des catégories qui sont relativement déséquilibrées par rapport aux autres (2000 N-REM vs. 12000 awake), et pourtant le F1 score ne drop pas énormément.

Amélioration

Plusieurs améliorations trouvant leur source dans la littérature scientifique relative aux MLP ont été essayées.

La première amélioration apportée est se situe au niveau du préprocessing des données. En effet, il faut oversampler les données qui sont en moins grandes quantités que les autres. Pour ce faire, la librairie `imbalanced-learn` est utilisée. Elle permet l'implémentation de plusieurs méthodes d'oversampling. Bien que la méthode choisie soit basique, elle n'est pas naïve. Cette méthode se nomme Synthetic Minority Over-sampling Technique (SMOTE) et a pour but de chercher les voisins des valeurs des classes minoritaires pour « tirer une ligne entre la valeur et ses voisins » et créer une nouvelle valeur sur cette ligne.

La deuxième amélioration mise en place est le `dropout` entre les différentes layers. Cela permet de forcer le training set à ne pas overfit sur ses données et amène à une meilleure généralisation.

La troisième amélioration est d'insérer des couches de `BatchNormalization` entre les différentes hidden layers. Cela permet d'aller plus vite de 1, mais aussi de garder les gradients dans une échelle acceptable. Bien que plus utile dans les architectures de réseaux complexes et denses, nous souhaitions l'utiliser afin d'acquérir de l'expérience.

La quatrième amélioration apportée est l'early stopping. Cette technique est une technique de régularisation. Elle permet de prévenir l'overfitting. Cependant elle fait apparaître un nouveau hyper paramètre qui est la patience ou le nombre d'epochs d'attente acceptable pour une amélioration avant de stopper.

La cinquième amélioration est l'utilisation de l'optimizer Adam. Cet optimizer va permettre d'adapter les learning rates en fonction de la magnitude des gradients récents ainsi que d'appliquer un certain momentum. Il est souvent utiliser dans les réseaux de neurones.

La métrique choisie pour compiler le modèle est `CategoricalAccuracy`. Elle permet de mettre un avant un choix de catégorie correct de la part du modèle. Cela permet d'en apprendre un peu plus sur les options de la fonction `compile` et son fonctionnement de manière générale.

Le dernier changement effectué est l'ajout d'une couche cachée avec 32 neurones. L'amélioration apportée par l'ajout de cette couche n'est cependant que marginale.

Résultat

La matrice présentée dans la Fig. 5 a donné le meilleur score ainsi que le meilleur score F1 associé. Le graph de l'entrainement est présenté dans la Fig. 6. Le F1 score global est de **0.8442**.

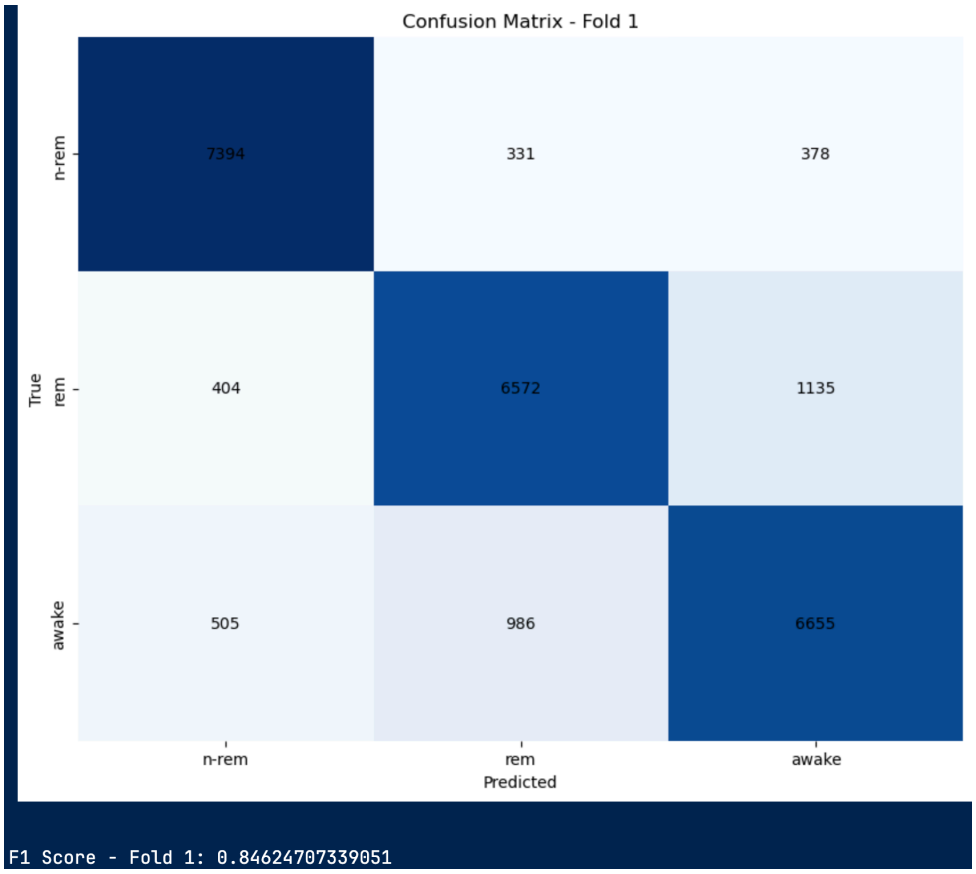

Fig. 5. – Matrice de confusion après amélioration

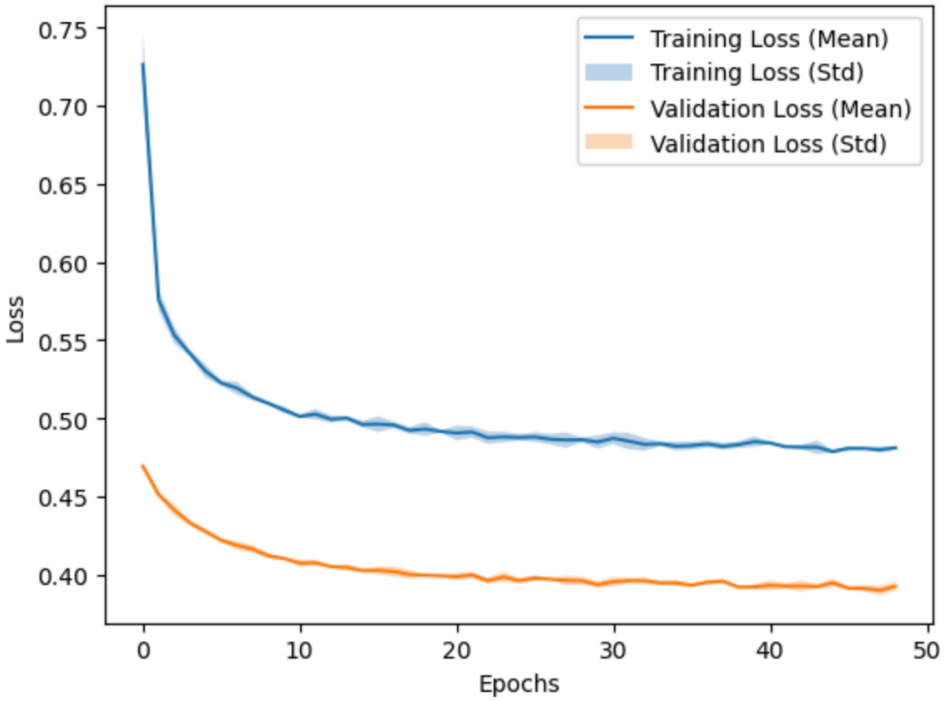

Fig. 6. – Loss par époque après amélioration

Il y a passablement de changement par rapport aux précédents graphes. Premièrement on observe que la courbe de validation est en-dessous de la courbe de training. Cela s'explique par les techniques de régulation mises en place. Cela montre que le modèle peut mieux généraliser et n'overfit pas sur ses data de training. Le pattern qu'il en sort est malgré tout passablement passablement atypique. Cette technique a aussi été testée sur l'expérience précédente, et les observations indiquent que le `dropout` est probablement culprit.

Des courbes très lisses comparées aux deux graphes précédents sont observées. Cela est dû à l'utilisation de régulation ainsi que l'optimizer Adam.

Nous observons qu'il y a très peu de variances entre les modèles. Les différentes `BatchNormalization` peuvent en être la cause, tout comme le nouvel optimizer.

Ensuite, on relève que le nombre d'epochs est proche de 50. En effet, l'early stopping mis en place a pour conséquence que les 3 folds ont un nombre différent d'epochs. La Fig. 7 présente les résultats obtenus.

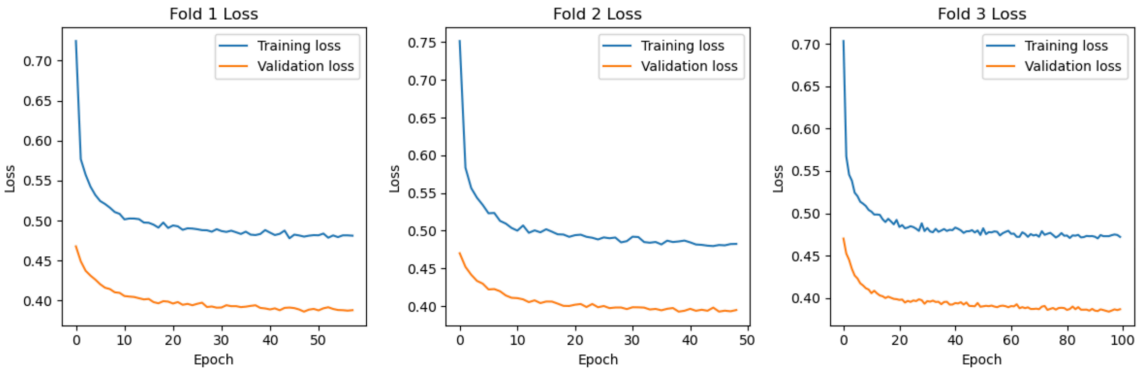

Fig. 7. – Loss par époque selon le fold

Conclusion

Au cours de ce travail, différentes techniques pour l'optimisation de MLP a des fin de classification ont été étudiées et implémentées. Cela a permis d'améliorer notre compréhension de la classification et de la mise en place de réseaux de neurones, particulièrement dans les domaines du choix d'architecture et de préprocessing.

Nous avons pu itérer sur plusieurs modèles et leurs hyper-paramètres associés. Ces différentes itérations ont permis d'améliorer notre compréhension des impacts des différents choix tels que le nombre de neurones présents dans les hidden layers ou encore le choix de l'optimizer.

A titre individuel, nous avons trouvé le processus découverte de nouvelles techniques telles que le `BatchNormalization` ou la métrique `CategoricalAccuracy` divertissant et instructif.