Classification des phases de sommeil des souris avec un MLP.

ARN - PW3

Table des matières

1. Etudiants **3**
2. Introduction **3**
3. Objectifs **3**
4. Première expérience **3**
4.1. Création du modèle MLP 3
4.1.1. Code de création du modèle 3
4.1.2. Résumé du modèle 4
4.2. Resultats 5
4.2.1. Fold 1 5
4.2.2. Fold 2 6
4.2.3. Fold 3 7
4.2.4. F1-Score final 7
4.3. Analyse 8
4.3.1. Modèle Entraîné 8
4.3.2. Historique d'Entraînement 8
4.3.3. Overfitting 8
4.3.4. Matrice de Confusion 8
4.3.5. F1-score 8
5. Deuxième expérience **8**
5.1. Création du modèle MLP 8
5.1.1. Code de création du modèle 8
5.1.2. Résumé du modèle 9
5.2. Resultats 10
5.2.1. Fold 1 10
5.2.2. Fold 2 11
5.2.3. Fold 3 12
5.2.4. F1-Score final 12
5.3. Analyse 13
5.3.1. Modèle Entraîné 13
5.3.2. Historique d'Entraînement 13
5.3.3. Overfitting 13
5.3.4. Matrice de Confusion 13
5.3.5. F1-score 13
6. Compétition **13**
6.1. Nouveautés de notre modèle pour la compétition 13

1. Etudiants

Ce laboratoire a été effectué par :

- Gwendal Piemontesi, ISCL
- Guillaume Trueb, ISCL

2. Introduction

Dans ce travail pratique, nous utilisons des réseaux de neurones de type Perceptron Multi-Couche (MLP) avec Keras pour classifier les phases de sommeil des souris. Les données exploitées sont similaires à celles des précédents travaux, avec un dataset supplémentaire pour l'entraînement et un autre pour les tests. Les notebooks ont été édités et exécutés dans Google Colab.

3. Objectifs

Voici les objectifs principaux de ce labo:

- Préparer et prétraiter les données disponibles afin d'entraîner un réseau de neurones pour une tâche de classification.
- Concevoir des expériences exploitant les données afin de développer un système de classification automatique.
- Concevoir, entraîner et évaluer des réseaux de neurones (MLP) pour résoudre un problème concret.
- Optimiser les hyperparamètres pour définir une architecture de réseau adaptée et obtenir un modèle performant.

4. Première expérience

Dans cette expérience, nous visons à classifier les états « **éveillé** » (awake) et « **endormi** » (asleep) des souris, en regroupant les phases **N-REM** et **REM** sous l'état « endormi ». L'objectif est de développer un modèle MLP capable de prédire ces deux états à partir des données EEG des souris. Nous évaluerons les performances du modèle en utilisant la perte d'entraînement, la perte de validation, la matrice de confusion et le F1-score.

4.1. Création du modèle MLP

Pour la création de notre modèle MLP (Perceptron Multi-Couche), nous avons utilisé une architecture comprenant deux couches cachées avec activation ReLU, associées à une régularisation L2 pour limiter le surapprentissage. Un taux de dropout de 30 % a été appliqué pour améliorer la robustesse du modèle. La couche de sortie utilise une activation sigmoid, adaptée à une classification binaire. Le modèle a été compilé avec la fonction de perte `binary_crossentropy` et l'optimiseur Adam avec un taux d'apprentissage de 0,001.

4.1.1. Code de création du modèle

```
model = Sequential([
        # Entrée avec les 25 caractéristiques
        Input(shape=(25,)),
        # Première couche cachée avec régularisation L2
        Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
        # Deuxième couche cachée avec régularisation L2
        Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
        # Dropout pour éviter le surapprentissage
        Dropout(0.3),
        # Couche de sortie
        Dense(1, activation='sigmoid')
    ])
    # Compiler le modèle avec l'optimiseur Adam
    model.compile(
        loss='binary_crossentropy',
        optimizer=keras.optimizers.Adam(learning_rate=0.001),
        metrics=['accuracy']
    )
```

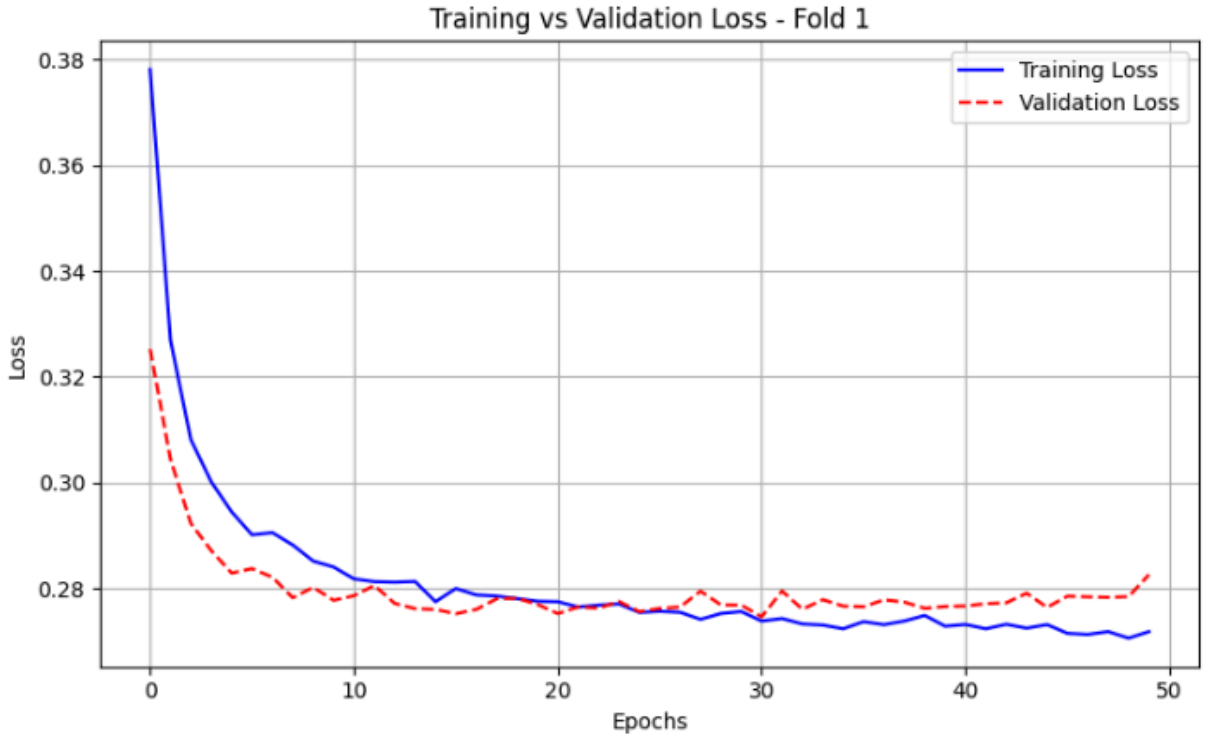
4.1.2. Résumé du modèle

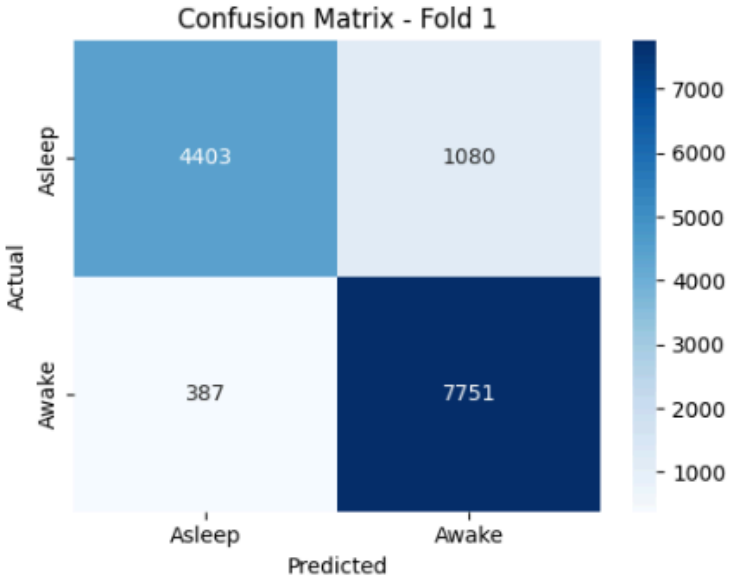
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	1,664
dense_1 (Dense)	(None, 32)	2,080
dropout (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 1)	33

Total params: 3,777 (14.75 KB)
Trainable params: 3,777 (14.75 KB)
Non-trainable params: 0 (0.00 B)

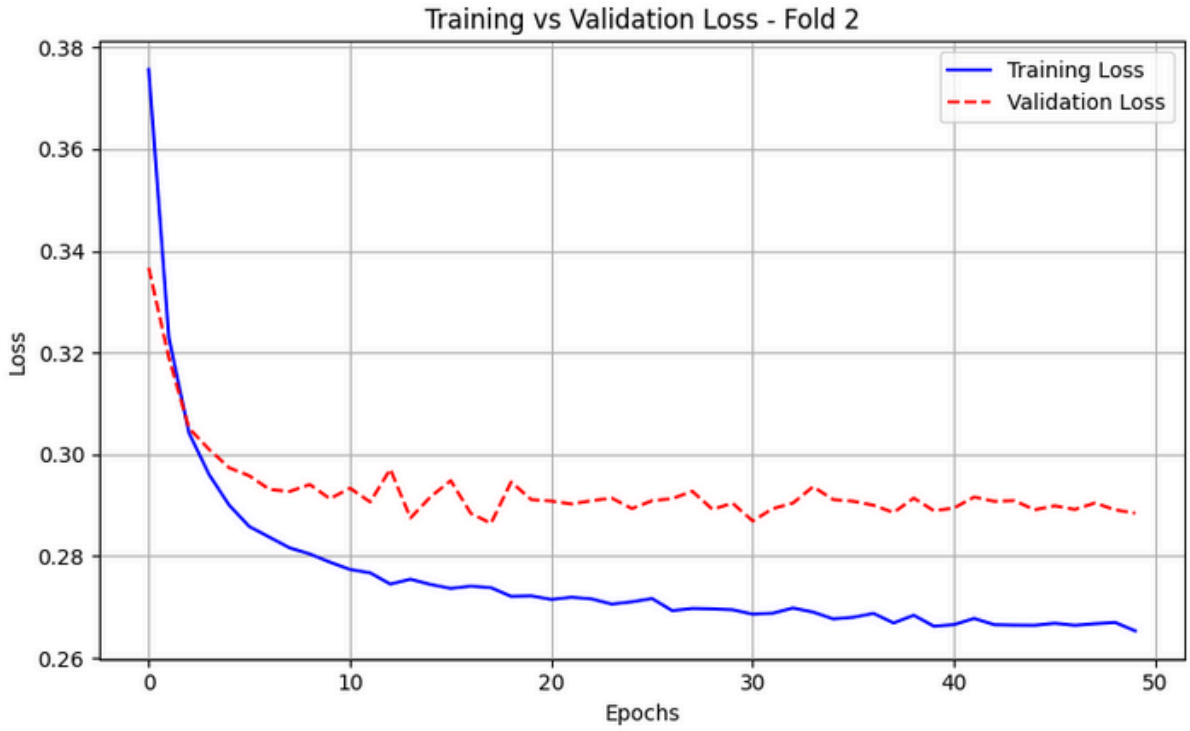
4.2. Resultats

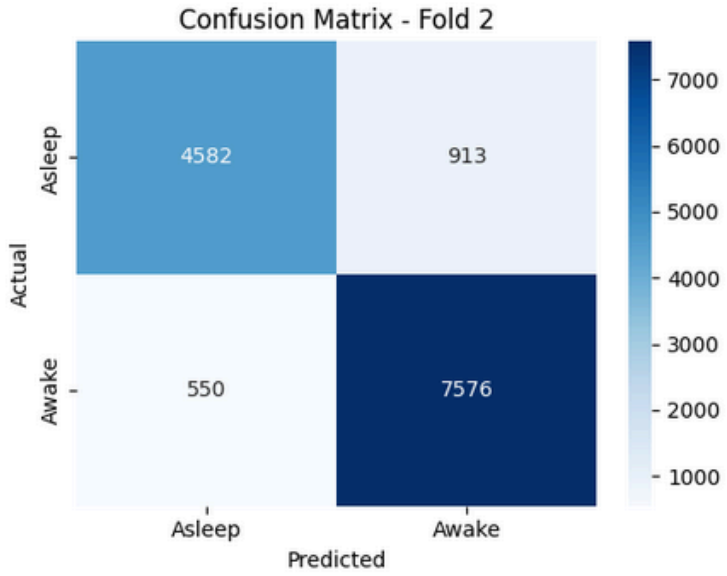
4.2.1. Fold 1



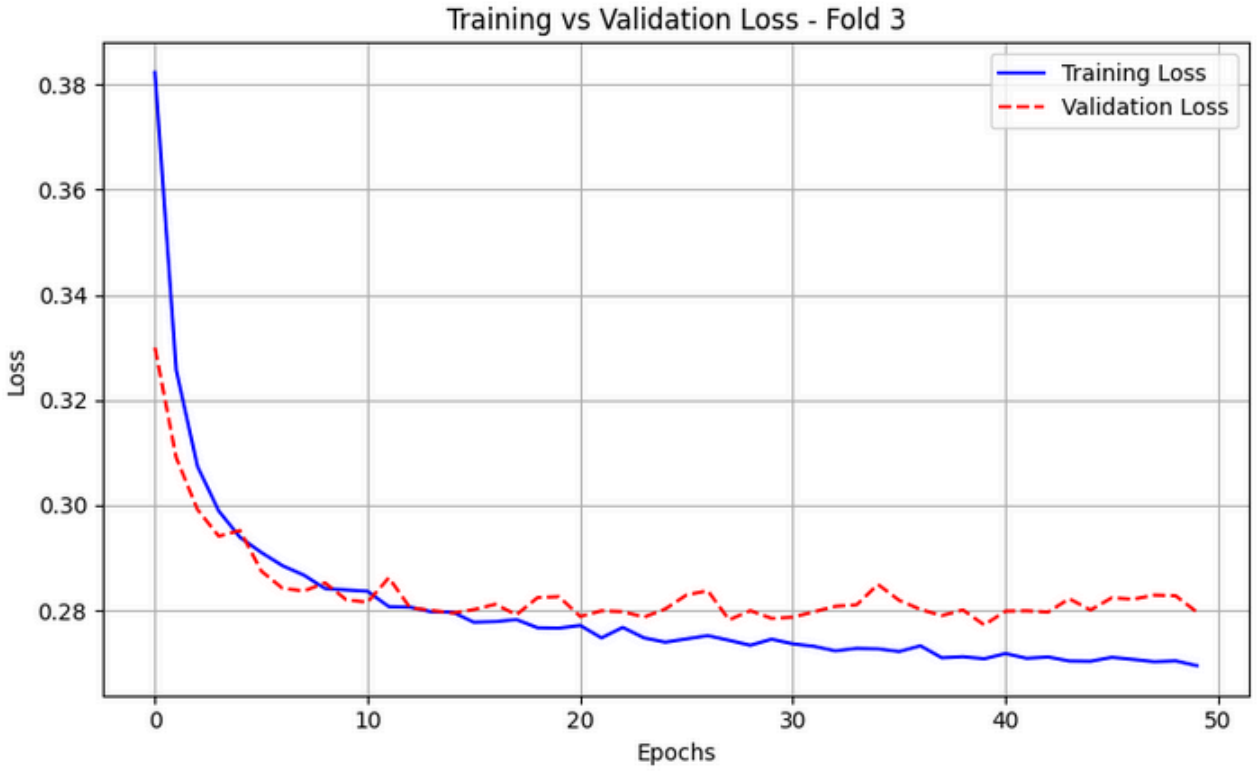


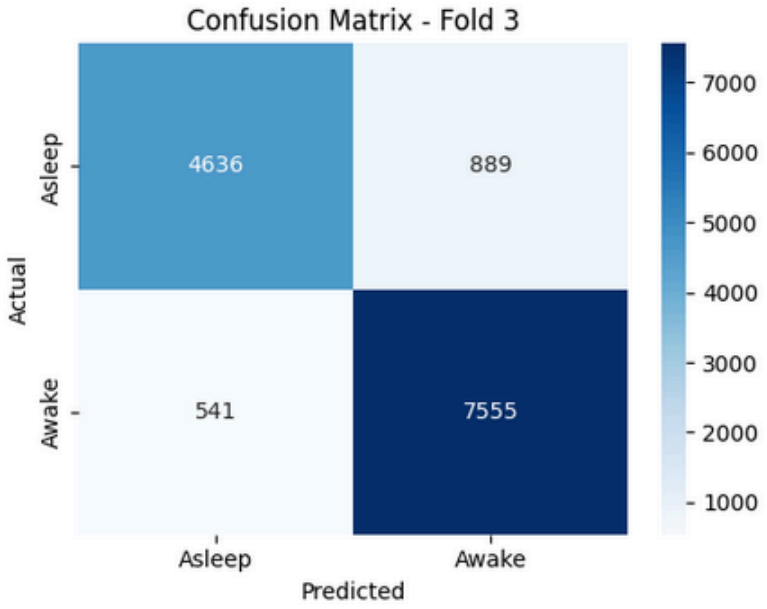
4.2.2. Fold 2


Training vs Validation Loss - Fold 2
Training Loss
Validation Loss
Loss
Epochs
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0
10
20
30
40
50


Confusion Matrix - Fold 2
Actual
Predicted
Asleep
Awake
4582
913
550
7576
7000
6000
5000
4000
3000
2000
1000

4.2.3. Fold 3





4.2.4. F1-Score final

F1-score moyen final : 0.9130

4.3. Analyse

4.3.1. Modèle Entraîné

Le modèle de classification binaire a été entraîné à l'aide de validation croisée sur 50 époques pour chaque fold.

4.3.2. Historique d'Entraînement

Pour les trois folds, les pertes d'entraînement et de validation convergent rapidement après environ 10 époques, ce qui indique une phase d'apprentissage efficace et une stabilisation rapide du modèle.

4.3.3. Overfitting

Un léger risque de surapprentissage est observé dans les trois folds (mais surtout le 2), mais il reste modéré. Un arrêt anticipé pourrait être envisagé pour éviter le surapprentissage et améliorer l'optimisation du modèle.

4.3.4. Matrice de Confusion

Les performances globales du modèle sont satisfaisantes dans les trois folds, bien qu'il y ait des erreurs de classification similaires :

- Faux négatifs (prédiction "Awake" pour une véritable classe "Asleep") : entre 889 et 1080.
- Faux positifs (prédiction "Asleep" pour une véritable classe "Awake") : entre 387 et 550.

4.3.5. F1-score

Le F1-score moyen pour les trois folds est de 0.9130, ce qui reflète une très bonne performance globale du modèle.

5. Deuxième expérience

Dans cette expérience, nous allons classifier les états des souris en trois catégories : « **éveillé** » (awake), « **N-REM** » et « **REM** ». L'objectif est de développer un modèle MLP capable de prédire ces trois états à partir des données EEG des souris. Les performances du modèle seront évaluées en utilisant la perte d'entraînement, la perte de validation, la matrice de confusion et le F1-score.

5.1. Création du modèle MLP

Pour la création de notre modèle MLP (Perceptron Multi-Couche), nous avons utilisé une architecture comprenant deux couches cachées avec activation ReLU, une régularisation L2 pour limiter le surapprentissage, ainsi qu'un taux de dropout de 30 % pour améliorer la robustesse du modèle. La couche de sortie utilise une activation softmax pour la classification en trois classes. Le modèle a été compilé avec la fonction de perte `categorical_crossentropy`, adaptée à la classification multiclasse, et l'optimiseur Adam avec un taux d'apprentissage de 0,001.

5.1.1. Code de création du modèle

```
model = Sequential([
        # Entrée avec les caractéristiques
        Input(shape=(25,)),
        # Première couche cachée avec régularisation L2
        Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
        # Deuxième couche cachée avec régularisation L2
        Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
        # Dropout pour éviter le surapprentissage
        Dropout(0.3),
        # Couche de sortie pour 3 classes
        Dense(3, activation='softmax')
    ])
```

```
# Compilation du modèle avec categorical_crossentropy pour la classification multiclasse
model.compile(
    loss='categorical_crossentropy',
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=['accuracy']
)
```

5.1.2. Résumé du modèle

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	1,664
dense_1 (Dense)	(None, 32)	2,080
dropout (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 3)	99

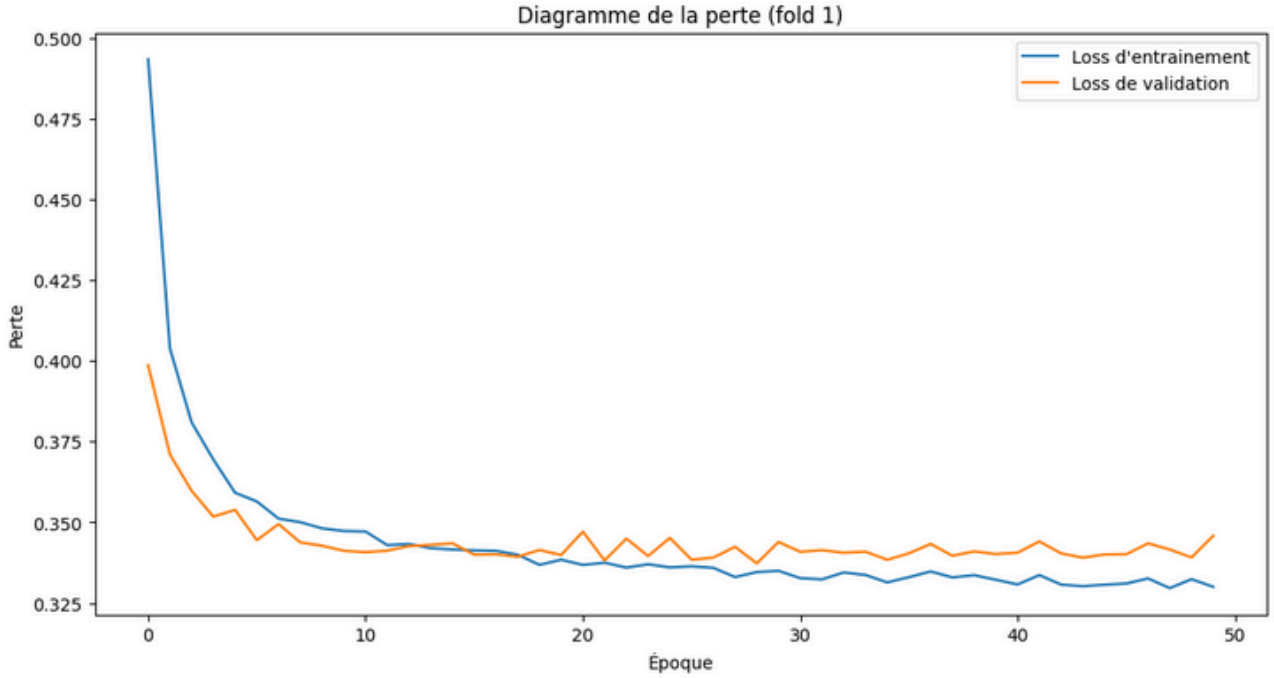
Total params: 3,843 (15.01 KB)

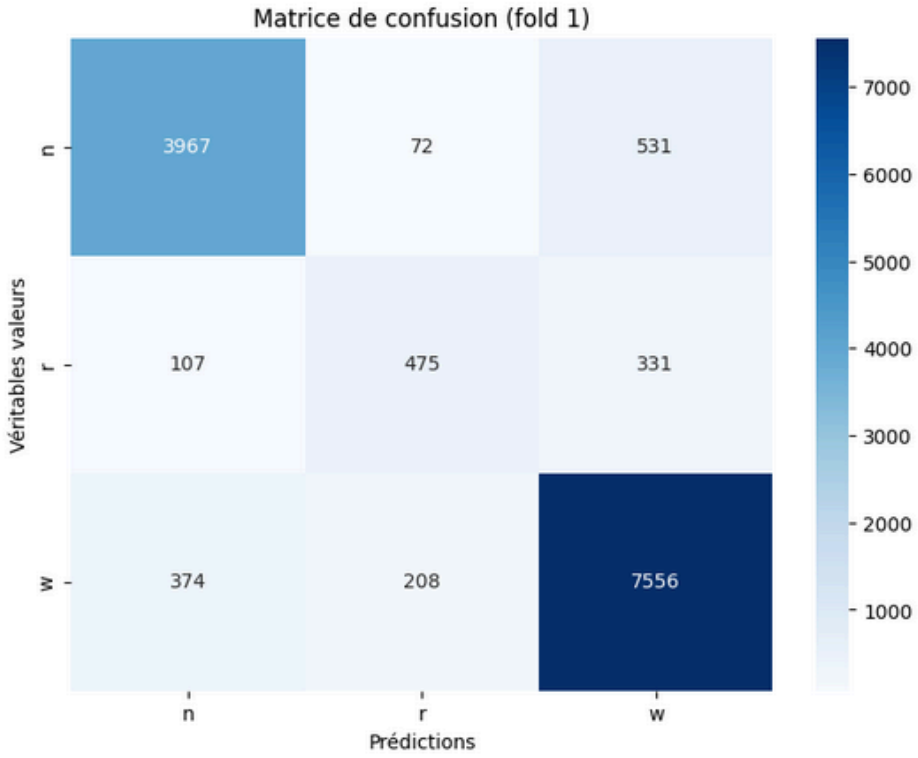
Trainable params: 3,843 (15.01 KB)

Non-trainable params: 0 (0.00 B)

5.2. Resultats

5.2.1. Fold 1





5.2.2. Fold 2

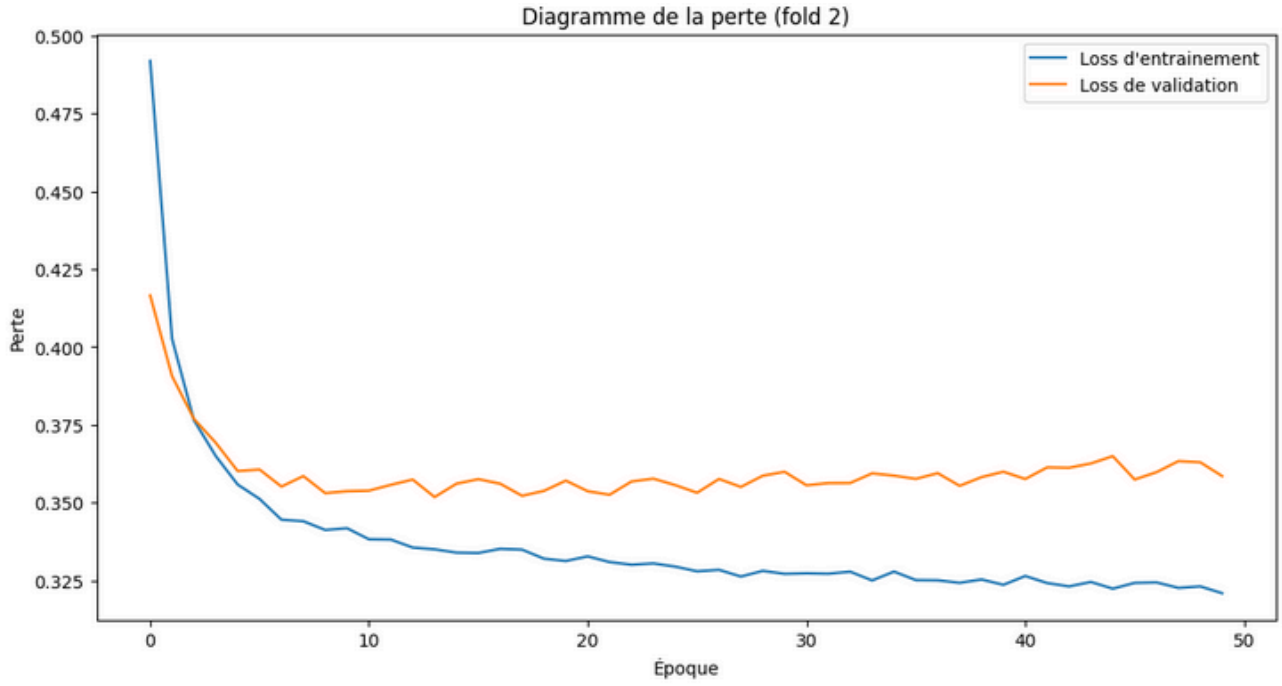
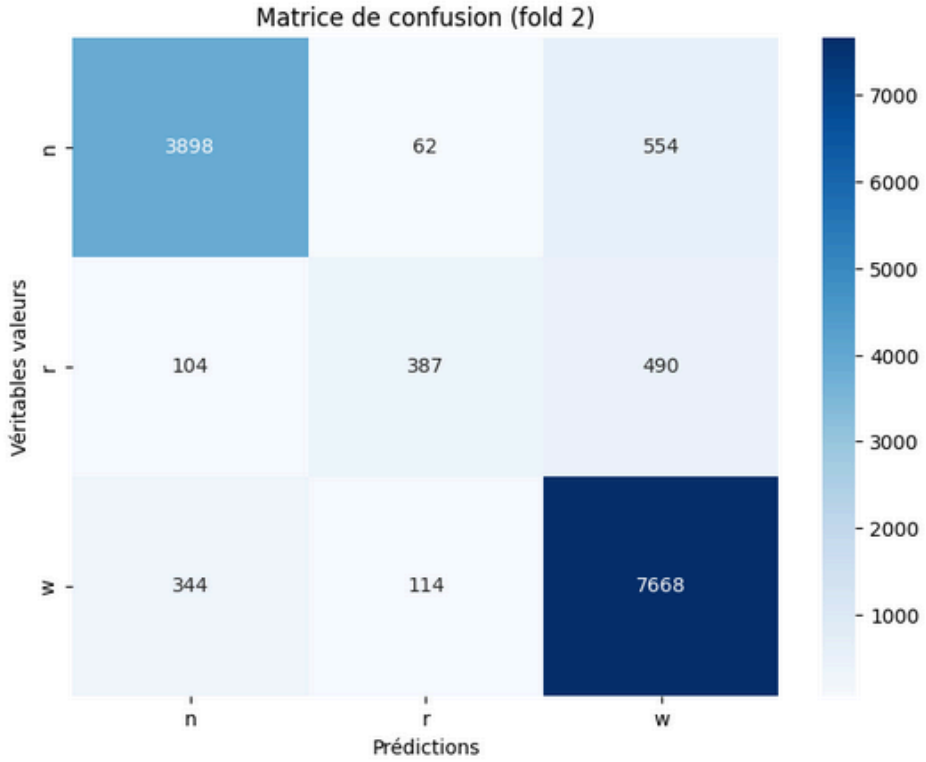
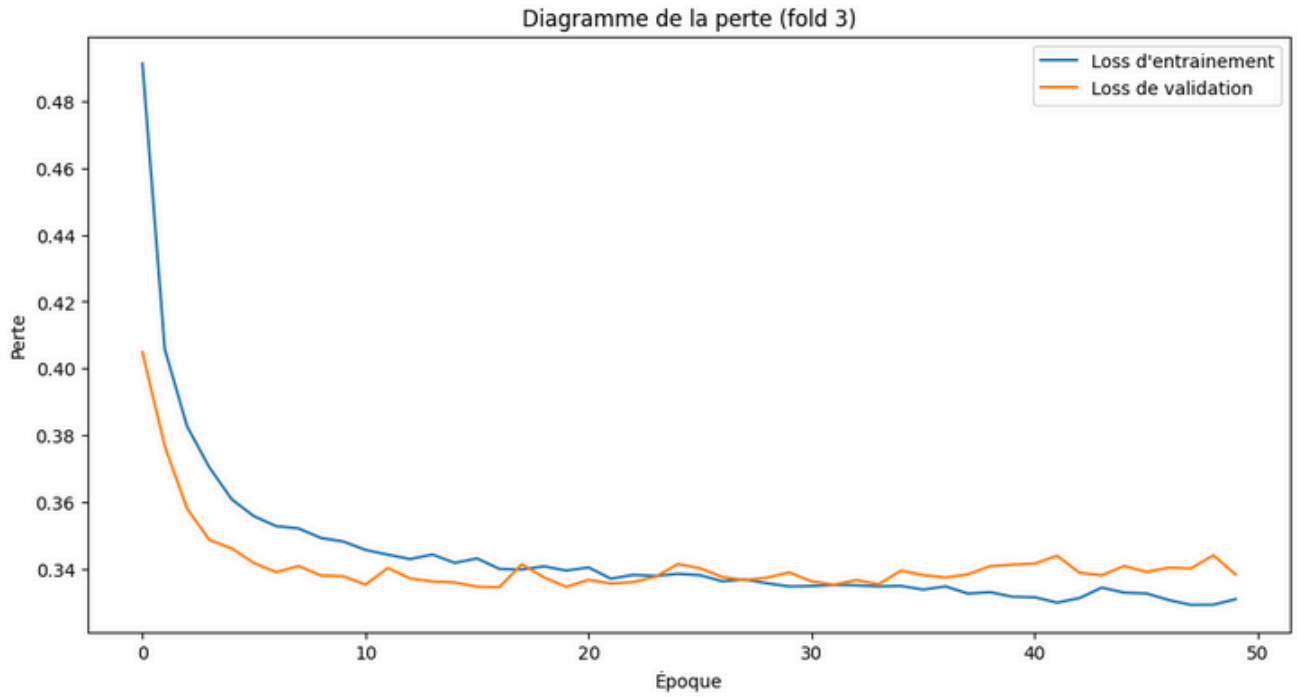
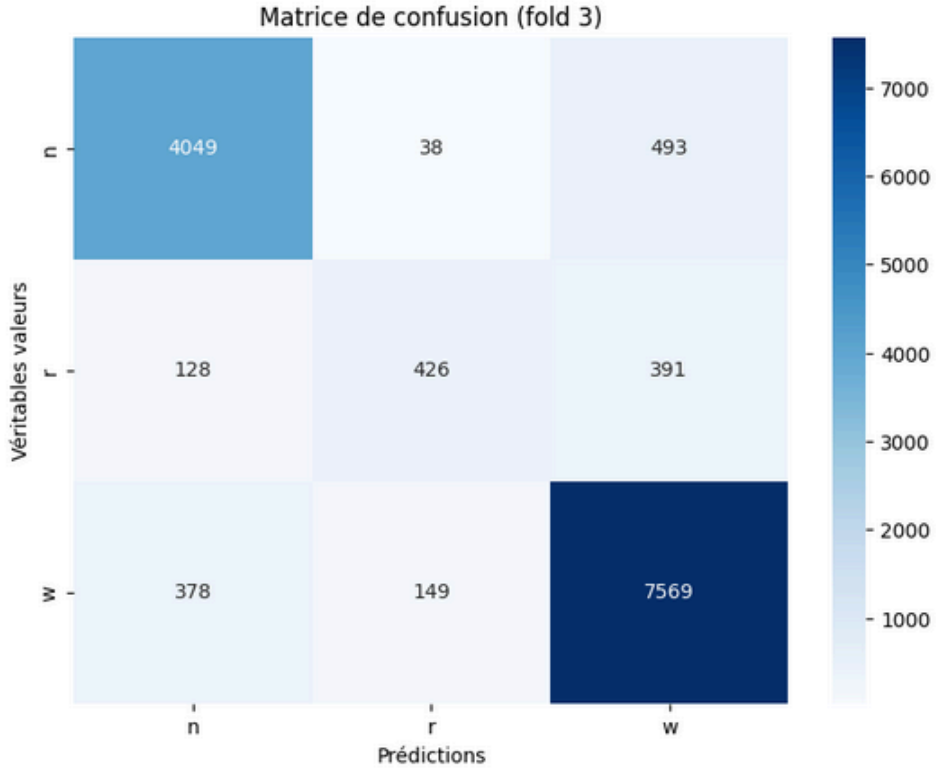


Diagramme de la perte (fold 2)
Loss d'entrainement
Loss de validation
Perte
Époque
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0
10
20
30
40
50


Matrice de confusion (fold 2)
Véritables valeurs
Prédictions
n
r
w
3898
62
554
104
387
490
344
114
7668
7000
6000
5000
4000
3000
2000
1000

5.2.3. Fold 3





5.2.4. F1-Score final

F1-score moyen final : 0.8809

Écart type de la précision : 0.0027

5.3. Analyse

5.3.1. Modèle Entraîné

Le modèle de classification multiclass a été entraîné à l'aide de validation croisée sur 50 époques pour chaque fold.

5.3.2. Historique d'Entraînement

Convergence rapide dans les 10 premières époques pour tous les folds. Les pertes se stabilisent autour des époques 15-20, avec des valeurs finales d'environ 0,33 pour l'entraînement et 0,34-0,36 pour la validation.

5.3.3. Overfitting

Léger surapprentissage observé, particulièrement dans le fold 2 (écart de 0,04 entre pertes d'entraînement et validation). Un arrêt anticipé vers l'époque 20 serait bénéfique pour optimiser la généralisation.

5.3.4. Matrice de Confusion

- Classe n: Bonne reconnaissance (3900-4050 prédictions correctes)
- Classe r: Performance limitée (390-475 prédictions correctes)
- Classe w: Excellente reconnaissance (7550-7670 prédictions correctes)

5.3.5. F1-score

Précision moyenne sur tous les folds: 0,8835 Écart type de la précision: 0,0048 (indiquant une bonne stabilité du modèle) Meilleure performance: Fold 5 avec précision de 0,8917

La classe r reste difficile à classifier correctement, principalement en raison du déséquilibre des classes. La faible variabilité des performances entre les folds confirme la robustesse du modèle.

6. Compétition

Cette compétition regroupe les groupes de notre classe afin de déterminer qui a su développer le meilleur modèle. La compétition repose sur le fait que notre modèle sera testé sur des données inconnues, et l'objectif est d'obtenir le meilleur modèle, basé sur le F1-score.

Pour la compétition, nous avons repris le modèle de la deuxième expérience en le modifiant légèrement, comme indiqué (au moins une nouvelle idée).

6.1. Nouveautés de notre modèle pour la compétition

- 5 folds au lieu de 3.
- Une nouvelle couche cachée.
- Plus de dropout.
- Early stopping.
- Mise à jour automatique du learning rate.