ARN - Travail Pratique 3

Auteurs : Nicolas Duprat & David Berger

First Experiment

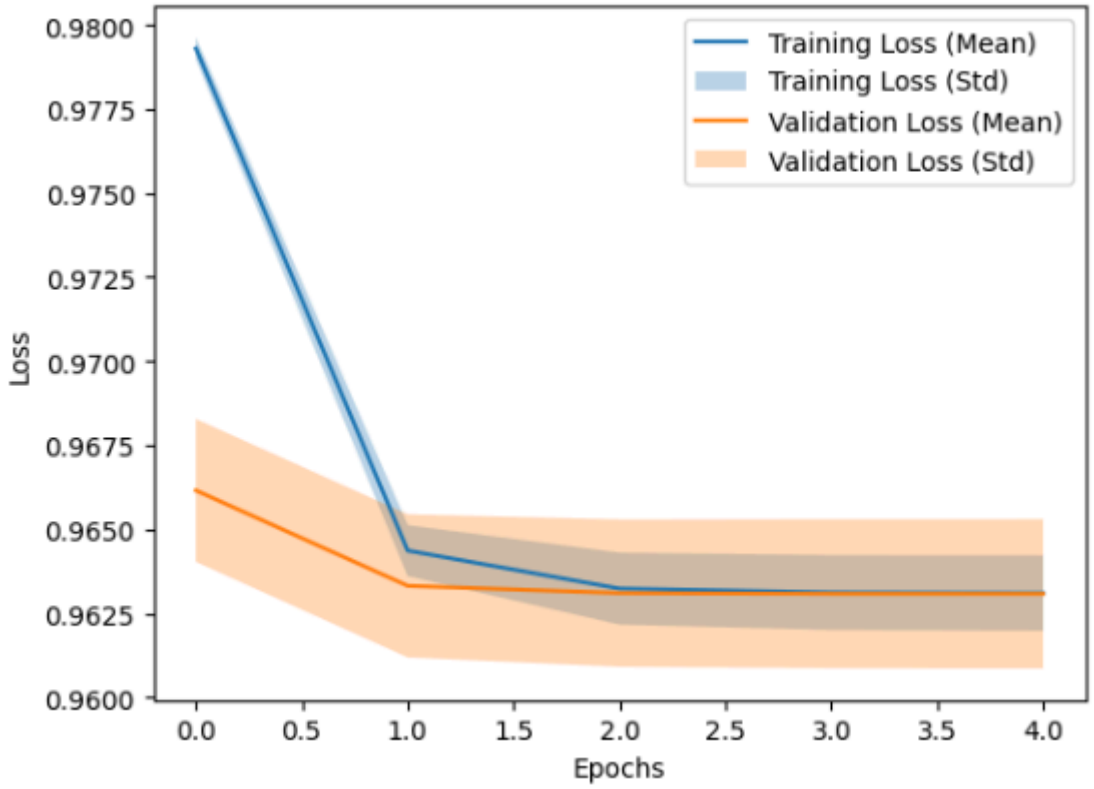
Pour la première expérimentation, nous devons classifier les états de sommeil d'une souris en deux classe, `awake` et `asleep`. Pour ce faire, nous avons normaliser les données de telle sorte à ce que `awake` = 1, et `asleep` = -1.
Ensuite il fallait diviser nos données en 3 parties (en utilisant la K-Cross Validation).

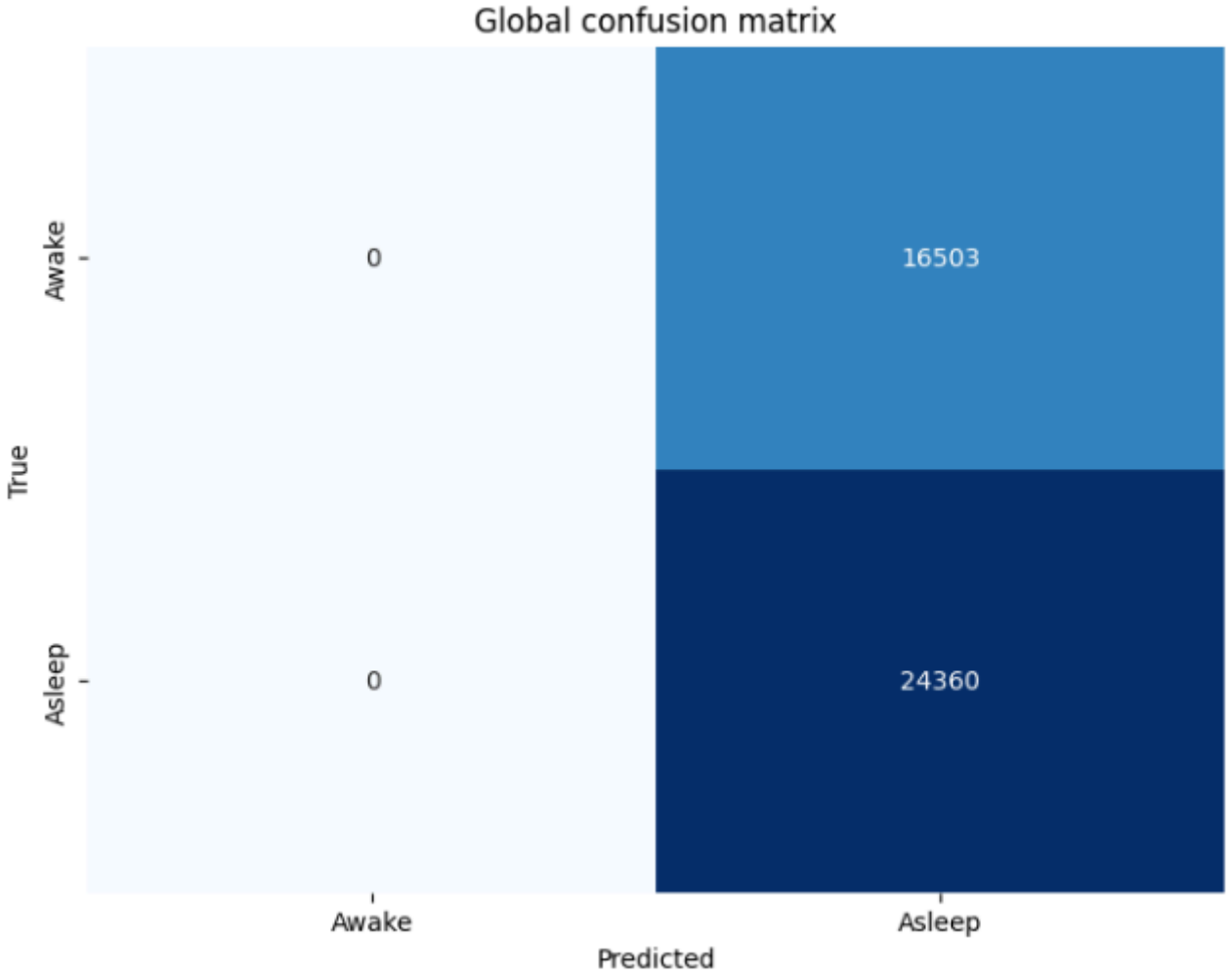
Pour le premier essai, nous avons repris exactement le même percpetron fait dans le travail pratique précédent, comme montré dans l'image ci-dessous.

```
mlp = keras.Sequential([
    layers.Input(shape=(25,)),
    layers.Dense(128, activation="tanh"), # Try different numbers of hidd
    layers.Dense(32, activation="tanh"),
])

# Experiment with hyperparameters here:
# momentum: [0, 0.8, 0.9, 0.99]
# learning_rate: [0.1, 0.01, 0.001, 0.0001]
mlp.compile(
    optimizer=keras.optimizers.SGD(learning_rate=0.0001, momentum=0.99),
    loss="mse",
)
```

Voici le résultat obtenu :





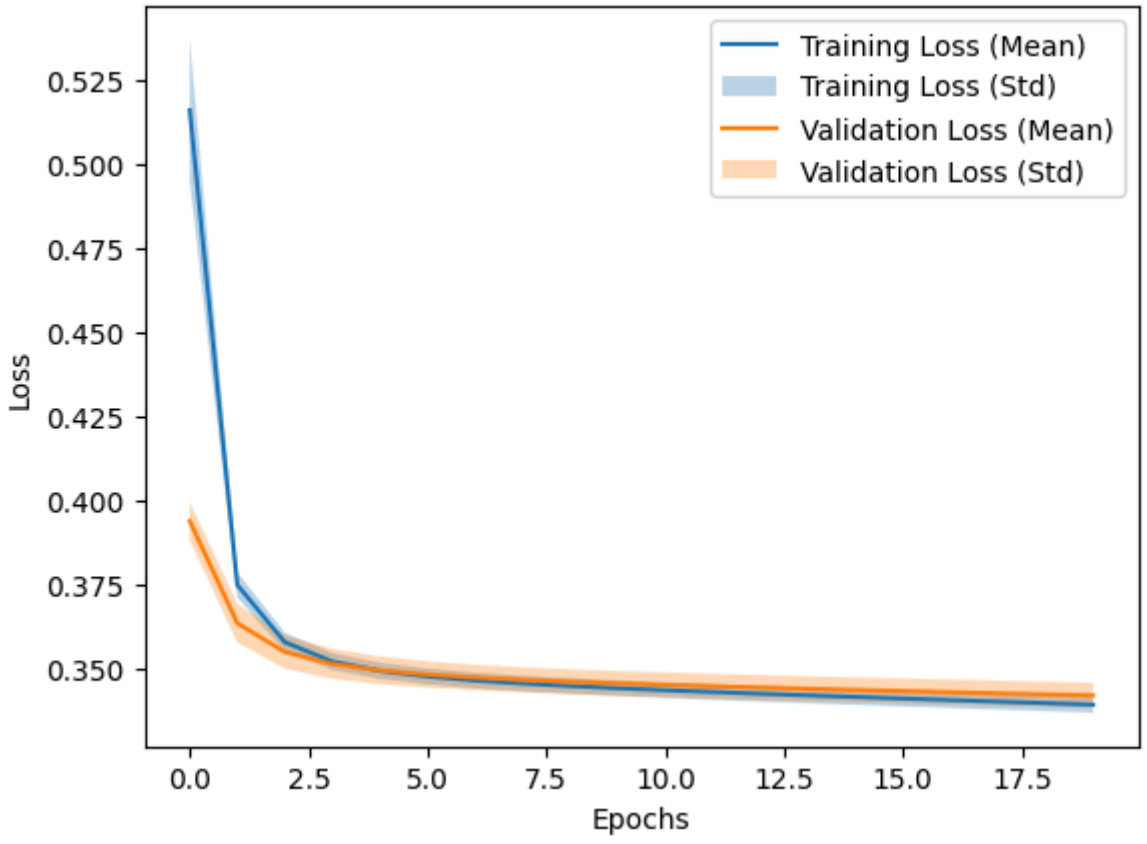
On remarque très facilement que ce n'est pas un résultat qui nous satisfait, tout est prédit en tant que `asleep`. Même en essayant de modifier les hyperparamètres, ca ne change rien.

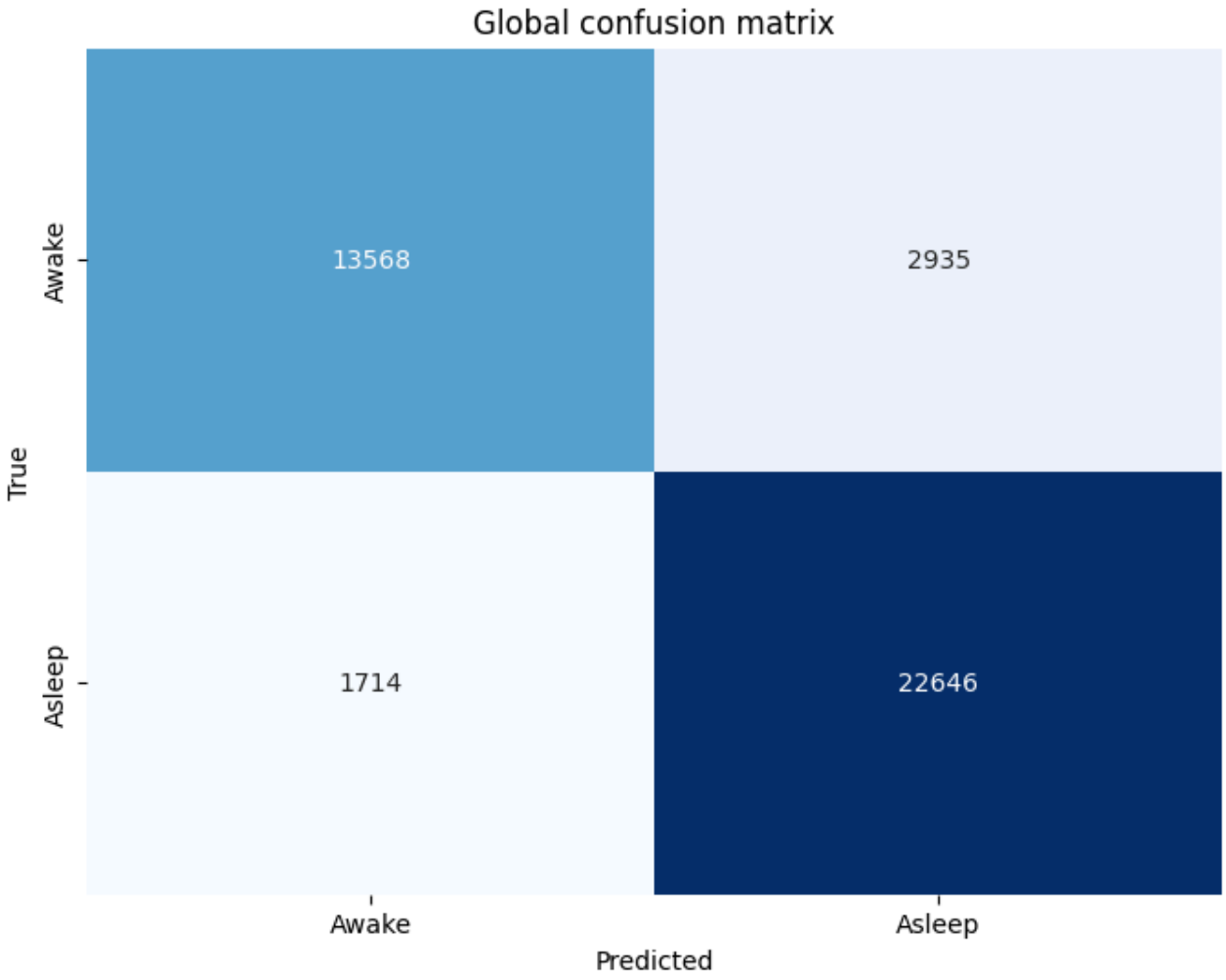
En utilisant un scaler et sa fonction `transform`, on obtient des meilleurs résultats. Nous avons joué un petit peu avec les hyperparamètres, et voici ce que ça nous donne.

```
history = mlp.fit(
    #x=mice_eeg_input[train_index], y=mice_eeg_output[train_index],
    x=scaler.fit_transform(mice_eeg_input[train_index]), y=mice_eeg_output[train_index],
    #validation_data=(mice_eeg_input[test_index], mice_eeg_output[test_index]),
    validation_data=(scaler.transform(mice_eeg_input[test_index]), mice_eeg_output[test_index]),
    epochs=10
)
```

```
mlp = keras.Sequential([
    layers.Input(shape=(25,)),
    layers.Dense(32, activation="tanh"), # Try different numbers of hidden neurons here (e.g. 2, 4, 8, 32, 128)
    layers.Dense(8, activation="tanh"),
])

# Experiment with hyperparameters here:
# momentum: [0, 0.8, 0.9, 0.99]
# learning_rate: [0.1, 0.01, 0.001, 0.0001]
mlp.compile(
    optimizer=keras.optimizers.SGD(learning_rate=0.001, momentum=0.9),
    loss="mse",
)
```



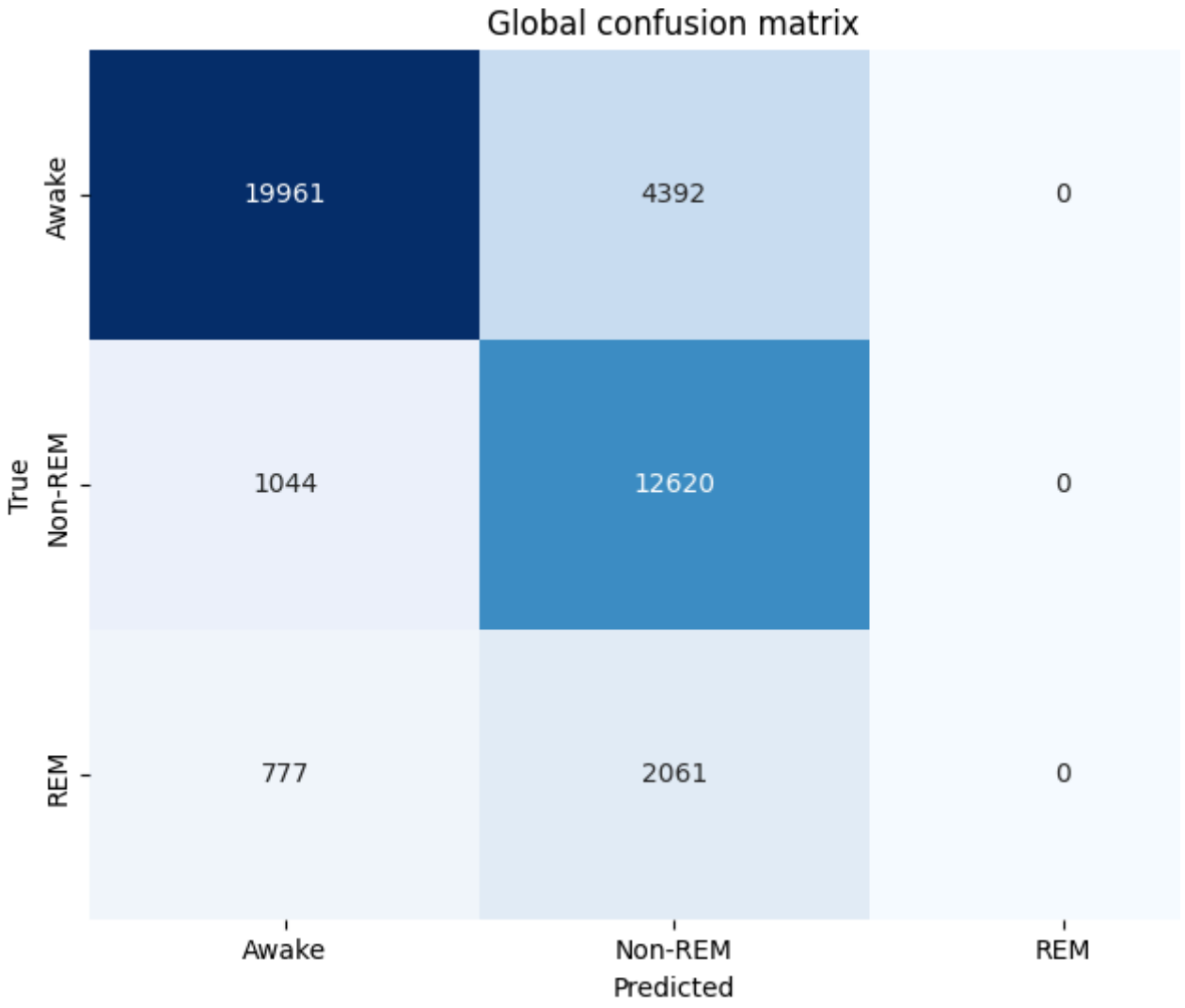


Nous n'utilisons seulement 20 epochs car on peut déjà voir que la fonction de validation perd de la valeur part rapport à celle d'entrainement, on ne veut pas causer d'overfitting. Ce modèle nous donne un F1-Score de `0.9068930905471627`, qui est satisfaisant pour passer à la suite.

Second Experiment

Pour cette deuxième expérimentation, nous avions trois classifications d'état de la souris : `awake`, `non-REM` et `REM`. Nous avons normalisé nos données de telle sorte à ce qu'on ait respectivement 0, 1 et 2 pour ces états. Nous avons donc du également modifier le code pour les matrices de confusion par rapport à celui donné, car cette fois-ci nous sommes en 3x3.

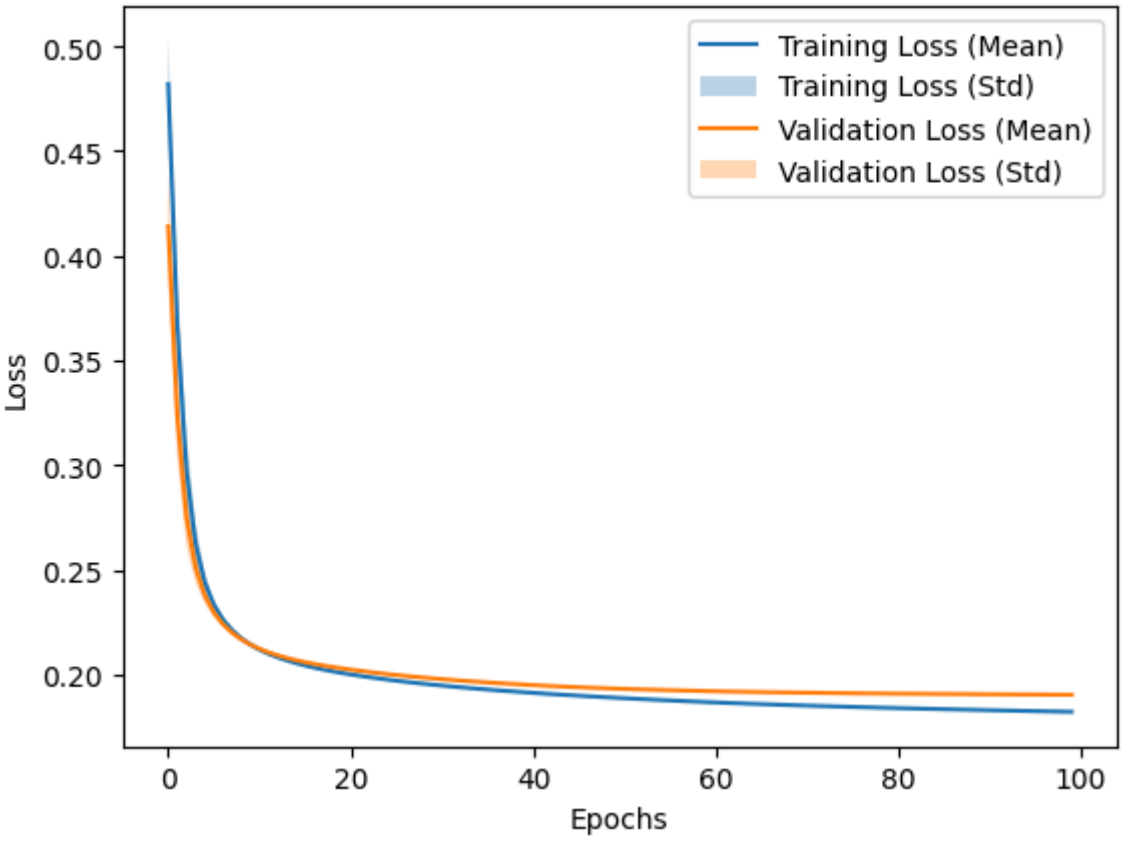
Premièrement, nous avons repris le même modèle que pour la partie 1.

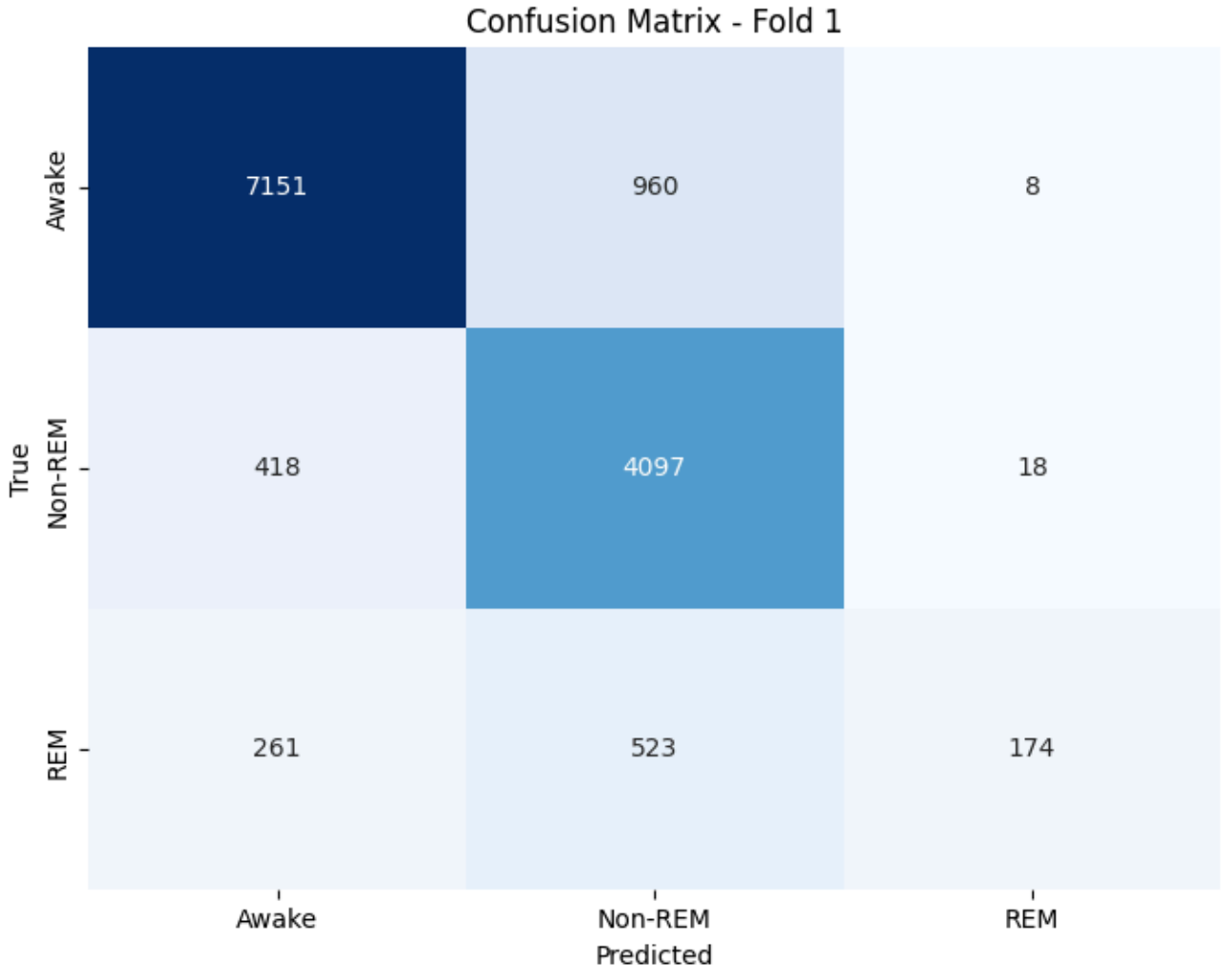


On peut vite remarquer que le MLP ne prédit aucun cas de REM, ce qui n'est pas voulu malgré le F-1 Score qui était correct (~0.78).

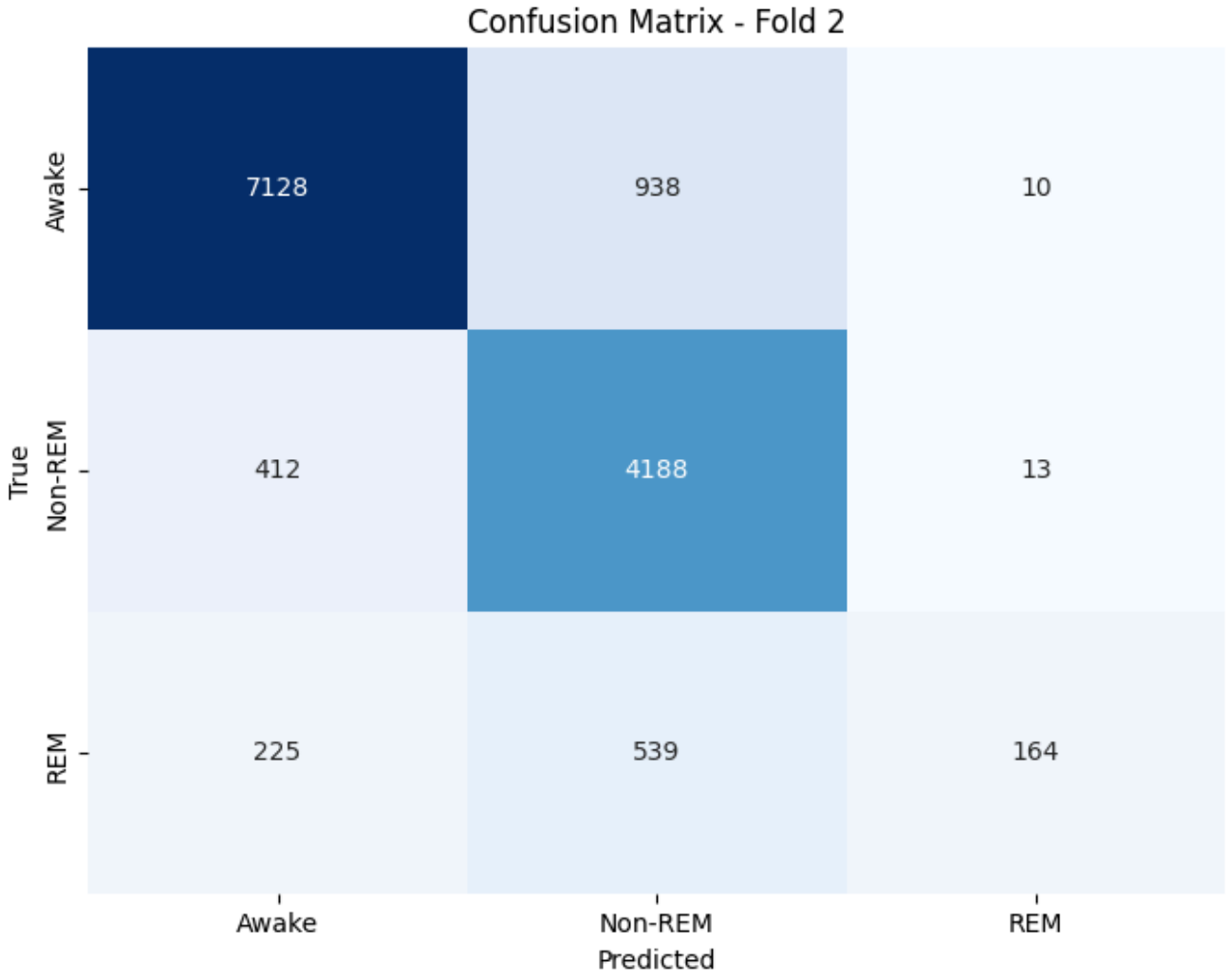
Nous avons décidé de changer de fonction d'activation pour espérer voir une amélioration et ça s'est produit avec la fonction ReLu. Après plusieurs essais, voici les hyperparamètres utilisés :

- Hidden Layer 1 : 32 neurones
- Hidden Layer 2 : 8 neurones
- Learning Rate : 0.001
- Momentum : 0.9
- Epoch : 100
  Nous nous arrêtons à 100 epochs, car à partir d'ici l'écart entre l'entrainement et la validation grandit de plus en plus alors que la validation stagne. C'est pour éviter de l'overfitting.

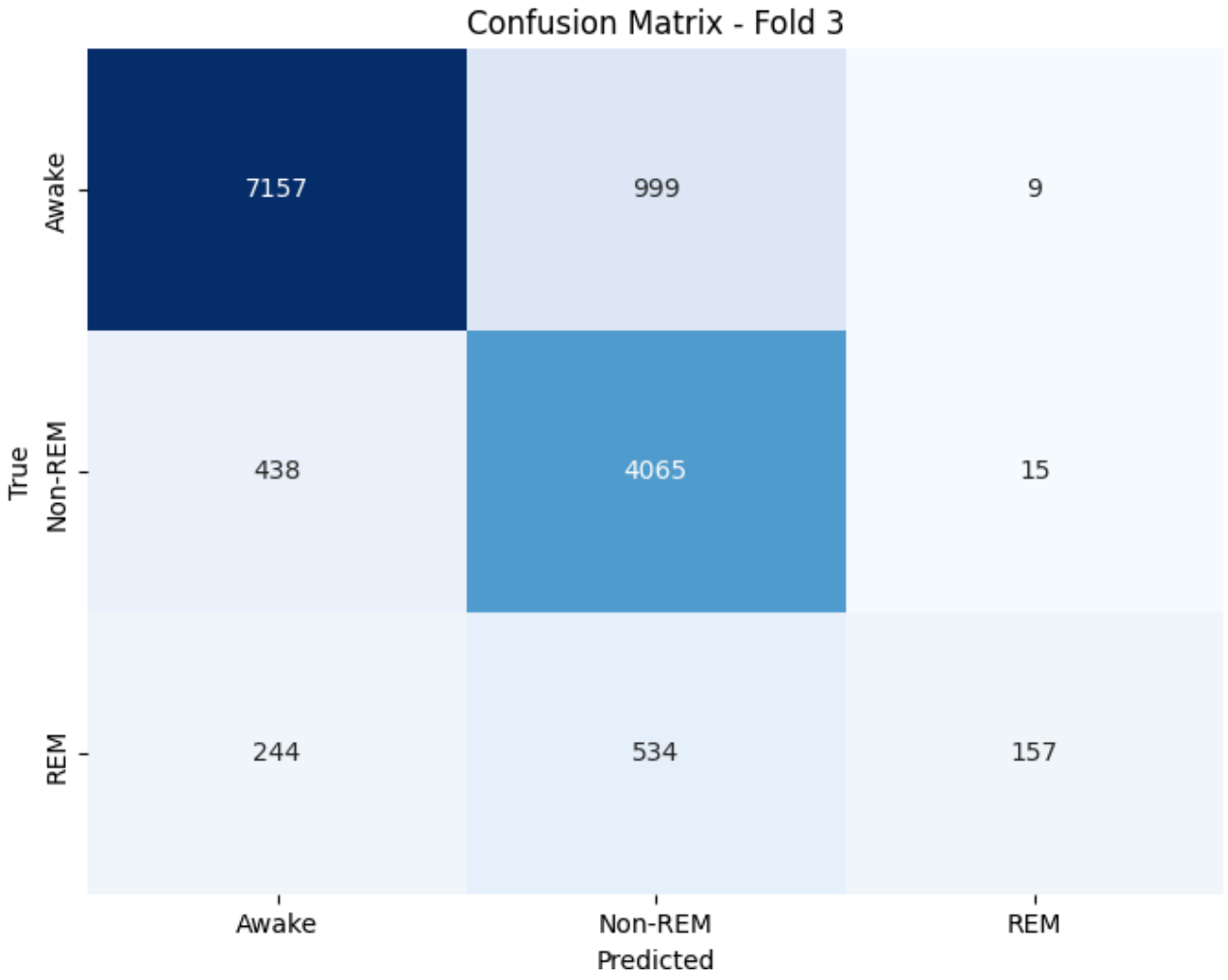




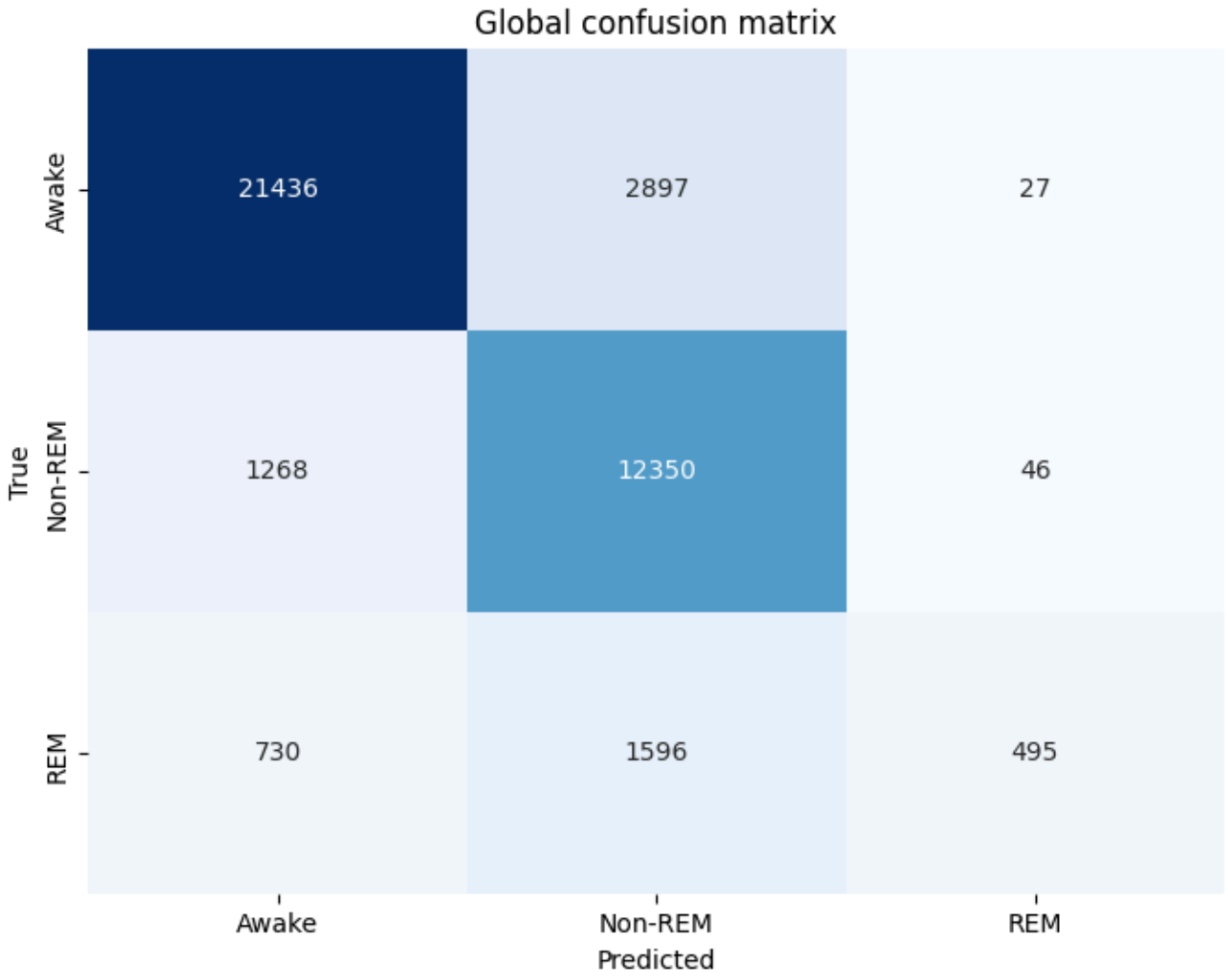
F1-Score Fold 1 : 0.8253341928537866



F1-Score Fold 2 : 0.8296356442464352



F1-Score Fold 3 : 0.8220403547430408

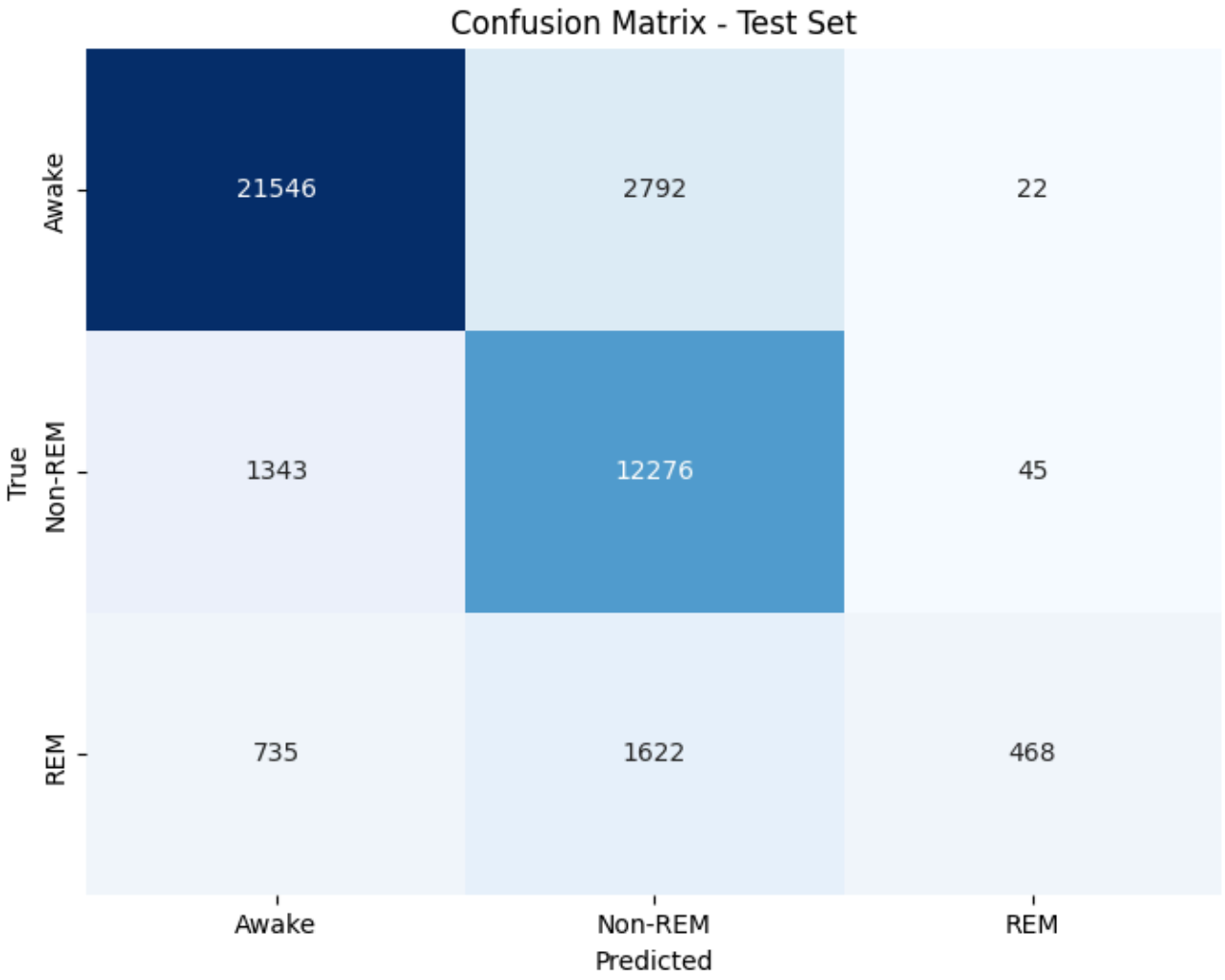


Bien que ce modèle ne soit pas parfaitement optimisé, le résultat est quand même satisfaisant. Avec un F1-Score global de 0.8256700639477543.

Competition

Nous avons donc utilisé le même code que pour la partie 2, mais cette fois-ci avec le dataset de test.

Voici la matrice de confusion sur ce dataset :



Avec un F1-Score de 0.8251050209911578, c'est un résutlat satisfaisant.

Analyse

Premièrement, nous avons un résultat qui nous satisfait. Avec un F1-Score comme celui là, c'est une bonne première expérience. Sur 40849 données total, notre précision est d'environ 84%.

Deuxièmement, nos modèles ne font pas d'overfitting ou d'underfitting. Nous avons choisis les bons hyperparamètres et arrêtez l'entrainement quand il le fallait.

Cependant, notre résultat n'est pas encore parfait. On peut remarquer un problème dans la prédiction de l'état `REM`, où seulement 16% cas de son bien prédit. `REM` avait depuis le début l'air d'être l'état le plus difficile à prédire, pour l'expérience 2 il n'y avait aucune détection correcte pour cet état au départ. Cette résultat peut être du au fait que c'est l'état dont nous avons le moins de données, seulement 7% du dataset.

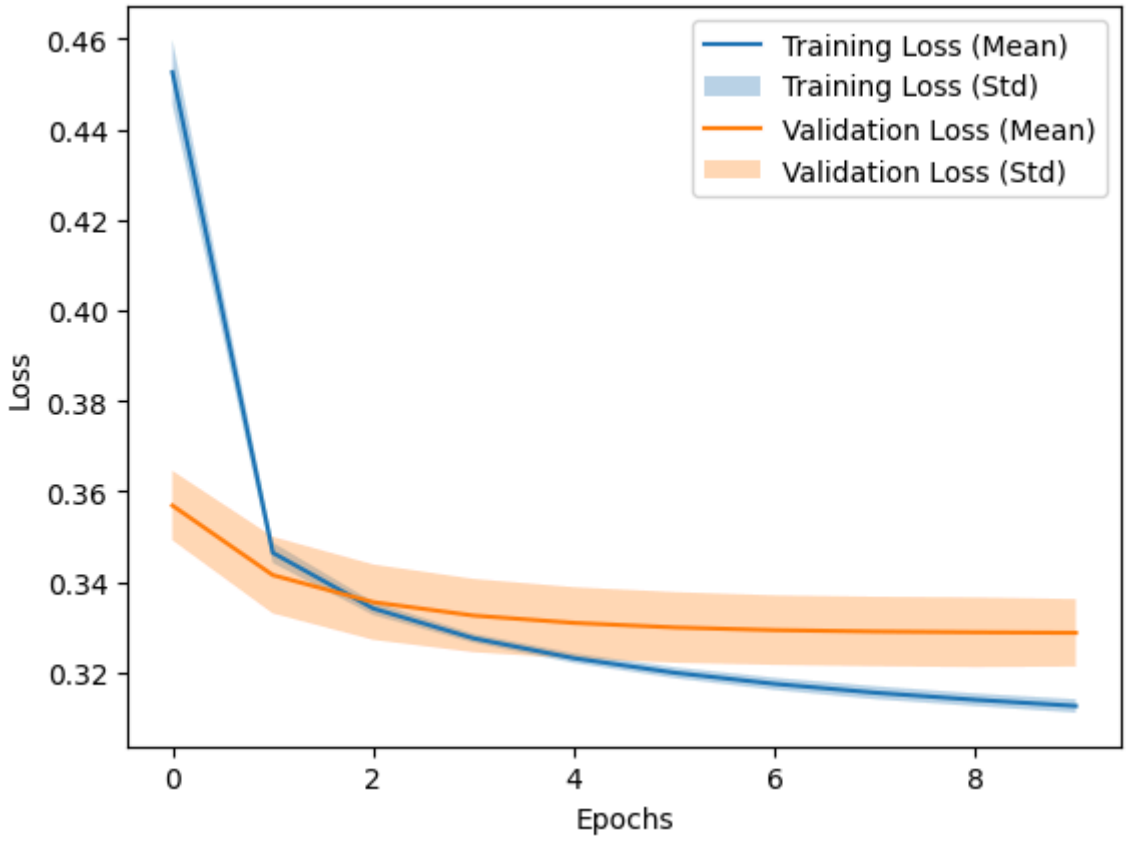
Une amélioration est encore possible, le modèle n'est sûrement pas le meilleur (même si c'est celui de nos quelques tests). On pourrait changer les couches cachées ou en ajouté. Regarder plus précisément les hyperparamètres, un par an, ajusterait aussi de peu les prédictions.

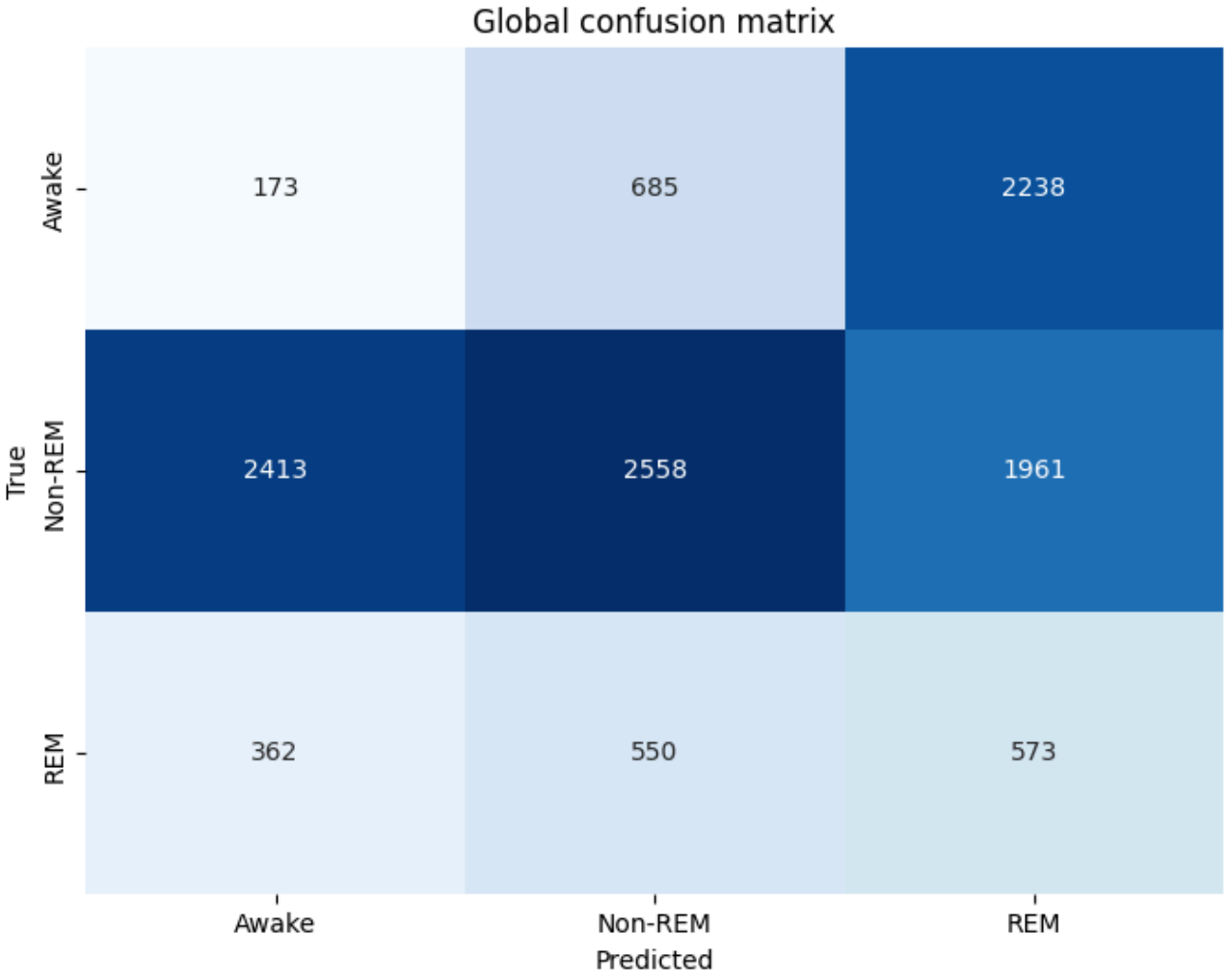
Un dernier point à aborder est que l'utilisation de SGD n'est peut-être pas la plus optimale, d'autres optimiseur existent et nous en abordant un (très brièvement) dans le prochain point.

Nouvelle idée

Nous nous sommes rendus compte après avoir tout effectué que nous avions oublié d'implémenter une nouvelle idée comme demandé. Nous avons décidé de ne pas tout recommencer mais nous avons tout de même voulu en essayer une après coup sur le dataset d'entrainement (sans pour autant chercher à vraiment l'optimiser).

Nous avons implémenter l'optimiseur Adam à la place de SGD tout en gardant la même structure du modèle. Pour les hyperparamètres, nous utilisons ceux par défauts. Nous avons également ajouté une valeur d'accuracy dans notre modèle.





Avec un F1-Score global de 0.11416899047295063, nous remarquons bien que ce modèle n'est pas optimal du tout... Malheureusement, nous n'avons pas réellement chercher plus loin pour le corriger.