Bordeaux INP
AQUITAINE

# Challenge Mewo

GISSAUD Alexandre - MARGUERITTE Gaëtan

December 2021

## 1 Le challenge Mewo

### 1.1 Introduction du challenge

Ce document rapporte l'activité et la progression de l'équipe de Alexandre Gissaud et Gaëtan Margueritte sur le challenge Mewo dans le cadre de l'enseignement "Apprentissage Automatique" encadré par Baptiste Pesquet.

Le challenge Mewo est un challenge disponible à l'adresse `https://challengedata.ens.fr/participants/challenges/43/`. Il consiste en un problème de classification multiple. Les données en entrées sont la sortie de réseaux de neurones spécialisés dans le traitement de signal sonore pour la classification. Ainsi, ces données sont regroupées dans un vecteur de 289 classes. La sortie attendue est un vecteur binaire de 248 classes. C'est donc un problème de décision.

### 1.2 Déroulement et progression du travail

Le code pour ce projet est disponible à l'adresse: `https://github.com/Carpilieng/MewoChallenge`. Ce dépôt a été utilisé pendant toute la durée du projet.

N'ayant pas d'expérience dans le machine learning auparavant, nous avons d'abord commencé par étudier et pratiquer dans des exercices de Kata `https://www.bpesquet.fr/mlkatas/`.

Une fois cela fait, nous avons essayé une approche par réseau neuronal en utilisant un modèle simple à 3 couches intermédiaires dense. Le modèle avait pour entrée le vecteur de 289 valeurs et pour sortie un vecteur de 248 valeurs. La fonction perte utilisée a été la "categorical crossentropy". La décision dans le vecteur de classe été prise selon un seuil fixe de 0.50, c'est à dire que pour chacune des valeurs du vecteur de sortie du réseau neuronal, on arrondie la valeur à 0 ou à 1 pour former le vecteur binaire de classes. Nous avons expérimenté plusieurs hyper-paramètres sans succès avec une très faible précision de moins de 10%. Nous avons donc abandonnée cette approche en raison de la faible réussite mais également car nos perspectives d'améliorations étaient nulles.

Nous avons alors essayé une approche basée sur un seuil variable au lieu d'un seuil fixe mais par conséquent sans l'utilisation de réseau neuronal. La définition du seuil par classe s'est faite par une forme de moyenne pondérée. L'algorithme

utilisé sera plus détaillé dans la section 2. Cette approche a été plus concluante et plus contrôlée car les méthodes utilisées étaient mieux comprises. Cependant, nous n'avons pas vu beaucoup d'améliorations possibles une fois cela mis en place donc nous sommes repartis sur une approche utilisant un réseau neuronal après avoir été conseillés par d'autres groupes.

Nous avons donc pu continuer notre première approche qui a été plus concluante grâce à notamment la correction de la fonction de perte en "binary crossentropy". Cette approche sera détaillée dans la section 2.

# 2 Solution apportée

## 2.1 Approche par seuil variable et moyenne pondérée

Nous avons essayé une approche basée sur un seuil variable pour chacune des classes. Pour cela, on défini une entrée d'indice k, la k-ième valeur des données d'entraînement et donc et $g_k(i)$ est la valeur terrain de la classe i pour l'entrée d'indice k. On considère que les 248 classes prédire donc $i \in [0, 248[$.

On considère alors les valeurs du vecteur d'entrée $v_k(i)$. On prédit les vecteurs $p_k$ à partir des vecteurs $v_k$ chacun pour une entrée d'indice $k$. $p_k(i)$ est défini selon la formule ci-dessous:

$$p_k(i) = 0 \text{ si } v_k(i) < s_{k,i}$$
$$p_k(i) = 1 \text{ sinon}$$

Le seuil $s_{j,i}$ est défini pour $k, j \leq n$ et $i \in [0, 248[$ défini tel que:

$$\begin{aligned} s_{0,i} &= 0.5 \\ s_{k+1,i} &= \frac{s_{k,i} * (n+1) + v_k(i)}{n+2} && \text{si } p_k(i) \neq g_k(i) \\ &= s_{k,i} && \text{sinon} \end{aligned}$$

Pour n entrées, le seuil est initialisé à 0.5 puis est modifié à chaque fois que la prédiction se trompe pour une entrée k et une classe i. La formule est adaptée de la formule de la moyenne. Ainsi la nouvelle valeur du seuil pour cette classe est la moyenne entre la moyenne à l'étape précédente et la valeur entraînant l'erreur.

Ainsi, nous avons utilisé les seuils finaux $sn, i$ pour calculer les vecteurs de prédiction. Ces seuils convergent selon l'ordre des données utilisées donc les résultats divergent selon les exécutions car l'on mélange les données avant l'apprentissage. En utilisant cette méthode, nous avons pu obtenir un score d' approximativement 0.395 au challenge.

### 2.2 Approche par réseau de neurones

Comme le vecteur Y de nos données est binaire, et une fois que nous avions bien compris les fonction de perte, nous avons essayé de créer un modèle de neurones qui utiliserait la binary crossentropy. Nous avons aussi normalisé les valeurs d'entrée à l'aide d'un Standard Scaler afin de faciliter l'entraînement. Ainsi, en utilisant les données d'entraînement avec 20% d'entre elles étant considérées comme un set de validation, nous avons créé un réseau de neurones dense afin d'essayer de rectifier les valeurs d'entrées et de prédire une sortie plus proche de la réalité.

Les 3 couches intermédiaires ont une activation ReLu tandis que la couche de sortie a une activation sigmoid, afin d'obtenir les probabilités pour chacune des 248 classes.

Nous avons fixé le seuil de décision à 0.50 fonctionnant de la même que dans la méthode précédente. Ici, nous arrondissons à 1 ou 0 simplement.

Les hyper-paramètres du réseau ainsi que les courbes de la loss et d'accuracy sont disponibles en figure 1. Étant donné que nous n'appliquons le seuil de décision qu'après l'entraînement, la métrique d'accuracy est très faible (inférieure à 10%), mais cet entraînement nous permet tout de même d'améliorer nos résultats par rapport au premier réseau de neurones utilisant categorical cross entropy comme fonction de perte, à une valeur approximativement de 0.364 au challenge.

Pour ce qui est de la taille des couches intermédiaires, manquant encore d'expérience pour en faire un choix éclairé, nous avons essayé de nombreuses valeurs et avons trouvé les résultats les plus efficaces avec une bonne rapidité avec les tailles suivantes:

- **Couche 1**: 300 neurones
- **Couche 2**: 280 neurones
- **Couche 3**: 260 neurones.

## 3 Conclusion

Ce projet, étant le premier en machine learning pour notre équipe, nous a permis de nous familiariser avec les concepts et d'essayer diverses approches pour répondre à un sujet complexe. Malgré le fait que notre méthode faisant appel à des réseau de neurones soit moins efficace que notre méthode adaptant le seuil pour chaque catégorie, nous avons eu l'occasion d'apprendre à construire et utiliser des modèles de neurones ainsi qu'à réaliser leur fine tuning. Nous avons aussi été confrontés à des situations présentant les problèmes classiques de machine learning, tels que l'overfitting.

Afin d'améliorer notre projet, il nous faudrait expérimenter en opérant à travers les différentes classes des données d'entrées, étant donné qu'on sait par

exemple que seules les 90 premières colonnes représentent les tag-genres. Il faudrait donc apprendre les liens entre chaque classe et les exploiter au sein d'un réseau de données plus complexe.

Nous pourrions aussi réaliser de la data augmentation, afin d'avoir plus de données uniques sur lesquelles s'entraîner, et continuer à faire du fine tuning sur le réseau existant afin de continuer à développer notre compréhension de ce genre de technologies.

# 4 Annexe

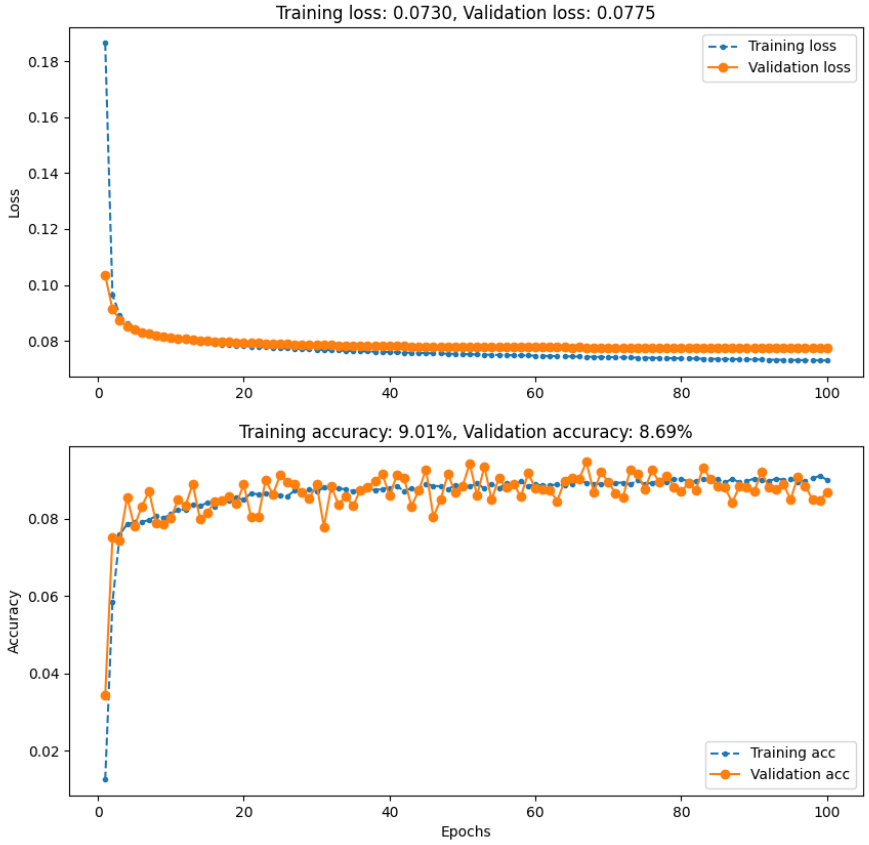


Figure 1: Loss (binary crossentropy= et accuracy de notre réseau de neurones. Hyper-paramètres: learning rate = 5e-5, epochs = 100, split entraînement/validation = 0.20, nombre de données en entrée: 110853