Supervised Learning and Speaker recognition using Neural Networks

Apprentissage par réseaux de neurones artificiels

ANNEN Rayane, MARTINS Alexis

HEIG VD

12.04.2023

Table des matières

Introduction **2**

Hommes et femmes **2**

Modèle . . . 3

Exploration des hyper-paramètres . . . 3

Évaluation des performances . . . 6

Hommes, femmes et enfants **7**

Modèle . . . 8

Exploration des hyper-paramètres . . . 8

Évaluation des performances . . . 10

Voix synthétisées et naturelles **11**

Modèle . . . 11

Exploration des hyper-paramètres . . . 11

Évaluation des performances . . . 14

Conclusion **15**

Introduction

Ce rapport présente le travail effectué pour le PW3. Dans ce travail pratique nous appliquons la méthodologie générale de modélisation data-driven afin de développer un algorithme capable de reconnaître des voix à partir de sons pré-enregistrés.

Les principales caractéristiques de la voix sont le timbre, le ton, le débit et l'intonation. Pour analyser nos données sonores nous avons utilisé les MFCC (*Mel-Frequency Cepstrum Coefficients*) afin d'obtenir les coefficients cepstraux des sons qui sont caractéristiques de la parole.

Finalement afin d'évaluer les performances du réseau nous avons utilisé la validation croisée. Dans le modèle final nous l'évaluons également avec une matrice de confusion ainsi que le F-score.

Hommes et femmes

Dans cette première partie nous devons créer un modèle qui est capable de classifier un son selon le sexe de la personne qui parle.

Nous à notre disposition 72 fichiers sonores .wav contenant précisément 36 sons de voix de femmes et 36 autres de voix d'hommes. Les sons représentent des voyelles en anglais.

Comme précisé dans l'introduction nous utilisons les MFCC pour classifier les sons. Pour chaque enregistrement nous avons 13 valeurs auxquelles nous avons pris la médiane.

Le MFC (*Mel-Frequency Cepstrum*) est la représentation du spectre de puissance à court terme d'un son adaptée à la non linéarité de l'oreille (échelle des mels), les coefficients qu'on calcule à partir de cette représentation sont caractéristiques de la parole.

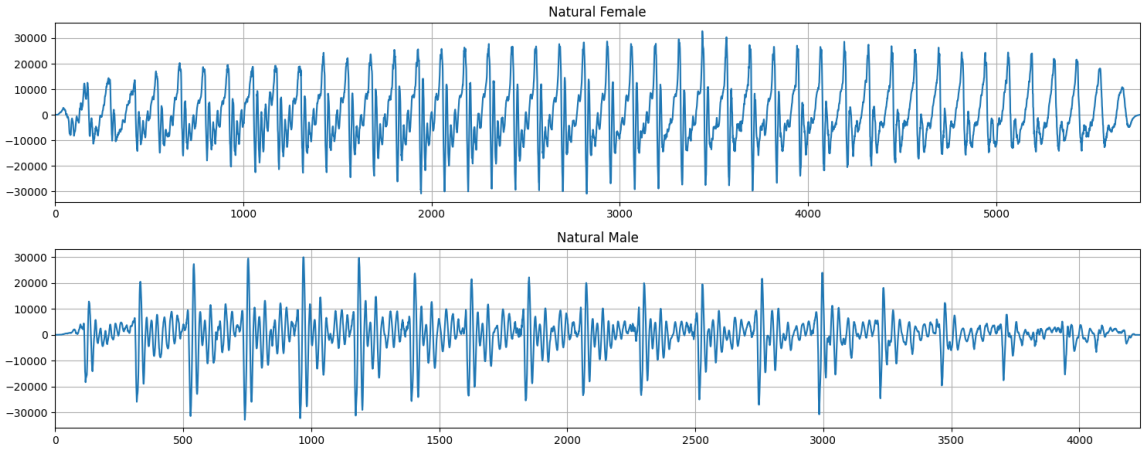

Figure 1: Représentation d'une donnée brute pour les hommes et les femmes

Chaque enregistrement est ensuite séparé en plusieurs fenêtres temporelles, pour chacune d'entre elles nous calculons le MFCC. Nous obtenons alors finalement pour chaque enregistrement 13 coefficients cepstraux.

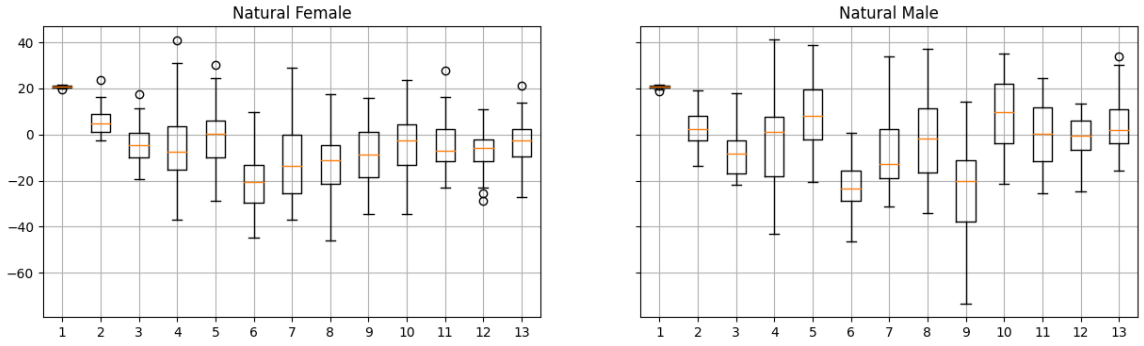

Figure 2: MFCC médian des enregistrements des hommes et des femmes (voix naturelle)

Nous annotons chaque enregistrement de la manière suivante :

- -1 pour une voix de femme
- 1 pour une voix d'homme

Modèle

Dans notre modèle nous utilisons une fonction d'activation $\tanh$, son ensemble image étant $]-1,1[$ nous fixons le seuil de séparation des classes à 0. Nous normalisons également les données dans l'intervalle $[0,1]$.

Exploration des hyper-paramètres

Le premier hyper-paramètre à définir était le nombre d'epochs. Nous avons décidé de tester à chaque fois avec 100, 300 et 600 epochs. A préciser que pour chaque valeur d'epochs testée, le test était fait avec 2, 4, 8, 16, 32 neurones

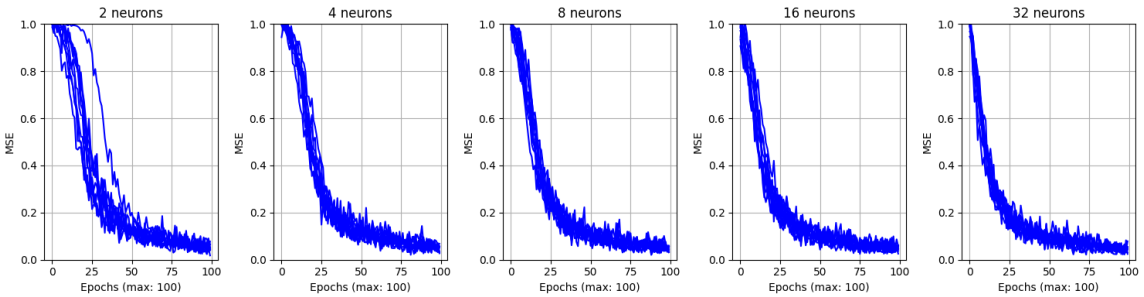

Figure 3: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 100 epochs

On observe que 100 epochs ne sont pas suffisants pour notre modèle. L'erreur est trop instable et hasardeuse.

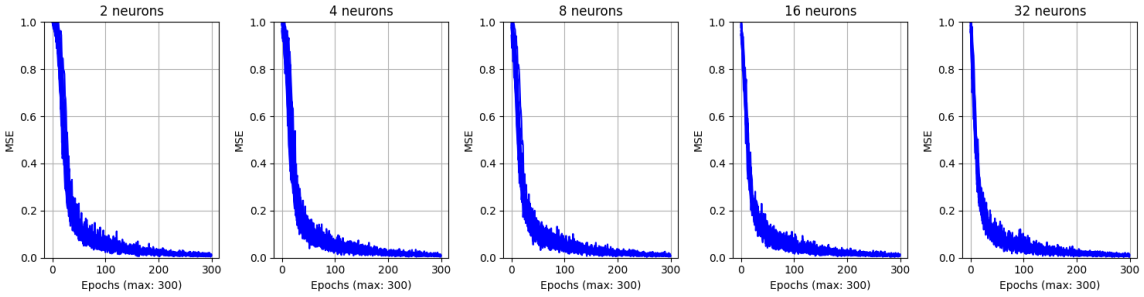

Figure 4: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 300 epochs

On voit que pour deux neurones ce n'est pas suffisant, toutefois pour 4, 8, 16 et 32 neurone cela est largement suffisant.

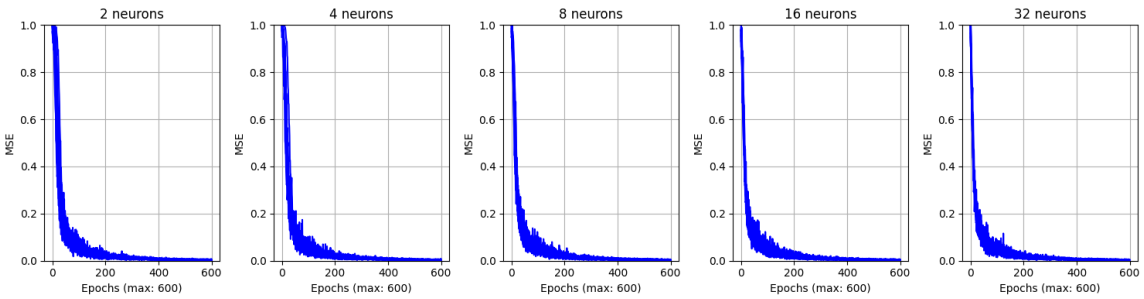

Figure 5: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 600 epochs

Cela est suffisant. Toutefois, nous devrons faire attention à l'overfitting. De plus, les gains par rapport à 300 epochs ne sont pas significatifs. On privilégiera un modèle plus simple avec moins d'epochs.

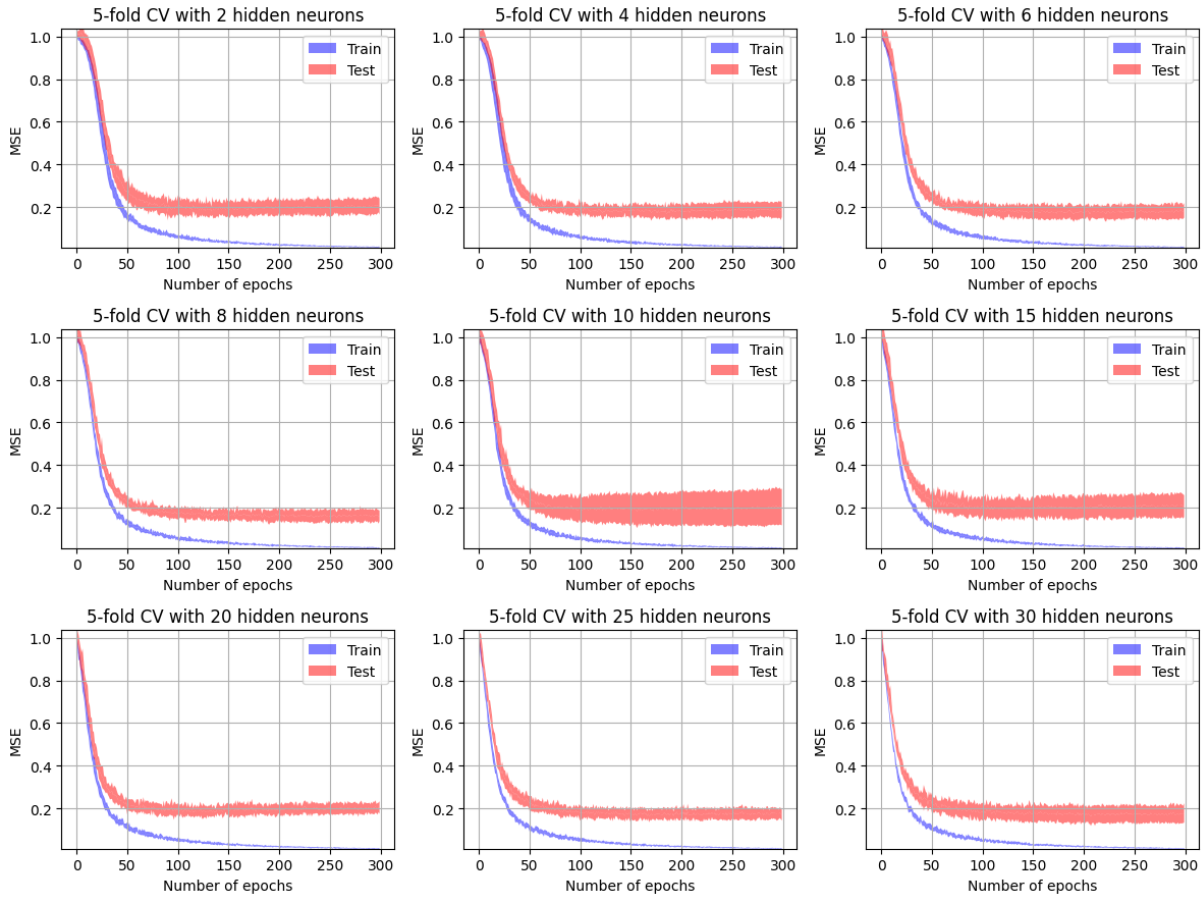

Figure 6: Validation croisée selon le nombre de neurones pour 300 epochs

Pour le choix du nombres de neurones, nous sommes restés avec les valeurs proposées en classe. C'est-à-dire que tous les tests sont effectués avec 2, 4, 6, 8, 10, 15, 20, 25 et 30 neurones. Pour cette étape, nous avons pris 300 epochs étant donné que c'était la meilleure valeur lors du test précédent.

Nous constatons que la plupart du temps le modèle fait de l'overfitting. Nous avons donc choisi un réseau à 8 neurones et 300 epochs, c'est celui qui nous paraît le mieux. L'erreur semble toujours être décroissante.

En résumé nous avons les hyper-paramètres suivants :

- Activation function: $\tanh$
- Nombre d'epochs : 300
- Nombre de neurones : 8
- Learning rate : 0.001
- Momentum 0.6
- Seuil : 0

Afin de déterminer les valeurs des autres hyper-paramètres nous avons utilisé la technique vue en classe qui est de tester des valeurs espacées de manière logarithmique, par exemple pour le learning rate, les valeurs 0.1, 0.01, 0.001.

Évaluation des performances

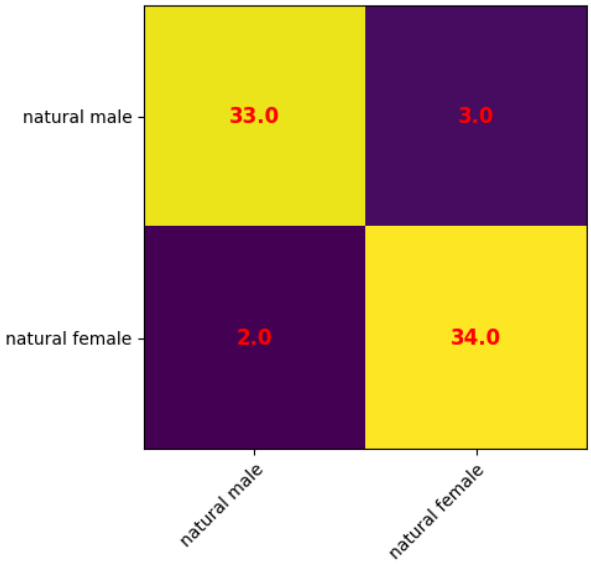

Figure 7: Matrice de confusion du modèle sélectionné

Voici les résultats obtenus pour l'erreur et le F-Score :

- MSE training: 0.0718
- MSE test: 0.202
- F1-Score: 0.929

On peut voir que l'erreur d'entraînement est très bonne. Cependant pour l'erreur de test, celle-ci est un peu haute. On a pu remarquer que le modèle commençait a légèrement overfit et cela pourrait donc expliquer ce résultat. Finalement, on remarque que le F1-Score est tout de même assez bon.

Hommes, femmes et enfants

Nous utilisons les données fournies dans la partie précédente ainsi que des nouveaux enregistrements sonores venants d'enfants. Nous avons à disposition 108 enregistrements sonores pour les enfants, nous avons trois fois plus d'enregistrements que pour les femmes ou les hommes car les enfants sont initialement réparties en trois sous-classes : ceux ayant 3, 5 ou 7 ans. Pour notre modèle nous ne ferons pas la distinction.

Dans un but d'équilibrage des classes nous ne sélectionnons que 36 de ces enregistrements.

En résumé, nous avons 36 enregistrements pour les enfants, 36 pour les femmes et 36 pour les hommes. Pour classifier ces enregistrements nous utilisons un codage one-hot :

- Homme : 100
- Femme : 010
- Enfant : 001

Pour évaluer les performances du modèle en utilisant une matrice de confusion nous avons dû effectuer des modifications dans l'algorithme fourni pour la calculer. Initialement les valeurs de sorties pouvaient être classées dans plusieurs catégories, voir pas du tout, e.g. 011. Nous avons modifié l'algorithme afin que nous prenions la valeur la plus haute à chaque fois en utilisant la méthode `np.argmax`.

Ci-dessous voici se trouve le code modifié :

```
1  def compute_confusion_matrix(target, output, threshold):
2      """
3      This function computes the confusion matrix for a given set of
           predictions.
4      Rows are the actual class and columns are the predicted class
5      """
6      assert len(target.shape) == 2, "target must be a 2-dimensional
           array"
7      assert len(output.shape) == 2, "output must be a 2-dimensional
           array"
8
9      if target.shape[1] == 1:
10         n_classes = 2
11         target_binary = np.concatenate((target > threshold, target <=
               threshold), axis=1)
12     else:
13         n_classes = target.shape[1]
14         target_binary = target > threshold
15
16     if output.shape[1] == 1:
17         output_binary = np.concatenate((output > threshold, output <=
               threshold), axis=1)
18     else:
```

```
        output_maxs = [np.argmax(tup) for tup in output]
        tuples = [[False] * n_classes for i in output_maxs]
        for i, t in enumerate(tuples): t[output_maxs[i]] = True
        output_binary = np.array(tuples)

    confusion_matrix = np.zeros((n_classes, n_classes))
    for t in np.arange(n_classes):
        for o in np.arange(n_classes):
            confusion_matrix[t,o] = np.sum(np.logical_and(target_binary
                [:,t], output_binary[:,o]))

    return confusion_matrix
```

Modèle

Nous avons choisi la fonction d'activation $\tanh$ avec un seuil à 0.5. Les données sont également normalisées.

Exploration des hyper-paramètres

Comme dans la première partie : on test pour 100, 200, 300 epochs et 2, 4, 8, 16, 32 neurones.

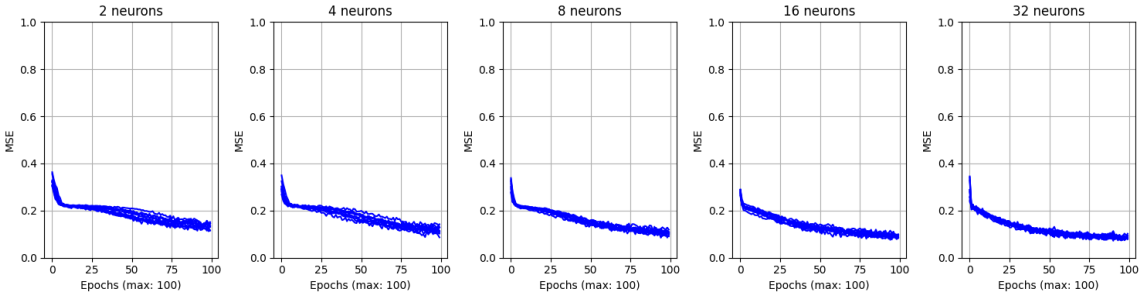

Figure 8: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 100 epochs

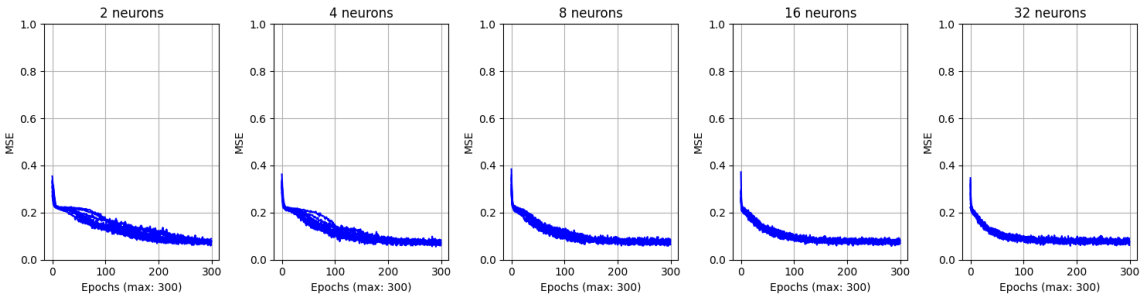

Figure 9: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 300 epochs

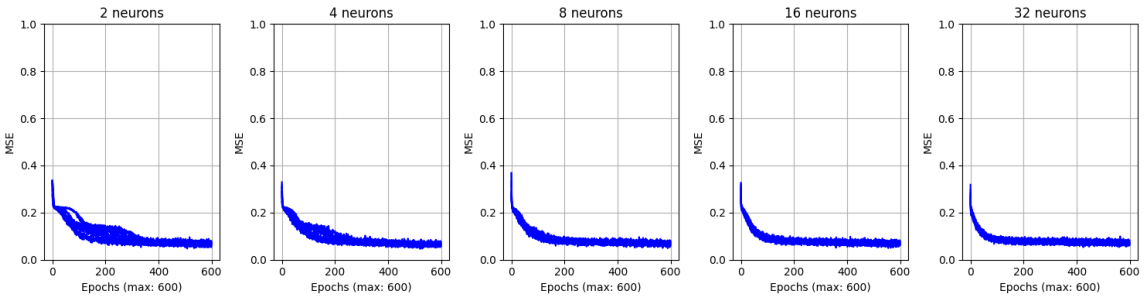

Figure 10: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 600 epochs

On voit que les premières observations à 100 epochs sont encore une fois un peu brouillon. Et les deux autres observations à 300 et 600 epochs sont assez proches. Nous avons donc décidé de sélectionner 300 epochs étant donné qu'il n'y avait pas forcément d'améliorations. De plus on pourrait tendre sur de l'overfitting en voulant pousser le nombre d'epochs.

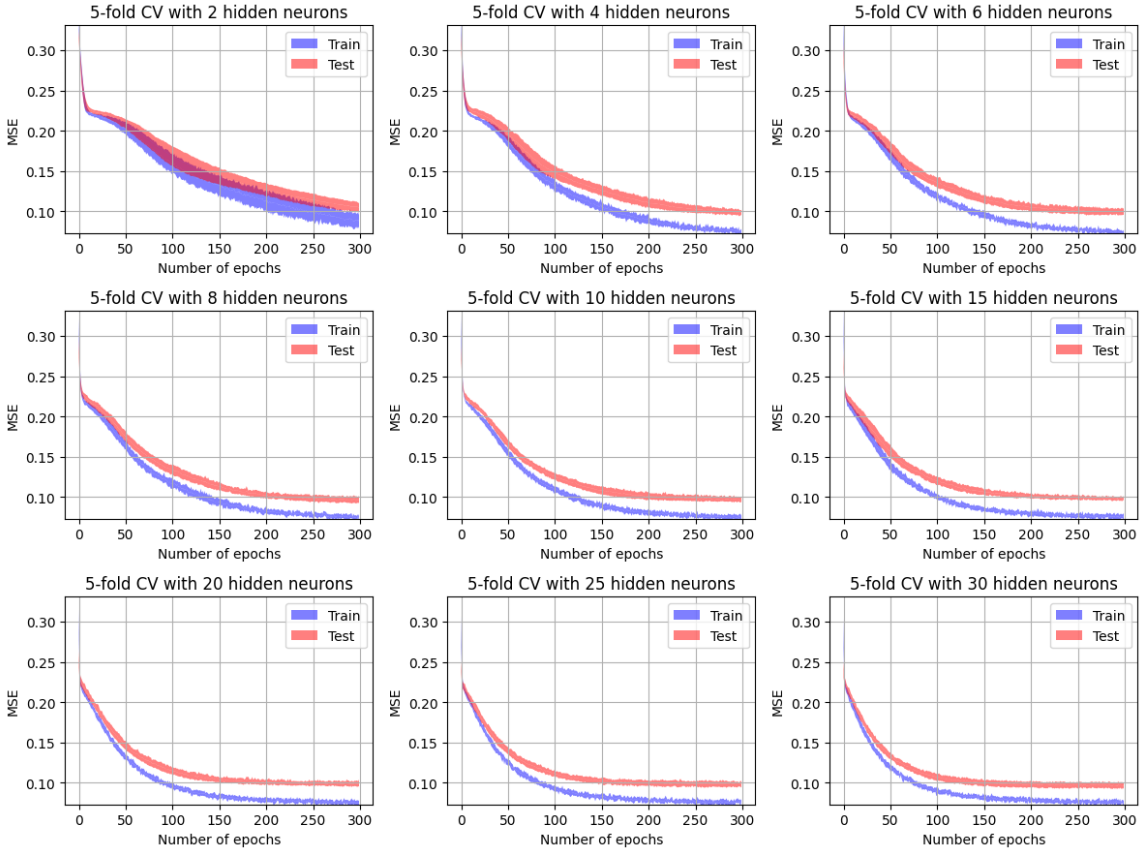

Figure 11: Validation croisée selon le nombre de neurones pour 300 epochs

Comme pour le modèle précédent on va privilégier un modèle plus simple qui ne nécessite pas énor-

mément de neurones. Nous avons choisi de partir sur le modèle à 8 neurones à nouveau, en effet, l'erreur n'est pas significativement différentes pour des modèles avec plus de neurones. On constate moins d'overfitting également de manière générale.

Hyper-paramètres choisis pour le modèle final :

- Activation function: $\tanh$
- Nombre d'epochs : 300
- Nombre de neurones : 8
- Learning rate : 0.001
- Momentum 0.7
- Seuil : 0.5

Évaluation des performances

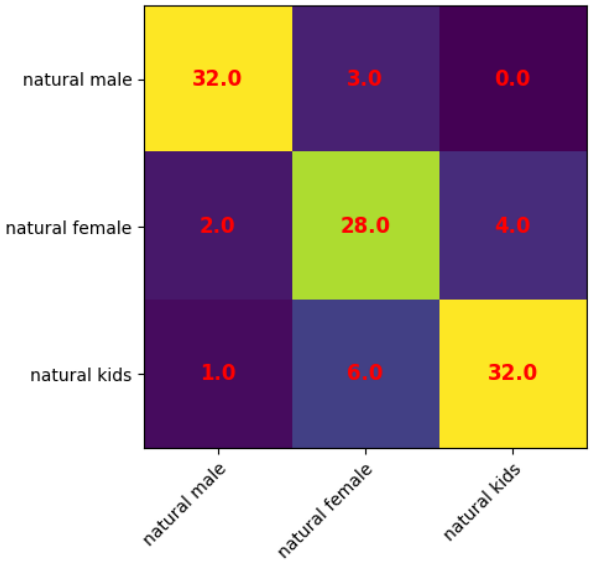

Figure 12: Matrice de confusion du modèle sélectionné

Résultat obtenus :

- MSE training: 0.0729
- MSE test: 0.102
- F1-Score pour chaque classe (homme-femme-enfants) : 0.914, 0.789, 0.853
- F1-Score (pondéré): 0.853

Pour ce modèle, on remarque qu'autant les résultats d'entraînement comme ceux de tests sont très bons. Le F1-Score pondéré est aussi suffisamment bon pour qu'on garde ce modèle.

Voix synthétisées et naturelles

Pour ce modèle nous classons nos voix selon des critères différents que les deux précédents. Nous classons les voix selon si elles sont synthétisées ou bien si elles sont naturelles. Nous avons donc un problème de classification à deux classes.

Nous avons encodés les valeurs de la manière suivante :

- Voix naturelle : 1
- Voix synthétisée : -1

Nous avons à dispositions pour chaque classe 180 enregistrements. Nous utilisons toujours les MFCC pour caractériser les sons.

Modèle

Nous avons choisi la fonction d'activation $\tanh$ avec un seuil à 0. L'ensemble des données a été normalisé entre $[0, 1]$.

Exploration des hyper-paramètres

Comme dans la première partie : on test pour 100, 200, 300 epochs et 2, 4, 8, 16, 32 neurones.

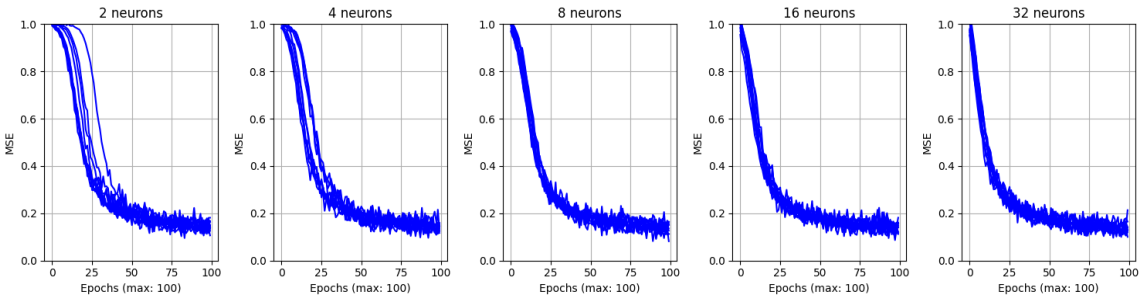

Figure 13: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 100 epochs

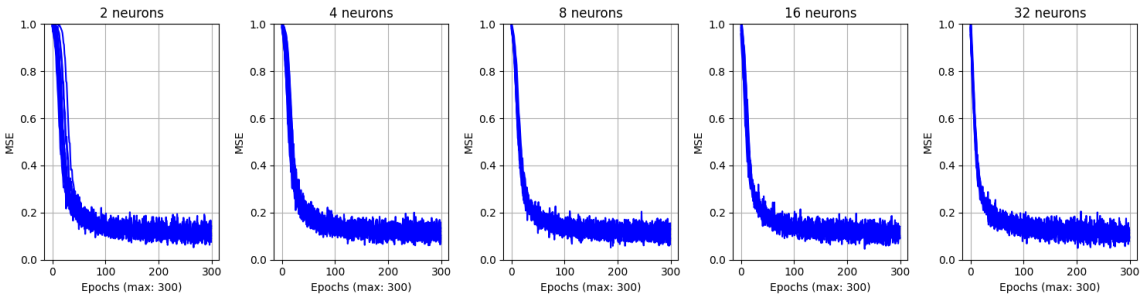

Figure 14: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 300 epochs

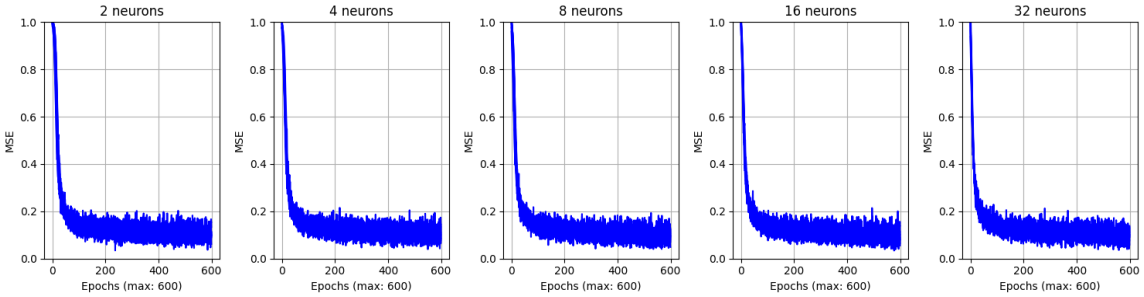

Figure 15: Erreur quadratique moyenne des réseaux selon le nombre de neurones pour 600 epochs

On remarque que pour 100 epochs ce n'est pas trop mauvais, mais encore trop variable. 300 epochs c'est correct, ce n'est pas très large et l'erreur n'est pas spécialement haute. Pour 600 epochs, la bande est vraiment trop large par rapport à la précédente. C'est pour ça que l'on a décidé de prendre les 300 epochs.

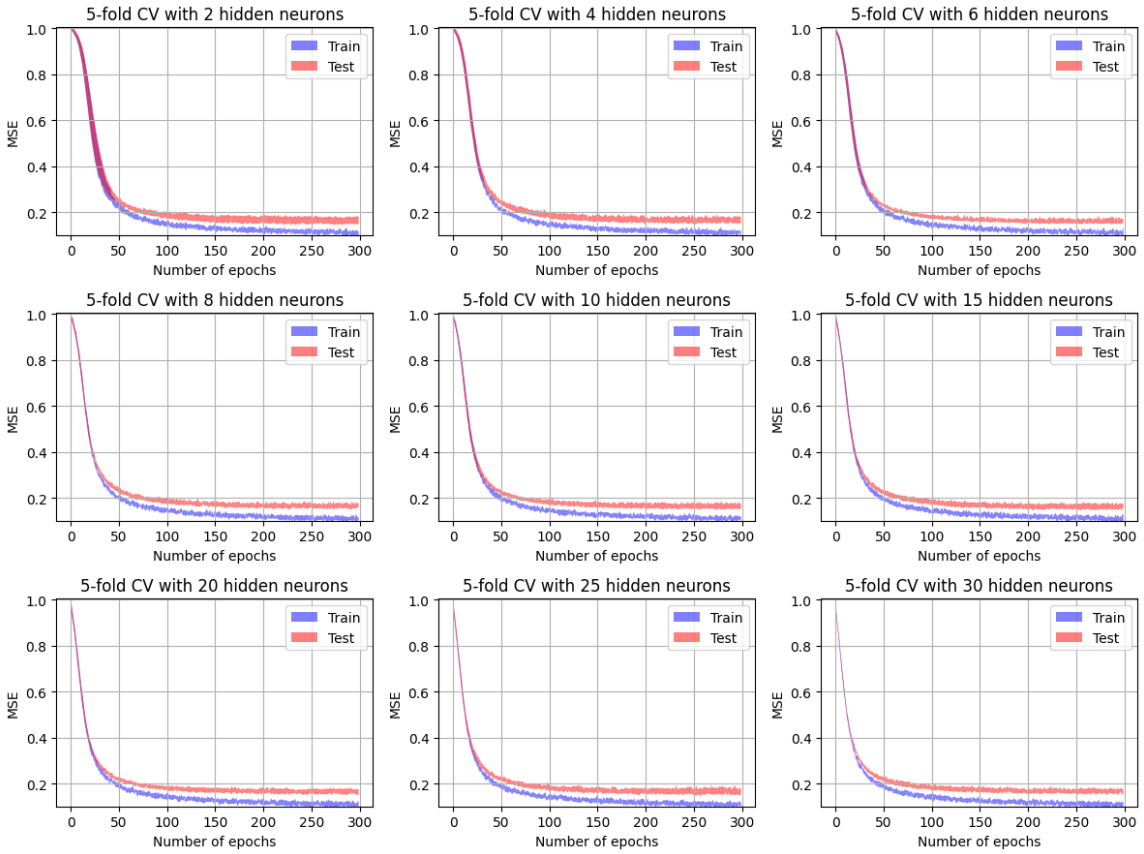

Figure 16: Validation croisée selon le nombre de neurones pour 300 epochs

On a hésité entre prendre un modèle basé sur 6 ou 8 neurones. Finalement, on a choisi celui avec 8 neurones, car celui avec 6 semblait commencer à légèrement overfit. Nous n'avons pas pris en compte les modèles avec plus de neurones, car on essaye de rester le plus simple possible. De plus, les modèles complexes semblaient commencer à overfit.

Hyper-paramètres choisis pour le modèle final :

- Activation function: $\tanh$
- Nombre d'epochs : 300
- Nombre de neurones : 8
- Learning rate : 0.001
- Momentum 0.6
- Seuil : 0

Évaluation des performances

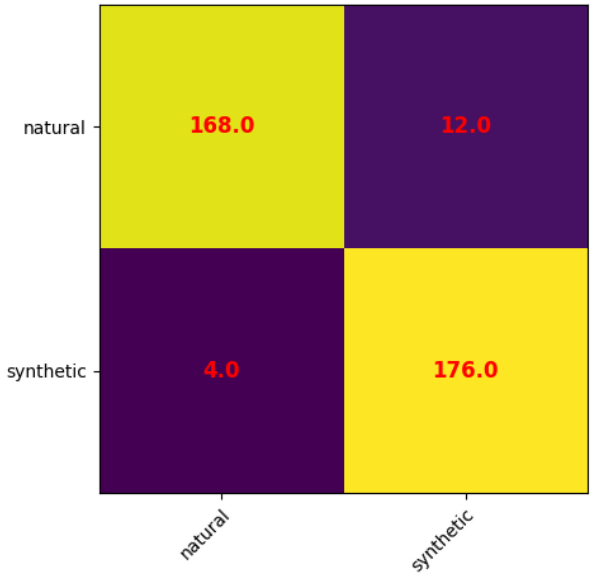

Figure 17: Matrice de confusion du modèle sélectionné

Résultat obtenus :

- MSE training: 0.111
- MSE test: 0.154
- F1-Score: 0.954

Finalement, on remarque que les deux erreurs sont suffisamment bonnes. De plus le F1-score est vraiment excellent.

Conclusion

En conclusion, nos modèles sont dans l'ensemble plutôt corrects pour une première fois avec un MLP. On pense tout de même que les modèles peuvent un peu souffrir du manque de données ou d'overfitting. Après dans le cadre d'un petit projet comme celui-ci, les résultats obtenus sont très bons et suffisants.