Classification par apprentissage profond

Hanna Bekkare
Ines Besbes
Mathilde Ferreira
Sara Rool

Sujet : Reconnaissance d'une langue parlée

Lien Github vers BD

I. Sujet : classification des langues

Sur les nouveaux smartphones, des fonctionnalités de traduction instantanée commencent à émerger. Nous avons été inspirées par cette technologie pour essayer de mettre en place un réseau pouvant reconnaître le pays d'origine d'une personne parlant dans un extrait de 5 secondes.

Nous avons décidé de choisir des pays qui ont des langues plus ou moins similaires :

- Tunisie/ Maroc
- France/ Québec
- Royaume-Uni/ Etats-Unis
- Italie/ Espagne/ Portugal
- Chine/ Japon/ Corée du Sud

Nous souhaitons tester l'hypothèse suivante: les langues provenant de la même racine sont plus difficiles à différencier.

II. Création et préparation de la base de donnée

A. Acquisition et annotation des données

Nous avons opté pour deux méthodes d'acquisition des données (dans la mesure du possible) :

1. Nous avons utilisé la bibliothèque Python "moviepy" pour extraire 5 extraits de 5 secondes d'une vidéo Youtube (discours politique, Ted talk, podcast, etc)

2. Nous nous sommes enregistrées pour avoir des audios de 5 secondes, et nous avons demandé à nos proches de le faire également.

Nous avons donc un script Python prenant en entrée des urls de vidéos Youtube, et qui enregistre et nomme les audios en fonction du pays et du sexe du locuteur. L'annotation et la création de label a été automatisée en reprenant le nom des fichiers audio. Les labels sont associés comme suit:

0 : portuguese	6 : american
1 : chinese	7 : english
2 : french	8 : korean
3 : quebecois	9 : spanish
4 : maroccan	10 : italian
5 : japanese	11 : tunisian

Figure 1 : *Labels des différents audios utilisés*

B. Pré-traitement des données audio

Pour préparer les données à l'entraînement, nous avons utilisé la librairie *Librosa* qui permet de manipuler, analyser, visualiser et extraire des caractéristiques des fichiers audio. En particulier, nous avons utilisé la fonction `librosa.load()` pour charger nos fichiers audio dans un format numérique MP3 et les convertir en tableaux numpy. Nous avons utilisé une fréquence d'échantillonnage de 22kHz. Cette valeur est usuelle pour échantillonner les voix et la musique.

Nous nous sommes aperçues que tous nos extraits n'étaient pas de même longueur. Nous avons décidé de conserver la taille minimale et de tronquer tous les autres extraits.

Une fois nos données mises à la même échelle, nous les normalisons puis nous appliquons une STFT pour passer d'un signal audio à un spectrogramme. Nous avons en sortie des images de taille (512, 164).

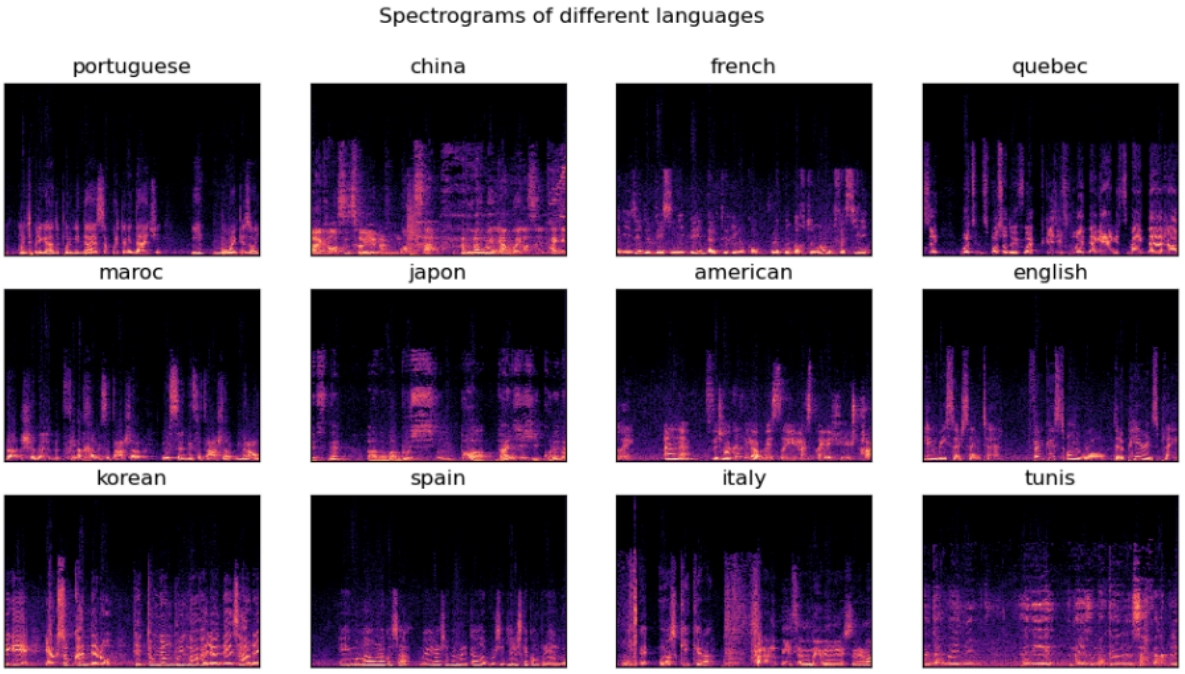

Figure 2 : *Exemple de spectrogramme pour chaque langue*

C. Partitionnement des données en ensembles d'entraînement, de validation et de test

Pour partitionner les données, nous avons décidé de sélectionner les 10 derniers (5 hommes, 5 femmes) enregistrements audio par langue pour constituer les ensembles de test et de validation. Ce qui nous donnera un total de 240 audios pour ces ensembles. Parallèlement, nous allouons les 960 audios restants à l'ensemble d'entraînement. Afin d'assurer une évaluation fiable du modèle, nous veillerons à ce qu'aucun locuteur présent dans l'ensemble d'entraînement ne soit inclus dans les ensembles de test et de validation.

III. Script de chargement de vos données

Afin de pouvoir passer nos spectres en entrée d'un réseau, nous créons une classe `AudioDataset()` qui nous permet de préparer les données en fonction du mode (train, validation, ou test). Elle peut appliquer des transformations optionnelles telles qu'une rotation aléatoire du spectre. La classe permet de créer des tenseurs prêts à être utilisés pour l'entraînement ou la validation avec leurs labels.

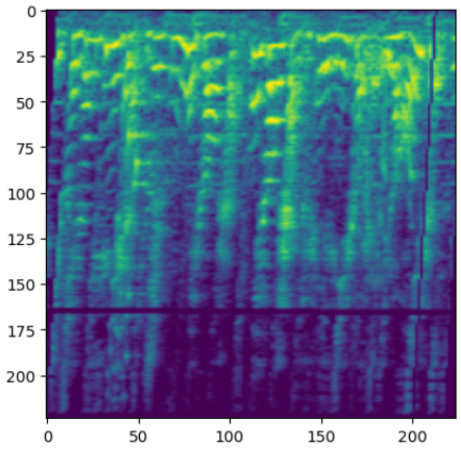

Figure 3 : *Exemple de spectrogramme augmenté*

Ensuite, nous définissons des `DataLoader` qui permettent de charger les données en lots, de manière efficace et parallélisée. Nous créons un dataloader pour chaque ensemble:

- `train_dataloader` est optimisé pour l'entraînement, il mélange les données avec des tailles de lots cohérentes ;
- `val_dataloader` permet une évaluation intermédiaire cohérente des performances pendant l'entraînement ;
- `test_dataloader` est utilisé pour évaluer les performances finales après l'entraînement.

IV. Notre pronostic sur la résolution du problème

La résolution du problème que nous souhaitons résoudre paraît complexe. Tout d'abord car de nombreux paramètres qui influent sur le spectre ne dépendent pas de la langue parlée. Par exemple, le timbre de la voix, l'intonation, les bruits de fond, la qualité de l'enregistrement. Nous ne savons pas si le réseau arrivera à différencier les intonations et les accents entre les différentes langues alors que même un humain peut avoir du mal à réaliser cette tâche. Enfin, comme le scrapping était chronophage, nous n'avons probablement pas un ensemble d'entraînement suffisamment grand et représentatif pour obtenir une bonne classification.

V. Entraînements du modèle

Nous avons effectué plusieurs apprentissages en commençant par l'architecture la plus simple et avons progressivement augmenté la complexité du réseau.

A. LeNet

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 6, 224, 224]             456
            Conv2d-2         [-1, 16, 108, 108]           2,416
            Linear-3                  [-1, 120]       5,598,840
            Linear-4                   [-1, 84]          10,164
            Linear-5                   [-1, 12]           1,020
================================================================
Total params: 5,612,896
Trainable params: 5,612,896
Non-trainable params: 0
```

Figure 4 : *Architecture du modèle LeNet*

Nous avons testé le modèle LeNet en premier, mais il s'est avéré très limité dans notre contexte. La précision sur l'ensemble de test est restée faible, atteignant seulement 8 %, alors qu'elle atteint presque 24% sur l'ensemble d'entraînement, ce qui témoigne que le modèle surapprend.
En effet, LeNet, étant initialement conçu pour des tâches simples comme la reconnaissance de chiffres manuscrits, n'est pas adapté à une tâche aussi complexe que la classification de langues basée sur des spectrogrammes.

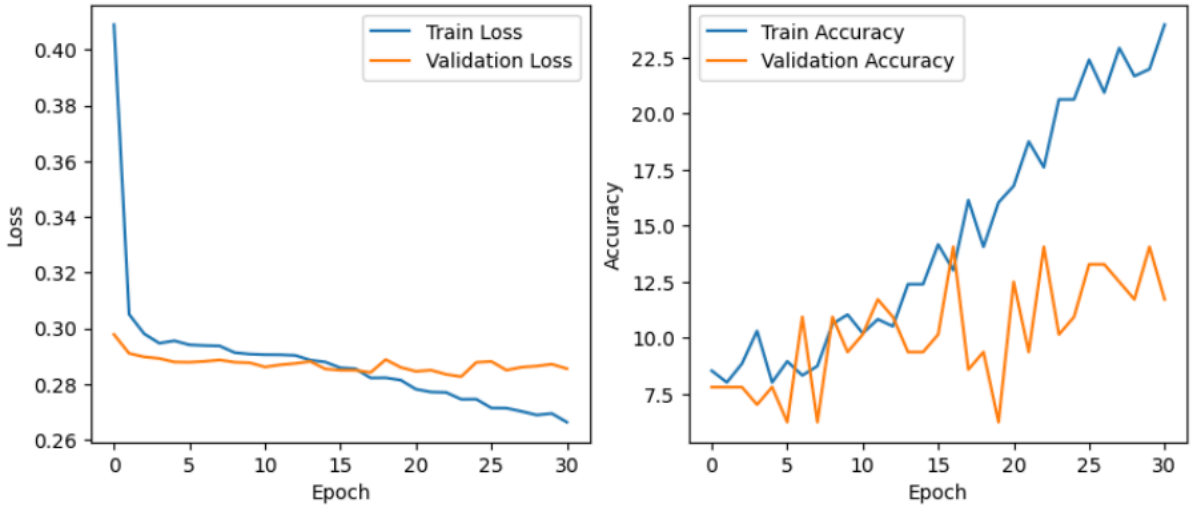

Figure 5 : *Loss et accuracy en fonction du nombre d'époques pour LeNet*

B. ResNet18

Le modèle ResNet18 a montré des performances intéressantes, notamment lorsqu'il est combiné à des paramètres ajustés :

- Taux d'apprentissage : 10^{-3}
- Décroissance de poids : 10^{-4}
- Dropout : 0,2 et 0,5 avec rotation aléatoire des spectrogrammes.

La précision sur l'ensemble de validation a atteint 24 %, ce qui est supérieur au modèle LeNet mais reste loin d'être satisfaisante. Cela suggère que ResNet18 commence à capturer certaines

caractéristiques des langues, mais qu'il est limité par la taille de notre ensemble de données (960 spectrogrammes pour l'entraînement et 240 pour la validation et le test).

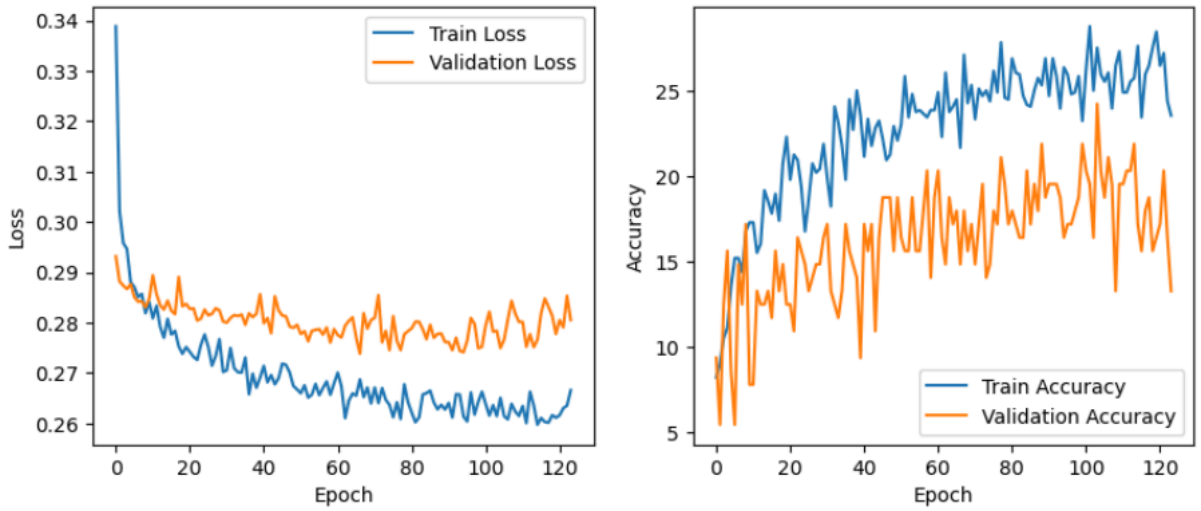

Figure 6 : *Loss et accuracy en fonction du nombre d'époques pour Resnet18*

- **Résultats sur l'ensemble d'entraînement :**

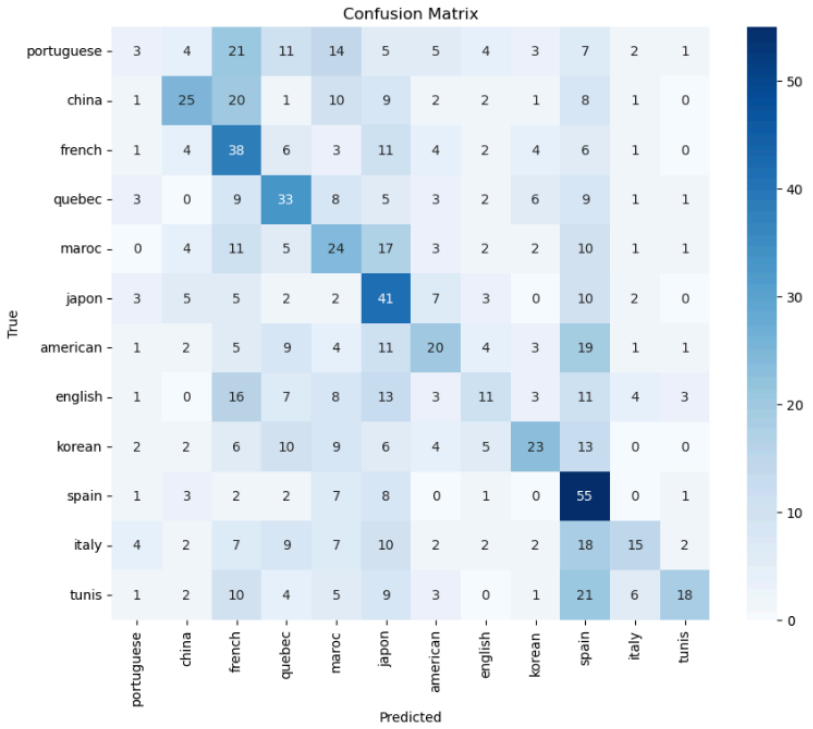

Figure 7 : *Matrice de confusion de l'ensemble de training pour ResNet18*

Lors de l'entraînement, les langues les mieux prédites par le modèle sont l'espagnol, le japonais et le français, tandis que les moins bien prédites sont le portugais, l'anglais et l'italien. De fait, on observe que le modèle parvient à distinguer des langues très similaires. Par exemple, il n'a confondu l'anglais et l'américain qu'à 11 reprises. En revanche, il a confondu le français et le chinois à 20 reprises,

malgré l'absence de similarités apparentes entre ces deux langues. Ainsi, le modèle pourrait être optimisé pour améliorer la reconnaissance des caractéristiques spécifiques à chaque langue.

Pour mieux analyser les performances du modèle, on a regroupé les pays partageant une langue commune dans une matrice de confusion fusionnée. Par exemple, la Tunisie et le Maroc, où l'arabe est parlé, ont été regroupés ensemble.

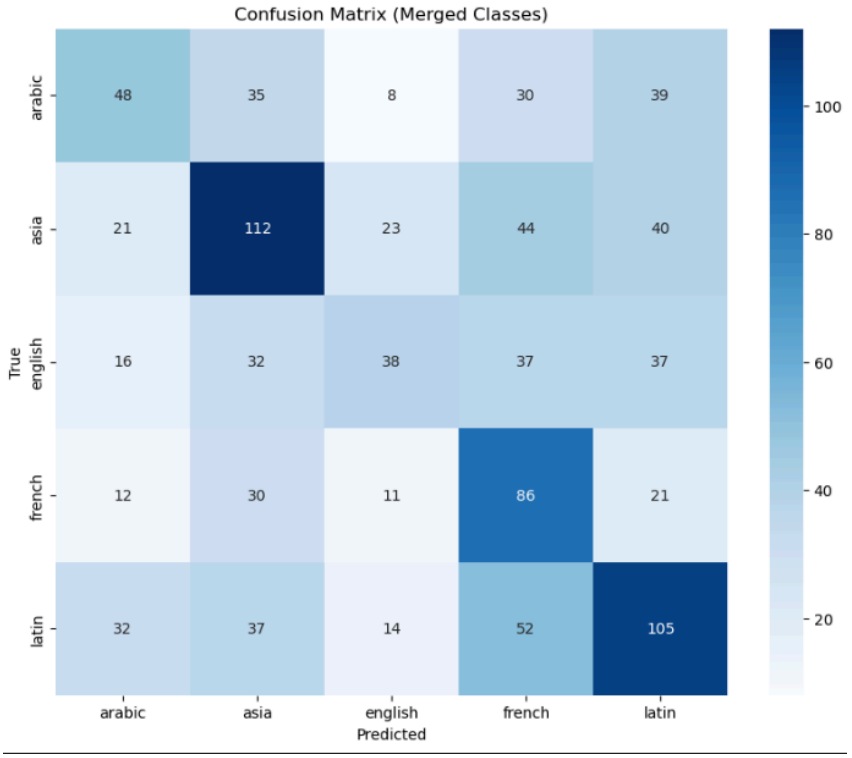

Figure 8 : *Matrice fusionnée de confusion de l'ensemble de training pour ResNet18*

On observe avec ResNet-18 sur l'ensemble d'entraînement que les langues asiatiques, latines et les variantes de la langue française figurent parmi les mieux prédites. Cela pourrait s'expliquer par le fait que ces groupes de langues présentent des caractéristiques linguistiques distinctes ou des structures spécifiques qui sont bien captées par les capacités d'extraction de traits de ResNet-18.

- **Résultats sur les ensembles de test/validation :**

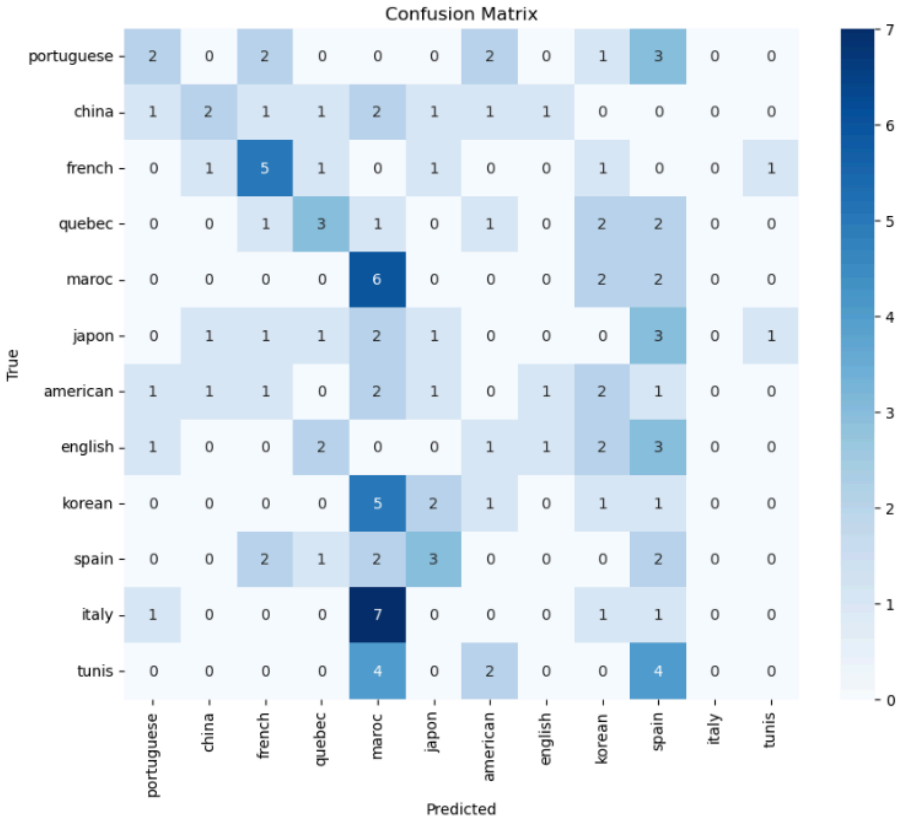

Figure 9 : *Matrice de confusion de l'ensemble de test pour ResNet18*

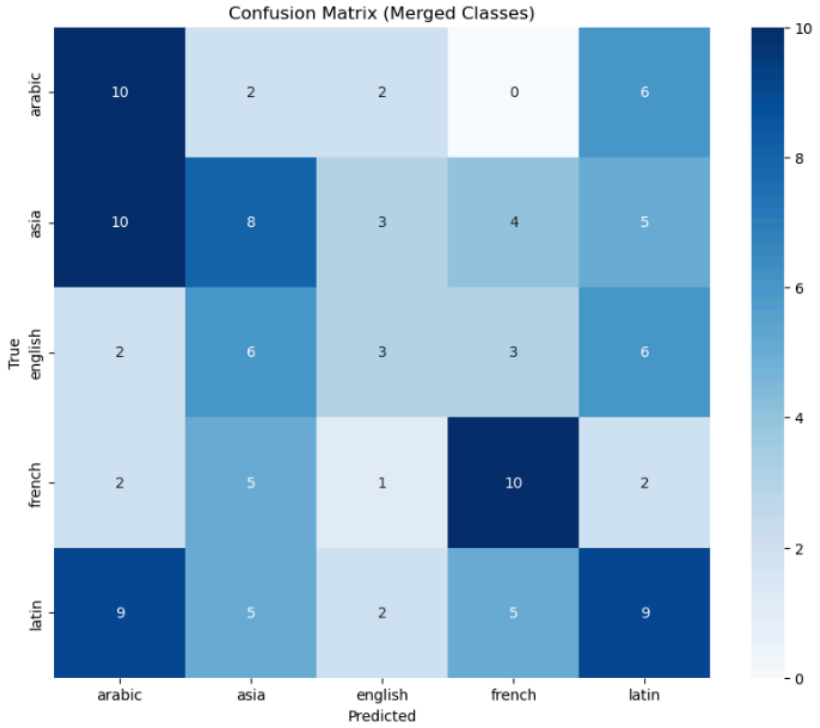

Figure 10 : *Matrice fusionnée de confusion de l'ensemble de validation et de test pour ResNet18*

Sur les ensembles de test et de validation, les performances du modèle sont inférieures à celles observées sur l'ensemble d'entraînement. Contrairement aux observations précédentes, les langues les mieux prédites sont l'arabe et le français. Cependant, ces résultats doivent être interprétés avec prudence. Par exemple, bien que l'arabe apparaît comme la langue la mieux prédite, le tunisien n'a jamais été correctement identifié (valeur de 0 dans la matrice de confusion).

L'insuffisance de données d'entraînement pourrait expliquer les résultats obtenus sur les ensembles de test et de validation.

On va par la suite essayer d'améliorer nos résultats en utilisant du transfer learning en partant avec un modèle ResNet18 pré-entrainé également sur des spectrogrammes de voix (Lien Github du réseau : https://github.com/elaaatif/spectrogram-voice-analysis-with-ResNet-152?tab=readme-ov-file).

C. ResNet18 pré-entraîné

- **Modèle**

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 64, 56, 56]          36,864
      BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 64, 56, 56]          36,864
      BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
       BasicBlock-18           [-1, 64, 56, 56]               0
           Conv2d-19          [-1, 128, 28, 28]          73,728
      BatchNorm2d-20          [-1, 128, 28, 28]             256
             ReLU-21          [-1, 128, 28, 28]               0
           Conv2d-22          [-1, 128, 28, 28]         147,456
...
Params size (MB): 43.66
Estimated Total Size (MB): 107.04
```

Figure 11 : *Architecture du modèle ResNet18 pré-entrainé*

Avec un modèle pré-entraîné, nous avons utilisé un taux d'apprentissage plus faible 10^{-5} pour ajuster le réseau à notre tâche spécifique. On remarque que les résultats sur l'entraînement et la précision sur le test restent encore limités, indiquant que les données ne suffisent pas à tirer pleinement parti du pré-entraînement.

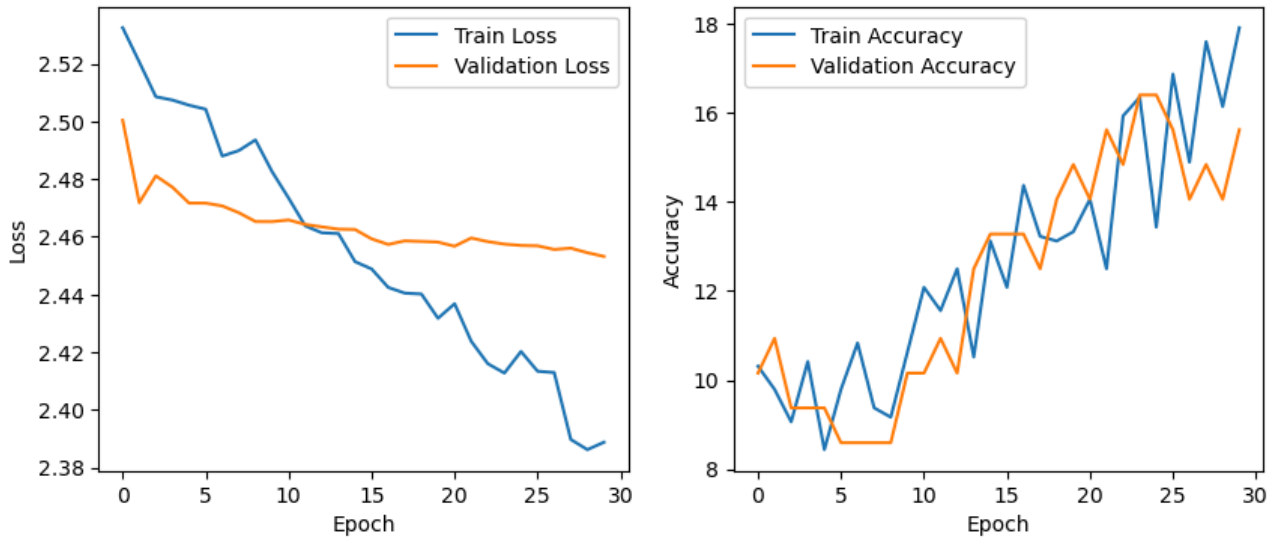

Figure 12 : *Loss et Accuracy en fonction du nombre d'époques pour ResNet18 pré-entrainé*

- **Résultats sur l'ensemble d'entraînement :**

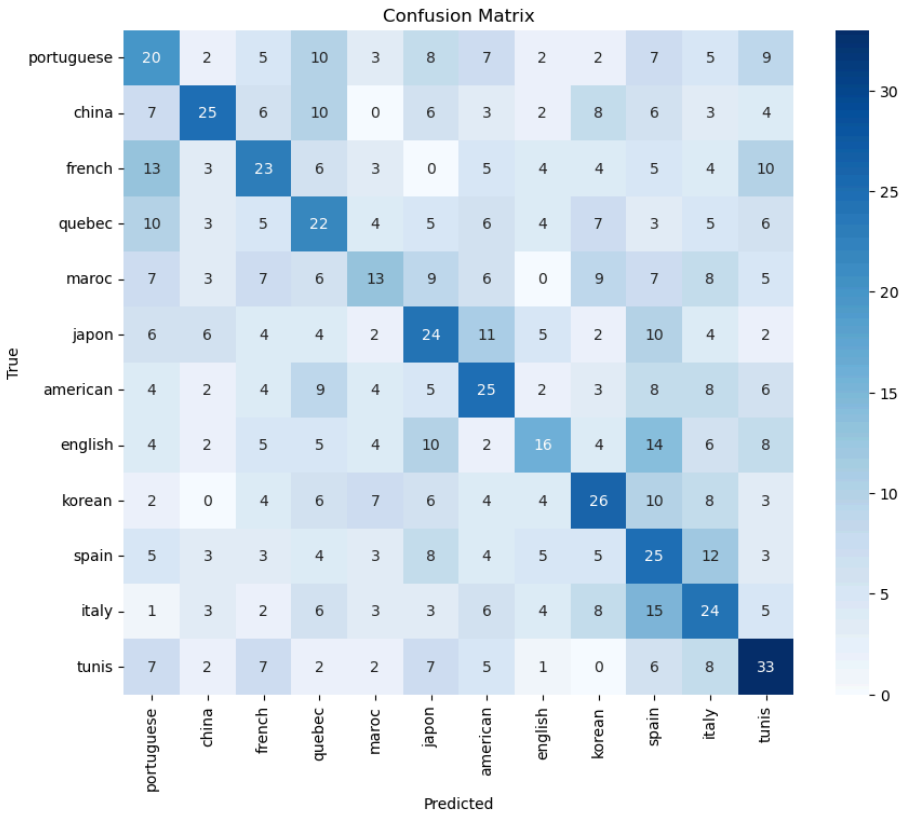

Figure 13 : *Matrice de confusion de l'ensemble de training pour ResNet18 pré-entraîné*

Avec le modèle ResNet-18 pré-entraîné, les langues les mieux prédites sont le tunisien et le coréen. Dans ce cas, la matrice de confusion montre que toutes les langues présentent des coefficients relativement similaires, autour de 25. Ces résultats suggèrent que le modèle atteint un niveau de performance globalement homogène entre les différentes langues, bien qu'il se distingue légèrement sur le tunisien et le coréen.

- **Résultats sur l'ensemble de test /validation :**

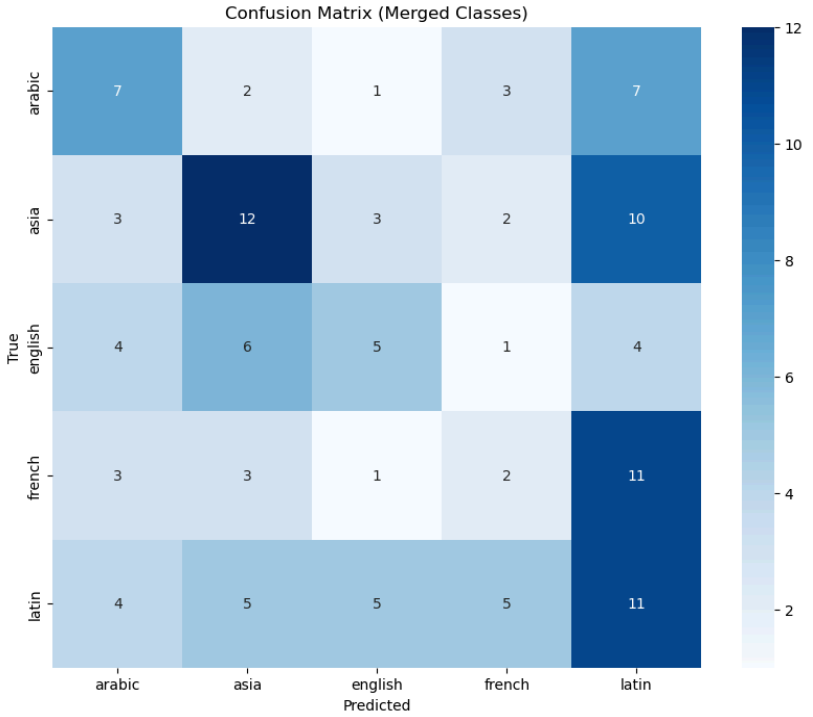

Figure 14 : *Matrice fusionnée de confusion de l'ensemble de test pour ResNet18 pré-entraîné*

On remarque, comme pour ResNet18, que les langues les mieux prédites sont les langues asiatiques et latines.

Ici,

D. VGG16 et VGG16 pré-entraîné

Les modèles VGG16 et VGG16 pré-entraîné ont également été testés. Avec des paramètres optimaux :

- taux d'apprentissage 10^{-6}
- décroissance de poids : 10^{-4}
- dropout : 0,4.

Ces modèles ont montré une meilleure capacité de généralisation que ResNet18, mais leurs performances sur le test restent insuffisantes pour une classification robuste.

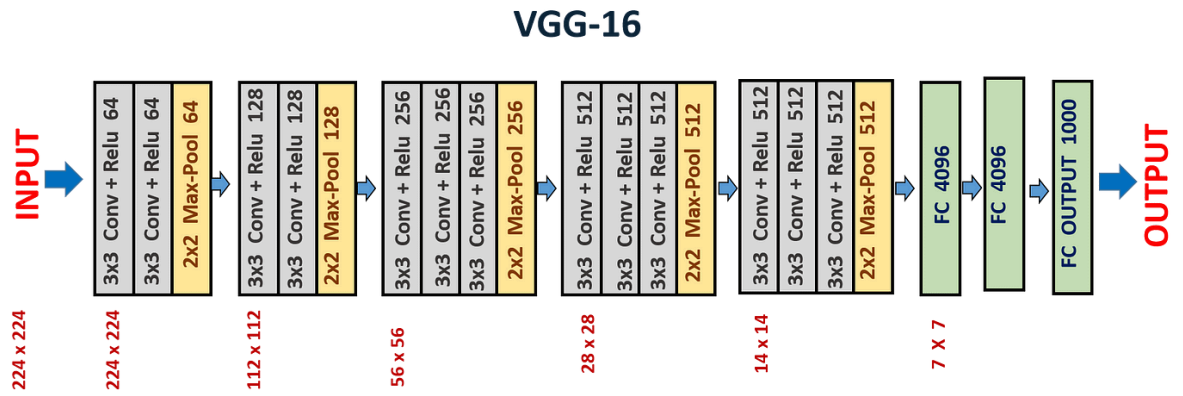

Figure 15 : *Architecture du modèle VGG16*

1. VGG16 non pré-entrainé

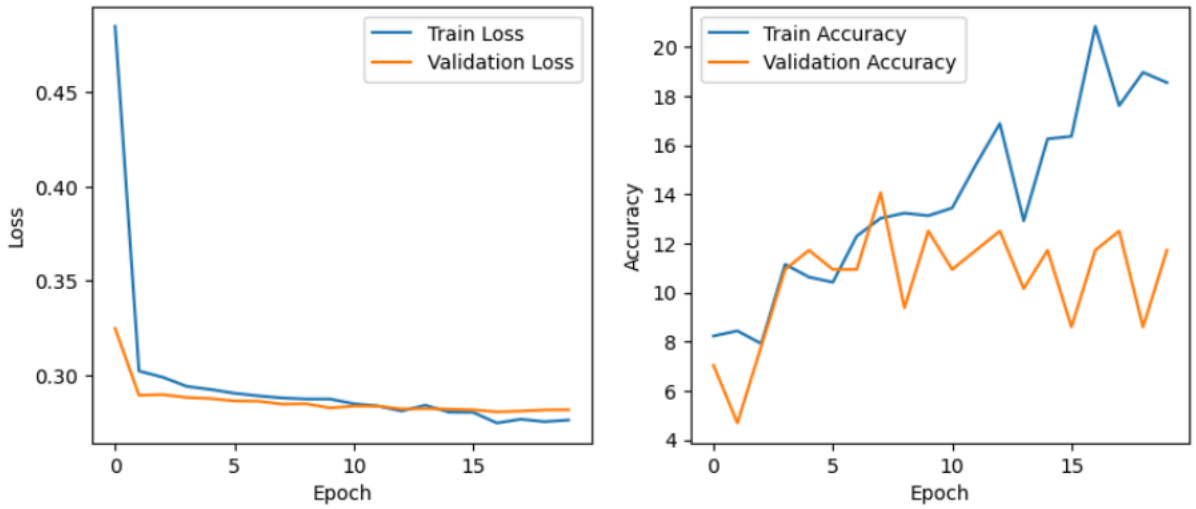

Figure 16 : *Loss et Accuracy en fonction du nombre d'époques*

Ici, le modèle VGG16 non pré-entrainé montre des signes clairs de sur-apprentissage. En effet, la précision sur l'ensemble d'entraînement (20%) est significativement plus élevée que celle sur l'ensemble de test (12%). Cet écart indique que le modèle s'est trop spécialisé sur les données d'entraînement, au détriment de sa capacité à généraliser sur de nouvelles données.

Cela peut s'expliquer par le fait que VGG16 soit un modèle très complexe (138 millions de paramètres). De fait, le jeu de données d'entraînement est probablement trop petit par rapport à la complexité du modèle.

Ainsi, une solution qu'on a choisie est d'utiliser du transfer learning en partant d'un modèle VGG16 pré-entraîné.

2. VGG16 pré-entrainé

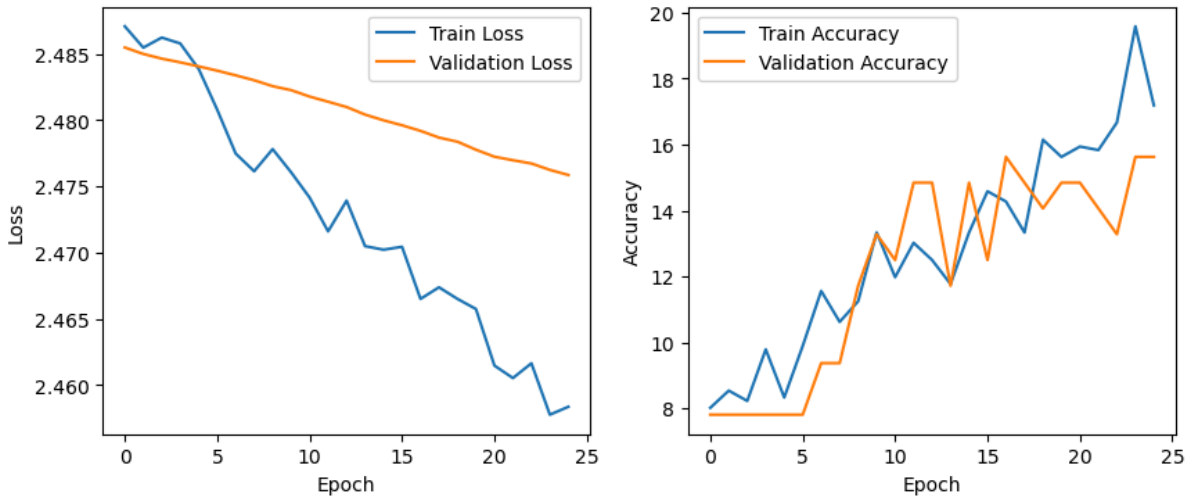

Figure 17 : *Loss et Accuracy en fonction du nombre d'époques pour VGG16 pré-entrainé*

La décroissance très lente de la fonction de perte pour l'ensemble de test indique que le modèle apprend lentement ou a du mal à généraliser. Cela peut suggérer que le modèle est trop complexe pour les données disponibles ou que l'apprentissage nécessite plus d'époques.
De plus, la différence entre la précision sur l'ensemble d'entraînement (20%) et celle sur l'ensemble de validation (16%) suggère un léger sur-apprentissage. VGG16 pré-entrainé performe un peu mieux sur les données qu'il a vues pendant l'entraînement.

- **Résultats sur l'ensemble d'entraînement :**

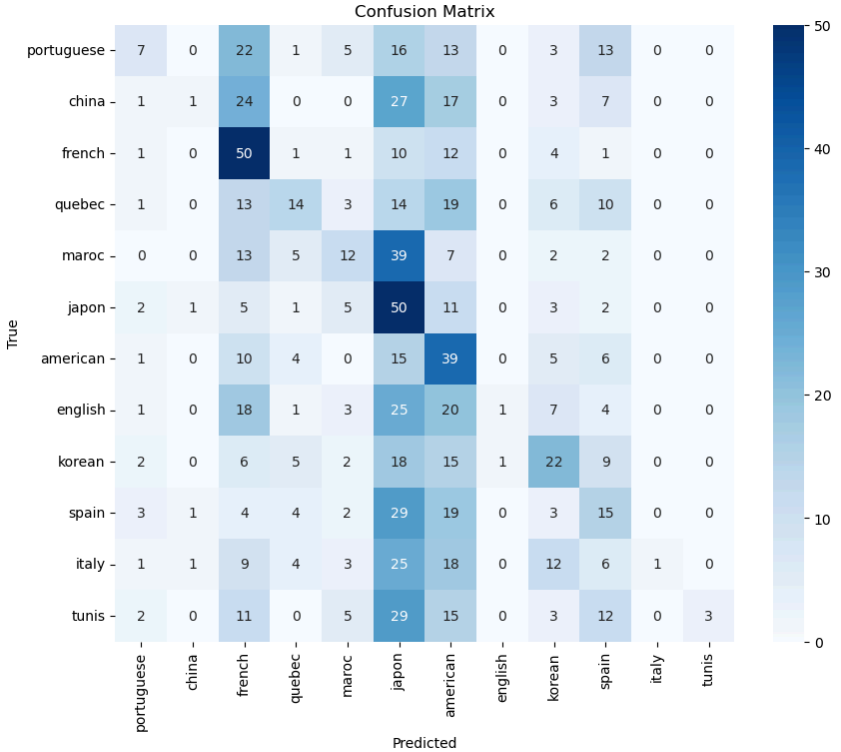

Figure 18 : *Matrice de confusion de l'ensemble de training pour VGG16 pré-entrainé*

Sur l'ensemble d'entraînement, le modèle parvient à prédire correctement le français et le chinois une fois sur deux. Cependant, certaines langues, comme l'italien ou le chinois, sont mal prédites globalement. De plus, le modèle montre une forte tendance à confondre le japonais avec de nombreuses autres langues, ce qui est le cas aussi avec l'américain.

- **Résultats sur l'ensemble de test/ validation :**

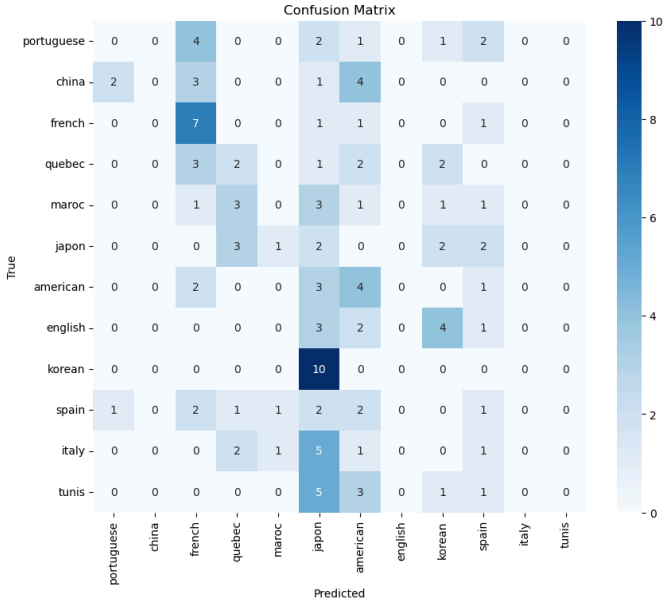

Figure 19 : *Matrice de confusion de l'ensemble de test pour VGG16 pré-entrainé*

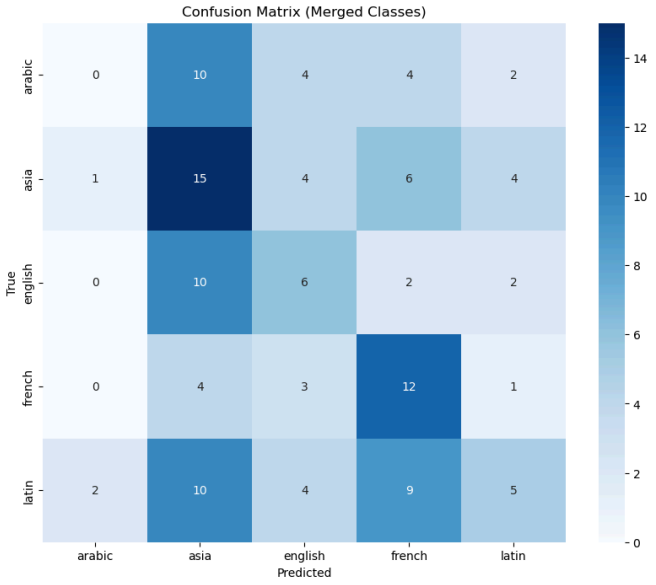

Figure 20 : *Matrice fusionnée de confusion des ensembles de test et validation pour VGG16 pré-entrainé*

La matrice de confusion dans laquelle on a combiné les résultats sur les ensembles de test et de validation révèle une performance supérieure du modèle dans la prédiction des langues asiatiques. Cependant, le modèle éprouve des difficultés à généraliser efficacement pour les autres langues. Cette limitation pourrait s'expliquer par l'insuffisance des données utilisées pour affiner le modèle pré-entraîné.

E. VIT-16

Le modèle ViT-16, basé sur les transformers, a été testé sur un entraînement limité à 5 époques. Cependant, il s'est avéré trop complexe pour notre jeu de données, menant à un sur-apprentissage. Par exemple, toutes les prédictions sur l'ensemble d'entraînement indiquaient la classe "québécois".

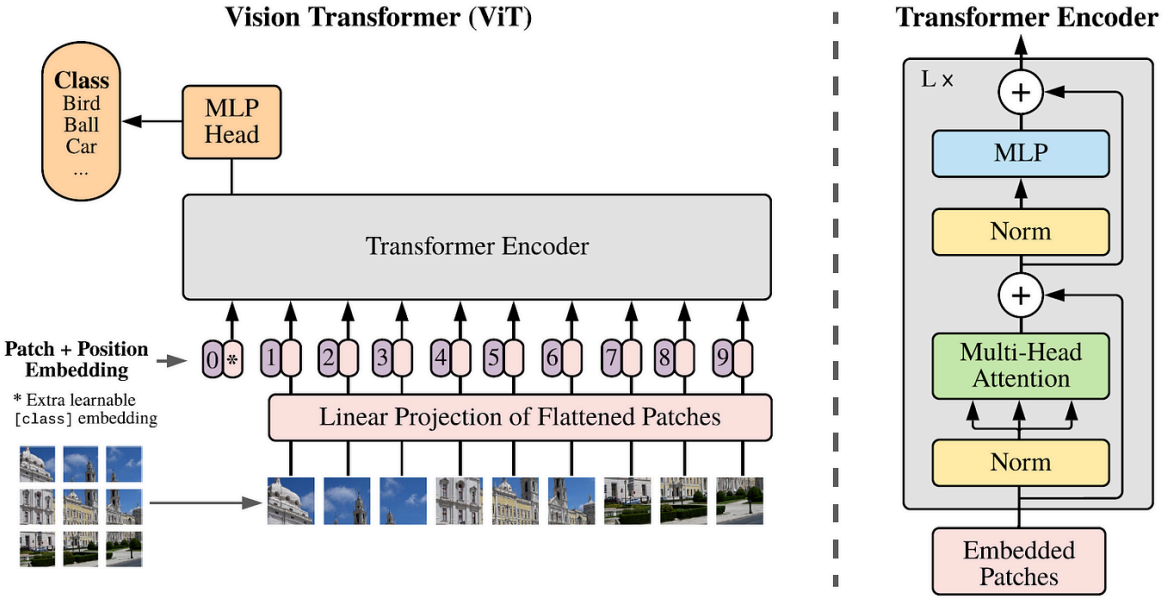

Figure 21 : *Architecture du modèle VIT-16 (86 millions de paramètres)*

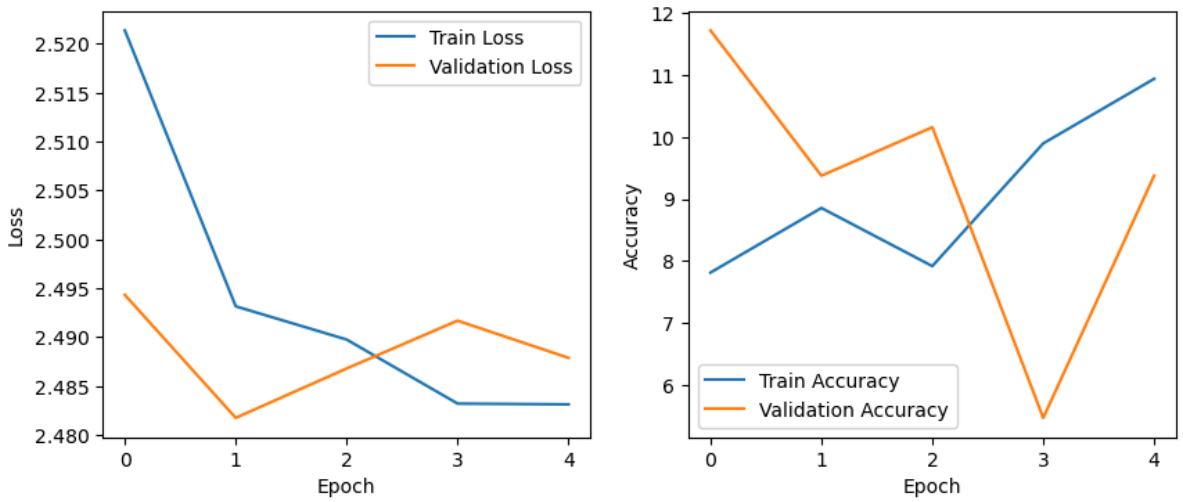

Figure 22 : *Loss et Accuracy en fonction du nombre d'époques pour VIT-16*

- **Résultats sur l'ensemble d'entraînement :**

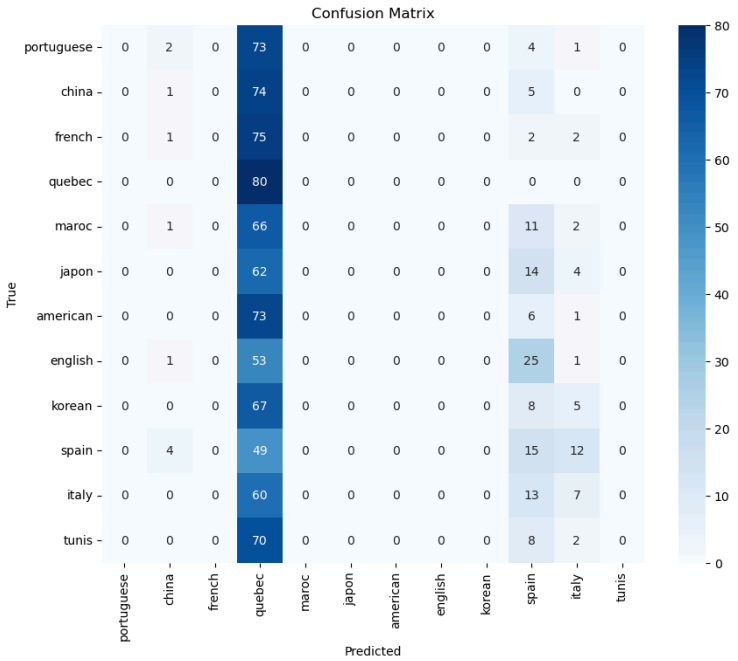

Figure 23 : *Matrice de confusion de l'ensemble de training pour VIT-16*

La matrice de confusion montre que le modèle ViT-16, après un entraînement limité à 5 époques, a systématiquement prédit la classe "québécois" pour toutes les données de l'ensemble d'entraînement, quelle que soit la classe réelle. Cela est visible par la colonne "québécois", qui contient toutes les valeurs non nulles, tandis que les autres colonnes possèdent des valeurs nulles.

En effet, le modèle est trop complexe (86 millions de paramètres) par rapport au volume et à la diversité des données d'entraînement, ce qui l'a conduit à mémoriser les données sans apprendre des généralisations utiles. Ce comportement est prévisible, dû à notre petit jeu de données d'entraînement.

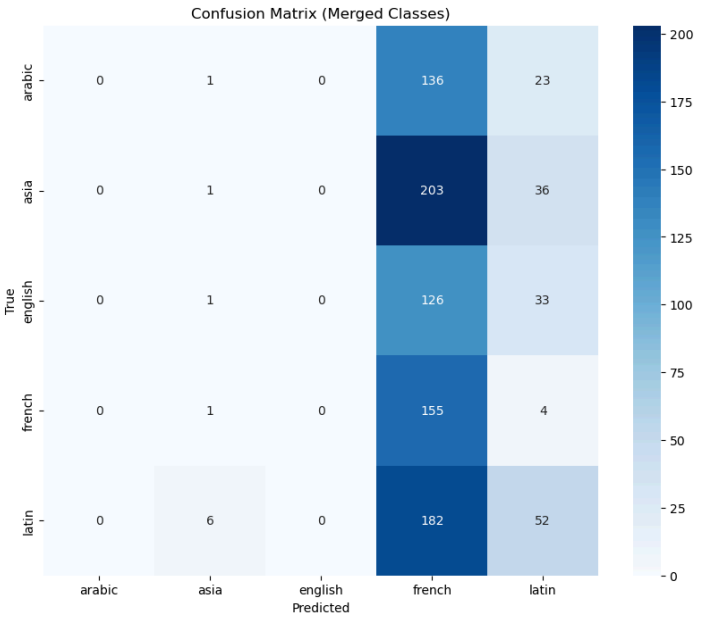

Figure 24 : *Matrice fusionnée de confusion de l'ensemble de training pour VIT-16*

- **Résultats sur l'ensemble de test :**

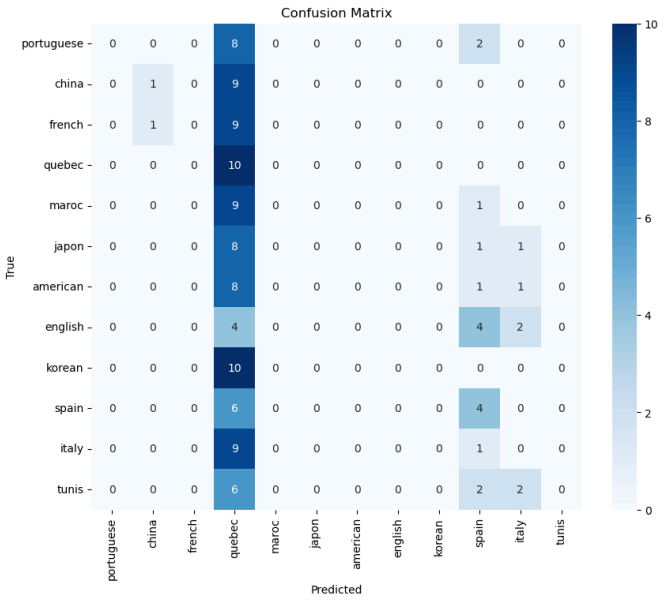

Figure 25 : *Matrice de confusion de l'ensemble de validation pour VIT-16*

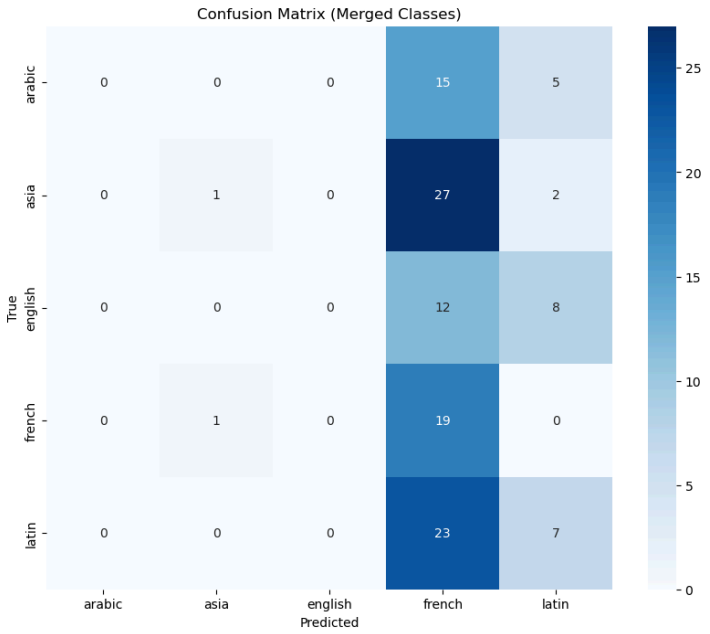

Figure 26 : *Matrice fusionné de confusion de l'ensemble de validation pour VIT-16*

Les observations faites sur les ensembles de test et de validation viennent confirmer les constatations initiales issues de l'ensemble d'entraînement. En effet, le modèle tend à confondre les langues dérivées du français avec les autres langues, suggérant une difficulté à distinguer de manière efficace ces différents groupes.

VI. Conclusion et perspectives

Ce projet nous a permis d'explorer la classification des langues en utilisant des spectrogrammes et d'en identifier les limites. Nous avons confirmé notre hypothèse que la tâche est difficile à accomplir avec nos capacités de temps et de calcul. Les résultats obtenus sont moins bons que nos attentes avec seulement 24% d'accuracy dans le meilleur des cas. Les points d'améliorations possibles sont les suivants:

- Augmenter la taille des données: collecter plus d'échantillons diversifiés pour chaque langue
- Tester d'autres architectures: explorer les modèles convolutionnels récurrents (CRNN).