Compte-rendu de l'apprentissage machine sur un dataset de voyelles

Etienne Ferrandi

16 juin 2022

1 Introduction

Il s'agit de classer automatiquement des images de voyelles manuscrites majuscules. On utilise des méthodes d'apprentissage machine supervisée (*k-nearest neighbors* (knn) et réseaux de neurones) et non-supervisée (*kmeans*). Les réseaux de neurones peuvent avoir soit une architecture simple, soit des couches de convolution. Le dataset sur lequel les modèles sont entraînés est un dossier de 4500 fichiers textes qui comporte un tableau de 18 lignes et 18 colonnes de nombres compris entre 0 et 255. Cela correspond au format RGB et donne à l'image une dimension 18x18. Un tel fichier est lisible par la librairie *matplotlib* de python qui visualise l'image selon la tonalité choisie. Parmi les autres librairies de python mobilisées pour réaliser ce travail, on utilise numpy et pandas pour la préparation des données à soumettre à l'algorithme de machine learning. Les méthodes de *kmeans* et de *knn* sont mises en place grâce à scikit learn. Pour les réseaux de neurones, on utilise tensorflow et keras. Si le modèle est entraîné sur un dataset de 4500 éléments, il sera testé sur un ensemble de 500 images. A chaque fichier du dataset d'entraînement, est associé un label : la bonne voyelle affichée dans l'image. La performance du modèle est évaluée en fonction de la différence entre la prédiction effectuée sur chaque image du dataset d'entraînement et les données cible qui sont les labels. On fera ensuite une prédiction de la voyelle représentée dans chaque image du dataset de test. On évaluera ensuite les performances de ce test en comparant les résultats de la prédiction avec les bonnes voyelles listées comme labels. On dispose de trois notebooks, un contenant les modèles de kmeans et de knn, deux autres avec les réseaux de neurones.

2 Implémentation de l'algorithme kmeans

Cette méthode d'apprentissage non-supervisée s'appuie sur un nombre k de clusters de données dans un dataset, nombre qu'on a soi-même déterminé. L'apprentissage se fait à mesure que chaque nouvelle donnée se rajoute au cluster dont le centroïde est le plus proche. Il s'agit à terme de minimiser la variance des différents clusters, c'est-à-dire la somme des distances entre chaque centroïde et les différentes observations incluses dans le même cluster.

2.1 Résultats

Ces mesures d'évaluation du modèle sont réalisées sur la prédiction sur l'échantillon de test. On observe que le modèle est peu performant car son *accuracy* n'est que de 60%, c'est-à-dire sur dix prédictions, il n'en émet que six qui soient correctes.

	precision	recall	f1-score	support
A	0.88	0.79	0.83	105
E	0.70	0.57	0.63	108
I	0.66	0.72	0.69	92
O	0.45	0.54	0.49	108
U	0.33	0.34	0.34	87
accuracy			0.60	500
macro avg	0.61	0.59	0.60	500
weighted avg	0.61	0.60	0.60	500

FIGURE 1 – Résultats Kmeans

La matrice de confusion de l'entraînement montre que c'est surtout concernant la voyelle 'U' qu'il y a des erreurs de prédiction. En effet, une bonne prédiction sur une image de 'U' est réalisée seulement trois fois sur dix. Près de la moitié des prédictions sur les images de 'U' sont la voyelle 'O'. De même, dans un tiers des cas, les prédictions sur la voyelle 'O' sont 'U'. On a donc une confusion entre 'U' et 'O' dont la forme est proche, qui est la cause majeure de la faible performance de ce modèle.

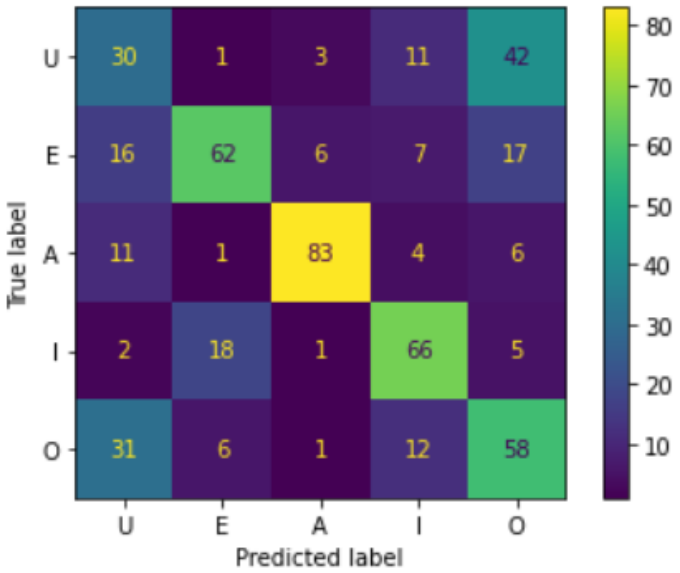

FIGURE 2 – matrice de confusion Kmeans

2.2 Erreurs

L'erreur ci-dessous visualisée confirme l'observation faite précédemment d'une confusion entre 'U' et 'O'. En effet, pour une image faisant figurer un 'O', c'est un 'U' qui est prédit. Les courbes du 'O' qui ont été repérées ressemblent fortement à celles du 'U', ce qui peut expliquer la confusion.

Sur cette deuxième erreur, c'est un 'U' qui a été prédit sur une image faisant figurer un 'E'. En effet, le 'E' a été écrit de telle manière qu'hormis le trait central et la boucle finale supérieure, il peut fortement ressembler à un 'U'.

```
TRUE :  O
PRED:  U

<matplotlib.image.AxesImage at 0x168152f8460>
```

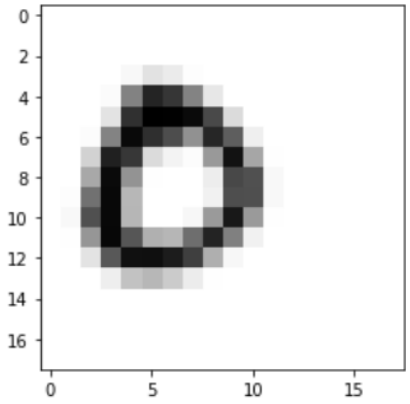

FIGURE 3 – Erreur 1 Kmeans

```
TRUE :  E
PRED:  U

<matplotlib.image.AxesImage at 0x1681536b100>
```

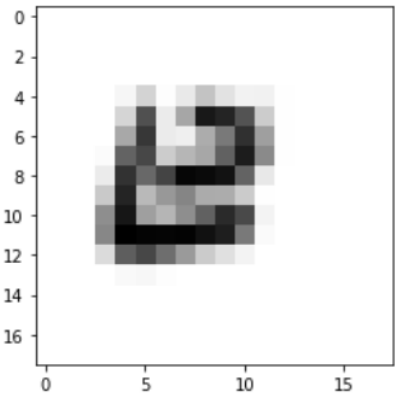

FIGURE 4 – Erreur 2 Kmeans

3 Implémentation de l'algorithme k-nearest neighbors

L'algorithme des knn est une méthode d'apprentissage supervisé. A chaque donnée, est associée un couple entrée-sortie. Pour une nouvelle observation X dont on veut prédire la variable de sortie, on commence par calculer toutes les distances de cette observation avec les autres données puis on retient les k données les plus proches de X selon un calcul de distance (par exemple euclidienne). On calcule enfin le mode des valeurs de sortie ces k données, c'est-à-dire la valeur de sortie parmi les k données qui apparaît le plus souvent. Celle-ci sera la valeur prédite pour la nouvelle observation X.

3.1 Résultats

Sur notre dataset d'entraînement, l'algorithme des knn affiche un score d'accuracy de 97%, ce qui en fait un modèle presque parfaitement fiable.

	precision	recall	f1-score	support
A	1.00	0.98	0.99	105
E	0.97	0.96	0.97	108
I	0.96	0.98	0.97	92
O	0.95	0.98	0.97	108
U	0.98	0.95	0.97	87
accuracy			0.97	500
macro avg	0.97	0.97	0.97	500
weighted avg	0.97	0.97	0.97	500

FIGURE 5 – Résultats KNN

La diagonale de la matrice de confusion montre que les erreurs sont très peu nombreuses.

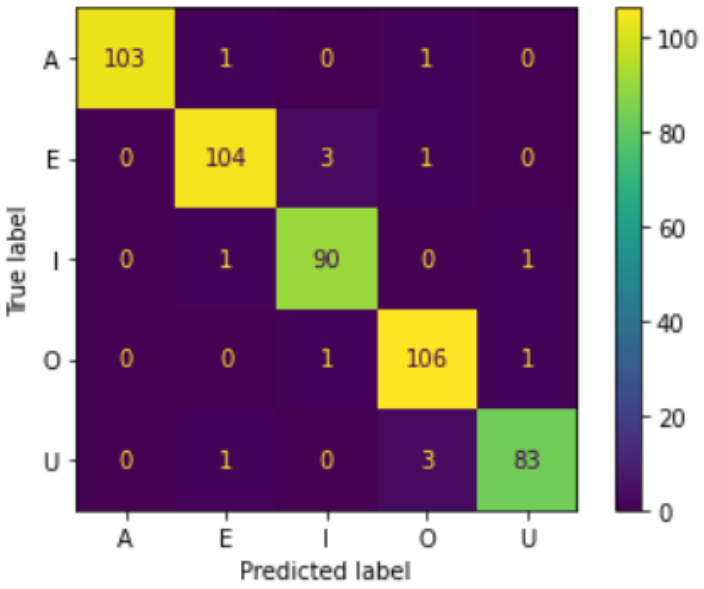

FIGURE 6 – Matrice de confusion KNN

3.2 Erreurs

Parmi les quatorze erreurs répertoriées, on observe qu'une voyelle 'I' a été prédite à tort comme un 'U'. Cependant, même un oeil ne reconnaît pas dans cette image un 'I'...

On peut apporter la même conclusion concernant cette lettre 'E' reconnue comme un 'O'. Cela peut être dû aux courbures de cette voyelle qui rappelle le 'O'.

```
TRUE VALUE :  I
PREDICTION:  U

<matplotlib.image.AxesImage at 0x1ac93f84a90>
```

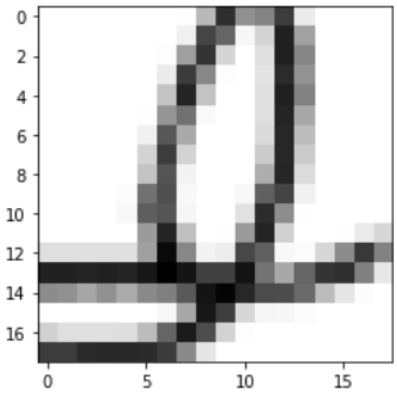

FIGURE 7 – Erreur 1 KNN

```
TRUE :  E
PRED:  O

<matplotlib.image.AxesImage at 0x168154ae9a0>
```

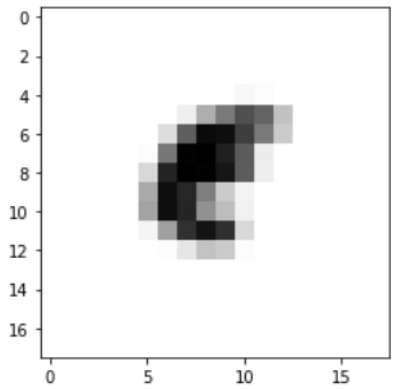

FIGURE 8 – Erreur 2 KNN

4 Implémentation d'un réseau de neurones (NN)

Le réseau de neurones NN utilisé est d'architecture simple car il comporte seulement les couches de base d'un RNN. Premièrement, on a la couche Flatten qui prend en entrée chaque image de dimension 18x18 et qui réduit cette matrice en un tableau à une dimension de 324 entrées. Deuxièmement, cette couche Dense est une simple couche qui connecte tous les neurones aux 128 neurones de la couche précédente par le calcul du produit des neurones en entrées avec leur poids associé, auquel est ajouté un biais, le tout pris dans une fonction d'activation, ici ReLu. Enfin, une dernière couche Dense prend en entrée 5 neurones de la couche précédente.

Le modèle est ensuite compilé au moyen d'un optimisateur utile pour minimiser la fonction de coût. Ici, on utilise Adam (Adaptive Moment Estimation) qui est un algorithme de descente de gradient. Il agit sur

une *cross entropy loss*, qui est la somme de la probabilité réelle multipliée par la probabilité logarithmique prédite sur toutes les classes. Pour évaluer le modèle, on s'appuie sur une métrique d'évaluation, l'*accuracy*.

4.1 Résultats

On entraîne le modèle sur le dataset d'entraînement pendant 10 époques, à l'issue desquelles on obtient une *accuracy* de plus de 92% et une loss de 25%. Sur le dataset de test, la performance exceptionnelle du modèle se vérifie puisque l'*accuracy* s'élève à plus de 93% et la *loss* à plus de 22%.

	precision	recall	f1-score	support
A	0.95	0.96	0.96	105
E	0.94	0.90	0.92	108
I	0.92	0.95	0.93	92
O	0.94	0.95	0.94	108
U	0.92	0.91	0.91	87
accuracy			0.93	500
macro avg	0.93	0.93	0.93	500
weighted avg	0.93	0.93	0.93	500

FIGURE 9 – Résultats RNN

On voit sur la matrice de confusion que la diagonale des bonnes prédictions est très bien remplie.

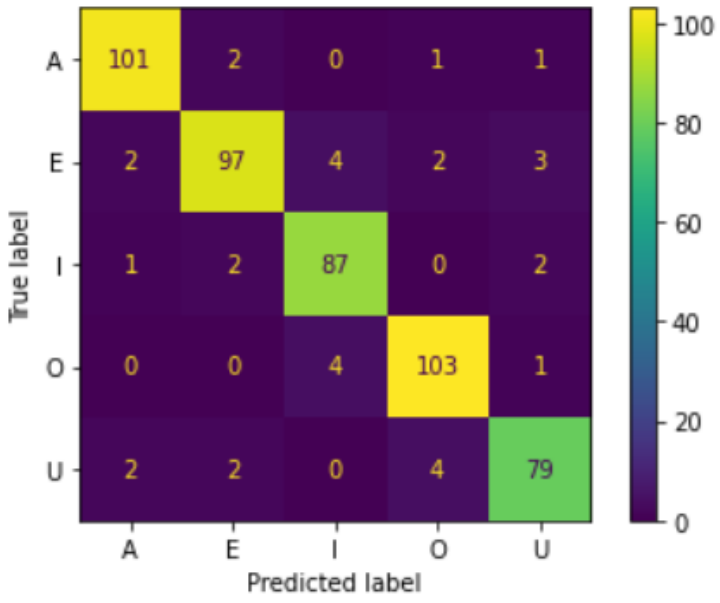

FIGURE 10 – Matrice de confusion RNN

4.2 Erreurs

Sur le dataset de test, le modèle enregistre seulement deux erreurs affichées ci-dessous. La voyelle 'E' est prédite comme un 'U', voire comme un 'A', pour les raisons expliquées ci-dessus. La voyelle 'I' est prédite

comme un 'U' à cause des courbes qui peuvent rappeler le 'U'.

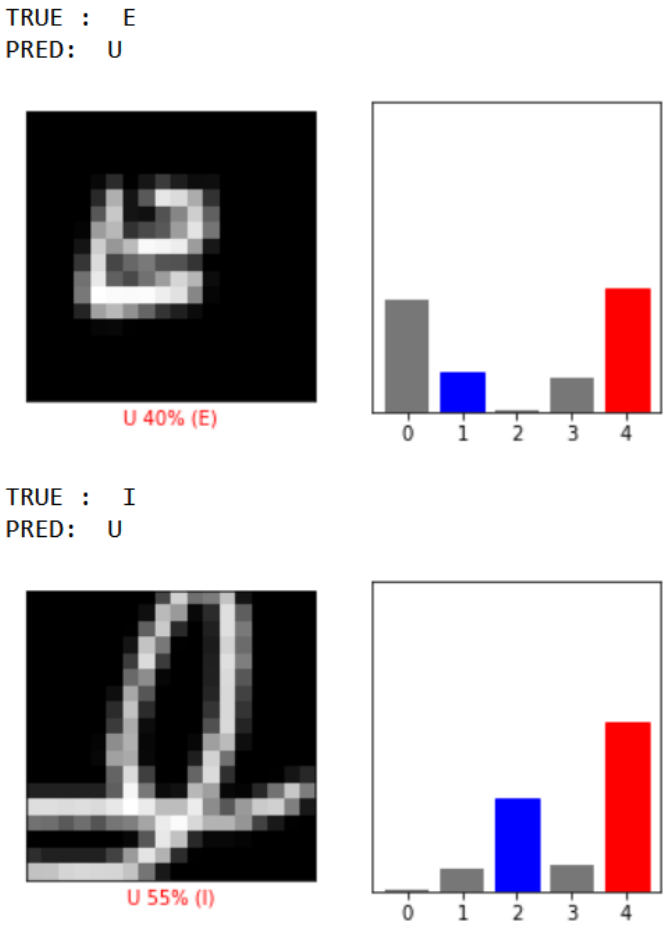

FIGURE 11 – Erreurs du RNN

5 Implémentation du Réseau de neurones convolutifs (CNN)

Ce dernier modèle diffère du précédent essentiellement en raison des couches de neurones de convolution qu'on ajoute aux couches déjà présentes. Au nombre de trois, elles traitent des motifs des images en entrée de dimension 3x3. La première couche contient 64 filtres et les deux suivantes, 32. Leur fonction d'activation est ReLu. Les couches de MaxPooling permettent de réduire la dimension de chaque image en extrayant la valeur la plus important de chaque motif des filtres.

5.1 Résultats

Avec une *accuracy* de 99% suite à l'entraînement du modèle sur le dataset de *train*, on peut dire que la performance est optimale.

	precision	recall	f1-score	support
A	1.00	1.00	1.00	105
E	1.00	0.99	1.00	108
I	0.99	0.98	0.98	92
O	1.00	0.99	1.00	108
U	0.97	1.00	0.98	87
accuracy			0.99	500
macro avg	0.99	0.99	0.99	500
weighted avg	0.99	0.99	0.99	500

FIGURE 12 – Résultats du CNN

La matrice de confusion montre qu'il y a en effet très peu d'erreurs.

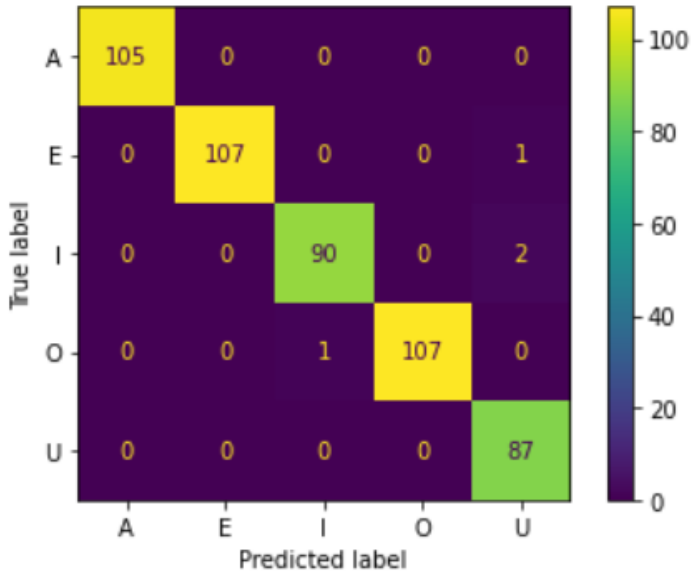

FIGURE 13 – Matrice de confusion CNN

5.2 Erreurs

Parmi les deux erreurs répertoriés, on retrouve l'image habituellement prédite comme étant un 'U' alors que c'est un 'I'. A la différence des résultats précédents, l'attribution à 'U' est à 100%. Le modèle testé sur le dataset de test persiste donc dans l'erreur pour cette image-ci. L'erreur de prédiction d'une voyelle 'O' comme un 'E' provient de l'aire assez réduite du cercle, qui peut rappeler la petite boucle au sommet du 'E'.

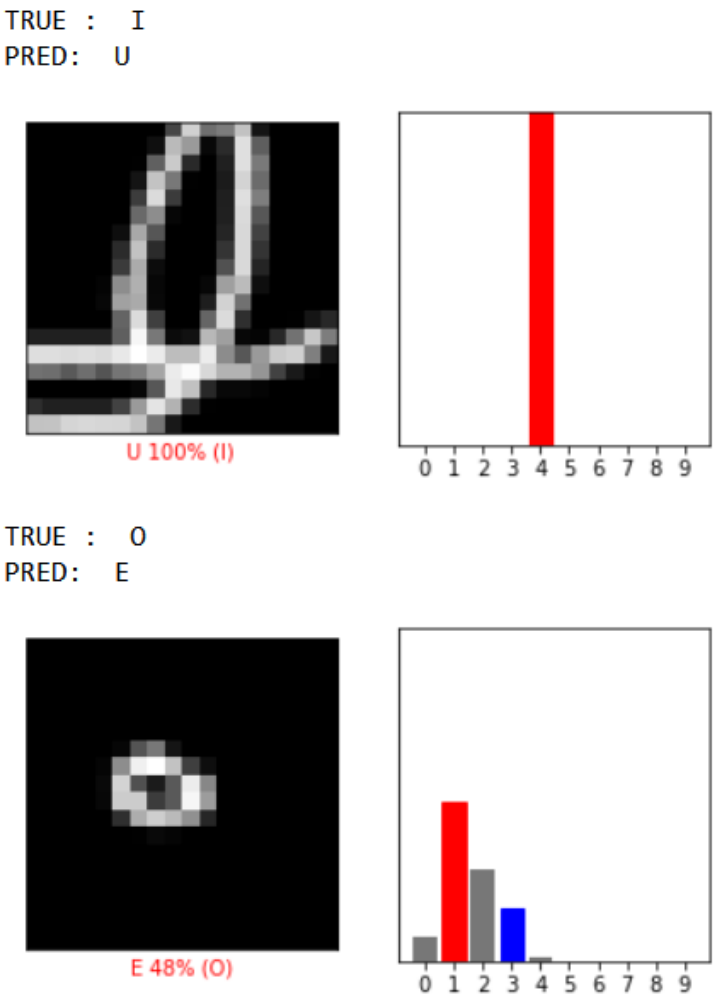

FIGURE 14 – Erreurs CNN

6 Conclusion

Parmi les quatre modèles entraînés, les deux modèles les plus performants sont les réseaux de neurones qui n'enregistrent que deux erreurs de prédiction sur le dataset de test. L'algorithme des knn est bien plus performant que celui des kmeans avec seulement quatorze erreurs de prédiction. De ces quatorze erreurs, on observe que les réseaux de neurones en résolvent un bon nombre. Ils sont capables de faire des prédictions sur des images dont la voyelle réellement affichée n'est pas particulièrement évidente, même pour un oeil humain.