

Compte rendu de la validation du cours d'apprentissage machine

Jean Barré

9 juin 2022

1 Introduction

Dans ce devoir, nous travaillons sur une application des algorithmes de l'apprentissage machine pour classifier automatiquement des images. Nous disposons d'un corpus de 5000 images de voyelles manuscrites majuscules.

Nous implémenterons différentes techniques d'apprentissage machine, en non-supervisée (clustering avec la méthode des k-means) et en supervisée (classification avec la méthode des k plus proches voisins). Nous mettrons en place également en supervisé différentes architectures de réseaux de neurones pour classifier nos images. L'objectif est d'évaluer ces différentes techniques sur notre tâche, qui est d'entraîner nos algorithmes à reconnaître la voyelle qui se trouve dans une image donnée. La tâche est à première vue difficile, puisque l'écriture manuscrite est très variable dans la forme, le trait des lettres, et même un œil humain avisé peut commettre des erreurs.

Nous fondons notre travail sur le langage de programmation Python et les bibliothèques construites au-dessus. Pour l'analyse des données et leur manipulation nous utilisons Pandas¹ et Numpy². Nous utilisons aussi Scikitlearn³, qui donne des outils efficaces pour l'analyse prédictive de données. Scikitlearn est assez simple à utiliser et implémente des algorithmes d'apprentissage machine au niveau de l'état de l'art. Pour la visualisation des résultats nous utilisons les bibliothèques seaborn⁴ et matplotlib⁵. Pour les réseaux de neurones, nous utiliserons les bibliothèques Tensorflow⁶ et Keras⁷.

Nous mettons en place les bases de l'apprentissage machine. Le jeu de données est séparée en deux échantillons.

- Un premier sur lequel nous entraînons le modèle statistique, c'est à dire que nous lui donnons le label associé pour chaque image.
- Un autre sur lequel nous évaluons ses performances de prédictions sur des données qu'il n'a jamais vu.

Nous mesurons ainsi la capacité des modèles à généraliser.

1. <https://pandas.pydata.org/>

2. <https://numpy.org/>

3. <https://scikit-learn.org/stable/index.html>

4. <https://seaborn.pydata.org/>

5. <https://matplotlib.org/>

6. <https://www.tensorflow.org/>

7. <https://keras.io/>

Nous séparons en quatre parties notre travail, une sur l'algorithme des K-moyens, une sur les K plus proches voisins, une sur un réseau de neurone avec une architecture simple et enfin une dernière avec les réseaux de neurones avec des couches de convolutions. Le code et les visualisations des sections 1 et 2 se trouvent dans le notebook numéro 1, tandis que les réseaux de neurones se trouvent dans le notebook numéro 2.

2 Implémentation de l'algorithme K-means

Dans cette section nous implémentons l'algorithme des K-moyens. Ce dernier est une approche de clustering qui, à partir d'un jeu de données, cherche à déterminer K clusters dans les données. Chaque nouvel élément est rattaché à un cluster à l'aide de la mesure de sa distance au plus proche centroïde, c'est à dire le centre du cluster. L'objectif de cette technique est de minimiser la variance au sein de chaque cluster, pour avoir k clusters les mieux définis possibles.

2.1 Résultats

La figure 1 montre les résultats de notre implémentation sur nos images. Les mesures sont réalisées sur la prédiction du modèle sur l'échantillon de test, en comparant la prédiction avec la vraie valeur de terrain.

	precision	recall	f1-score	support
A	0.88	0.79	0.83	105
E	0.71	0.56	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 – Rapport de classification du KMEANS

Notre modèle est assez peu performant. Il atteint 60% d'efficacité (accuracy), c'est à dire qu'il réalise une prédiction correcte 6 fois sur 10. Le modèle est plutôt bon pour reconnaître les voyelles A et I, mais beaucoup moins pour reconnaître les voyelles O et U (il reconnaît le U sans erreurs seulement 1 fois sur 3). On pourrait expliquer cela par le fait que l'écriture manuscrite du A et du I est assez marqué tandis que le O et le U sont des formes plus similaires, ce qui pourrait expliquer la confusion du modèle, mais ce ne sont à ce stade que des hypothèses.

Pour y voir plus clair, nous projetons ces résultats dans une matrice de confusion en figure 2. Dans le cas d'une prédiction parfaite, seulement la diagonale de la matrice de confusion est remplie.

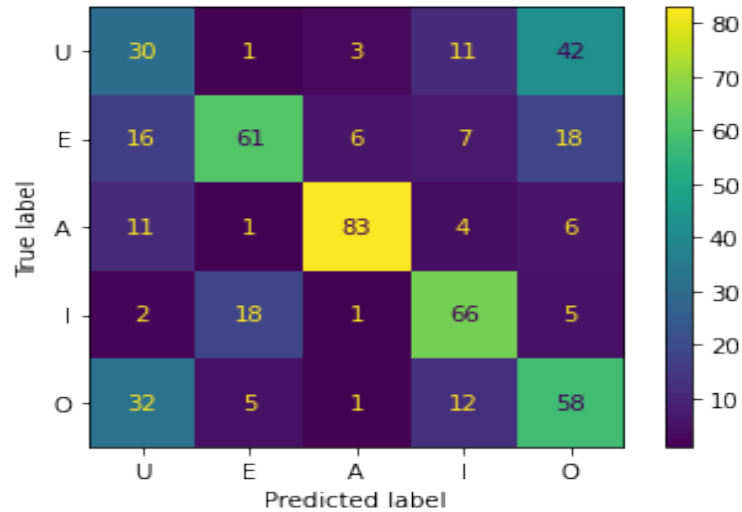


FIGURE 2 – Matrice de confusion du KMEANS

2.2 Matrice de confusion

Maintenant, on constate les erreurs de manière plus visuelle. On voit bien la confusion du modèle pour les lettre U et O, avec de nombreuses erreurs de prédiction concernant ces deux lettres.

2.3 Étude des erreurs

Pour comprendre davantage les erreurs, nous étudions trois d'entre elles.

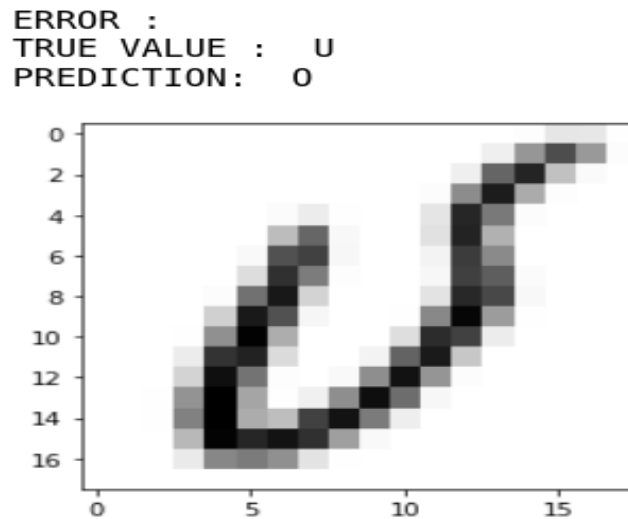


FIGURE 3 –

Pour notre première erreur en figure 3, le modèle a prédit l'étiquette O pour cette image. Cela est comptabilisé comme une erreur, puisque la vraie valeur est un U. On comprend l'erreur du modèle car la lettre en question est assez mal dessinée, avec un trait un peu bizarre en haut à droite de l'image.

ERROR :
TRUE VALUE : E
PREDICTION: O

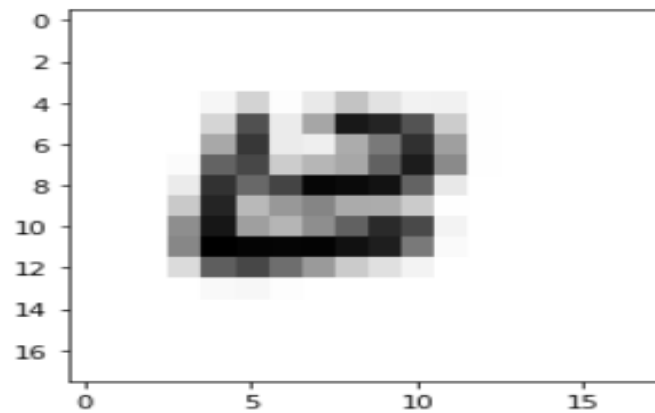


FIGURE 4 –

Pour notre deuxième erreur en figure 4, le modèle a prédit O alors que la valeur de terrain était le E. Encore une fois, le E est assez mal réalisé et le modèle ne parvient pas à reconnaître la lettre donnée. La donnée de terrain est beaucoup trop bruitée ici.

ERROR :
TRUE VALUE : I
PREDICTION: E

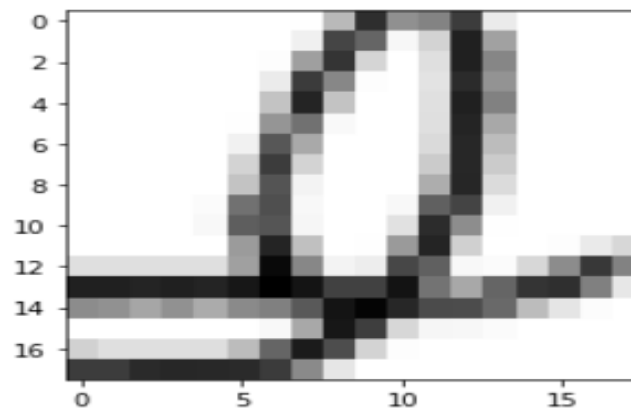


FIGURE 5 –

Pour cette dernière erreur en figure 5, le modèle prédit un E alors que la vraie valeur est un I. Sur ce coup là, on ne peut pas en vouloir au modèle, qui fait face à un I très bizarre. On peut conclure que le modèle a du mal dès que la lettre manuscrit sort trop de la norme sur laquelle il s'est entraînée.

3 Implémentation de l’algorithme K-Nearest-Neighbors

Dans cette section nous implémentons l’algorithme des K plus proches voisins. Ce dernier est une approche de d’apprentissage supervisée qui, à partir d’un jeu de données, et de K couples entrées-sorties, détermine avec un vote de ses K couples la sortie associée à une nouvelle donnée. Dans notre tâche, l’algorithme retiendra pour un échantillon donné la classe la plus représentée parmi les k sorties associées aux k entrées les plus proches de cet échantillon.

3.1 Résultats

La figure 6 montre les résultats de notre implémentation de cet algorithme sur nos images.

	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 6 – Rapport de classification du KNN

On constate que les résultats sont bien meilleurs que précédemment. Le modèle atteint 97% d’efficacité, ce qui est une prédiction quasiment parfaite.

3.2 Matrice de confusion

Nous projetons dans une matrice de confusion en figure 7 les résultats.

La diagonale de la matrice de confusion est bien remplie, quelques erreurs restent à constater mais les performances sont vraiment bonnes.

3.3 Étude des erreurs

La figure 8 montre que l’erreur précédemment faite par l’algorithme des kmeans est corrigée par le knn. La lettre U est bien prédite dans notre cas.

Il en va de même avec la figure 9. La lettre E est bien prédite par notre modèle. Il semble que le KNN soit un algorithme bien plus adapté à notre tâche que le KMEANS.

Cependant, comme on peut le constater en figure 10, le modèle est capable de faire des erreurs. Cette dernière tend à confirmer que la donnée de terrain est soit fausse soit vraiment illisible.

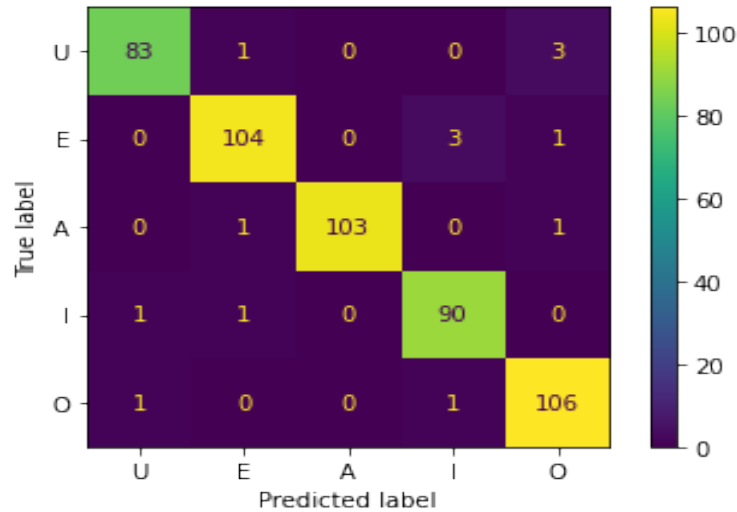


FIGURE 7 – Matrice de confusion du KNN

NO ERROR :
TRUE VALUE : U
PREDICTION: U

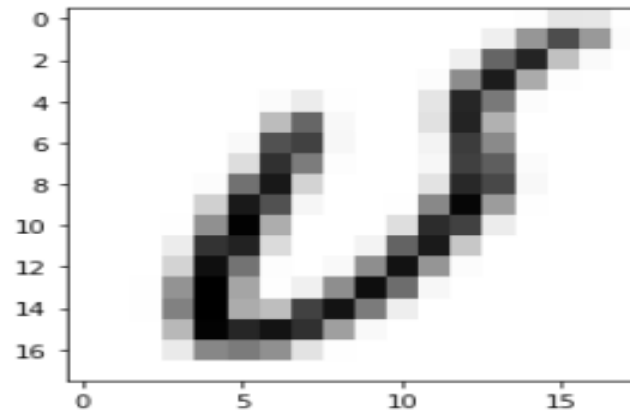


FIGURE 8 –

4 Implémentation d'un réseau de neurones (NN) pour la classification d'images

Dans cette section, nous implémentons un réseau de neurones grâce au code donné et aux bibliothèques Tensorflow et Keras. L'architecture de ce réseau de neurones est « simple » dans le sens où elle comprend seulement les couches de base d'un réseau de neurone. Les réseaux de neurones permettent de résoudre des problèmes complexes de prédiction et font parti du champ de l'apprentissage profond. Un réseau de neurone est composé de différentes couches qui sont elles mêmes composées de nœuds, ou « neurones ». Ces derniers possèdent un certain poids et c'est cet élément qui va permettre l'apprentissage d'inférences statistiques. En effet, les poids des neurones sont optimisés au fil des époques de l'apprentissage (c'est à dire une traversée complète du réseau). Un élément important à noter est la fonction de perte, qui est utilisée pour optimiser les poids du

NO ERROR :
TRUE VALUE : E
PREDICTION: E

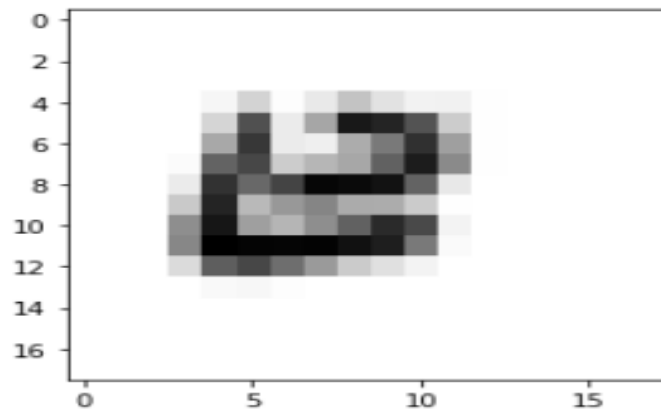


FIGURE 9 –

ERROR :
TRUE VALUE : I
PREDICTION: U

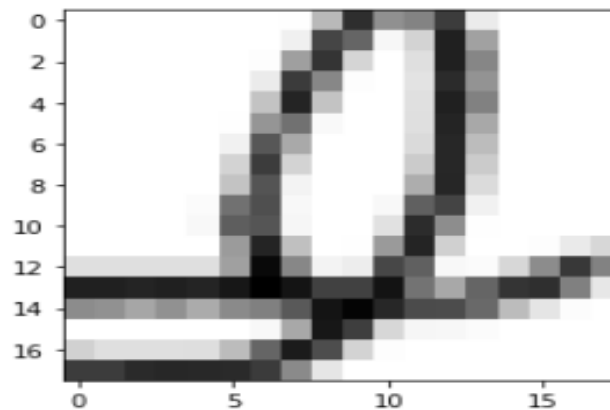


FIGURE 10 –

modèle à chaque époque. Il s'agit de la fonction dont le résultat sera minimisée par lors de l'entraînement du modèle. Nous implémentons un réseau de neurones « simple » pour voir comment il réagit à notre tâche.

4.1 Résultats

Les résultats du réseau sont très bons comme on peut le constater en figure 11, le modèle atteint 92% d'efficacité. C'est un peu moins bon que le KNN mais cela reste très solide.

Les voyelles A et O sont particulièrement bien reconnues, la moins bonne étant la lettre E, mais elle est quand même prédite de la bonne manière 9 fois sur 10.

	precision	recall	f1-score	support
A	0.96	0.94	0.95	105
E	0.93	0.88	0.90	108
I	0.89	0.95	0.92	92
O	0.94	0.94	0.94	108
U	0.90	0.92	0.91	87
accuracy			0.92	500
macro avg	0.92	0.92	0.92	500
weighted avg	0.92	0.92	0.92	500

FIGURE 11 – Rapport de classification du réseau de neurones

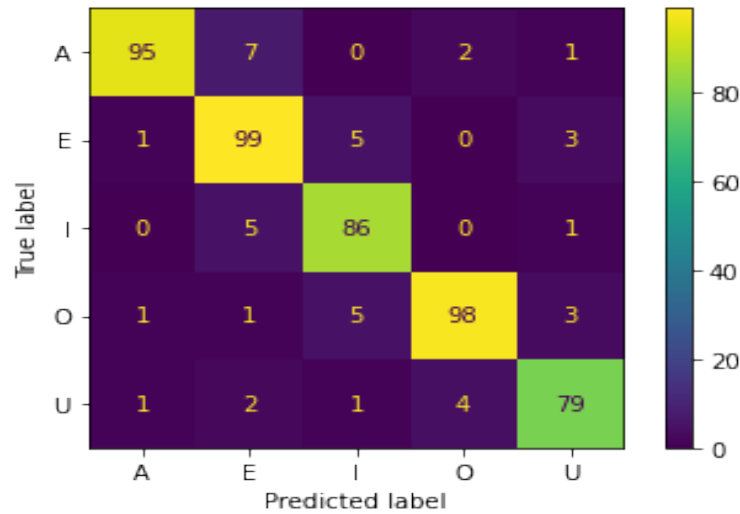


FIGURE 12 – Matrice de confusion du réseau de neurones à convolutions

4.2 Matrice de confusion

La matrice de confusion en figure 12 montre que la diagonale de bonne prédiction est bien remplie, avec quelques erreurs notamment pour le lettre I qui est prédite un peu trop souvent (5 fois à la place de la voyelle O et 5 fois à la place de E). Dans l'ensemble les résultats sont très bons.

4.3 Étude des erreurs

Avec ces méthodes, on peut récupérer la probabilité associée à chaque classe pour une voyelle donnée.

Par exemple, en figure 13, on peut voir que le modèle est assez sûr de lui, il prédit la voyelle I. Cependant, on constate une petite barre en 1, qui est la lettre E.

En figure 14, le modèle fait une erreur puisqu'il prédit la voyelle U alors que la valeur terrain est la voyelle O. On constate une vraie hésitation du modèle, puisque la barre en 3, qui correspond à la lettre O, n'est pas loin de gagner la mise.

La figure 15 montre une dernière erreur du modèle. Cette fois-ci, la lettre U est prédite alors que la vraie voyelle est la E. Le E est assez mal dessiné, ce qui pourrait

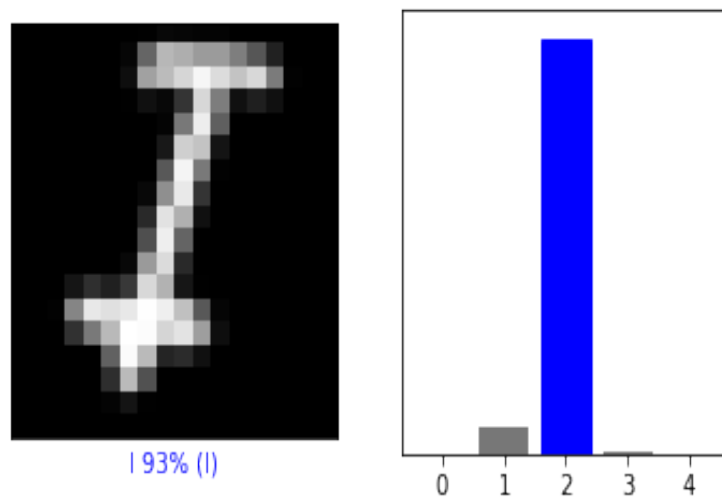


FIGURE 13 –

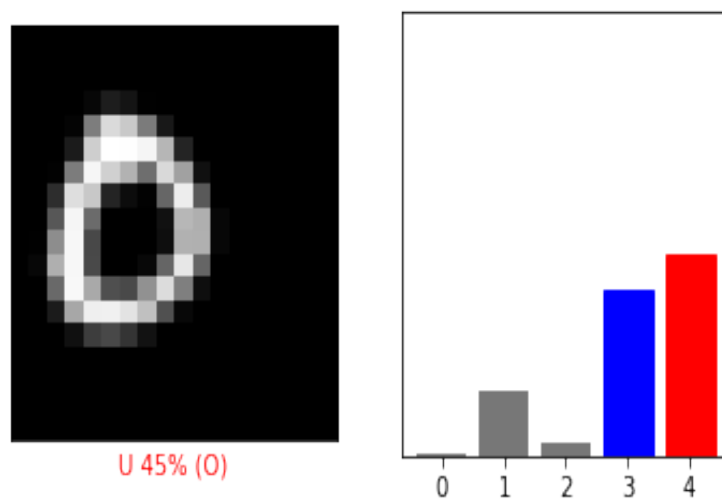


FIGURE 14 –

expliquer cette erreur, d'autant que la probabilité du E en 1 est assez élevé.

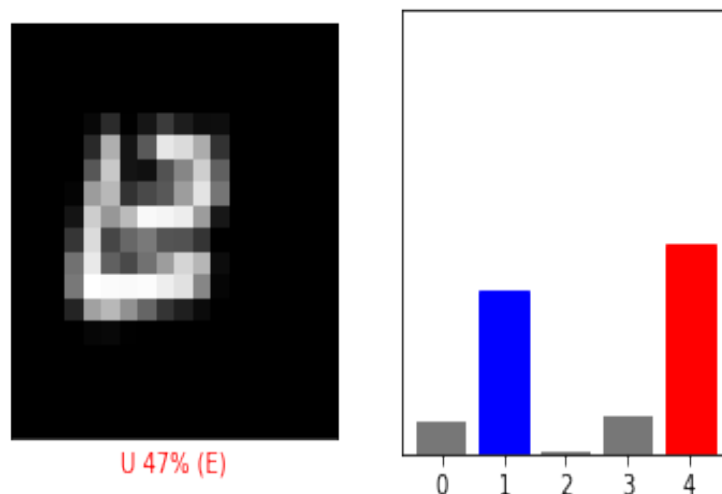


FIGURE 15 –

5 Implémentation d'un réseau de neurones à convolutions (CNN) pour la classification d'images

Notre dernier algorithme consiste à ajouter une couche de convolution au réseau de neurones précédant. La couche de convolution permet de spécialiser le modèle pour la gestion des images. Cette couche consiste à présenter les données d'une image d'une certaine manière qui permet aux neurones de la couche de prendre en compte les contextes d'un pixel d'une image dans leur poids de décision.

5.1 Résultats

En figure 16 nous projetons les résultats de cette approche. Ce sont les meilleurs résultats de nos quatre modèles, avec 98% d'efficacité. Le modèle reconnaît parfaitement les voyelles qu'on lui présente.

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

FIGURE 16 – Rapport de classification du réseau de neurones à convolutions

5.2 Matrice de confusion

La matrice de confusion en figure 17 montre une diagonale de prédiction quasiment parfaite. On pourrait souligner que le O est un peu trop prédit, avec deux O à la place de deux E par exemple, mais cela reste un épiphénomène.

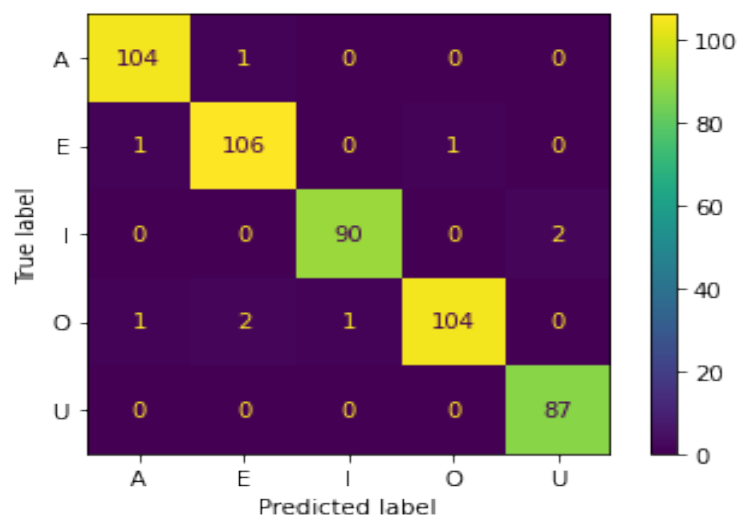


FIGURE 17 – Matrice de confusion du réseau de neurones à convolutions

5.3 Étude des erreurs

On constate dans l'étude des erreurs que le modèle est très sûr de lui. Par exemple en figure 18, la voyelle I a une probabilité d'être un I selon le modèle de 100%. L'hésitation minimale en figure 13 du réseau de neurones « simple » n'existe plus.

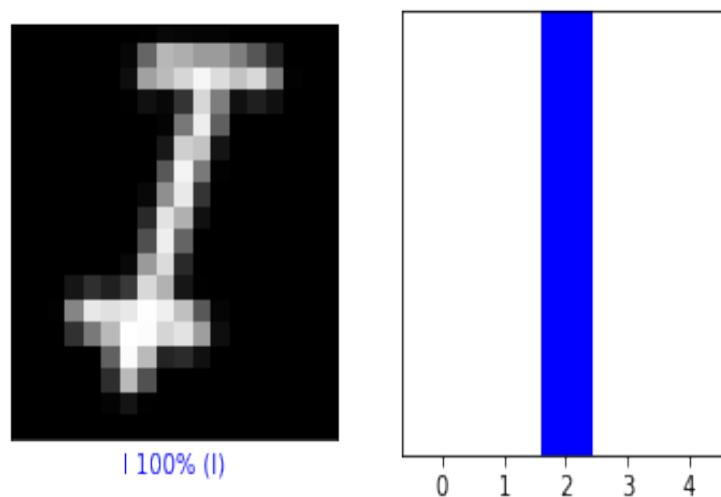


FIGURE 18 –

Il en va de même en figure 19. Contrairement à la figure 14, le modèle ne fait pas l'erreur mais est à 100% sûr de lui.

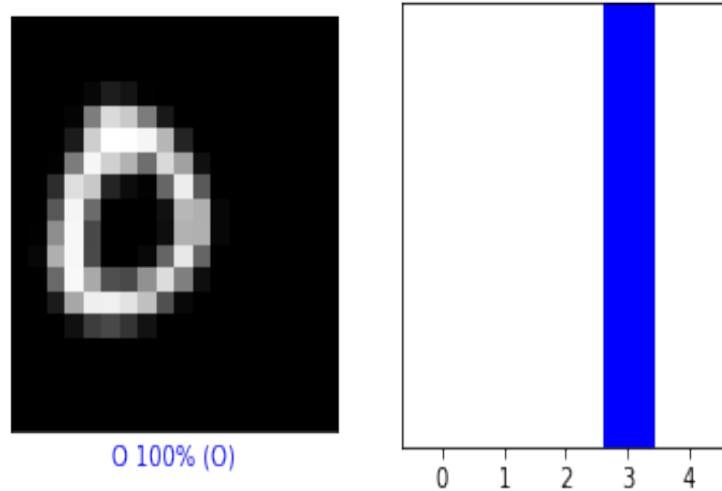


FIGURE 19 –

La figure 20 montre qu'il reste une mini hésitation de 2% pour la lettre A, alors qu'en figure 15 elle était d'environ 10%.

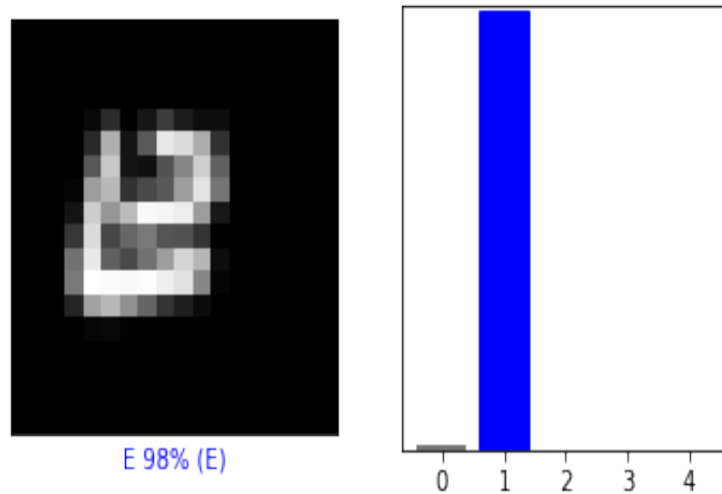


FIGURE 20 –

Cependant, notre réseau de neurones à convolutions n'est pas infallible, et la vue d'ensemble de la figure 21 permet de le constater.

L'erreur

6 Conclusion

Ainsi, nous avons pu entraîner quatre différents algorithmes d'apprentissage machine sur la tâche de classification de voyelles manuscrites. Nous avons fait appel à différents champ de l'apprentissage machine, en contexte non-supervisé avec le clustering ainsi qu'en contexte supervisé avec la classification du K plus proche voisin et nos deux réseaux de neurones.

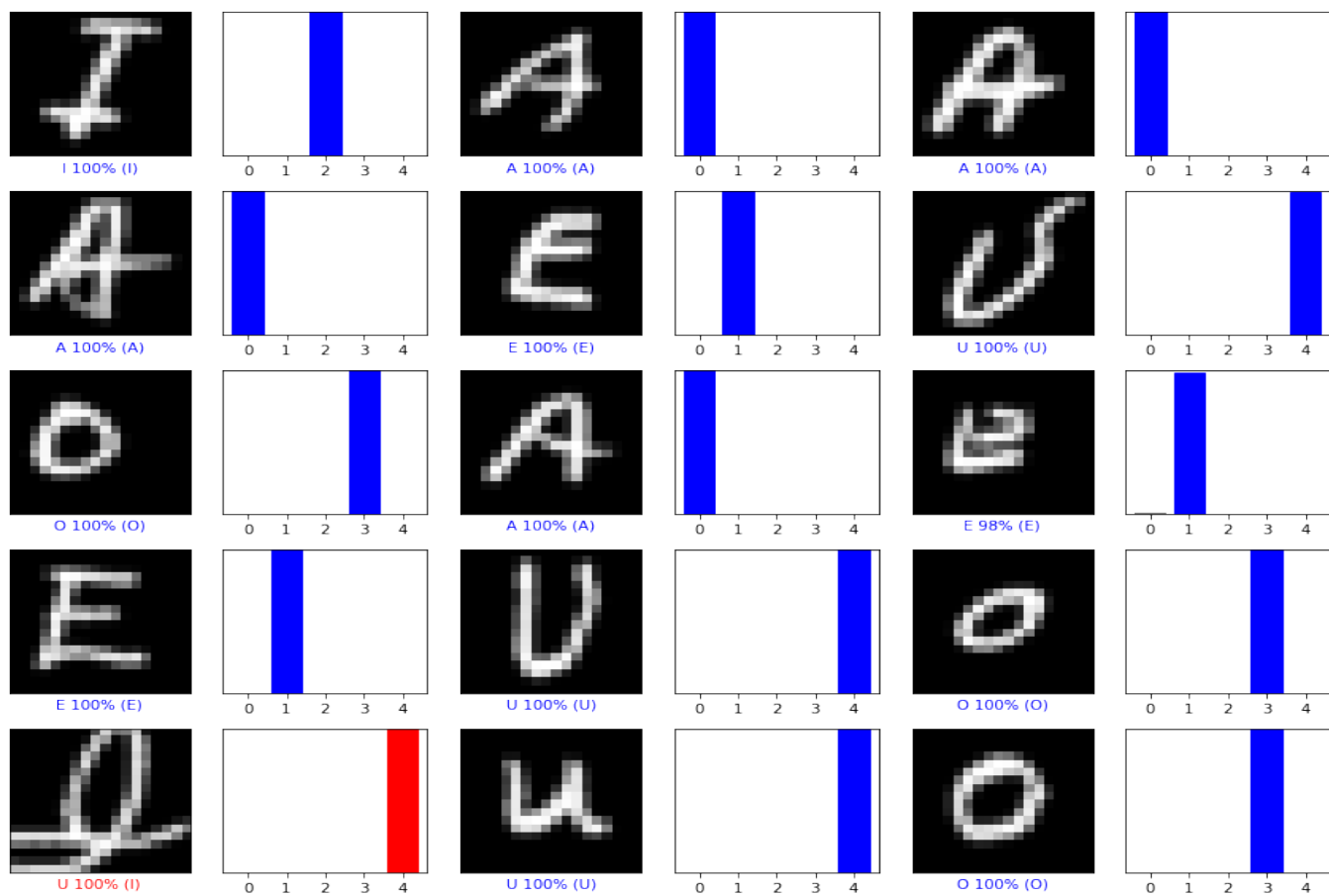


FIGURE 21 –