Classification des chiffres manuscrits

23/03/2022

Sommaire

1 - Import des données

Les données sont récupérées sur internet, et se décomposent en deux dossiers principaux, un dossier pour les images de train, un dossier pour les images de test.

Les images sont triées par dossier de labels.

Des boucles permettent de récupérer les images, le module opencv transforme les images en np.arrays, et les labels sont mis dans une liste à part.

Toutes les listes sont converties en np.arrays. Les dimensions des tableaux sont vérifiées et il n'y a pas d'erreur. Il y a 60000 images dans le dossier d'entraînement, 10000 dans le dossier de test.

Les images sont des carrés de 28 pixels, avec 3 canaux de couleurs (R, V, B).

```
Taille des données d'entraînement : (60000, 28, 28, 3)
Taille de la cible d'entraînement : (60000,)
Taille des données de test : (10000, 28, 28, 3)
Taille de la cible de test : (10000,)
```

2 - Stratégies de traitement

Différentes stratégies de traitement des images ont été testées dans le modèle ci-après.

La première a consisté à garder des images avec les 3 canaux.

La deuxième à ne conserver qu'un canal (import en niveaux de gris).

Pour la troisième est ajoutée à la deuxième une normalisation (min-max).

Pour la quatrième, tous les niveaux de gris sont définis à blanc ou noir (seuil défini à 50 pour le partage 0 - 255).

3 – Création du modèle sur tensorflow

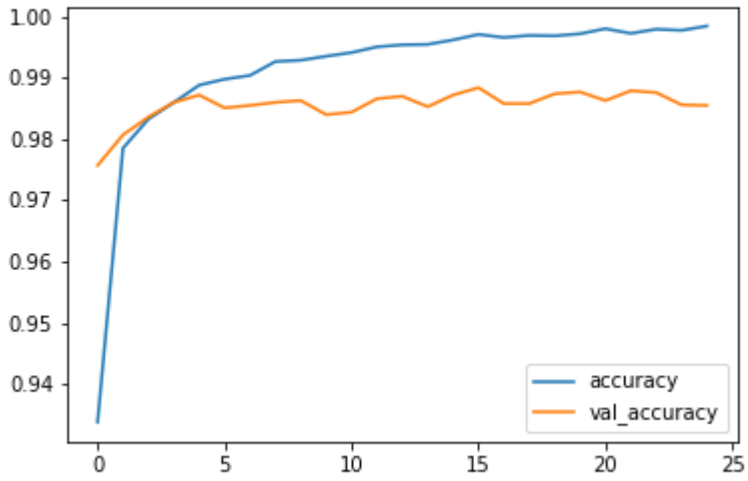
Tensorflow est utilisé avec Keras.

Avec un optimiseur de type Adam, une fonction de perte ‘parse_categorical_crossentropy’, des seuils de déclenchements de types sigmoïde puis Softmax pour la couche de sortie, les accuracies sont en moyenne de 0.988.

Une couche de convolution donne de meilleurs résultats que sans.

Les courbes d’apprentissage démontrent que le nombre d’epochs optimal pour ce modèle est de 3, au-delà l’apprentissage est en over-fitting :

(accuracy en ordonnée, epochs en abscisse)



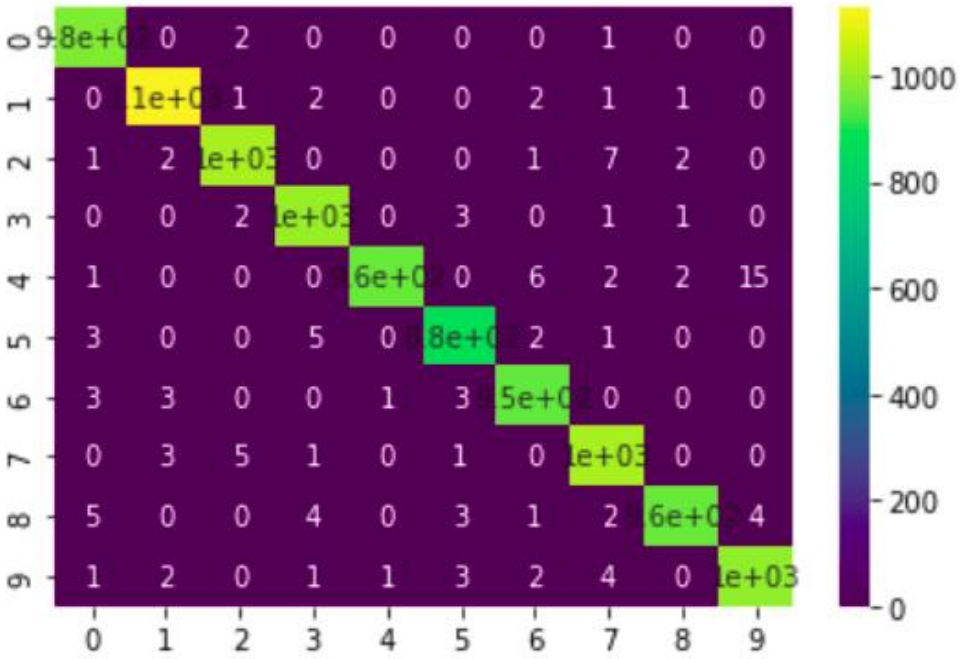
Les stratégies de traitement définies précédemment ne change que de façon insignifiante le score et la matrice de confusion.

Rappel des caractéristiques :

Layer (type)	Output Shape	Param #
conv2d_37 (Conv2D)	(None, 23, 23, 70)	7630
max_pooling2d_30 (MaxPooling	(None, 7, 7, 70)	0
conv2d_38 (Conv2D)	(None, 4, 4, 80)	89680
max_pooling2d_31 (MaxPooling	(None, 2, 2, 80)	0
flatten_6 (Flatten)	(None, 320)	0
dense_6 (Dense)	(None, 10)	3210

Total params: 100,520
Trainable params: 100,520
Non-trainable params: 0

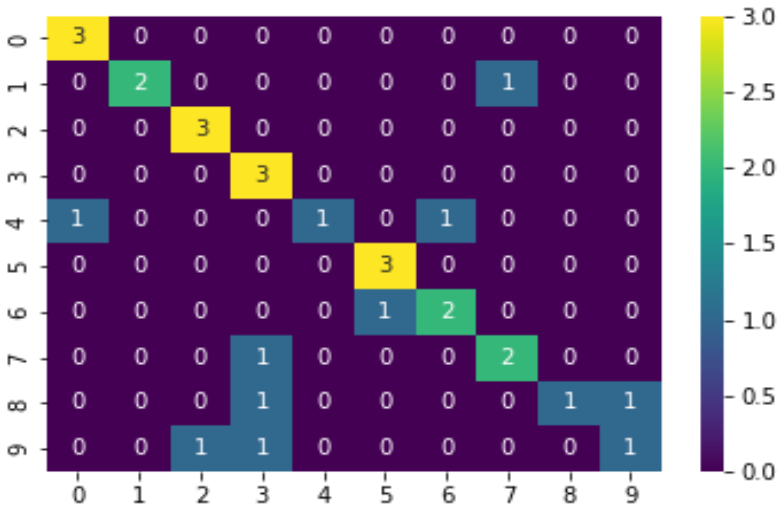
Matrice de Confusion sur les données de test :



4 – Essai dans une application

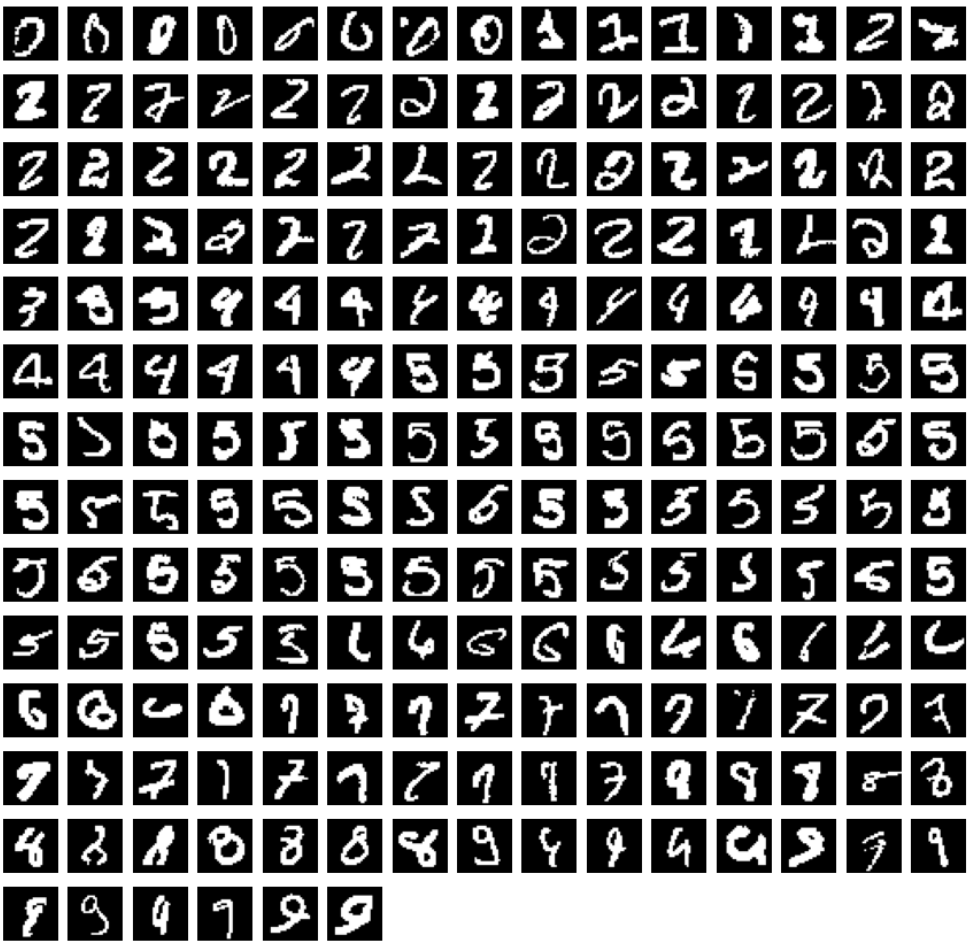
Une série de chiffres est créée avec le logiciel The Gimp. Les images ont une taille de 28 pixels, un fond noir et une écriture en nuance de gris.

Nous ne constatons que 60% de réussite.



5 – visualisation des erreurs

Voici les erreurs de prédictions sur les données de test :



Et sur les données d’évaluation :

Les erreurs d'apprentissage sur les données d'entraînement montrent des chiffres tronqués, illisibles : mal labélisés.

Un tri succin (environ 500 photos écartées), ne montre pas d'amélioration en refaisant l'entraînement du jeu de données.

Les photos des erreurs sont regroupées dans l'image 'erreur_train.png' sur github.

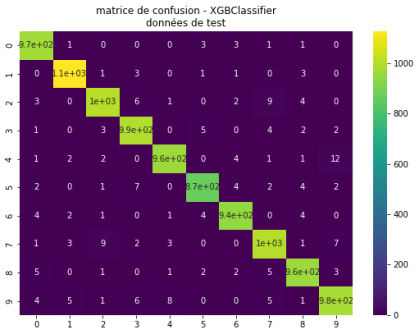
6 – Comparaisons

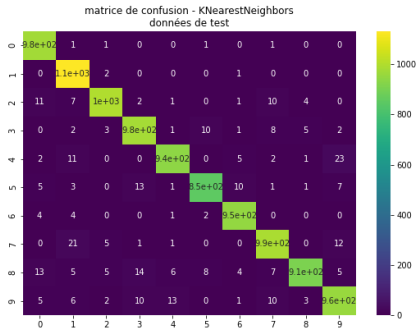
Par curiosité, une comparaison est faite avec trois différents algorithmes (sans recherche d'hyperparamètres pour ne pas augmenter de façon exponentielle le temps de calcul) de Scikit-learn :

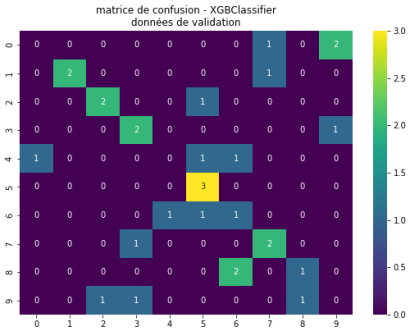
- Algorithme de KNearestNeighbors
- Ensemble learning boosting avec XGBClassifier
- Un réseau de neurone : MLPClassifier

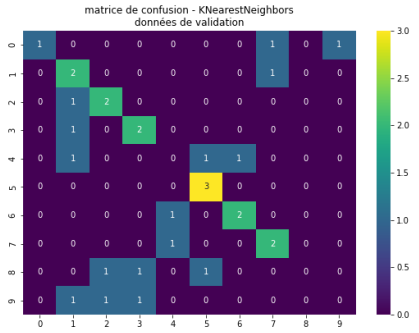
Les images sont bien sûr mises en Dataframe, grâce à la fonction ravel de Numpy.

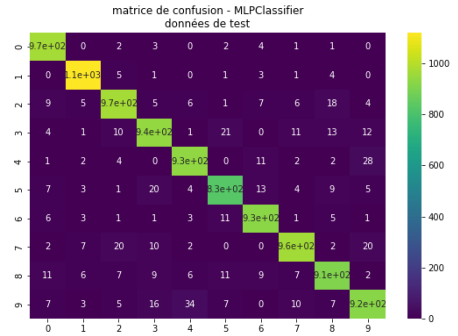
Les accuracies dépassent les 0.90%, mais les matrices de confusion sont moins concluantes :

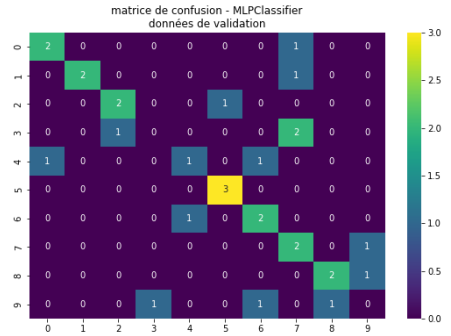

matrice de confusion - XGBClassifier
données de test


matrice de confusion - KNearestNeighbors
données de test


matrice de confusion - XGBClassifier
données de validation


matrice de confusion - KNearestNeighbors
données de validation


matrice de confusion - MLPClassifier
données de test


matrice de confusion - MLPClassifier
données de validation

7 - Conclusion

Les meilleurs scores, les meilleures prédictions, sont obtenues par le réseau de neurones CNN avec Tenserflow et Keras.

Les prédictions sont très mauvaises (0.60%).

Il est possible qu'il manque quelques hyperparamètres au modèle.

Néanmoins, il y a clairement des données mal étiquetées dans le jeu de données. Dans le cadre de cette étude, nous ne passerons pas en revue les 60000 données du jeu d'entraînement. Il est probable qu'un étiquetage plus minutieux améliorerait les prédictions.