HAI918I - Image, Sécurité et Deep Learning
TP5

Ingo Diab

2023

Table des matières

1 Introduction

Les convolutional neural networks (CNN) constituent un type de réseau de neurones composés de deux parties, la partie feature extraction et la partie réseau de neurones classique.

La première partie est un ensemble de couches chacune composée de convolutions (avec un certain nombre de filtres), d'une fonction d'activation permettant au réseau de décrire des phénomènes complexes ainsi que d'un pooling. Cette partie sert à réduire la taille de l'image en la sous échantillonnant (pooling) et à mettre en évidence certaines caractéristiques de l'image (convolutions avec des filtres).

Une fois l'image passée dans la partie "feature extractor" nous obtenons de petites images que nous devons transformer en vecteur et concaténer afin de former le vecteur caractéristique (couche de Flattening). Ce vecteur sert de couche d'entrée à notre réseau de neurones classique (la couche d'entrée étant composée d'un neurone par valeur du vecteur caractéristique). Ces valeurs seront ensuite propagées dans le réseau de neurones à travers toutes les couches internes où chaque neurone aura comme valeur la somme pondérée des neurones de la couche précédente à laquelle on applique une fonction d'activation.

Le résultat de la prédiction se fera dans la dernière couche où nous avons autant de neurones qu'il y a de classes dans notre modèle (sauf si nous faisons de la classification et que nous n'avons que 2 classes, ici un seul neurone peut suffire).

Lors de l'entraînement en apprentissage supervisé, les poids des filtres de convolutions ainsi que des liaisons entre les neurones sont initialisés aléatoirement. La première image du dataset à servir pour l'entraînement aura donc une prédiction aléatoire. Après avoir propagé l'image jusqu'à la couche de sortie (forward propagation), nous faisons remonter la différence entre la prédiction et le résultat attendu dans le sens inverse (backward propagation) afin de renforcer/corriger les poids.

Les CNN peuvent être utilisés afin de générer des images, identifier des entités mais dans ce tp nous utiliserons ce concept dans le but de classifier des images.

2 Matériel et méthode

Ce modèle à pour but de classifier des chiffres manuscrits en 10 classes possibles (de 0 à 9). Pour entraîner notre modèle, nous avons utilisé le dataset MNIST disponible dans la liste de datasets de Keras. Ce dataset est composé

d'images 28x28 de chiffres noirs manuscrits sur fond blanc, notre modèle prendra donc en entrée des images 28x28.

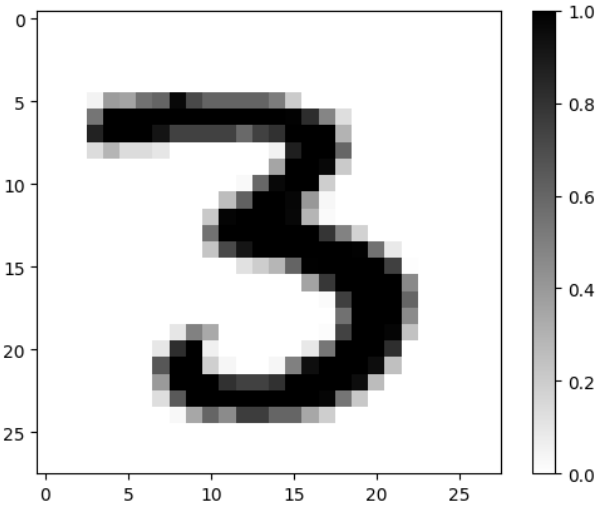

FIGURE 1 – Exemple d'image du dataset MNIST

Avant d'utiliser notre dataset, nous devons normaliser les images en divisant chaque pixel par 255 afin de n'avoir que des valeurs entre 0 et 1. Ceci évite que les pixels blancs (valeur à 255) aient une influence beaucoup plus importante sur l'apprentissage que les pixels noirs (valeur à 0).

La partie feature extraction de mon modèle est composée de 3 couches :

1. Une convolution Conv2D avec 2 filtres de taille 3x3, une fonction d'activation sigmoid et un maxpooling 2x2.
2. Une convolution Conv2D avec 8 filtres de taille 3x3, une fonction d'activation sigmoid et un maxpooling 2x2.
3. Une convolution Conv2D avec 32 filtres de taille 3x3, une fonction d'activation sigmoid et un maxpooling 2x2.

Les autres paramètres tels que le pas, le padding, etc. sont laissés par défaut afin de garder un modèle simple.

La partie réseau classique de mon modèle est composée de 3 couches :

1. Un flatten afin d'obtenir le vecteur caractéristique.
2. Une couche de neurone interne composée de 100 neurones ainsi que d'une fonction d'activation sigmoid.
3. Une couche de sortie composée de 10 neurones (car 10 classes différentes) et d'une fonction d'activation softmax afin d'avoir des résultats entre 0 et 1.

Notre modèle est ensuite compilé avec l'optimizer Adam, la fonction de loss est 'sparse_categorical_crossentropy' puisque nous avons plus de 2 classes et

nous nous intéressons à l'accuracy de notre modèle.

L'entraînement se fait avec un batch de taille 32 (nous prenons les images par groupe de 32) et nous faisons 20 epochs (20 fois 1 forward propagation + 1 backward propagation).

3 Résultats et discussion

Notre modèle nous donne des résultats très satisfaisant.

Ici nous avons l'évolution de l'accuracy de notre modèle pendant l'entraînement. En bleu, l'accuracy sur la partie 'train' du dataset et en orange, l'accuracy sur la partie 'validation' du dataset (partie qui n'a pas été utilisée à l'entraînement). Nous pouvons constater deux choses. La train accuracy est très semblable à la validation accuracy ce qui montre qu'il n'y pas de sur/sous apprentissage de la part du modèle. Nous pouvons aussi constater que les deux courbes croissent ce qui montre que notre modèle est de plus en plus performant lors de l'entraînement.

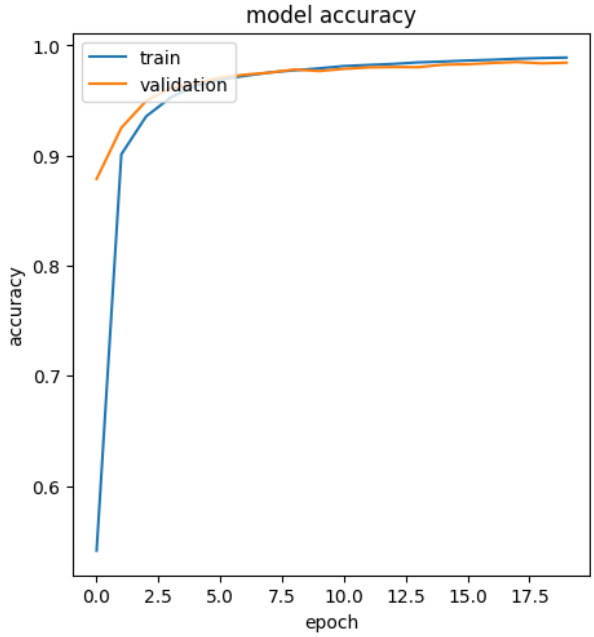

FIGURE 2 – Accuracy obtenue

La loss est inversement proportionnelle à l'accuracy : plus l'entraînement avance, plus la loss est faible, que ce soit sur la partie train ou sur la partie validation. Notre modèle fait donc de moins en moins d'erreurs.

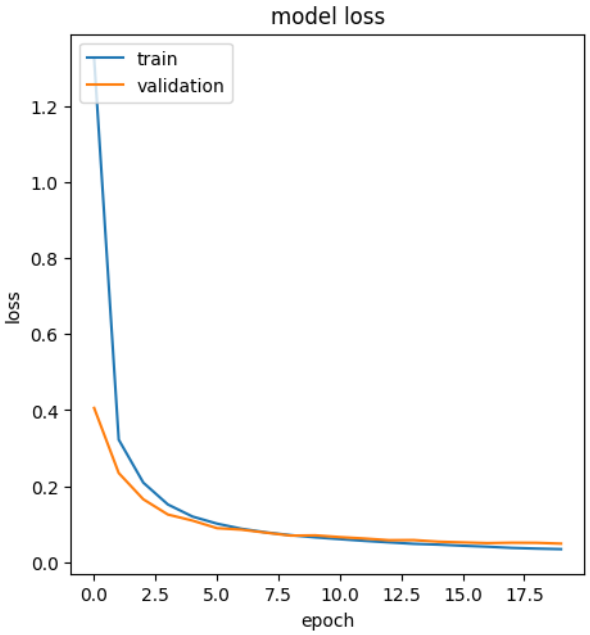

FIGURE 3 – Loss obtenue

L'entraînement de notre modèle se conclut avec une accuracy de 98.4%.

```
Epoch 20/20
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0345 - accuracy: 0.9890 - val_loss: 0.0490 - val_accuracy: 0.9844
```

FIGURE 4 – Dernière epoch

Lorsque nous effectuons des prédictions sur notre jeu de validation, nous obtenons cette matrice de confusion. Nous voyons que la quasi totalité des images ont été prédit avec le bon label, ce qui confirme l'accuracy de notre modèle.

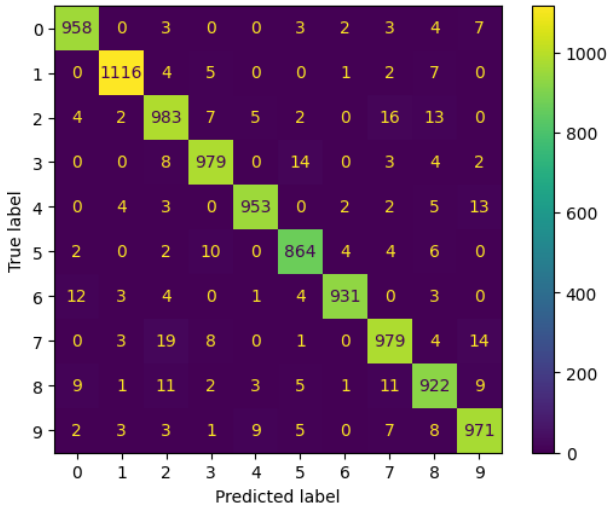

FIGURE 5 – Matrice de confusion

Pour tester notre modèle avec une image qui n'appartient pas au dataset, j'ai créé une image 28x28 sur GIMP.

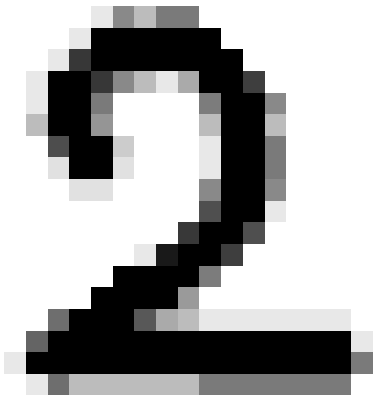

FIGURE 6 – Image crée sur GIMP

Après prédiction du modèle, nous obtenons ce résultat qui est le bon.

FIGURE 7 – Prédiction du modèle

4 Conclusion

Pour conclure ce TP, nous avons vu comment était créé un modèle de CNN utilisant l'apprentissage supervisé pour classifier des images. Nous avons eu l'occasion d'expérimenter l'entraînement ainsi que le test d'un modèle permettant de classifier les chiffres manuscrits. Notre modèle possède une très bonne accuracy mais qui peut encore être augmentée. Nous pourrions expérimenter d'autres modèles (plus/moins de couches de convolutions avec plus/moins de filtres, plus/moins de couches de neurones avec plus/moins de neurones, changer les fonctions d'activation ou l'optimizer, etc.). Nous pourrions aussi nous intéresser au Transfer Learning permettant de baser notre modèle sur un ap-

prentissage plus complet déjà réalisé ou afficher les features maps afin d'essayer de comprendre ce qui ressort des filtres et sur quoi ce base notre modèle actuel.

Concernant le TP et le CM, je les ai trouvé intéressant et constituent un bon moyen de commencer à prendre en main les CNN.