Compte Rendu Tp Indexation

Dû le 19 septembre 2023

Ammar Mohamed Wassim

Table des matières

1 Introduction

Dans ce compte rendu, nous examinons l'utilisation d'un réseau de neurones convolutionnels pour classifier des images de la base de données MNIST. Nous présentons la méthodologie de construction et d'entraînement du CNN, puis nous évaluons sa performance à l'aide de diverses métriques telles que la précision, la perte, la matrice de confusion et le rapport de classification. Ce compte rendu détaillé permettra de comprendre comment un CNN peut être utilisé efficacement pour la classification d'images, en utilisant des mesures objectives pour évaluer sa qualité.

2 Implémentation

2.1 Importation des bibliothèques

```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist
from sklearn.metrics import confusion_matrix,classification_report
import scikitplot as skplt
```

Figure 1 – importation des bibliotheques

```
from keras.models import Sequential,Model

from tensorflow.keras.layers import
 Input, InputLayer,Reshape,Conv2D,MaxPooling2D,Dense, Flatten

from tensorflow.keras.layers import Dense, Dropout, Activation
```

Figure 2 – importation des bibliotheques

2.2 Chargement de la base de données MNIST

```python
def load_data(path):
    with np.load(path) as f:
        x_train, y_train = f['x_train'], f['y_train']
        x_test, y_test = f['x_test'], f['y_test']
        return (x_train, y_train), (x_test, y_test)
(X_train, y_train), (X_test, y_test) = load_data('mnist.npz')
```

FIGURE 3 – Chargement de la base de données

2.3 Affichage de quelque image avec leurs labels

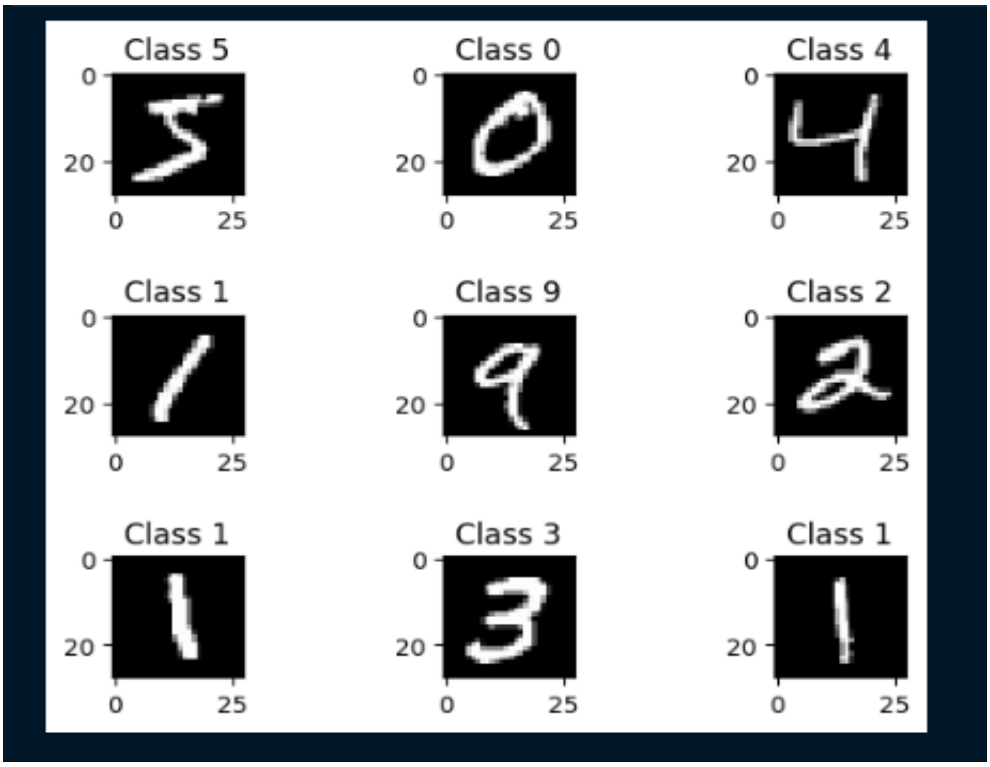

FIGURE 4 – Affichage de quelque image avec leurs labels

2.4 Préparation des données pour l'apprentissage

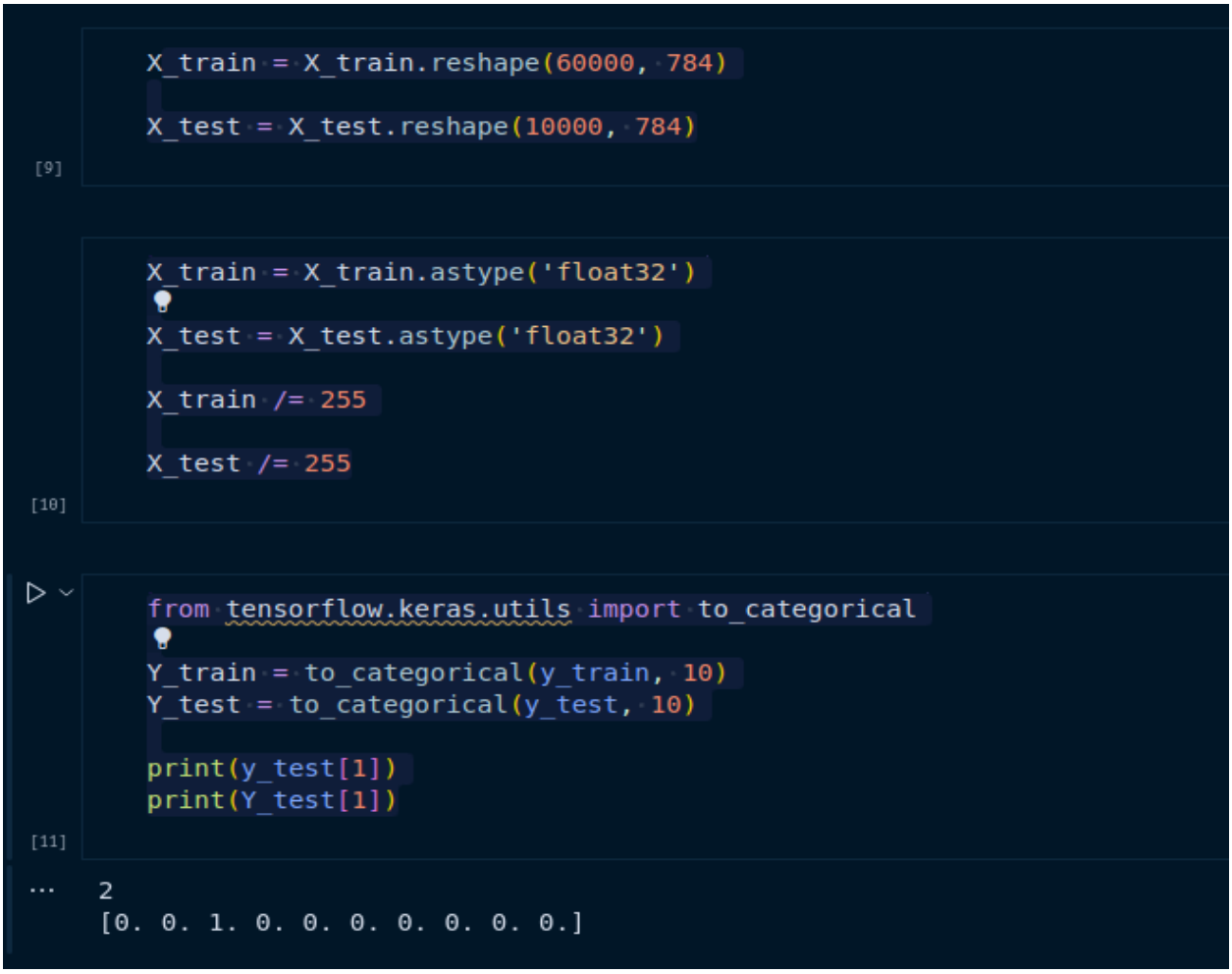

FIGURE 5 – Préparation des données pour l'apprentissage

2.5 Création du réseau de neurones

```
[19]  mode12.summary()

...   Model: "model_1"
      _________________________________________________________________
       Layer (type)                Output Shape              Param #
      =================================================================
       input_4 (InputLayer)        [(None, 784)]             0

       reshape_3 (Reshape)         (None, 28, 28, 1)         0

       layer_conv1 (Conv2D)        (None, 28, 28, 16)        416

       max_pooling2d_6 (MaxPoolin  (None, 14, 14, 16)        0
       g2D)

       layer_conv2 (Conv2D)        (None, 14, 14, 36)        14436

       max_pooling2d_7 (MaxPoolin  (None, 7, 7, 36)          0
       g2D)

       flatten_3 (Flatten)         (None, 1764)              0

       dense_6 (Dense)             (None, 128)               225920

       dense_7 (Dense)             (None, 10)                1290

      =================================================================
      Total params: 242062 (945.55 KB)
      Trainable params: 242062 (945.55 KB)
      Non-trainable params: 0 (0.00 Byte)
      _________________________________________________________________
```

Figure 6 – Création du réseau de neurones

2.6 Apprentissage

```
[20]
model.compile(loss='categorical_crossentropy',metrics=['accuracy'],
optimizer='adam')

[21]
model.fit(X_train, Y_train,batch_size=128, epochs=10,verbose=1)

2023-09-17 13:05:56.381873: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 188160000 exceeds 10% of free system memory.
Epoch 1/10
  3/469 [..............................] - ETA: 20s - loss: 2.2773 - accuracy: 0.0885
2023-09-17 13:05:57.556755: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 24774400 exceeds 10% of free system memory.
2023-09-17 13:05:57.556793: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 24774400 exceeds 10% of free system memory.
2023-09-17 13:05:57.595369: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 24774400 exceeds 10% of free system memory.
2023-09-17 13:05:57.595491: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 24774400 exceeds 10% of free system memory.
469/469 [==============================] - 20s 42ms/step - loss: 0.2081 - accuracy: 0.9377
Epoch 2/10
469/469 [==============================] - 19s 41ms/step - loss: 0.0557 - accuracy: 0.9827
Epoch 3/10
469/469 [==============================] - 19s 40ms/step - loss: 0.0396 - accuracy: 0.9873
Epoch 4/10
469/469 [==============================] - 19s 40ms/step - loss: 0.0300 - accuracy: 0.9905
Epoch 5/10
469/469 [==============================] - 19s 40ms/step - loss: 0.0234 - accuracy: 0.9926
Epoch 6/10
469/469 [==============================] - 19s 41ms/step - loss: 0.0193 - accuracy: 0.9937
Epoch 7/10
469/469 [==============================] - 20s 42ms/step - loss: 0.0149 - accuracy: 0.9951
Epoch 8/10
469/469 [==============================] - 20s 42ms/step - loss: 0.0122 - accuracy: 0.9962
Epoch 9/10
469/469 [==============================] - 20s 42ms/step - loss: 0.0096 - accuracy: 0.9970
Epoch 10/10
469/469 [==============================] - 21s 44ms/step - loss: 0.0094 - accuracy: 0.9966

<keras.src.callbacks.History at 0x7fa0d0e19210>
```

Figure 7 – Apprentissage

2.7 Evaluation

```
[22]
score = model.evaluate(X_test, Y_test, verbose=1)

for name, value in zip(model.metrics_names, score):
    print(name, value)

313/313 [==============================] - 1s 3ms/step - loss: 0.0341 - accuracy: 0.9902
loss 0.03408381715416908
accuracy 0.9901999831199646
```

Figure 8 – Evaluation

2.8 Affichage de quelque image avec leurs labels et les prédictions

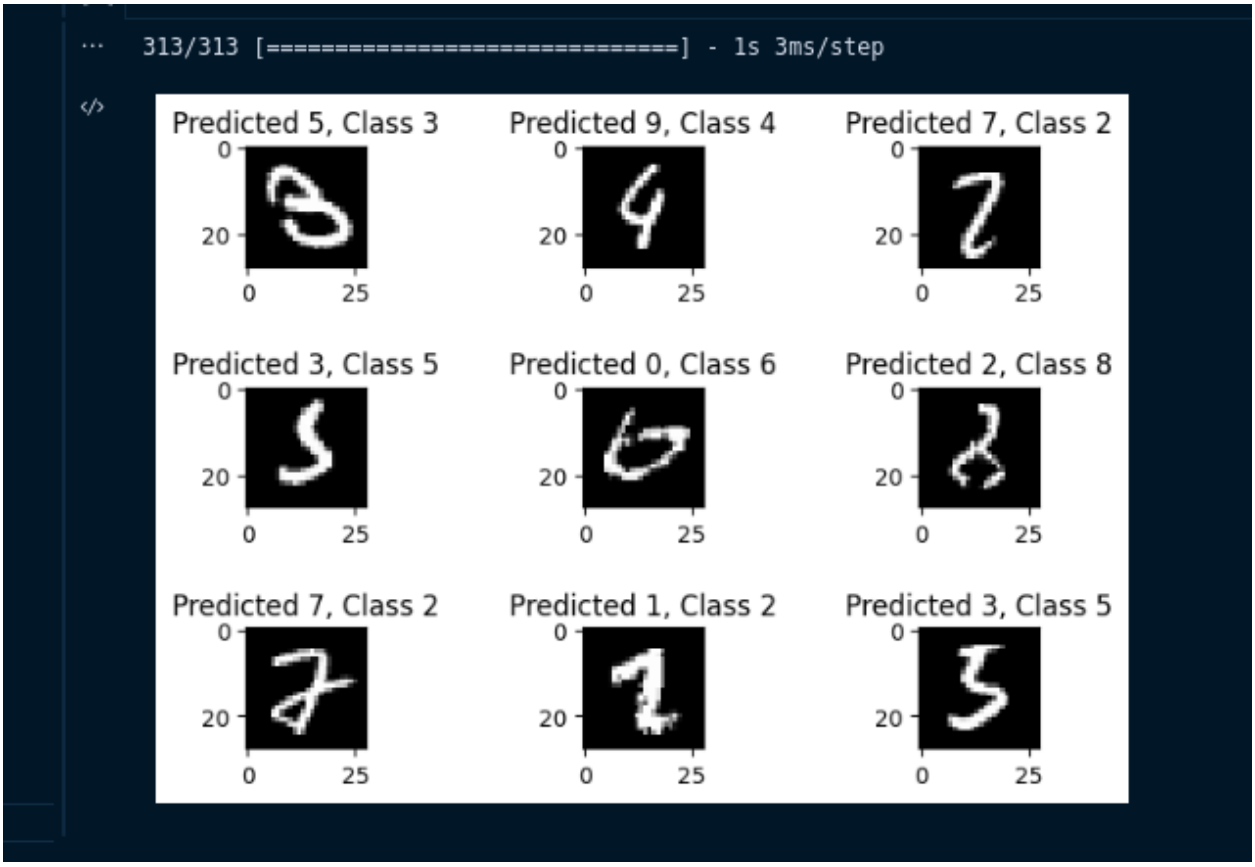

FIGURE 9 – Affichage de quelque image avec leurs labels et les prédictions

2.9 Affichage de la matrice de confusion

```
skplt.metrics.plot_confusion_matrix(y_test, predicted_classes,
normalize=False)

plt.show()
```

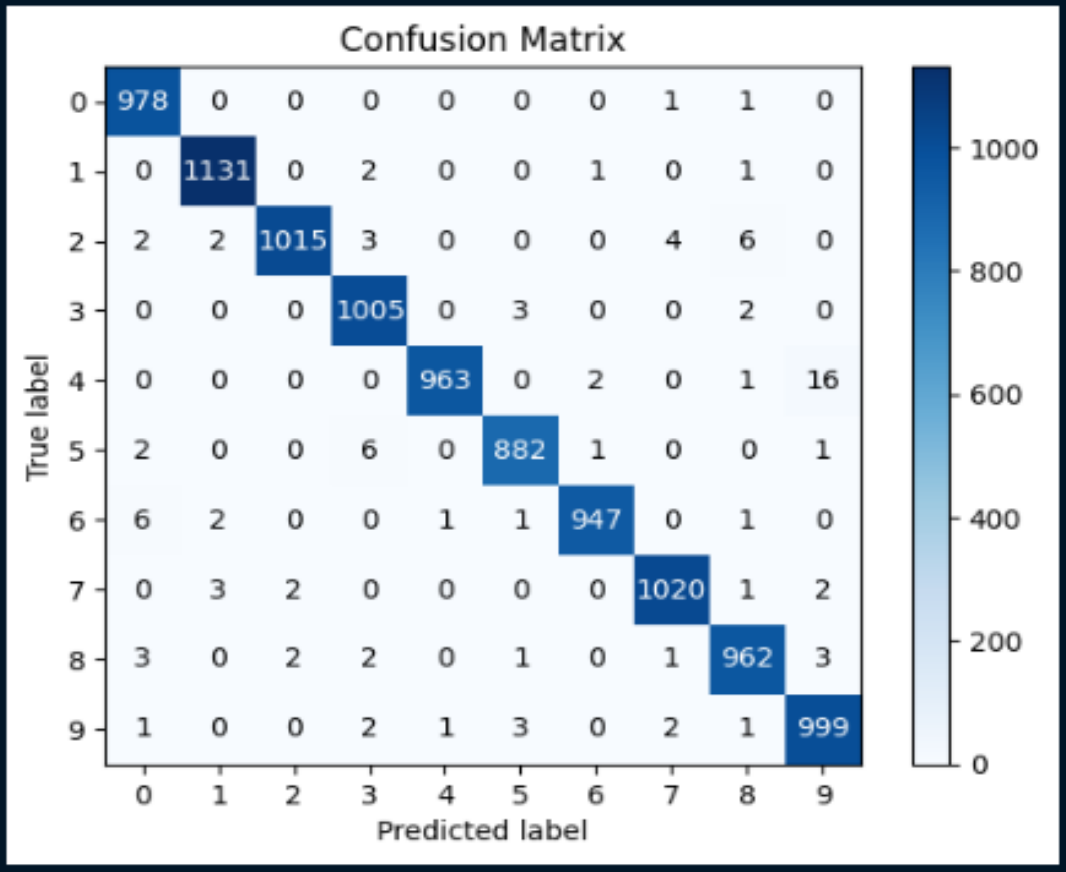

FIGURE 10 – Affichage de la matrice de confusion

2.10 Affichage du rapport de classification

```
print (classification_report(y_test, predicted_classes, target_names=None))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	980
1	0.99	1.00	1.00	1135
2	1.00	0.98	0.99	1032
3	0.99	1.00	0.99	1010
4	1.00	0.98	0.99	982
5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958
7	0.99	0.99	0.99	1028
8	0.99	0.99	0.99	974
9	0.98	0.99	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

FIGURE 11 – Affichage du rapport de classification

3 Conclusion

En conclusion, ce compte rendu a exploré l'utilisation d'un réseau de neurones convolutionnels (CNN) pour la classification d'images de la base de données MNIST. Les métriques d'évaluation telles que la précision, la perte, la matrice de confusion et le rapport de classification ont été utilisées pour évaluer la performance du modèle. Les résultats ont démontré l'efficacité du CNN dans la classification de chiffres manuscrits, ce qui en fait un outil précieux pour de nombreuses applications de vision par ordinateur. Ce tutoriel a ainsi permis de mieux comprendre comment utiliser un CNN pour résoudre des tâches de classification d'images.