Rapport projet 12 : Classification manuscrite

L'objectif de ce projet d'appréhender le *deep learning* (à travers Tensorflow) et de progresser dans la manipulation d'images et la reconnaissance manuscrite.

Ce projet repose sur la data MNIST (**Mixed Mixed National Institute of Standards and Technology).** Cette dernière est composée d'images représentant des chiffres écrits à la main. Le dataset original regroupe 60000 images d'apprentissage et 10000 images de test, issues d'une base de données antérieure, appelée simplement NIST.

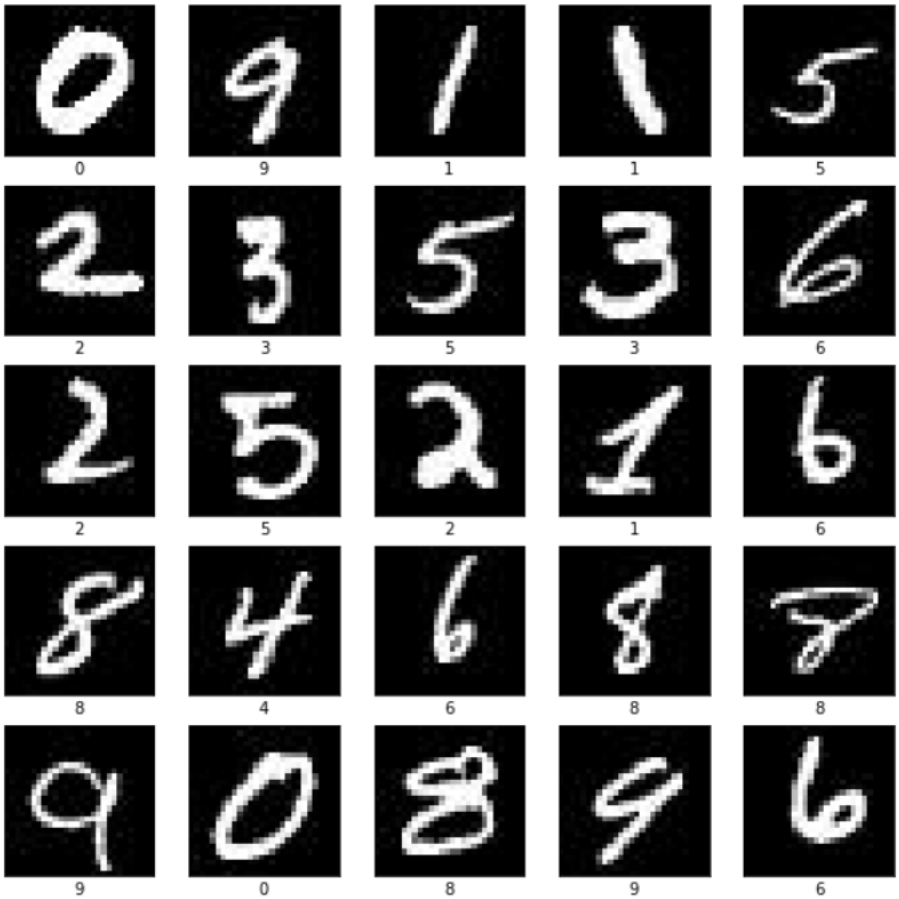
Ce dataset s'importe en une seule ligne de code : « `from keras.datasets import mnist` »

Dans notre cas, nous avons téléchargés à part les images dans deux dossiers différents : train et test.

Il faut noter que les images sont en noir et blanc, normalisées centrées de 28 pixels de côté.

Exemple d'images :

Voici le type d'images qui composent notre dataset.



Importation des données :

Données Train :

```
from tqdm import tqdm

directory = ".\\data\\MNIST_Dataset_JPG_format\\MNIST_JPG_training\\"

X_train = []
# liste pour enregistrer les labels
y_train = []
# l va servir à numéroter les labels : dans la boucle, l s'incrémente de 1 à chaque passage
l = -1


for x in tqdm(os.listdir(directory)):
    # incrémentation de l
    l = l +1
    for y in os.listdir(directory + x +"\\"):
        image = cv2.imread(directory + x +"\\"+y)
        X_train.append(image)
        y_train.append(l)
```

```
100%|██████████| 10/10 [01:03<00:00,  6.32s/it]
```

Données Test :

```
directory = ".\\data\\MNIST_Dataset_JPG_format\\MNIST_JPG_testing\\"

X_test = []
# liste pour enregistrer les labels
y_test = []
# l va servir à numéroter les labels : dans la boucle, l s'incrémente de 1 à chaque passage
l = -1


for x in tqdm(os.listdir(directory)):
    # incrémentation de l
    l = l +1
    for y in os.listdir(directory + x +"\\"):
        image = cv2.imread(directory + x +"\\"+y)
        X_test.append(image)
        y_test.append(l)
```

```
100%|██████████| 10/10 [00:11<00:00,  1.12s/it]
```

Architecture CNN sur Tensorflow

L'objectif est d'ici de créer notre propre réseau de neurones à convolution à l'aide de la bibliothèque Tensorflow développée par Google.

Plus d'information sur : https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8

Une fois notre modèle sélectionner puis entraîné, nous évaluons ses résultats avant de le sauvegarder. Cela nous permettra de le réutiliser par la suite.

Sauvegarde du modèle

```
filepath='./modelCNN'
tf.keras.models.save_model(model, filepath, overwrite=True,
                           include_optimizer=True, save_format=None,
                           signatures=None, options=None, save_traces=True)
loaded_model = tf.keras.models.load_model(filepath)
```

```
INFO:tensorflow:Assets written to: ./modelCNN\assets
```

Test de l'application

Pour tester notre modèle nous avons utilisé des images qu'il n'avait jamais vues. Pour cela nous sommes allés sur Kaggle récupérer un échantillon de test.

Nous sélectionnons à chaque fois une image aléatoire dans notre dossier que l'on présente au modèle.

```
# Chemin des images
chemin = "./test_pictures/"
liste_test = os.listdir(chemin)


# on choisi une image aléatoire à chaque fois
rand = random.choice(liste_test)

# on décide du nouveau chemin de l'élément aléatoire
route = "./test_pictures/" + rand

# on affiche l'élément aléatoire avec la variable route
image = cv2.imread(route)

plt.figure(figsize=(2,2))
plt.subplot()
plt.xticks([])
plt.yticks([])
plt.imshow(image)
plt.show()
```

Une fois que le modèle a étudié l'image, il nous renvoie une prédiction :

```python
# Prédiction
filepath = './modelCNN'
liste_label = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
loaded_model = tf.keras.models.load_model(filepath)
prediction = loaded_model.predict(img)

resultat = liste_label[np.argmax(prediction)]
print('Prédiction :',resultat)
```

Application fichier.py

```
%run test.py
```

```
Prédiction : 8
```