Feuille 3. : Tensorflow, Keras, CNN

1 Préambule

Nous avons vu comment créer un premier réseau de neurones simple afin de reconnaître des chiffres manuscrits. La précision de notre réseau simple est de `98.11%`

Dans de TD, nous allons construire un réseau CNN. Notre réseau contiendra des couches de convolution, des couches de pooling et des couches de dropout.

2 CNN avec keras

Nous allons utiliser le dataset `mnist` pour apprendre à reconnaître tous les nombres de 0 à 9. Pour cela, il faut suivre les étapes suivantes. Les explications du code sont données au tableau :

1. On commence par importer les données du dataset :

```
from keras.datasets import mnist
```

2. On importe les fonctions et les classes dont on aura besoin :

```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
K.set_image_dim_ordering('th')
```

3. On initialise le générateur de nombres aléatoires avec la même graine :

```
seed = 7
numpy.random.seed(seed)
```

4. A présent, on charge les données :

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
```

5. On normalise les valeurs et on transforme les chiffres de sortie en données catégorielles :

```
X_train = X_train / 255
X_test = X_test / 255

y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
%X_train = X_train /255
%X_test = X_test /255
```

6. A présent, nous allons définir notre modèle, i.e., un CNN :

```
def cnn_model():
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(1, 28, 28), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
return model
```

7. Puis de le construire et de l'évaluer :

```
cnn = cnn_model()
cnn.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbo
scores = cnn.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

Remarque. Vous pouvez "jouer" sur le nombre epoch si le temps d'entrainement est trop important sur votre machine. Observez la qualité de votre modèle en fonction de ce nombre.

3 A vous de jouer : Un réseau plus complexe

Définissez un modèle avec les composants suivants et testez-le :

1. une convolution à 30 filtres avec des feature maps de taille 5x5
2. un maxpooling de taille 2x2
3. une convolution à 15 features avec des feature maps de taille 3x3
4. un maxpooling de taille 2x2
5. un dropout avec $p = 20\%$
6. une couche Flatten
7. un réseau complètement connecté avec 128 neurones et une fonction d'activation relu
8. un réseau complètement connecté avec 50 neurones et une fonction d'activation relu
9. une couche de sortie.