IA - Réseaux de neurones
TP3 : Découverte de Keras

Romain Marie

2020-2021

3iL INGENIEURS
</les sciences informatiques>

Dans ce TP, l'objectif va être de prendre de la hauteur par rapport aux séances précédentes, en se concentrant non plus sur les détails d'implémentation d'un réseau de neurones, mais plutôt sur sa mise en pratique, et l'étude de ses performances. Nous utiliserons pour cela la bibliothèque Python **Keras**, que vous avez normalement déjà installée.

Un compte rendu (noté) vous sera demandé pour ce TP. Essayez donc de répondre aux questions au fil de l'eau (pendant les phases d'apprentissage par exemple), et pas d'une traite à la fin. Les questions nécessitant une réponse à développer dans le compte rendu apparaissent avec une *

1 Mise en place

1. Commencez par lancer **Anaconda prompt**, en passant par la barre de recherche de Windows si vous n'avez pas créé de raccourci
2. Activez le contexte TP3iL grâce à la commande **activate TP3iL**.
3. Lancez l'IDE Spyder grâce à la commande **spyder**.

2 MNIST, le retour

Au risque de vous écœurer définitivement des chiffres, vous allez reprendre le problème du TP précédent, mais cette fois sur Keras, et avec plusieurs structures de réseaux différentes. Ce choix est motivé par les raisons suivantes :

— la banque d'images MNIST est directement incluse dans Keras, ce qui facilite GRANDEMENT son utilisation

— Le problème de reconnaissance de chiffres est suffisamment modeste pour être entrainable sur une séance de TP

1. Chargez dans Spyder le fichier MnistTP.py qui vous a été fourni avec le sujet. Il définit puis entraine le réseau de neurones vu lors de la dernière séance.
2. * Après avoir pris quelques minutes pour étudier le fichier, et sans avoir peur de l'exécuter, expliquez en quelques mots l'utilité, les paramètres et le résultat des lignes suivantes :

(a) Y_train = np_utils.to_categorical(Y_train,10)

(b) modele.add(Flatten(input_shape=(28,28,1)))

(c) modele.add(Dense(32,activation='sigmoid'))

(d) modele.summary()

(e) h1 = modele.fit(X_train, Y_train, batch_size=32,epochs=10, verbose=1,validation_data=(X_test,Y_test))

(f) modele.save('modele_TP2.h5') ;

3. * A partir de l'organisation des neurones et de la taille des images en entrée (28x28), expliquez pourquoi cette architecture de réseau comporte 25450 paramètres à régler.
4. * Après avoir exécuté le script, expliquez en quelques mots à quoi correspondent les courbes orange et bleues, et discutez des performances du réseau de neurones sur les images test.
5. Quitte à le ré-exécuter une fois, notez également dans un coin le temps que dure une epoch de la phase d'apprentissage.

3 Premier CNN

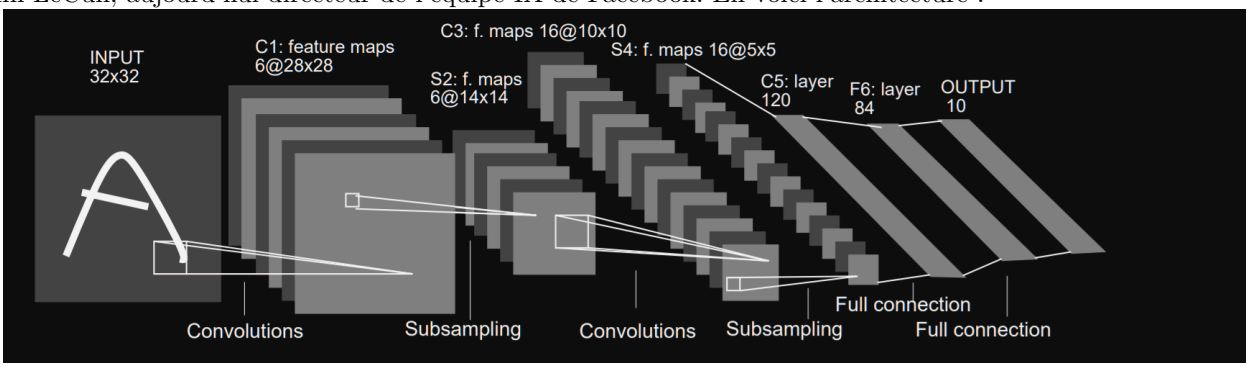
1. Copiez-collez le fichier MnistTP.py, et renommez la copie (MnistTP-avecConv2D.py).
2. Dans ce dernier fichier, modifiez la structure du réseau considéré de sorte qu'il respecte la séquence **entrée -> convolution -> max pooling -> flatten -> dense -> sortie**.
   Sans toucher à ce qui existe déjà, il vous faudra donc ajouter :
   (a) Une couche de 16 noyaux de convolution, sans zéro padding, avec un masque de dimension 3x3, un pas de 1, et une fonction d'activation ReLU. Vous utiliserez pour cela la syntaxe suivante : **model.add(Convolution2D(16,(3,3),activation='relu',input_shape=(28,28,1)**

   (b) Une couche de max pooling, de dimension 2x2. Vous utiliserez pour cela la syntaxe suivante : **model.add(MaxPooling2D(pool_size=(2,2))**

   (c) Modifez également le nom du fichier qui sert à sauvegarder votre modèle, de sorte à ne pas écraser le travail réalisé à la section précédente
3. * Dessinez la structure du réseau de neurones ainsi défini. Vous indiquerez bien sûr la succession de couches du réseau, mais aussi leurs dimensions respectives.
4. * Combien de paramètres sont alors à régler ? Vous justifierez là encore votre raisonnement.
5. * Après avoir entrainé ce nouveau réseau, comparez le avec le premier, en terme de :
   — Temps d'apprentissage
   — Nombre de paramètres
   — Qualité du résultat

4 LeNet

Je vous propose un petit challenge : programmer LeNet, c'est à dire le réseau de neurones proposé par Yann LeCun, aujourd'hui directeur de l'équipe IA de Facebook. En voici l'architecture :



1. * A partir de l'image ci-dessus, identifiez :
   (a) Le nombre de couches du réseau,
   (b) les propriétés de chaque couche
   (c) le nombre de paramètres de chaque couche
2. En utilisant une copie de l'un des fichiers précédents, définissez la structure de ce réseau dans Keras.
3. Après avoir modifié le nom du fichier de sauvegarde du modèle, entrainez ce réseau
4. * Discutez le résultat obtenu par rapport aux deux autres réseaux, en utilisant les mêmes critères que dans la section précédente.

5 Validation expérimentale

A ce stade, vous avez normalement 3 fichiers correspondant à la sauvegarde de vos 3 réseaux de neurones entrainés. L'idée va être maintenant de les utiliser dans une application très simple qui charge un fichier image et identifie le chiffre qui y est inscrit. Pour vous aider, un prototype du script à écrire vous est fourni (testMnist.py).

1. Après l'avoir ouvert, prenez quelques instants pour étudier le contenu (très sommaire) de ce script.
2. Remplacez les ? ? ? par les noms de vos fichiers contenant les modèles.
3. Remplacez les ? ? ? ? ? par une demande de prédiction pour chacun des 3 modèles.
4. Dans paint, créez une image portant le bon nom (cf. code), et dessinez-y le chiffre de votre choix. Vous utiliserez le premier pinceau dans la deuxième taille.
5. Lancez le script testMnist.py. Si tout est correctement codé, vous devriez avoir les 3 prédictions avec les indices de confiance.
6. Puisque ces prédictions sont dans une boucle infinie, vous pouvez modifier votre fichier image, l'enregistrer, et, en appuyant sur entrée dans la console de Spyder, avoir les nouvelles prédictions pour cette nouvelle image. Comparez les performances des 3 réseaux sur plusieurs chiffres différents. N'hésitez pas à les challenger.
7. * Discutez de la cohérence entre les mesures de performances obtenues dans les sections précédentes, et la prédiction réalisée par chaque réseau (sur plusieurs exemples).

6 Pour aller plus loin

1. En créant une nouvelle copie de l'un des fichiers précédents, définissez (arbitrairement mais en réfléchissant un peu) et entrainez votre propre réseau de neurones.
2. Adaptez le fichier testMnist.py pour comparer ses performances aux 3 réseaux précédents.