



RAPPORT TP 4:

CNN sous KERAS

Réalisé par :

BOUJIDA Hafssa

CHARAFI Asmaa

AIT-JILLALI Nouhaila

EL-ASRI Noussaiba

Encadré par:

M. IBN ELHAJ

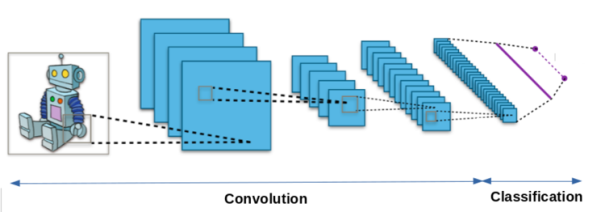
Smart ICT ine2

2020/2021

INTRODUCTION

On appelle réseau neuronal convolutif, ou réseau de neurones à convolution, (CNN pour Convolutional Neural Network) un type de réseau de neurones artificiels utilisé dans la reconnaissance et le traitement des images, et spécialement conçu pour l'analyse des pixels.

Dans les CNN, les « neurones » sont disposés comme ceux du lobe frontal, siège du traitement des stimuli visuels chez l'homme. Les couches de neurones sont organisées de façon à couvrir tout le champ visuel afin d'éviter le problème du traitement d'images fragmentées des réseaux neuronaux classiques.



OBJECTIF

Notre TP est divisé en deux parties :

1-Application du CNN pour la classification de texte

2-Application du CNN à à la base des images CIFAR10

Pour chaque partie nous allons :

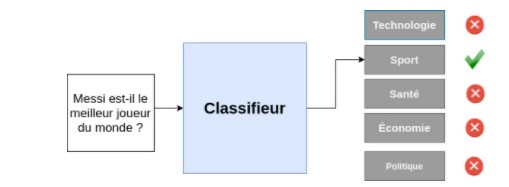
1. Introduire les notions de base.
2. Exécuter le code Python sous Spyder en illustrant les résultats.
3. Faire variez les paramètres du modèle ainsi que des mesures et interprétez les résultats.
4. Conclusion.

**PARTIE 1 : CNN pour la classification de texte**

 Cette partie sera consacrée à la **classification du texte** à l'aide de l'apprentissage automatique sous forme de **réseau de neurones à convolution**.

1. Introduction :

La classification (ou catégorisation) de textes est l’une des tâches de traitement du langage naturel les plus courantes. Elle consiste à associer un texte non-structuré à un tag, qui correspond à une classe bien précise.



Faire de la classification c’est tout simplement mettre une étiquette sur un texte, lui faisant ainsi appartenir à un groupe ou classe bien définie.

Le problème de classification de texte peut être traité à partir de différentes approches, par exemple, en considérant la fréquence d'occurrence des mots dans un texte donné par rapport à l'occurrence de ces mots dans le corpus complet.

D'autre part, il existe d'autres approches où le texte est modélisé comme une séquence de mots ou de caractères, ce type d'approche fait principalement appel à des modèles basés sur des architectures de **réseaux neuronaux récurrents** .

Les CNN que nous allons utiliser appliquent une suite de transformations non linéaires sur les données pour faire émerger des caractéristiques utiles à la classification.

L’algorithme utilisé aura donc besoin d’apprentissage pour en fin de cycle réaliser les classifications en fonction des scores obtenus.

1. Le code python exécuté sur Spyder :



Figure 1 :

Premièrement on a importé les bibliothèques numpy et tensorflow, et puis on a préparé le dataset pour notre model et puis le divisé en trainSet et testSet .

Il est nécessaire d’initialiser « Input Shape » sinon le modèle séquentiel ne va pas être construit.

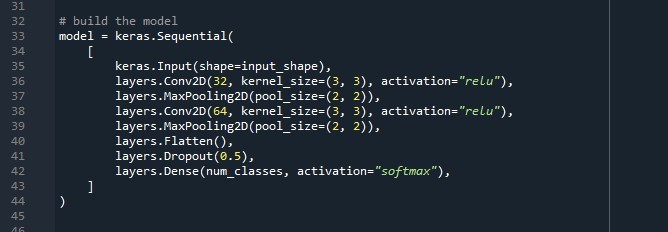


Figure 2 : On a créé un modèle séquentiel en passant une liste de couches ( layers ) au constructeur séquentiel .

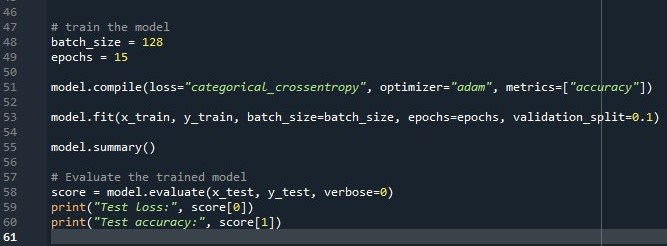
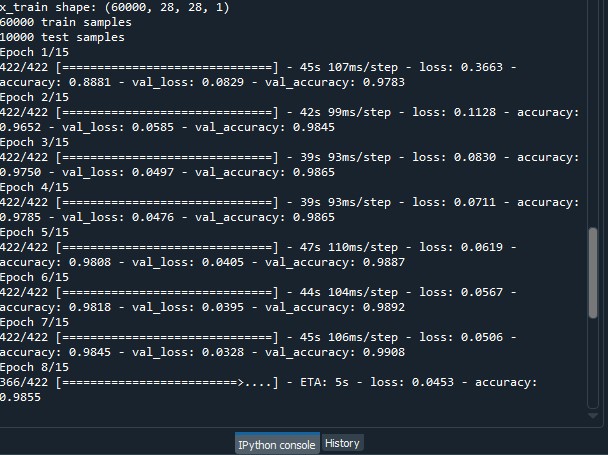


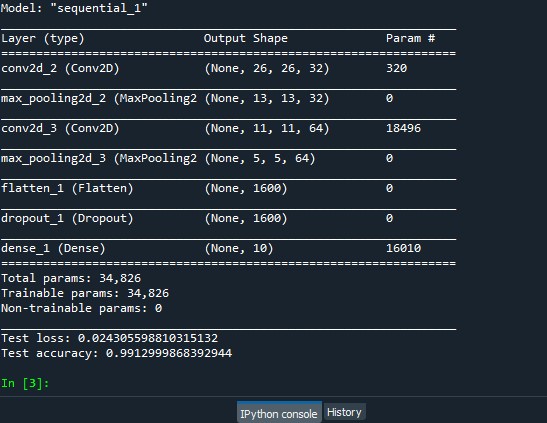
Figure 3 :

On a entraîné le modèle pour évaluer sa précision en fixant le nombre d'époches( qui est l’hyper paramètre qui définit le nombre de fois où l'algorithme d'apprentissage fonctionnera sur le training set) à 15 .

Une fois notre modèle est construit, on a appelé sa méthode summary () pour afficher son contenu.

\* **L’exécution du code donne les résultats suivants :**

****

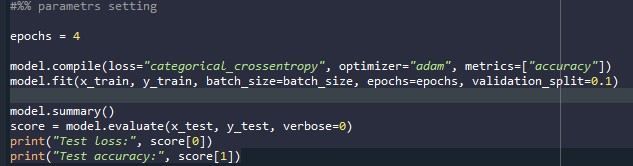
****

**On a donc:**

Test loss: 0.024305598810315132

Test accuracy: 0.9912999868392944

**C -** Variation des paramètres et comparaison des résultats.

maintenant on va varier les paramètres (on va diminuer le nombre des époches à 5)

et puis comparer les résultats :

result epoche 5.jpg

On remarque donc qu’après la diminution du nombre d’epoches, l’accuracy a augmenté unremarquablement mais la valeur loss a diminué par rapport au résultats précédents .

Conclusion :

Avoir une très grande taille d'époque n'améliorera pas nécessairement votre précision. Les tailles d'époque peuvent augmenter la précision jusqu'à une certaine limite au-delà de laquelle vous commencez à sur-ajuster ( over-fitting problem) votre modèle. Avoir un très bas entraînera également un sous-ajustement (under-fitting problem).

**PARTIE 2: CNN appliqué à la base des images CIFAR10**

1. Introduction :

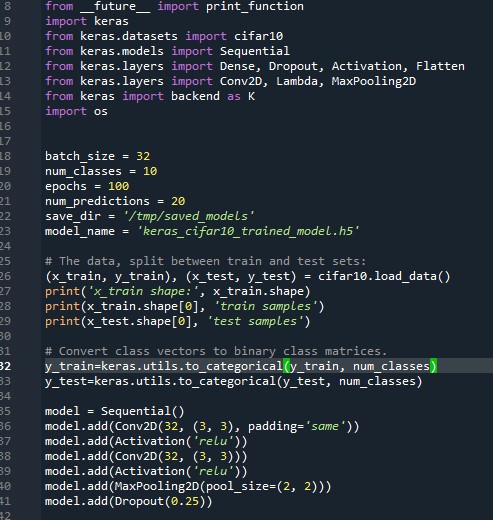
L'ensemble de données CIFAR-10 est une collection d'images couramment utilisées pour former des algorithmes d'apprentissage automatique et de vision par ordinateur. Il s'agit de l'un des ensembles de données les plus utilisés pour la recherche sur l'apprentissage automatique.

Et puisque les CNN sont spécialement conçus pour traiter des images en entrée, ils ont tendance à être les meilleurs à connaitre les images dans CIFAR-10.

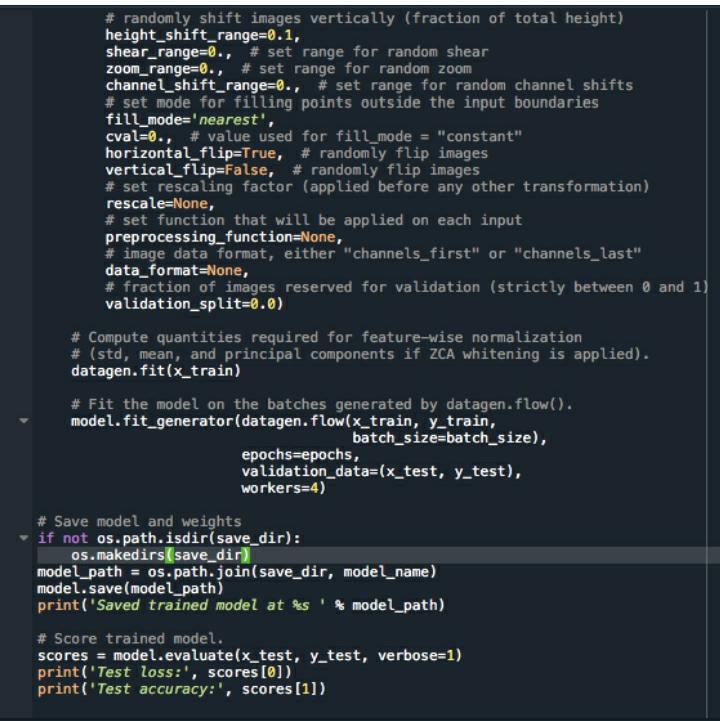
Ce qui est l’objectif de cette partie.

1. Le code python exécuté sur Spyder :

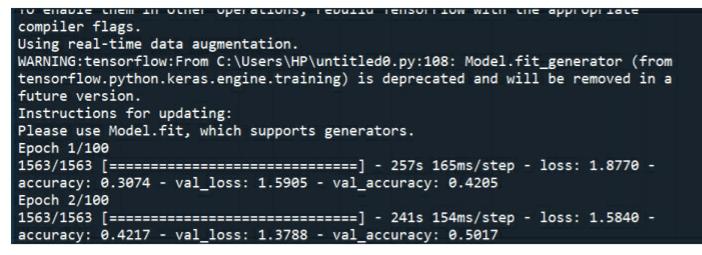
On commence d’abord par l’installation du dataset CIFAR10.







L’exécution du code donne les résultats suivants :



Conclusion : le modèle a donné des valeurs importantes d’accuracy et de loss, mais qui doivent encore être améliorées.

NB : on a rencontré des problèmes lors de l’exécution et c’est pour cette raison

qu’on a pas arrivé à faire variez les paramètres du modèle.