Université Nice Côte d'Azur
Enrico FORMENTI

Année 2020/21
M1 Info
NNL

Examen sur projet

Table des matières

1 Avant de commencer **1**

2 Le projet à réaliser **2**
2.1 Préparation . . . 2
2.2 Entrainement . . . 2
2.3 Prédiction et test . . . 2

3 Le projet et sa notation **2**
3.1 Code (C) . . . 3
3.2 Performances (P) . . . 3
3.3 Rapport (R) . . . 3
3.4 La notation . . . 3

4 Les groupes **3**

5 Ce qu'il faut rendre et quand **3**
5.1 Pénalités de retard et demandes d'extension . . . 3

6 Comment rendre son projet **4**

7 Au secours ! **4**

1 Avant de commencer

Le projet nécessite de travailler dans un environnement adapté. Il faudra notamment installer le librairies suivantes si vous ne l'avez pas déjà fait :

— `NumPy` : normalement vous avez déjà installé cette librairie, veillez simplement à avoir la toute dernière version.
— `TensorFlow` : librairie spécialisée pour l'apprentissage profond (*deep learning*). Vous trouvez les instruction pour l'installation ici. Nous allons surtout utiliser sa bibliothèque Keras qui est spécialement conçue pour le modèle de réseaux que nous allons utiliser. Normalement, Keras est installé directement avec TensorFlow.
— Il faudra aussi suivre cet excellent tutoriel qui explique le modèle CNN (Convolutional Neural Networks) qui conviendra utiliser pour mener à bien le projet. Il s'agit de réseau de neurones multicouche (layers) dans lesquelles chaque layer a un but spécifique. De plus, la réponse de chaque couche s'obtient par convolution d'un opérateur adapté sur l'entrée. Ce tutoriel explique la structure globale d'un CNN et son fonctionnement.
— Ce serait sans doute un atout avoir accès à [1] et lire attentivement les sections : 2.2, 2.3, 2.4, 3.1, 3.2 ainsi que le chapitre 5.

Nous allons très brièvement reprendre les layers les plus communs dans le traitement d'images en disant quelques mots pour chacun et, surtout, en renvoyant sur des tutoriels ou sur les API Keras pour de plus amples explications ou pour le paramétrage. Voici donc une liste de layer que nous pensons utiles par la suite (ordre alphabétique) :

conv2D il s'agit d'un layer qui applique un opérateur de convolution sur une structure 2D (image) pour donner en sortie une structure sur laquelle on appliquera d'autres layers. Vous trouvez ici un bon exemple d'application et ici un joli tutoriel qui explique comment choisir les paramètres.

dense il s'agit d'un layer dans lequel le graphe de connexion est complet. Le mieux c'est de consulter directement l'API Keras ici.

dropout il s'agit d'un layer utilisé à des fins de régularisation. Un tutoriel intéressant sur son utilisation (mais sans code) se trouve ici.

maxpooling2D il s'agit d'un layer qui sert à réduire la dimensionalité de l'entrée en prenant le max sur chaque fenêtre de l'opérateur de convolution qu'on a passé en paramètre. Ici vous trouvez l'API de Keras.

2 Le projet à réaliser

Nous cherchons à réaliser un classificateur d'images à partir d'un data set fourni par Intel. Un classificateur d'images est un réseaux qui prend en entrée une image et cherche à lui attribuer une catégorie parmi un certain nombre de catégories sur lesquelles il a été entrainé.

Dans notre cas, voulons construire un classificateur de photo de scènes naturelles comme ceux qu'on pourrait intégrer à un appareil photo intelligent qui à chaque prise de photo va taguer l'image avec le nom d'une catégorie.

Notre data set prévoit les six catégories suivantes (en anglais comme dans l'original) :

— *buildings*
— *forest*
— *glacier*
— *mountain*
— *sea*
— *street*

Chaque catégorie contient un nombre variable d'images, toutes de taille 150x150, pour un total de plus de 150k éléments. De plus, les images sont déjà regroupées en trois ensembles : *training*, *test* et *prédiction* avec une signification et un usage évident.

La base de donnée se trouve sur Kaggle à cet adresse. Attention : la taille est conséquente (305Mo).

Le projet doit être structuré en trois parties : **préparation**, **entrainement**, **prédiction et test**.

2.1 Préparation

Dans cette phase vous allez devoir les pré-traitements nécessaires au bon fonctionnement des algorithmes d'apprentissage que vous allez utiliser par la suite. Par exemple, il faudra encoder convenablement les noms des catégories ou encore vérifier si les images ne sont pas corrompues (dans ce dernier cas il faudra les éliminer) ou encore s'elles ont la bonne taille.

La dernière étape de la préparation visera à définir le modèle de CNN que vous allez utiliser dans la partie suivante.

2.2 Entrainement

Dans cette partie il s'agira de compiler le modèle (ce serait bien de sauver dans un fichier la structure du modèle ainsi que quelques caractéristiques intéressantes – par exemple le nombre et le type de couches, le nombre de connections par couche et le nombre de connexions globales ou encore le nombre total de neurones utilisés). Puis, de lancer l'entrainement du modèle. Il vous appartient d'expérimenter (et documenter les expériences menées ça sera sûrement un plus pour l'évaluation finale) pour trouver le meilleur paramétrage de votre modèle.

2.3 Prédiction et test

Il s'agira d'écrire un *predictor* adapté au modèle que vous avez entrainé et ensuite évaluer les performances comme nous avons vu en TD. En gros, nous voudrions une commande `predict` qui prend en entrée le nom de fichier d'une image et une chaine de caractères `mode` qui renvoie le nom de la catégorie correspondante à l'image si `mode='category'` ou un vecteur de probabilités si `mode='probabilities'` où chaque élément du vecteur indique la probabilité que l'image appartienne à la catégorie correspondante. Dans votre rapport, ce sera bien que vous preniez du recul par rapport à votre travail et juger par exemple votre modèle souffre de overfitting, underfitting ou plus en général quelles pourraient être les amélioration à apporter.

3 Le projet et sa notation

Le projet de cette année voudrais être représentatif d'un cas d'application réelle des notions vues en cours et il consistera donc de trois parties :

(C) le code ;

(P) performances ;

(R) le rapport.

Chacune des ces parties sera détaillée dans la suite de ce document.

3.1 Code (C)

Comme dans tout projet informatiques, il s'agira de produire un programme selon les modalités spécifiées dans les sections suivantes. Ce que nous allons juger dans cette section c'est la qualité du code, la documentation fournie, la reutilisabilité et la généricité.

3.2 Performances (P)

Dans cette section nous allons mesurer les performances du code en termes de : qualité des réponses (*accuracy*), temps d'apprentissage, adaptabilité. Pour l'adaptabilité, nous allons prendre un petit jeu de photos (de la bonne taille) et nous allons vérifier la qualité des réponses.

3.3 Rapport (R)

La dernière partie du projet (mais pas la moins importante !) prévoit la production d'un petit rapport (préférablement en LaTeX) contenant *a minima* :

- un descriptif du sujet et de ses finalités ;
- un descriptif des données utilisées ;
- la méthodologie suivie pour répondre aux questions posées ;
- les résultats obtenus et leur visualisation graphique autant que possible ;
- une section de conclusion et perspectives ;
- une courte présentation des membres du groupe et de leur rôle dans le projet.

3.4 La notation

Chaque partie du projet sera notée sur 5 points et va donc constituer la note $C + P + R$ à laquelle s'ajoutera la note de TP (sur 5 points aussi).

Attention : des outils antiplagiat (par rapport aux autres projets ou par rapport à ce qu'on pourrait trouver sur le web) seront utilisés lors de l'évaluation des points (C) et (R). Des mesures seront prises en cas de plagiat manifeste.

4 Les groupes

Chaque groupe est composé de 2 ou 3 étudiants. Les groupes de moins de 2 ou plus de 3 étudiants sont interdis. Pour inscrire votre groupe au projet il faudra se rendre sur le site Moodle de notre cours, choisir l'onglet `Projet` et accomplir l'activité `Formation des groupes`.

Attention : l'activité a une durée limitée. Elle fermera le **23 décembre à midi**.

5 Ce qu'il faut rendre et quand

Il faudra rendre au professeur mail un fichier nommé `NNL-GX.zip` où `X` est l'identifiant du groupe. Cet archive doit impérativement contenir quatre répertoires : `PRE`, `R`, `FIG` et `REP`. Ces répertoires doivent contenir, respectivement :

- les scripts ou programmes utilisés pour le traitement des données ou si la base de donnée a nécessité d'un pre-traitement ;
- les scripts Python pour effectuer le reste des tâches demandées dans le projet ;
- les scripts python pour générer les graphismes utilises dans le rapport ;
- le rapport final (au format LaTeXvous donner droit à un petit bonus) avec les fichiers annexes (figures, tables, etc).

Attention : le projet est à rendre avant le **25 janvier 2021 à midi**.

5.1 Pénalités de retard et demandes d'extension

Si le projet n'est pas rendu dans les temps prévus, il faudra s'attendre à des pénalités proportionnelles à l'ampleur du retard. Aucune extension (non pénalisé) n'est autorisée, sauf cas exceptionnels qui seront jugés au cas par cas.

6 Comment rendre son projet

Vous pouvez rendre votre projet en allant sur l'ongle `Projet` du site Moodle de notre cours. Vous y trouverez une activité qui s'appelle `Rendre son projet`. L'approbation d'un seul membre du projet suffira pour compléter l'activité.

7 Au secours !

Si pendant le projet vous restez bloqué pour une raison quelconque, n'en restez pas la et appliquez l'algorithme habituel :

1. redoublez vos efforts pendant un temps fini (1 jour maxi) ;
2. si l'étape 1. n'a pas donné les résultats attendus alors demandez à vos camarades ;
3. si l'étape 2. ne fonctionne pas non plus alors demandez à vos camarades de l'an passé (en M2 en ce moment) ;
4. si ça n'a toujours pas marché alors contactez le prof d'urgence !

Les étapes 1. à 3. ne doivent pas prendre plus que 3 jours !

Bon courage à tous !

Références

[1] François Chollet. *Deep learning with Python*. Manning publishing, 2017.