Projet TensorFlow

Classification d'images

1. Objectif

Ce projet a pour but de vous familiariser avec TensorFlow au travers d'un projet ou il vous sera demandé de classifier des images.

Ne vous en faites pas, même si l'objectif peut paraître compliqué de prime abord, il n'en est rien ! Si vous avez suivi assidûment les laboratoires précédents tout devrait bien se passer. Des tutoriels et des explications théoriques vous sont proposés à la fin de ce document afin de vous aider dans sa réalisation.

1.1. Objectifs du projet

Á la fin de ce projet, l'étudiant sera capable de :

1. Charger un dataset d'images et l'utiliser dans un un réseau de neurone utilisant TensorFlow
2. Créer un modèle TensorFlow convolutionnel
3. Modifier un réseau de neurone existant pour classifier les images via le transfer learning
4. Représenter les résultats obtenus

2. Modalités pratiques

- Le projet devra être réalisé via un Jupyter Notebook contenant le code du projet ainsi que les illustrations demandées.
- Peut-être réalisé par groupes de deux étudiants. **Attention, le jour de l'examen les deux étudiants du groupe devront-être capables d'expliquer le code et les concepts théoriques utilisés dans le projet. La note finale n'est pas une note de groupe. Des questions individuelles pourront aussi être posées portant, entre-autres, sur les laboratoires respectifs des étudiants.**
- Le but des groupes est de favoriser une dynamique collaborative propice à une réflexion approfondie et à l'émergence d'interrogations pertinentes. L'interaction entre les membres du groupe vise à encourager une démarche réflexive, permettant ainsi une exploration approfondie des concepts abordés dans ce projet.
- Le projet doit-être zippé et nommé de cette façon : projetTensofFlow2024Nom1Prenom1Nom2Prenom2.zip. Cette archive zip doit contenir toutes les ressources nécessaires au bon fonctionnement de votre code. N'incluez pas les données que vous téléchargez avec le code…
- Le projet devra-être rendu pour le dimanche 16/06/2024 avant l'examen à l'adresse mail a.tillieux@helmo.be. Si le projet à été réalisé par deux, n'oubliez pas de mettre votre coéquipier en copie du mail.
- N'oubliez pas que les laboratoires devront aussi être rendus pour le dimanche 16/06/2024. Un mail de confirmation vous sera envoyé afin de confirmer la bonne réception de vos travaux. Si aucun mail de confirmation n'a été envoyé le lendemain, n'hésitez pas à renvoyer un mail…
- Une présentation de 20 minutes devra être réalisée le jour de l'examen. Cette présentation présentera les résultats, mettra en avant les problèmes rencontrés et les solutions trouvées et proposera des pistes d'améliorations.
- Pour rappel, le rapport (notebook) du projet et sa présentation valent pour 50% de la note de l'UE.

3. Énoncé

3.1. Première partie : Classification d'images

Pour la première partie du projet, il est demandé de :

1. Choisir un dataset pour la classification d'images proposé par TensorFlow. Vous avez le choix entre les datasets cats-vs-dogs, horses_or_humans ou rock_paper_scissors. Vous devez télécharger ces images avec **tfds** (méthode load ?) !
2. Décrire le dataset choisi (type d'image, nombre d'images, nombre de classes, montrer plusieurs exemples d'images, etc.).
3. Déterminer les paramètres de taille d'image, du nombre de neurones en entrée et la taille du batch. Justifiez vos choix.
4. Répartir le dataset en données d'apprentissage et de validation. Justifiez la taille choisie pour chacun des ensembles. Expliquez pour quelle raison cette étape est nécessaire.
5. **Normalisez** les données pour le réseau de neurones. Justifiez la valeur choisie. Expliquez pourquoi il est important de normaliser les données. Tout est expliqué dans les tutoriels…
6. Créer un réseau custom, composé d'un nombre de séquences de convolution/pooling choisi par vos soins. Le choix doit être argumenté. Pour vous simplifier la tâche, il est proposé de travailler avec une couche de convolution de ce type:
   ```
   layers.Conv2D(x, 3, padding = 'same', activation = 'relu')
   ```
   avec x qui prend une valeur de $16 * 2^n$, avec n = 0, 1, 2, 3,... l'indice de la couche de convolution. Vous pouvez tester l'optimiseur *adam* et la fonction coût de type *SparseCategoricalCrossentropy*.
7. Discuter les résultats obtenus en termes de performance et d'apprentissage. A noter qu'il est normal d'obtenir de piètres résultats avec un réseau avec une seule couche de convolution. Représentez notamment la valeur de la fonction coût et accuracy pendant l'apprentissage. Expliquez les différences obtenues entre la valeur sur les données d'apprentissage et de validation (overfitting ?).
8. Une fois que vous arrivez à un résultat satisfaisant, prenez une photo originale que vous joindrez au projet et donnez là en prédiction à votre réseau de neurones. Quelles sont les prédictions, sont-elles exactes ? N'oubliez pas de redimensionner l'image avant de la donner à votre réseau…

3.2 Deuxième partie : Transfer Learning

La deuxième partie consiste à utiliser le *Transfer learning* en utilisant le même dataset qu'utilisé dans la première partie. Il vous est demandé de :

1. Utiliser et charger le modèle *MobileNetV2*. La dernière couche ne devra pas être utilisée et les poids pré-entraînés devront être utilisés :
   ```
   base_model = tf.keras.applications.MobileNetV2(input_shape = (... ,
   ... , ...), weights = 'imagenet')
   ```

Puisque ce modèle ne doit pas être modifié, il faut bien préciser qu'il ne peut pas être entraîné `base_model.trainable = False`

Décrivez brièvement ce réseau et expliquez ce qui est obtenu si cette dernière couche est enlevée.

2. Rajouter une couche qui permettra de classifier par rapport à vos données choisies. Il est proposé de rajouter la couche suivante :

   `tf.keras.layers.Dense(n)`

   avec n le nombre de classes s'il y a plus de deux classes sinon n=1.
3. Introduire une couche de dropout. Expliquez à quel endroit de votre réseau elle fait le plus sens et expliquez pourquoi elle peut être utilisée.
4. Discuter les performances obtenues et représenter l'évolution de l'accuracy et de la fonction coût pendant l'apprentissage.
5. Comparer les performances avec le réseau custom créé dans la première partie de l'énoncé.
6. Discuter de différentes manières d'améliorer votre apprentissage.
7. Une fois que vous arrivez à un résultat satisfaisant, utilisez la même photo originale que vous avez utilisée pour la première partie et donnez là en prédiction à votre réseau de neurones. Quelles sont les prédictions, sont-elles exactes ? N'oubliez pas de redimensionner l'image avant de la donner à votre réseau…

4. Théorie

4.1. Les Datasets

Un dataset est un ensemble de données (images, pistes audio, données de capteurs, etc.) qui ont été correctement annotées. Par exemple, le dataset ImageNet est un ensemble de 14 millions d'images qui ont été annotées à la main avec la classe correspondante.

Un dataset doit impérativement avoir un nombre suffisant d'exemplaires pour chacune des classes sans quoi l'apprentissage sera forcément de mauvaise qualité. Par exemple, ImageNet possède au alentour du millier d'exemplaires pour chacune des classes (1000 images de chat, 1000 de chien, ...).

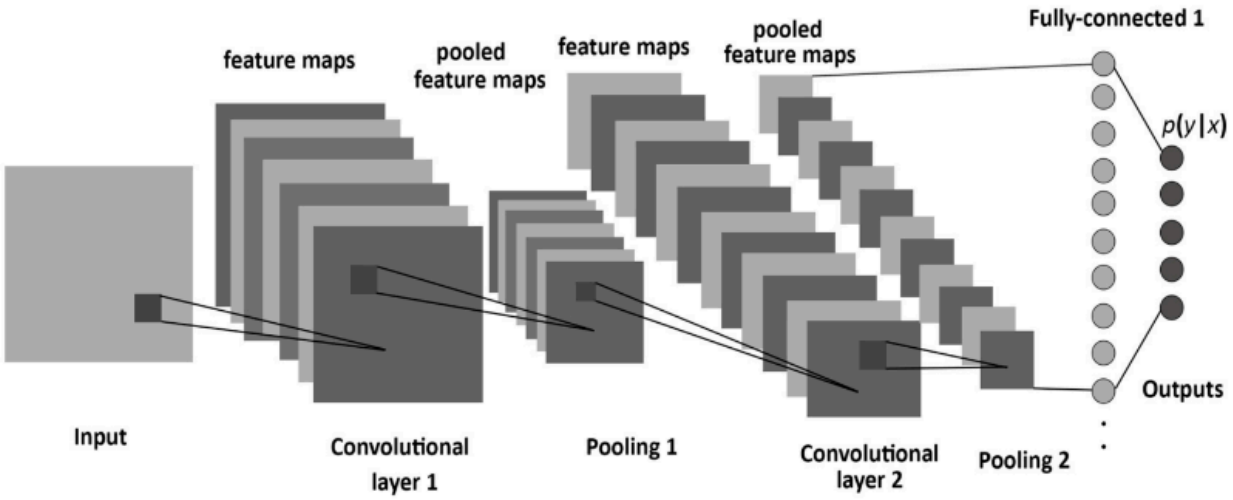
Afin de réussir à réaliser un apprentissage correct, le réseau de neurones à entraîner a généralement besoin d'avoir des images standardisées. Par exemple, le réseau MobileNet a besoin d'entrées comprises entre -1 et 1 (chaque canal de chaque pixel doit valoir entre -1 et 1). Pour pouvoir utiliser des images classiques qui ont les canaux RGB codés sur 1 byte (0-255), il est donc obligatoire de réaliser une transformation préalable. Un autre exemple est la taille des images qui doit correspondre au nombre de neurones de la première couche. Ci-dessous, on peut voir les solutions apportées aux exemples donnés.

```
from tensorflow.keras.preprocessing import image
# On peut forcer la taille des images chargées via le * target_size *
img = image.load_img(img_path,target_size =(224 , 224))
# On crée une couche de neurone spécifiquement pour le resizing,
# ce sera la première couche du réseau
layer_resc = tf.keras.layers.Rescaling(1./127.5, offset = -1)
```

Il est aussi commun de voir ces opérations remplacées par une couche de neurones qui se charge de la normalisation.

4.2. Le convolutional neural network (CNN)

Afin de réussir à faire de la classification d'image, un type de réseau est particulièrement adapté : les convolutional neural network (CNN). Je vous invite à regarder cette vidéo qui décrit ce type de réseau.



Comme on le voit, un réseau de type CNN consiste à appliquer séquentiellement des couches de convolution et de pooling. Ces deux étapes sont répétées un certain nombre de fois. Les convolutions peuvent être apparentées à un filtre appliqué à l'image originale afin de détecter un ensemble de caractéristiques importantes. Le pooling correspond simplement à une compression. Enfin, une dernière couche entièrement connectée permettra de lier la sortie de la dernière couche de pooling à chacune des classes utilisées dans notre problème de classification.

4.3. Le transfer learning

Le transfer learning consiste à utiliser un réseau existant et à ne ré-entraîner que les dernières couches afin de classifier par rapport à des nouvelles classes.
Les réseaux utilisés aujourd'hui font l'objet d'études approfondies et sont bien plus complexes que celui utilisé à la section précédente. Par exemple, le modèle *MobileNetV2* possède 3.5 millions de paramètres et 105 couches. Ce modèle est particulièrement utilisé quand peu de ressources sont disponibles, comme sur un téléphone. Il est utilisé pour la classification d'image et peut, par exemple, être entraîné sur le dataset *ImageNet*. On considère généralement que toutes les couches de convolutions permettent de comprendre l'image qui est donnée en entrée et il n'est donc pas nécessaire de modifier ces couches.

4.2. Tutoriels

Afin de mener à bien ce projet, je vous conseille de suivre ces tutoriels proposés par TensorFlow. Ils vous permettront :

1. De comprendre comment utiliser les CNN pour la classification.
2. Comprendre comment classifier une image.
3. De comprendre comment fonctionne le transfert learning. (Seule cette partie est intéressante pour le projet)

Bon travail !
PS : Félicitations ! Tu es maintenant capable de créer un réseau de neurones permettant de classifier toutes sortes d'images !!