Rapport SAE 501

Semaine 8

MALORON Arthur,
DOTTO Matis,
MANICK Luc,
BELMADANI Abdourrahmane

Sujets abordées

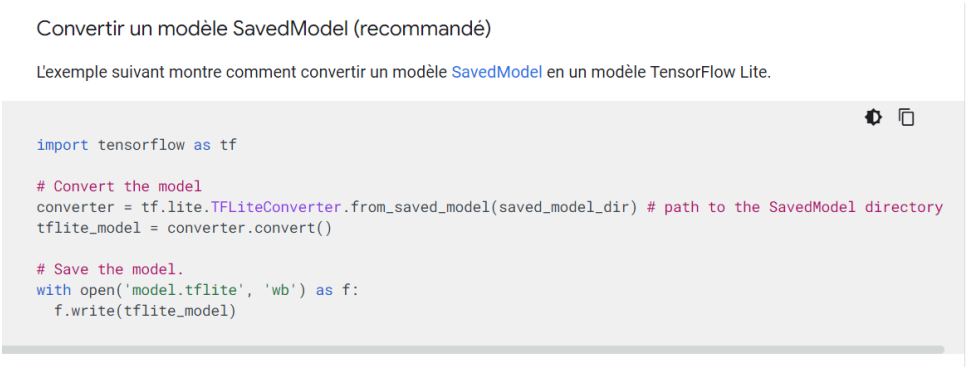
Entrainement du modèle

Au cours de cette semaine nous avons encore travaillé sur le jeu d'essai …

Conversion du modèle

Le modèle que nous essayons de convertir en .tflite est le modèle SSD Mobilenet V2 320x320 provenant du tensorflow detection model zoo.

Nous avons d'abord réessayer la solution officielle de tensorflow. A savoir l'utilisation du TFFLiteConverter de l'API python de tensorflow.


Convertir un modèle SavedModel (recommandé)

L'exemple suivant montre comment convertir un modèle SavedModel en un modèle TensorFlow Lite.

```
import tensorflow as tf

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir) # path to the SavedModel directory
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Cependant, cette solution nous renvoie des résultats plutôt inattendus. En effet, il y a 8 tensors de sortie, ce qui ne semble pas correct pour un modèle SSD Mobilenet.

L'erreur que nous avons est AssertionError: StatefulPartitionedCall:1 is not in graph. Dans une version pyhton plus haute, le modèle se build tout de même mais les tensors d'output sont toujours incorrects.

Nous avons ensuite tenté d'utiliser les scripts présents sur le github officiel de tensorflow/models.
Etant donné que nous utilisons un modèle SSD Mobilenet, nous avons voulu utiliser le script export_tflite_ssd_graph.py dans le dossier research/object_detection. Il y a cependant eu des problèmes au niveau des dépendances, ne trouvant aucune version pour laquelle les packages compilés avec protobuf peuvent tous être importés sans erreur.

Label des images

Nous avons effectué des labels pour d'autres classes, nous avons ajouté 3 nouvelles classes.
L'objectif est d'avoir une dizaine de classes.

Déploiement et hébergement de l'API

Durant cette semaine, nous avons déployé l'API sur les serveurs de l'IUT. Cependant, certains problèmes subsistent. L'API renvoie correctement des données, mais uniquement depuis l'index de l'API. Nous suspectons que les serveurs de l'IUT ne sont pas configurés pour reconnaître les différentes routes.

Face à cette situation, plusieurs choix s'offrent à nous. D'abord, nous pouvons poursuivre nos recherches pour tenter de configurer l'hébergement sur les serveurs de l'IUT, ce qui serait le plus pratique. Sinon, nous envisagerons d'utiliser des services tiers spécialisés dans l'hébergement de projets web, afin d'assurer le fonctionnement de notre API et de la base de données.

Pour éviter de perdre du temps, nous allons travailler en parallèle sur l'implémentation de l'envoi d'images vers l'API. Nous commencerons par l'envoi depuis Android, avant de coder la récupération et l'enregistrement des images au sein de l'API.

Objectifs hebdomadaires

- Test d'entraînement de l'IA sur au moins 5 classes (100 images par classe)
- Conversion d'un modèle SSD mobilenet en .tflite
- Trouver une solution pour l'hébergement de l'API
- Implémenter l'envoi des images vers l'API

Lien du Trello

https://trello.com/invite/b/670e75ea832d0d8d7b7625cf/ATTI30db0fce264f5fcced5a98e25047eb67DF2B89AE/s501,