Tutoriel création d'une image docker prête à entraîner des modèles du Tensorflow Model Zoo

Eustache Le Bihan

1 S'assurer que l'hôte à bien CUDA installé

1. Dans un terminal de l'ordinateur hôte, exécuter :

   ```
   nvidia-smi
   ```

2. Regarder la version de CUDA installée sur le résultat affiché. Ex : CUDA Version : 11.3
3. Dans un terminal de l'ordinateur hôte, exécuter (en changeant éventuellement la version de CUDA) :

   ```
   docker run --rm --gpus all nvidia/cuda:11.0-base
      nvidia-smi
   ```

   Cette dernière doit afficher le même résultat que la commande i), mais puisqu'elle est exécutée depuis un container, cela confirme que les containers ont bien accès aux ressources GPU.

2 Télécharger l'image docker et démarrer un container avec cette image

On télécharge ici une image docker avec tensorflow GPU installé ainsi que jupiter notebook.

1. Dans un terminal de l'ordinateur hôte, exécuter :

   ```
   docker pull tensorflow/tensorflow:latest-gpu-jupyter
   ```

2. Démarrer un container avec cette image.
   - Choisir un nom : ici tf_ship_detection
   - Choisir un volume à monter dans le container, c'est-à-dire qu'il partagera avec l'hôte (ici /home/elebihan/ship_data) et choisir où le monter (ici /tf/ship_data). Dans un terminal de l'ordinateur hôte, exécuter :

```
docker run -dit --gpus all --name tf_ship_detection -v
    /home/elebihan/ship_data:/tf/ship_data tensorflow/
   tensorflow:latest-gpu-jupyter
```

3. Dans un terminal de l'ordinateur hôte, exécuter :

   ```
   docker ps
   ```

   Le container que l'on vient de créer doit apparaître dans la liste affichée.

3 Installer les dépendances nécessaires dans le container

1. On va maintenant ouvrir un terminal dans le container.
   Dans un terminal de l'ordinateur hôte, exécuter (remplacer le nom utilisé pour le container) :

   ```
   docker exec -it tf_ship_detection bash
   ```

2. *étape propre à un ordinateur utilisé à l'IRENav*
   Pour donner un accès internet au container dans un terminal du container, auquel on accède via étape i), exécuter :

   ```
   export HTTPS_PROXY=http://proxy.ecole-navale.fr:8080
   export HTTP_PROXY=http://proxy.ecole-navale.fr:8080
   export https_proxy=http://proxy.ecole-navale.fr:8080
   export http_proxy=http://proxy.ecole-navale.fr:8080
   ```

   Puis exécuter pour vérifier que l'accès à internet est effectif :

   ```
   apt-get update
   ```

3. Installer Git :

   ```
   apt-get install git
   ```

4. Installer tensorflow Object Detection API dans le répertoire tf et installer protobuf.
   Dans un terminal du container, auquel on accède via étape 1., exécuter :

   ```
   cd tf
   git clone https://github.com/tensorflow/models
   apt-get install protobuf-compiler
   ```

5. Dans un terminal du container, auquel on accède via étape 1., exécuter :

   ```
   cd models/research && protoc object_detection/protos
      /*.proto --python_out=. && cp object_detection/
      packages/tf2/setup.py . && python -m pip install .
   ```

4 Vérifier que le container est prêt

1. Dans un terminal du container, auquel on accède via étape 2-1., exécuter :

   ```
   python /tf/models/research/object_detection/builders/
      model_builder_tf2_test.py
   ```

   Le script est censé retourner : OK (skipped=1)

5 Sauvegarder le container dans une nouvelle image

1. Dans un terminal de l'hôte, exécuter (remplacer le nom utilisé pour le container et le nom souhaité pour l'image :

   ```
   docker commit tf_ship_detection tf_base_config
   ```