8INF919 : AA pour les données massives

Devoir #2
Entraînement d'un Réseau de Neurones Profond de type CNN sur plusieurs GPU et Cluster

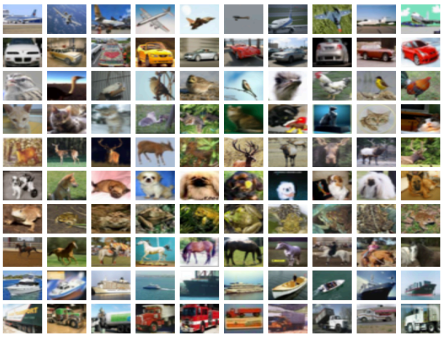
Le travail peut se faire en binôme
Date de remise : le jeudi 7 Décembre 2023

1. But

- Étudier un réseau de neurones profond de type convolution (CNN) pour classer des images;
- Utilisation de deux bibliothèques Keras et TensorFlow comme plateforme pour l'apprentissage profond;
- Utilisation de plusieurs GPU et un cluster de plusieurs nœuds (4 nœuds, 32 CPU)

2. Étude de cas

L'étude concerne la distribution de l'apprentissage à grande échelle avec l'API tf.distribute.Strategy, un outil pour la distribution de l'entrainement d'un modèle sur une machine dotée de plusieurs GPU ou un cluster doté de plusieurs CPU. Cette étude est subdivisée en deux parties d'exploration avec respectivement deux stratégies : (i) la mise en miroir sur GPU et, (ii) le serveur de paramètres sur cluster. Dans chacune des parties, vous entraînerez un réseau de neurones profond en utilisant une architecture de type CNN, pour classer des images de couleur contenues dans le dataset « CIFAR-10 » intégré dans Keras (https://keras.io/api/datasets/cifar10/). Vous pouvez aussi télécharger le dataset du site officiel suivant de l'Université de Toronto : https://www.cs.toronto.edu/~kriz/cifar.html



Le jeu de données est composé de 60 000 petites images en **couleur** de 32 × 32 pixels, réparti en un jeu d'entrainement constitué de 50.000 images et un jeu de test de 10.000 images. L'ensemble est classé en 10 catégories d'entités physique, tels que des avions, camions, voitures, animaux. La correspondance de tous les entiers 0-9 aux étiquettes de classe est répertorié comme suit: (0 : Avion, 1 : Voiture, 2 : Oiseau, 3 : Chat, 4 : Biche, 5 : Chien, 6 : Grenouille, 7 : Cheval, 8 : Bateau, 9: Camion). Il s'agit d'un problème de classification qui est plus difficile qu'un CNN manipulant des images de niveaux de gris de 28X28, comme celui de MNIST ou de Fashion_MNIST.

Le but de votre travail est d'élaborer une architecture d'un CNN se rapprochant le plus possible d'une performance au-delà de 81%, **et pourquoi pas un peu plus**[1], dont le nombre de paramètres (poids du réseau de neurones) peut varier entre **0.5 et 7 millions**, en moyennant l'accélération de l'entrainement du fait de sa distribution. **Vous pouvez vous inspirer de toute sorte d'architecture existante de CNN mais il faut qu'elle soit entrainable avec les ressources que vous avez !**

2.1 Exploration de la stratégie de mise en miroir sur GPU

La stratégie de mise en miroir consiste à répliquer le modèle sur plusieurs GPU. La plupart des machines qui sont à votre disposition (Colab de Google[2], Salle-Maîtrise au P4-6600) ne dispose que d'un seul GPU. Pour avoir accès à une configuration distribuée de GPU, vous pouvez créer des GPUs **virtuels** à partir d'un seul GPU. Pour cela, vous devez exécuter l'instruction de configuration suivante et qui doit être effectuée juste après l'importation du module tensorflow :

```
import tensorflow as tf

#le code suivant divise le premier GPU (GPU 0) en 2 GPU -périphériques- virtuels. Chacun 2Gio de RAM.
#les instructions suivantes doivent être effectuées juste après l'importation du module tensorflow.

physical_gpus = tf.config.experimental.list_physical_devices("GPU")
tf.config.experimental.set_virtual_device_configuration(
    physical_gpus[0],
    [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048),
     tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048)])
```

Vous pouvez vérifier la prise en charge du nombre de GPU en affichant la nouvelle configuration matérielle d'exécution en tapant les instructions suivantes. Au préalable, il faut définir l'objet stratégie de mise en miroir, par l'instanciation de la classe MirroredStrategy :

[1] Ça demeure un défi où on organise plusieurs compétitions dans le monde ! N'oublions pas on travaille avec des petites images de 32X32 pixels !

[2] https://colab.google/ **la version Cloud de TensorFlow qui offre gratuitement la plateforme et le matériel nécessaire pour accélérer l'entraînement. Vous pouvez aussi utiliser votre propre installation**

Création d'un objet de type MirroredStrategy. Ceci va gérer la distribution et fournira un gestionnaire de contexte (`MirroredStrategy.scope`) afin de construire le réseau de neurones.

```
[ ]  strategy = tf.distribute.MirroredStrategy()
```

```
[ ]  print('Nombre de périphériques (GPU): {}'.format(strategy.num_replicas_in_sync))
```

```
Nombre de périphériques (GPU): 2
```

Veuillez noter que si vous décidez de changer de configuration, par exemple passer à 4 GPU, vous devez arrêter et redémarrer le kernel.

Pour utiliser la version GPU de Tensorflow sur l'une des machines de la salle de Maitrise (P4-6600), vous devez suivre la démarche que j'ai décrit en annexe, en page 6.

Veuillez répondre aux sous-questions suivantes:

2.1.1 Implanter un CNN dont l'architecture reste à déterminer par vous et qui doit améliorer l'exactitude (accuracy), au-delà de 81%, en d'autres termes, au minimum 81% avec les images de test.

2.1.2 Tracer la courbe d'apprentissage du CNN obtenu en 2.1.1 et analyser son comportement selon les différents cas de figures suivantes : **1GPU, 2 GPU, 4 GPU**. La taille du **lot d'un réplicat** peut être fixé à 64 images. Commentez les résultats obtenus.

2.1.3 Faîtes une étude comparative sur le modèle obtenu en 2.1.2 en s'appuyant sur les critères suivants : l'exactitude, le temps d'entraînement, la quantité de données transférée pour le **calcul du gradient moyen** selon les différents cas de figures **2GPU, 4 GPU.** Argumentez les résultats obtenus.

2.2 Exploration de la stratégie de serveur de paramètres sur le cluster de calcul Québec

La stratégie de serveur de paramètres consiste à répliquer le modèle sur les différents noeud-workers tout en centralisant les paramètres et le calcul du gradient moyen sur un nœud (maitre). Il s'agit d'expérimenter l'architecture du CNN trouvée dans la partie précédente, éventuellement de la modifier, pour tenter d'améliorer son exactitude tout en tirant profit des ressources du cluster de calcul Québec : **uqac-8inf919.calculquebec.cloud**.

2.2.1 Considérer l'architecture du CNN trouvée à la question 2.1, partie 1;

2.2.2 Entrainer le CNN en fixant la taille du BATCH_SIZE à **256**.

2.2.3 Mesurer l'exactitude et le temps d'entrainement dans les 2 cas de figures : 2 nœuds et 4 nœuds.

2.2.4 Comparer les résultats avec ceux de la partie 1 selon les cas figures 1GPU, 2GPU, 4 GPU.

Pour effectuer cette étape, vous devez :

1. Vous connecter au cluster, **via ssh**, en utilisant votre identifiant
2. Se positionner sur la racine ($HOME) ;
3. Charger les modules requis par TensorFlow en tapant la commande suivante après le signe $

   [usager@login1 ~]$ ***module load python/3***
4. Créer un nouvel environnement Python,

   [usager@login1 ~]$ ***virtualenv --no-download tensorflow***
5. Activer le nouvel environnement

   [usager@login1 ~]$ ***source tensorflow/bin/activate***
6. Installez TensorFlow dans votre nouvel environnement virtuel en utilisant la commande suivante :

   (tensorflow) [usager@login1 ~]$ ***pip install --no-index tensorflow==2.8***
7. Une fois le tout est installé, vous devez télécharger et transférer les fichiers suivants sur le cluster à partir du site moodle du cours dans la section devoir #2 :

   7.1. le dataset « **cifar-10-batches-py.tar.gz** » et le déposer dans le répertoire **~./keras/datasets**

   7.2. les 3 fichiers de configuration : « **config_env.sh** », « **launch_training.sh** », « **tensorflow-multiworker.sh** »

   7.3. Changer leurs statuts par la commande suivante « chmod u+x » :

   [usager@login1 ~]$ ***chmod u+x config_env.sh***

   [usager@login1 ~]$ ***chmod u+x launch_training.sh***

   [usager@login1 ~]$ ***chmod u+x tensorflow-multiworker.sh***

   7.4. le fichier python « **tensorflow-multiworker.py** » qui servira de modèle. **Il faut modifier juste la partie du modèle du réseau de neurones profond sans toucher à la partie configuration matérielle. Veuillez voir l'exemple.**

7.5. Lancer un job en exécutant la commande sbatch

[usager@login1 ~]$: sbatch ***tensorflow-multiworker.sh***

Voici un exemple du fichier de code en python dans la page suivante. Vous devez procéder au changement de la **ligne 36** pour définir l'architecture (modèle) du CNN. Le reste ne doit pas être changé. Les premières lignes est une déclaration de la configuration matérielle du serveur et les autres nœuds workers et ainsi qu'une spécification de la stratégie de communication en anneau. Veuillez noter que la taille du lot est déjà configurée à 256 images. Idem pour les dernières lignes de code où on charge le dataset cifar10 et ainsi le lancement de l'entrainement à l'aide de la fonction fit(). Bien entendu, vous pouvez importer d'autres modules python ou tensorflow.

Launcher | tensorflow-multiworker.py

```
15
16 args = parser.parse_args()
17
18 host_list = expand_hostlist(os.environ['SLURM_NODELIST'])
19
20 port = 3456
21
22 host_list = [f"{host}:{port}" for host in host_list]
23
24 host_dict = {'worker': host_list}
25
26 cluster_config = tf.distribute.cluster_resolver.SimpleClusterResolver(ClusterSpec(host_dict),
27                                                                        task_type="worker",
28                                                                        task_id=int(os.environ['SLURM_PROCID']),
29                                                                        rpc_layer='grpc')
30 comm_options =
   tf.distribute.experimental.CommunicationOptions(implementation=tf.distribute.experimental.CommunicationImplementation.RING)
31
32 strategy = tf.distribute.MultiWorkerMirroredStrategy(cluster_resolver=cluster_config, communication_options=comm_options)
33
34 with strategy.scope():
35
36     ##ICI VOUS DEVEZ DEFINIR L'ARCHITECTURE DU CNN
37     ##....
38
39     model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
40         optimizer=tf.keras.optimizers.SGD(learning_rate=args.lr),metrics=['accuracy'])
41
42 ### This next line will attempt to download the CIFAR10 dataset from the internet if you don't already have it stored in
   ~/.keras/datasets.
43 ### Run this line on a login node prior to submitting your job, or manually download the data from
44 ### https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz, rename to "cifar-10-batches-py.tar.gz" and place it under
   ~/.keras/datasets
45
46 (x_train, y_train),_ = tf.keras.datasets.cifar10.load_data()
47
48 dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(args.batch_size)
49
50 EPOCHS = 50
51
52 from time import time
53
54 t0 = time()
55 model.fit(dataset, epochs=EPOCHS)
56 tt = time() - t0
57 print("classifier trained in {} seconds".format(round(tt,3)))
58
```

3. Livrable

3.1 Fournir un ou plusieurs fichiers **ipynb** comprenant les étapes demandées;

3.2 Commenter chacune des étapes en faisant référence aux questions posées;

3.3 Indiquer dans l'introduction du fichier **ipynb,** si vous avez ajouté de nouvelles fonctionnalités non-demandées. Il faut les commenter

3.4 Indiquer le nom et prénom.

3.5 Mettez le tout dans un seul fichier compressé .zip

Annexe Utilisation de TensorFlow-GPU au P4-6600 du DIM, UQAC

Pour utiliser la version GPU de Tensorflow sur l'une des **machines de la salle de Maîtrise (P4-6600),** vous devez suivre la démarche suivante en installant **la version 2.10 de TensorFlow** compatible avec les cartes graphiques NVIDIA GetForce RTX 3060 Ti:

1. S'authentifier sur un des **postes du P4-6600**
2. Dans le menu démarrer, lancer le lien « ***Anaconda Prompt (Anaconda3)*** »;
3. Créer l'environnement **tf** avec la commande **conda** en tapant :

   conda create -n tf python==3.9

4. Activer l'environnement **tf** avec la commande conda:

   conda activate tf

5. Installer la version **2.10** de tensorflow avec la commande **pip** :

   pip install tensorflow==2.10

6. Installer les modules manquants :

   pip install jupyterlab ipykernel pandas scipy matplotlib scikit-learn

7. Lancer Jupyter lab :

   jupyter lab

NB :

- Si vous vous reconnectez **sur le même poste**, une autre fois, vous devez juste activer l'environnement **tf** : ***conda activate tf***
- Par contre, si vous utilisez un autre poste, vous devez refaire la démarche (1-7).
- L'utilisation d'un GPU-Tensorflow sur **Colab de Google**, ne nécessite aucune installation préalable, mais il n'y aucune garantie quant à sa disponibilité illimitée. C'est la raison pour laquelle, il faut utiliser le mécanisme de rappel (callback) basé sur des points de contrôle de l'entrainement afin de sauvegarder le modèle et le reprendre à n'importe quel moment en cas de rupture[3] !

[3] Veuillez utiliser l'exemple du cours, chapitre7, ou bien consulter la bibliothèque **Keras** en cliquant sur : https://keras.io/api/callbacks/model_checkpoint/