TD2 – IA embarquée

L'objectif de ce TD est découvrir le framework de deep learning PyTorch[1] et de mettre en œuvre un modèle de classification d'images sur un dispositif embarqué. Il s'agira dans notre cas de la carte NVIDIA Jetson Nano[2].

1. Installation et initiation à PyTorch

PyTorch est un module de deep learning pour Python, parmi les plus populaires. Dans ce TD nous allons apprendre à l'utiliser pour la classification d'images par des réseaux de neurones convolutionnels.

La procédure d'installation est décrite sur la page https://pytorch.org/get-started/locally/.
En fonction de la configuration du système, la ligne de commande appropriée devra être exécutée dans un terminal.
Exécuter le code Python suivant pour vérifier l'installation.

```
import torch
dtype = torch.FloatTensor
N, D = 14, 10
x = torch.randn(N, D).type(dtype)
print(x)
import torchvision.models as models
alexnet = models.alexnet()
print(alexnet)
```

Afin de se familiariser avec PyTorch, le tutoriel suivant devra être étudié : https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

2. Connexion à la carte Jetson Nano

La connexion entre l'ordinateur et la carte se fera par un cable ethernet croisé (https://en.wikipedia.org/wiki/Ethernet_crossover_cable). La carte a été configurée avec les paramètres IPv4 suivants : IP: 10.0.0.1, Netmask: 255.0.0.0. Il est nécessaire de configurer la connexion de l'ordinateur avec ces paramètres : IP: 10.0.0.2, Netmask: 255.0.0.0. L'interface de configuration de la connexion dépend du système d'exploitation (Windows, MacOS, Linux).

Une fois configurée, la connexion à la carte pourra se faire par SSH. Sous MacOS et Linux, il suffit d'ouvrir un terminal, d'exécuter la commande suivante et de saisir le mot de passe ("embedded").

```
ssh -Y embedded@10.0.0.1
```

[1] http://pytorch.org
[2] https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

Sous Windows, il faudra télécharger et installer le logiciel MobaXterm (https://mobaxterm.mobatek.net), puis se connecter en utilisant les adresse et identifiant ci-dessus.

Les commandes Unix de base utiles lors de l'utilisation d'un terminal sont présentées ici : https://doc.ubuntu-fr.org/tutoriel/console_commandes_de_base

Vous pouvez copier des fichiers de votre ordinateur à la carte et réciproquement en utilisant la commande **scp**. Par exemple :

```
scp embedded@10.0.0.1:camera/face_detect.py .
```

pour copier dans le répertoire courant de votre ordinateur le fichier « face_detect.py » se trouvant dans le répertoire « camera » de la carte, et

```
scp mon_programme.py embedded@10.0.0.1:toto/
```

pour copier dans le répertoire « toto » de la carte le fichier « mon_programme.py » se trouvant dans le répertoire courant de votre ordinateur.

Éteindre proprement la carte

ATTENTION : Avant de couper l'alimentation de la carte, il est impératif d'éteindre correctement le système Linux avec la commande :

```
shutdown -h now
```

(Au cas où l'arrêt ne se fasse pas en raison d'un autre utilisateur connecté, suivre les instructions renvoyées par la commande ci-dessus pour finaliser l'arrêt).

3. Test de la carte

Si la carte est en fonctionnement, commencer par éteindre le système Linux (voir ci-dessus) puis débrancher le cordon d'alimentation. Connecter la caméra à la carte comme décrit à (https://www.jetsonhacks.com/2019/04/02/jetson-nano-raspberry-pi-camera/). Allumer ensuite la carte, puis une fois connecté, aller dans le répertoire "camera" à partir du terminal et exécuter la commande suivante :

```
python3 face_detect.py
```

Sous MacOS, il sera nécessaire qu'un serveur graphique X soit installé sur votre système pour permettre l'affichage sur votre ordinateur du flux d'images provenant de la carte Jetson. Xquartz remplit cette fonction : https://www.xquartz.org

4. *Utilisation d'un modèle CNN pré-entrainé*

Nous allons maintenant utiliser un modèle pré-entrainé sur le jeu de données imagenet pour faire de la classification d'images
(https://pytorch.org/docs/stable/torchvision/models.html).
Créez votre répertoire de travail et positionnez-vous dedans avec les commandes :

```
mkdir ~/mon_nom
cd ~/mon_nom
```

Les noms des classes pouvant être détectées par le modèle sont donnés dans le fichier "imagenet-simple-labels.json" qui sera utilisé par le programme pour indiquer la classe reconnue à partir d'une image.

Télécharger ce fichier dans votre répertoire de travail en utilisant la commande :

```
wget http://perso.ec-lyon.fr/alberto.bosio/teaching/ia/imagenet-simple-labels.json
```

Créer un répertoire "images" et télécharger l'image de test :

```
wget http://perso.ec-lyon.fr/alberto.bosio/teaching/ia/dog.png
```

Télécharger le code Python suivant :

```
wget http://perso.ec-lyon.fr/alberto.bosio/teaching/ia/run1.py
```

Expérimentations :

- Etudier le code et les résultats obtenus.
- Modifier le code afin d'utiliser les images de la caméra comme entrée du classifieur.
- Quantifier les performances de l'inférence : réaliser l'inférence sur la carte un certain nombre de fois et évaluer le temps d'inférence. Afficher les résultats sur un graphique (avec matplotlib). Faire de même sur votre ordinateur et comparer. Il faudra pour cela avoir enregistré les images ayant été traitées sur la carte obtenues à partir de la caméra.
- Expérimenter avec d'autres modèles CNN pré-entrainés.

5. *Transfer Learning*

Pour ce travail, nous allons utiliser un modèle pré-entrainé (ResNet18) comme extracteur de descripteurs et allons affiner la classification en entrainant uniquement la dernière couche fully connected du réseau. Ainsi, la couche de sortie du réseau pré-entrainé sera remplacée par une couche adaptée aux nouvelles classes à reconnaître qui seront dans notre cas fourmis (ants) et abeilles (bees).

Télécharger et décompresser le jeu de données en exécutant les commandes suivantes dans le terminal :

```
wget https://download.pytorch.org/tutorial/hymenoptera_data.zip
unzip hymenoptera_data.zip
```

Télécharger et exécuter le code python suivant afin d'afficher quelques images du jeu de données :

```
wget http://perso.ec-lyon.fr/alberto.bosio/teaching/ia/run2.py
```

Télécharger et exécuter le code python suivant qui utilise un modèle pré-entrainé ResNet18 en ayant remplacé la couche de sortie pour la classification ants/bees :

```
wget http://perso.ec-lyon.fr/alberto.bosio/teaching/ia/run3.py
```

Expérimentations :

- Etudier le code et les résultats obtenus.
- Comparer les performances d'inférence de la carte par rapport à votre ordinateur.
- Modifier le code pour prendre en entrée les images de la caméra. Comme il y a peu de chances que des fourmis et des abeilles se trouvent dans la salle, afficher sur l'écran de l'ordinateur des images de ces insectes et filmer l'écran avec la caméra pour obtenir les résultats de classification.

6. Travail à rendre

Ce travail peut être réalisé par trinôme (en raison du nombre de cartes disponibles), et un compte-rendu numérique devra être produit. Il devra contenir notamment les explications concernant le principe des fonctions que vous avez programmées ainsi que leur code commenté. Les expérimentations que vous aurez réalisées devront être décrites et les résultats également commentés. Le programme complet (ensemble des fichiers .py) devra être joint au fichier pdf au sein d'une archive « .zip » qui devra être déposée sur le site « Moodle» pour le jeudi 25 novembre 2021.