IFT 725 : Devoir 3

Travail par équipe de 2 ou de 3

Le but de ce devoir est de vous introduire à la bibliothèque *pyTorch*, aux réseaux à convolutions avancés ainsi qu'aux méthodes de segmentation. Bien qu'il soit possible d'entraîner un réseau sur CPU, il est beaucoup plus rapide et utile d'utiliser un GPU (processeur de carte graphique). Si vous n'avez pas accès à un GPU permettant d'effectuer des calculs scientifiques (souvent de marque NVidia), vous pourrez utiliser gratuitement les services de « Google Colab ». Pour ce faire, des instructions ont été mises à votre disposition dans le fichier «README.md ».

1. **[1 point]** Afin de vous familiariser avec *pyTorch,* vous devez d'abord effectuer un exercice simple inspiré des devoirs 1 et 2. En effet, il vous est demandé de remplir le document **`pytorch_tutorial.ipynb`**, en y mettant le code de Justin Johnson disponible sur le site suivant :

   https://pytorch.org/tutorials/beginner/pytorch_with_examples.html

   Comme vous pouvez le constater, cette page contient plusieurs petits paragraphes de code (des « snippets ») grâce auxquels on vous introduits aux variables et tenseurs *pyTorch*, à leur utilisation au sein d'un réseau de neurones pleinement connecté puis d'un réseau à convolution. Vous devez donc simplement recopier ces paragraphes de code dans le document **`pytorch_tutorial.ipynb`** (généralement un paragraphe de code par cellule jupyter-notebook). Cependant, afin de s'assurer que vous comprenez le code recopié, chaque paragraphe de code recopié doit être accompagné d'une description [en français] du contenu des principales variables ainsi que du fonctionnement globale du code.

2. **[1 point]** Du code vous a été fourni pour entraîner et tester des CNNs. Il y a les modèles `VanillaCNN`, `AlexNet`, `VGGNet` et `ResNet`. Vous pouvez les tester et les entraîner sur différentes bases de données dont `CIFAR10` et `VSHN`. Prenez le temps d'exécuter le code et d'en comprendre le contenu. L'exécution s'effectue via la commande suivante dans un terminal linux :

   ```
   python train.py --model=CnnVanilla --dataset=svhn --num-epochs=10
   ```

   Si vous voulez exécuter le code sur CPU (par exemple, sur votre laptop), vous n'avez qu'à remplacer « `use_cuda=True` » par « `use_cuda=False` » dans le fichier `train.py`. Si vous n'avez pas de GPU à votre disposition et souhaitez utiliser google colab, utilisez le fichier `train.ipynb` pour exécuter votre code.

   Une fois que vous avez compris le code, vous devez ajouter du code pour effectuer de l'augmentation de données pour CIFAR10 et SVHN. Vous devez effectuer les augmentations de données suivantes :

   - rotations aléatoires de quelques degrés,
   - des variations au niveau des contrastes et des couleurs
   - des flips horizontaux aléatoires.
   - Des crops aléatoires

   En principe, cela devrait améliorer légèrement les performances des modèles. **Pour la remise**, en plus du code, vous devez remettre un fichier `TP3.pdf` dans lequel vous précisez les endroits où vous avez ajouter du code. Vous devez également y mettre les courbes d'entraînement et de validation pour au moins 20 epochs, pour chaque base de données et pour au moins un modèle.

   NOTE : votre augmentation de données doit être activable avec l'option `--data_aug` du fichier `train.py`.

3. **[2 points]** Par la suite, vous devez implémenter le `IFT725_NET`. C'est un réseau inventé pour ce travail. Ce réseau doit contenir les blocs convolutifs que voici :

   1) Au début, quelques opérations de base du type « conv-batch-norm-relu »
   2) Ensuite au moins 1 bloc dense inspiré du modèle « denseNet) (voir https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a)
   3) suivi de quelques blocs résiduels inspirés de « resNet » (https://medium.com/ai%C2%B3-theory-practice-business/resblock-a-trick-to-impove-the-model-8ba11891c52a)
   4) suivi de quelques blocs de couches « bottleneck » avec ou sans connexion résiduelle, c'est au choix (https://stats.stackexchange.com/questions/421029/implementation-of-bottleneck-layers-in-residual-neural-networks)
   5) et au final des couches pleinement connectées.

   À noter que le code des blocs doit être dans le fichier `CNNBlocks.py`. De plus, dans le fichier `TP3.pdf`, vous devez mettre une représentation graphique de votre réseau.

4. **[4 points]** On vous a également fourni le code d'un U-Net permettant de segmenter des images IRM cardiaques en 4 classes : ventricule gauche, ventricule droit, myocarde et fond. Dans cette section, vous devez :

   (a) ajouter du code permettant d'effectuer de l'augmentation de données (rotations aléatoires, ajustement des niveaux de gris, crop aléatoires) et
   (b) vous créez un nouveau fichier « IFT725_UNET » dans lequel vous devez créer votre propre réseau de segmentation inspiré du Unet mais comprenant également des connexions résiduelles et denses. Soyez originaux! Libre à vous de déterminer la structure de votre choix.
   (c) **IMPORTANT! Vous devez effectuer une recherche d'hyperparamètres sur les serveurs de Calculs Québec.**
   (d) dans le fichier `TP3.pdf`, vous devez mettre une représentation graphique de votre réseau, spécifier où vous avez mis l'augmentation de données et donner les courbes d'apprentissage et de validation avec et sans augmentation de données. Vous devez également y mettre **le résultat de votre recherche d'hyperparamètres.**