IFT780: Travail pratique 3

Segmentation cardiaque

Le travail pratique 3 vous demandera d'effectuer de la segmentation sur le Automated Cardiac Diagnosis Challenge (ACDC) dataset[1].

Commencez d'abord par exécuter:

```
pip install -r requirements.txt
```

Les classes sont les suivantes:

1. Background
2. Ventricule droite
3. Myocarde
4. Ventricule gauche

Les données ne contiennent qu'une seule modalité (ciné-IRM) et sont acquises en 3D. Le code fourni (train.py) vous permet de lancer un entraînement pendant N epochs avec un modèle, de spécifier le learning rate, l'optimiseur et la batch size. Vous avez le droit de tout modifier dans le code pour répondre à vos besoins. Les sous-sections suivantes décrirons les "features" à ajouter au code pour gérer vos entraînements. **À noter**: toutes ces caractéristiques doivent être modifiable à même la ligne de commande !

1. Architectures (3 points)

Vous devrez implémenter trois architectures différentes, excluant celle fournie, en plus d'en concevoir une de votre cru. Vous pouvez vous inspirer du IFT780Net du TP2 si vous manquez d'inspiration. Vous devrez inclure dans votre rapport une figure représentant votre architecture, comme vu dans les notes du cours. Pour les deux autres, il vous est conseillé de fouiller un peu sur internet et dans les notes pour des architectures qui sont reconnues pour bien performer sur la segmentation.

2. Recherche d'hyperparamètres (2 points)

Vous devrez modifier le code fourni pour permettre d'effectuer une recherche d'hyperparamètres pour les modèles implémentés. Les hyperparamètres à considérer pourraient inclure le taux d'apprentissage, l'optimiseur à utiliser, ou des paramètres de vos architectures.

3. Checkpointing (1 point)

Vous devrez implémenter une façon de sauvegarder le modèle pendant l'entraînement, ainsi qu'une façon de reprendre celle-ci après un arrêt (une panne de courant ou un Ctrl-C mal placé, par exemple).

4. Data augmentation (2 points)

Vous devrez ajouter des fonctionnalités pour permettre l'ajout de plusieurs types de "data augmentation" pendant l'entraînement.

[1] https://acdc.creatis.insa-lyon.fr/

5. Bonus: Loss alternative (1 points)

Alors que la l'entropie croisée est présentement utilisée comme fonction de perte, pleins d'alternatives existent. Vous devrez implémenter une autre loss afin d'entraîner le réseau, expliquer ses avantages et les démontrer. Pour vous inspirer:[2]

6. Rapport

Pour chacune des fonctionnalités mentionnées précédemment, vous devrez mentionner dans votre rapport comment et où dans le code celles-ci ont été ajoutées, ainsi qu'un exemple d'utilisation de celles-ci et de l'impact qu'elles ont sur l'entraînement. Vous devrez aussi mentionner votre processus pour trouver la meilleure combinaison de paramètres d'entraînement qui vous donnera les meilleurs performances sur l'ensemble de test.

Finalement, vous devrez inclure les courbes d'apprentissage pertinentes ainsi qu'une analyse de celles-ci. **Votre rapport devrait être assez complet pour me permettre facilement de reproduire moi-même vos expériences.**

7. Code (2 points)

Vous serez évalués sur la qualité du code remis ainsi que sa facilité d'usage. Imaginez que vous donneriez le code à votre collègue qui n'as pas suivi le cours comme vous, mais qui aimerait tout de même entraîner des réseaux de neurones. **Je ne devrais pas avoir à modifier le code lors de la correction**. Il devra donc être possible de changer le modèle utilisé, les hyperparamètres, de faire une recherche d'hyperparamètres, etc. sans modifier le code, seulement en changeant les paramètres passés en ligne de commande. **Il ne faut rien hardcoder**.

8. Remise

Vous devrez remettre votre rapport, votre code et les poids de votre meilleur modèle afin que je puisse en tester les performances.

[2] Jadon, S. (2020, October). A survey of loss functions for semantic segmentation. In 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) (pp. 1-7). IEEE., `https://arxiv.org/pdf/2006.14822.pdf`