IFT 780 : Devoir 4

Travail par équipe de 3 ou 4

Ce travail porte sur la segmentation d'images par résonance magnétique (IRM) cardiaques. Les données ne contiennent qu'une seule modalité (ciné-IRM) et ont été acquises en 3D. Le but est de classifier des tranches 2D de ces images et ce, en quatre classes :

1. Fond
2. Ventricule droit
3. Myocarde
4. Ventricule gauche.

Veuillez vous référer aux figures 1 à 4 pour une illustration des images à segmenter et des cartes de segmentation associées. Pour ce travail, nous utiliserons la base de données ACDC (https://www.creatis.insa-lyon.fr/Challenge/acdc/) contenant les images de 100 sujets avec leur vérité terrain. Afin de vous simplifier la tâche, nous avons mis les données dans un fichier « HDF5 » que vous pouvez télécharger sur le site web du cours (https://drive.google.com/file/d/1DAYgtBNeqXcmMQjkAH4njYYm6vmvXpel/view). Vous devez le décompresser et mettre le fichier hdf5 dans le répertoire « data/ ». Si vous ne connaissez pas ce format, veuillez consulter les tutoriels suivants :

https://www.oreilly.com/library/view/python-and-hdf5/9781491944981/ch01.html
https://sh-tsang.medium.com/tutorial-creating-hierarchical-data-format-hdf5-dataset-79cfc99d2613

Le code fourni (en particulier `train.py`) vous permet entre autre de lancer l'entraînement de divers modèles pendant N *epochs*, de spécifier le *learning rate*, de sélectionner un optimiseur ainsi que la taille des *batches*. Avant d'exécuter le code, commencez par exécuter :

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

si vous travaillez sur votre ordinateur personnel équipé d'un bon GPU ou dans l'environnement Colab comme vous l'avez fait pour les travaux pratiques antérieurs. Une fois votre environnement configuré, vous pouvez exécuter la commande

```
python train.py --help
```

pour voir les options s'offrant à vous. Si vous exécutez la commande

```
python train.py --model UNet
```

un réseau de segmentation « UNet » (voir les notes de cours pour plus de détail quant à l'architecture de ce réseau) s'entraînera sur les données d'ACDC pour 10 epochs et ce, à raison d'environ 1 à 2 minutes par *epoch* sur un bon GPU.

Le code contient également les fonctionnalités nécessaires pour entraîner différents réseaux de classification d'images (CIFAR10) comme *ResNet* et *AlexNet*. **Bien que ces fonctionnalités ne fassent pas partie de ce devoir à proprement parler, nous les avons incluses afin d'illustrer ce qu'est un « Model Zoo ». Du code d'augmentation de données vous est également fourni.**

Les sous-sections suivantes décrivent les fonctionnalités à ajouter au code. À noter: toutes ces caractéristiques doivent être pilotables en ligne de commande.

1. Architectures (5 points)

Vous devez programmer les deux modèles de segmentation suivants :

- **YourUNet**
- **YourSegNet**

YourUNet : Partant du code UNet inclut au devoir, vous devez modifier l'architecture du réseau afin d'y donner une touche personnelle. Vous devez ajouter **au moins trois caractéristiques nouvelles** comme par exemple : des sorties multi-multirésolution, des couches denses, des couches résiduelles, des convolutions séparables, des couches « bottleneck », etc.

YourSegNet : Vous devez proposer une architecture originale et différente du UNet dont la structure est non-symétrique (encodeur/decodeur). Vous pouvez par exemple vous inspirer du « fullnet » ou du « DeepLabV3 » présentés en classe. Vous pouvez également vous inspirer de modèles trouvés sur internet. **Mais attention, en aucun cas vous ne pouvez copier-coller de solution provenant d'internet ou d'une autre équipe**.

NOTE : vos architectures devraient avoir une justesse (accuracy) de validation d'au moins 75%.

Vous devez aussi proposer pour chaque architecture, une nouvelle fonction de perte (*loss*) autre que l'entropie croisée.

Vous devrez inclure dans votre rapport une figure représentant vos architectures, comme dans les notes du cours.

2. *Checkpointing* (1 point)

Vous devrez programmer une façon de sauvegarder en temps réel durant l'entraînement le modèle ayant la meilleure justesse en validation. Le but ici est d'avoir une copie du meilleur modèle et ce, même si une panne de courant devait compromettre l'entraînement.

3. Augmentation de données (2 points)

Vous devrez ajouter deux types (au choix) d'augmentation de données. Un exemple de code pytorch pour l'augmentation de données CIFAR10 est disponible dans le fichier `train.py`.

4. Rapport (2 points)

Pour chacune des fonctionnalités mentionnées précédemment, vous devez spécifier dans un rapport comment et où dans le code celles-ci ont été ajoutées. Vous devez également donner les lignes de commande permettant d'exécuter votre code. Vous devrez aussi mettre des exemples d'augmentation de données (images et cibles) que vous avez développé. Vous devrez aussi inclure des courbes d'apprentissage ainsi qu'une analyse de celles-ci.

Votre rapport devrait être suffisamment complet pour permettre de **facilement** reproduire vos expériences.

Votre rapport doit avoir la structure suivante, **ni plus ni moins**:

1. Architectures
   - Liste des fichiers modifiés
   - Illustration des architectures
   - Description de la nouvelle *loss*
   - Courbes d'entraînement avec une courte description.
   - Lignes de commande pour exécuter votre code.

2. *Checkpointing*
   - Liste des fichiers modifiés
   - Lignes de commande pour exécuter votre code.

3. Augmentation de données
   - Liste des fichiers modifiés
   - Illustration de l'effet de l'augmentation de données sur quelques données
   - Description du type d'augmentation de données utilisé
   - Lignes de commande pour exécuter votre code.

Pas d'introduction, pas de conclusion, pas de texte inutile. Soyez direct et concis!

5. Code

Vous pouvez modifier tous les fichiers jugés nécessaires dont `train.py`. Seule contrainte : que le code soit propre, fonctionnel et facile d'usage. Rien ne devrait être *hardcodé*.

6. Remise

Vous devrez remettre le lien vers votre repo `gitLab`, votre rapport, votre code et les poids de vos meilleurs modèles. **Ne soumettez pas de fichiers temporaires ni de fichiers de données.**

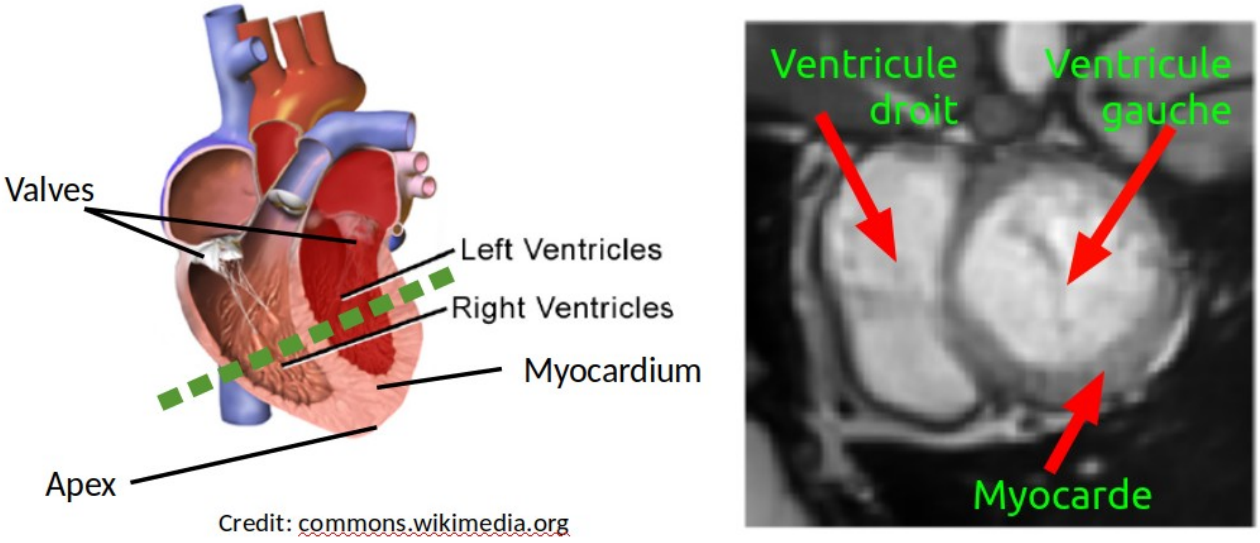

Figure 1. (Gauche) Anatomie cardiaque. (Droite) IRM cardiaque prise le long du trait pointillé vert (à gauche).

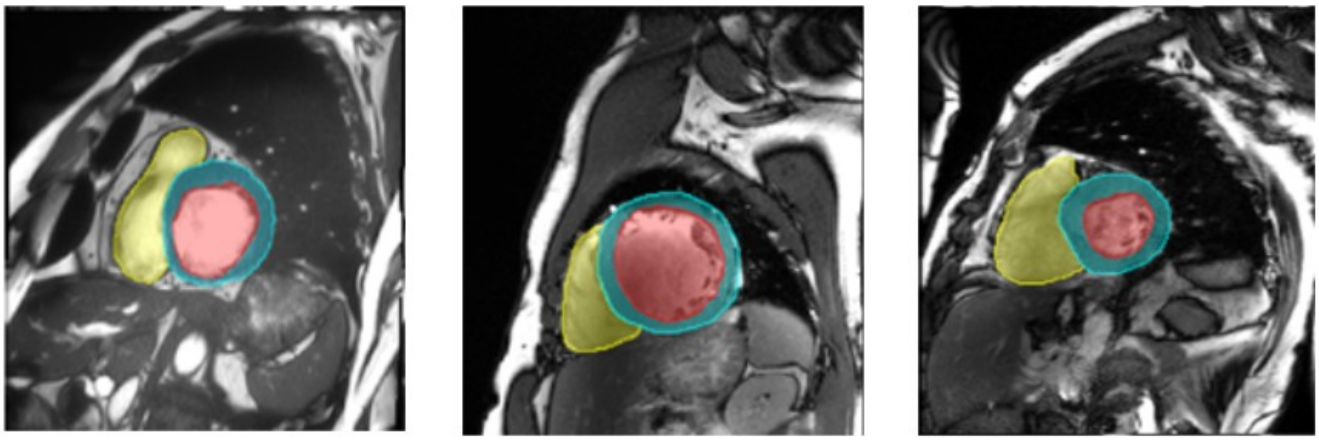

Figure 2. Illustration de trois résultats de segmentation superposés à leur image d'origine

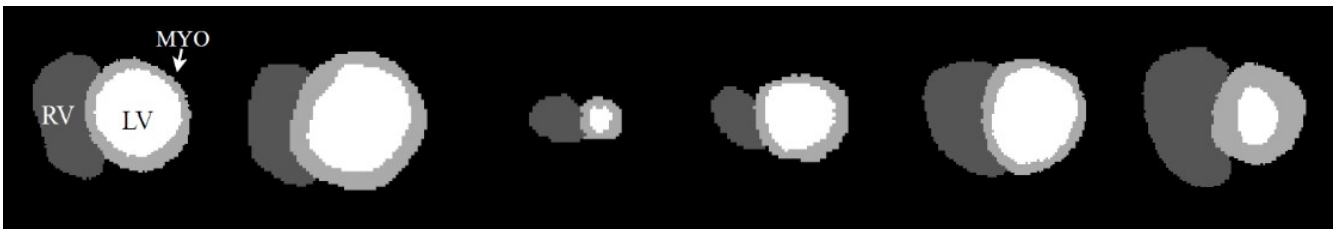

Figure 3. Illustration de six cartes de segmentation produites par un réseau de segmentation.

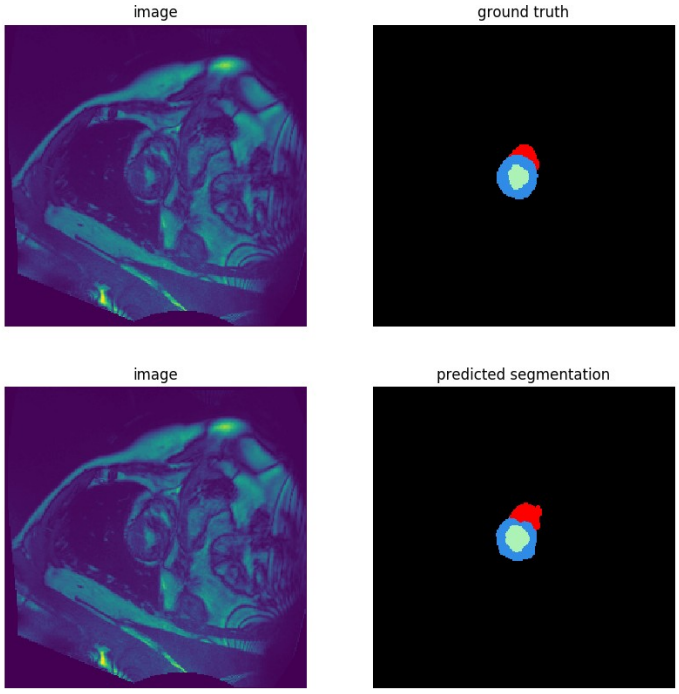

Figure 4. Résultat typique obtenu avec un réseau ayant une justesse de validation de 79%.

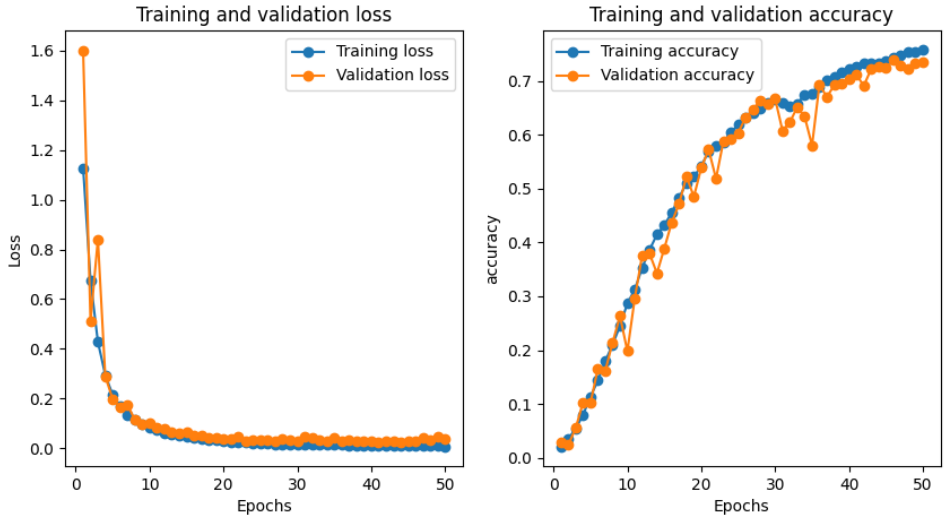

Figure 5. Courbes d'entraînement d'un bon réseau de segmentation après 50 *epochs*.