Devoir 4

par

Alexandre Ferland : FERA2409
Yohan Finet : FINY2701
Hugo Freitas Costinha : FREH1101

présenté à
Pierre-Marc Jodoin

dans le cadre du cours

IFT780: Réseaux neuronaux

DÉPARTEMENT D'INFORMATIQUE
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 17 avril 2023

Table des matières

Architecture

Liste des fichiers modifiés

train.py

Modification du paramètre –loss pour choisir quelle loss utiliser parmi Dice, Jaccard, Hinge et Entropie Croisée

Complétion concernant l'argument model pour utiliser les nouvelles architectures.

Création d'une variable loss fn pour passer la loss choisi en argument au Model Trainer

yourUNET.py

Implémentation d'une architecture type UNet

yourSegNet.py

Implémentation d'une architecture type SegNet

CustomLosses.py

Implémentation des loss utilisables

Illustration des architectures

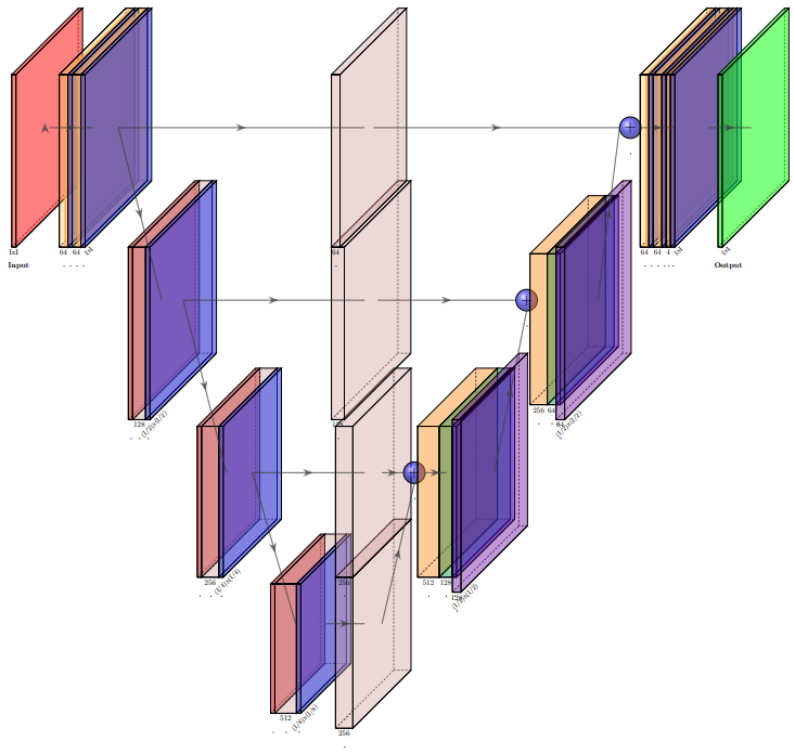


FIGURE 1 – yourUNET

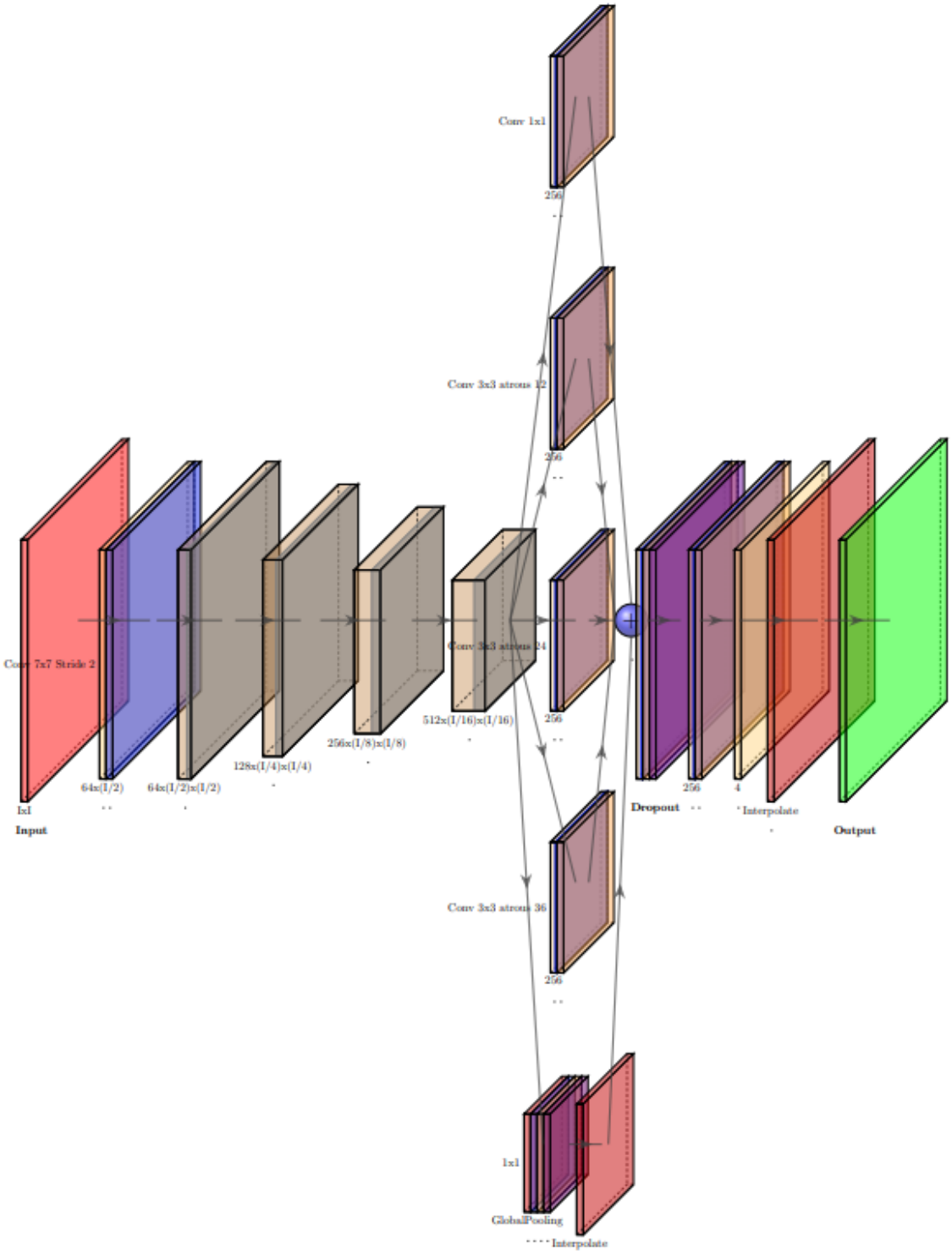


FIGURE 2 – yourSegNet

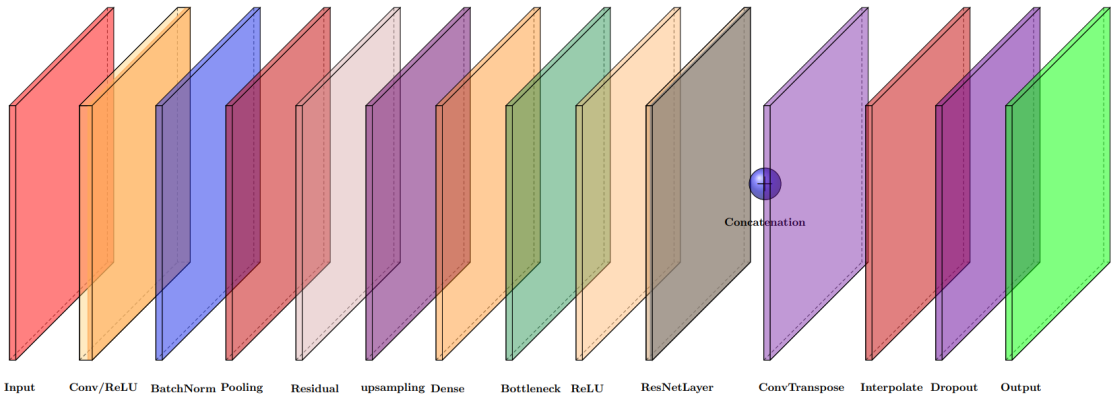


FIGURE 3 – Légendes

Vous pourrez trouver toutes ces illustrations en format pdf dans la soumissions Turnin.

Description des nouvelles loss

JaccardLoss

Loss de type IoU.
Indice de Jaccard :

$$J = \frac{|Output \cap Target|}{|Output \cup Target|}$$

JaccardLoss :

$$J_\sigma = 1 - J$$

HingeLoss

$$Hinge\_loss(x, y) = \frac{\sum_i max(0, margin - x[y] + x[i])}{x.size(0)}$$

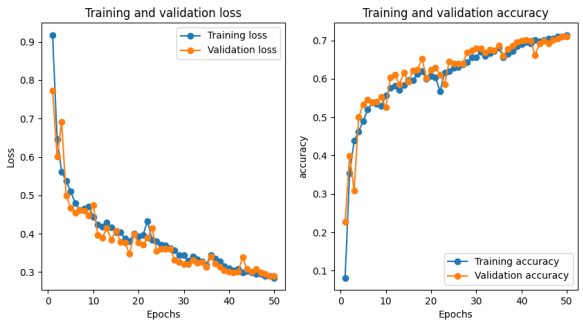
DiceLoss

$$DiceLoss = 1 - (\frac{1}{n\_class} \sum_{class} \frac{2TP_{class}}{2TP_{class} + FP_{class} + FN_{class}})$$

Courbes d'entraînement avec une courte description


```
Epoch: 12 of 20
100%|
Validation loss 0.234
Validation accuracy 0.766
Epoch: 13 of 20
100%|
Validation loss 0.274
Validation accuracy 0.726
Epoch: 14 of 20
100%|
Validation loss 0.236
Validation accuracy 0.764
Epoch: 15 of 20
100%|
Validation loss 0.212
Validation accuracy 0.788
Epoch: 16 of 20
100%|
Validation loss 0.227
Validation accuracy 0.773
Epoch: 17 of 20
100%|
Validation loss 0.311
Validation accuracy 0.689
```

FIGURE 4




```
Epoch: 46 of 50
100%|
Validation loss 0.309
Validation accuracy 0.691
Epoch: 47 of 50
100%|
Validation loss 0.299
Validation accuracy 0.701
Epoch: 48 of 50
100%|
Validation loss 0.296
Validation accuracy 0.704
Epoch: 49 of 50
100%|
Validation loss 0.290
Validation accuracy 0.710
Epoch: 50 of 50
100%|
Validation loss 0.289
Validation accuracy 0.711
Finished training.
```

FIGURE 5

Lignes de commande pour exécuter le code

```
$ python3 train.py --model=yourUNet --num-epochs=20 --batch_size=4 --save_train=False
  --lr=0.02 --loss=Dice --optimizer=SGD

$ python3 train.py --model=yourSegNet --num-epochs=50 --batch_size=8 --save_train=False
  --lr=0.015 --loss=Dice --optimizer=SGD
```

Checkpointing

Liste des fichiers modifiés

train.py

Un ajout de paramètre à été fait pour activé ou non la sauvegarde du modèle durant l'entrainement (Voir la prochaine section sur les lignes de commandes)

CNNTrainTestManager.py

C'est dans la classe CNNTrainTestManager que nous avons l'option pour sauvegarder le model durant l'entrainement. Un nouveau "backup" est créé et permet de sauvegarder le model ainsi qu'un fichier json avec les informations suivantes :

- best_epoch : Le meilleur epoch jusqu'a maintenant
- current_epoch : L'epoch ou le modèle est rendu
- best_val_acc : La meilleur accuracy de validation
- val_loss : La loss de validation par rapport à la meilleur accuracy de validation
- train_loss : La loss d'entrainement par rapport à la meilleur accuracy de validation
- train_acc : L'accuracy d'entrainement par rapport à la meilleur accuracy de validation

Lignes de commande pour exécuter le code

De base, la sauvegarde est activé par défaut, mais il est possible de modifier ce comportement avec l'argument *–save_train*. Les deux valeur possible sont : "True" ou "False" et la valeur par défaut est "True".

```
$ python train.py --save_train=True
```

Augmentation de données

Liste des fichiers modifiés

train.py

Il s'agit du seul fichier que nous avons modifié pour appliquer notre augmentation de données.

Illustration de l'effet de l'augmentation de données sur quelques données

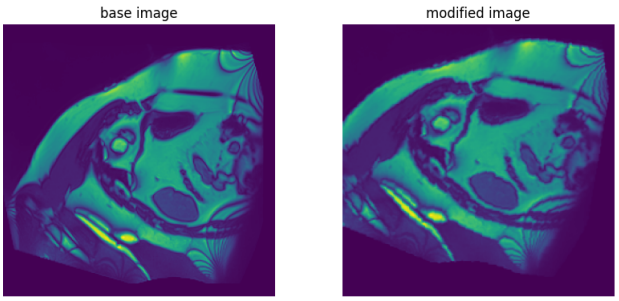


FIGURE 6

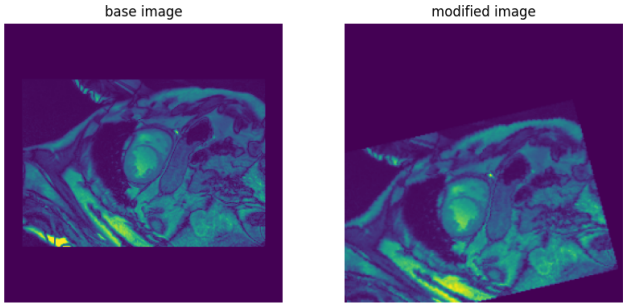


FIGURE 7

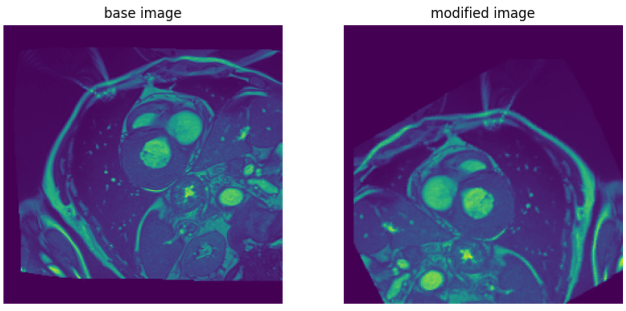


FIGURE 8

Description du type d'augmentation de données utilisé

- RandomHorizontalFlip(p=0.5) : Permet de faire un miroir de l'image sur l'axe horizontal.
- RandomRotation(degrees=30) : Permet de faire une rotation de plus ou moins 30 degrés
- RandomResizedCrop(size=256, scale=(0.9, 1.0), antialias=False) : Recadre l'image pour faire une image de moins de pixels avec un recadrage entre 0.9 et 1.0 fois la taille de l'image

Lignes de commande pour exécuter le code

Il suffit d'ajouter la l'option data_aug pour activer l'augmentation de données.

```
$ python train.py --data_aug
```