nnUNet

Read me to use nnUNet

Please cite the following paper when using **nnUNet**

Isensee, F., Jaeger, P.F., Kohl, S.A.A. et al. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation." Nat Methods (2020). https://doi.org/10.1038/s41592-020-01008-z

If you have questions or suggestions, feel free to open an issue at https://github.com/MIC-DKFZ/nnUNet

Cette notice détails tout le nécessaire à savoir pour utiliser la méthode nnUNet.
Par défaut, il y a 1000 epochs pour l'entrainement. Il est tout de même possible de modifier le nombre d'epochs si vous le souhaitez. Le script est fait pour prendre en compte les images au format nifti et au format tiff.
Il y a 3 commandes à connaitre et à taper dans un terminal. Avant de commencer, vous devez vous positionner dans le bon répertoire. C'est à dire dans le répertoire `configure_docker_for_nnUNet`.
Si vous utilisez ce code dans un autre ordinateur, vous devez connaitre le répertoire dans lequel vous avez copier le dossier `configure_docker_for_nnUNet` et utiliser la commande `cd` pour y accéder.
Sinon, si vous êtes sur l'ordinateur de Tristan, tapez la commande `cd mnt/sdb2/Adama/nuclei_benchmark/nnUNet/configure_docker_for_nnUNet` pour accéder au dossier. Une fois à l'intérieur du dossier, suivez les instructions suivantes.

Note :

Si vous souhaitez choisir un nombre d'epochs, vous devez modifier le fichier `nb_epochs.py` qui est situé dans le répertoire courant. L'image ci-dessous montre le contenu du fichier.

configure_docker_for_nnUNet > nb_epochs.py

```
1  class nnUNetTrainer_Experimental(nnUNetTrainerV2) :
2      def __init__(self, plans_file, fold, output_folder=None, dataset_directory=None, batch_dice=True, stage=None,
3                   unpack_data=True, deterministic=True, fp16=False):
4          # Appelez au constructeur de la classe mère
5          super().__init__(plans_file, fold, output_folder, dataset_directory, batch_dice, stage, unpack_data, deterministic, fp16)
6          self.max_num_epochs = 1000
```

A la sixième ligne de ce fichier, le nombre d'epochs est spécifié (1000 par défaut). Entrez le nombre d'epochs que vous souhaitez.

Construction de l'environnement Docker

La première commande vous permettra de construire l'environnement docker nécessaire à l'exécution de nnUNet. Pour lancer cette commande, vous devez connaitre les chemins vers les images du dataset que vous allez utiliser, le chemin vers le dossier dans lequel vous voulez stocker vos prédictions et votre modèle ainsi que le nom de la tâche qui correspond à la partie segmentée en

général. (Ex : Nucleus pour une segmentation du noyau, Chromocenter pour une segmentation des chromocentres).
Une fois que vous disposez de ces informations, tapez la commande `./build.sh imagesTr_path labelsTr_path output_path task_name` en remplaçant :
imagesTr_path par le chemin vers le répertoire des images utilisées pour l'entrainement;
labelsTr_path par le chemin vers le répertoire des labels utilisés pour l'entrainement.
output_path par le chemin vers le répertoire dans lequel vous voulez sauvegarder les prédictions et le modèle entrainé.
task_name par le nom de la tache.

Entrainement du modèle de nnUNet

Pour l'entrainement, il existe un fichier `.csv` qui sera utilisé pour diviser le dataset en données d'entrainement et de test.
Si vous ne souhaitez pas faire d'entrainement et que vous voulez juste utiliser un modèle déjà existant pour effectuer des prédictions, alors passez à la Section directement à la **Prédiction**
Pour entrainer le modèle de nnUNet tapez la commande suivante :
`./train_nnUNet.sh imagesTr_path labelsTr_path task_name` en remplaçant :
imagesTr_path par le chemin vers le répertoire des images utilisées pour l'entrainement.
labelsTr_path par le chemin vers le répertoire des labels utilisés pour l'entrainement.
output_path par le chemin vers le répertoire dans lequel vous voulez sauvegarder les prédictions et le modèle entrainé.
task_name par le nom de la tache.

Attention, les paramètres doivent être les mèmes que pour la commande précédente. Vous ne pouvez pas indiquer des chemins de répertoires différents pour la commande ./build.sh et pour la commande ./train_nnUNet.sh.

Prédiction

Après avoir exécuté les deux commandes précédentes et après avoir attendu que l'entrainement termine, vous pouvez maintenant effectuer des prédictions en tapant la commande suivante :
`./make_predictions.sh model_path output_path task_name` en remplaçant :
model_path par le chemin vers le répertoire contenant le modèle. Le modèle doit etre dans un dossié nommé `nnUNet_trained_models`. Donc `model_path` doit etre remplacé par le chemin vers le répertoire contenant le dossier `nnUNet_trained_models`.
output_path que vous avez spécifié. par le chemin vers le répertoire dans lequel vous voulez sauvegarder les prédictions.
task_name par le nom de la tache.

Les prédictions seront donc sauvegardées dans le répertoire **output_path** que vous avez spécifié.

Remarques

Si vous voulez tester nnUNet sur des images au format tiff, vos données doivent être stockées dans des dossiers avec des noms spécifiques.
imagesTr : Dossier contenant les images de train
labelsTr : Dossier contenat les labels des images de train.
imagesTs : Dossier contenant les images de test.
labelsTs : Dossier contenant les labels des images de test
Ce script marchera à condition que vous disposiez d'un fichier `dataset.json` conforme à vos données. Le fichier `.json` doit être dans le même répertoire que les dossiers imagesTr, imagesTs et labelsTr.
Consultez la documentation officiel de nnUNet pour plus de détails. (https://github.com/MIC-DKFZ/nnUNet)