

## UERB Médecine / DI 5 : Option Imagerie Médicale

### TP – Projet : MONAI, un Framework python pour mettre en œuvre de l'apprentissage profond sur des données d'images médicales.

#### 1. Contexte général

**MONAI : Medical Open Network for AI**

MONAI est un Framework [open-source](#) pour l'apprentissage profond en imagerie médicales, il fait partie de l'écosystème [PyTorch](#).

L'ambition d'un tel Framework est de:

- Développer une communauté d'utilisateurs en recherche académique, industrielle et clinique qui collabore sur un même projet,
- Créer des workflows de « Deep Learning » pour l'analyse d'images médicales, du début jusqu'à la fin,
- Fournir aux chercheurs une manière optimisée et standardisée de créer et d'évaluer des modèles de « Deep Learning ».

#### Features

---

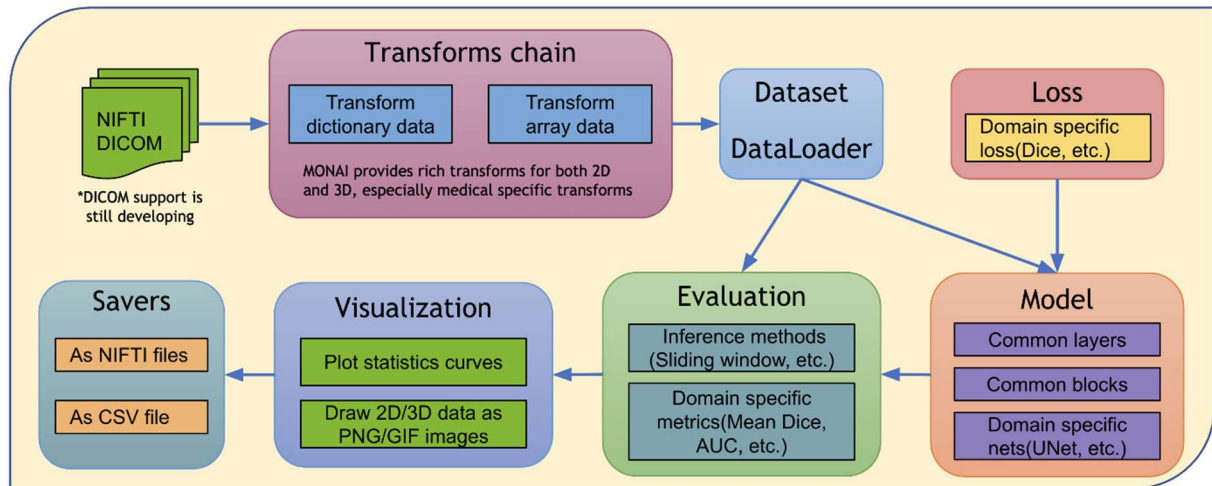
- flexible pre-processing for multi-dimensional medical imaging data;
  - compositional & portable APIs for ease of integration in existing workflows;
  - domain-specific implementations for networks, losses, evaluation metrics and more;
  - customizable design for varying user expertise;
  - multi-GPU data parallelism support.
- 

Avec un tel Framework il est possible de :

- Démarrer avec des réseaux pré-entraînés en utilisant les fonctions prédéfinies fournies,
- Adapter des réseaux existants pour traiter vos données images,
- Construire rapidement de nouvelles architectures pour des problèmes spécifiques d'analyse,
- Lancer l'entraînement sur une architecture personnalisée
- Résoudre des problèmes de segmentation d'organes ou de structures bien identifiées.

- Website: <https://monai.io/>
- API documentation: <https://docs.monai.io>

## Architecture générale :



**Research** - implementation of recently published research in medical imaging, such as COVID-19 models, automatic model parallelism

Model Parallel

COVID-19 CT Image Segmentation

Other Topics

**Examples & Application** - show highlight features and integration with other packages

Segmentation

Classification

Perf Acceleration

Modules Demo

Public Datasets

**Workflows** - engines, metrics and event-handlers that are compatible with PyTorch ignite APIs, support AMP and distributed data parallel

### Engines

SupervisedTrainer GANTrainer  
SupervisedEvaluator  
EnsembleEvaluator

### Event-Handlers

CheckpointLoader CheckpointSaver TensorBoardHandlers  
ValidationHandler MetricLogger SegmentationSaver  
ClassificationSaver StatsHandler LrScheduleHandler

### Metrics

MeanDice  
ROCAUC  
ConfusionMatrix

**Components** - independent modules that can be integrated into PyTorch programs directly

### Data

CacheDataset  
SmartCacheDataset  
PersistentDataset  
ImdbDataset  
ArrayDataset  
GridDataset  
PatchDataset  
Image readers  
Cross validation API

### Savers

NiftiSaver PNGSaver  
CSVSaver

### Inferers

SimpleInferer  
SlidingWindowInferer

### Optimizers

Novograd, Layer wise LR

### Losses

DiceLoss + extensions  
FocalLoss TverskyLoss

### Visualize

Plot 3D / 2D images  
GradCam feature map

### Networks

UNet / DynUNet / VNet  
DenseNet / SENets  
SegResNet / AHNet  
Common Blocks  
Common Layers

### Transforms

Spatial Transforms  
Intensity Transforms  
IO Transforms  
Utility Transforms  
Post Transforms

MeanDice  
Confusion matrix  
Occlusion sensitivity

### Metrics

Surface distance  
Hausdorff distance  
ROCAUC

## 2. Installation de l'environnement

### 2.1 Python et MONAI

Vous pouvez choisir votre OS (Unix/Windows) pour installer l'environnement python qui va recevoir l'installation de MONAI. Nous décrivons une manière de procéder ici, basée sur l'utilitaire miniconda3 :

- 1) Créer un environnement en utilisant « conda3 » :

```
conda create -n MONAI python=3.9
```

- 2) Activer l'environnement

```
conda activate MONAI (sous unix)
```

- 3) Installer le package MONAI, from PyPi

```
pip install -r https://raw.githubusercontent.com/Project-MONAI/MONAI/master/requirements-dev.txt
```

```
pip install itk
```

```
pip install matplotlib
```

```
pip install scikit-learn
```

A ce stade, vous devriez avoir un environnement python fonctionnel, avec le module MONAI également fonctionnel.

### 2.2 Installation de Jupyter Notebook

Les Notebook jupyter (.ipynb) sont des documents intégrant à la fois du texte formaté et des commentaires. Ils sont très utiles pour partager du code et des résultats de calculs. Nous vous proposons ici de mettre en œuvre un serveur Jupyter Notebook local afin d'être en mesure d'exécuter des notebooks en se connectant à un noyau python via le navigateur. Cette méthodologie permet d'exploiter des architectures distantes, avec des sorties en graphiques (satiques) ou images.

## conda

```
conda install -c conda-forge jupyterlab
```

## pip

```
pip install jupyterlab
```

Il existe également une implémentation par google (google collab) qui peut servir pour avoir accès à une infrastructure matérielle GPU pour faire du DL assez facilement. Le principal problème de l'environnement Google collab est sa volatilité. En effet, un environnement ainsi qu'un espace temporaire est alloué à l'utilisateur. Dans ce TP, vous serez invités à tester l'exécution de vos pipelines d'entraînement sur cette architecture. (<https://colab.research.google.com>)

### 2.3 Vérifier la disponibilité de GPUs

La disponibilité de GPUs nvidia se vérifie par la commande : `nvidia-smi`

Repérer le matériel disponible, son nom, la version de cuda et sa VRAM.

## 3. Partie I : Découverte et mise en œuvre d'exemples

### 3.0 Prise en main du Framework au travers d'un exemple simple.

<https://github.com/Project-MONAI/tutorials/tree/master/modules/engines>

- Reprenez cet exemple, identifiez et découpez les différentes phases et exécutez-les pas à pas dans un jupyter notebook pour en visualiser les différentes étapes.
- Recherchez et notez les grandes étapes d'un pipeline classique d'apprentissage sous pyTorch.
- Recherchez dans cet exemple la structure du réseau et sa configuration. Il sera parfois nécessaire de naviguer dans les sources du framework MONAI.

### 3.1 Jupyter Notebooks : connexion à Tensorboard pour le suivi temps réel de la phase d'apprentissage

Tensorboard est un outil intéressant et complet pour obtenir des sorties visuelles au cours de la phase d'apprentissage qui peut parfois être longue. Lisez attentivement ce tutoriel et mettez-le en œuvre. Testez avec l'exemple précédent.

[https://pytorch.org/tutorials/intermediate/tensorboard\\_tutorial.html](https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html)

### 3.2 La classification d'images médicales 2D

#### 3.2.1 Objectif

L'objectif ici est de produire une classification d'images médicales issues d'une banque de données connue. Pour plus de facilité cet exemple se base sur des données images 2D (issues

d'images volumiques 3D). Comme le volume d'image nécessaires est important, les temps d'apprentissages sont réduits avec des images 2D (1 coupe).

### 3.2.2 Données

En suivant les étapes décrites dans la page ci-dessous, récupérez les données nécessaires, puis lancez l'exécution sur ces données. En navigant dans le code, retrouvez l'architecture de l'application, isolez les différents modules mis en œuvre.

<https://colab.research.google.com/drive/1wy8XUSnNWlhDNazFdvGBHLfdkGvOHBKe>

### 3.2.3 Travail à faire

- Observation et compréhension du code
  - Recherchez et notez les grandes étapes d'un pipeline classique d'apprentissage sous pyTorch.
  - Recherchez et notez l'architecture du réseau de neurone en jeu ici.
  - Essayez de comprendre l'organisation des couches et leur fonctionnement.
  - Repérez la fonction « loss », de quelle méthode s'agit il ?
- Mise en œuvre de l'apprentissage : Essais
  - Repérez la phase d'augmentation des données ? En quoi consiste t'elle ?
  - Faites un essai en local en utilisant 1) le CPU, 2) le GPU (via votre serveur local jupyter notebook)
  - Faites un essai avec l'environnement google collab (avec GPU activé) et comparez les performances.
  - Faites des essais en modifiant les paramètres suivants:  
(Vous conserverez les résultats notamment la valeur de loss au cours des iterations pour pouvoir comparer les résultats.)
    - Expliquez ce que sont les batchs. Modifiez les tailles de batch,
    - Expliquez ce que sont les epochs. Modifiez le nombre d'epoch,
    - Expliquez ce qu'est le learning rate. Modifiez le et observez.
    - Modifiez le ratio d'échantillons utilisés en validation/tests
    - Modifiez la fonction loss, parmi celles disponibles.
    - Modifiez l'optimizer et conservez les résultats pour comparer les performances (en temps et en précision)
  - Analysez les résultats précédents et commentez.
- Ecrivez un script python (hors jupyter notebook) qui permettra de classifier une image fournie en entrée. Le script copiera en sortie l'image classifiée dans un dossier du nom de la classe.