REUNION CLIPS 27/02

PRESENTATION DU TRAVAIL

- 1. Réussite du fonctionnement du programme sans Docker
 - → Assez long : si 2 fichiers PET, le programme essaie de segmenter l'un puis l'autre
 - → Réussi à utiliser Slicer pour visualiser les segmentations
 - → Prochaine étape : Produire les segmentations manquantes et obtenir les métriques manquantes
 - → 1^{er} test effectué sur 11011101221002 (vérifier le résultat mais semble ne pas fonctionner)
- 2. Lecture du code d'entrainement : problème de documentation

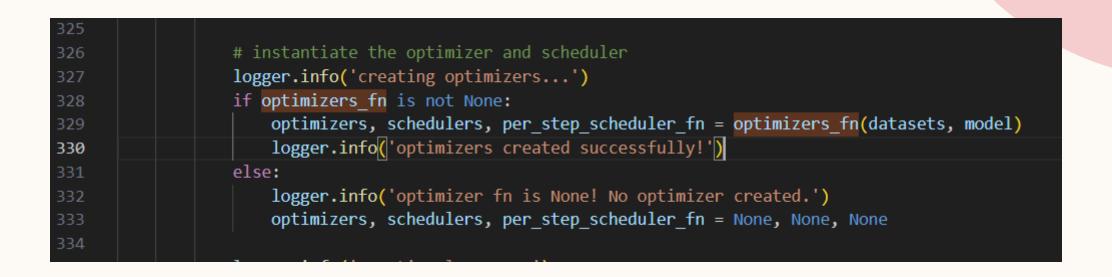
PRESENTATION DU TRAVAIL

```
class TrainerV2:
   def init (
            self,
           callbacks per batch=None,
           callbacks per batch loss terms=None,
           callbacks per epoch=default per epoch callbacks(),
           callbacks pre training=default pre training callbacks(),
           callbacks post training=default post training callbacks(),
           trainer callbacks per batch=trainer callbacks per batch,
           run epoch fn=epoch train eval,
           logging level=logging.DEBUG,
            skip eval epoch 0=True):
       Args:
           callbacks per batch:
           callbacks per batch loss terms:
           callbacks per epoch:
           callbacks_pre_training:
           callbacks post training:
           trainer callbacks per batch:
           run epoch fn:
            skip eval epoch 0: if ``True``, validation/test will not be run for epoch 0
```

```
class TrainerV2:
                                                                                     > optimizers fn
                                                                                                          Aa ab ** 2 of 10
         def fit(self,
                 options,
                  datasets,
                  model: nn.Module,
                  optimizers fn,
                 losses fn=default sum all losses,
                  loss creator=create losses fn,
                  log path=None.
                  with final evaluation=True,
190
                 history=None,
                 erase logging folder=True,
                 eval every X epoch=1) -> RunMetadata:
              Fit the model
                 datasets: a functor returning a dictionary of datasets. Alternatively, datasets infos can be specified.
                              `inputs fn` must return one of:
                              * datasets: dictionary of dataset
                              * (datasets, datasets infos): dictionary of dataset and additional infos
                             We define:
                              * datasets: a dictionary of dataset. a dataset is a dictionary of splits.
                               a split is a dictionary of batched features.
                              * Datasets infos are additional infos useful for the debugging of the
                               dataset (e.g., class mappings, sample UIDs). Datasets infos are
                               typically much smaller than datasets should be loaded in
                                loadable in memory
                  model: a `Module` or a `ModuleDict
                  losses fn:
                  log path: the path of the logs to be exported during the training of the model.
                     if the `log path` is not an absolute path, the options.workflow options.logging directory
                     is used as root
                  with final evaluation:
                  erase logging folder: if `True`, the logging will be erased when fitting starts
                  eval every X epoch: evaluate the model every `X` epochs
```

A-t-on des nouvelles de Vancouver?

PRESENTATION DU TRAVAIL



DATE DE PROCHAINE RÉUNION?