Calcul intensif : Identification de deux plateformes pour manipuler des modèles deep learning facilement

Université d'Angers - 21 octobre 2022

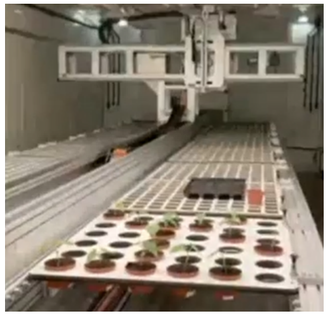
Herearii Metuarea (LARIS-INRAe)

Sommaire

1. Motivation
2. DEEPaaS
3. Napari
4. Conclusion

Motivation

Processus d'analyse des données expérimentales

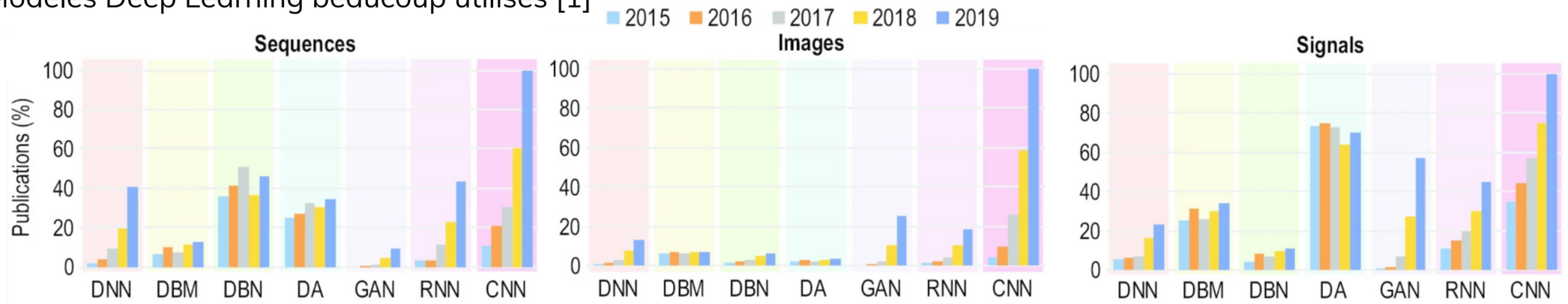


Acquisition de données
Phénotypage de semence

Traitement des données
Machine/ Deep learning

Analyse de données
Outil informatique

Modèles Deep Learning beaucoup utilisés [1]



Nombre de publications incluant un modèle de deep learning [2]

[1] O'Mahony, Niall, et al. "Deep learning vs. traditional computer vision." Science and information conference. Springer, Cham, 2019.

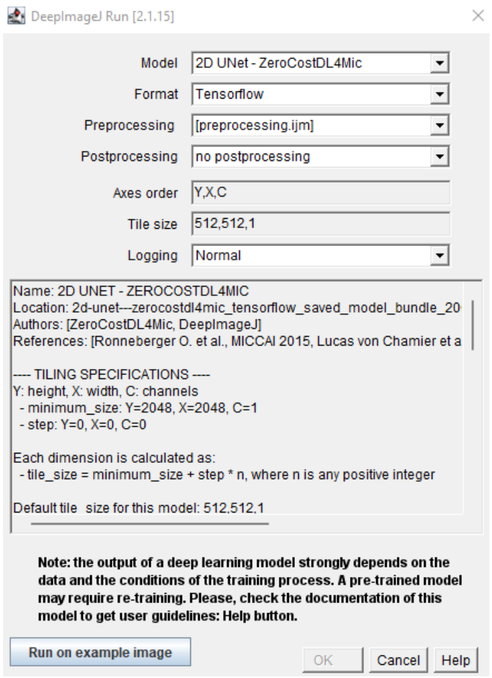
[2] Mahmud, Mufti, et al. "Deep learning in mining biological data." Cognitive computation 13.1 (2021): 1-33.

Motivation

Deux plateformes pour manipuler des modèles deep learning

DeepImageJ

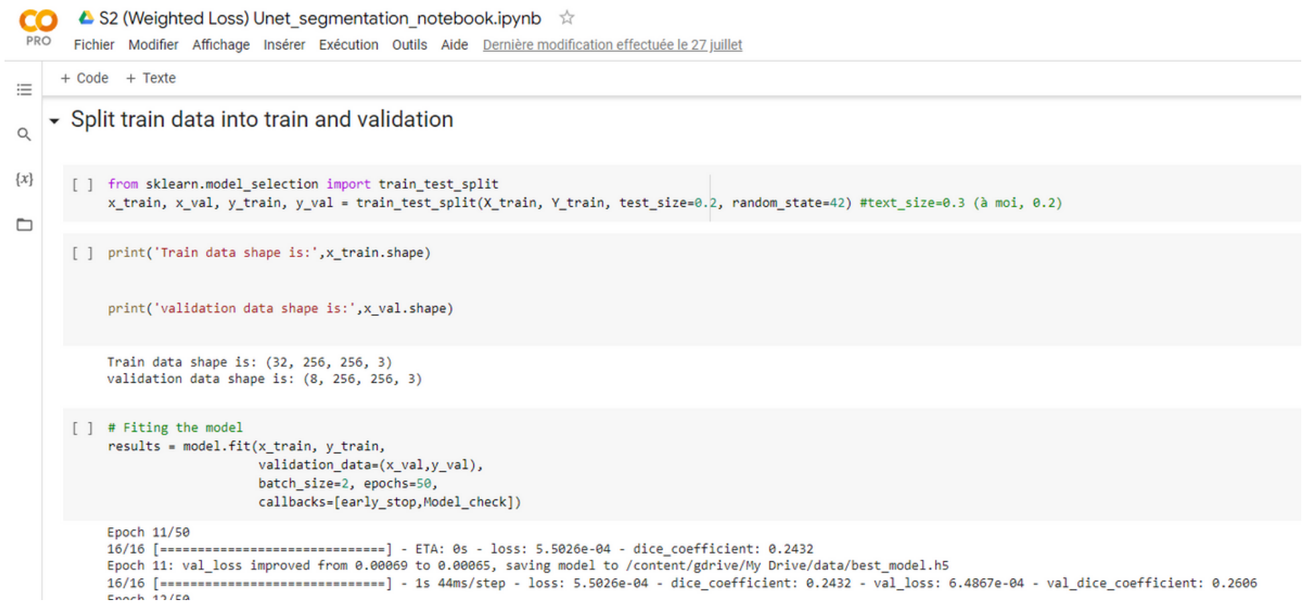
Utiliser un modèle pré-entrainé à partir d'un plugin



Plugin structuré en langage Java
Impossible de personnaliser le plugin

Google Colaboratory

Plateforme pour concevoir et manipuler des modèles deep learning à partir du cloud Google



Nécessité d'un savoir-faire en programmation
Droit de lecture et d'exploitation des données personnelles par Google

Deux plateformes pour manipuler les modèles DL

Utilisation des librairies en Python 

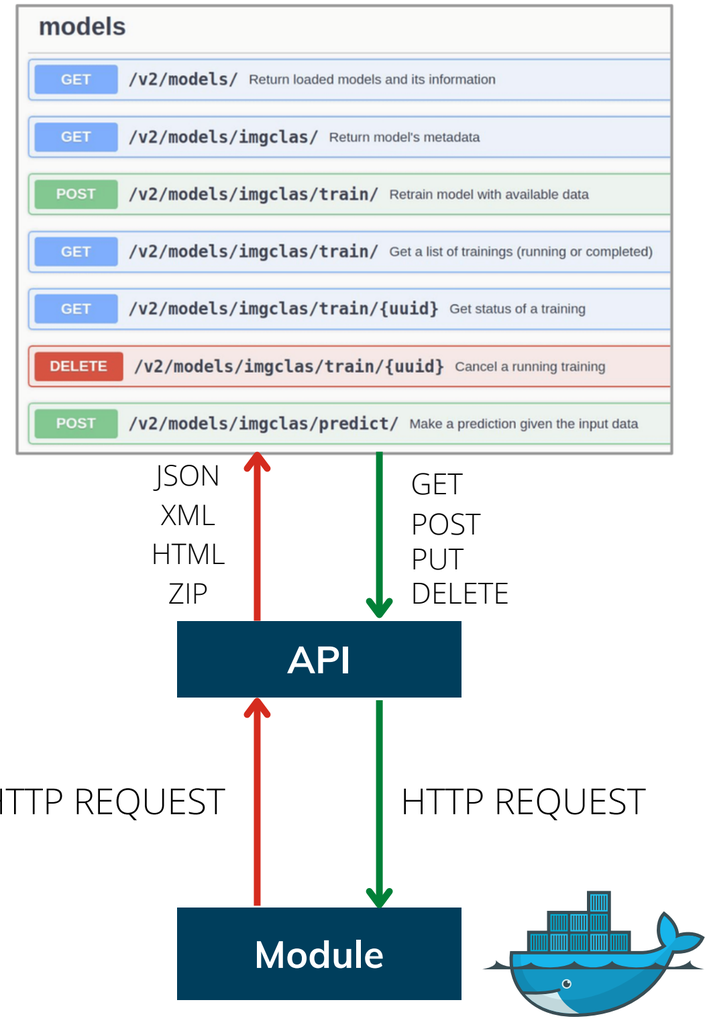
Interface utilisateur pour manipuler des modèles DL sur un cloud européen

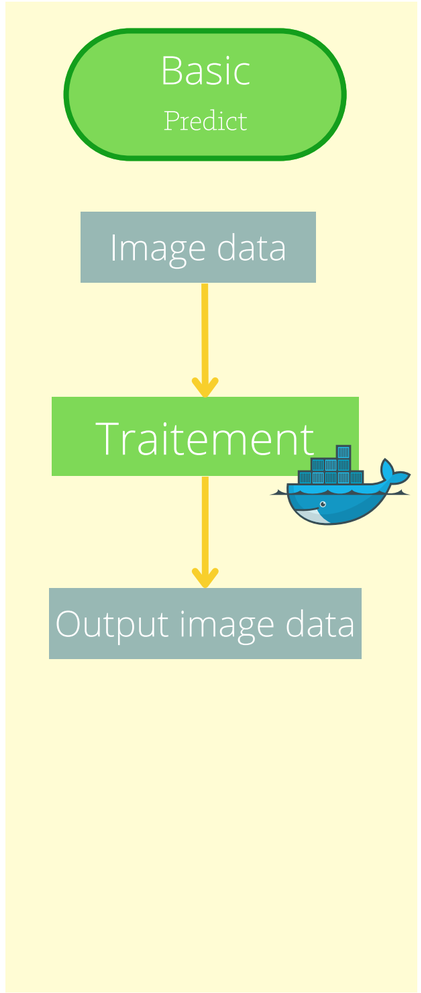
Interface utilisateur pour traiter et analyser des images 2D/3D

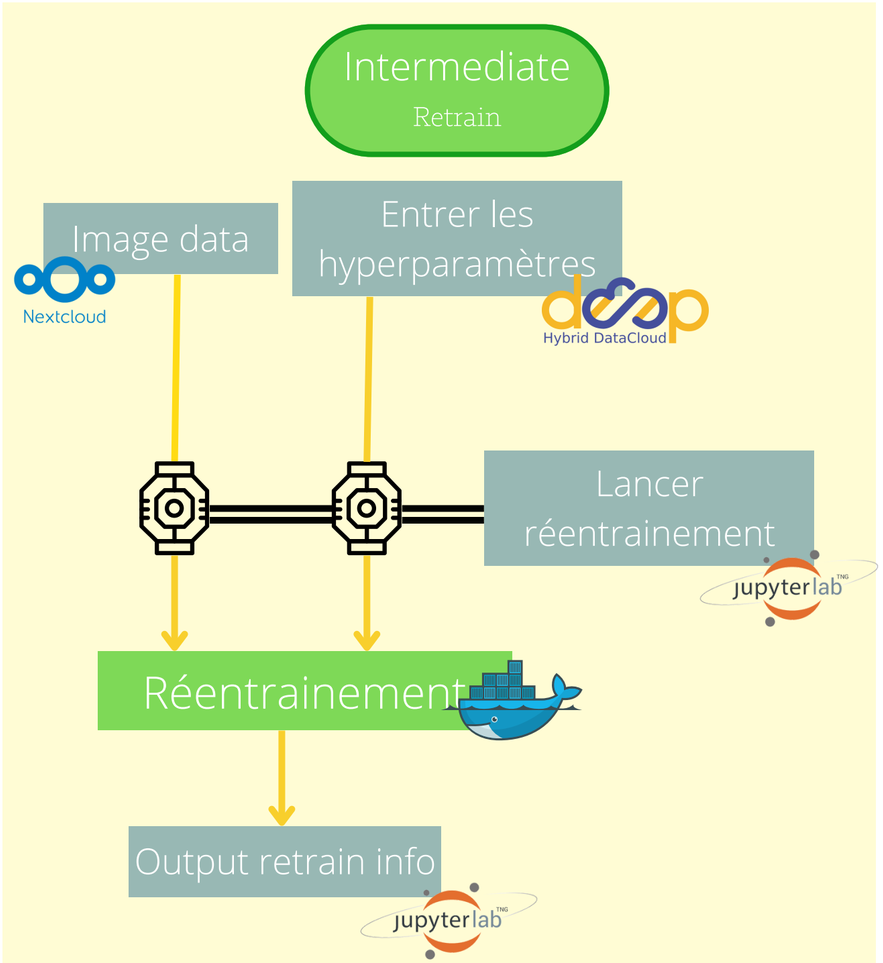
DEEPaaS

Napari

Fonctionnement de la plateforme : REST API



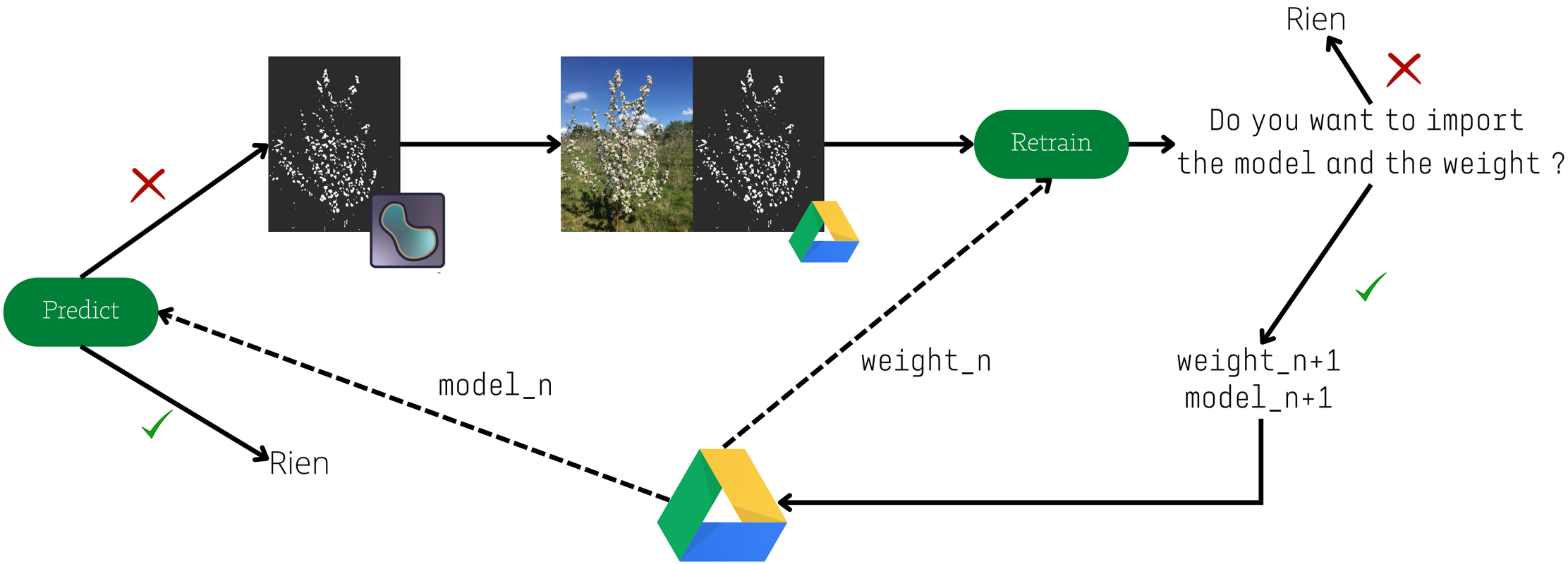




Tutoriel d'utilisation

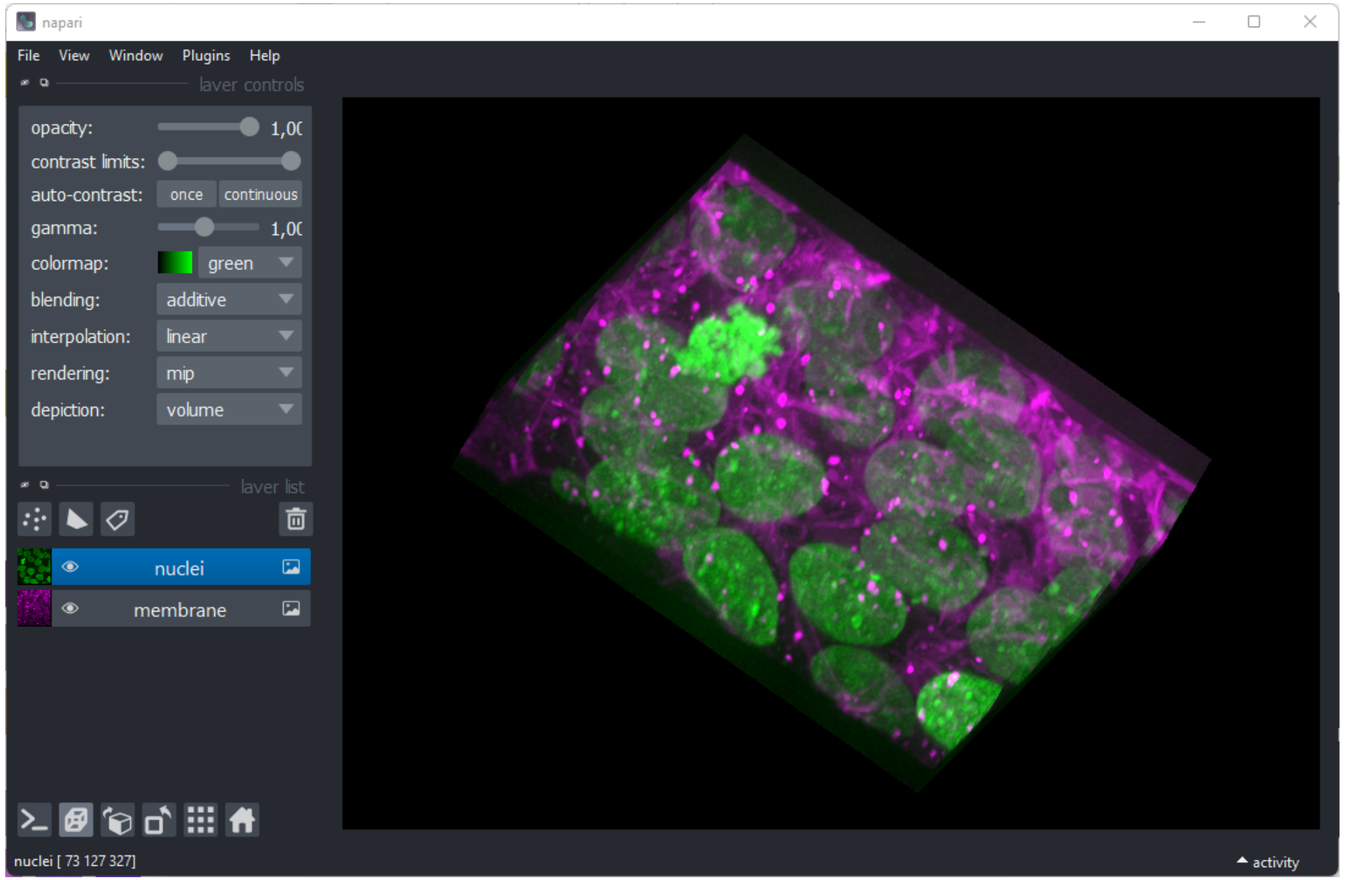
Module de réentrainement d'un modèle collaboratif

Vincent Nègre
INRAE



DEEPaaS

Napari



Visionneuse de données à plusieurs dimensions en python

open-source, community-developed

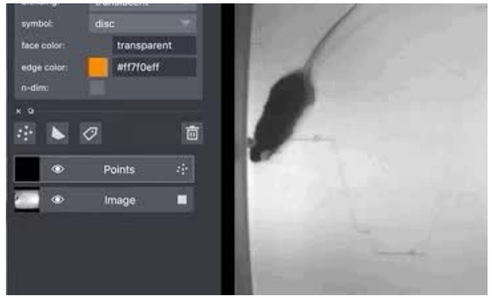
napari

Traitement de données lourd DASK

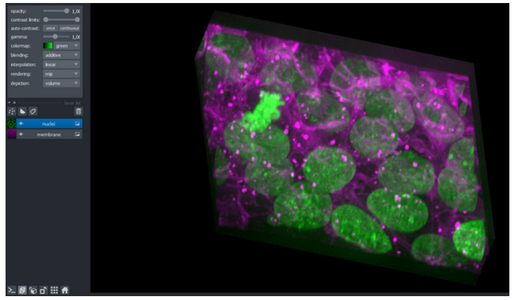
Visualisation 2D/3D VisPy

Interface utilisateur Qt

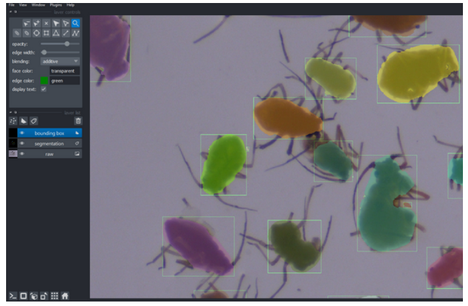
Manipulation de données NumPy



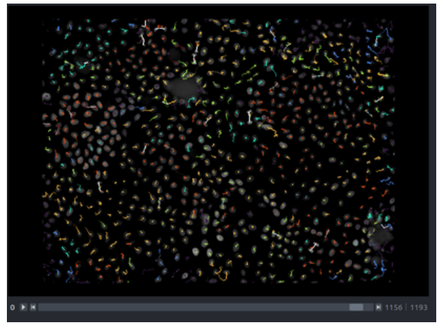
Annotation



Process heavy data



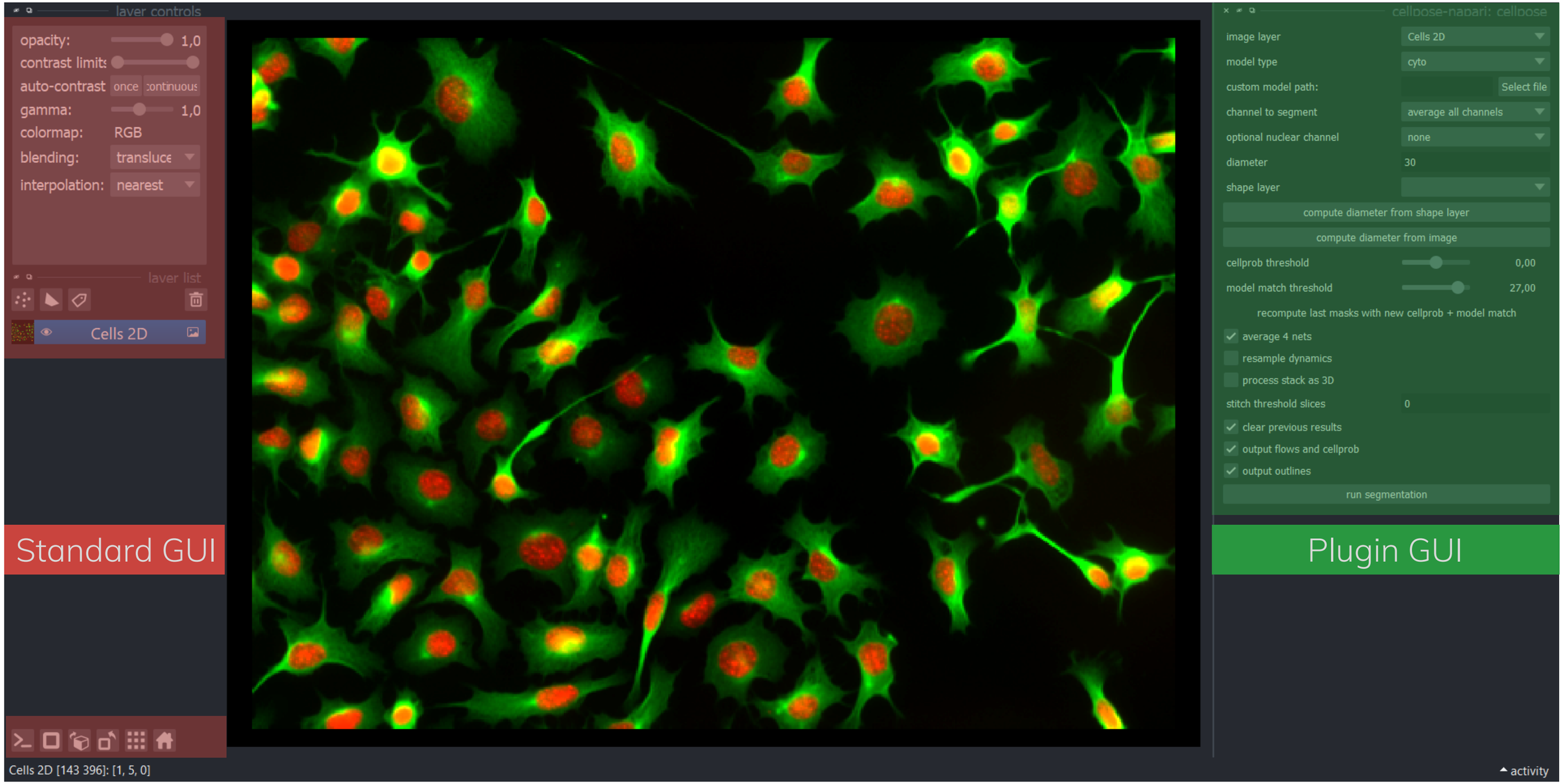
Segmentation



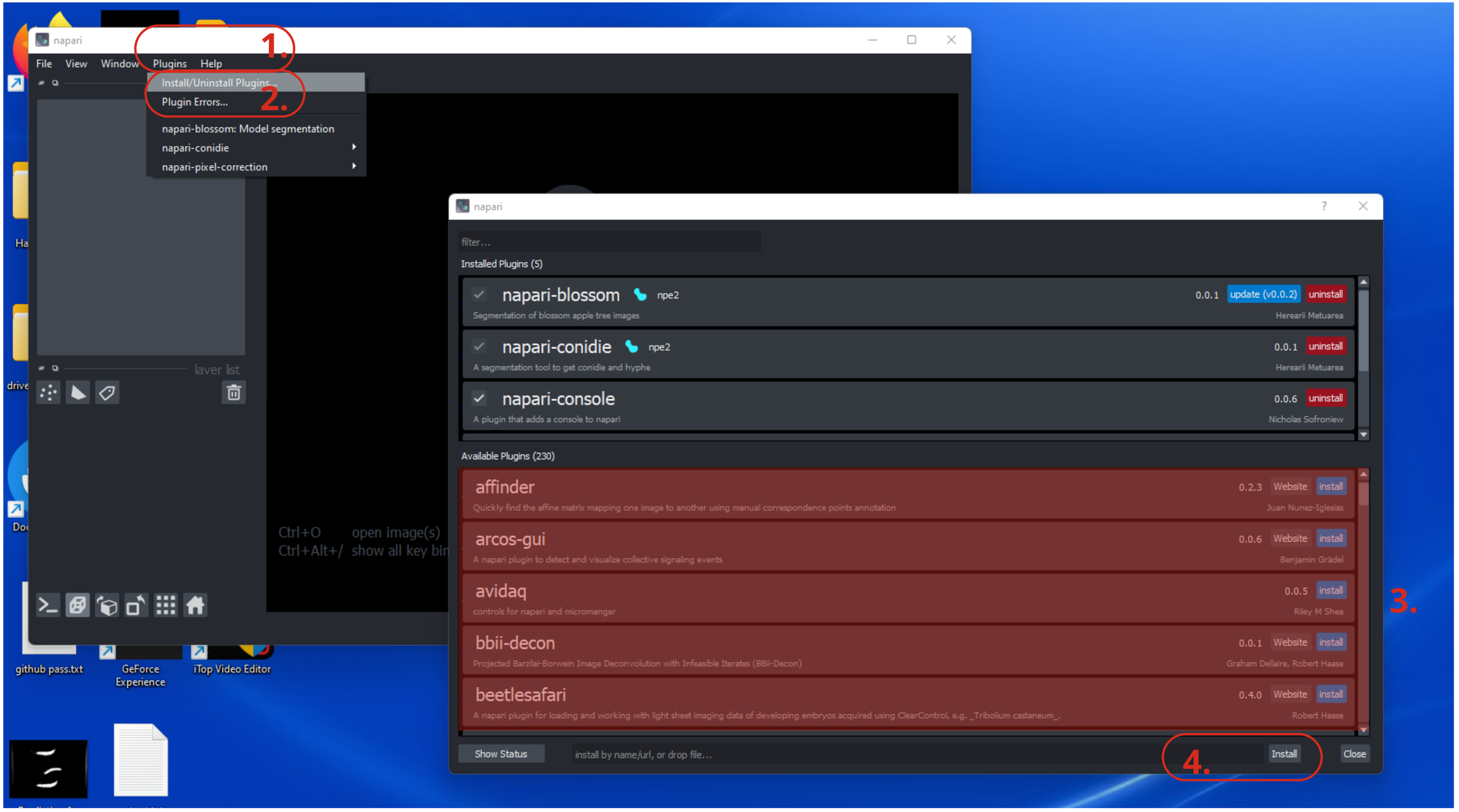
Tracking

Interface utilisateur de napari

plugin : cellpose-napari



Installation d'un plugin sur Napari



DEEPaaS

Napari

Fabrication d'un plugin

Code Inférence sur Google Colab

Input

```
sample_path = '/content/gdrive/My Drive/Input/IMG_3330.JPG'
```

Découpage

```
img1 = imread(sample_path)[:,:,:3]
img1_list = get_mosaic(img1)
```

Importation du modèle

```
model_New = tf.keras.models.load_model('/content/gdrive/My Drive/data/best_model_W_BCE_
    chpping.h5',custom_objects={'dice_coefficient': dice_coefficient})
```

```
taille_p = 256
X_ensemble = np.zeros((len(img1_list), taille_p, taille_p, 3), dtype=np.uint8)

for n in range(len(img1_list)):
  sz1_x,sz2_x,sz3_x = img1_list[n].shape
  if (sz1_x,sz2_x)==(256,256):
    X_ensemble[n]=img1_list[n]
```

Segmentation et Seuil optimal

```
preds_test = model_New.predict(X_ensemble, verbose=1)
preds_test_opt = (preds_test > 0.2).astype(np.uint8)
```

Output

```
output_image = reconstruire(img1,preds_test_opt)
```

Image Input

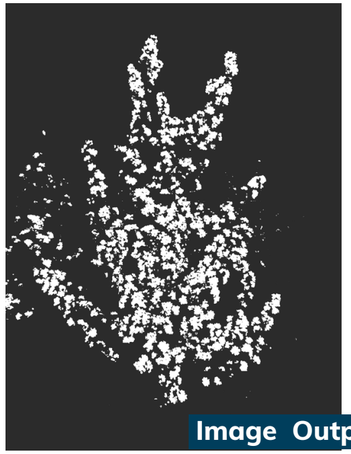


Image Output

Code Inférence sur Napari

```
def image_segmentation(
    layer: ImageData
    ) -> ImageData:

    img1_list = get_mosaic(layer)

    model_New = tf.keras.models.load_model(os.path.join(paths.get_models_dir(),'best_model_W_BCE
                _chpping.h5'),custom_objects={'dice_coefficient': dice_coefficient})
    taille_p = 256
    X_ensemble = np.zeros((len(img1_list), taille_p, taille_p, 3), dtype=np.uint8)
    for n in range(len(img1_list)):
      sz1_x,sz2_x,sz3_x = img1_list[n].shape
      if (sz1_x,sz2_x)==(256,256):
        X_ensemble[n]=img1_list[n]

    preds_test = model_New.predict(X_ensemble, verbose=1)
    preds_test_opt = (preds_test > 0.2).astype(np.uint8)
    output_image = reconstruire(img1,preds_test_opt)
    return np.squeeze(output_image[:,:,0])
@magicgui(call_button="Load",filename={"label": "Pick a file:"})
def get_data(filename=pathlib.Path.cwd()) -> ImageData:
    return imread(filename)[:,:,:3]

@magicgui(call_button="Run")
def do_model_segmentation(
    layer: ImageData) -> ImageData:
    show_info('Succes !')
    return do_image_segmentation(layer)
```

Interface utilisateur

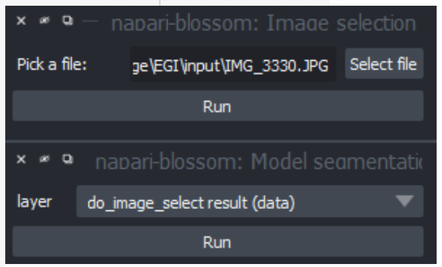


Image Input

Image Output

Conclusion

napari

Interaction avec les données et la chaîne de traitement

Possibilité de visualiser des données 2D/3D sans exploiter un processeur graphique performant

Accès à tous du module et du code source

deep Hybrid DataCloud

Amélioration du modèle par une collaboration des utilisateurs

napari + deep Hybrid DataCloud

Ouverture à la conception de module à tous

Merci pour votre attention