

Webinaire IA & Geodata #3

Se lancer dans la GeoIA open-source : les possibilités, les outils, les limites

Ludovic Pokker, Président Fondateur Nextelia
2 décembre 2025



Présentation

Nextelia en quelques mots :

- Entreprise fondée en mi-2024
- Implantée dans les Hauts-de-France
- Spécialiste des **applications géospatiales sur mesure & GeoAI**
- Philosophie open source, centrée sur l'**utilité métier concrète**
- Accompagne une clientèle gravitant autour de la sphère publique :
Conseil, Formation, Développement, Production de données

Parcours personnel :

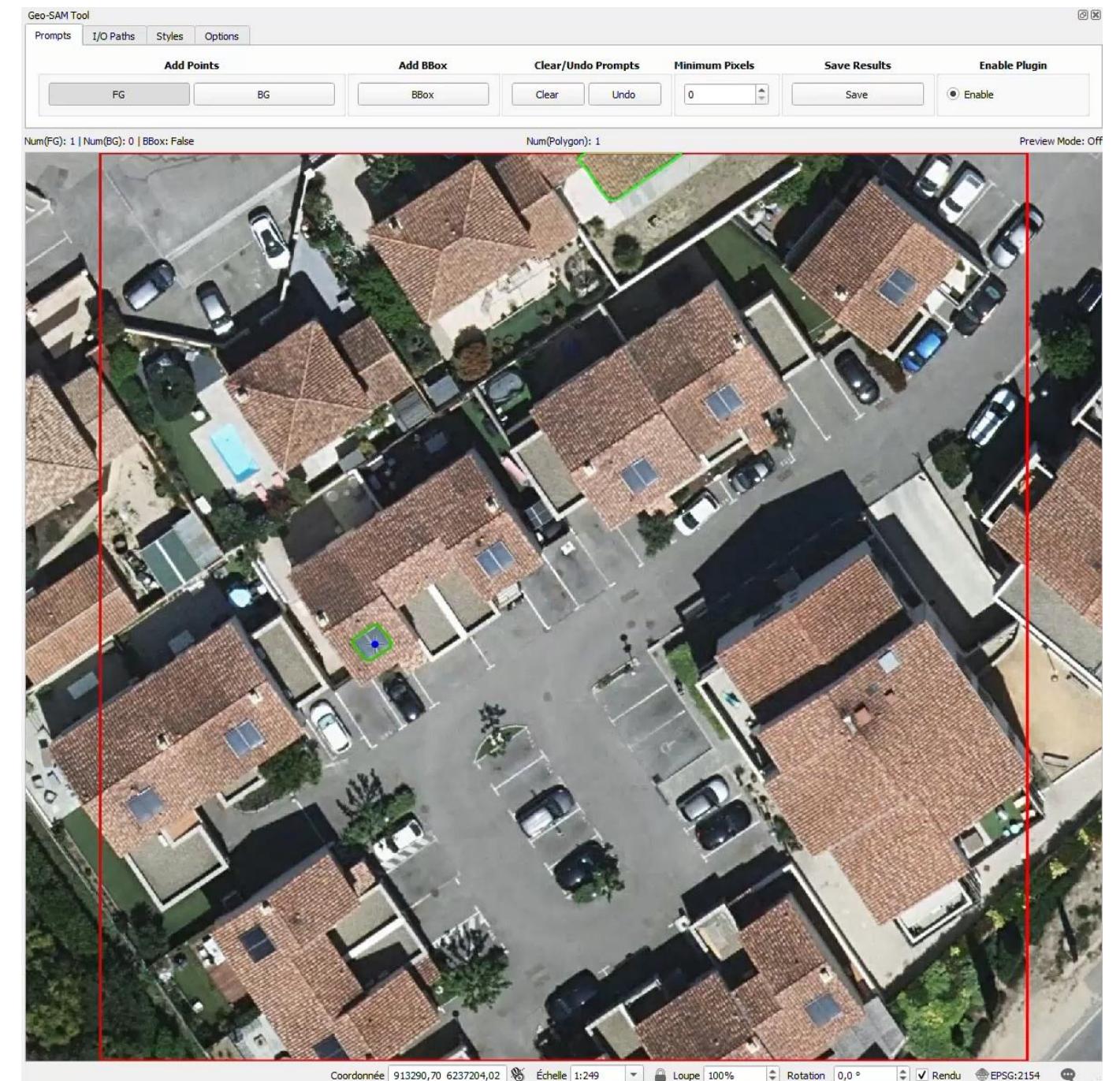
- Expérience de près de 15 ans en collectivités et agence d'urbanisme



Programme

Démarrer techniquement un projet GeoIA :

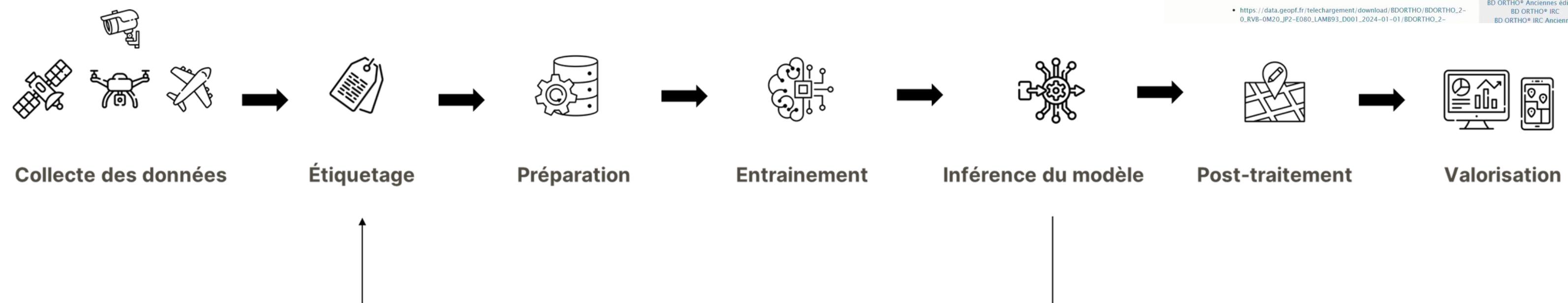
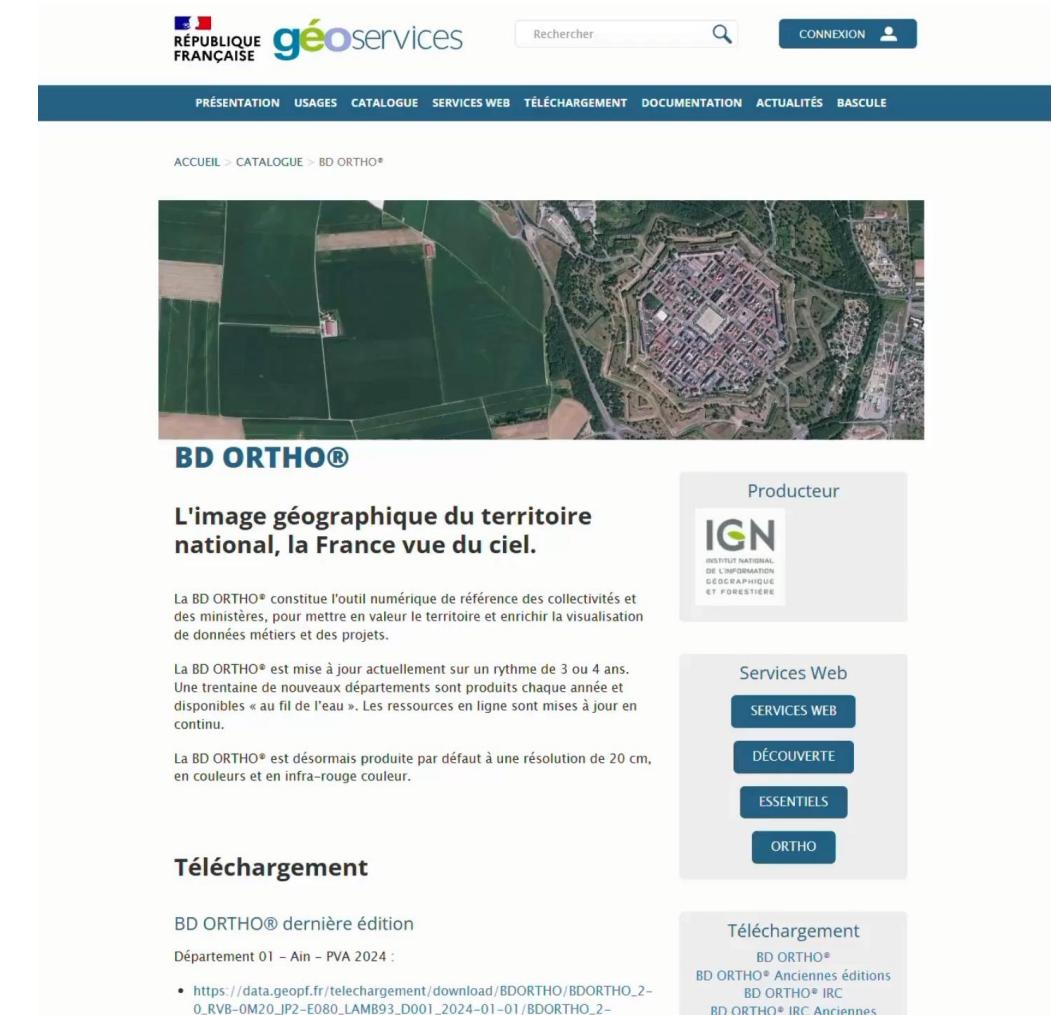
- les 3 étapes incontournables + le défi à relever pour conserver l'intégrité de la donnée
- Présentation de solutions actuelles reconnues par la communauté, leurs avantages et inconvénients + démonstrations live
- Conclusion et ouverture sur les plugins en développement
- Questions et échanges



Rappel de la vision globale d'un projet

Ce schéma représente l'enchaînement des étapes essentielles depuis la collecte de données géospatiales brutes jusqu'à l'exploitation des résultats.

A noter, la plupart des applications territoriales, reposent sur le **Deep Learning** avec des **approches supervisées/par transfert**.

The screenshot shows the official page for BD ORTHO® on the République Française geoportail.fr. The main content area displays a high-resolution orthophoto of a rural landscape. Below the image, the text "BD ORTHO®" is displayed, followed by a description: "L'image géographique du territoire national, la France vue du ciel." It also mentions the producer (IGN) and provides links for web services.

Quand la GeoAI nécessite des outils dédiés



Etiqueter

Associer des exemples à des classes pour construire la base d'apprentissage.



Entraîner

Apprendre à partir des données pour créer un modèle performant.



Prédire

Utiliser le modèle pour analyser de nouvelles données géospatiales.

Class catalog

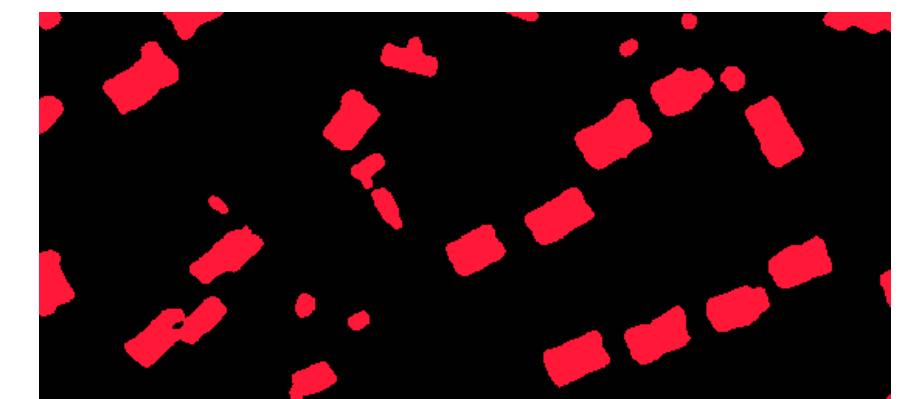
ID	Label	Symbol
1 1	Grass	
2 2	Tree	
3 3	Road	
4 4	Building	

Training Log: Clear

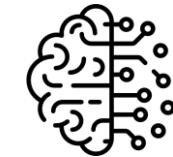
```

    Val Precision: 0.7943
    Val Recall: 0.8169
    Learning Rate: 0.001000

    Epoch 33/50
    Time: 0.7s
    Train Loss: 0.1354
    Val Loss: 0.3548
  
```



Un enjeu : conserver la dimension spatiale



Etiqueter

- Les étiquettes sont créées dans un **espace géographique**
- La **dimension spatiale se perd** lors de la conversion dans les formats d'entraînement standard
- Sont perdu(e)s :
 - Coordonnées géographiques réelles
 - Système de référence spatiale
 - La résolution spatiale
 - Le lien explicite avec la tuile géoréférencée...

Entraîner

PYTORCH

- Ne gère pas nativement la dimension géographique
- Une image vue par pytorch est un tableau de nombres (appelé **tenseur**)
- Exemple image RGB de 256x256 :
Tenseur de forme = [3, 256, 256]
- Chaque combinaison ligne/colonne correspond à un pixel qui est représenté et les valeurs représentent l'intensité de chaque canal pour ce pixel,
- Aucun contexte spatial n'est associé (latitude, longitude, projection, résolution, extent...)

Prédire

- Pytorch renvoi **une prédiction dans l'espace de tenseurs**, pas dans l'espace géographique
- Pour obtenir un résultat géographiquement cohérent, indispensable d'utiliser des **métadonnées géospatiales**
- Et de s'assurer que l'inférence respecte les mêmes conditions que l'entraînement (même résolution en particulier)

Portrait robot de l'extension idéale



Un outil idéal devrait :

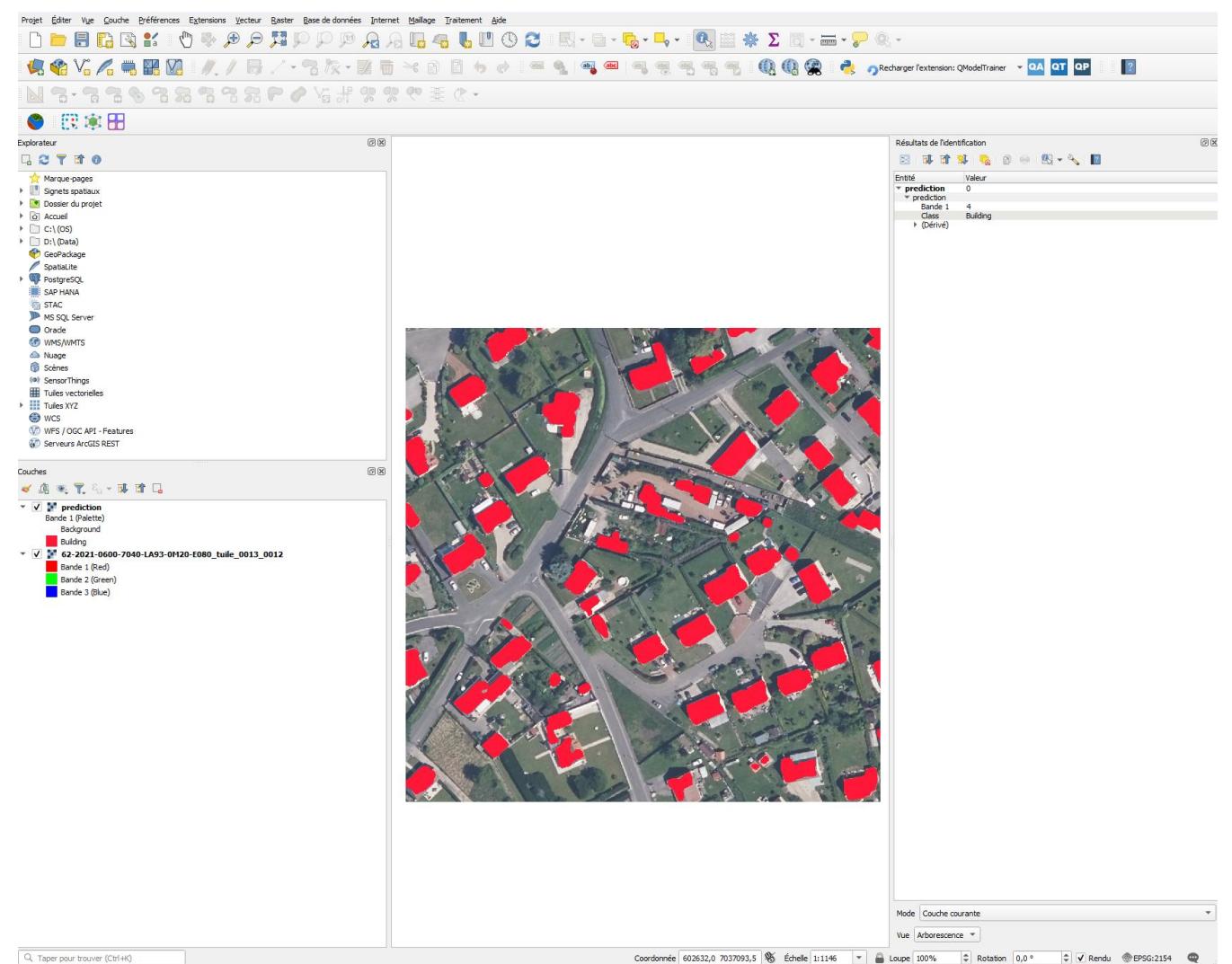
- **Couvrir les 3 étapes GeoAI : étiqueter → entraîner → prédire**
- **Préserver et exploiter les métadonnées géospatiales (CRS, résolution, emprise) tout au long de la chaîne**
- **S'intégrer à QGIS ou produire directement une couche SIG prête à l'emploi**



Avertissement

Quel que soit l'outil, **une base solide en deep learning est indispensable** :

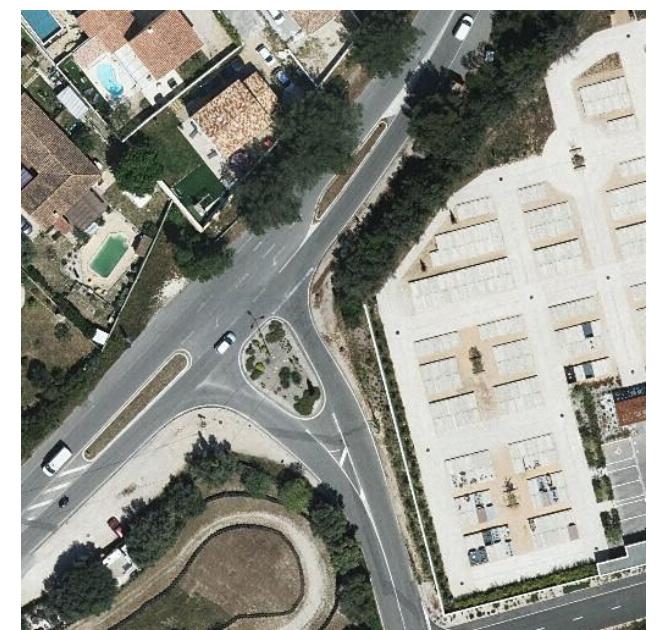
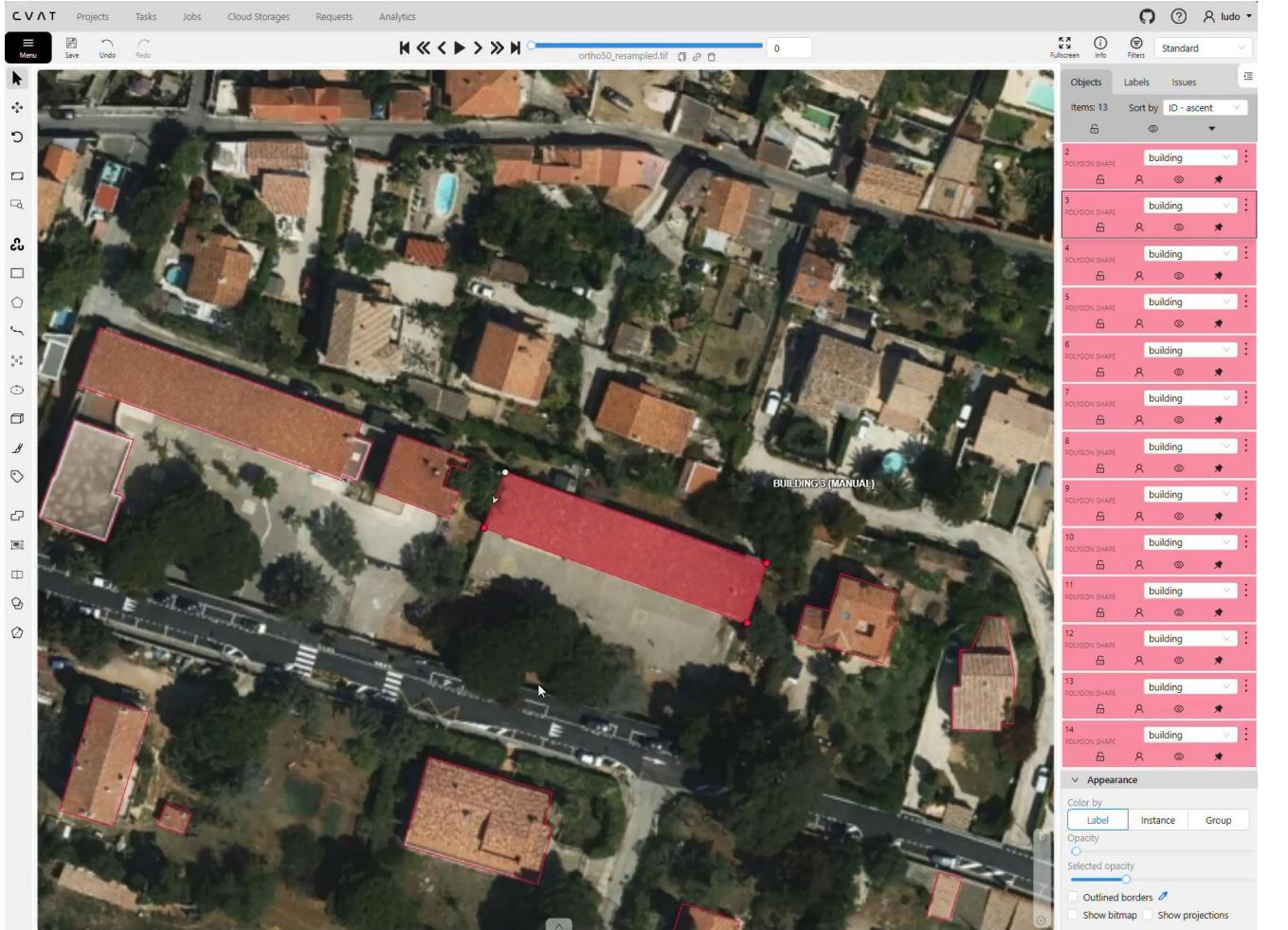
- fonctionnement d'un réseau neuronal
- paramètres / hyperparamètres
- notions clés



Panorama des outils GeoAI dans l'écosystème open source

Etiqueter :

- **Phase la moins dotée en outils adaptés** pour les images géospatiales alors qu'elle est la **phase la plus critique** : définit ce que le modèle doit apprendre.
- Solutions classiques :
 - Utiliser des outils de computer vision « classiques » (CVAT, LabelMe...)
 - + : Annoter rapidement des rasters
 - + : Export vers des formats standard
 - - : Ne gèrent pas les spécificités géospatiales : pas de découpages en tuiles, chevauchement, géoréférencement...
 - QGIS:
 - + : Possible en créant des masques ou des geojson
 - + : Des extensions pour tenter de faciliter l'annotation (GeoSam)
 - - : Processus laborieux



Panorama des outils GeoAI dans l'écosystème open source

Entrainement :

- Qgis ne propose **pas d'outils natifs pour l'entraînement**.
 - Une extension recensée dans le catalogue (Qlearn) mais uniquement basée sur u-net
 - Workflow **recommandé : Python** externe

Inférence :

- Export dans un format SIG depuis Python
 - Une extension dédiée à l'inférence : DEEPNESS

Jupyter Instance_Seg_Pytorch Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Markdown

JupyterLab Python 3 (ipykernel)

```
legend_patches = [mpatches.Patch(color=cat_id_to_color[cat['id']],  
                                 label=cat['name'])  
                  for cat in categories]  
fig.legend(handles=legend_patches, loc='upper center',  
           bbox_to_anchor=(0.5, 0.98), ncol=len(categories),  
           frameon=True, fontsize=10)  
  
plt.tight_layout()  
plt.subplots_adjust(top=0.92)  
plt.savefig('dataset_samples.png', dpi=150, bbox_inches='tight')  
plt.close()  
  
print(f"\n/\n Visualisation sauvegardée : dataset_samples.png")  
print(f"\n/\n Statistiques des exemples affichés :")  
for img_idx in sample_indices:  
    img_id = image_ids[img_idx]  
    img_info = coco.loadImgs(img_id)[0]  
    ann_ids = coco.getAnnIds(imgIds=img_id)  
    anns = coco.loadAnns(ann_ids)  
  
    # Compter les annotations par catégorie  
    cat_counts = defaultdict(int)  
    for ann in anns:  
        cat_name = [cat['name'] for cat in categories if cat['id'] == ann['category_id']][0]  
        cat_counts[cat_name] += 1  
  
    print(f" {img_info['file_name']}: ", end="")  
    print(", ".join([f"{count} {name}" for name, count in cat_counts.items()]))  
  
# Afficher dans le notebook  
display(IPImage('dataset_samples.png'))
```



Transformations et augmentation de données

Demo #1 : GeoSAM / GeoOSAM

SAM – Segment Anything Model (Meta)

- Modèle de segmentation universelle
- Fonctionne à partir de prompts (points, boîtes...)
- Très performant pour générer des masques d'objets **sans entraînement spécifique**
- Pas entrainé sur des données géospatiales mais bonne généralisation

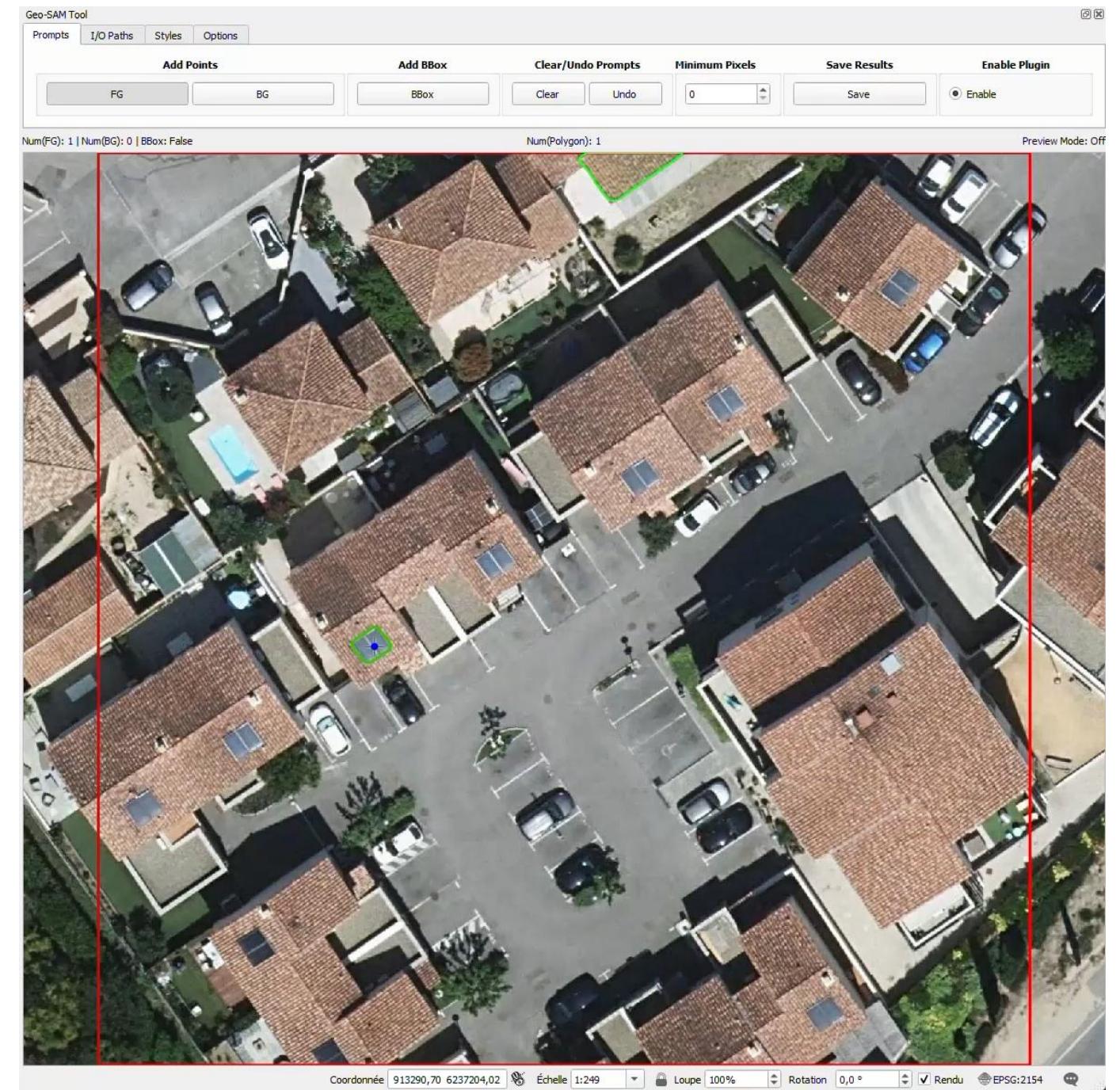
GeoSAM

- Extension destinée à assister l'**étiquetage** ou accélérer la **numérisation**
- Utilise SAM 1 en arrière plan
- Facile à utiliser, pas de compétences en DL pré-requises
- Cas d'usage limité, pas conçu pour les grands rasters



GeoOSAM

- Extension similaire plus récente (en développement ?)
- Utilise SAM 2 (différences non notables mais checkpoints plus légers)
- Des pistes d'amélioration intéressantes (ui, système de classes, export, moins de configuration...)
- Mais des lacunes importantes (pas de points background ! Pas de fusion, export en... shapefile et un fichier par classe)



Demo #2 : Deepness

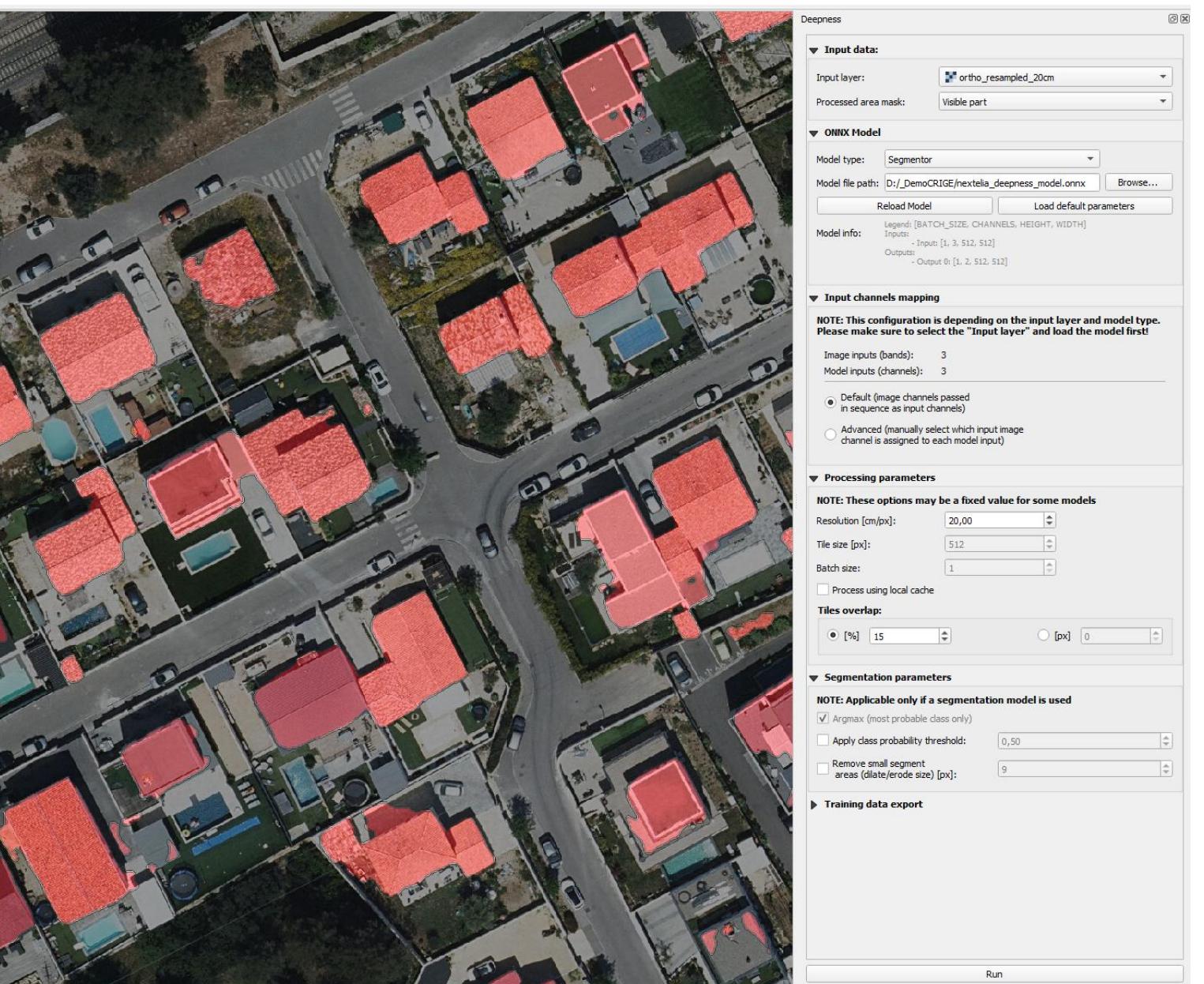
Extension facilitant l'inférence et la préparation de données directement dans QGIS

Avantages

- Directement intégré à QGIS
- Idéal pour l'inférence et la préparation sans coder
- Simplifie le passage du modèle à la couche SIG finale
- Accessible pour les équipes non spécialistes du deep learning

Limites

- Ne gère pas l'entraînement
- Moins flexible qu'un pipeline Python complet
- Les modèles doivent respecter les formats attendus (PyTorch, YOLO, ONNX...)
- Interface complexe
- Certains modèles usuels non supportés (mask r-cnn par exemple / lié à la volonté de privilégier les formats facilement exportables en ONNX)



Demo #3 – Python / Pytorch

Python

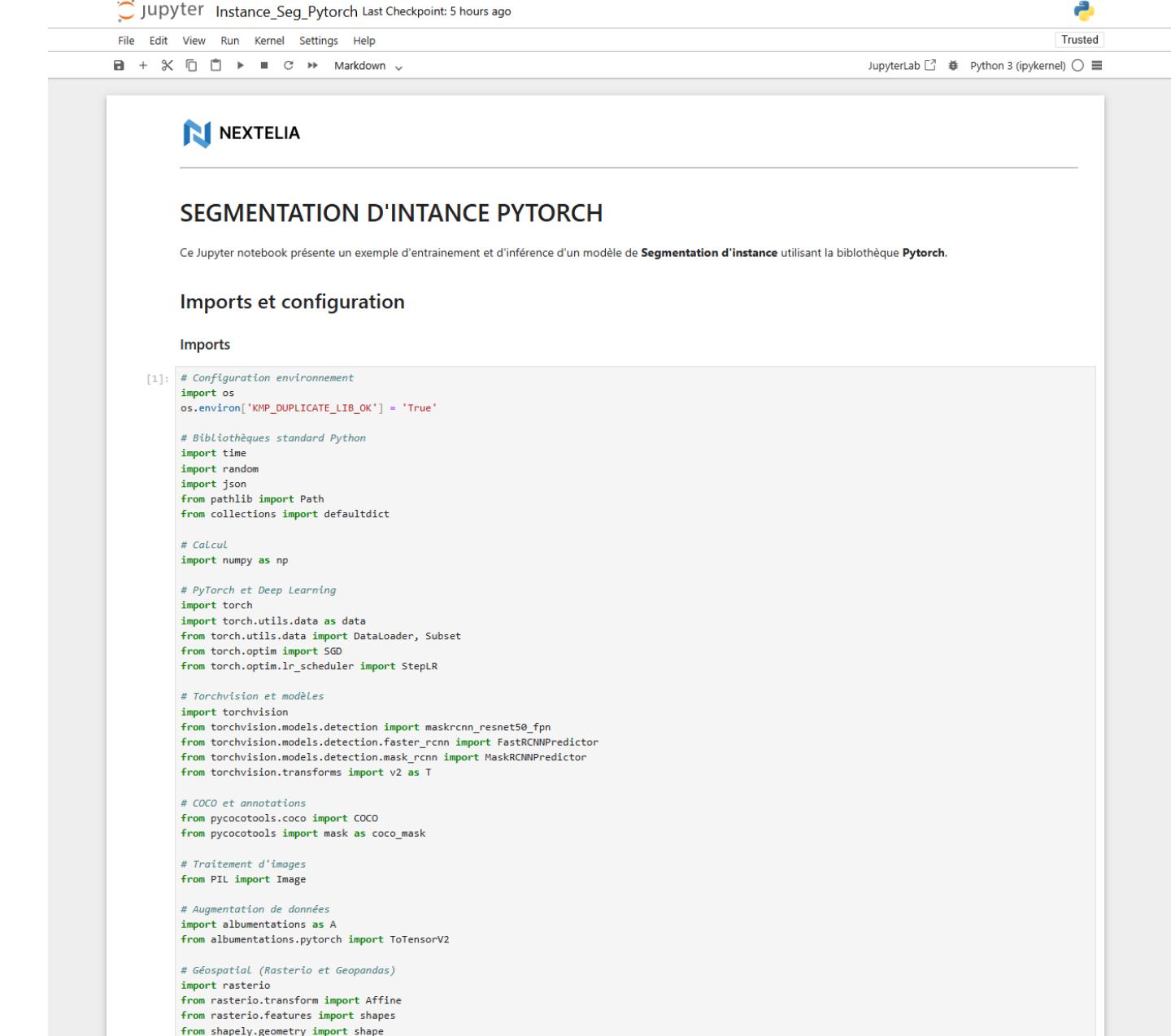
- Langage open source le plus utilisé, réputé pour sa lisibilité, sa courbe d'apprentissage progressive et sa flexibilité

Pytorch

- Framework open-source (Meta/Facebook, 2016)
- Permet l'entraînement, la validation et l'inférence de modèles
- Massivement adopté par la communauté. Référence en GeoAI.

Point clés

- Utilise des tenseurs pour représenter les données
- Nécessite une solution d'étiquetage externe (labelme, cvat...)
- Offre la plus grande souplesse mais demande :
 - Des compétences en développement python
 - Des solides bases en deep learning

```
[1]: # Configuration environnement
import os
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'

# Bibliothèques standard Python
import time
import random
import json
from pathlib import Path
from collections import defaultdict

# Calcul
import numpy as np

# PyTorch et Deep Learning
import torch
import torch.utils.data as data
from torch.utils.data import DataLoader, Subset
from torch.optim import SGD
from torch.optim.lr_scheduler import StepLR

# Torchvision et modèles
import torchvision
from torchvision.models.detection import maskrcnn_resnet50_fpn
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
from torchvision.transforms import v2 as T

# COCO et annotations
from pycocotools.coco import COCO
from pycocotools import mask as coco_mask

# Traitement d'images
from PIL import Image

# Augmentation de données
import albumentations as A
from albumentations.pytorch import ToTensorV2

# Géospatial (Rasterio et Geopandas)
import rasterio
from rasterio.transform import Affine
from rasterio.features import shapes
from shapely.geometry import shape
import geopandas as gpd
```

Intégration SIG

- Les résultats peuvent être exportés en rasters ou couches vectorielles, directement utilisables dans QGIS



Demo #4 - Python - Opengeos / GeoAI

Opengeos/Geoai

- Bibliothèque python open source visant à simplifier le workflow GeoAI
- Ajoute des fonctions utiles, une couche d'abstraction au dessus de Pytorch
- Modèles prêts à l'emploi (classification, détection, segmentation, basé sur Pytorch et Transformers)
- Support SIG large : Geotiff, GeoJSON, Shapefile...

Avantages :

- Réduit la quantité de code Pytorch pur
- Idéal pour s'initier, prototyper
- Intègre des modules utiles pour les données géospatiales

Limites :

- Reste dépendant de Pytorch : abstrait mais ne remplace pas la nécessaire compréhension des modèles
- Flexibilité plus faible qu'un workflow Pytorch « from scratch »
- Moins adapté pour des architectures customs ou des besoins spécifiques

▼ Train a Semantic Segmentation Model using Segmentation-Models-PyTorch

 Open in Colab

This notebook demonstrates how to train semantic segmentation models for object detection (e.g., building detection) using the [segmentation-models-pytorch](#) library. Unlike instance segmentation with Mask R-CNN, this approach treats the task as pixel-level binary classification.

Install packages

To use the new functionality, ensure the required packages are installed.

```
[4] 30 s %pip install geoai
Afficher la sortie masquée
```

Import libraries

```
[5] 45 s import geoai
```

Download sample data

We'll use the same dataset as the Mask R-CNN example for consistency.

```
[ ] train_raster_url = (
    "https://huggingface.co/datasets/giswqs/geospatial/resolve/main/naip_rgb_train.tif"
)
train_vector_url = "https://huggingface.co/datasets/giswqs/geospatial/resolve/main/naip_train_buildings.geojson"
test_raster_url = (
    "https://huggingface.co/datasets/giswqs/geospatial/resolve/main/naip_test.tif"
)

[ ] train_raster_path = geoai.download_file(train_raster_url)
train_vector_path = geoai.download_file(train_vector_url)
test_raster_path = geoai.download_file(test_raster_url)

[ ] Downloading naip_rgb_train.tif: 100%|██████████| 8.88M/8.88M [00:00<00:00, 73.5MB/s]
Downloading naip_train_buildings.geojson: 334KB [00:00, 92.1MB/s]
Downloading naip_test.tif: 100%|██████████| 19.7M/19.7M [00:00<00:00, 48.8MB/s]
```

Visualize sample data

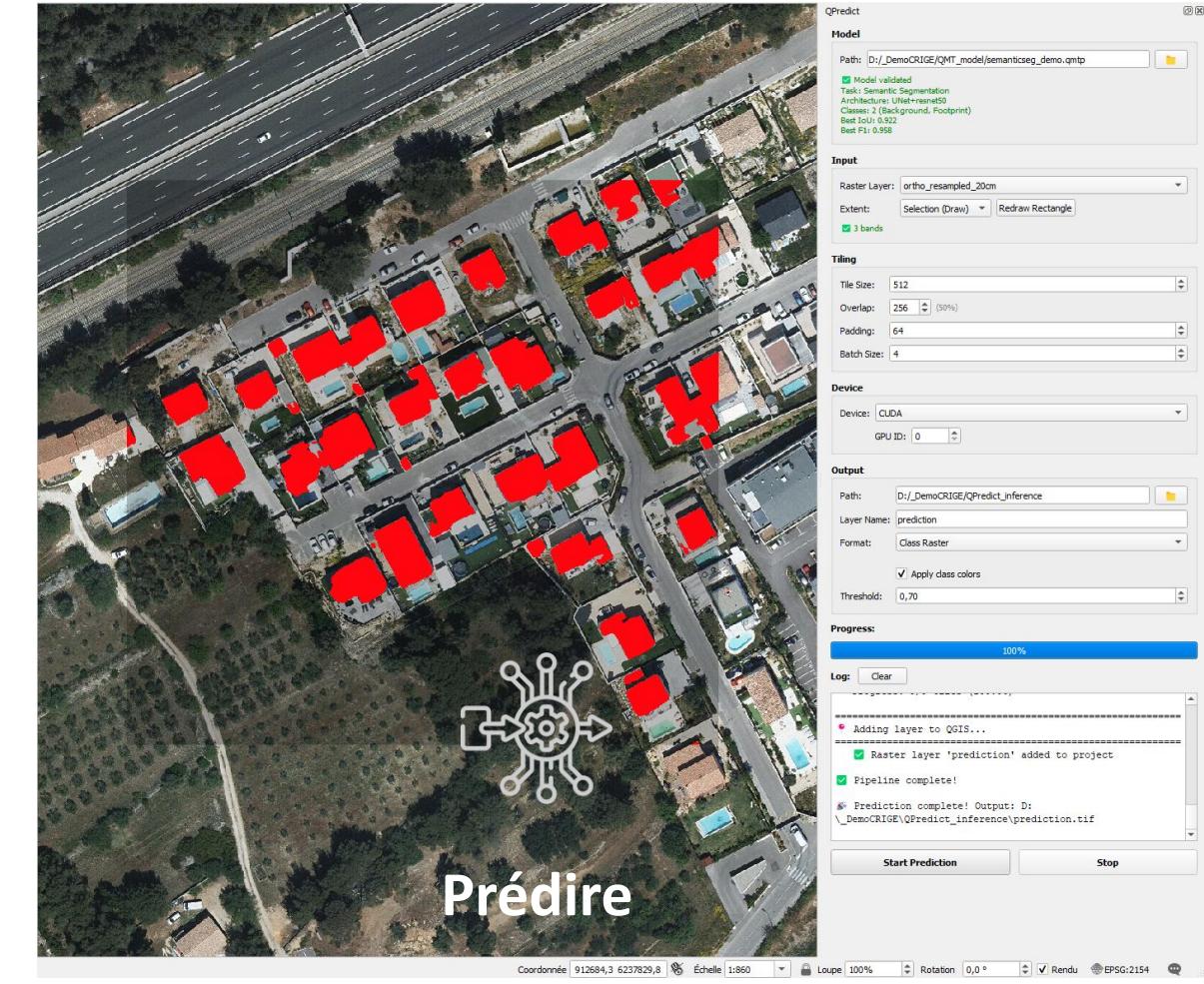
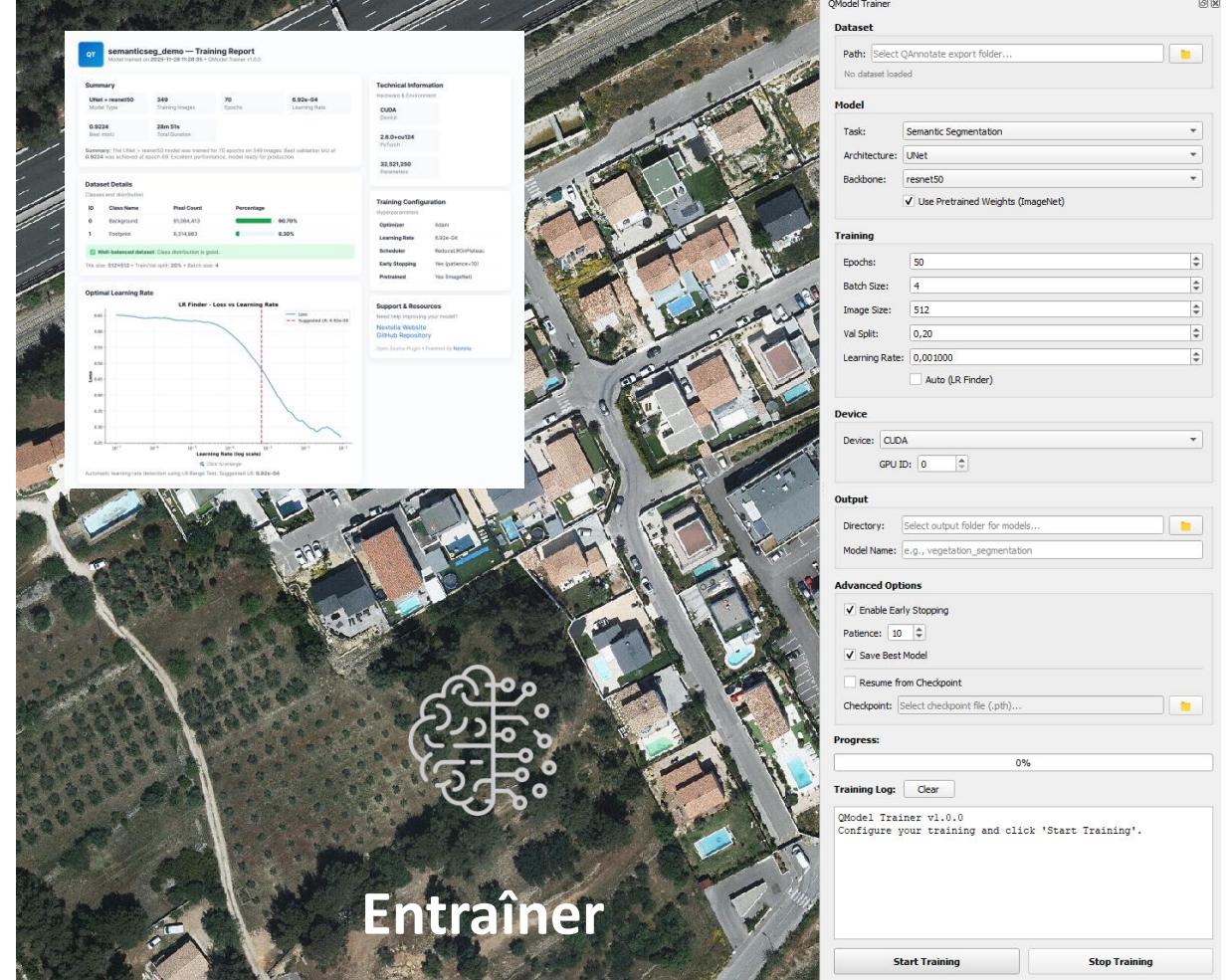
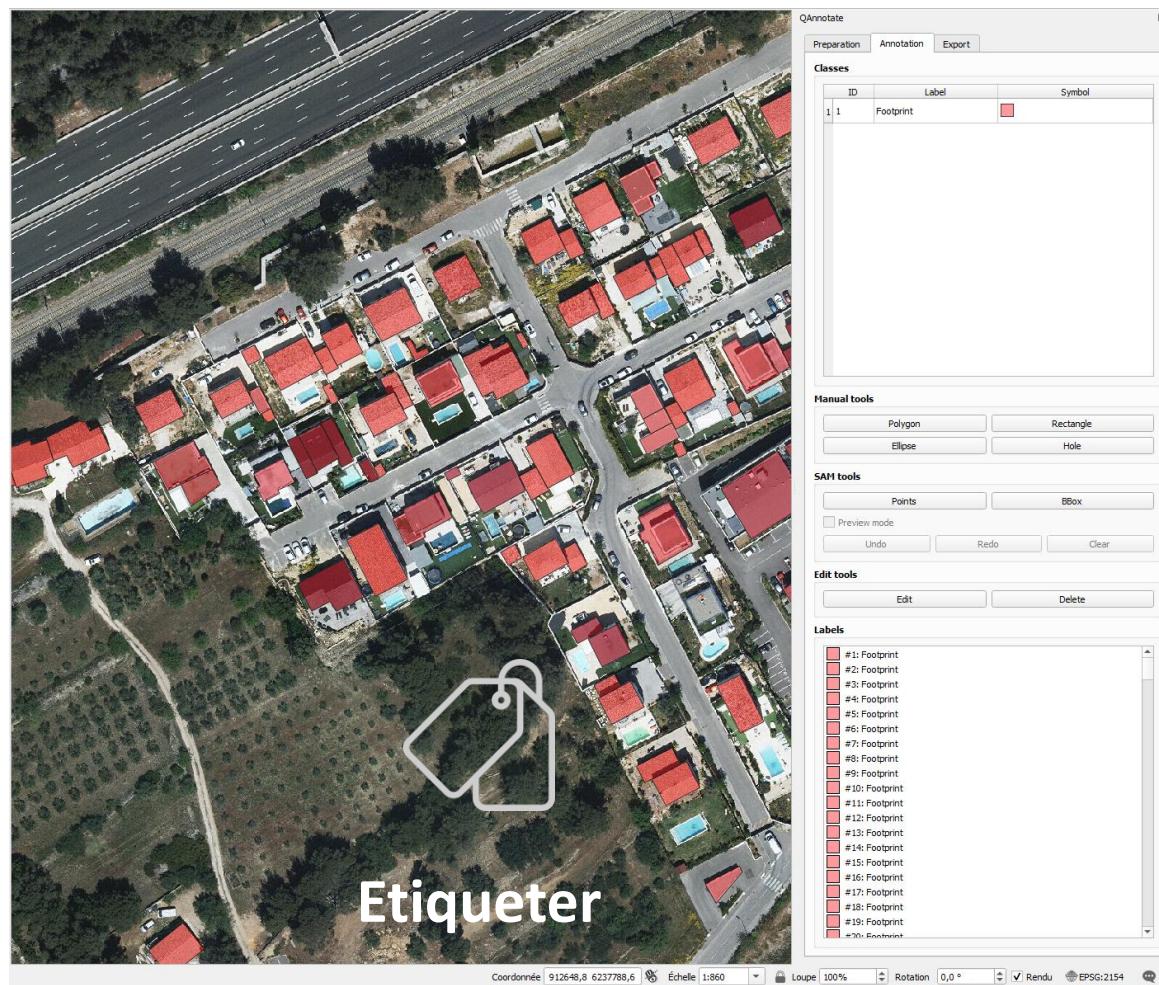
```
[ ] geoai.get_raster_info(train_raster_path)
{'driver': 'GTiff',
 'width': 2503,
```



Vers une approche plus intégrée : extensions en développement

Résumé des attentes pour une approche plus intégrée dans QGIS:

- Couvrir les 3 étapes clés GeoAI : *étiqueter* → *entraîner* → *prédire*
- Préserver et exploiter les métadonnées géospatiales (CRS, résolution, emprise) tout au long de la chaîne
- Produire directement une couche SIG prête à l'emploi



Questions / Réponses



Ludovic POKKER

Président

Web : www.nextelia.fr

Mail : ludovic@nextelia.fr

Tél : 06 41 09 11 75



Valentin Douarre

Animateur Géodatalab – Partenariats & Open Innovation

Web : www.crike-paca.org

Mail : valentin.douarre@crige-paca.org

Tél : 04 42 90 67 86