Tutoriel YOLO Deepness

Segmentation d'arbres

Sommaire

1. Présentation

Le plugin "Deepness" est conçu pour les utilisateurs occasionnels de QGIS, simplifiant l'utilisation des Réseaux Neuronaux Profonds grâce à une interface facile à utiliser. Les utilisateurs peuvent traiter leurs données sans avoir besoin de connaissances approfondies en apprentissage automatique, en tirant parti de la capacité du plugin à gérer des tâches complexes de traitement.

Le plugin facilite l'application de techniques de Machine Learning telles que la segmentation, la détection et la régression sur des orthophotos raster. Il s'intègre parfaitement à QGIS, permettant aux utilisateurs d'importer des données de diverses sources, de revoir les résultats et d'exporter les sorties dans des formats SIG natifs.

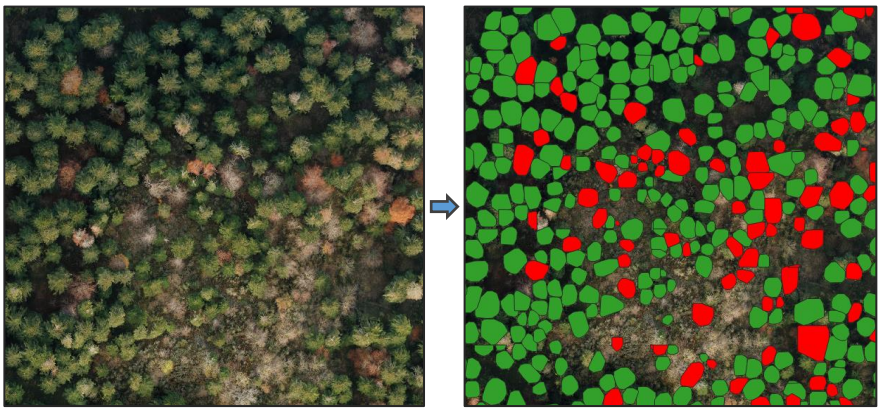


Figure 1 - Exemple simple de segmentation dans un jeune peuplement forestier.

Le plugin prend en charge une gamme de modèles déjà entrainées, ainsi que d'options de traitement. Il offre des fonctionnalités telles que la limitation de la zone de traitement, et un outil d'exportation de données d'entraînement. Compatible avec plusieurs systèmes d'exploitation, il fournit aux utilisateurs avancés des options de paramétrage pour une personnalisation supplémentaire. C'est un outil open source et l'équipe de développement est ouverte aux retours et aux opportunités de collaboration.

Felipe AGUIAR MARTIN, 2024

2. Installation et nécessitées

Tout d'abord, avec le logiciel QGIS déjà installé, l'installation du plug-in **Deepness** se fait via l'onglet "Extensions" (Figure 2).

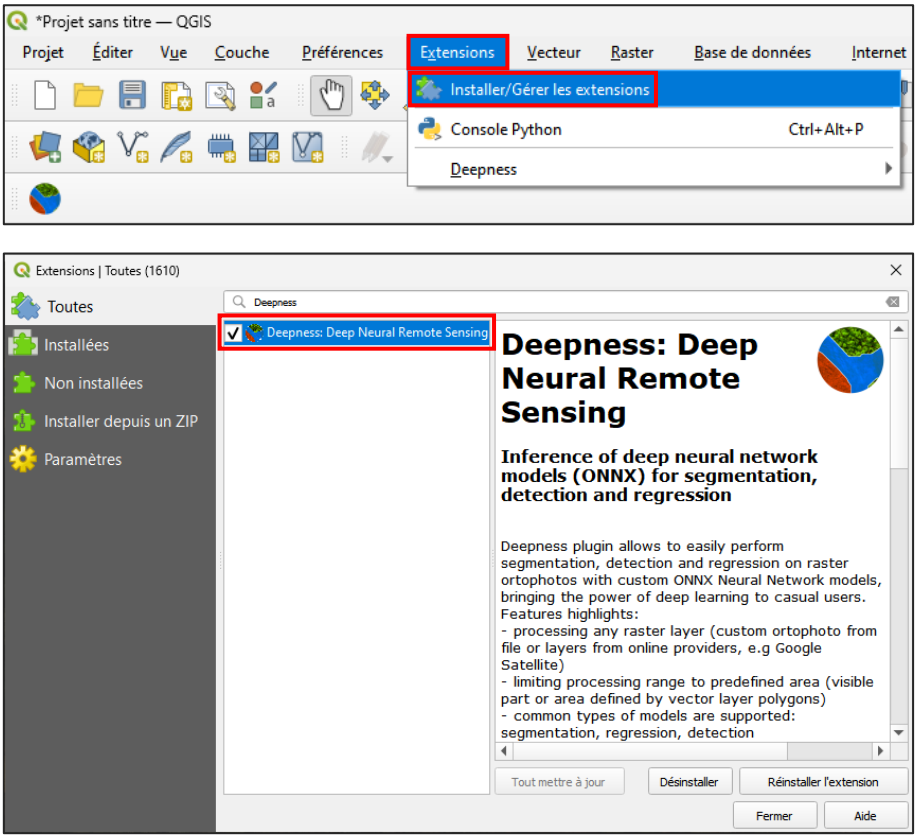


Figure 2 - Instalation plug-in Deepness.

Par ailleurs, il est nécessaire de créer un compte sur les plateformes Roboflow et Ultralytics. Ces outils en ligne proposent des versions gratuites avec des limitations de stockage. Des versions payantes sont disponibles pour des besoins spécifiques ou pour le stockage de gros volumes de données.

Pour les tâches de labélisation et d'entraînement d'un modèle de petite ou moyenne taille, la version gratuite devrait être suffisante.

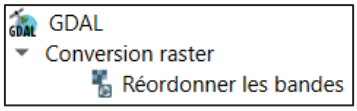
Roboflow : https://app.roboflow.com/

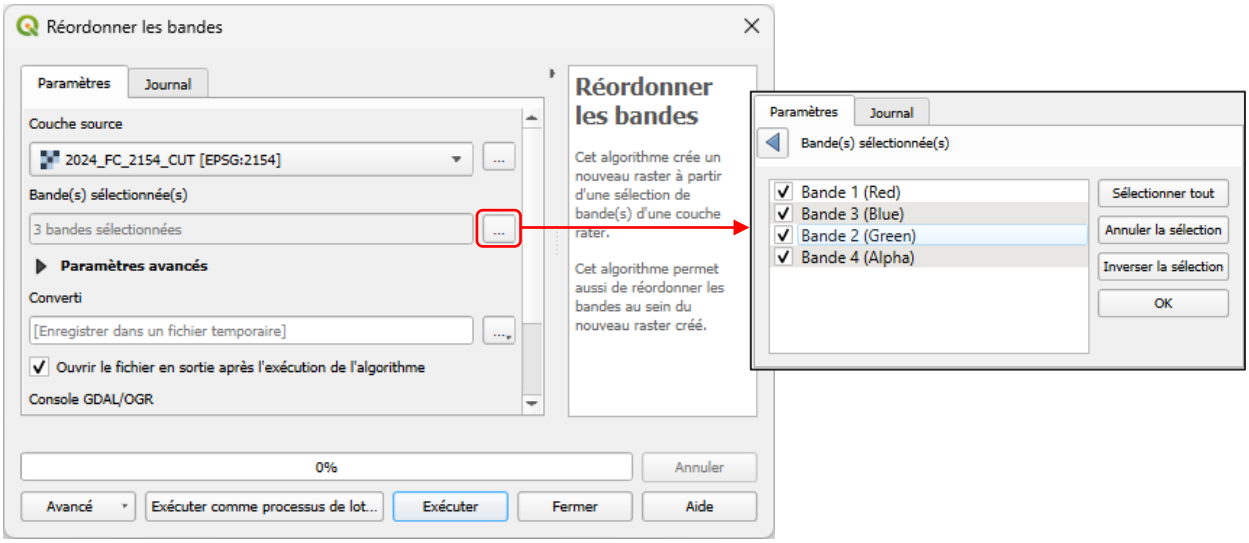
Ultralytics : https://hub.ultralytics.com/

3. Ajustement des données

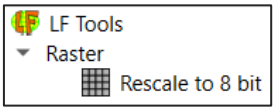
Si les données sont différentes de 8 bits, 3 bandes (ou 3 bandes + bande de transparence) et ne sont pas dans un bon système de coordonnées de référence (SCR), des ajustements sur l'image seront nécessaires. Par exemple, pour les données multispéctrales, il est souvent nécessaire, et même conseillé, de réordonner les bandes pour créer une image en fausses couleurs.

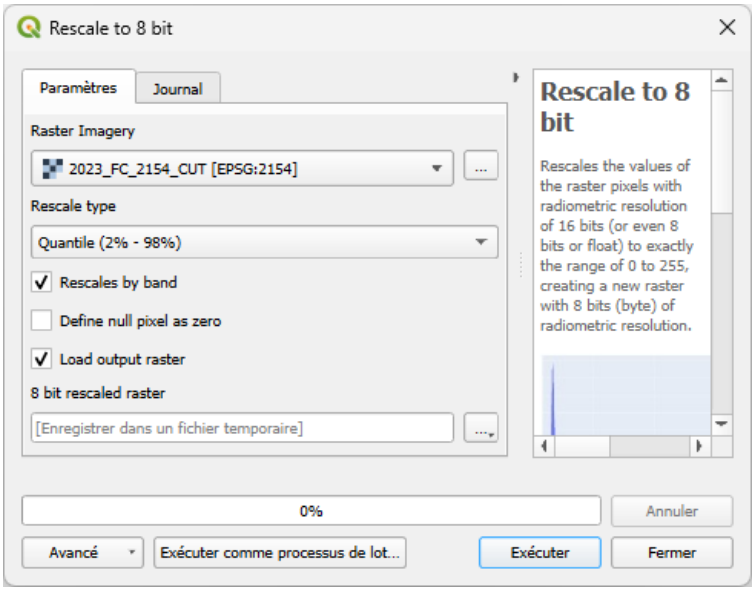
- **Réordoner les bandes** (formation fausse couleur, par exemple)



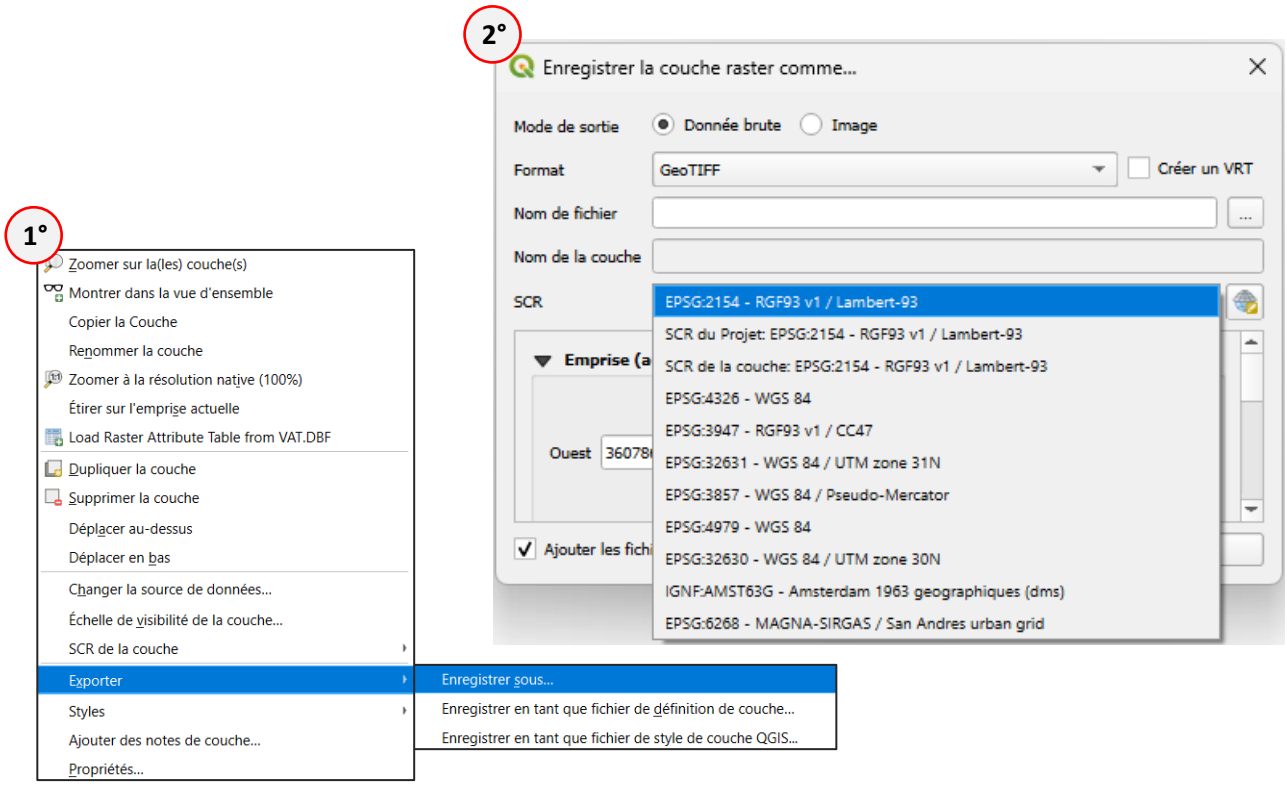


- **Redimensionnement 8bit**

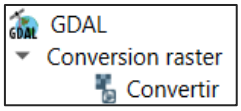


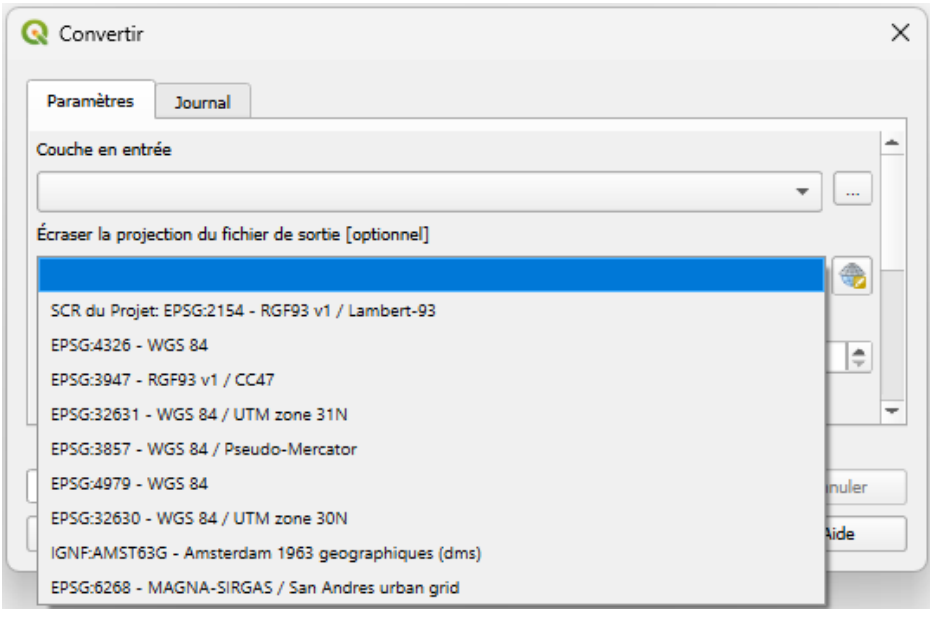


- **Conversion SCR**



OU ENCORE...





Informations nécessaires :

*Dans le cas où l'entraînement du modèle est effectué sur votre ordinateur (sans recours à une **machine virtuelle**, comme Google Colab), il convient de noter que **Python** et la bibliothèque "**pip**" doivent être installés. De plus, une **carte graphique compatible avec CUDA** est indispensable.

* De plus, les données raster doivent être en **3 bandes** visibles (ou 4 avec une bande alpha invisible) et en **8 bits**, pour permettre leur traitement, du tuilage à la segmentation.

4. Utilisation

Il s'agit d'une chaîne de traitement visant entrainer un modèle, sans avoir besoin de connaissances avancées en programmation et en apprentissage automatique, pour obtenir un résultat de segmentation à partir d'un entraînement avec des données propres. Pour cela, quatre outils principaux sont nécessaires : le logiciel QGIS, le site internet Roboflow, le site internet Ultralytics et Python.

1. **QGIS** : Utilisé pour préparer et manipuler des données géospatiales. L'utilisateur peut importer, visualiser et exporter des données raster et vectorielles, facilitant la création de jeux de données pour l'entraînement des modèles de segmentation à partir du plug-in Deepness.
2. **Roboflow** : Plateforme en ligne pour la labélisation, l'augmentation et l'organisation des données d'images. Les données préparées dans QGIS sont chargées dans Roboflow, où elles sont labélisées et améliorées pour accroître la qualité et la diversité du jeu de données d'entraînement.
3. **Ultralytics** : Site utilisé pour entraîner des modèles d'apprentissage profond, en particulier la bibliothèque YOLO (You Only Look Once) pour la détection et la segmentation d'objets. L'utilisateur charge les données labélisées dans Roboflow vers Ultralytics, où les modèles sont entraînés.
4. **Python** : Langage de programmation utilisé pour intégrer et automatiser le flux de travail. Dans le scénario de ce tutoriel, Python servira principalement pour tourner les codes fournis par Ultralytics.

Liens utiles :

Plug-in QGIS : https://plugins.qgis.org/plugins/deepness/

Documentation : https://qgis-plugin-deepness.readthedocs.io/en/latest/index.html

GitHub : https://github.com/PUTvision/qgis-plugin-deepness

4.1. Tuilage

Dans cette première étape, le processus de découpe de l'image .tif en petites tuiles utilisables dans le processus de labellisation et entrainement sera réalisé. Il est important de définir principalement le chevauchement et la résolution souhaitée (Figure 3), car cela aura une incidence directe sur la quantité de données à prétraiter, ainsi que sur la qualité des résultats du modèle généré.

Pour procéder au traitement de tuilage, il est strictement nécessaire d'avoir un fichier .tif avec 3 bandes en 8 bits.

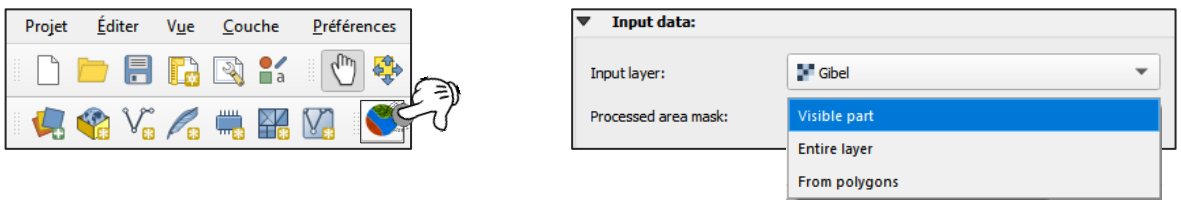


Figure 3 - Paramétrage de traitement de Tuilage.

- Instructions :

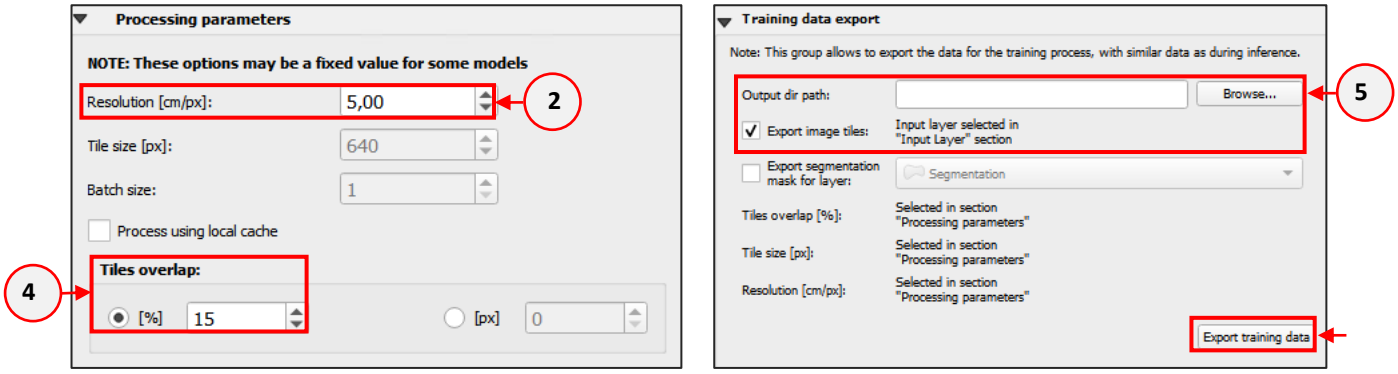
→ **1.** Commencez par ouvrir le logiciel et ajouter l'orthophotographie contenant l'image. Sélectionnez votre zone.

→ **2.** Faites défiler jusqu'à la résolution (cm/px). Elle devrait être proche de la Ground Sample Distance (GSD) de votre orthophoto.

→ **3.** Choisissez soit une taille de tuile de 640 ou de 1280 selon votre orthophoto. La taille 640 est recommandée, mais la taille 1280 peut être nécessaire pour détecter les objets très petits.

→ **4.** Maintenez le chevauchement des tuiles à 15%.

→ **5.** Faites défiler jusqu'au chemin du répertoire de sortie et choisissez le chemin. Cochez "Export image tiles" et appuyez sur "Export training data".



Paramétrages de traitements plug-in Deepness QGIS :

https://qgis-plugin-deepness.readthedocs.io/en/latest/main/main_ui_explanation.html#processing-parameters

À la fin, un dossier contenant les tuiles découpées sera exporté, comme illustré dans l'exemple (Figure 4).

Figure 4 - Exemple de dossier de tuiles exporté.

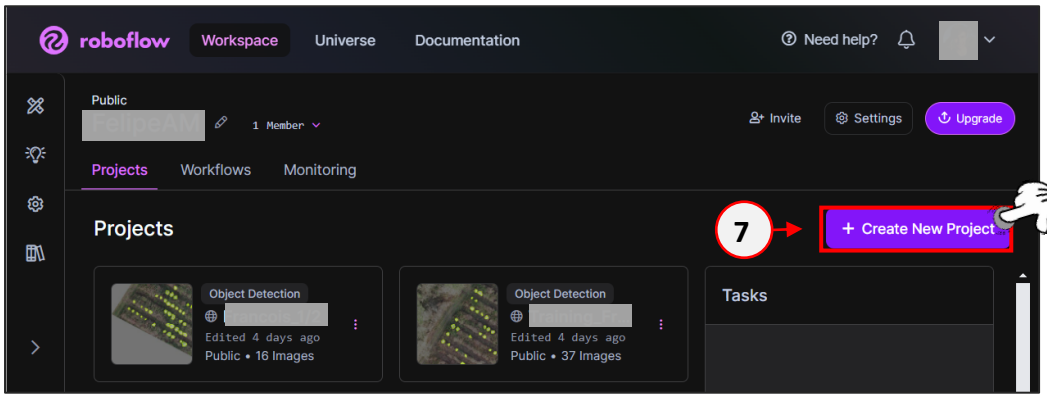
4.2. Labélisation

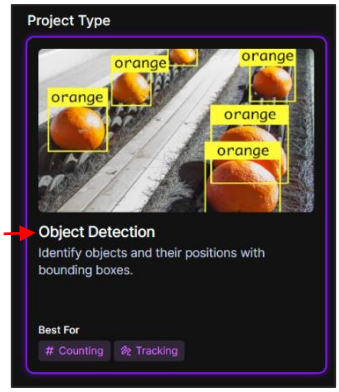
Cette étape consiste à indiquer sur les images la nature de chaque objet, ce qui permettra au modèle d'être entraîné selon vos instructions. Avec l'outil Roboflow, il est possible de réaliser l'étiquetage en utilisant une fonctionnalité de "Smart Polygon", qui permet de délimiter la région contenant l'objet et d'attribuer un nom à cette délimitation.

- Instructions :

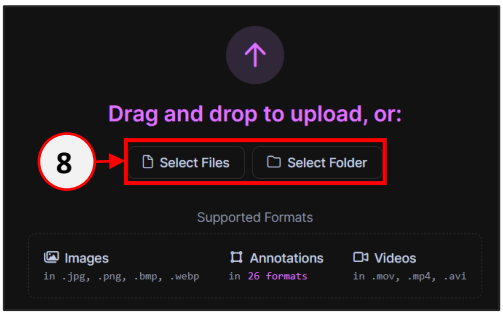
→ **6.** Créez un compte sur www.roboflow.com.

→ **7.** Créez un nouveau projet et choisissez le type "Object Détection".

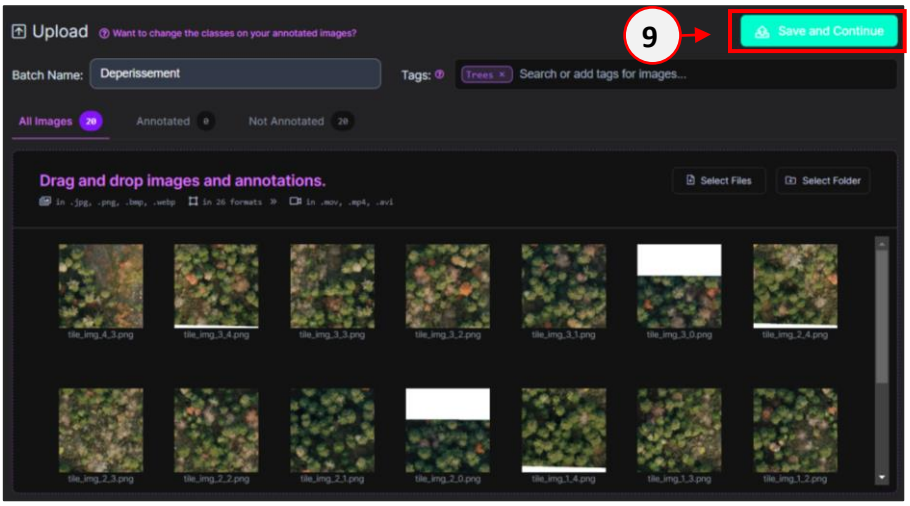


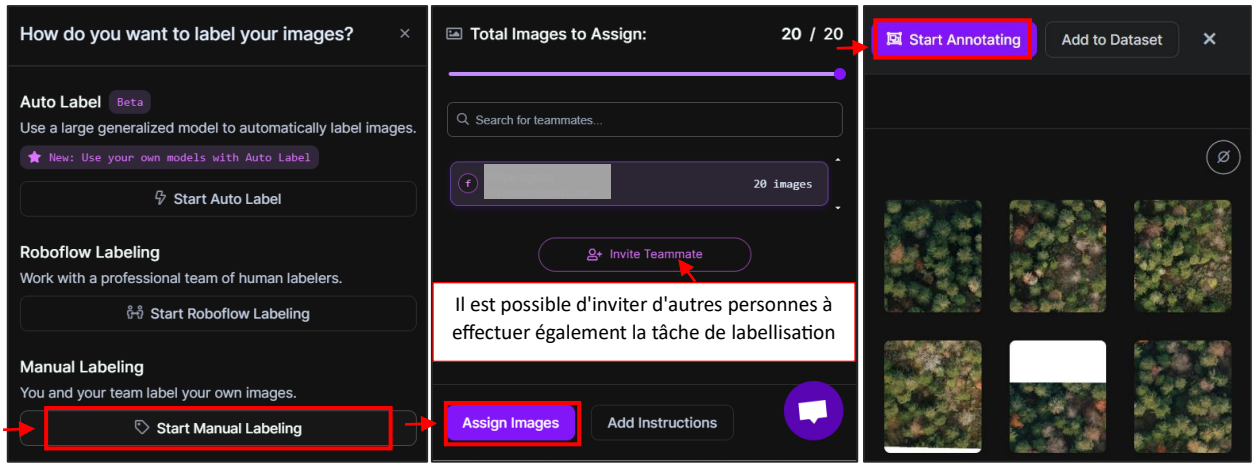


→ **8.** Allez sur "Select Files" et téléchargez vos tuiles. Vous voudrez peut-être les examiner et supprimer celles qui ne montrent pas votre objet.

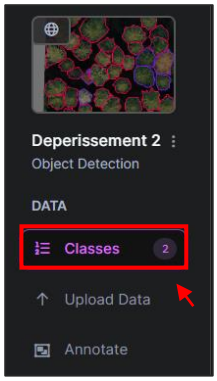


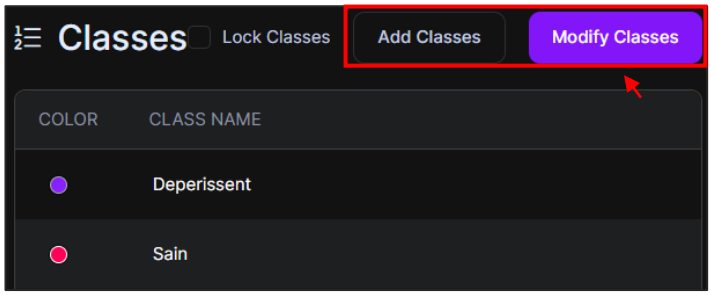
→ **9.** Appuyez sur "Save and Continue", puis sur "Start Manual Labeling" et "Assign Images", et enfin sur “Start Annotating”.



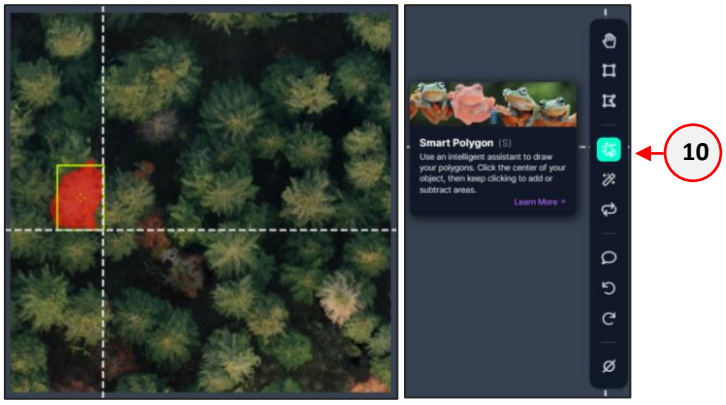


*Il est possible de créer, effacer et modifier les classes dans l'onglet 'Classes' indiqué dans la colonne à gauche.





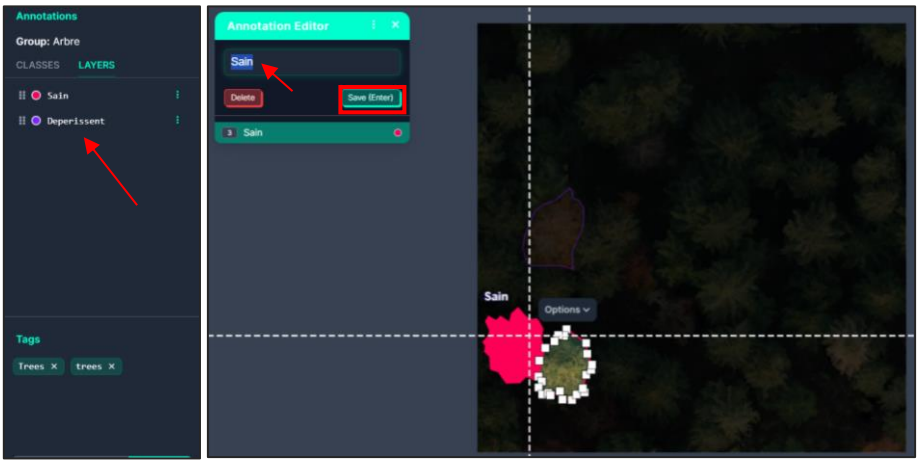
→ **10.** Sur le côté droit, choisissez la fonction "Smart Polygon", puis "Enhanced". Elle est alimentée par Segment Anything.



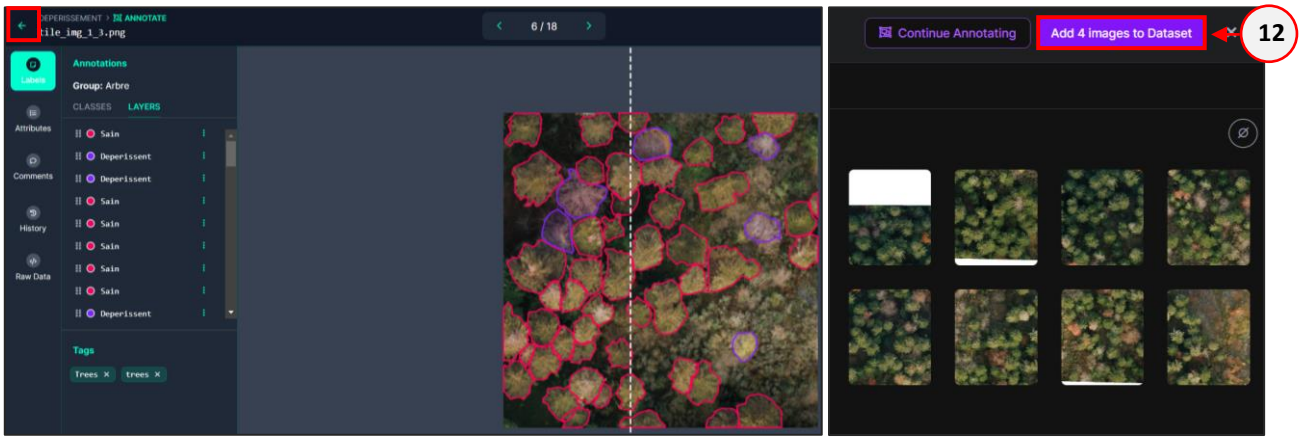
https://blog.roboflow.com/automated-polygon-labeling-computer-vision/

→ **11.** Cliquez sur votre objet. Si la sélection magique peut l'identifier comme un objet, vous pouvez vous attendre à ce que votre modèle fonctionne correctement. Si elle sélectionne une zone plus grande, vous devrez cliquer autour de votre cible pour l'aider à détecter l'objet désiré.

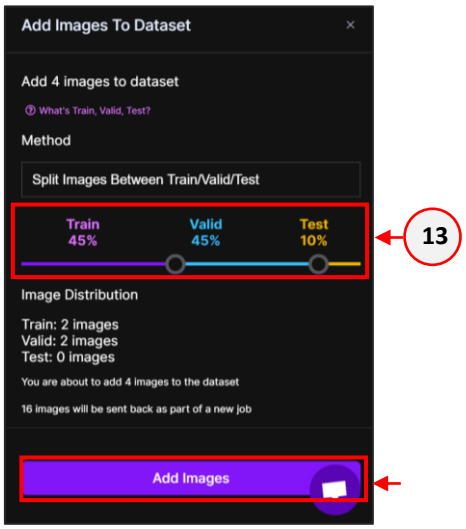
*Il est possible également la création des classes pendant la labélisation.



→ **12.** Lorsque vous avez terminé toutes les annotations, revenez en arrière et appuyez sur "Add X images to Dataset".



→ **13.** À cette étape, décidez du nombre d'images à utiliser pour l'entraînement, la validation et le test. Choisissez les proportions et cliquez sur 'Ajouter des images (Add Images)' :

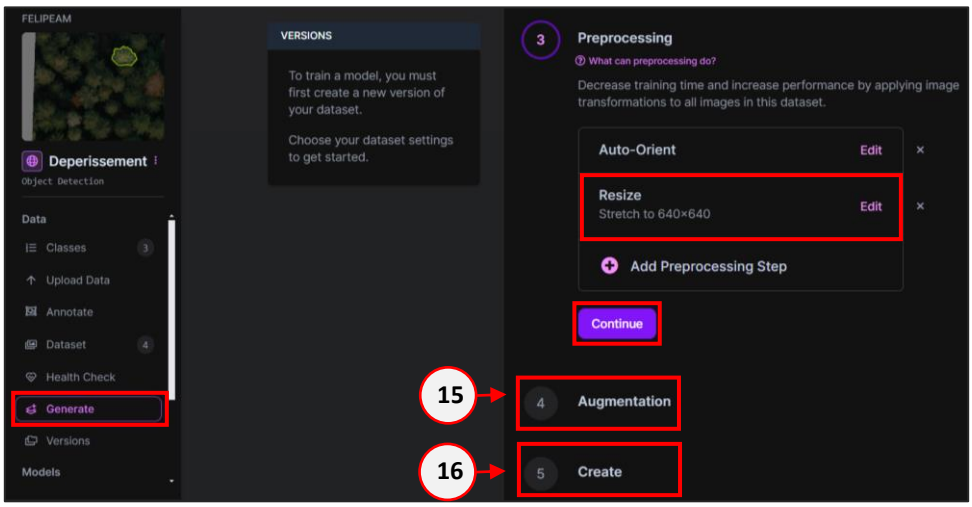


→ **14.** Dans l'onglet "Generate", à l'option "Preprocessing", assurez-vous que la résolution de redimensionnement est correcte pour vos images. Cliquez sur "Continue".

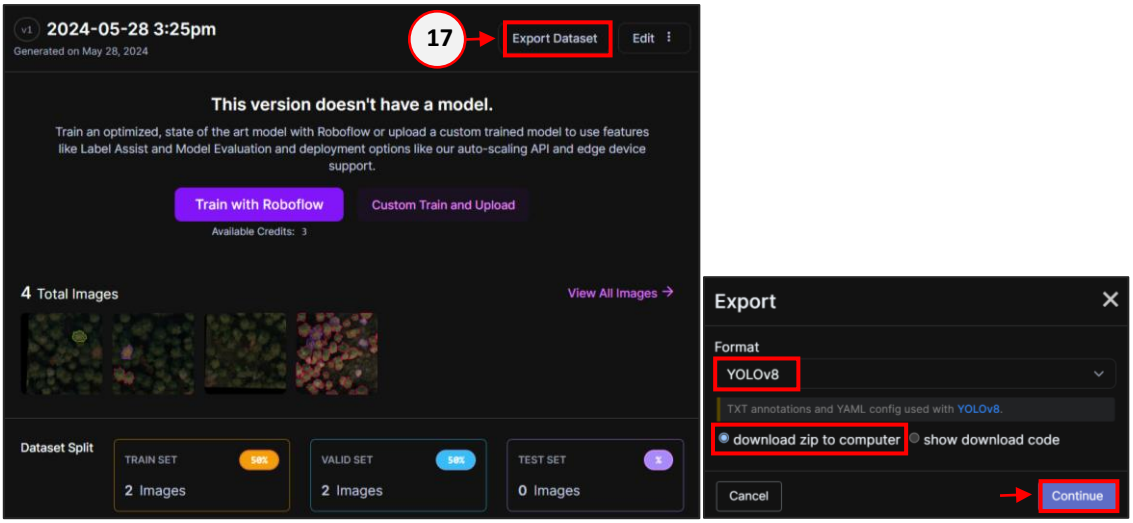
→ **15.** Dans "Augmentation", vous pouvez choisir différents facteurs affectant votre objet. Optionnel.

*Cela permet d'augmenter le nombre d'images d'entraînement en créant des images avec des caractéristiques variées, augmentant ainsi les variations de la même image.

→ **16.** Cliquez sur "Continue" puis "Create".



→ **17.** Cliquez sur "Export Dataset". Sous "TXT", choisissez soit YOLOv5 PyTorch, soit YOLOv8 (YOLOv5 PyTorch est optimal pour les images de 1280 px et YOLOv8 pour celles de 640 px). Cochez "download zip to computer" et "Continue".



4.3. Entrainement

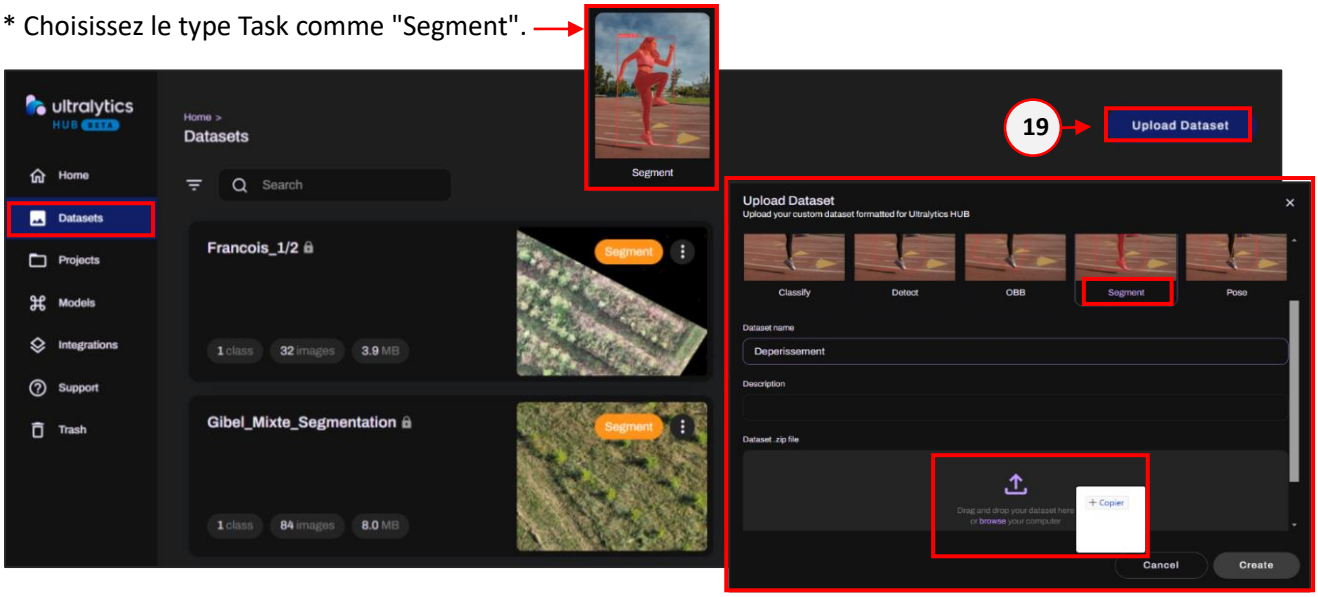
Ceci est l'étape d'entrainement à partir des tuiles labelisées précédemment, pour obtenir un modèle .onnx utilisable avec l'outil Deepness. En utilisant les algorithmes YOLO choisis, le modèle sera entraîné à reconnaître les caractéristiques identifiées et étiquetées. Cette étape comprend une première partie sur le site Ultralytics, suivie d'une autre partie en Python qui communique directement avec le site et se termine par l'exportation du modèle au format .onnx.

- Instructions :

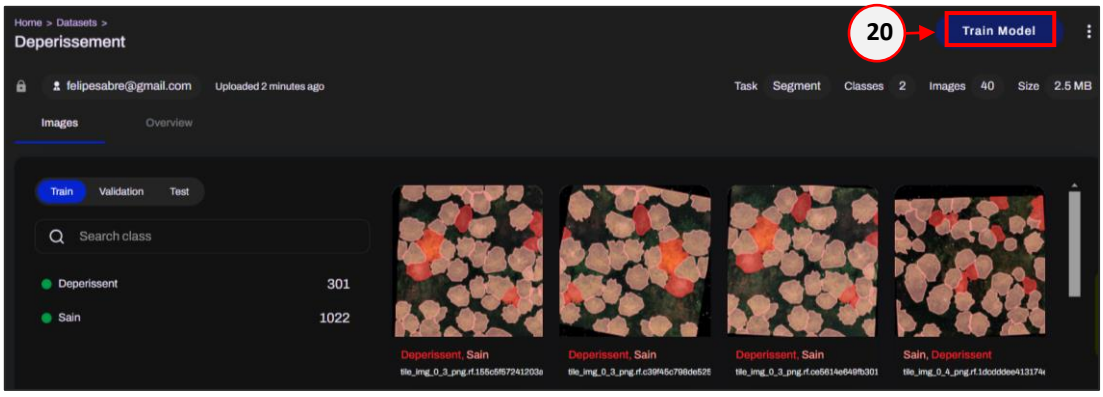
→ **18.** Créez un compte sur Ultralytics : https://ultralytics.com.

→ **19.** Allez dans "Datasets" puis "Upload Dataset" et choisissez votre fichier zip téléchargé.

* Choisissez le type Task comme "Segment".

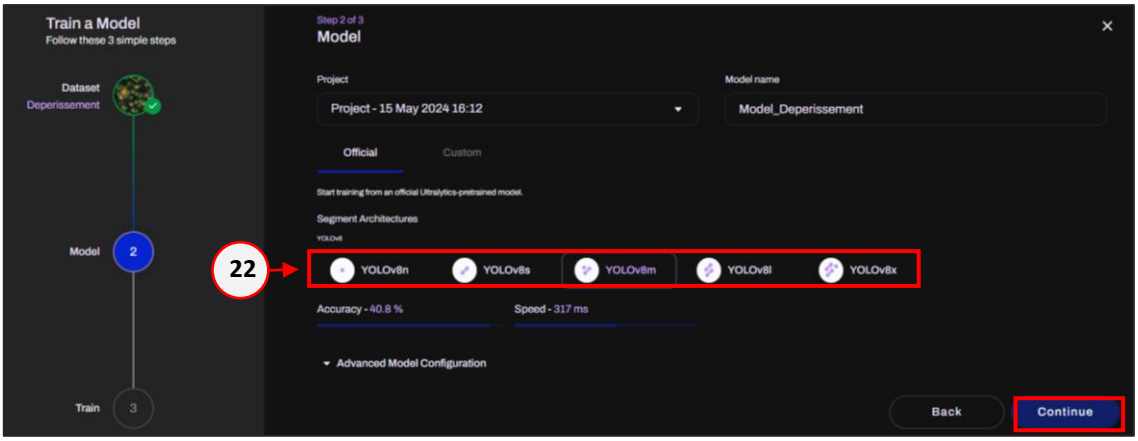


→ **20.** Cliquez sur votre dataset téléchargé puis sur "Train Model".

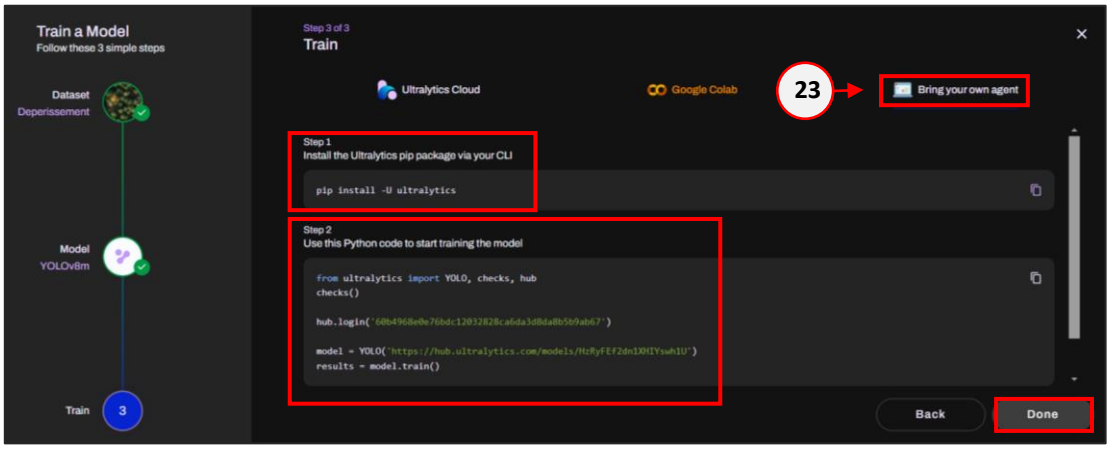


→ **21.** Choisissez votre modèle pré-entraîné. YOLOv8 (recommandé) et YOLOv5-6 sont optimaux pour les images de 1280 px.

→ **22.** Optez pour un modèle pré-entraîné moyen pour l'instant, soit YOLOv8m ou YOLOv5m6u, selon la taille de votre dataset et vos besoins en vitesse d'inférence. Consultez les paramètres Accuracy et Speed. Les plus petits sont meilleurs pour l'inférence en streaming et les plus grands pour les grands datasets.

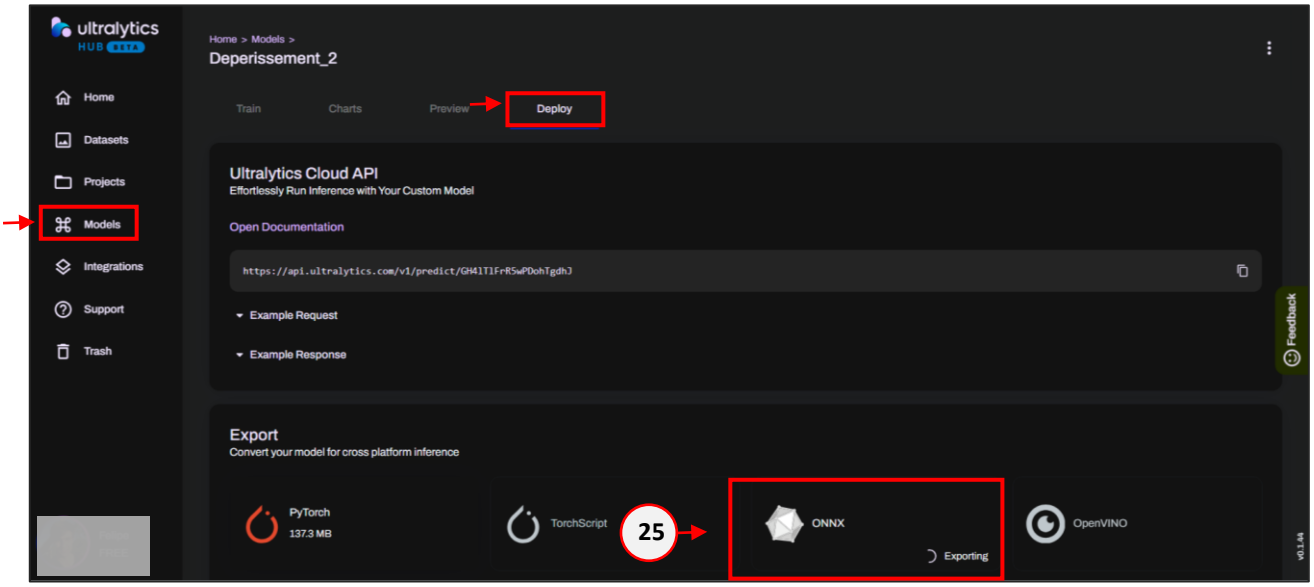


→ **23.** À l'étape suivante, entraînez votre modèle en utilisant Python en cliquant sur "Bring your own agent". Collez les codes proposés dans la console Python (Ctrl+C/V), en veillant à avoir pip installé et à installer la bibliothèque Ultralytic.



*Une fois que le code commence à s'exécuter dans la console Python, cliquez sur "Done" pour que les résultats s'affichent en direct sur Ultralytics.

→ **25.** Une fois votre modèle entraîné, allez dans “Models” et choisissez votre modèle. Allez dans “Deploy”, puis exportez et téléchargez le fichier “ONNX”.



4.4. Segmentation

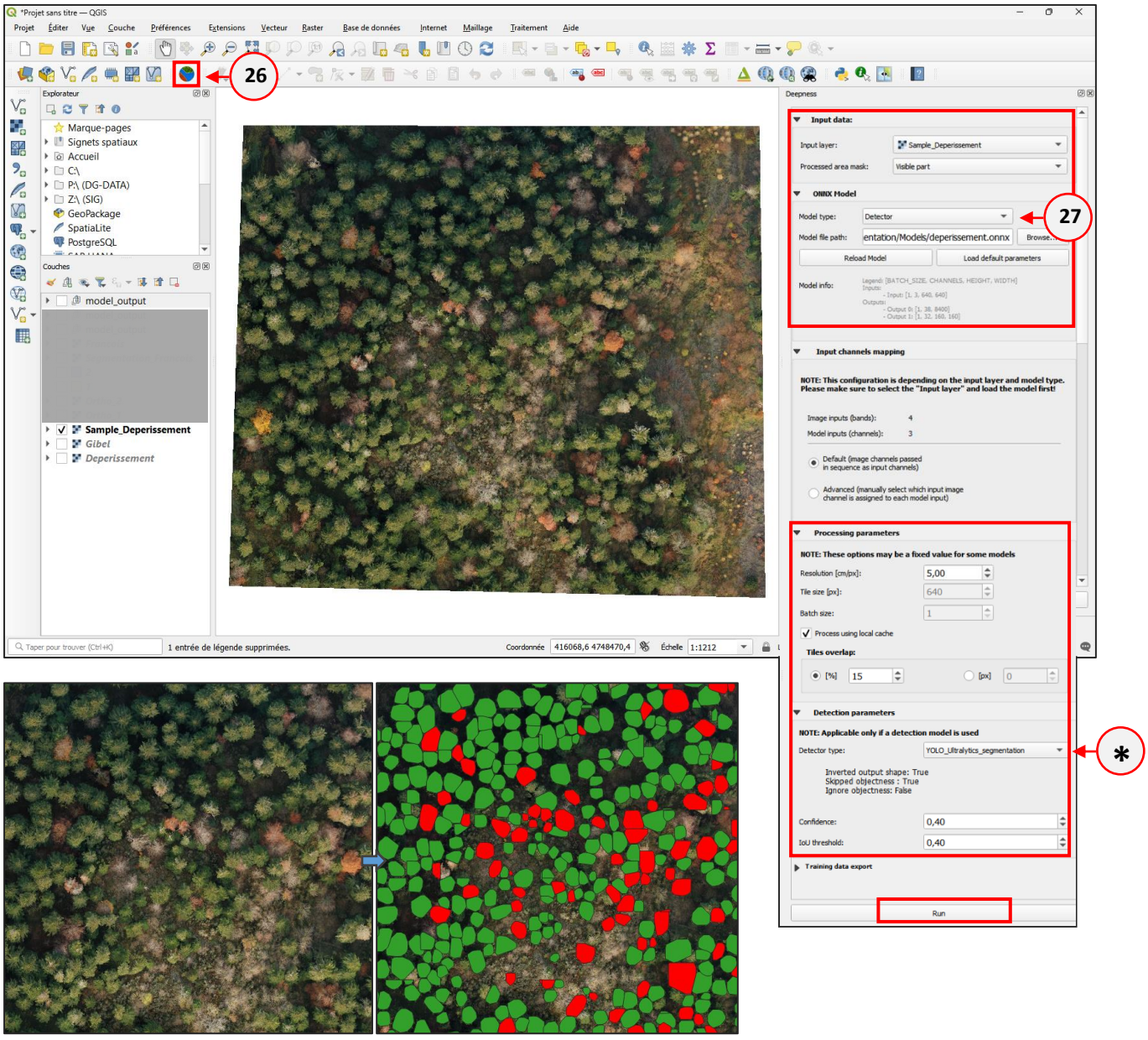
Cette étape consiste à appliquer le modèle entraîné pour segmenter l'image souhaitée. Il est intéressant d'appliquer la segmentation sur une zone différente de celle utilisée pour l'entraînement, mais contenant des caractéristiques similaires, puisque l'identification se base sur les cibles avec caractéristiques indiquées précédemment. Cette étape se déroule dans le logiciel QGIS, en utilisant l'outil du plug-in Deepness.

- Instructions :

→ **26.** Retournez dans QGIS et ouvrez Deepness. Importez votre orthophoto et sélectionnez la zone souhaitée.

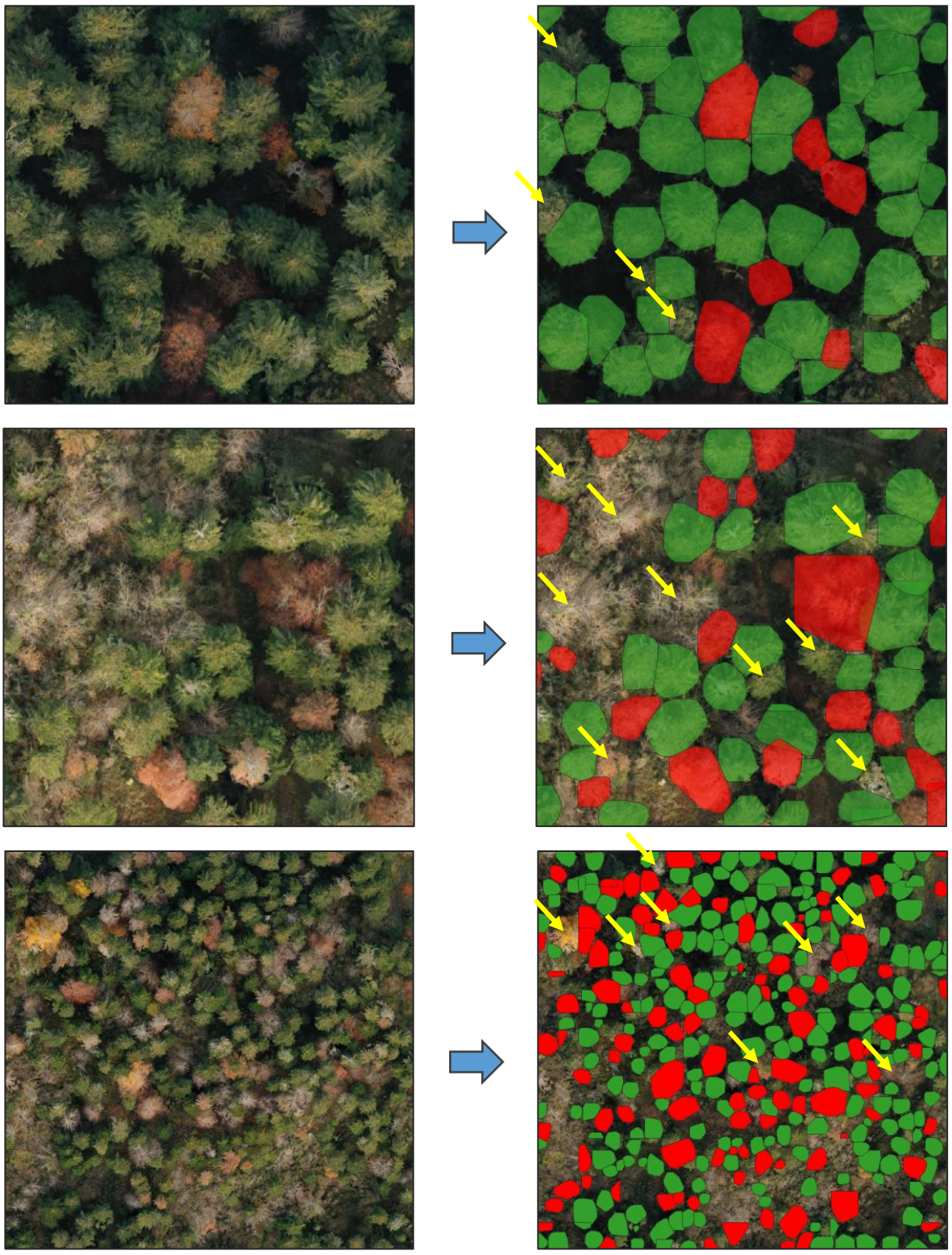
→ **27.** Ensuite, importez votre modèle de détection ONNX, avec le **Model Type = Detector**. Choisissez la résolution cm/px correcte et la taille des tuiles utilisées lors de l'entraînement de votre modèle.

*Choisissez le modèle de détection : **Detector type = YOLO_Ultralyitcs_segmentation**. Laissez le reste par défaut pour le moment. Faites défiler vers le bas et cliquez sur “Run”.



- Conclusion :

Ceci est une méthode simple pour entraîner et appliquer le traitement de segmentation basé sur l'objet. Il est important de se rappeler que la fiabilité et les performances sont meilleures lorsque les données utilisées pour l'entraînement du modèle sont plus nombreuses et de meilleure qualité. Cependant, cela montre qu'il peut y avoir encore des erreurs de segmentation. Il est donc nécessaire d'étudier la cause de ce problème, de faire des tests et d'améliorer l'entraînement des modèles.



Interprétation des résultats

Chaque métrique fournit des indications sur les domaines où le modèle peut être amélioré, que ce soit par des ajustements du modèle lui-même, de ses données d'entraînement, ou des méthodes utilisées pour la détection et la classification.

Indicateurs :

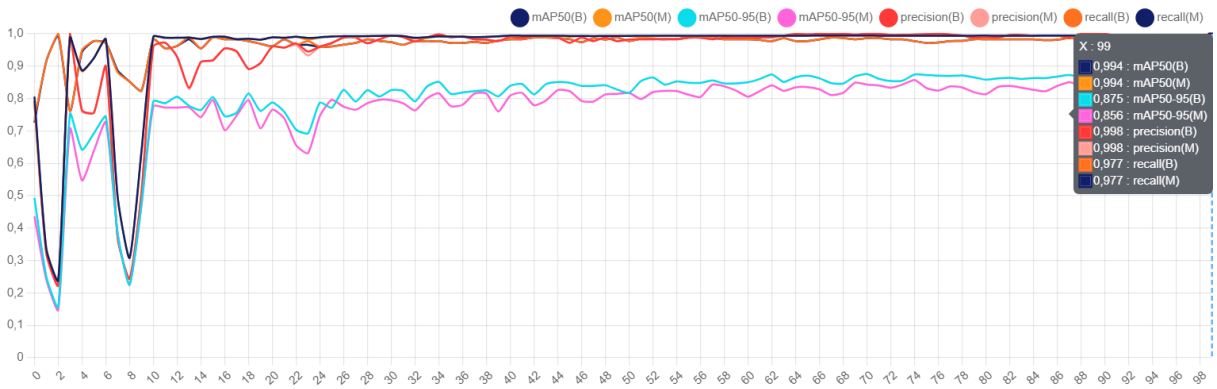
- **Précision (P)** : Mesure combien de détections sont correctes.
- **Rappel (R)** : Capacité du modèle à détecter tous les objets.
- **mAP50** : Précision moyenne à un seuil de 0,50 ; mesure les détections faciles.
- **mAP50-95** : Précision moyenne à des seuils de 0,50 à 0,95 ; évalue la performance globale.
- **IoU** : Indicateur de la précision de la localisation des objets.
- **F1 Score** : Équilibre entre précision et rappel.
- **(M)** : fait référence à la moyenne "macro", qui calcule la métrique de performance moyenne pour chaque classe.
- **(B)** : fait référencer à la meilleure métrique "best" de performance obtenue par le modèle durant l'entraînement.

Problèmes et solutions :

- **Faible mAP** : Le modèle a besoin d'améliorations générales.
- **Faible IoU** : Le modèle ne localise pas bien les objets ; améliorer les boîtes englobantes.
- **Faible Précision** : Trop de fausses détections ; ajuster les seuils de confiance.
- **Faible Rappel** : Le modèle rate des objets réels ; utiliser plus de données.
- **F1 Score déséquilibré** : Précision et rappel ne sont pas équilibrés.
- **Faible AP pour certaines classes** : Utiliser plus de données pour ces classes ou ajuster les poids pendant l'entraînement.

Exemples de cas :

- **Cas 1** : Faible Précision mais bon Rappel ; ajuster les seuils de confiance.
- **Cas 2** : Bon Rappel mais faible IoU ; améliorer la localisation des objets.
- **Cas 3** : Certaines classes ont un AP bas ; utiliser plus de données pour ces classes.



Sources : https://docs.ultralytics.com/guides/yolo-performance-metrics/#introduction