

# Table de matières :

[Table de matières : 2](#_TOC_250016)

[Traitement d’images: 3](#_TOC_250015)

[Contexte: 3](#_TOC_250014)

[Les algorithmes: 3](#_TOC_250013)

[Les étapes suivies lors du traitement du dataset: 4](#_TOC_250012)

[Installation des dépendances Yolov5: 4](#_TOC_250011)

[Prétraitement des données: 4](#_TOC_250010)

1. [Seuillage, Redimensionnement, Convolution de contraste 4](#_TOC_250009)
2. [Utilisation du prétraitement offert par la plateforme: 6](#_TOC_250008)

[Traitement des données: 7](#_TOC_250007)

[Utilisation de la plateforme pour traiter les données: 7](#_TOC_250006)

[Prétraitement: 10](#_TOC_250005)

[Augmentation: 11](#_TOC_250004)

[Entraînement du modèle personnalisé YOLOv5: 13](#_TOC_250003)

[Les résultats du modèle sur les images de test: 14](#_TOC_250002)

[Test du modèle de Roboflow: 16](#_TOC_250001)

[Conclusion: 18](#_TOC_250000)

# Traitement d’images:

Le traitement d'images dans le cadre de la science des données implique l'utilisation d'algorithmes d'apprentissage automatique et de traitement d'images pour extraire des informations utiles à partir d'images numériques. Il peut inclure des tâches telles que la reconnaissance d'image, la segmentation d'image, la classification d'image, la génération d'image et la reconstruction d'image.

# Contexte:

Dans le cadre du traitement d’image pour apprentissage profond (deep learning), nous allons traiter un dataset constitué d’un ensemble de données de radiographies dentaires panoramiques anonymes et dépersonnalisées de 116 patients, prises au Noor Medical Imaging Center, à Qom, en Iran. Les sujets couvrent un large éventail de conditions dentaires, allant de la santé à l'édentation partielle ou complète.

# Les algorithmes:

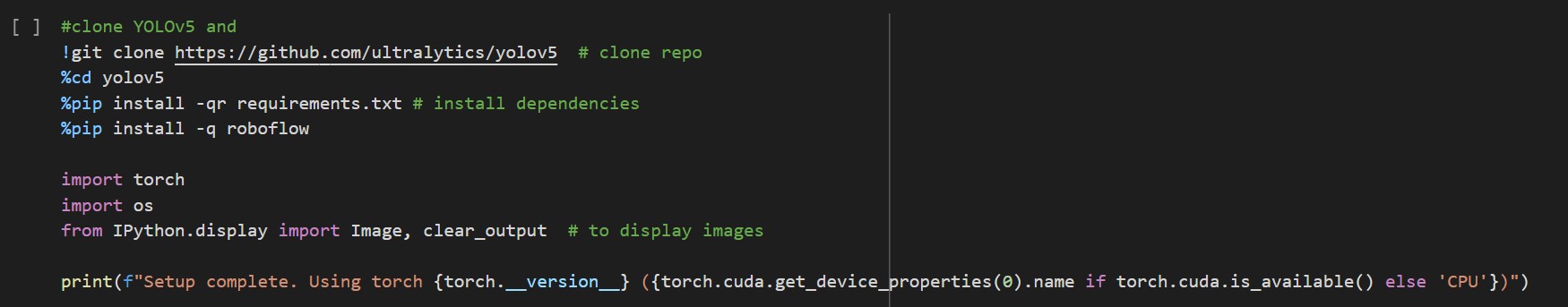
YOLO (You Only Look Once) est un système populaire de détection d'objets en temps réel. YOLOX n'est pas une variante connue de YOLO, mais il pourrait s'agir d'une implémentation personnalisée. Pour entraîner YOLO sur des objets personnalisés, on doit suivre les étapes générales suivantes :

1. Collecter et annoter un jeu de données d'images contenant les objets que nous souhaitons détecter. Les annotations doivent inclure les coordonnées des boîtes englobantes des objets dans les images.
2. Diviser le jeu de données en ensembles d'entraînement et de test.
3. Télécharger les poids et les fichiers de configuration pré-entraînés de YOLO et les utiliser comme point de départ pour l'entraînement de notre modèle.
4. Utiliser un outil comme Darknet pour entraîner le modèle YOLO sur notre jeu de données annoté.
5. Ajuster le modèle en utilisant l'ensemble de test et ajuster les paramètres pour obtenir les meilleures performances.
6. Une fois le modèle entraîné, nous pourrons l'utiliser pour détecter les objets personnalisés dans de nouvelles images.

# Les étapes suivies lors du traitement du dataset:

1. Installer les dépendances Yolov5
2. Assembler notre jeu de données
3. Entraîner notre modèle personnalisé YOLOv5

## Installation des dépendances Yolov5:

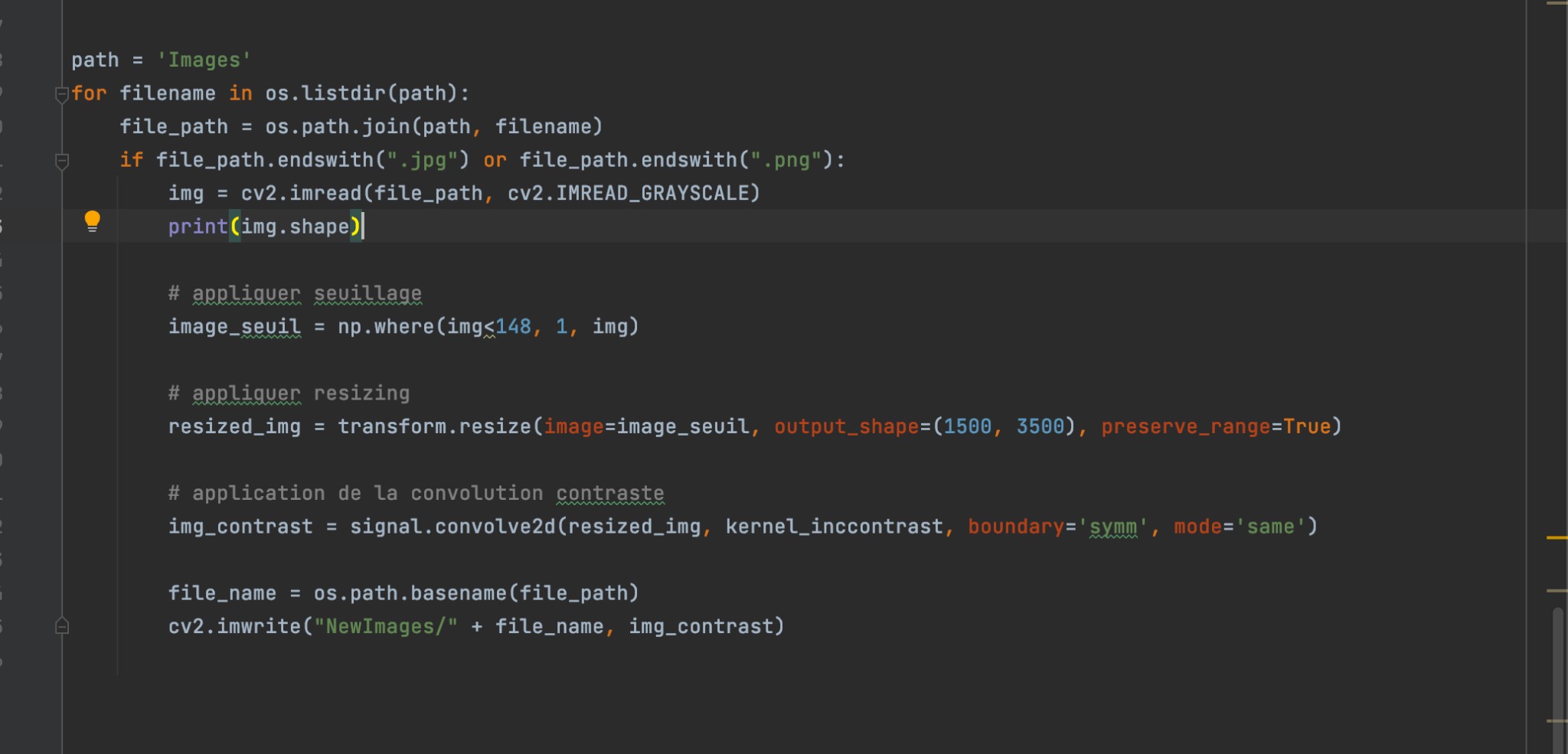


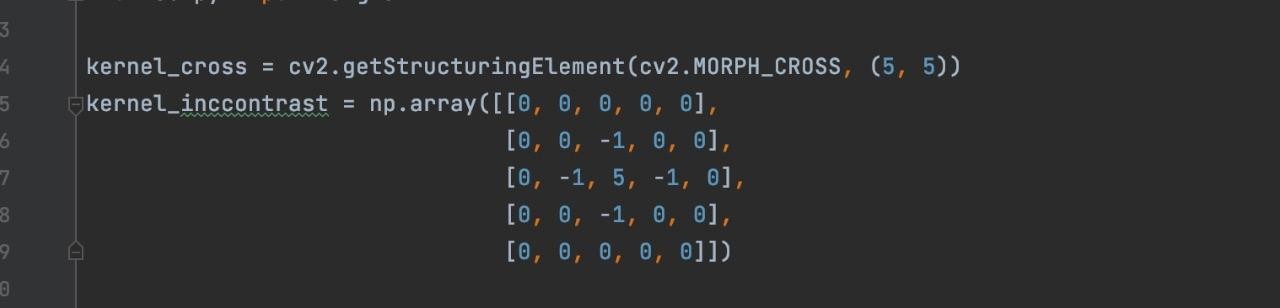
On clone le dépôt YOLOv5 de GitHub, on navigue par la suite dans le répertoire du dépôt et installe les dépendances requises pour le projet. On utilise également l'installateur de packages pip pour installer la bibliothèque Roboflow. On importe également les bibliothèques nécessaires telles que torch, os et IPython.display. Enfin, on affiche la version de torch et le périphérique (CPU ou GPU) qui est utilisé.

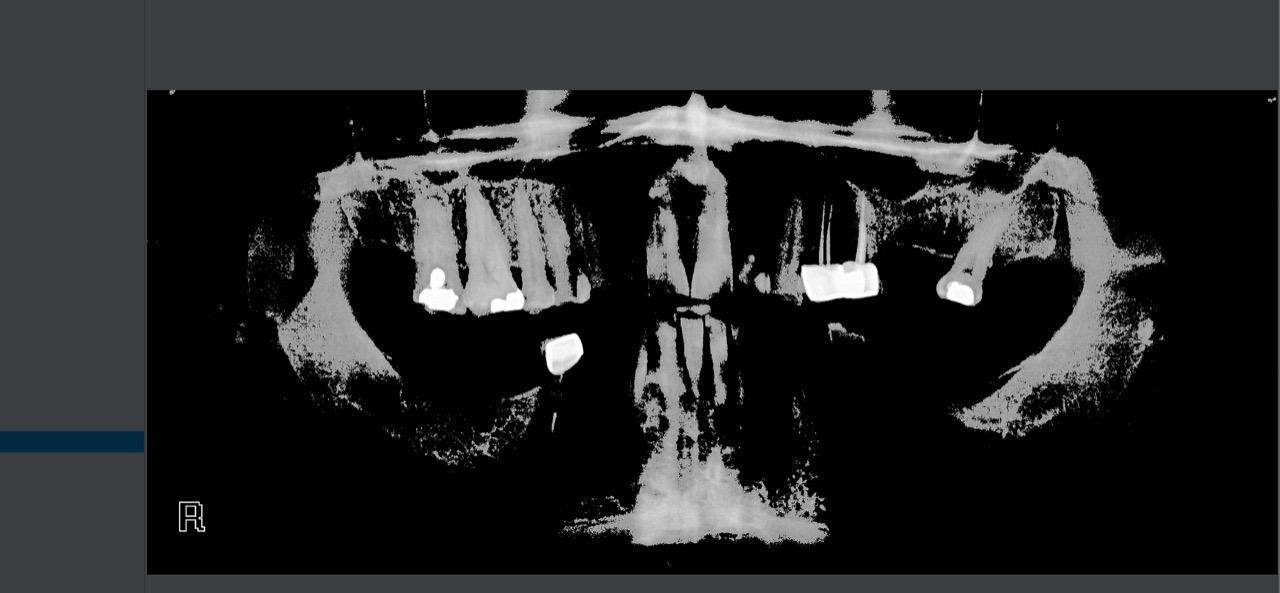
**Prétraitement des données:**

On a effectué le prétraitement des images en utilisant deux méthodes que nous allons détailler par la suite:

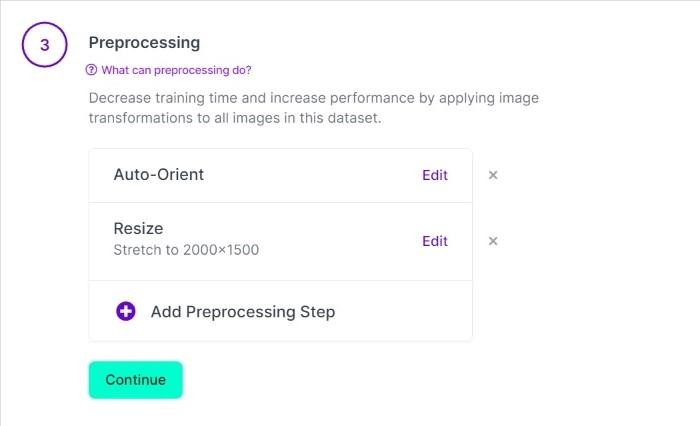
### Seuillage, Redimensionnement, Convolution de contraste

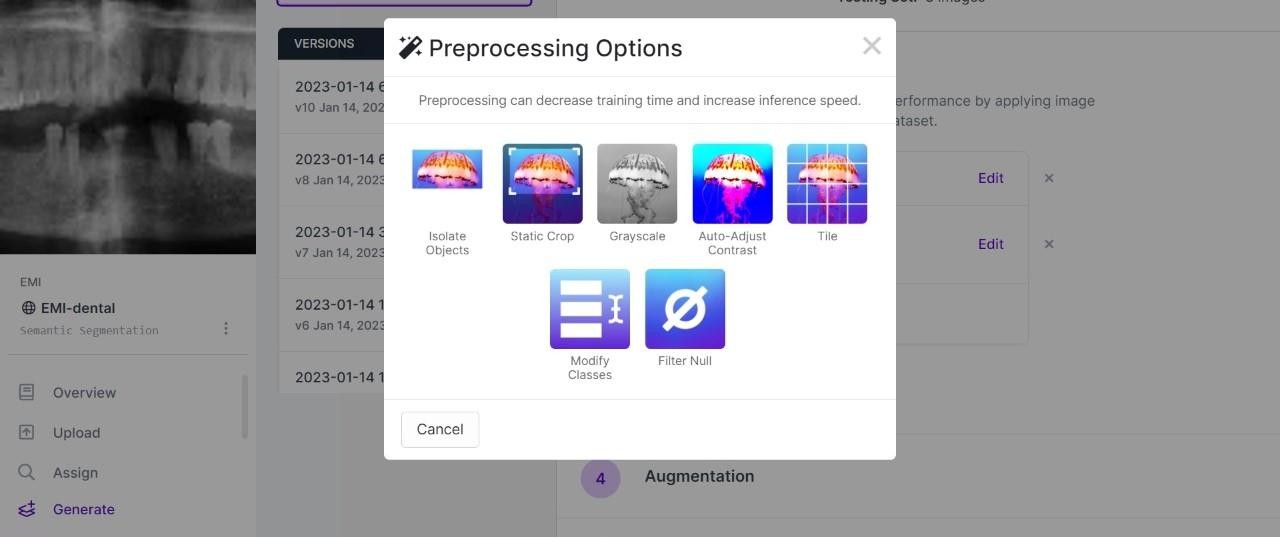






1. **Utilisation du prétraitement offert par la plateforme:**





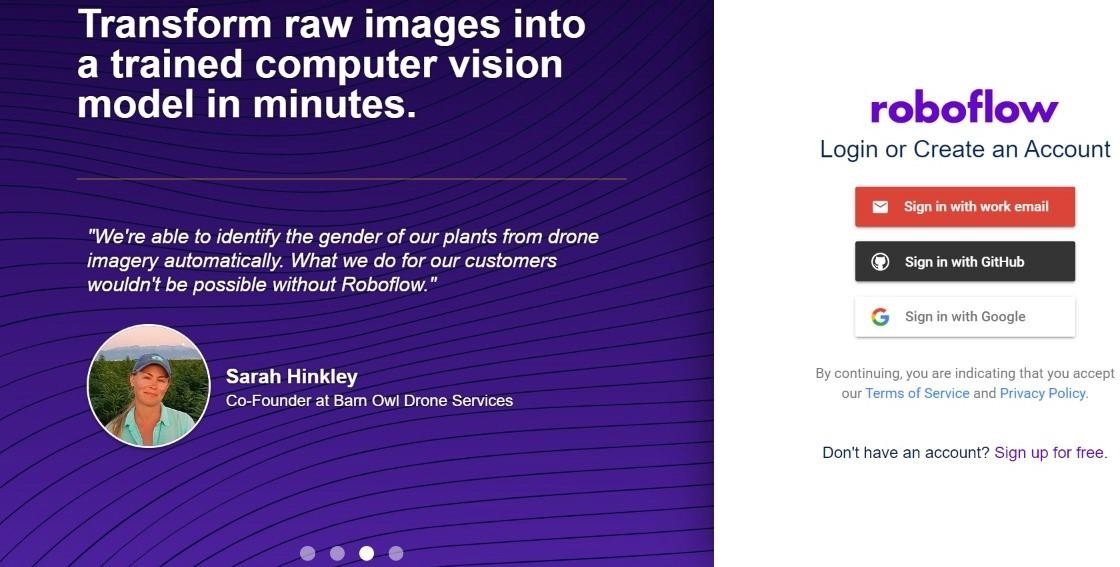
**Traitement des données:**

Roboflow est une plate-forme de vision par ordinateur pour l'annotation et la gestion des données. Il propose une interface web pour annoter des images et des vidéos, ainsi qu'un ensemble d'API pour accéder et gérer les jeux de données de manière programmatique. Il peut également aider à automatiser le processus d'annotation, le prétraitement et le

post-traitement des données, et peut aider à formater les données pour fonctionner de manière transparente avec les frameworks d'apprentissage automatique populaires, y compris YOLO.

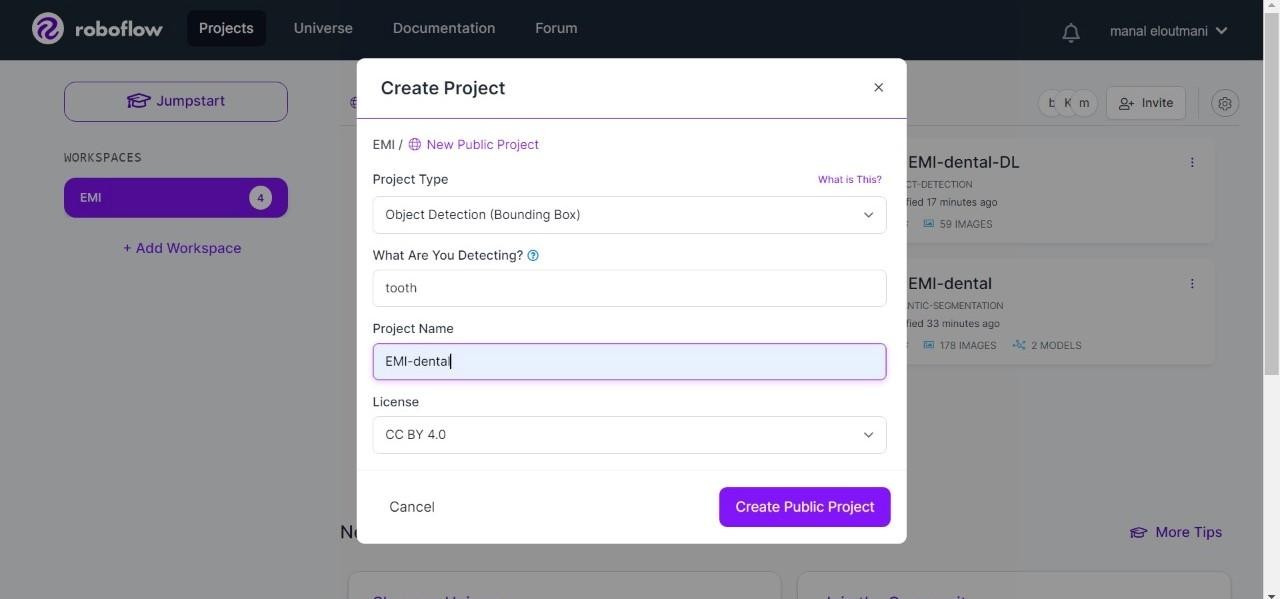
### Utilisation de la plateforme pour traiter les données:

Afin de profiter des fonctionnalités proposées par Roboflow, on doit, dans un premier temps, créer un compte sur la plateform:

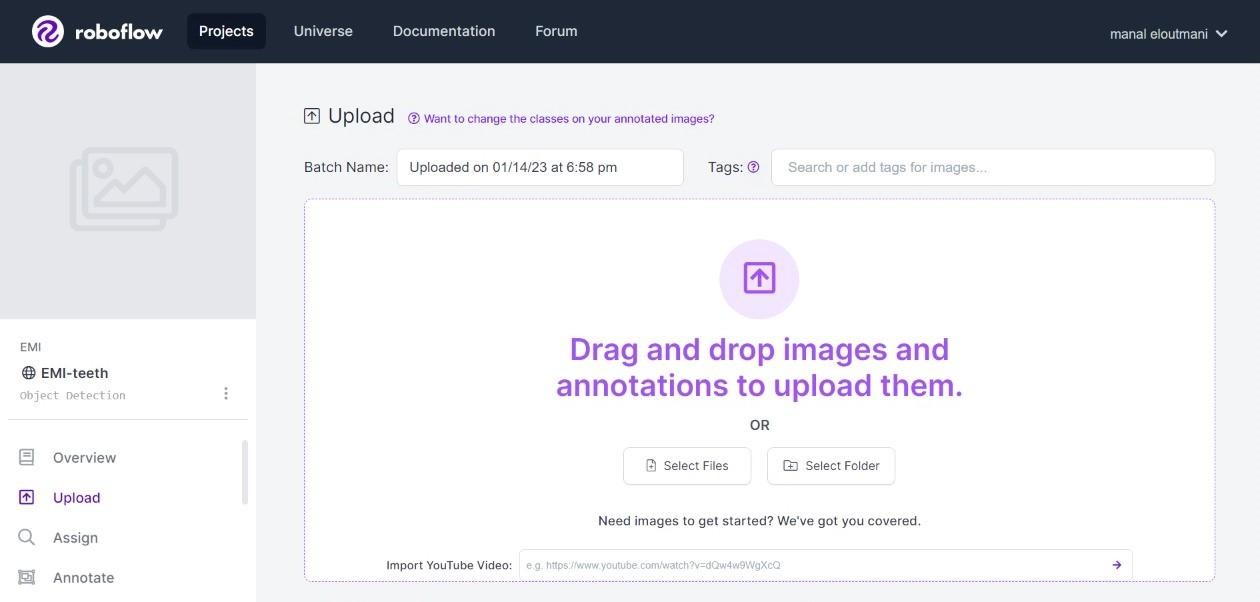


Après avoir créer un compte, on passe à la création du projet: On renseigne le type du projet

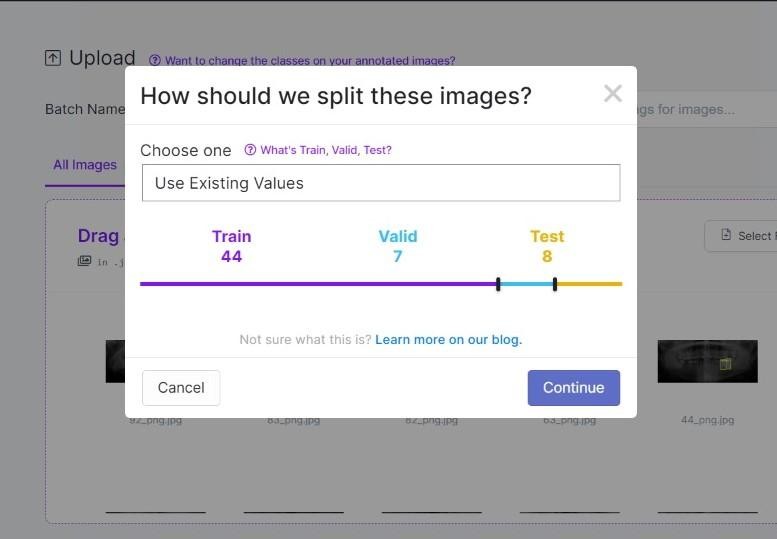
= Object Detection, les objets à détecter, le nom du projet ainsi que la licence.



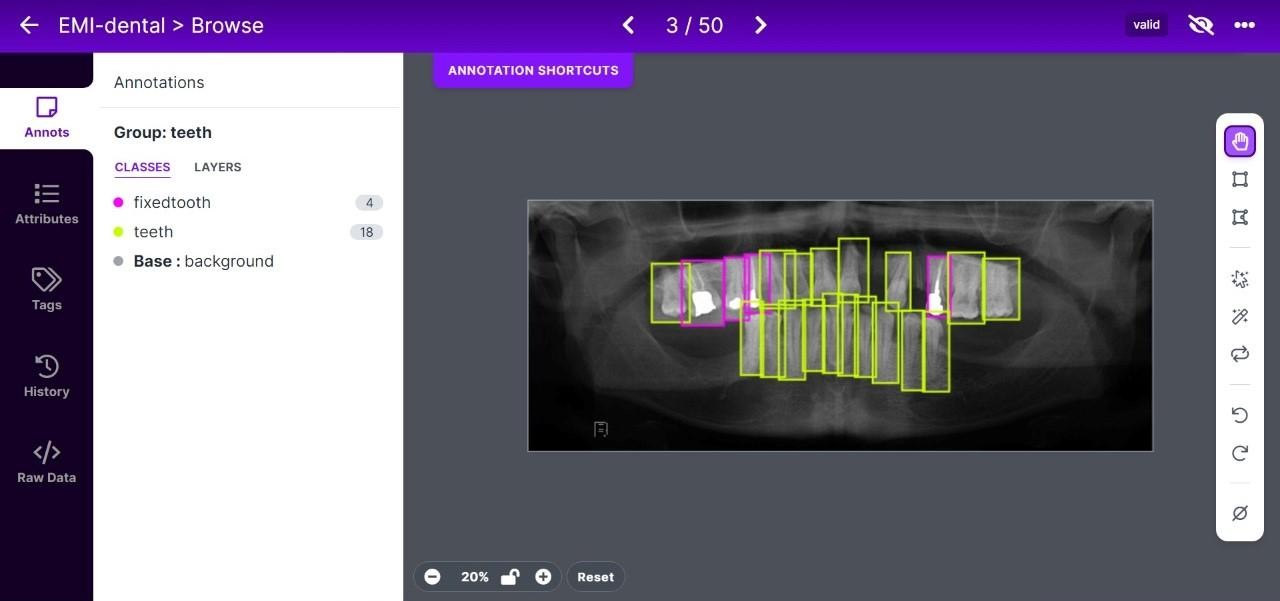
On télécharge par la suite les images qu’on souhaite traiter:



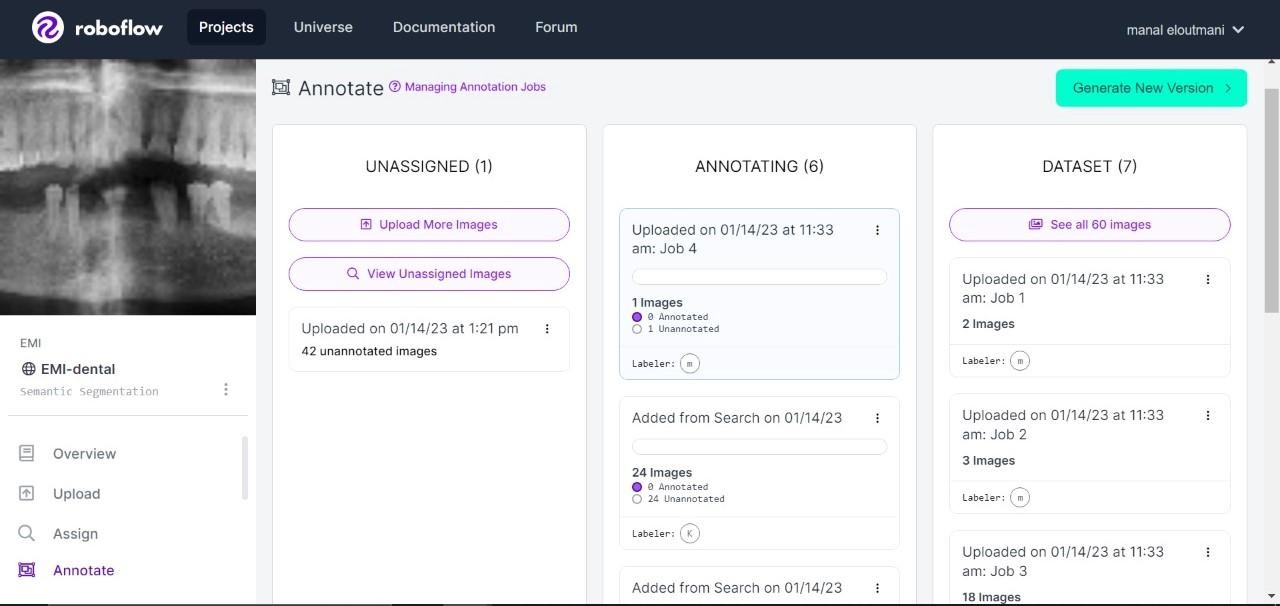
Afin d'entraîner notre modèle ainsi de le tester par la suite, on a précisé le pourcentage des données dédiés à l'entraînement et celles au test:



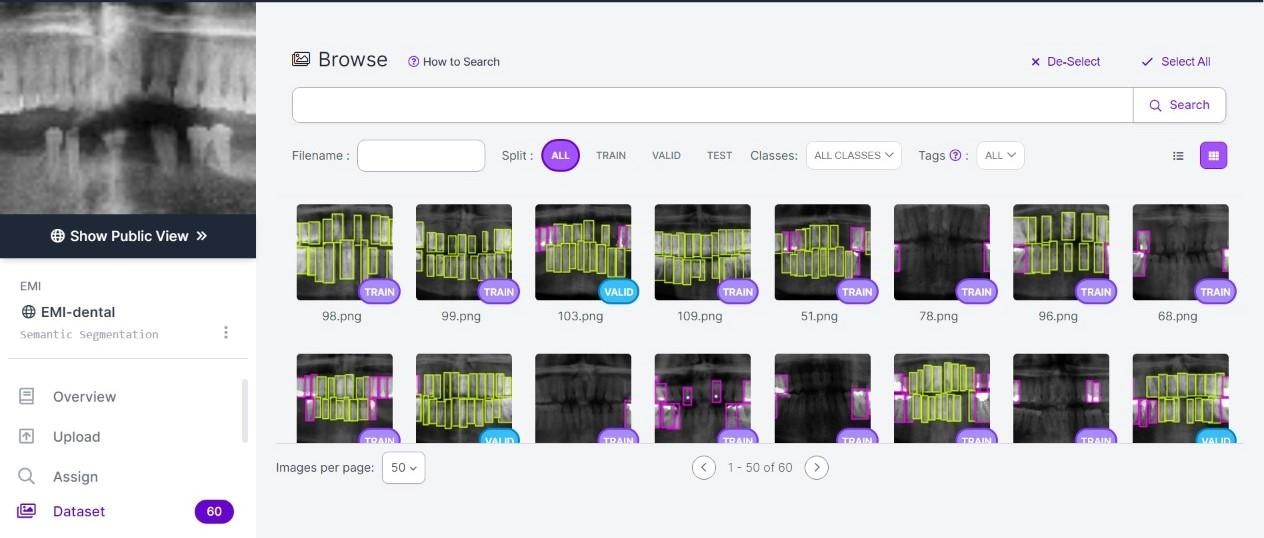
On passe ensuite à l’annotation des images téléchargées. On a choisi deux classes d’annotation: tooth pour faire référence aux dents saines et fixedtooth pour référencer les dents corrigées:



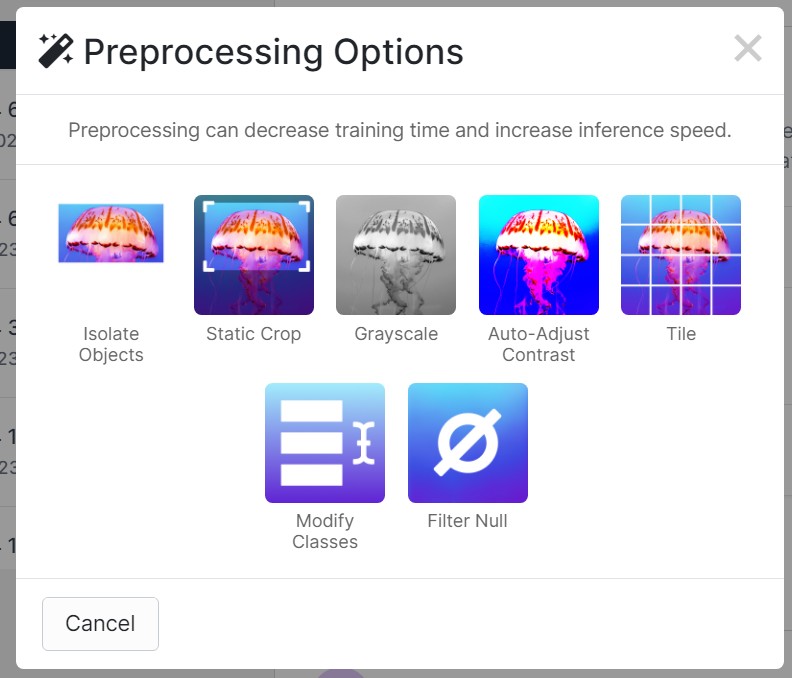
Afin d'accélérer l’annotation des données, puisqu'il s’agit d’une tâche manuelle et répétitive, nous avons réparti la tâche entre les membres du groupe:



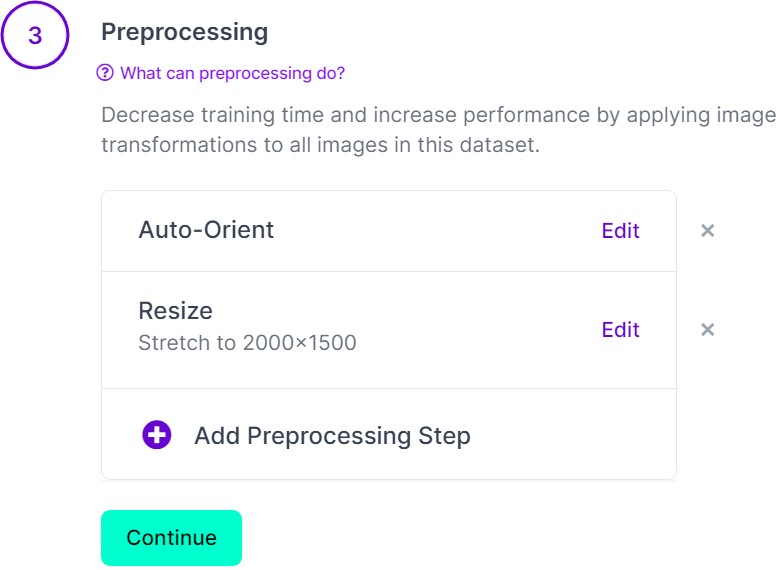
Après avoir finir l’annotation des images, on les ajoute au niveau du dataset:



**Prétraitement:**

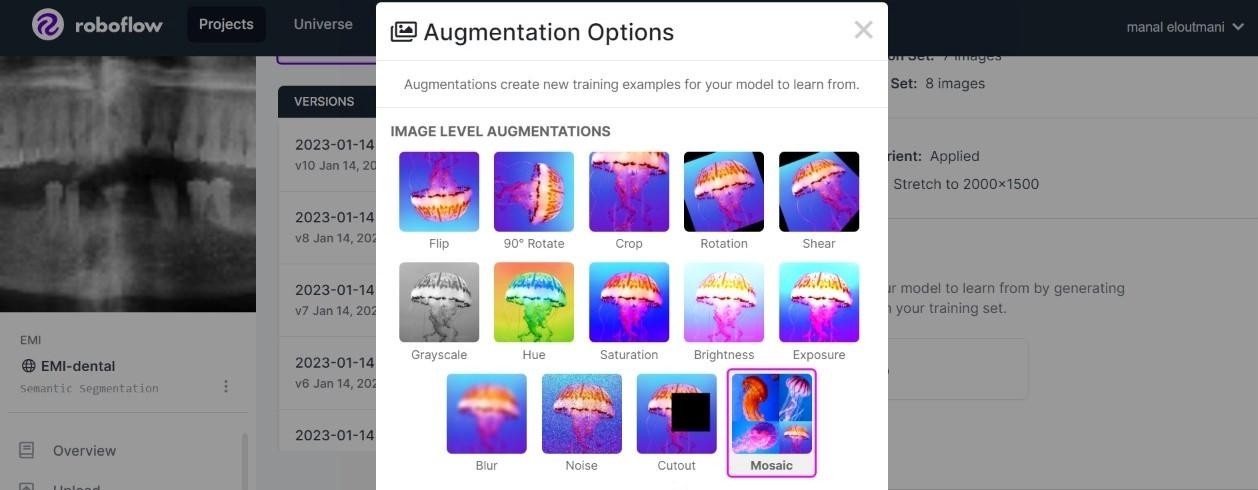
Roboflow offre les différents prétraitements suivants:

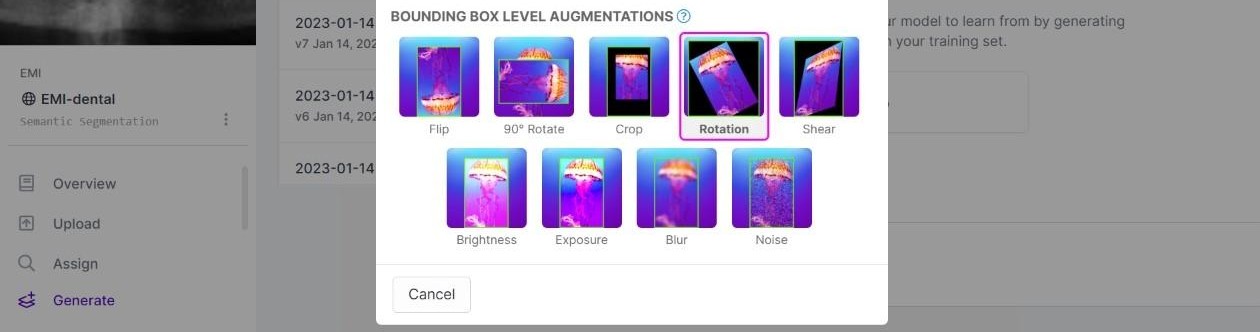
Ainsi que le Resizing:



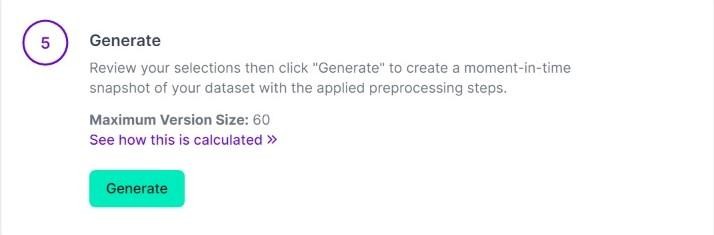
**Augmentation:**

Il offre aussi les différentes fonctionnalités d’augmentation

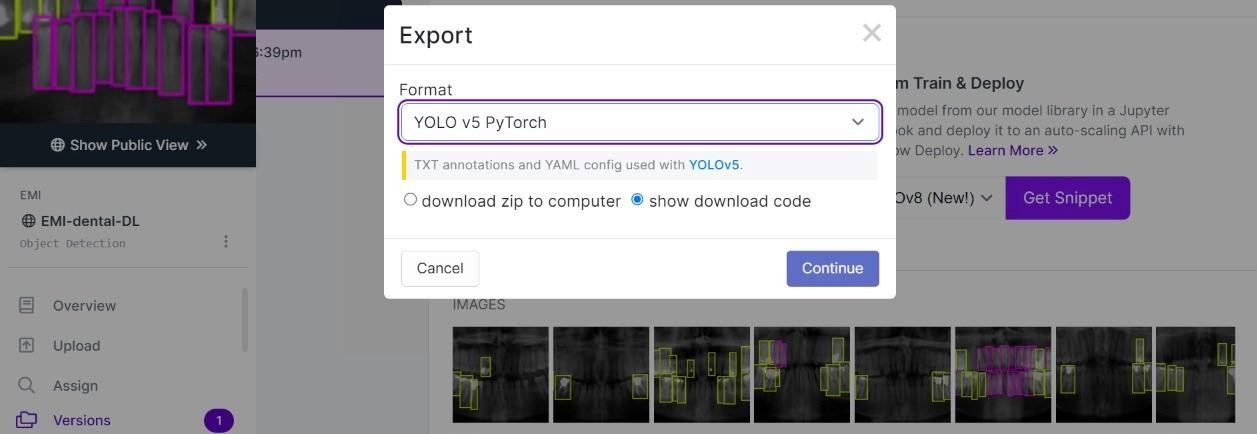




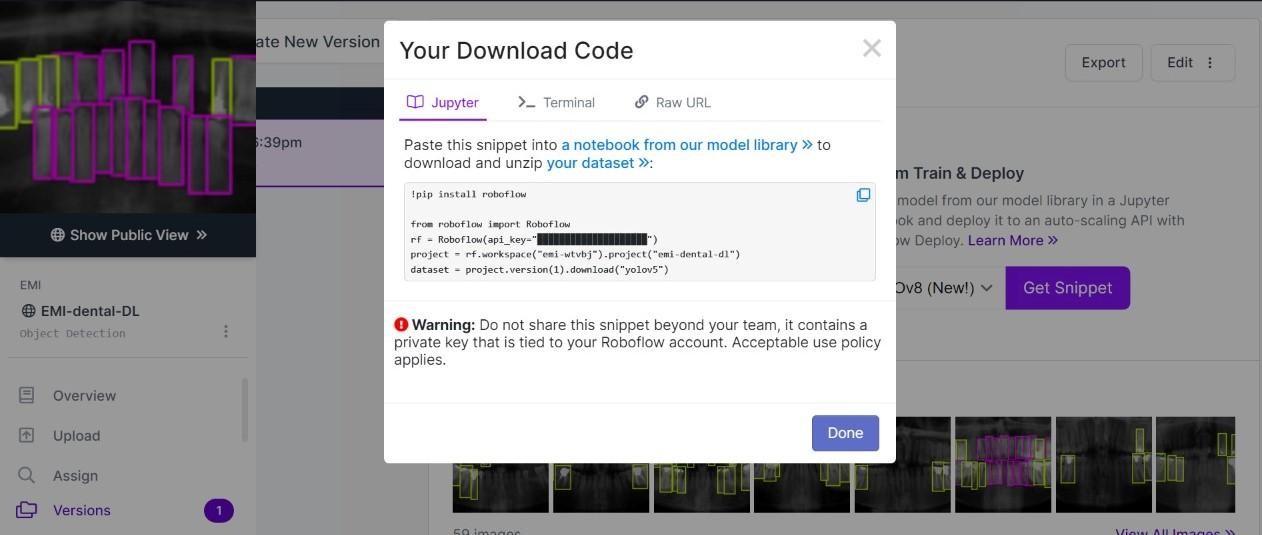
On passe à la génération du dataset pour l’utiliser dans l'entraînement des données:



On choisit le format du dataset qui est en l'occurrence YOLO v5 PyTorch :



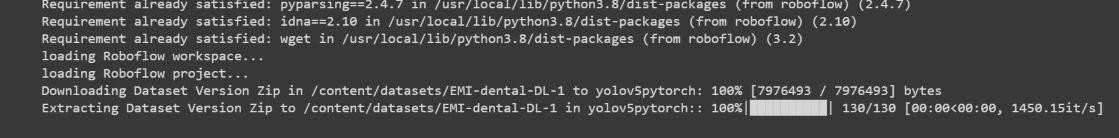
On génère donc le code qui permet la récupération du dataset:



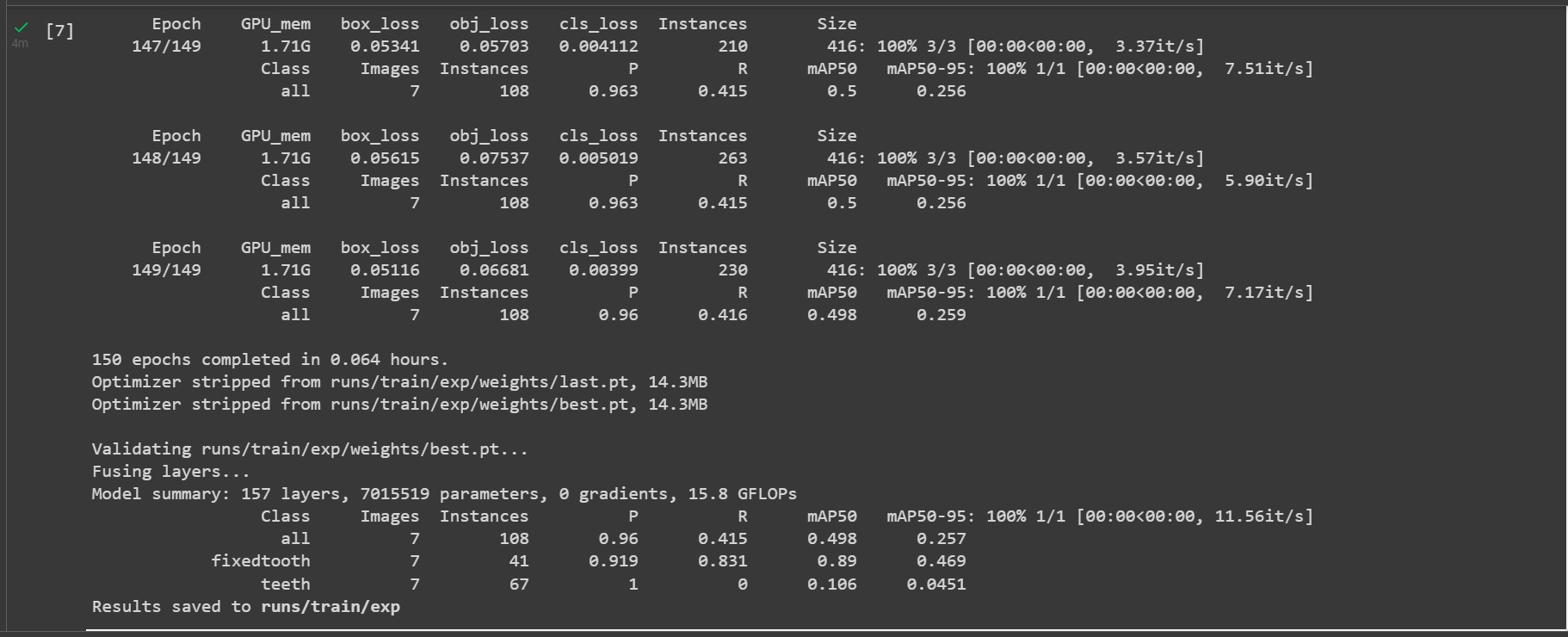
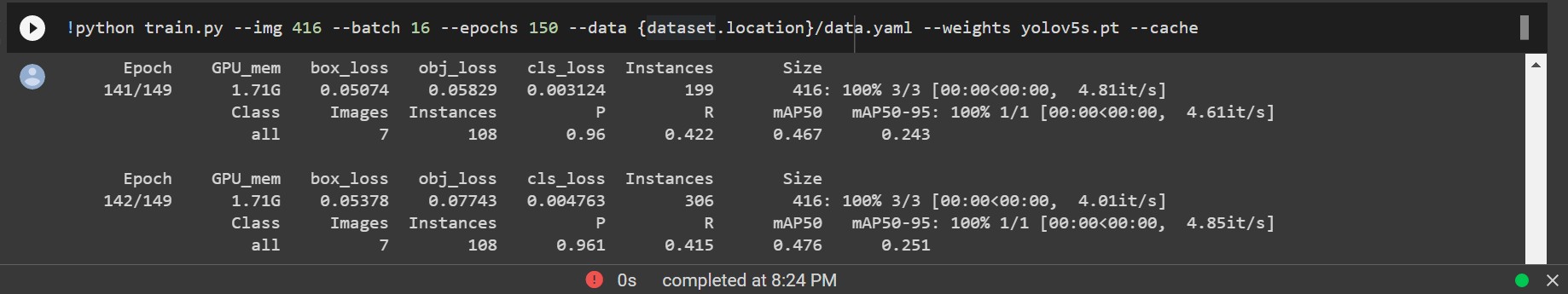
On copie le code généré dans note notebook:



On récupère avec succès notre dataset :

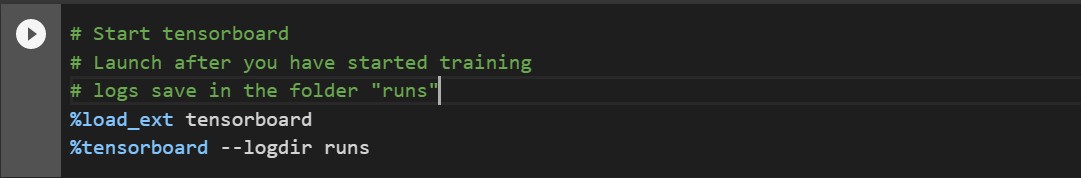


**Entraînement du modèle personnalisé YOLOv5:**



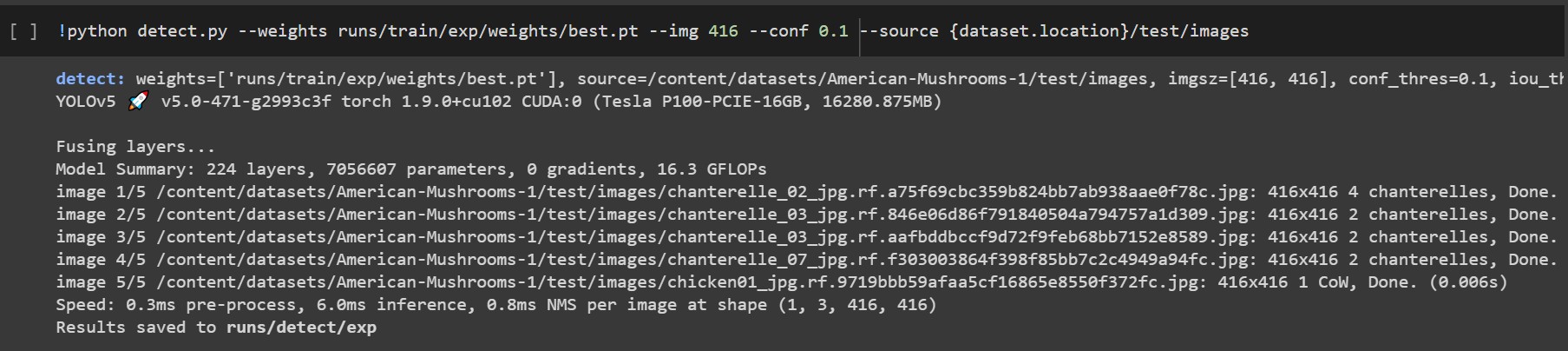
On utilise la commande ci-dessus pour exécuter le script train.py, qui entraîne un modèle YOLOv5 sur un ensemble de données donné.

* Le flag --img est utilisé pour définir la taille des images d'entrée à 416 pixels.
* Le flag --batch est utilisé pour définir la taille du lot à 16.
* Le flag --epochs est utilisé pour définir le nombre d'époques d'apprentissage à 150.
* Le flag --data est utilisé pour spécifier l'emplacement du fichier de configuration des données, dans ce cas, un fichier nommé data.yaml dans l'emplacement du jeu de données.
* Le flag --weights est utilisé pour spécifier les poids initiaux à utiliser pour le modèle.
* Le flag --cache est utilisé pour activer la mise en cache de l'ensemble de données, ce qui accélère le processus d'apprentissage.



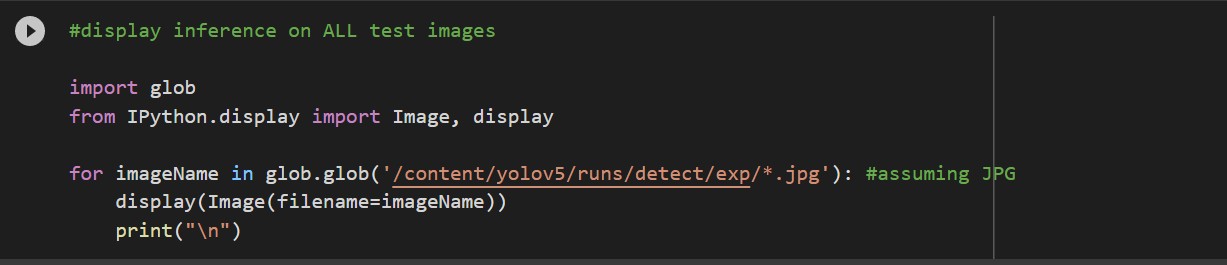
Ce code est utilisé pour lancer TensorBoard, un outil Web permettant de visualiser et d'analyser les résultats de l'entraînement d'un modèle d'apprentissage automatique. La commande **%load\_ext tensorboard** charge l'extension TensorBoard dans le notebook Jupyter actuel. La commande **%tensorboard --logdir runs** lance TensorBoard et spécifie le répertoire dans lequel les logs d'entraînement sont enregistrés. Dans ce cas, les logs sont enregistrés dans un dossier nommé "runs". Une fois que TensorBoard est lancé, il s'ouvre

dans une nouvelle fenêtre de navigateur et affiche diverses informations sur l'entraînement du modèle, telles que la perte d'entraînement et de validation, le taux d'apprentissage et le nombre de paramètres du modèle.



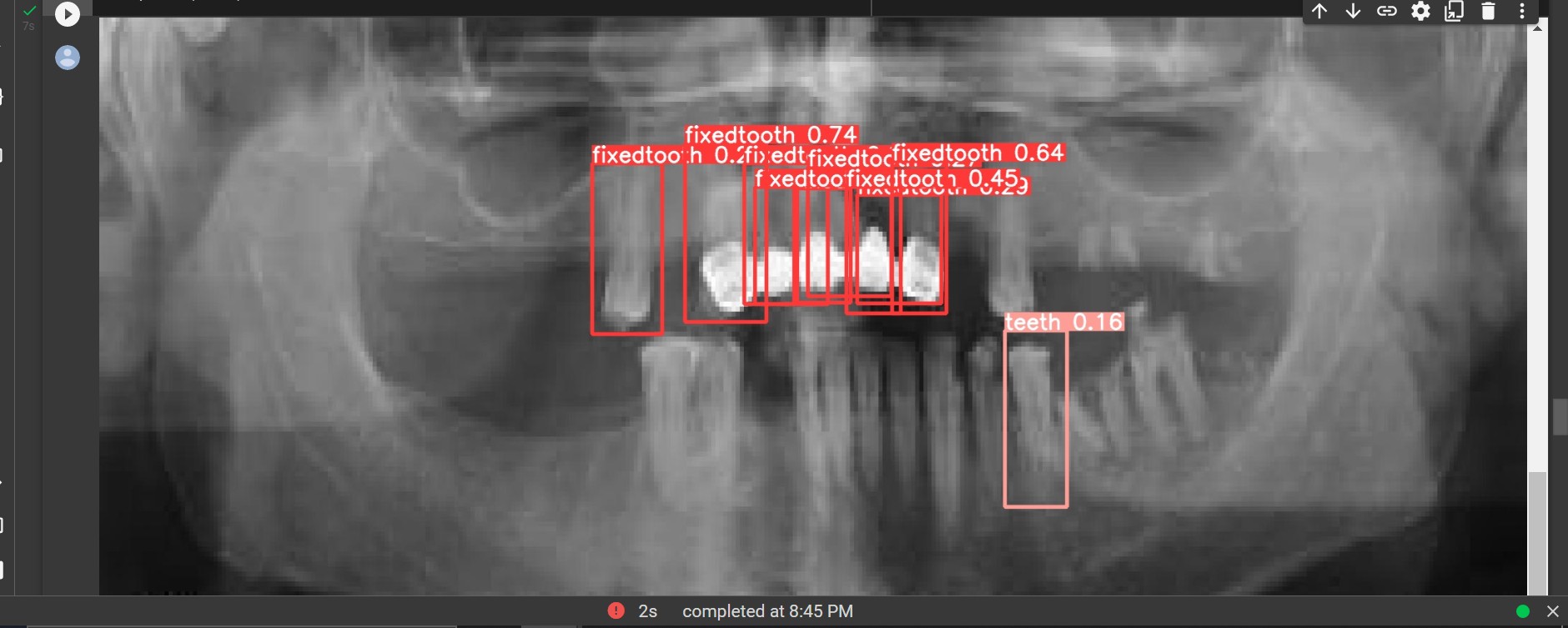
Cette commande exécute le script detect.py, qui applique un modèle YOLOv5 entraîné pour détecter des objets dans les images d'un ensemble de données donné.

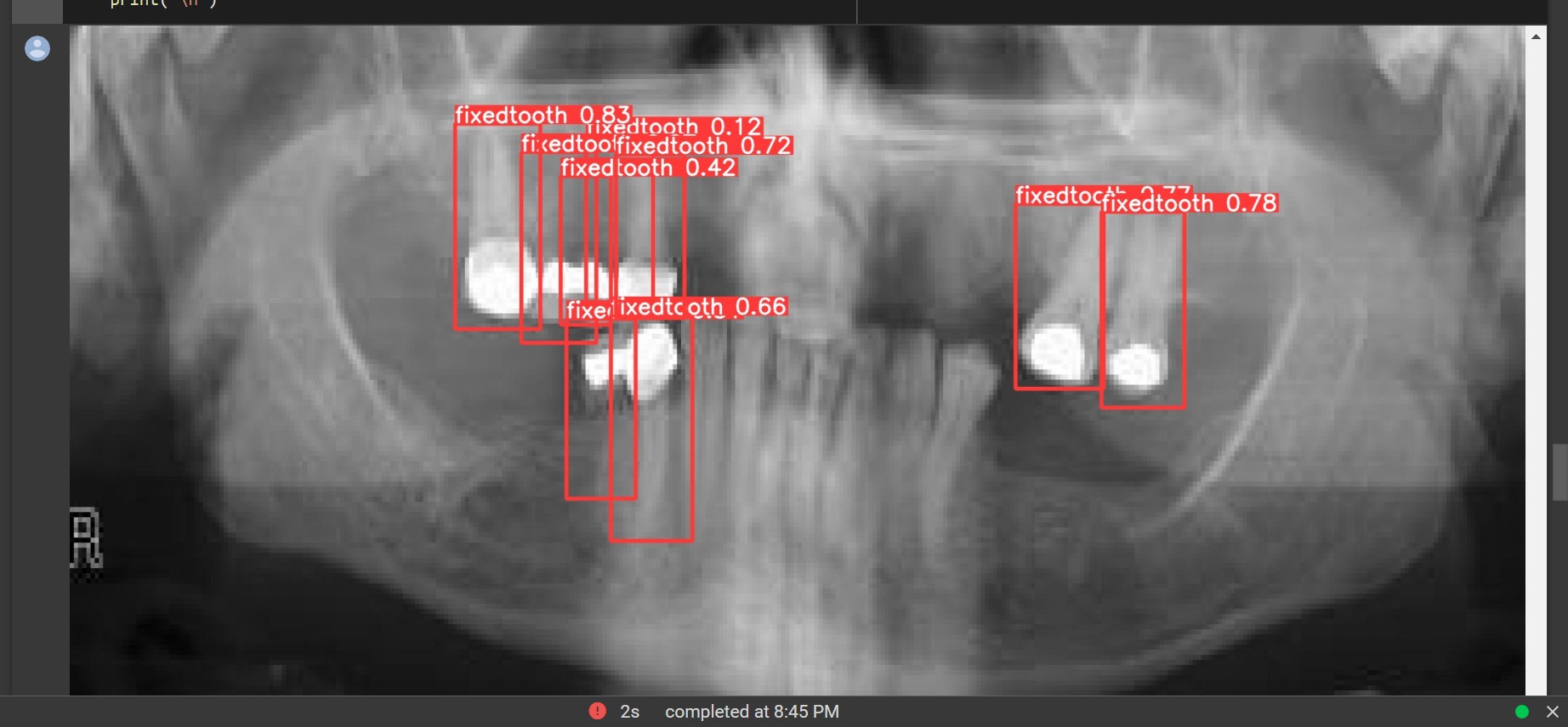
* Le flag **--weights** est utilisé pour spécifier le chemin d'accès au fichier des poids du modèle entraîné. Dans ce cas, les meilleurs poids du processus d'apprentissage sont situés dans le dossier "runs/train/exp/weights" avec le nom "best.pt".
* Le flag **--img** est utilisé pour définir la taille des images d'entrée à 416 pixels.
* Le flag **--conf** est utilisé pour définir le seuil de confiance pour la détection d'objets à 0,1, ce qui signifie que seules les détections avec un niveau de confiance d'au moins 0,1 seront affichées.
* Le flag **--source** est utilisé pour spécifier l'emplacement des images sources, dans ce cas, le dossier "images" dans le dossier "test" de l'emplacement de l'ensemble de données.



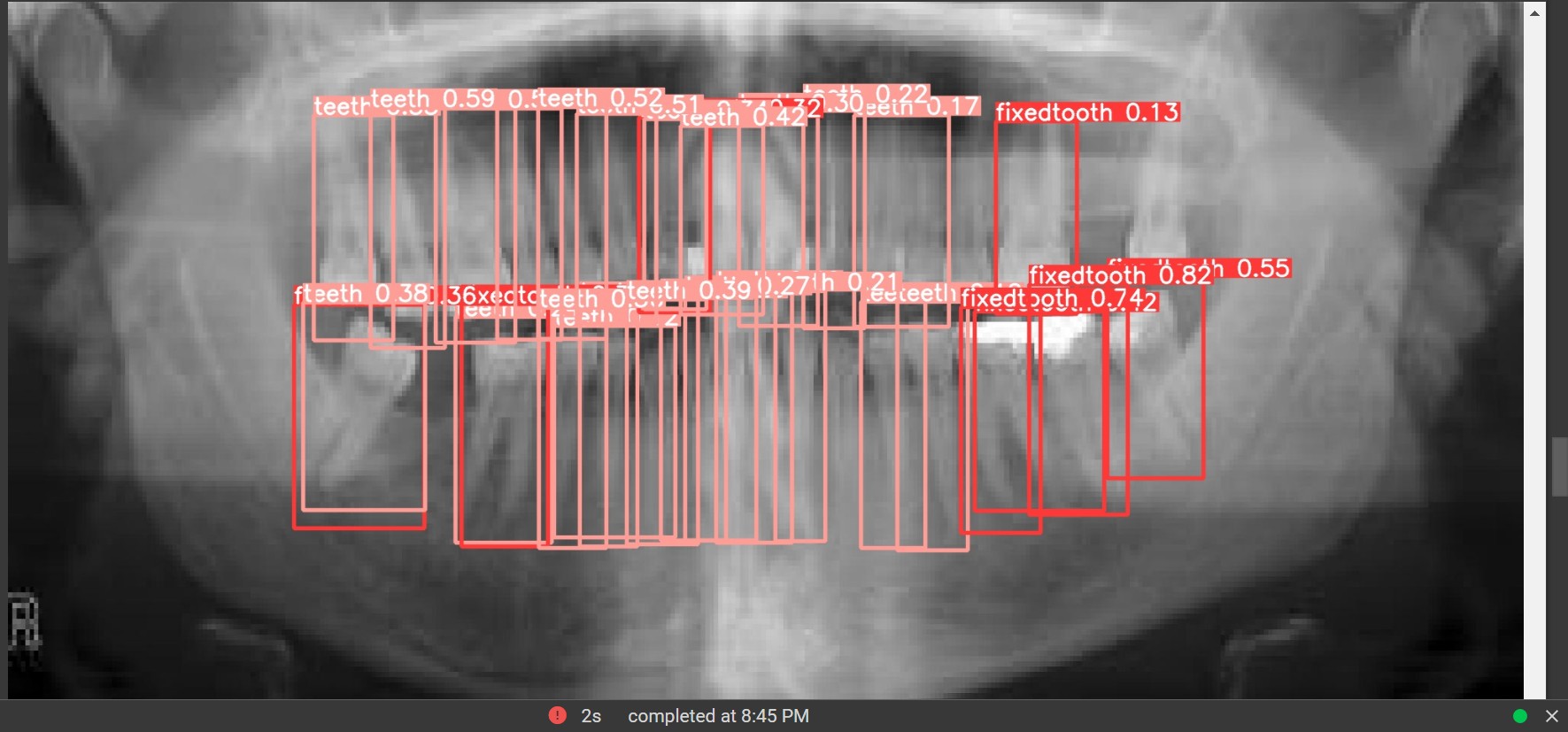
# Les résultats du modèle sur les images de test:

On constate que la dent qui n’est pas corrigée est justement considérée comme corrigée:



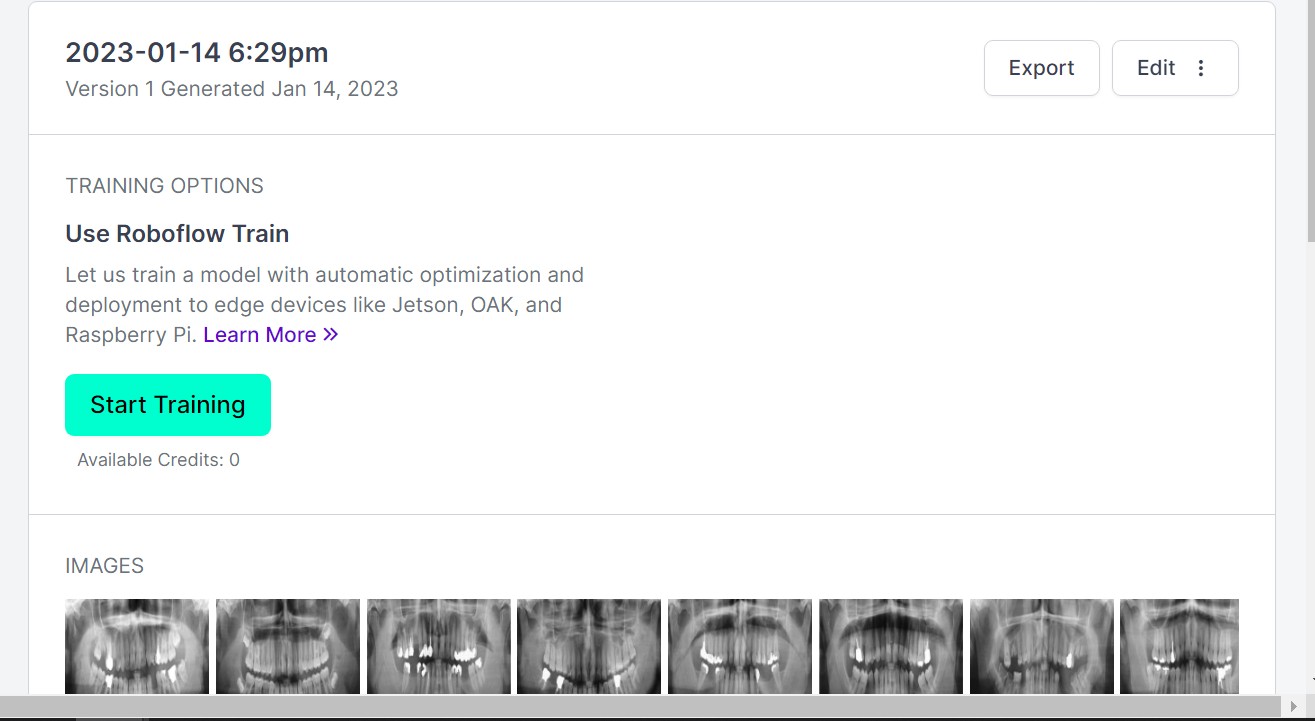


Puisque l’annotation des dents saines n’est faite qu’au niveau de quelques images (Contrainte de temps), l’identification des dents saines ne se fait également qu’au niveau de quelques images comme la suivantes:

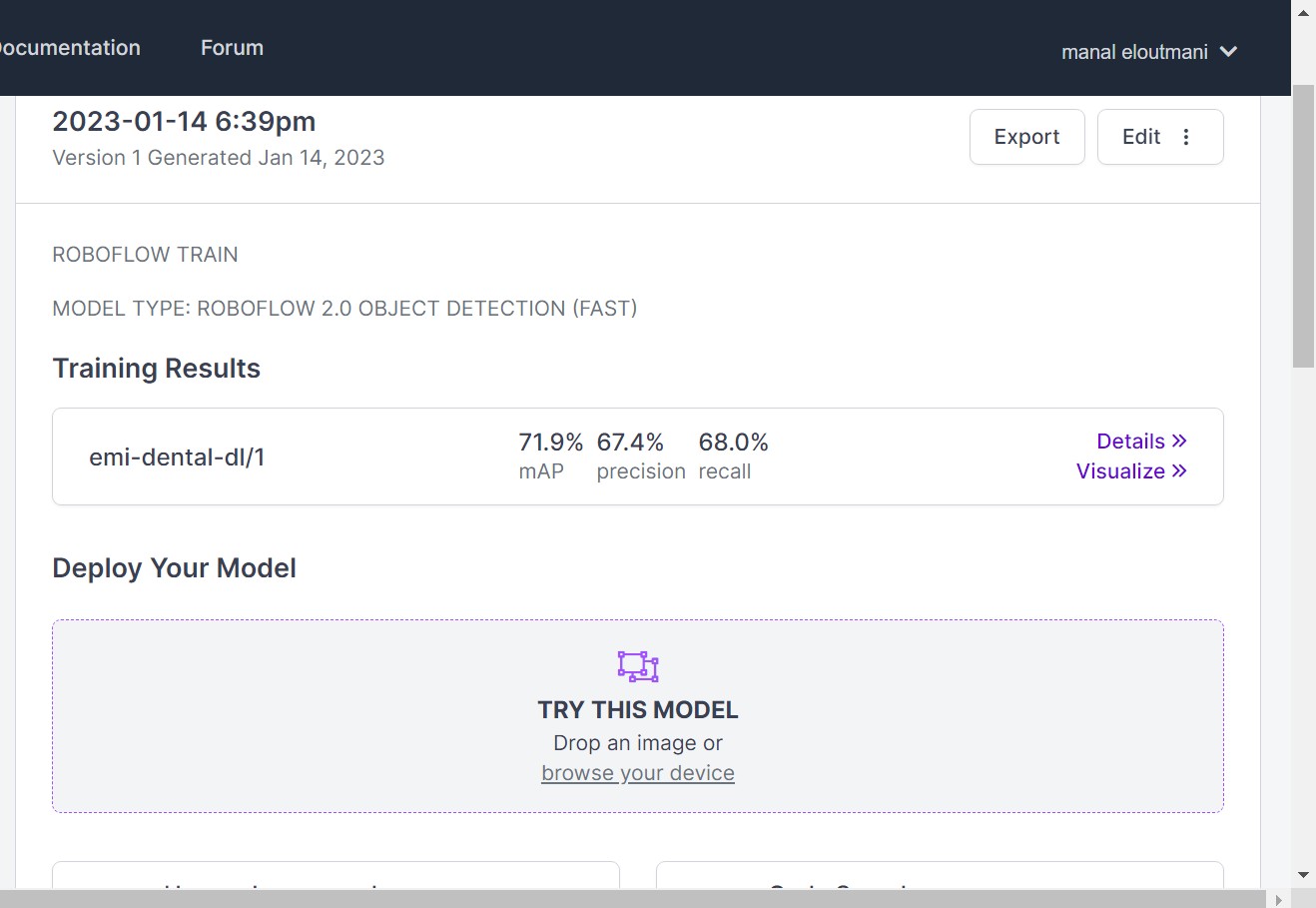


# Test du modèle de Roboflow:

Roboflow nous offre la possibilité d'entraîner un modèle au niveau de la plateforme :

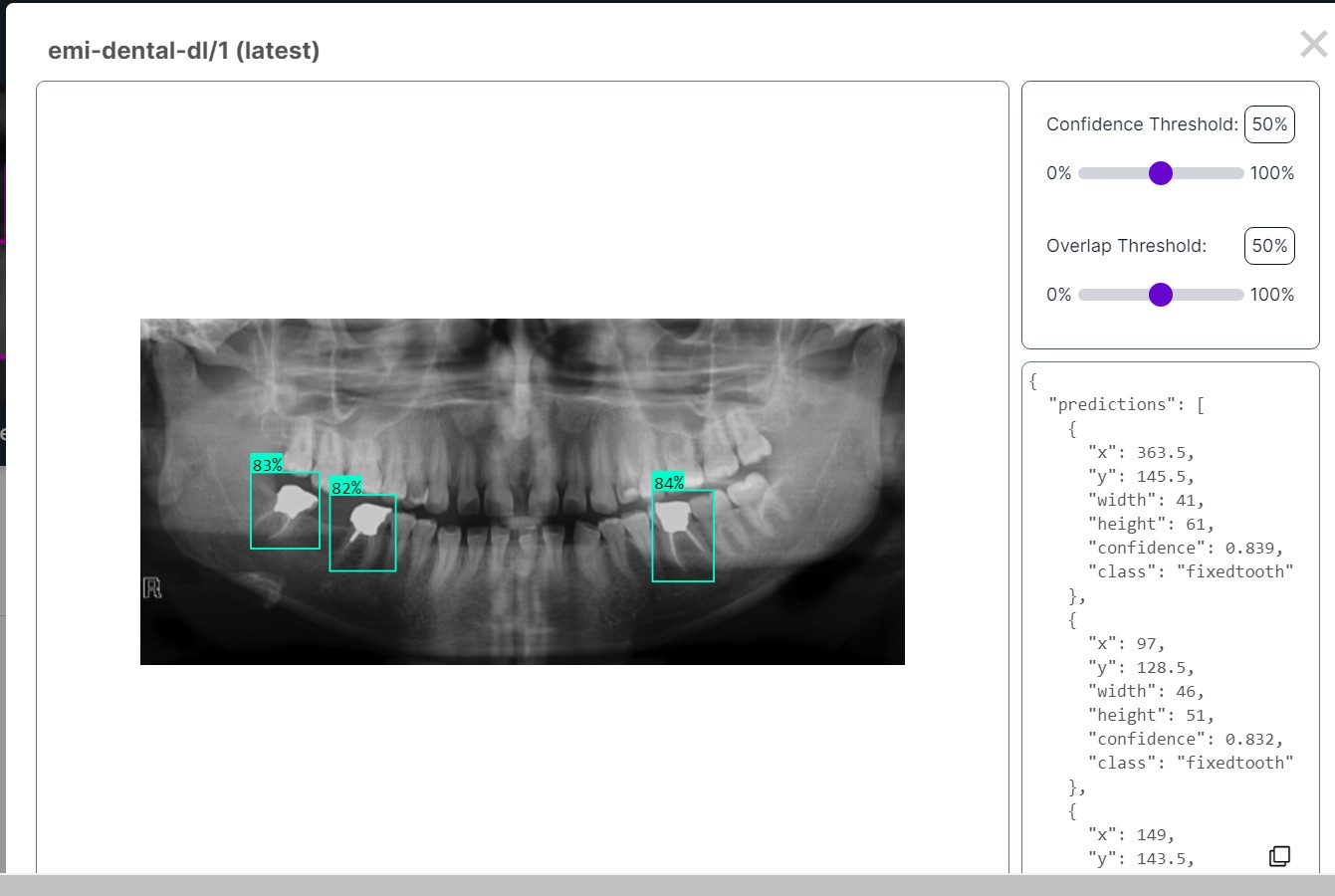


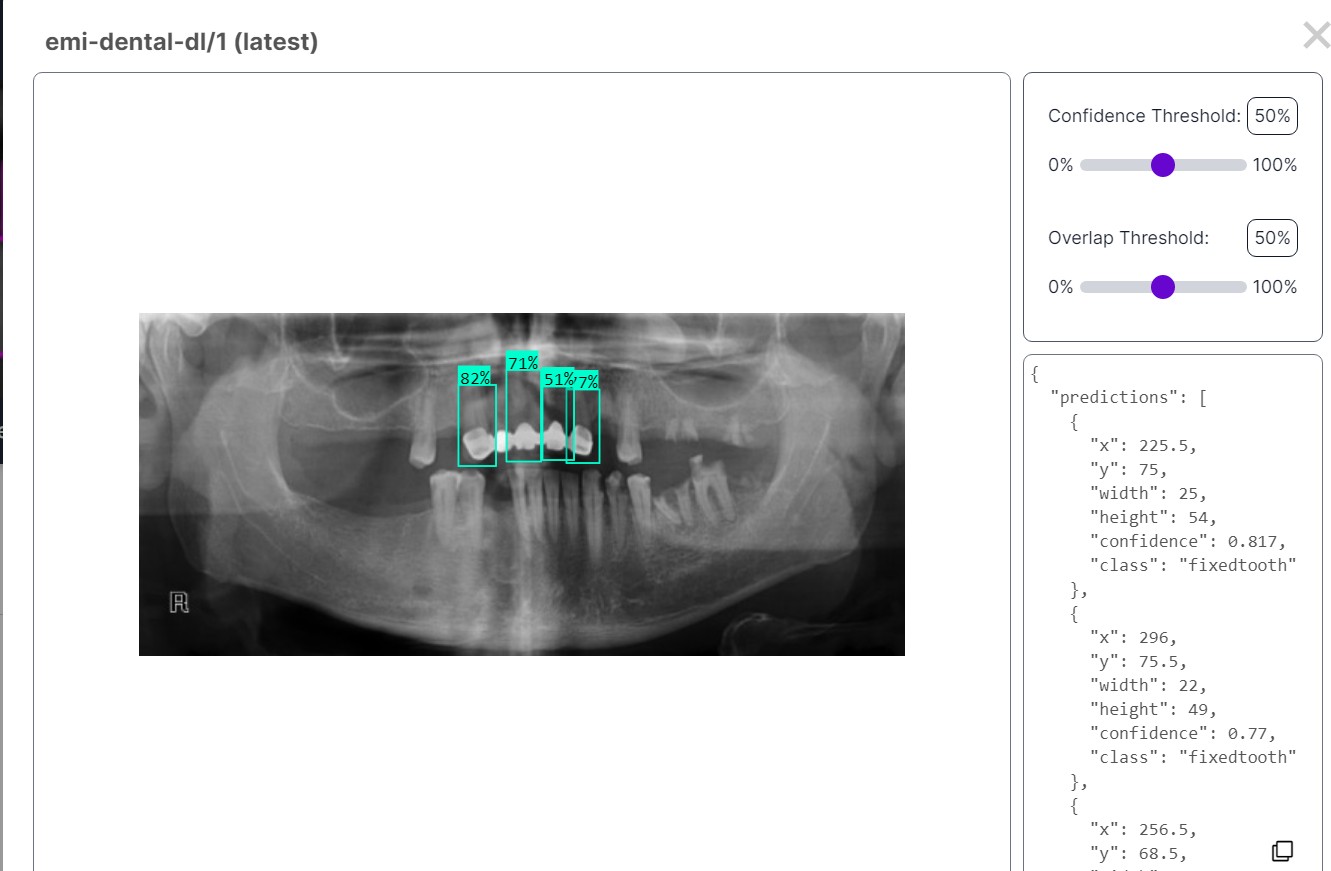
Le modèle final a une précision de 67.4%:



On applique le modèle sur une image de notre choix.

On retrouve ci-dessous le résultat : on constate qu’il identifie les dents corrigées en affichant la précision de cette identification:





# Conclusion:

Pour conclure, le projet de traitement des images de radiographie dentaire a montré qu'il est possible d'utiliser des techniques d'apprentissage automatique pour détecter des anomalies dentaires avec une précision relativement bonne.

Cependant, il est important de noter que le projet nécessite encore des améliorations pour augmenter la précision et la robustesse des résultats, et pour gérer les biais éventuels dans les données d'entraînement.

Dans l'ensemble, ce projet a mis en évidence la puissance de l'apprentissage automatique pour améliorer les diagnostics dentaires et pourrait avoir des implications importantes pour la pratique clinique.