Projet 8
Participez à une compétition Kaggle

Consignes

Consignes :

Vous devrez participer à la compétition dans la mesure du possible, ou en tout cas obtenir des résultats mesurables.
Vous passerez par toutes les étapes de l'analyse : récupération, nettoyage des données, analyse exploratoire,création de plusieurs modèles et mesure de leurs performances, etc.

Github du projet : https://github.com/ismailazdad/uwmgit

- Enrichir les réalisations d'autres membres de la communauté de professionnels
- Rédiger une note méthodologique afin de communiquer sa démarche de modélisation
- Utiliser un logiciel de version de code pour assurer l’intégration du modèle
- Présenter son code aux standards PEP 8

Kernels kaggle

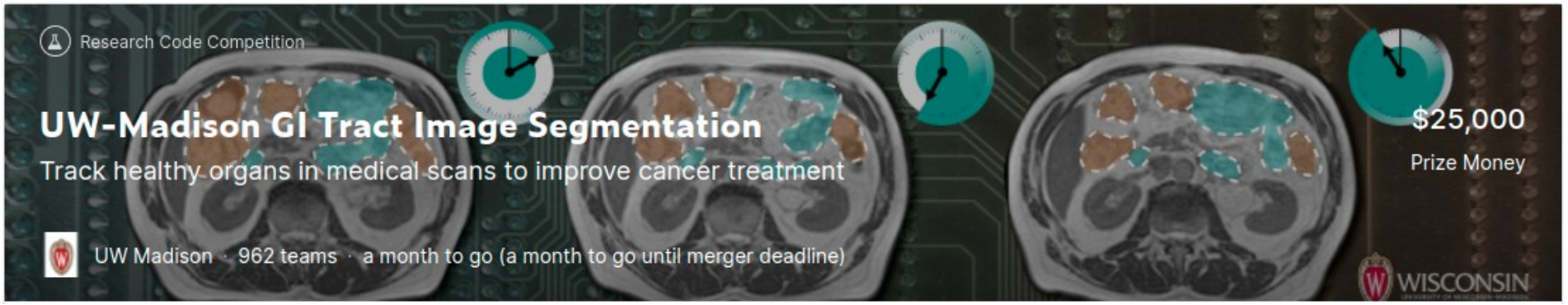
Exploration et analyse : lien

Entraînement du modele : lien

Soumission : lien

Github : lien

Compétition choisi



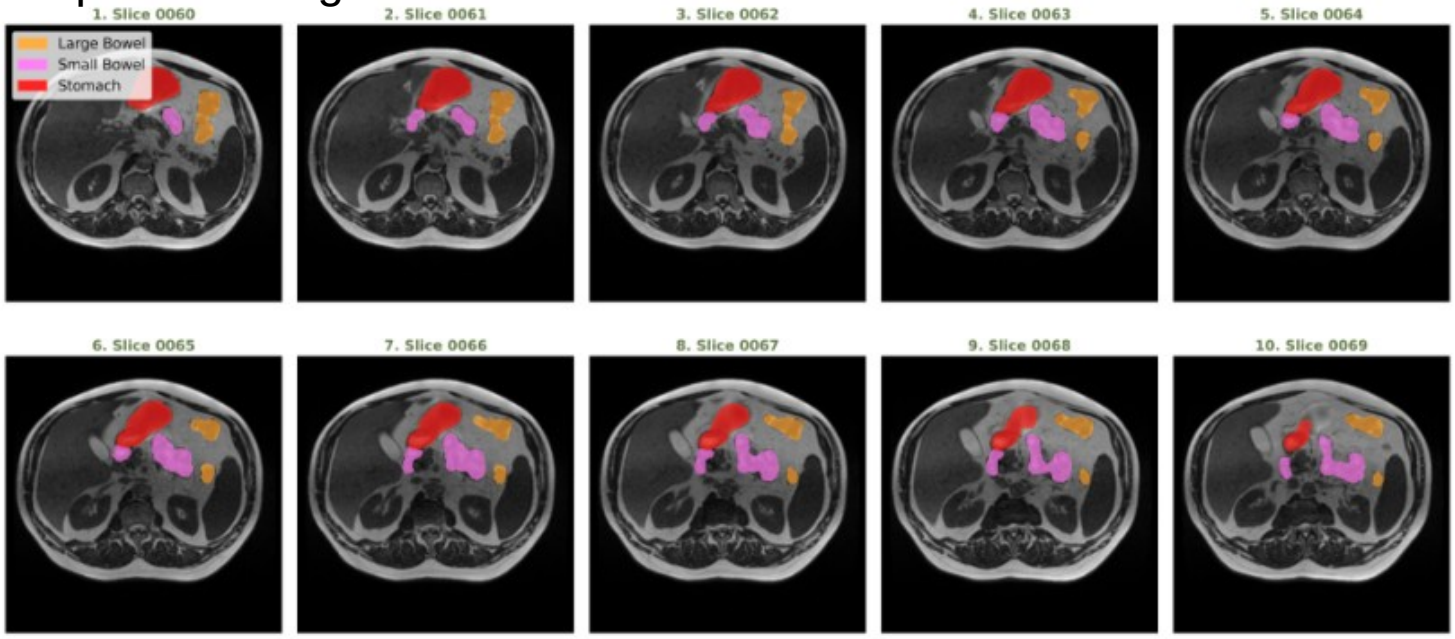
Segmentation sémantique des d images :
Aider les personnes souffrant de cancer du tractus gastro intestinal lors du scanner

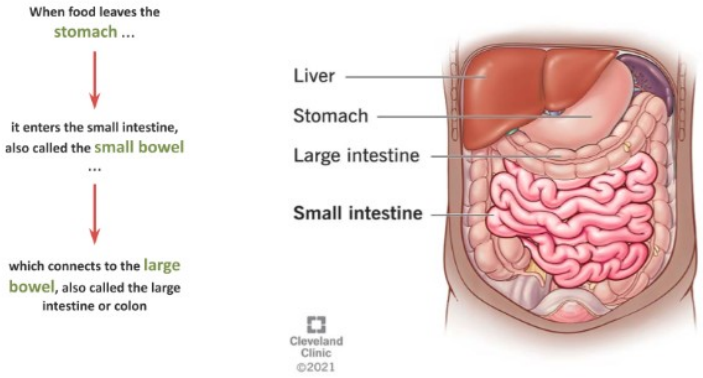
Généralement administrée en 10 à 15 minutes par jour pendant 1 à 6 semaines.Les radio-oncologues tentent de délivrer de fortes doses de rayonnement à l'aide de faisceaux de rayons X dirigés vers les tumeurs tout en évitant l'estomac et les intestins. Grâce à une technologie plus récente telle que l'imagerie par résonance magnétique intégrée et les systèmes d'accélérateur linéaire, également connus sous le nom de MR-Linacs, les oncologues sont en mesure de visualiser la position quotidienne de la tumeur et des intestins, qui peut varier d'un jour à l'autre. Dans ces scans, les radio-oncologues doivent tracer manuellement la position de l'estomac et des intestins afin d'ajuster la direction des faisceaux de rayons X pour augmenter la dose administrée à la tumeur et éviter l'estomac et les intestins.

Il s'agit d'un processus long et laborieux qui peut prolonger les traitements de 15 minutes par jour à une heure par jour, ce qui peut être difficile à tolérer pour les patients, à moins que l'apprentissage en profondeur ne puisse aider à automatiser le processus de segmentation. Une méthode pour segmenter l'estomac et les intestins rendrait les traitements beaucoup plus rapides et permettrait à un plus grand nombre de patients d'obtenir un traitement plus efficace.

But

Repérer les organes a éviter dans l IRM



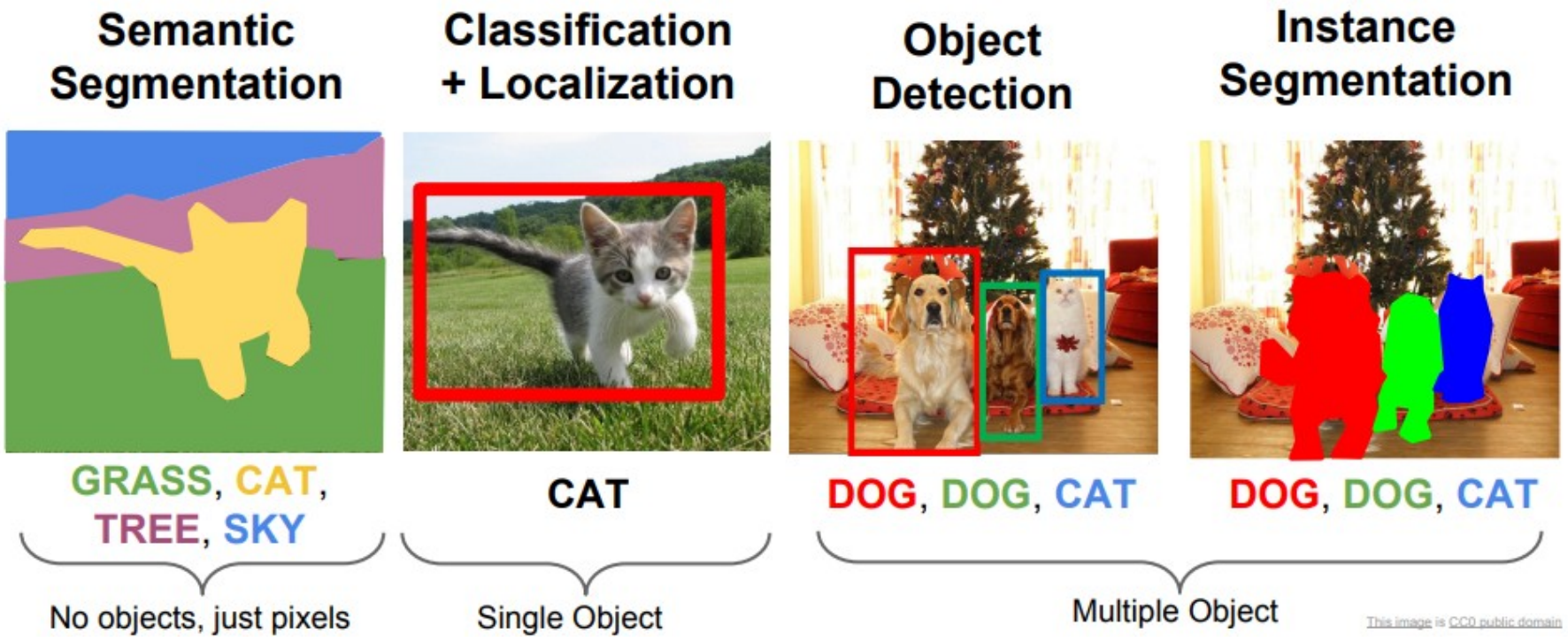


Qu'est-ce qu'une IRM ?

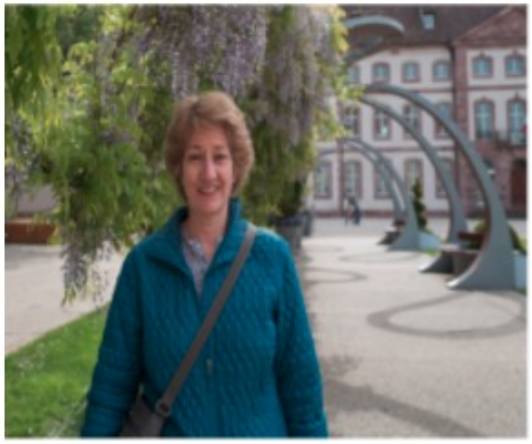
L'imagerie par résonance magnétique (IRM) est un type d'examen qui utilise des champs magnétiques puissants
et des ondes radio pour produire des images détaillées de l'intérieur du corps.

Un scanner IRM est un grand tube qui contient des aimants puissants.
Vous vous allongez à l'intérieur du tube pendant le scan.

Instance segmentation



Détection des pixels par classes

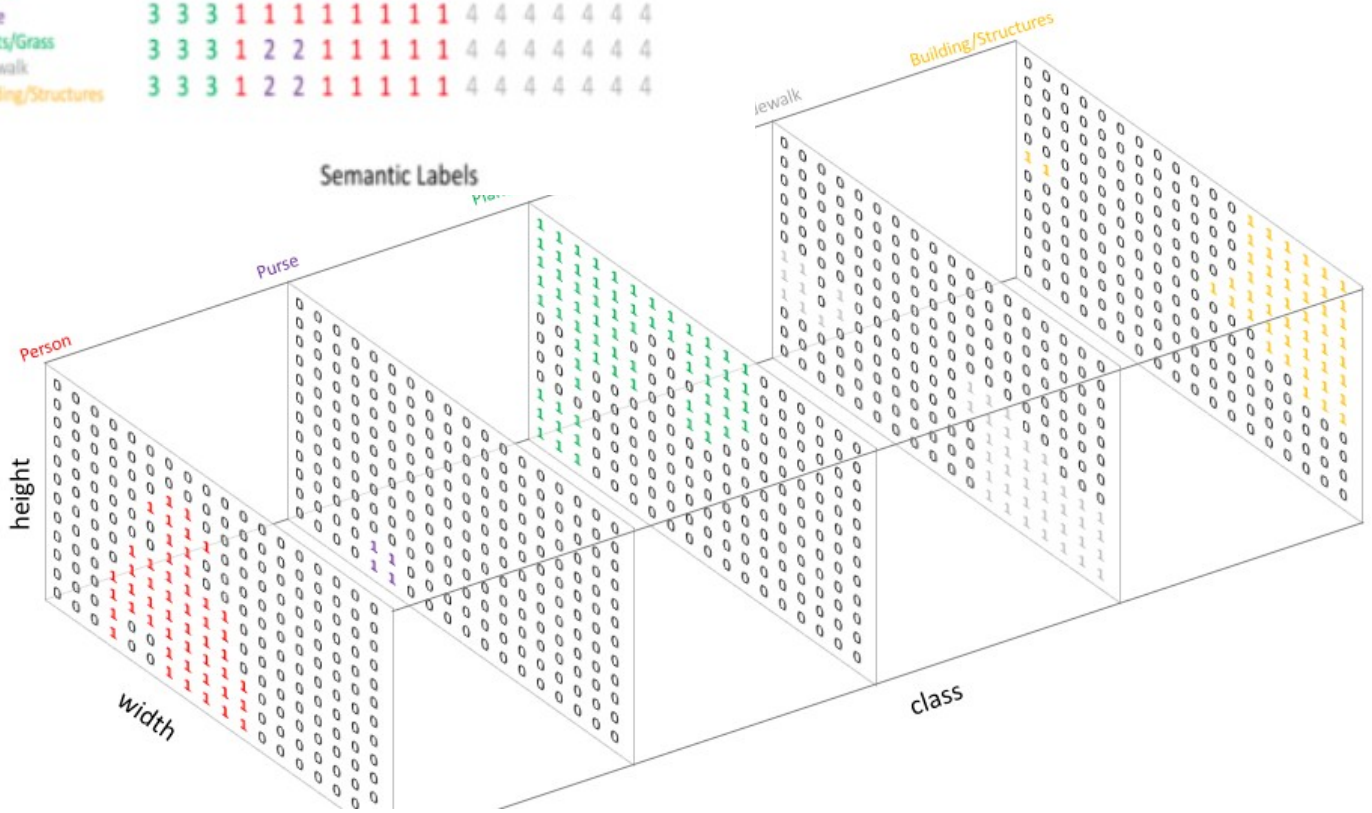


Input

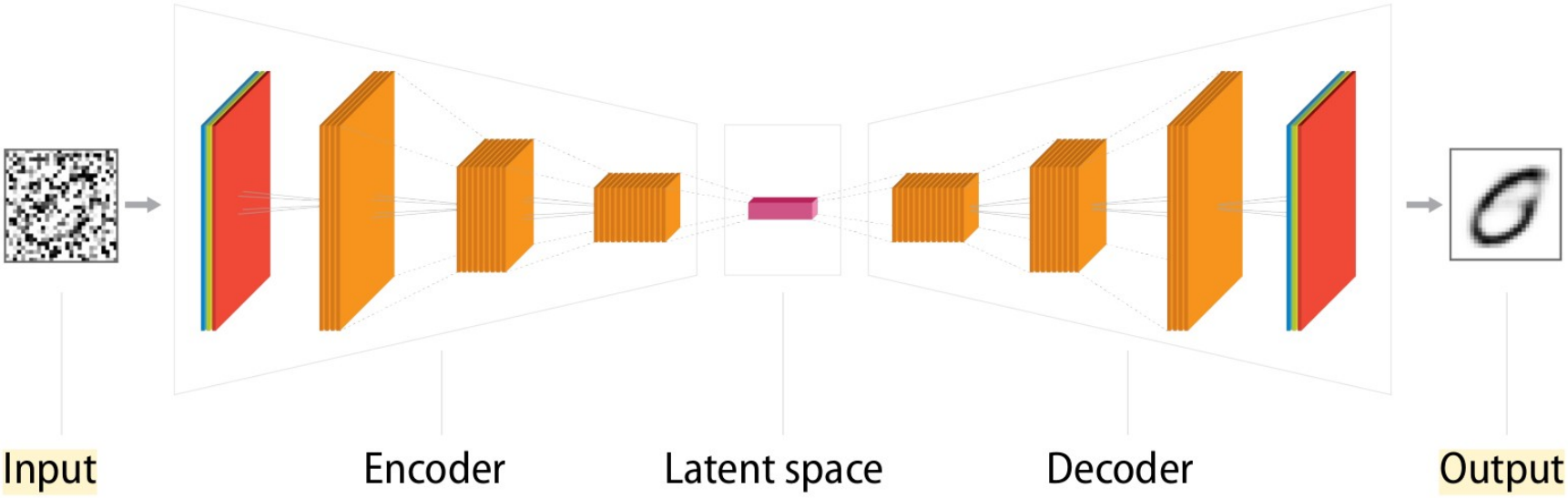
segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

Semantic Labels



Auto encoder

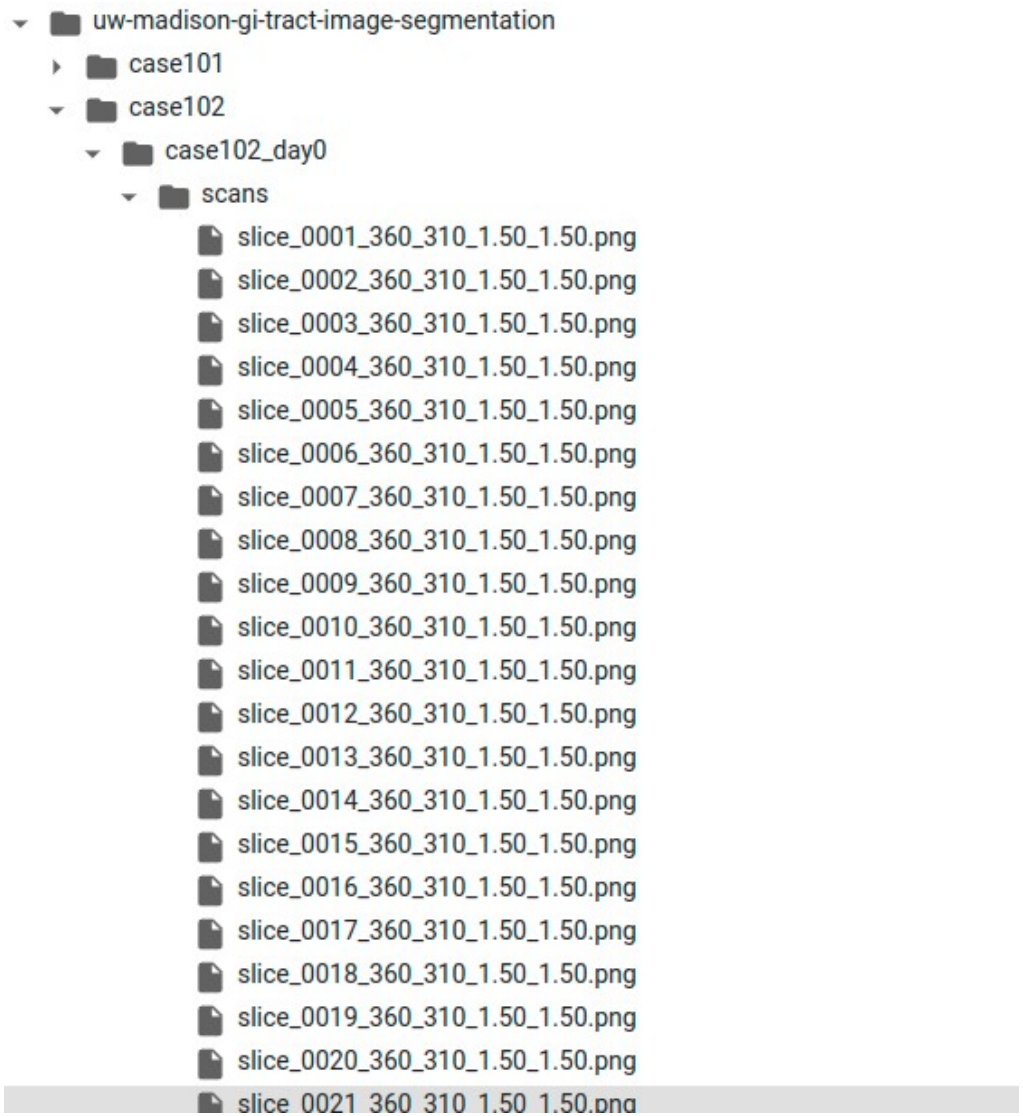


le dataset exploration

Segmentation : renseigne les coordonnes RLE de nos organes
Id : identifiant du cas ce décomposant par convention de nommage CAS_JOUR_NUM
Class : class d organe correspondant aux code RLE
case_id : numéro du cas
day_id : jours
slice_id : numéro de découpe de l image
Path : chemin de l image
Filename : nom de l image
Unique_filename : création d un identifiant unique
height : hauteur de l image
Width : largeur de l image
Size : combo WxH

id	class	segmentation	case_id	day_id	slice_id	path
case123_day20_slice_0065	stomach	28094 3 28358 7 28623 9 28889 9 29155 9 29421 ...	123	20	0065	/content/uw-madison-gi-tract-image-segmentatio...
case123_day20_slice_0066	stomach	27561 8 27825 11 28090 13 28355 14 28620 15 28...	123	20	0066	/content/uw-madison-gi-tract-image-segmentatio...
case123_day20_slice_0067	stomach	15323 4 15587 8 15852 10 16117 11 16383 12 166...	123	20	0067	/content/uw-madison-gi-tract-image-segmentatio...
case123_day20_slice_0068	stomach	14792 5 15056 9 15321 11 15587 11 15852 13 161...	123	20	0068	/content/uw-madison-gi-tract-image-segmentatio...
case123_day20_slice_0069	stomach	14526 6 14789 12 15054 14 15319 16 15584 17 15...	123	20	0069	/content/uw-madison-gi-tract-image-segmentatio...

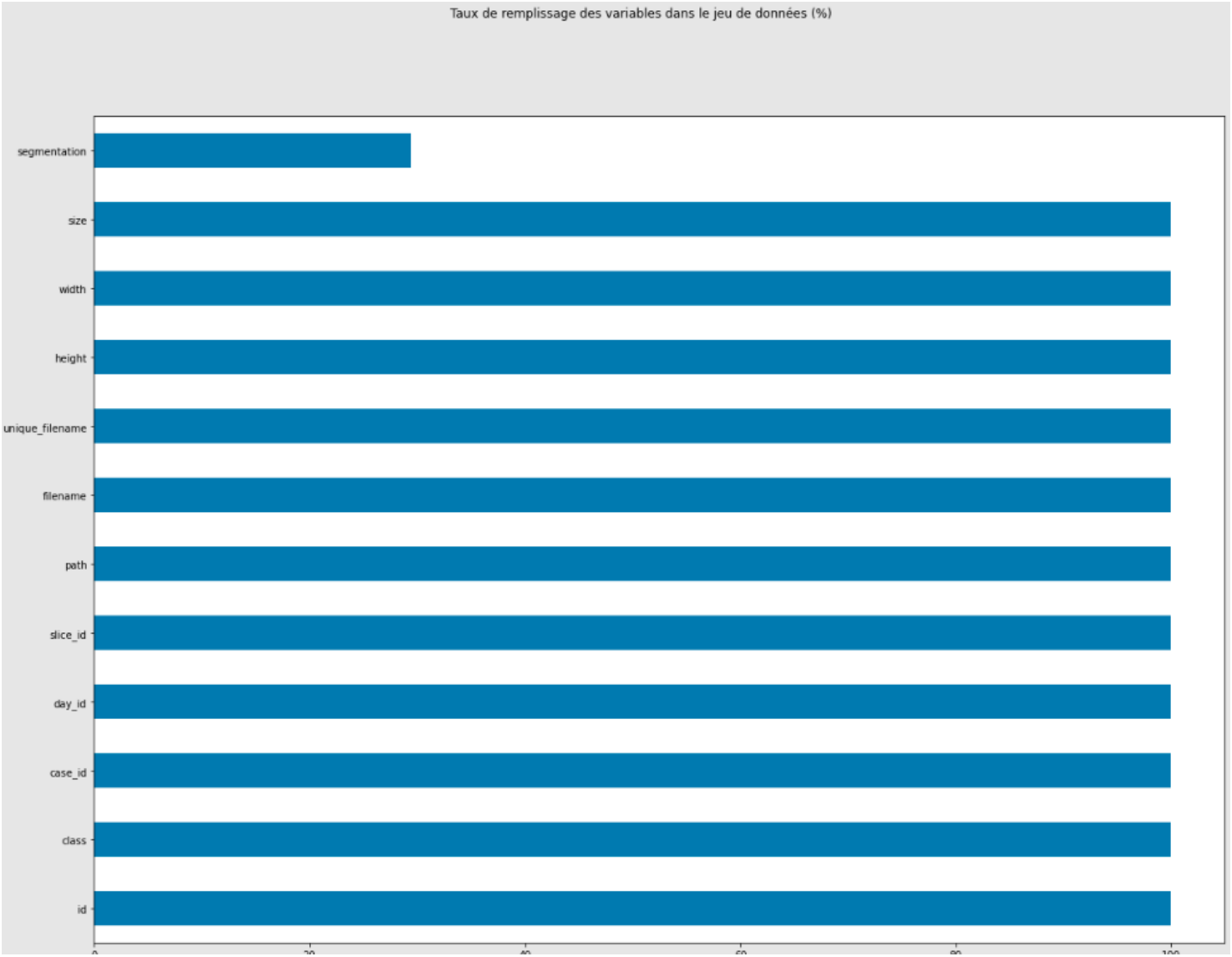
Répartition du dataset



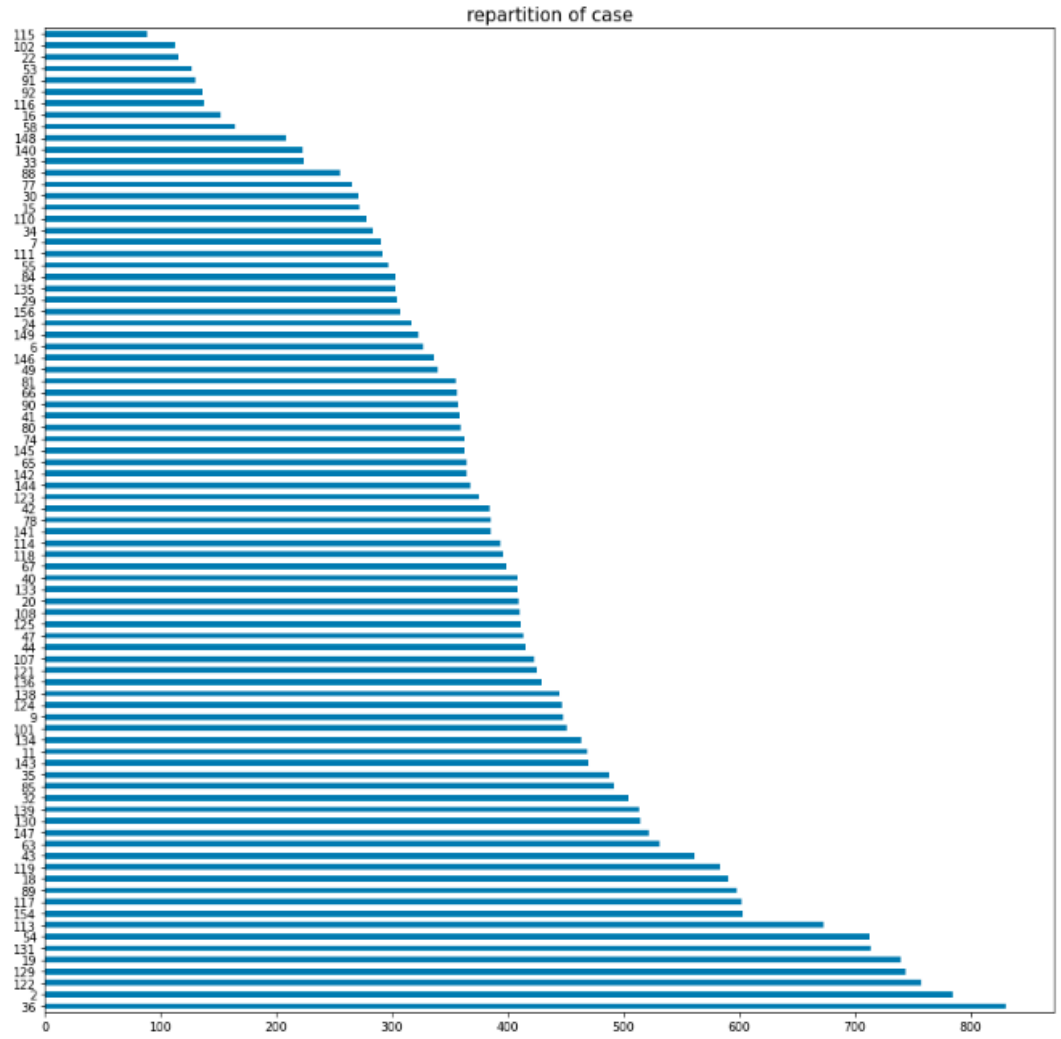
Le data Processing à nécessite de s adapte au contrainte du dataset pour la compétition

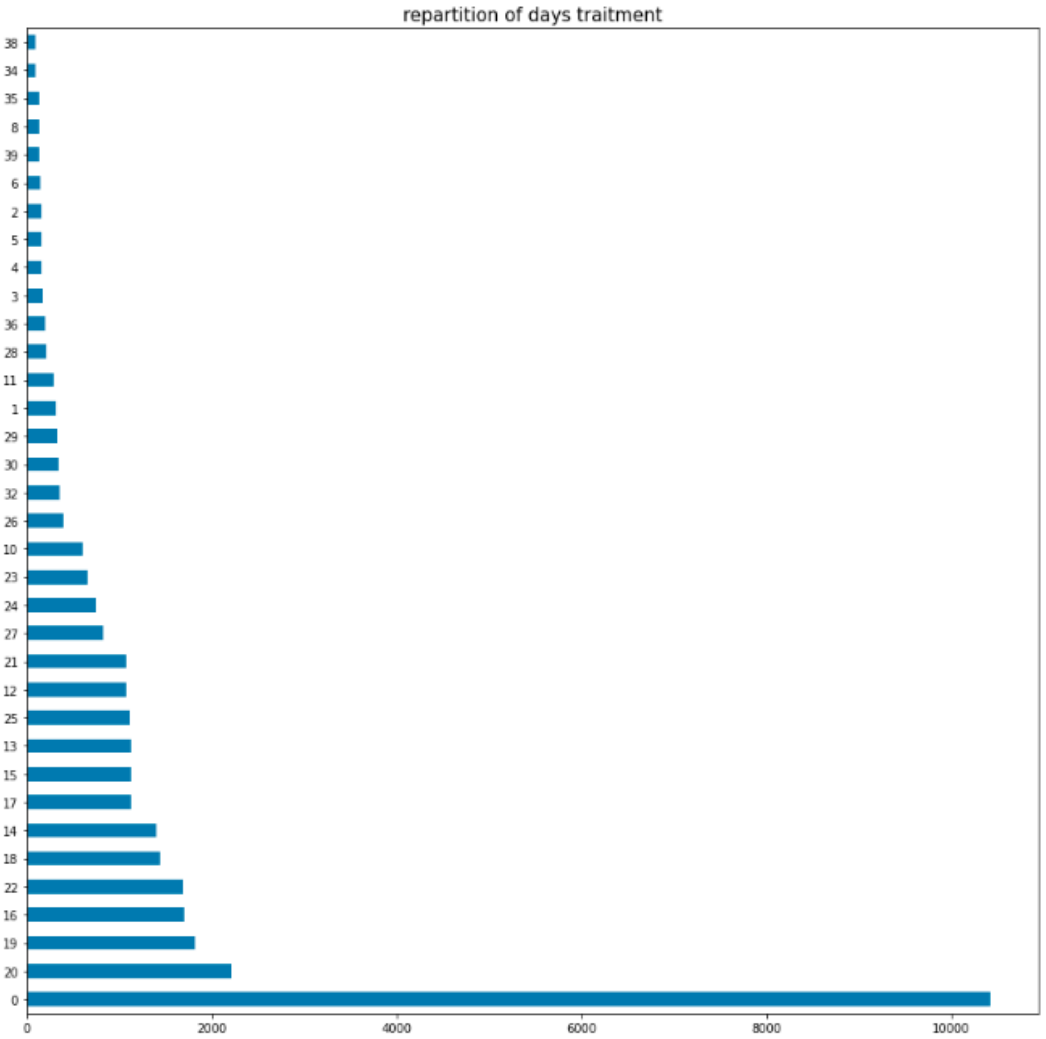
En effet nous devrons gérer les chemins des images, qui ne sont pas les mêmes suivants les patients

Analyse exploratoire

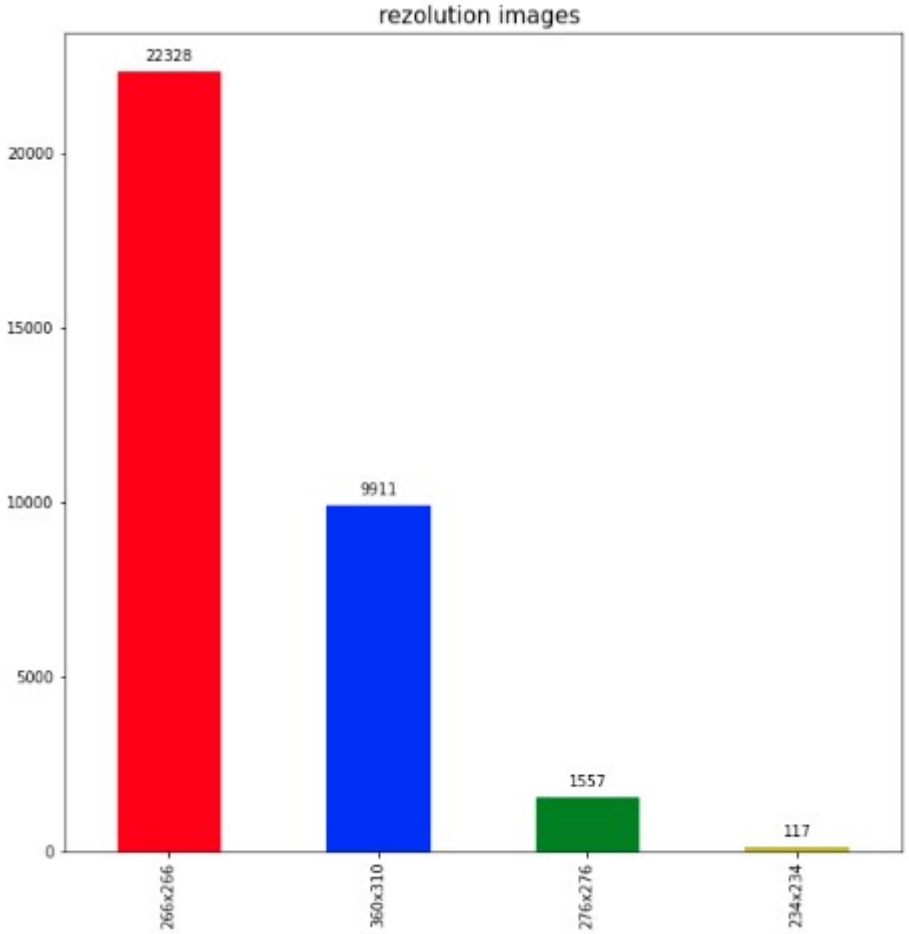


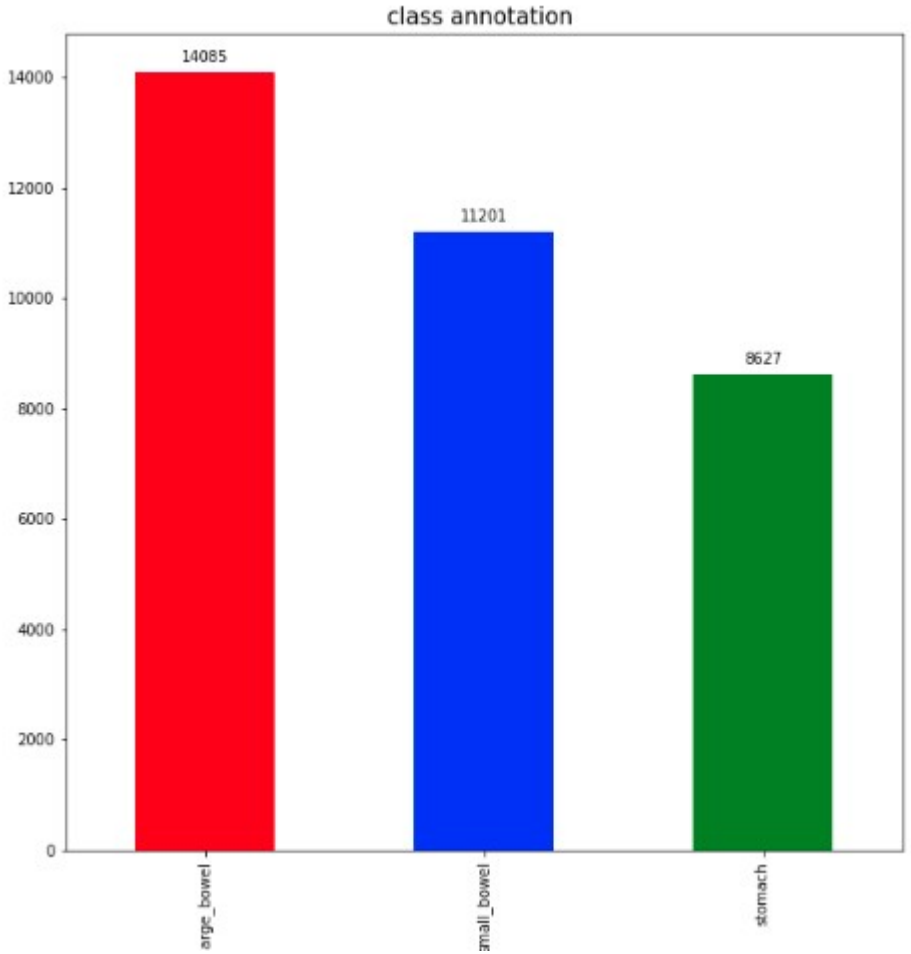
Analyse exploratoire





Analyse exploratoire





Contribution coco format

Pre processing management

L'ensemble de données Common Objects in Context (COCO) de Microsoft est l'ensemble de données de détection d'objets le plus populaire à l'heure actuelle. Il est largement utilisé pour comparer les performances des méthodes de vision par ordinateur.

En raison de la popularité de l'ensemble de données, le format utilisé par COCO pour stocker les annotations est souvent le format de référence
lors de la création d'un nouvel ensemble de données de détection d'objet personnalisé.

Merci a mohandass(
https://www.kaggle.com/code/mohanrobotics/cocoformat-dataset-creation-instance-segmentation)

Qui a commencer un travail de conversion du dataset , vers le format COCO, le code développer a du être modifier pour fonctionner correctement, afin de respecter le standard coco et fonctionner avec la librairie cocoapi de microsoft.
Une fois les fichiers coco json prêt, j ai ajouter une contribution a la communauté en développant un generateur de dataset qui s appuie sur ce fichier json uniquement et prêt pour l entrainement
Cette classe est agnostique , elle peut théoriquement être utiliser dans n importe quel framework , comme pytorch par exemple

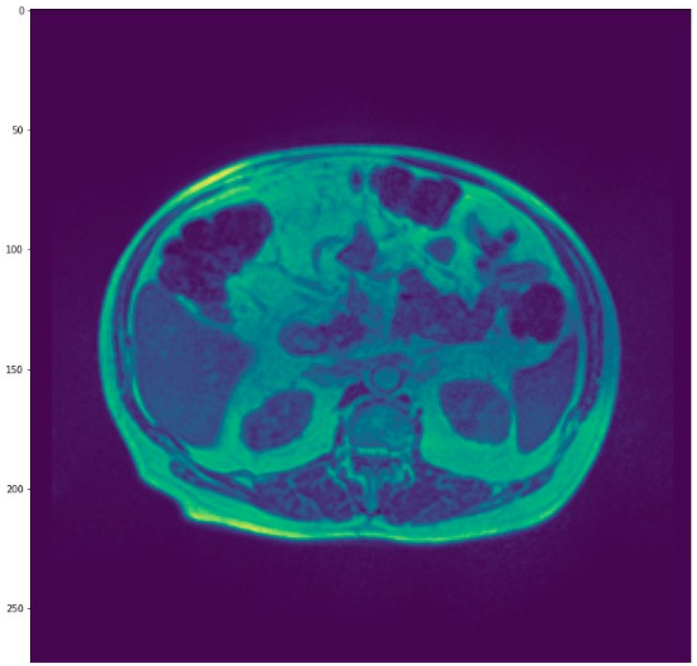
Génération du dataset a partir du fichier

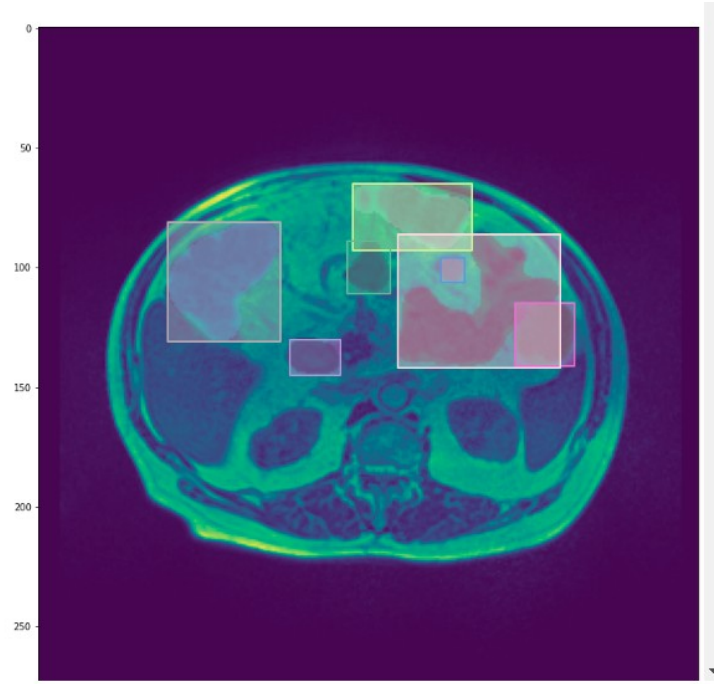
```python
class DataGeneratorFromCocoJson(tf.keras.utils.Sequence):
  # function getting info dataset from json coco
  # Batch size
  # subset train or test for annotations
  # image_list to develop...
  # classes classe wanted
  # input image size tuple (X,X)
  # annFile path to annoted coco json file file
    def __init__(self, batch_size = batch_size, subset="train",image_list=[]
                 ,classes=[], input_image_size=(128,128),annFile='',shuffle=False):

        super().__init__()
        self.subset = subset
        self.batch_size = batch_size
        self.indexes = np.arange(len(image_list))
        self.image_list= image_list
        self.classes= classes
        self.input_image_size= (input_image_size)
        self.dataset_size = len(image_list)
        self.coco = COCO(annFile)
        catIds = self.coco.getCatIds(catNms=self.classes)
        self.catIds = catIds
        self.cats = self.coco.loadCats(catIDs)
        self.imgIds = self.coco.getImgIds()
        self.shuffle = shuffle
        self.on_epoch_end()


    def __len__(self):
      return int(len(self.image_list)/self.batch_size)

    def on_epoch_end(self):
        if self.shuffle == True:
            np.random.shuffle(self.indexes)


    def getClassName(self,classID, cats):
        for i in range(len(cats)):
            if cats[i]['id']==classID:
                return cats[i]['name']
        return None

    def getNormalMask(self,image_id,catIds):
        annIds = self.coco.getAnnIds(image_id, catIds=catIds, iscrowd=None)
        anns = self.coco.loadAnns(annIds)
        cats = self.coco.loadCats(catIds)
        train_mask = np.zeros(self.input_image_size,dtype=np.uint8)
        for a in range(len(anns)):
            className = self.getClassName(anns[a]['category_id'], cats)
            pixel_value = self.classes.index(className)+1
            new_mask = cv2.resize(self.coco.annToMask(anns[a])*pixel_value, self.input_image_s
            train_mask = np.maximum(new_mask, train_mask)
            # train_mask = new_mask / 255.0
        return train_mask
```

```python
    def getLevelsMask(self,image_id):
      #for each category , we get the x mask and add it to mask list
      res = []
      mask = np.zeros((self.input_image_size))
      for j,categorie in enumerate(self.catIds):
        annIds = coco.getAnnIds(image_id, catIds=categorie, iscrowd=None)
        anns = coco.loadAnns(annIds)
        mask = self.getNormalMask(image_id,categorie)
        res.append( mask)
      return res

    def getImage(self,file_path):
        train_img = cv2.imread(file_path, cv2.IMREAD_ANYDEPTH)
        train_img = cv2.resize(train_img, (self.input_image_size))
        train_img = train_img.astype(np.float32) / 255.
        if (len(train_img.shape)==3 and train_img.shape[2]==3):
            return train_img
        else:
            stacked_img = np.stack((train_img,)*3, axis=-1)
            return stacked_img

    def get_image_Infos_by_path_id(self, node):
        for dict in self.image_list:
            if dict['file_name'] == node:
                return dict


    def __getitem__(self, index):
        X = np.empty((self.batch_size,128,128,3))
        y = np.empty((self.batch_size,128,128,3))
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        for i in range(len(indexes)):
          value = indexes[i]
          img_info = self.image_list[value]
          w = img_info['height']
          h = img_info['width']
          X[i,] = self.getImage(img_info['file_name'])
          mask_train = self.getLevelsMask(img_info['id'])
          for j in self.catIds:
            y[i,:,:,j] = mask_train[j]
            y[i,:,:,j] = mask_train[j]
            y[i,:,:,j] = mask_train[j]

        X = np.array(X)
        y = np.array(y)

        if self.subset == 'train':
            return X, y
        else:
            return X
```

Aperçus des annotations coco

Grace a la librairie pycotools , nous pouvons avoir un aperçus des annotations plus facilement

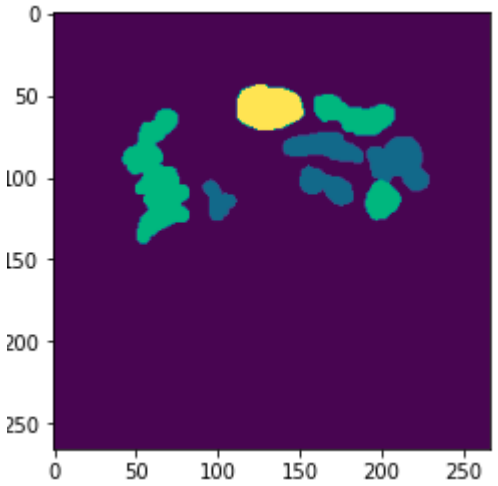




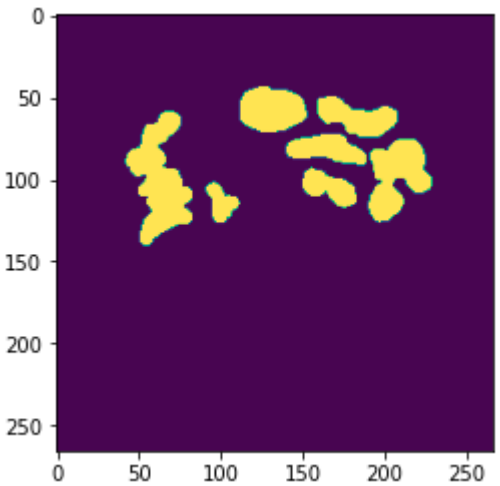
Travail sur les masks

Normalisation des images, et gestion des couches des masques

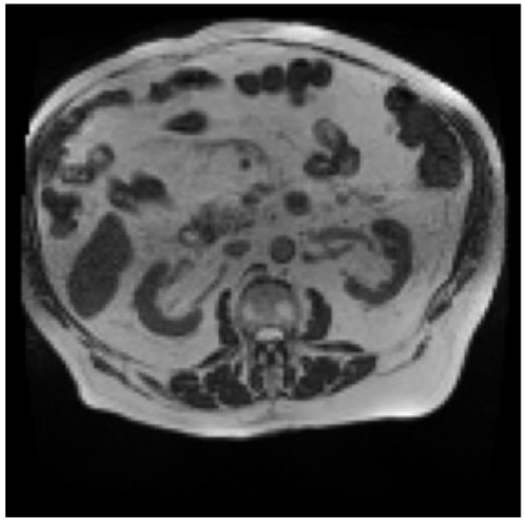
Masque multiple

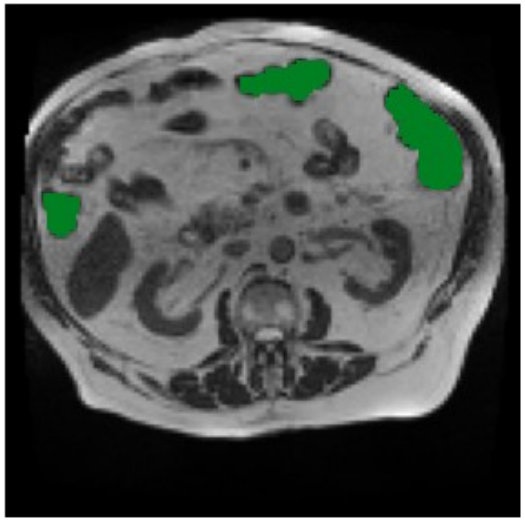


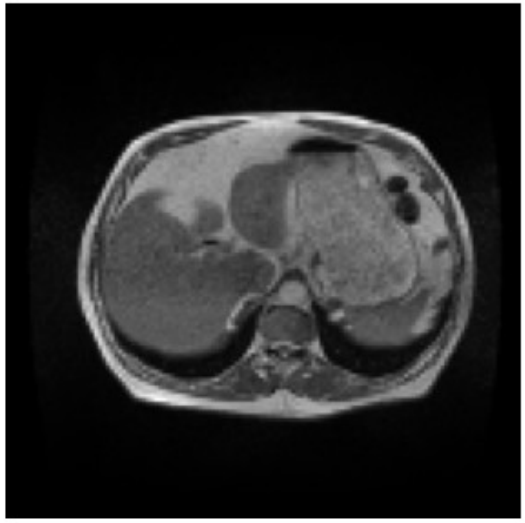
Masque binaire

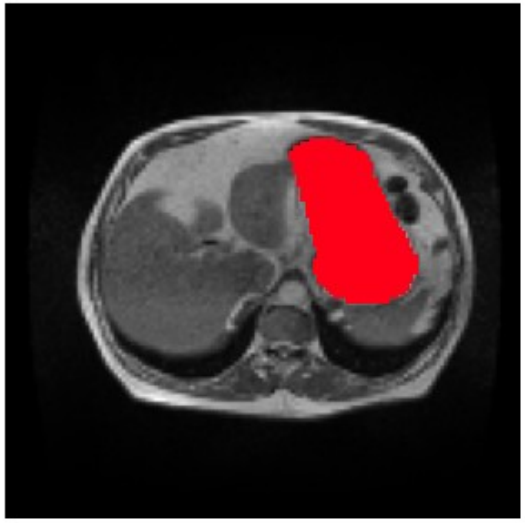


Test des annotations des fichiers









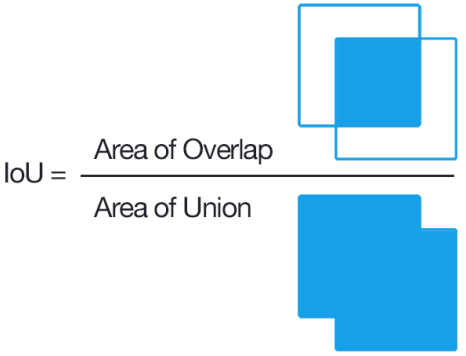
Récupération des informations de chaque masque, du fichier coco
Et vérification des annotations avant l'entraînement

Metriques

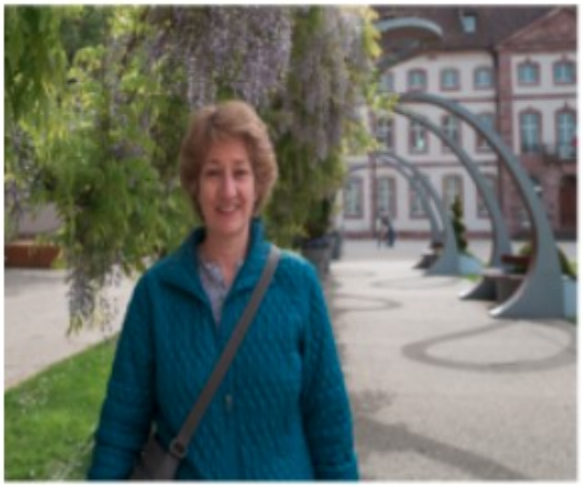
Blue is predicted bounding box and red is ground truth bounding box

Intersection

Union

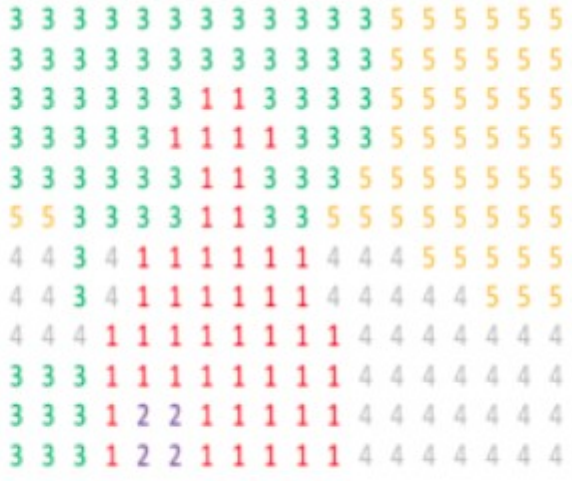


coefficient de Dice : ce coefficient mesure la similarité entre deux ensembles de données. Cet index est sans doute devenu l'outil le plus largement utilisé dans la validation des algorithmes de segmentation d'images



Input

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures



Semantic Labels

Detectron2 architecture

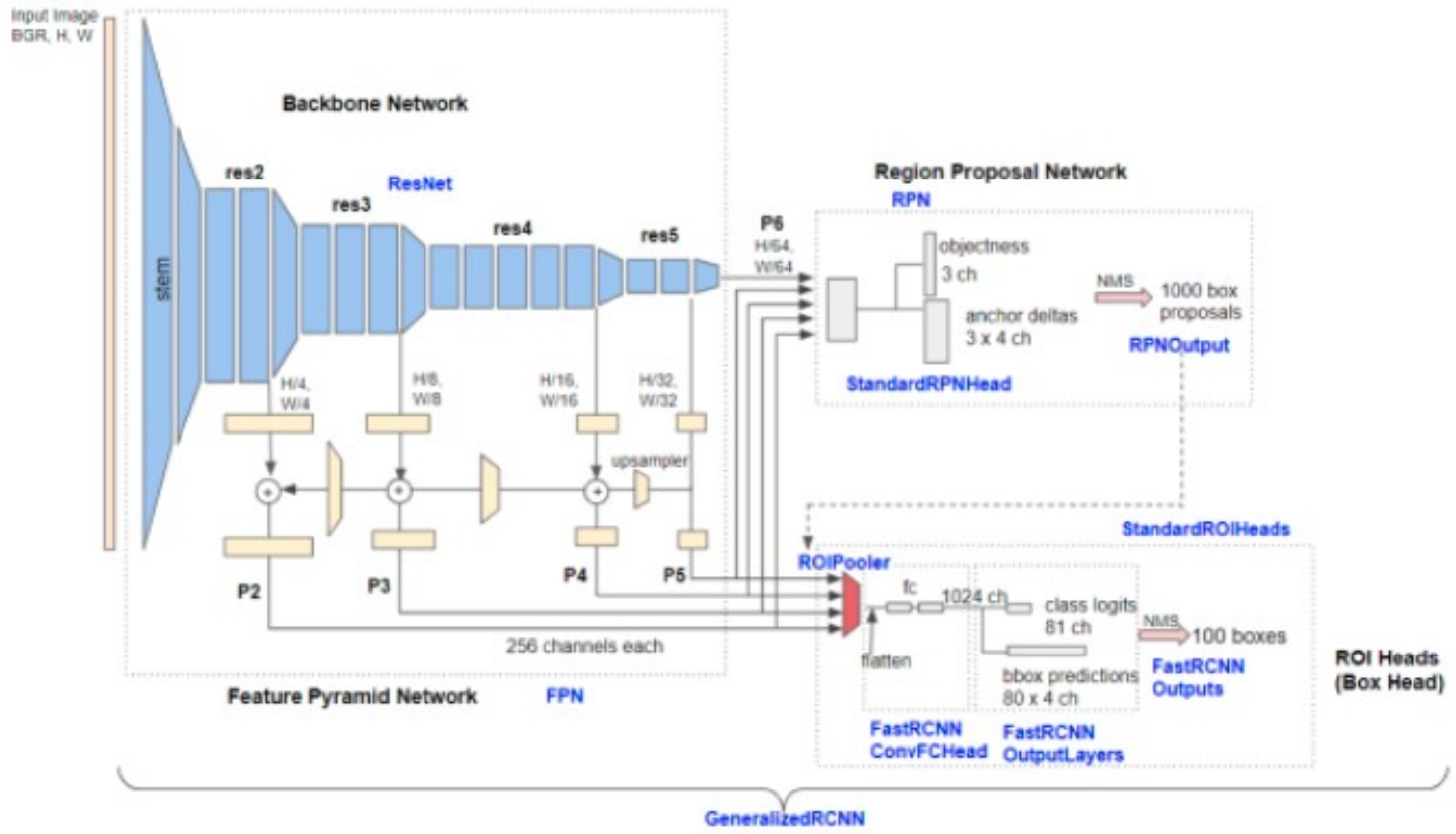
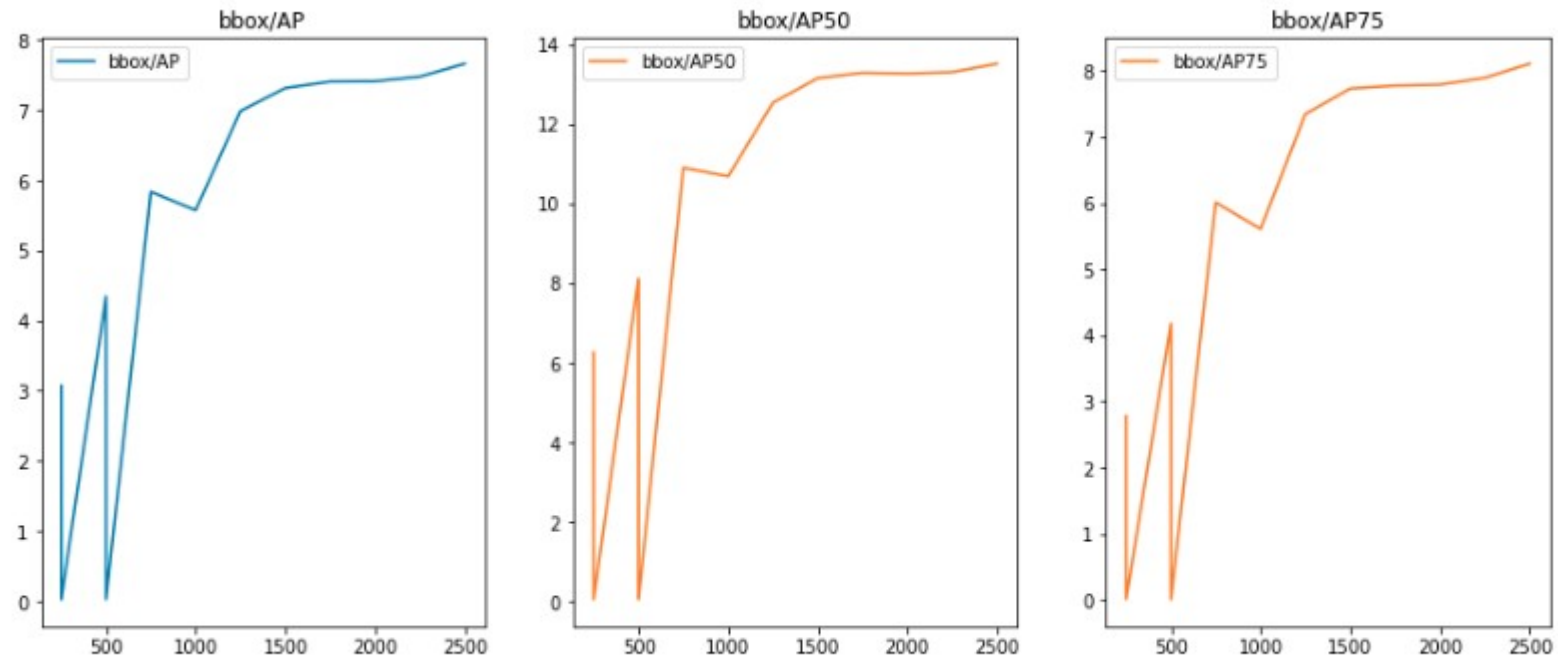


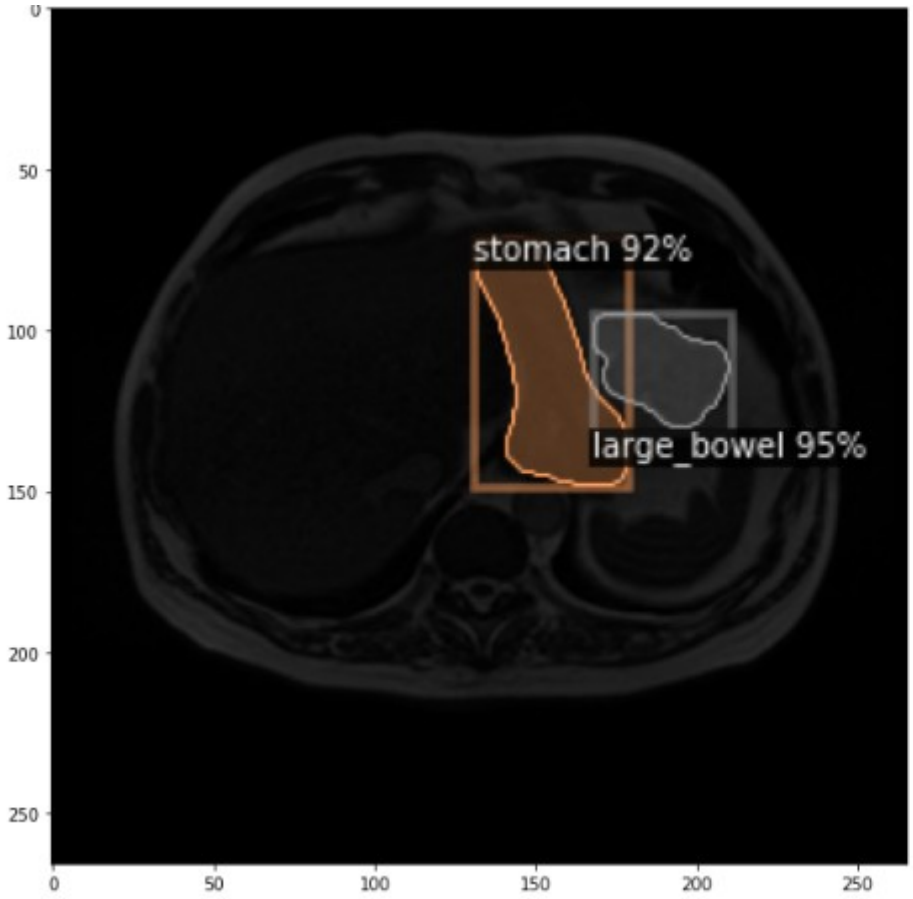
Figure 3. Detailed architecture of Base-RCNN-FPN. Blue labels represent class names.

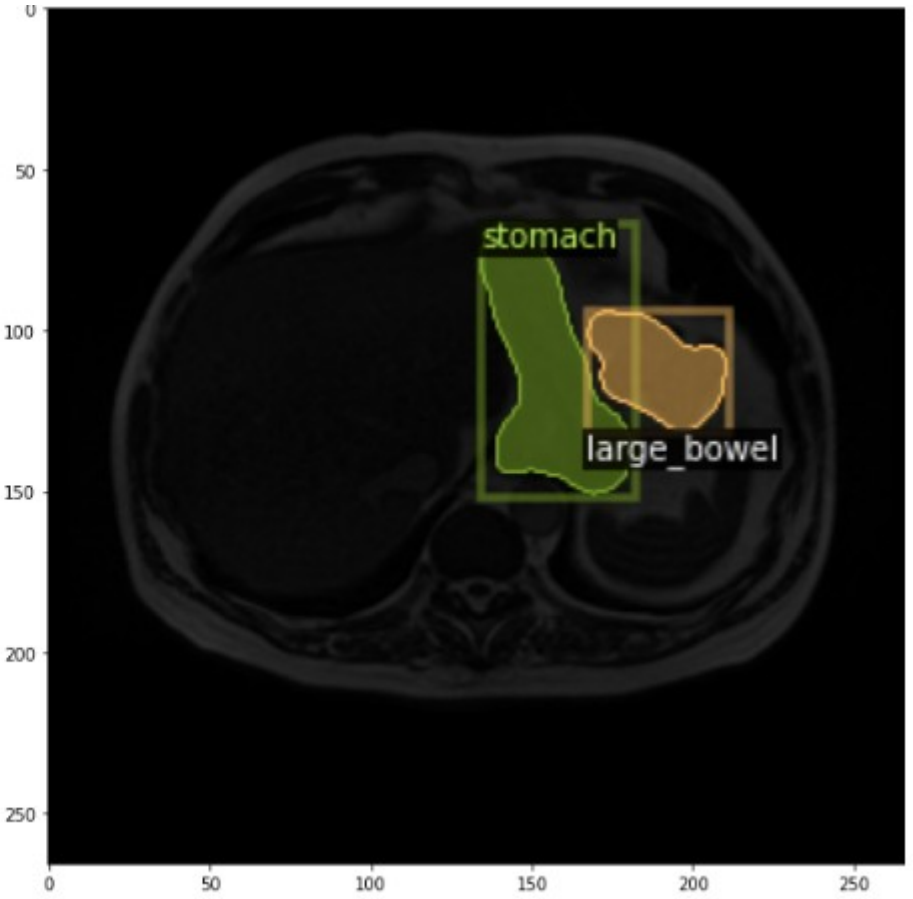
Detectron2 résultats

https://github.com/ismailazdad/uwmgit/blob/main/02_detectron2_model.ipynb

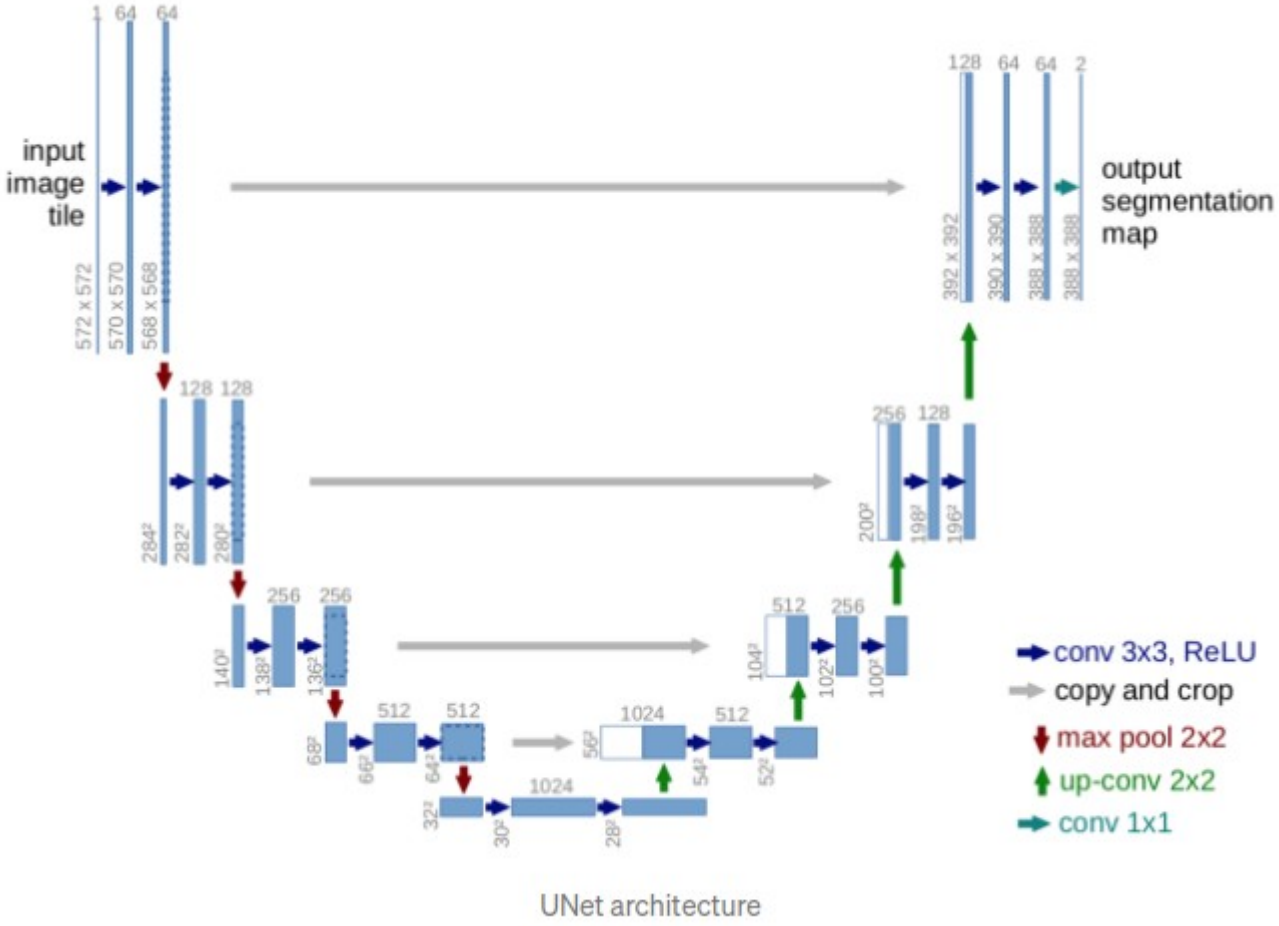


Detectron2 inférence





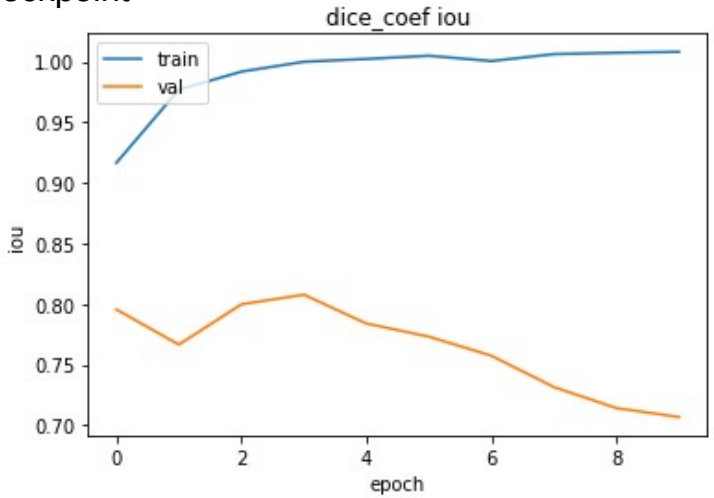
Unet architecture

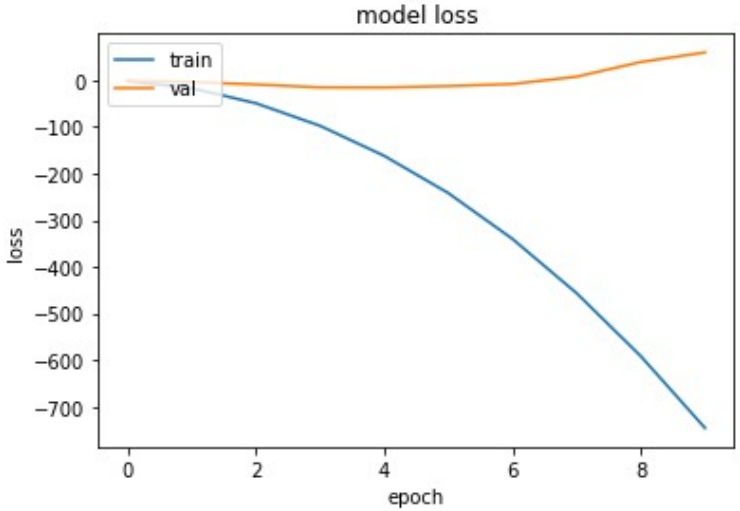


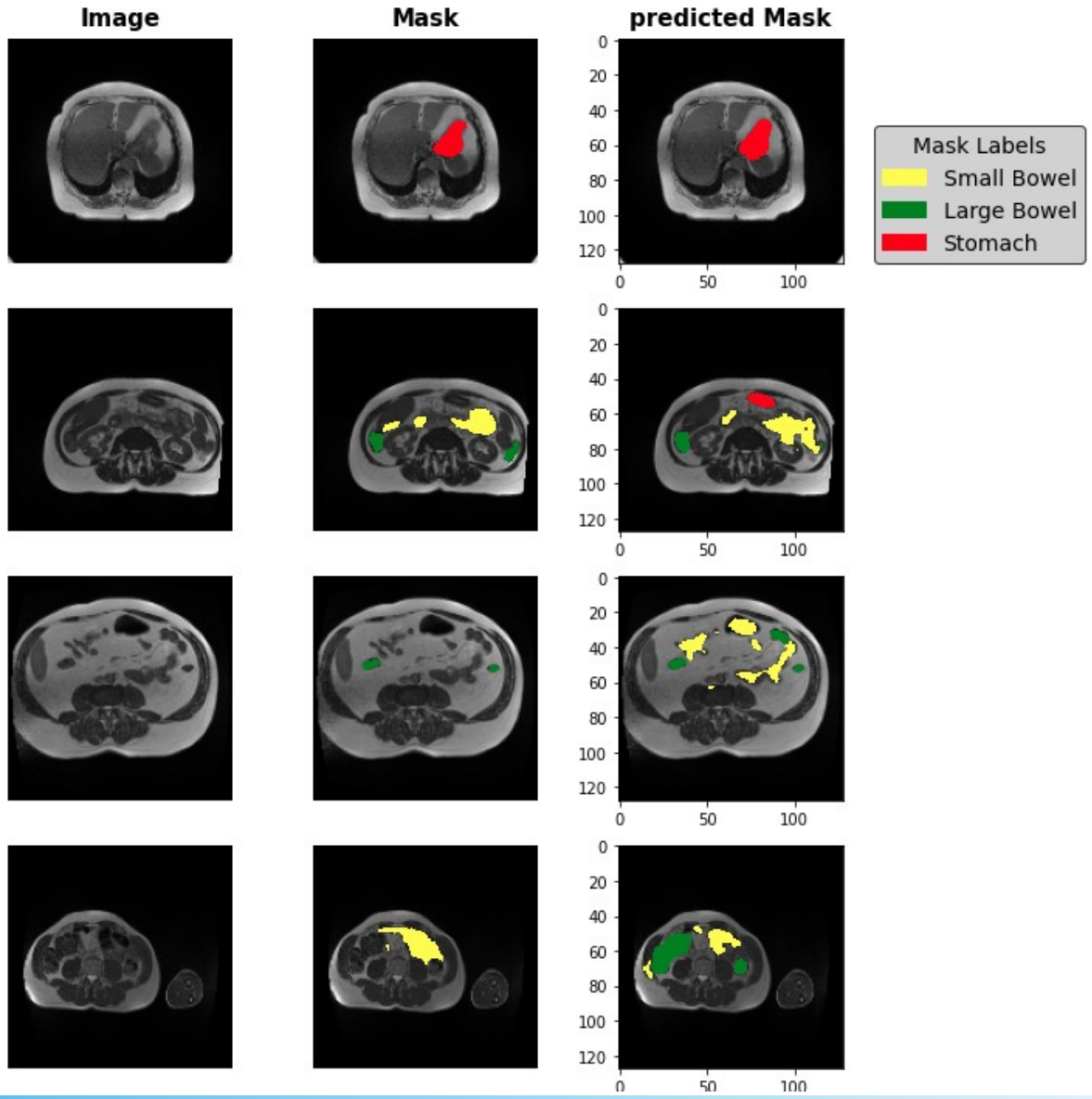
UNet architecture

Unet résultat inférence

Optimisation du learning rate :
early_stopping,
checkpoint







Difficulté rencontrer

- Normalisation des images et des masques depuis l'api coco
- Développement du générateur
- Problème avec Unet sur le loss lors de l entraînement (api ? , encodage rle?)
- La data augmentation (ImageDataGenerator) de tensorflow avec le format coco na pas fonctionner. Les participants utilisant la data augmentation , on majoritairement utiliser le framework Albumination avec pytorch.

La contrainte pour ce projet est que si nous augmentons les données , nous devons faire la même modification pour l image et le masque, ce qui à étais fait , mais les résultats montre que nous sommes en under-fiting…

https://github.com/ismailazdad/uwmgit/blob/main/05_keras_augmentation_fail.ipynb

Conclusion

Projet intéressant mais difficile à appréhender

L Api coco n est pas entièrement développer (notamment pour l évaluation du modèle), peut d exemple sont fourni par microsoft malheureusement

Le génération de données aux formats coco n avait pas étais proposer pour cette compétition, j espéré que ma contribution va servir à la communauté et que quelqu'un trouvera la solution pour ajouter la data augmentation avec la stack technique Keras Unet Coco

Après quelques recherche il s avère que tensorflow est en train de développer la gestion des fichier json, mais il est toujours à titre expérimental, ce qui doit certainement expliquer le problème rencontré,

https://www.tensorflow.org/io/api_docs/python/tfio/experimental/serialization/decode_json

Detectron2 n est pas performant pour notre cas, mais très facile à implémenter, il implémente le format coco nativement

Le transfert learning avec Unet ont de bien meilleur résultats

La segmentation sémantique sous tensorflow offre moins de flexibilité que pytorch, pour la data augmentation

Fin

Thank you
Merci
Arigato
Obrigado
khob khun