Détection d'Objets via DETR , Yolov5 et Analyse de la Tension Artérielle avec la Mixture Gaussienne

Présenté a

Mr. Faghihi Usef

Pour le Cours

INF1022-E Projet de synthèse

Par

Compaoré Yann Djamel

Date de soumission : [2023-09-03]

Résumé

L'architecture DETR et l’algorithme YOLOv5 révolutionnent la reconnaissance d'objets en images. Cette étude explore les techniques utilisées pour la compréhension du texte et pour la détection d'objets dans les images. Parallèlement, nous avons utilisé le modèle de mélange gaussien pour analyser l'impact du sodium sur la pression artérielle. Cette combinaison de méthodes promet des avancées significatives dans l'analyse d'images et la compréhension des relations causales.

Table des matières:

1. Introduction

La reconnaissance d'objets et la compréhension des données sont les fondements de l'apprentissage automatique et de la science des données. Ces domaines sont importants pour transformer de grandes quantités de données brutes en informations utiles et exploitables. Dans le cadre de ce projet plus large, nous nous sommes lancés dans une exploration approfondie d'une variété de techniques et d'algorithmes, à la fois génériques et spécifiques, pour faire face à divers scénarios et défis.

Dans un premier temps, nous avons ciblé l'architecture DETR (Détection Transformer). Cette technologie relativement nouvelle intègre des éléments des premiers transformateurs populaires dans le domaine du traitement du langage naturel. L'utilisation de DETR vise à révolutionner la façon dont nous reconnaissons les objets dans les images et à fournir une alternative aux méthodes traditionnelles.

Au fur et à mesure que nos recherches progressaient, nous avons relevé le défi de détecter différents types de camions très pertinents pour des industries spécifiques. Pour cela, j'ai décidé d'utiliser l'algorithme YOLOv5. YOLO est l'acronyme de "You Only Look Once" et est connu pour sa rapidité et son efficacité de détection d'objets en temps réel. La cinquième version, YOLOv5, apporte des améliorations significatives par rapport aux versions précédentes, augmentant la précision tout en maintenant des performances optimales. Enfin, nous avons diversifié notre approche en plongeant dans des simulations axées sur la santé. Notre objectif était d'en déduire la relation causale entre le sodium et la pression artérielle, un problème majeur de santé publique. Pour améliorer cette analyse, nous avons également utilisé une technique de clustering avancée, le modèle de mélange gaussien. Ce modèle nous a permis de segmenter efficacement nos données et de mettre en évidence des modèles et des tendances qui pourraient autrement passer inaperçus.

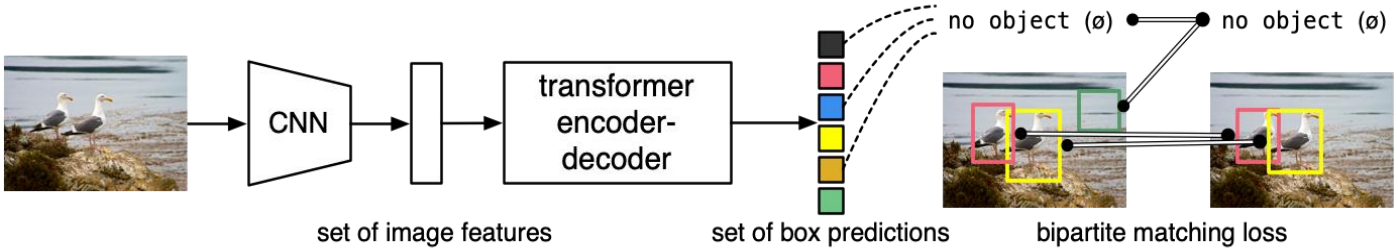
2. Architecture DETR

2.1 Introduction à l'architecture DETR

L'architecture DETR (Détection Transformer) est sans doute l'innovation majeure dans le domaine de la détection d'objets. Ceci est fondamentalement différent de l'approche traditionnelle qui a longtemps dominé le domaine de la vision par ordinateur. Alors que les techniques traditionnelles reposent principalement sur des techniques de cadrage et de régression pour reconnaître et localiser des objets dans les images, DETR adopte une approche différente et utilise des mécanismes d'attention. Ce mécanisme permet de prédire directement non seulement la présence d'un objet, mais aussi sa position exacte dans l'image, sans étapes intermédiaires compliquées.

Ce qui rend le DETR particulièrement attrayant, c'est sa capacité à combiner de manière transparente les concepts de transformateurs et de réseaux de neurones convolutifs (CNN). Les transformateurs ont été conçus à l'origine pour le traitement du langage naturel, mais ont montré une grande adaptabilité à divers domaines, y compris la vision par ordinateur. En combinant ces principes avec des CNN spécifiquement conçus pour traiter des données visuelles telles que des images, DETR peut tirer parti des atouts de ces deux architectures majeures.

En résumé, l'architecture DETR reflète une nouvelle ère dans la détection d'objets, représentant une rupture avec les paradigmes traditionnels tout en ouvrant la porte à de nouvelles possibilités en vision par ordinateur. La possibilité de combiner des techniques éprouvées de manière innovante offre des perspectives intéressantes pour les développements futurs dans la détection et la reconnaissance d'objets.



Facebook Research. (2020). Detection Transformer (DETR). GitHub. https://github.com/facebookresearch/detr

2.2 Utilisation du modèle pré-entraîné

Ce projet a utilisé un modèle DETR pré-entraîné accessible depuis la plateforme GitHub. L'utilisation de modèles préformés vous permet de tirer parti de la puissance des réseaux préformés sur d'énormes ensembles de données tout en économisant beaucoup de temps de formation. Le modèle en question nommé detr_resnet101 est le résultat d'un entraînement intensif sur un grand jeu de données. Cela lui permet de détecter des objets sur une large plage avec une précision étonnante, ce qui en fait un choix idéal pour notre projet.

```
[ ]  import torch as th
     import torchvision.transforms as T
     import requests
     from PIL import Image, ImageDraw, ImageFont
```

```
[ ]  model = th.hub.load('facebookresearch/detr', 'detr_resnet101', pretrained=True)
     model.eval()
     model = model.cuda()
```

2.3 Traitement des images

La qualité des prédictions du modèle dépend fortement de la façon dont les données sont présentées au modèle. Le prétraitement des images est donc une étape essentielle et incontournable dans le processus de travail. Avant que les images ne soient envoyées au modèle DETR, elles sont prétraitées selon un ensemble d'étapes bien définies.
La première partie de ce prétraitement implique la normalisation de l'image. Utilisez des transformations standard pour adapter chaque pixel d'une image à une plage de valeurs spécifique. Cette étape est importante car elle uniformise les données et facilite la manipulation du modèle lors de la reconnaissance. Mais la standardisation n'est pas le seul changement appliqué. D'autres personnalisations telles que le redimensionnement et l'étalonnage des couleurs peuvent également être incluses en fonction des besoins de votre projet particulier. Ces transformations jouent un rôle important en garantissant

que les images sont non seulement dans un format compatible avec le modèle, mais également optimisées pour une précision de reconnaissance maximale.

Il est donc important de choisir le bon modèle, mais la façon dont vous présentez les données est tout aussi importante.

```
[ ]  # standard PyTorch mean-std input image normalization
     transform = T.Compose([
         T.ToTensor(),
         T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
     ])

     CLASSES = [
         'N/A', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
         'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'N/A',
         'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse',
         'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'N/A', 'backpack',
         'umbrella', 'N/A', 'N/A', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis',
         'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove',
         'skateboard', 'surfboard', 'tennis racket', 'bottle', 'N/A', 'wine glass',
         'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich',
         'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake',
         'chair', 'couch', 'potted plant', 'bed', 'N/A', 'dining table', 'N/A',
         'N/A', 'toilet', 'N/A', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',
         'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'N/A',
         'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier',
         'toothbrush'
     ]
```

```
[ ]  url = input()
```

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTLWYgHl_I3Zb6Cru7KafbAy98_5u-U-kcdFw&usqp=CAU

```
[ ]  img = Image.open(requests.get(url, stream=True).raw).resize((800,600)).convert('RGB')
```

```
img_tens = transform(img).unsqueeze(0).cuda()
```

```
with th.no_grad():
  output = model(img_tens)
```

```
[ ] im2 = img.copy()
    drw = ImageDraw.Draw(im2)
    pred_logits=output['pred_logits'][0][:, :len(CLASSES)]
    pred_boxes=output['pred_boxes'][0]

    max_output = pred_logits.softmax(-1).max(-1)
    topk = max_output.values.topk(15)

    pred_logits = pred_logits[topk.indices]
    pred_boxes = pred_boxes[topk.indices]
    pred_logits.shape
```

```
torch.Size([15, 91])
```

```
  cls = logits.argmax()
  if cls >= len(CLASSES):
    continue
  label = CLASSES[cls]
  print(label)
  box = box.cpu() * th.Tensor([800, 600, 800, 600])
  x, y, w, h = box
  x0, x1 = x-w//2, x+w//2
  y0, y1 = y-h//2, y+h//2
  drw.rectangle([x0, y0, x1, y1], outline='red', width=5)
  drw.text((x, y), label, fill='white')
```

```
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
dog
```

2.4 Résultats et visualisation

Après avoir exécuté le modèle sur une image donnée, nous avons obtenu des résultats sous deux formes principales : les classes et les boîtes englobantes. Plus précisément, chaque objet détecté dans une image se voit attribuer une « classe » qui indique la nature ou le type de l'objet (comme « voiture », « chat », « arbre », etc.). En même temps, une "boîte englobante" est générée pour chacun de ces objets, contraignant optiquement leur position et leur taille dans l'image.

Cette visualisation est très utile, notamment si vous souhaitez vérifier ou valider rapidement les performances de votre modèle. Une représentation graphique est fournie qui rend les résultats très compréhensibles, permettant à toute personne sans connaissances techniques approfondies de comprendre les caractéristiques et les conclusions du modèle. En fin de compte, la combinaison de ces prédictions précises avec une visualisation efficace crée un outil puissant pour l'analyse et l'interprétation des images.

Avant :

Après :

3. Détection de camions avec YOLOv5

3.1 Présentation de YOLOv5

YOLO, acronyme de "You Only Look Once", est une famille d'algorithmes de détection d'objets qui ont révolutionné le domaine par leur rapidité et leur précision. La dernière version de cette série, YOLOv5, offre des améliorations significatives par rapport à ses prédécesseurs en termes de performances et d'efficacité.

3.2 Collecte des donnees.

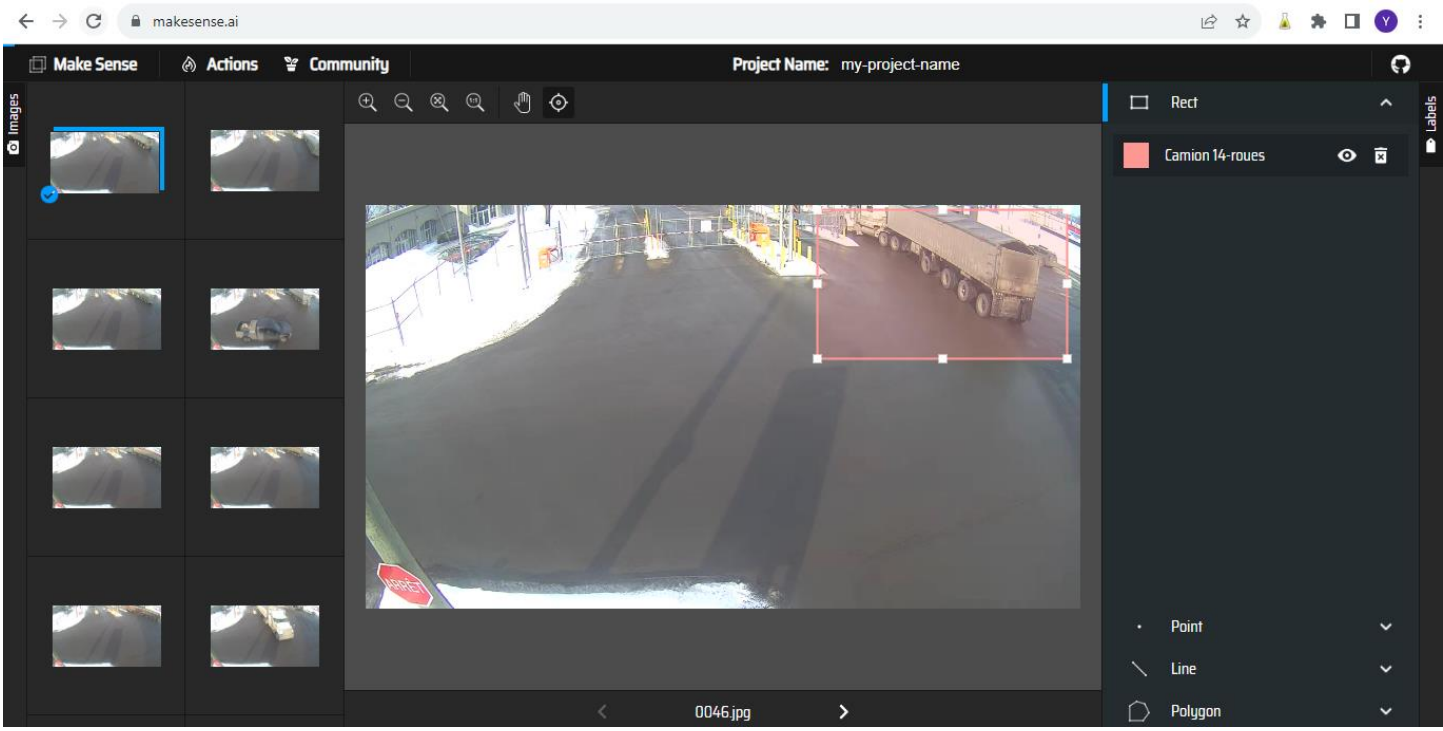
Afin de disposer de suffisamment d'images à annoter, nous avons décomposé des vidéos de camions en images en utilisant un script python.

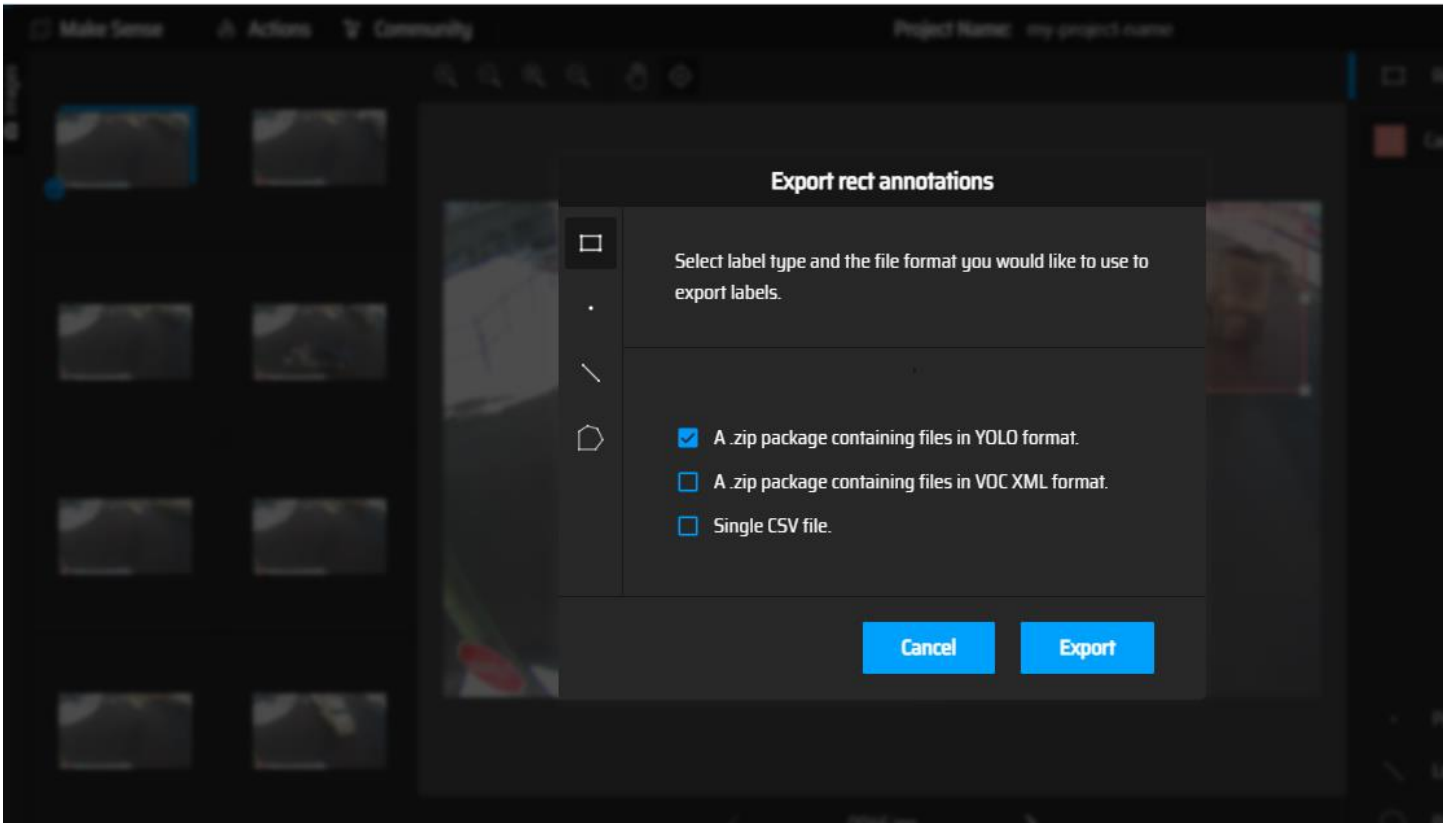
```python
import cv2
import os

def decompose_video_to_images(chemin_video, sortie, image_prefix='frame'):
    if not os.path.exists(sortie):
        os.makedirs(sortie)

    # Ouvrir la vidéo
    cap = cv2.VideoCapture(chemin_video)
    if not cap.isOpened():
        print("Erreur: Impossible d'ouvrir la vidéo.")
        return

    frame_count = 0
    while True:
        # Lire une image de la vidéo
        ret, frame = cap.read()
        if not ret:
            break

        # Sauvegarder l'image
        image_path = os.path.join(sortie, f"{image_prefix}_{frame_count}.jpg")
        cv2.imwrite(image_path, frame)
        frame_count += 1

    cap.release()
    print(f"{frame_count} images ont été sauvegardées dans {sortie}.")

# Utilisation:
decompose_video_to_images('../video.mp4', '../train_data/images')

```

3.3. Annotation des images

L'annotation également appelée labelling de nos images est une étape longue mais qui peut être simplifiée grâce à certains outils en ligne tel que makesense.ai

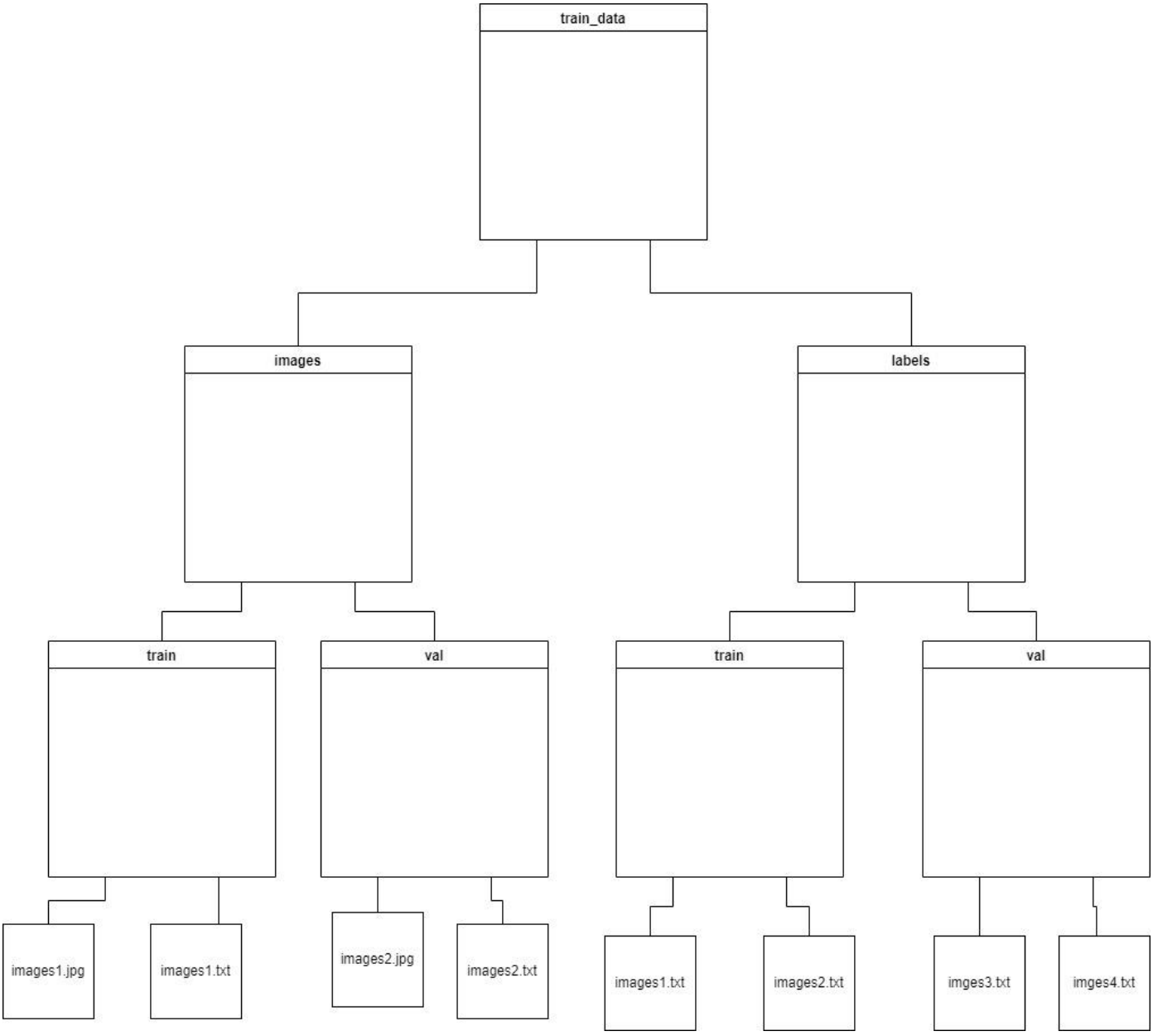
makesense.ai est un outil en ligne gratuit pour étiqueter les photos. Grâce à l'utilisation d'un navigateur, il ne nécessite aucune installation compliquée, il suffit de visiter le site Web et vous êtes prêt à commencer. Peu importe le système d'exploitation sur lequel vous utilisez. Le format de sortie des annotations est le format yolo





3.4. Organisation des donnees

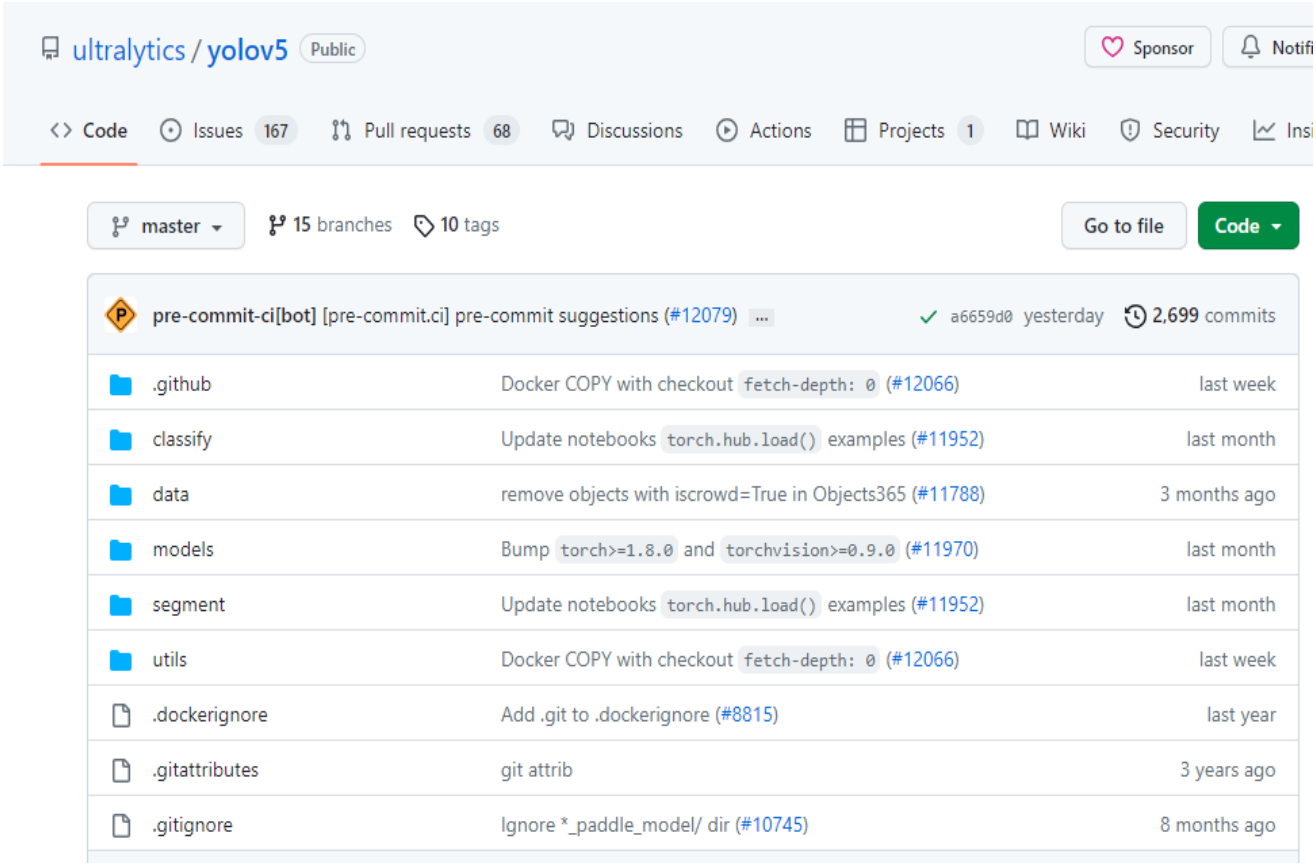
Après avoir terminé d'annoter les images et exporter leur annotations respectives, l'étape suivante est d'organiser nos images et annotations selon une architecture précise avec 70% de donnée d'entrainement et 30 % pour la validation.



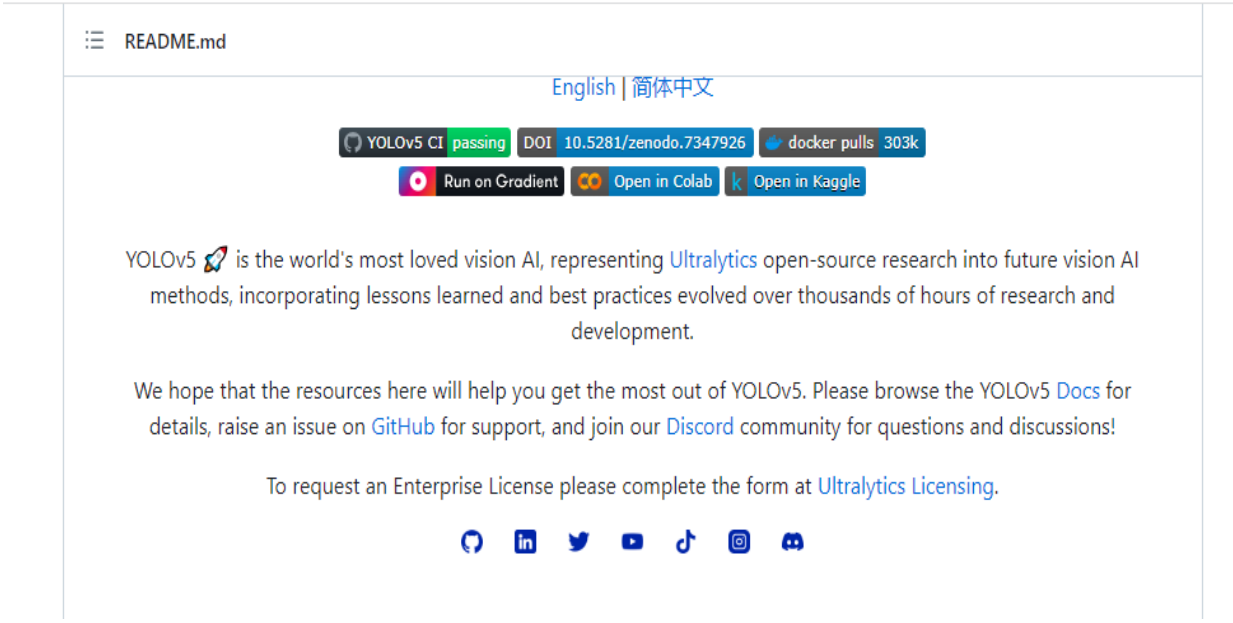
3.5. Configuration de l'environnement d'entrainement

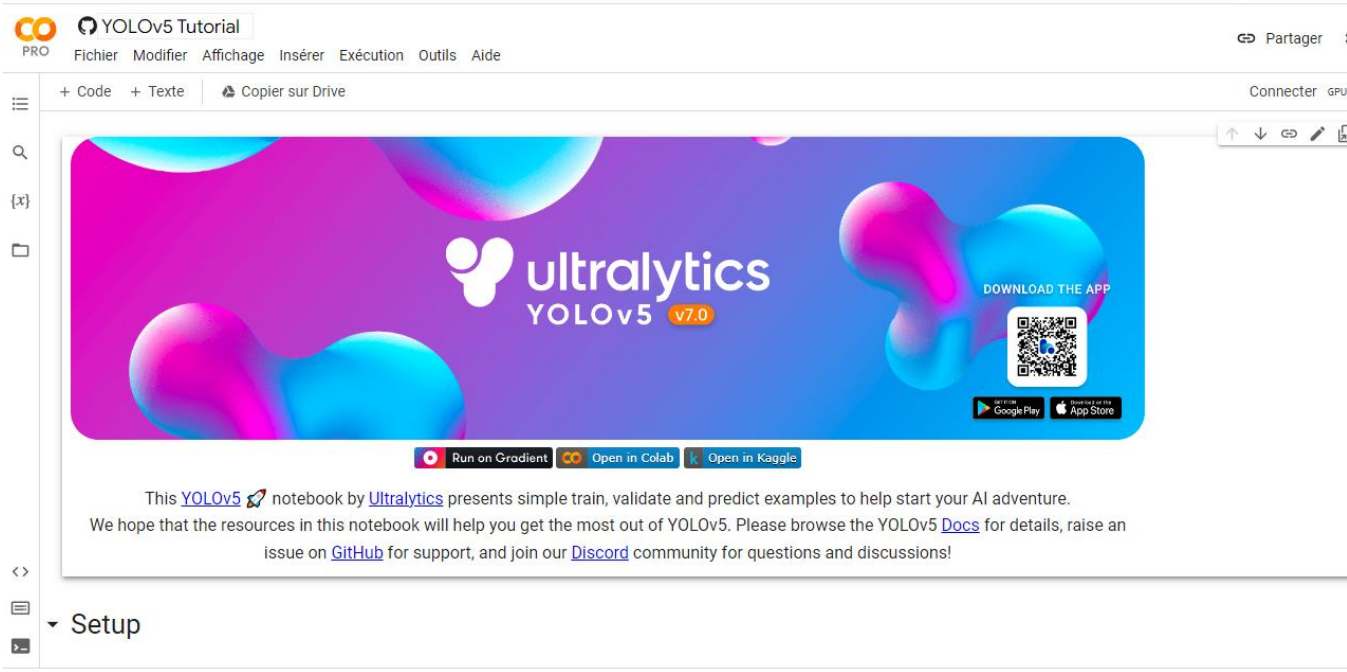
Pour ce projet nous avons utilisé l'algorithme yolov5 disponible sur GitHub.

a) Accès au dépôt GitHub de YOLOv5.



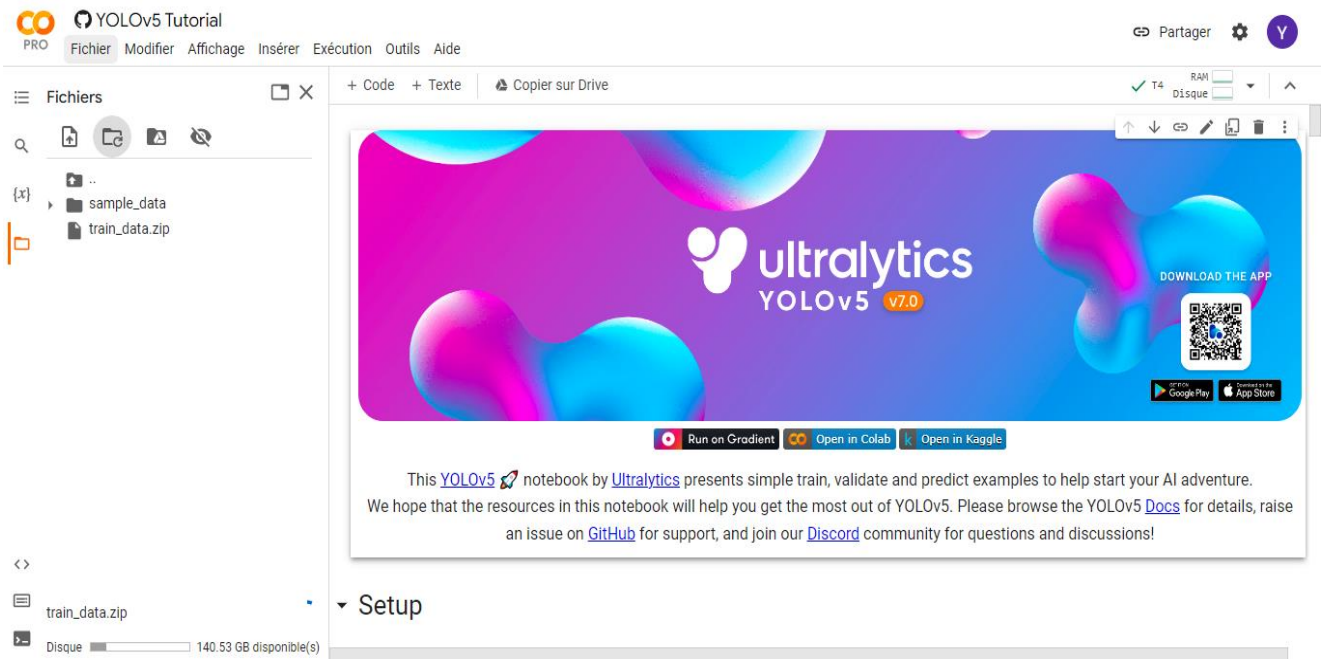
b) Ouverture du dépôt dans Google Colab.





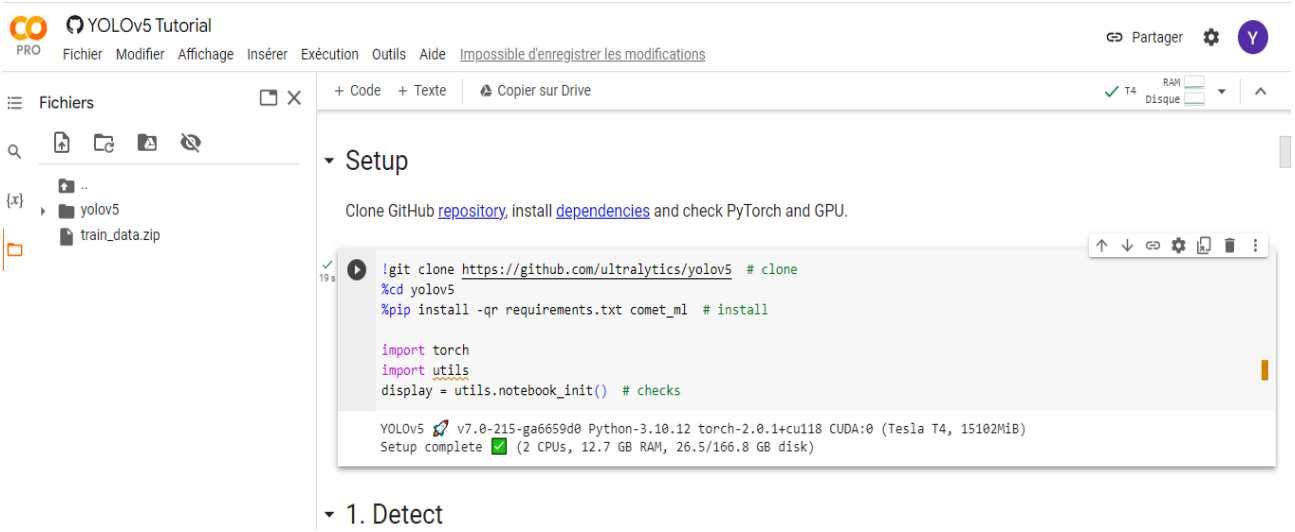
c) Importation de nos donnees train_data archivé dans Colab.

Après avoir ouvert le notebook sur google colab, l'on peut importer notre dataset directement dans google collab



3.6. Préparation pour l'entrainement

a) Clonage et configuration initiale pour YOLOv5 dans Colab.



b) Modification du fichier coco128.yaml pour l'adapter aux besoins spécifiques du projet :

- Renommé en custum_data.yaml.

  Le fichier à renommé se trouver dans : /content/yolov5/data/coco128.yaml

- Mise à jour des classes pour inclure "camion 4-roues" jusqu'à "camion 16-roues".

  Par la suite nous allons modifier les chemins d'accès et classes se trouvant dans notre fichier renommé :

  Avant :

```yaml

# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/coco128  # dataset root dir
train: images/train2017  # train images (relative to 'path') 128 images
val: images/train2017  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
names:
  0: person
  1: bicycle
  2: car
  3: motorcycle
  4: airplane
  5: bus
  6: train
  7: truck
  8: boat
  9: traffic light
  10: fire hydrant
  11: stop sign
  12: parking meter
  13: bench
  14: bird
  15: cat
  16: dog
  17: horse
  18: sheep
  19: cow
  20: elephant
  21: bear
  22: zebra
```

Après:

C: > Users > Utilisateur > Desktop > Session 5 > ProjetCamion > custum_data.yaml

```yaml
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
#COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017) by Ultralytics
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128  ← downloads here (7 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../train_data  # dataset root dir
train: images/train  # train images (relative to 'path')
val: images/val  # val images (relative to 'path')


# Classes
names:
  0: Camion 4-roues
  1: Camion 8-roues
  2: Camion 10-roues
  3: Camion 12-roues
  4: Camion 14-roues
  5: Camion 16-roues
```

3.7. Entrainement du modèle

L'entrainement est une étape cruciale car il est important de bien ajuster les hyperparamètres afin de ne pas biaiser les performances du modèle.

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 1 --epochs 20 --data custum_data.yaml --weights yolov5s.pt --cache
```

3.8 Détection des camions dans une vidéo.

Après l'entrainement, notre modèle (best.pt) se trouvera dans le répertoire /content/yolov5/runs/train/exp/weights/

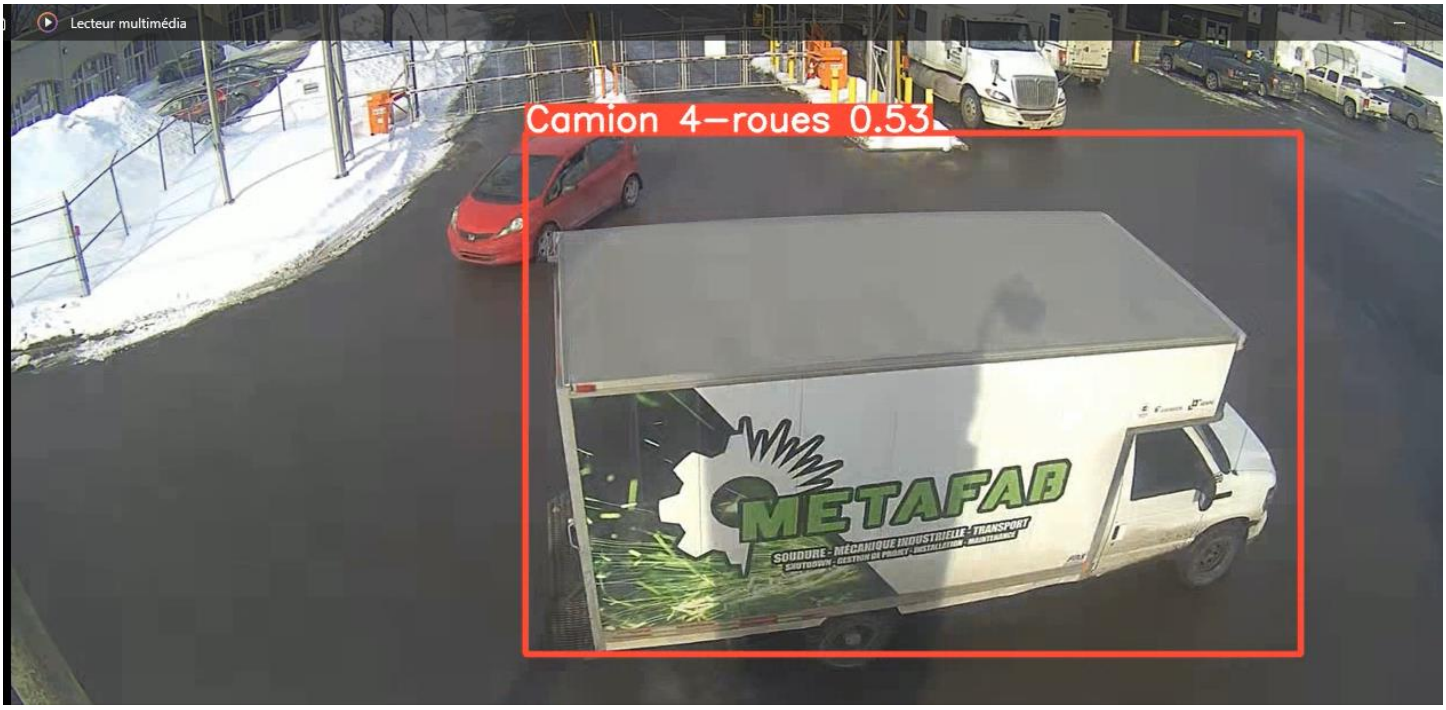
On le testera ensuite sur une vidéo de camions de plusieurs types préalablement importer dans le notebook.

```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --img 640 --conf 0.5 --source ../video5.
# display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

3.9. Résultats

Après avoir exécuter la cellule ci-dessus, il devrait se créer un répertoire /content/yolov5/runs/detects/exp/ dans le notebook dans lequel se trouve notre fichier vidéo avec toutes les détections. Téléchargez-le !!!


Lecteur multimédia
Camion 8-roues


Lecteur multimédia
Camion 4-roues 0.53
METAFAB
SOUDURE - MÉCANIQUE INDUSTRIELLE - TRANSPORT

4. Clustering avec l'algorithme de mixture Gaussien

4.1 Aperçu générale de nos données

La fonction generate_data génère un ensemble de données simulées sur la pression artérielle en fonction de l'âge et de la teneur en sodium. Utilisez les paramètres par défaut pour définir la taille de l'échantillon, l'effet du sodium et d'autres variables. Les utilisateurs peuvent choisir soit une thérapie binaire au sodium, soit une thérapie continue au sodium. Les données renvoyées incluent la tension artérielle, la concentration de sodium, l'âge et la protéinurie sous forme de trames de données.

```
def generate_data(n=1000, seed=0, beta1=1.05, alpha1=0.4, alpha2=0.3, binary_treatment=True, binary_cutoff=3.5):
    np.random.seed(seed)
    age = np.random.normal(65, 5, n)
    sodium = age / 18 + np.random.normal(size=n)
    if binary_treatment:
        if binary_cutoff is None:
            binary_cutoff = sodium.mean()
        sodium = (sodium > binary_cutoff).astype(int)
    blood_pressure = beta1 * sodium + 2 * age + np.random.normal(size=n)
    proteinuria = alpha1 * sodium + alpha2 * blood_pressure + np.random.normal(size=n)
    hypertension = (blood_pressure >= 140).astype(int)  # not used, but could be used for binary outcomes
    return pd.DataFrame({'blood_pressure': blood_pressure, 'sodium': sodium,
                         'age': age, 'proteinuria': proteinuria})
```

4.2. Application de la mixture Gaussienne

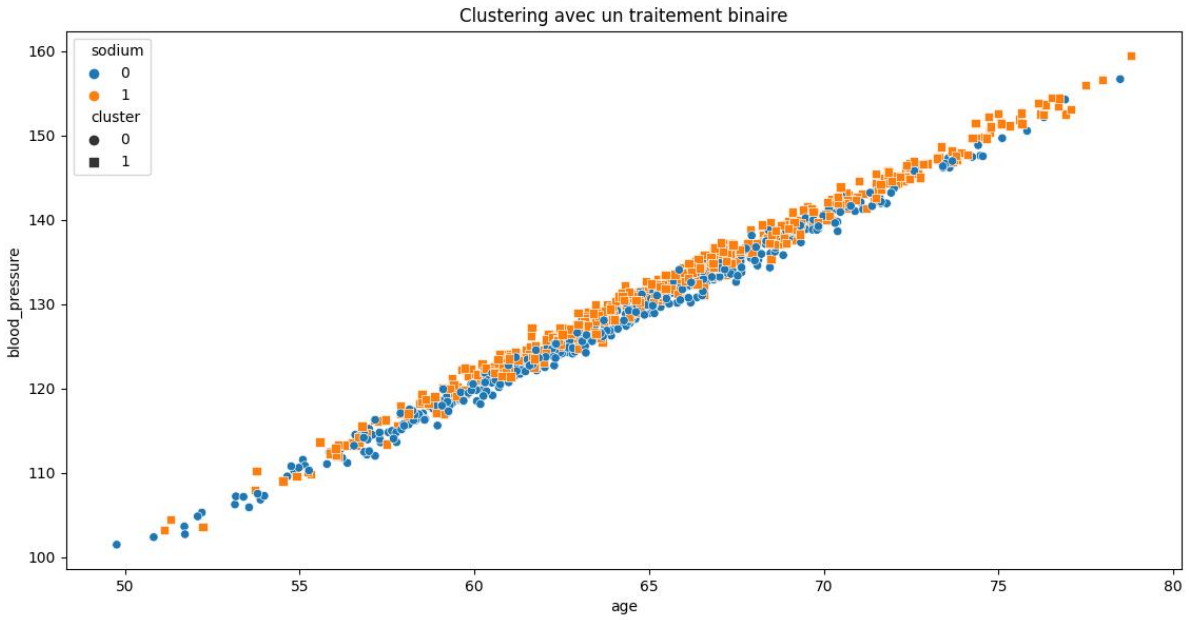
a) Traitement binaire

```python
#Traitement binaire

df = generate_data(n=1000, beta1=1.05, alpha1=.4, alpha2=.3, binary_treatment=True)

#Preparez les donnees pour le clustering
df = df[['sodium', 'blood_pressure', 'age', 'proteinuria']]

scaler = StandardScaler()

df1= scaler.fit_transform(df)

# Appliquez la Mixture Gaussienne
gmm = GaussianMixture(n_components=2)
gmm.fit(df1)
clusters = gmm.predict(df1)

# Ajoutez les clusters au DataFrame
df['cluster'] = clusters
print(df.head(10))
```

```
(env) PS C:\Users\Utilisateur\Desktop\Session 5\Mixture Gaussienne> python sodium_
example.py
   sodium  blood_pressure        age  proteinuria  cluster
0       1      147.157602  73.820262    46.140554        1
1       1      133.339602  67.000786    40.970603        1
2       0      139.833515  69.893690    41.835567        0
3       1      152.500558  76.204466    46.401798        1
4       1      149.644768  74.337790    44.082575        1
5       1      120.573362  60.113611    36.178275        1
6       1      139.780100  69.750442    42.419283        1
7       0      128.005583  64.243214    38.501097        0
8       1      130.721397  64.483906    38.085803        1
9       1      136.085130  67.052993    41.553162        1
```

Visualisation :

```python
# Plotting
plt.figure(figsize=(10, 7))
sns.scatterplot(x='age', y='blood_pressure', hue='sodium', style='cluster', data=df, markers=["o", "s"])
plt.title('Clustering avec un traitement binaire')
plt.show()
```



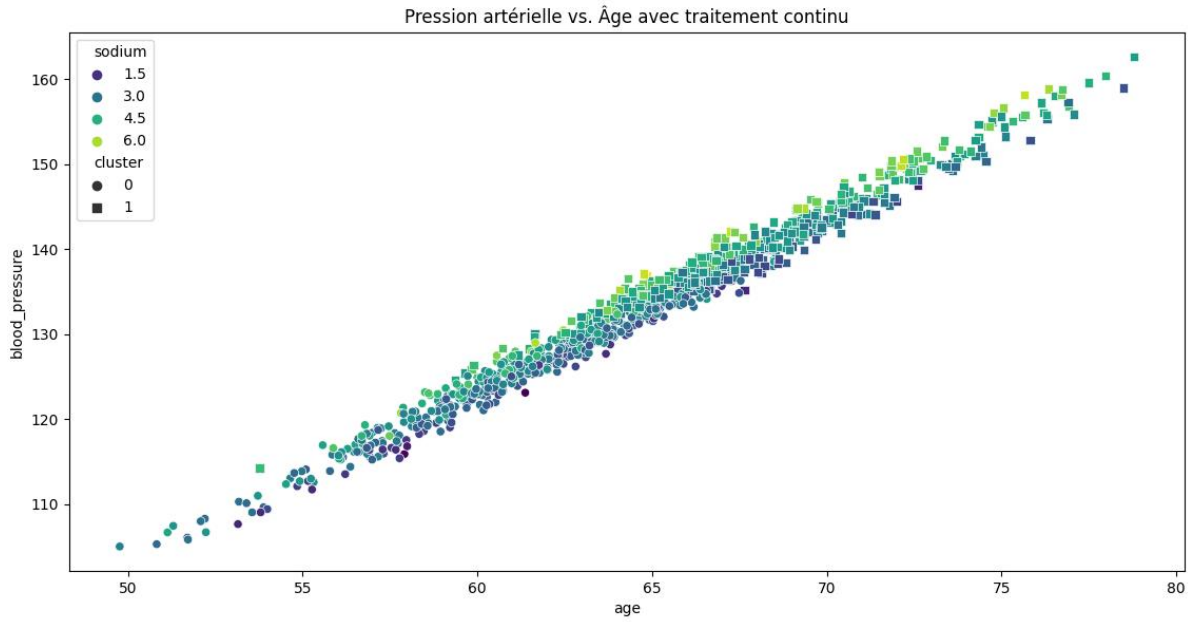
b) Traitement continu

```python
#Traitement continue

df2 = generate_data(n=1000, beta1=1.05, alpha1=.4, alpha2=.3, binary_treatment=False)

#Preparez les donnees pour le clustering
df2 = df2[['sodium', 'blood_pressure', 'age', 'proteinuria']]

scaler = StandardScaler()

df3= scaler.fit_transform(df2)

# Appliquez la Mixture Gaussienne
gmm = GaussianMixture(n_components=2)
gmm.fit(df3)
clusters = gmm.predict(df3)

# Ajoutez les clusters au DataFrame
df2['cluster'] = clusters
print(df2.head(10))
```

	sodium	blood_pressure	age	proteinuria	cluster
0	4.657088	150.997545	73.820262	48.755372	1
1	4.614740	137.135079	67.000786	43.555142	0
2	3.460668	143.467216	69.893690	44.309945	1
3	4.338295	156.005768	76.204466	48.788679	1
4	4.357931	153.170595	74.337790	46.483495	1
5	3.541125	123.241543	60.113611	37.995179	0
6	4.415798	143.366688	69.750442	44.861578	1
7	1.750990	129.844122	64.243214	39.753054	0
8	3.533115	133.381168	64.483906	39.896980	1
9	3.964200	139.197540	67.052993	43.672565	1

Visualisation :

```
plt.figure(figsize=(10, 7))
sns.scatterplot(x='age', y='blood_pressure', hue='sodium', style='cluster', data=df2, markers=["o", "s"], palette="viridis")
plt.title('Clustering avec traitement continu')
plt.show()
```



5. Discussion

Les progrès dans le domaine de la reconnaissance d'objets au cours de la dernière décennie ont été vraiment remarquables. Nous sommes passés de méthodes primitives à des algorithmes sophistiqués capables de traiter des images avec une précision étonnante.

DETR et YOLOv5 :

L'introduction d'architectures telles que DETR et YOLOv5 en est la preuve. DETR a révolutionné la façon dont nous abordons cette tâche en intégrant un transformateur, qui est courant dans le traitement du langage naturel, avec détection d'objet. Il élimine le besoin d'approches de cadrage traditionnelles, simplifiant le processus tout en maintenant une grande précision. D'autre part, YOLOv5 améliore encore l'efficacité de la

série YOLO, rendant la détection d'objets en temps réel non seulement possible mais également efficace.

Détection de piste spécifique :

En se concentrant sur une tâche spécifique, telle que la détection de piste basée sur le nombre de roues, nous pouvons voir comment ces algorithmes peuvent être adaptés à des besoins de niche très spécifiques. Bien sûr, de tels défis nécessitent une attention particulière à la fois à l'annotation des données et à la formation du modèle pour s'assurer que les nuances sont correctement capturées. Les premiers résultats sont prometteurs, mais comme toute technologie en évolution, il y a toujours place à l'amélioration.

Simulation pour l'analyse des causes profondes :

L'inférence est un domaine sensible et l'utilisation de données synthétiques comme ce que nous avons fait pour étudier les effets du sodium sur la tension artérielle représente une approche innovante. Bien que ces simulations soient basées sur des données non réelles, les chercheurs contrôlent tous les paramètres, fournissant un environnement de test exempt de biais potentiels présents dans les données réelles. Cela crée une plate-forme puissante pour comprendre les connexions et tester d'abord différentes hypothèses.

En résumé, ces efforts reflètent une époque où la technologie et l'innovation travaillent ensemble pour repousser les limites de ce que nous pensions possible, ouvrant la voie à de nouvelles découvertes et avancées dans le monde de l'intelligence artificielle et de la recherche.

6. Conclusion

Ce projet démontre comment la science des données et l'apprentissage automatique peuvent être utilisés pour répondre à une grande variété de questions complexes. En explorant différents domaines, de la vision par ordinateur à l'analyse causale, nous avons pu démontrer l'étendue et la profondeur des outils à notre disposition.

Détection d'objets avec DETR et YOLOv5 :

L'utilisation de DETR et YOLOv5 montre des progrès rapides dans la reconnaissance d'objets. Ces dernières architectures ont non seulement amélioré la précision, mais ont également introduit de nouvelles méthodes de traitement d'image, démontrant ainsi les progrès continus de la vision par ordinateur. Les résultats obtenus, notamment dans la reconnaissance spécifique des types de camions, montrent comment le modèle généraliste peut être adapté à des besoins spécifiques avec une grande précision.
Simulation et analyse des causes profondes :
D'autre part, les simulations pour comprendre la causalité ont mis en évidence un aspect souvent négligé de la science des données : l'importance d'une analyse statistique rigoureuse. Plutôt que de s'appuyer uniquement sur des données du monde réel, qui peuvent être sujettes à divers biais, les simulations nous ont permis de créer des scénarios contrôlés pour tester nos hypothèses. Cette approche fournit un moyen robuste d'explorer les associations potentielles, soulignant la nécessité de valider les résultats dans des études supplémentaires basées sur des données du monde réel.
Certificat de fin d'études :
Chaque composante de ce projet, qu'il s'agisse de la détection d'objets ou de l'analyse causale, a apporté ses propres défis et avantages. Mais ce qui ressort le plus, c'est la capacité de la science des données à personnaliser et à répondre à une grande variété de questions, qu'elles concernent la vision, la santé ou d'autres domaines. Ce projet reflète l'incroyable potentiel de l'apprentissage automatique et de la science des données et comment ils continuent de façonner et de redéfinir notre compréhension du monde qui nous entoure.

7. Recommandations

Pour les recherches futures, il serait avantageux d'étudier d'autres architectures de détection d'objets et de comparer leur efficacité. De plus, d'autres études pourraient être menées pour valider nos conclusions concernant l'effet du sodium sur la pression artérielle à l'aide de données réelles.

8. Références

Facebook Research. (2020). Détection Transformer (DETR). GitHub. https://github.com/facebookresearch/detr.

Meunier, François. (17/03/2022). Collection de vidéos de camions. Université du Québec à Trois-Rivières. http://dmiftp.uqtr.ca/Public/FMeunier/ImagesETVideo/VideoCamions/

Ultralytics. (2023). YOLOv5: Fifth version of You Only Look Once, a real-time object detection system. GitHub. https://github.com/ultralytics/yolov5

Neal, Brady. (2020). Exemple de code sur la causalité : Sodium. Dans Code pour le livre sur la causalité. GitHub. https://github.com/bradyneal/causal-book-code/blob/master/sodium_example.py

OpenAI (2020). ChatGPT. Disponible sur: https://www.openai.com/

Makesense.ai - Outil d'annotation d'images en ligne. Accessible à : https://www.makesense.ai (consulté le [08-2023]).

9. Annexes

Dépôt GitHub : Ultralytics - Collection d'outils et de scripts

Description : Ce dépôt contient une autre version de l'algorithme yolo (Yolov8)

Organisation : Ultralytics

Langage de Programmation : Python

URL: https://github.com/ultralytics/ultralytics