

Recent Advanced in Machine Learning

Computer vision : détection d'objets ([Github](#))

Réalisé par : Adjoua HOUNDONOUGBO & Morel Mbedi

Master 2 MALIA, Université Lyon 2

Introduction

Ce projet s'inscrit dans le cadre du cours de « Advanced method in Machine Learning » que nous avons suivi pendant la formation du master 2 de l'université Lumière Lyon 2. Le but est de faire la détection d'objets dans des images en utilisant les modèles pré-entraînés et spécialisés dans cette tâche.

En pratique, il s'agira de tester le modèle pré-entraîné avec les données tests utilisées, afin d'évaluer de façon qualitative et quantitative les résultats obtenus dans ce cas exploratoire. Ensuite, il faut tester le modèle pré-entraîné avec des images collectées sur l'internet. Enfin, nous finirons par faire la fine tuning pour voir si les résultats obtenus dans ce cas sont meilleurs que dans les deux cas précédents.

Pour réaliser toutes ces tâches, nous avons choisi de faire la détection des 4 objets, notamment la détection de chien, de chat, des pommes et des oranges.

Description de l'architecture du modèle de réseau de neurones utilisé

Dans le problème de vision par ordinateur, il existe plusieurs modèles pré-entraînés pour faire la détection d'objets sur les images. Dans le cadre de ce projet, nous allons utiliser le modèle `fasterrcnn_resnet50_fpn` basé sur la librairie `pytorch`. Il a été pré-entraîné sur plusieurs données de détection d'objets, notamment celles de COCO qui comprennent 91 classes ou type d'objets différents. Le modèle Faster R-CNN avec ResNet50 est une version améliorée de Faster R-CNN qui utilise la résilience des blocs résiduels et la pyramide de caractéristiques pour améliorer la précision de la détection d'objet.

Fonctionnement de l'architecture

Faster R-CNN est une architecture de détection d'objets en deux étapes. La première étape consiste à proposer des régions d'intérêt (RoI) dans l'image qui pourraient contenir des objets.

La deuxième étape consiste à classer les RoI proposées et à prédire leur boîte englobante (bounding box) pour localiser l'objet. Plus précisément, le modèle "faster-rcnn-resnet50-fpn-coco-torch" utilise une architecture de réseau de neurones convolutifs appelée ResNet-50 comme extracteur de caractéristiques pour les RoI proposées. Il utilise également une fonction d'activation appelée "ReLU" (Rectified Linear Unit) pour introduire de la non-linéarité dans le modèle pour les couches de convulsion et le softmax a été utilisé pour la couche de sortie qui permet de prédire les probabilités de plusieurs classes d'objets simultanément

Les différents paramètres du modèle :

- **Weights** : les poids du modèle pré-entraînés à mettre à True (weight=True) pour utiliser ces poids.
- **Num_classes** : nombre de classes à détecter, nous allons prendre 4 classes
- **weights_backbone** : les poids pré-entraînés du CNN qui extraient les caractéristiques de l'image. Ce modèle est le ResNet50.
- **trainable_backbone_layers** : précise le nombre de couches trainable par défaut c'est 3

Pour implémenter ce modèle, nous avons utilisé la librairie fiftyone qui comprend plusieurs modèles pré-entraînés et plusieurs méthodes d'évaluation des problèmes de détection d'objets.

Description de la librairie « [fiftyone](#) »

Fiftyone est une librairie riche en méthodes de résolution des tâches de vision par ordinateur. Il permet de charger, modifier, visualiser et évaluer facilement vos données ainsi que toutes les étiquettes associées (classifications, détections, etc.). Ils fournissent une interface interactive sur python et cohérente pour charger des images, des vidéos, des annotations et des prédictions de modèles dans un format qui peut être visualisé dans l'application FiftyOne.

Méthodes d'évaluations de fiftyone

FiftyOne fournit une variété de méthodes intégrées pour évaluer les prédictions de votre modèle, y compris les régressions, les classifications, les détections, les polygones, les segmentations d'instance et sémantiques, sur les ensembles de données d'images et de vidéos. Lorsque vous évaluez un modèle dans FiftyOne, vous avez accès aux métriques agrégées standard telles que les rapports de classification, les matrices de confusion et les courbes PR pour votre modèle. En outre, FiftyOne peut également enregistrer des statistiques précises telles que la précision et le nombre de faux positifs au niveau de l'échantillon, que vous pouvez

exploiter via son interface interactive pour explorer de manière interactive les forces et les faiblesses de vos modèles sur des échantillons de données individuels.

L'analyse au niveau de l'échantillon conduit souvent à des informations essentielles qui vous aideront à améliorer vos ensembles de données et vos modèles. Par exemple, l'affichage des échantillons avec le plus de prédictions faussement positives peut révéler des erreurs dans votre schéma d'annotation. Ou bien, l'affichage du groupe d'échantillons avec la précision la plus faible peut révéler des lacunes dans votre ensemble de données d'apprentissage que vous devez combler afin d'améliorer les performances de votre modèle. L'un des principaux objectifs de FiftyOne est de vous aider à découvrir ces informations sur vos données !

Métriques pour évaluer les détections d'objets

Nous avons défini les métriques nécessaires à l'évaluation de notre détection d'objet.

ious: Il s'agit d'un dictionnaire qui contient les IoUs (Intersection over Union) correspondant aux correspondances de détection et de vérité terrain pour chaque image. Les IoUs représentent le chevauchement entre une boîte de détection et la vérité terrain associée.

La métrique IoU (Intersection over Union) représente le ratio de l'intersection entre la prédiction et le ground truth, divisé par l'union entre ces deux. Un IoU de 1.0 indique que la prédiction correspond parfaitement au ground truth, tandis qu'un IoU de 0 indique qu'il n'y a pas de correspondance entre les deux.

mAP : Il s'agit d'un flottant qui représente la moyenne de la moyenne de précision moyenne (mAP) pour toutes les classes. La métrique mAP (Mean Average Precision) est une mesure couramment utilisée pour évaluer la précision des modèles de détection d'objets. Elle est calculée en moyennant les précisions à différents niveaux de rappel pour différentes valeurs de seuil de confiance. Un score de mAP élevé indique une meilleure précision de la détection.

precision: mesure la proportion de vrais positifs (VP) que le modèle a réussi à identifier parmi tous les positifs réels (VP + faux négatifs). $\text{Rappel} = \text{VP} / (\text{VP} + \text{faux négatifs})$

recall: mesure la proportion de vrais positifs (VP) parmi tous les exemples classés positifs (VP + faux positifs). $\text{Précision} = \text{VP} / (\text{VP} + \text{faux positifs})$

Le F-score (ou F1-score) : est une mesure qui combine la précision et le rappel en une seule valeur. Il est défini comme la moyenne harmonique de la précision et du rappel :

$$\text{F-score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

Le F-score permet de prendre en compte à la fois la précision et le rappel dans une seule mesure, ce qui est utile lorsque l'on souhaite évaluer un modèle en tenant compte à la fois des faux positifs et des faux négatifs.

La matrice de confusion : c'est le croisement entre la vérité terrain et ce qui a été prédit par le modèle. Il permet de voir comment le modèle se trompe en classant un objet dans une autre.

Toutes ces métriques seront obtenues grâce à la librairie fiftyone

Exploration du modèle pré-entraîné sur les images du dataset coco

Tout d'abord, faisons un tour sur le dataset coco avec la librairie fiftyone, car il offre plusieurs fonctionnalités et interface graphique puissantes conçues spécifiquement pour nous permettre d'explorer, d'analyser et de travailler avec vos données aussi facilement que possible.

Le code ci-dessous charge les données coco avec la librairie fiftyone de 4 classes choisies en prenons un échantillon dde 200 images des labels cat, dog, apple et orange

```
1 import fiftyone as fo
2 import fiftyone.zoo as foz
3
4 dataset = foz.load_zoo_dataset("coco-2017",
5                               split="validation",
6                               classes=["cat", "dog", "apple", "orange"],
7                               max_samples=200,
8                               dataset_dir = "/content/datasetcoco",
9                               shuffle=True)
```

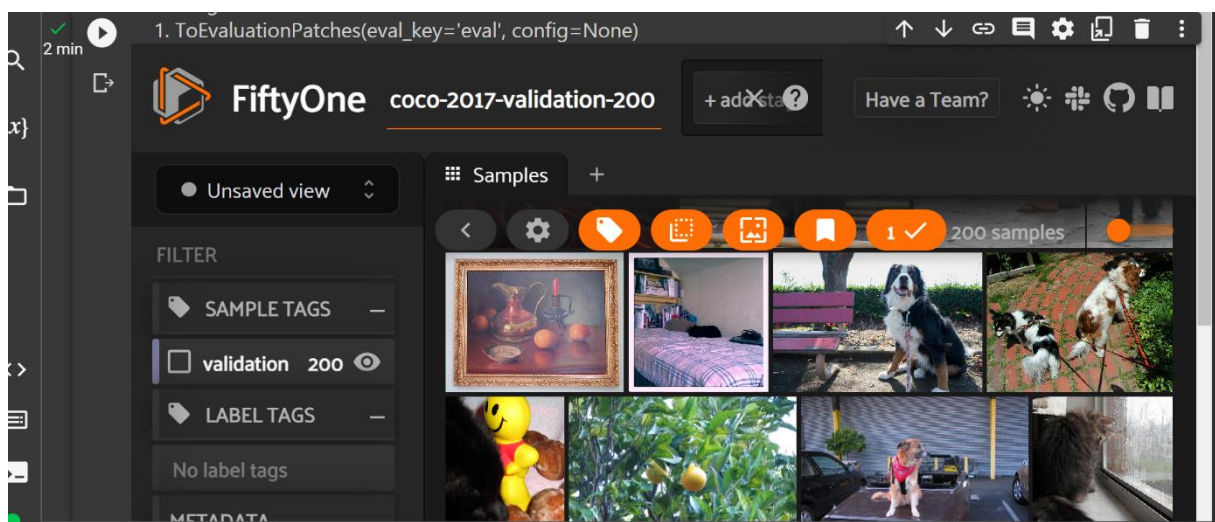
Dans fiftyone, il existe des méthodes qui permettent de mettre aisément en place une tâche de détection d'objet grâce. Le code ci-dessous permet d'utiliser le modèle pré-entraîné fasterrcnn dans fiftyone :

```
1 # chargement du modèle pré-entraîné
2 model = foz.load_zoo_model("faster-rcnn-resnet50-fpn-coco-torch")
3 dataset.apply_model(model, label_field="predictions")
4
5 # détection
6 results = dataset.evaluate_detections(
7     "predictions", gt_field="ground_truth", eval_key="eval", compute_mAP=True,
8     classwise=False)
9
10 # affichage map
11 print(results.mAP())
12 # ex: 0.358
```

Avec fiftyone, nous pouvons explorer les résultats via une interface interactive grâce à cette ligne de code.

```
1 import fiftyone as fo
2
3 #interface d'exploration des données
4 session = fo.launch_app(dataset)
5
```

Sur cette interface, nous pouvons explorer des images en faisant des filtres comme indiqué dans l'image ci-dessous. Par exemple, nous pourrions voir les images qui sont des faux positifs et leurs métadonnées ou des vrais positifs :









Commentaire :

Les images sur les données dataset coco sont bruitées, certains chats ou chiens sont cachés et le modèle n'arrive pas à le détecter.

Maintenant essayons de tester ce modèle sur nos propres images.

Nous avons collecté quelques images de chat, de chien, d' oranges et de pommes sur internet. On va utiliser ce modèle pré-entraînés et voir si le modèle détecte toujours bien sur les images qui ont été remplacées pour mettre en place cet algorithme ;

Voici un extrait des images collectées sur l'internet

			
chat	Pomme	chien	orange

Le code ci-dessous nous permettra de détecter les objets dans des images via le modèle pré-entraîné. Nous avons limité le modèle à détecter qu'aux quatre classes de nos jeux de données


```

1 def detection_object(img):
2     img = read_image(img)
3     # initialisation du modèle avec les poids pré-entraîné
4     weights = FasterRCNN_ResNet50_FPN_V2_Weights.COCO_V1.DEFAULT
5     model = fasterrcnn_resnet50_fpn_v2(weights=weights, box_score_thresh=0.9)
6     model.eval()
7     # inference des poids pré-entraîné sur nos images
8     preprocess = weights.transforms()
9     batch = [preprocess(img)]
10    #définissons nos classes
11    name_classe=["cat","dog","orange","apple"]
12    # application pour détecter les images
13    prediction = model(batch)[0]
14    labels = [weights.meta["categories"][i] for i in prediction["labels"]]
15    labels=[label for label in labels if label in name_classe]
16    box = draw_bounding_boxes(img, boxes=prediction["boxes"],
17                              labels=labels,
18                              colors="red",
19                              width=4, font_size=30)
20    im = to_pil_image(box.detach())
21    im.show()

```

Récupérons les fichiers nos images qui sont stockées dans le répertoire « Myimages » et détection

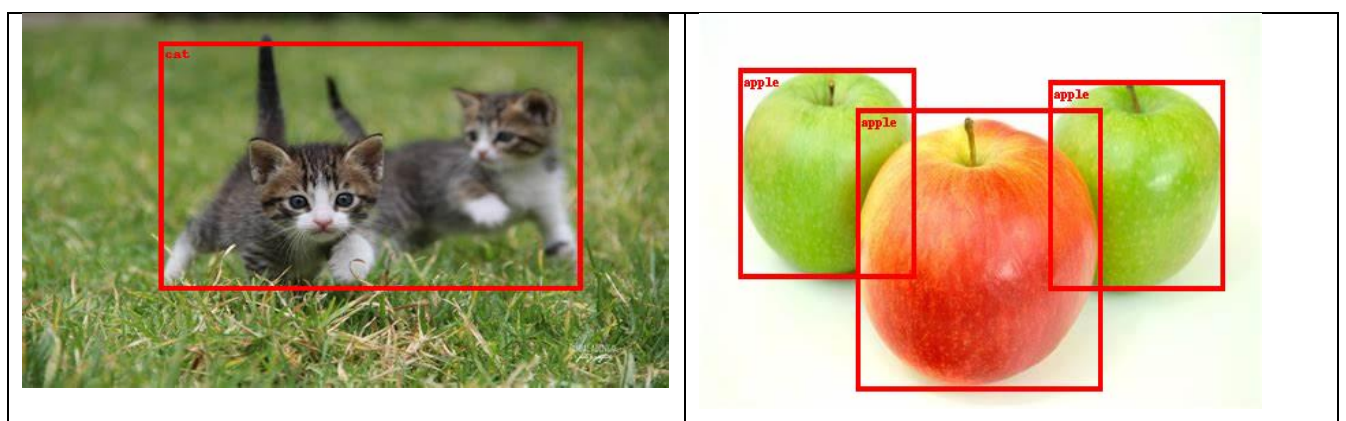
```

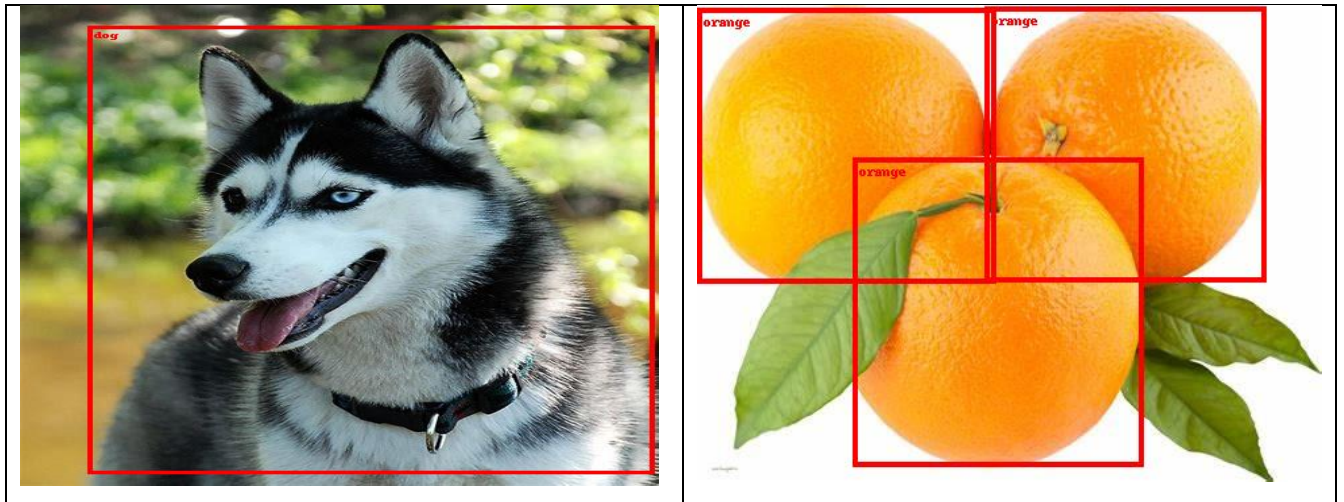
1 #utilisation de la fonction pour détecter les images
2 os.chdir("/content/My_images")
3 #chargement des images
4 path="/content/My_images"
5 url_img=[img for img in os.listdir(path)]
6 #détection
7 for img in url_img:
8     detection_object(img)

```

Faisons une boucle pour détecter nos 4 classes dans les images.

De façon qualitative, le modèle pré-entraîné arrive facilement à détecter nos quatre types d'objets. Comme le montre l'extrait des résultats ci-dessous :





Exploration du modèle pré-entraîné en changeant les paramètres (fine tuning)

Pour le fine tuning, nous avons juste restreint le modèle à 4 classes et explorer les résultats obtenus

Le code suivant nous montre comment garder que les 4 classes que nous voulons ré-entraîner

```

1 import fiftyone as fo
2 import fiftyone.zoo as foz
3 # objets à détecter
4 classes = ["dog", "cat", "apple", "orange"]
5 #initialisation du modèle pré-entraîné
6 model = foz.load_zoo_model("faster-rcnn-resnet50-fpn-coco-torch")
7
8 #résultats
9 results = dataset.evaluate_detections(
10 "predictions", gt_field="ground_truth", eval_key="eval_predictions", classwise=False, compute_mAP=True)
11

```

La librairie fiftyone permet d'explorer les résultats grâce aux attributs et méthodes de l'objet « results » avec la commande

```

1 # Afficher les attributs et les méthodes de l'objet "results"
2 dir(results)

```

Nous pourrions afficher le mAP du modèle et les métriques de chaque classe avec la méthode report

```

1 print(results.mAP0)
2 # ex: 0.358
0.32907682987926523

```

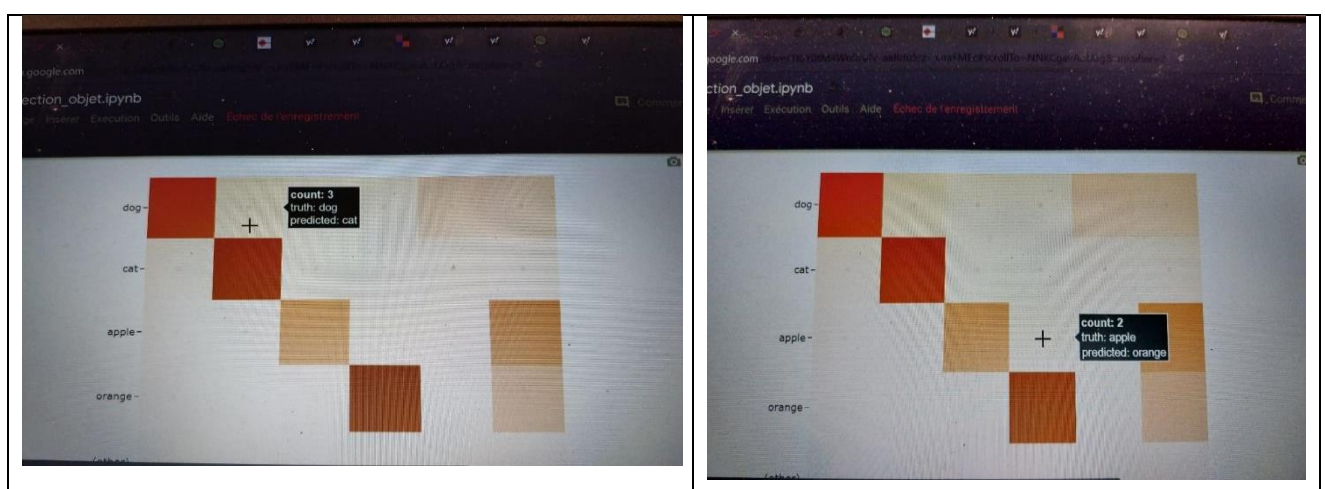

1 #évaluation des resultats				
2 res=results.print_report(classes=classes)				
	precision	recall	f1-score	support
dog	0.63	0.86	0.72	83
cat	0.63	0.92	0.75	90
apple	0.33	0.50	0.40	112
orange	0.45	0.80	0.58	142
micro avg	0.48	0.76	0.59	427
macro avg	0.51	0.77	0.61	427
weighted avg	0.49	0.76	0.59	427

Commentaire :

Le modèle a une mAP de .032 qui est faible. Le modèle a du mal à détecter les images. Cela peut être dû au fait que les objets sur les images sont tellement cachés. De même, la précision de chaque classe est faible, car le modèle n'arrive pas à détecter les vrais labels des objets.

Nous pouvons aussi afficher la matrice de confusion pour voir comment le modèle se trompe pour détecter les objets

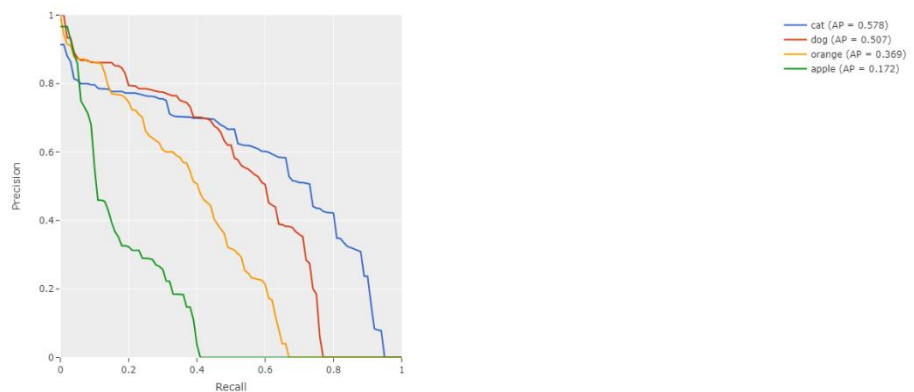
1 #matrice de confusion	
2 plot = results.plot_confusion_matrix(classes=classes)	
3 plot.show()	



Commentaire :

En explorant cette matrice, on voit que 3 chiens ont été mal prédit en le labellisant cat. Donc le modèle se trompe dans ce cas. Tout comme 2 pommes ont été mal prédite en le détectant comme orange. La matrice de confusion nous permet vraiment de comprendre dans quelles sont les classes où le modèle se trompe davantage et dans quelle classe ces objets ont été faussement classés.

Nous pouvons aussi calculer les courbes de précision moyenne (map) de chaque classe :



Conclusion

En définitive, le domaine de la vision par ordinateur, en particulier celui de la détection d'objets, sont des tâches qui ont déjà fait preuve de leur efficacité avec plusieurs librairies et une forte communauté qui travaillent dans ce domaine. Cependant, les améliorations restent toujours à faire, surtout lorsque nous avons des données bruitées qu'il faut bien traiter d'abord avant de les passer dans les modèles de détection d'objets.

Ce projet nous a aussi appris qu'il existe de nombreuses use case de vision par ordinateur dans plusieurs domaines notamment dans l'environnement, la santé, l'agriculture, etc.

Référence bibliographique

1- modeles fasterRCNN : [Fatsercnn-resnet](#)

2- [article pour apprendre fitfyone](#) : Article fiftyone