

Projet COTS

Rapport de Projet - Participation à un concours
Kaggle

Efkan TUREDI
27 February 2022

Introduction

Dans ce rapport, nous allons faire état de nos travaux et conclusions concernant le projet COTS. Ce rapport s'articulera en 4 parties:

1. Problématique
2. Revue des méthodes existantes
3. Dataset
4. Résultats et conclusions

Problématique

Nous avons choisi ce concours à cause de notre appétence pour les problématiques de computer vision. Dans le projet que nous avons choisi, nous devons réaliser un algorithme qui permet de reconnaître une race d'étoile de mer "COTS" dont la propagation déstabilise l'éco-système sous marin.

Nous disposons d'un dataset de photos provenant de 3 vidéos, qui nous sont soumis par les équipes de défense du corail sous marin en Australie.

Nous devons mettre en place une solution d'apprentissage automatique qui permet de faciliter le travail de ces équipes et permettre un diagnostic plus rapide qui permettrait de se prémunir plus rapidement contre les proliférations de COTS.

Revue des méthodes existantes

Revue des méthodes mises en place

Dans cette partie nous allons faire un tour rapide des deux familles d'algorithmes les plus connues dans ce domaine: la famille R-CNN et la famille YOLO

R-CNN: Développé dans un papier de 2014: <https://arxiv.org/abs/1311.2524>
La structure de cette méthode repose sur 3 modules qui s'enchaînent:

Module 1: Proposition de régions. Génère et extrait des catégories indépendantes des régions c-a-d des contours candidats.

Module 2: Extraction de caractéristiques. Extraire des features de chaque région candidat à l'aide d'un CNN.

Module 3: Classification. Classifier les features obtenues dans l'une des classes disponibles à l'aide d'un softmax, logit ou SVM par exemple

L'inconvénient majeur de cette méthode est qu'elle est très lente même avec de grosses ressources. Sans rentrer dans les détails des implémentations, des améliorations / variantes de cet algorithme ont été développées au fur et à mesure du temps. Nous mettons les références vers les papiers en question dans le domaine

Fast R-CNN: Développé dans un papier de 2015: <https://arxiv.org/abs/1504.08083>

Faster R-CNN: Développé dans un papier de 2016: <https://arxiv.org/abs/1506.01497>

La famille des algorithmes YOLO est la plus utilisée à l'heure actuelle, car elle permet de faire des itérations rapides de modèle et ne nécessite pas des énormes ressources. Cette famille est très utilisée pour les applications en IoT

YOLO: Développé par Joseph Redmon en 2015: <https://arxiv.org/abs/1506.02640>

Elle consiste à utiliser un seul et unique CNN de bout en bout , qui prend une photo en entrée, dessine les contours des objets, et s'occupe de mettre un label sur chaque conteneur obtenus.

Au fur et à mesure des années, YOLO a connu des améliorations dans le temps de calcul jusqu'à avoir des v4 et v3.

Pourquoi nous choisissons la famille YOLO

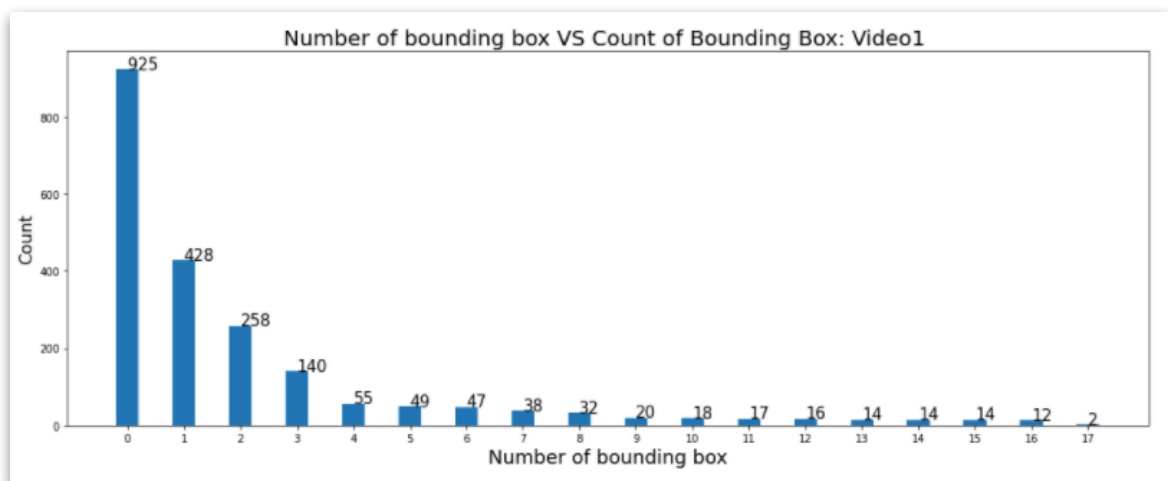
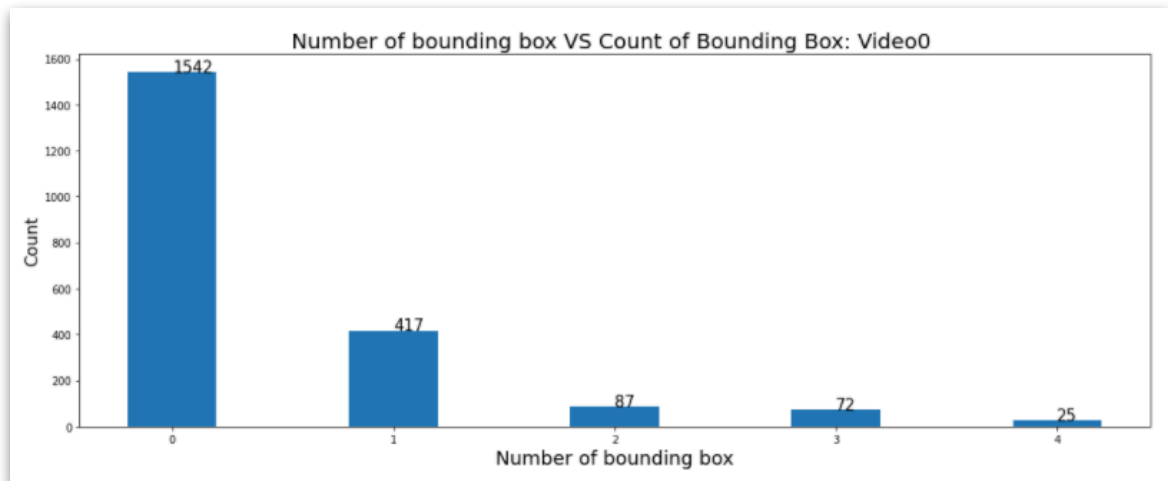
Cette décision est motivée par plusieurs facteurs:

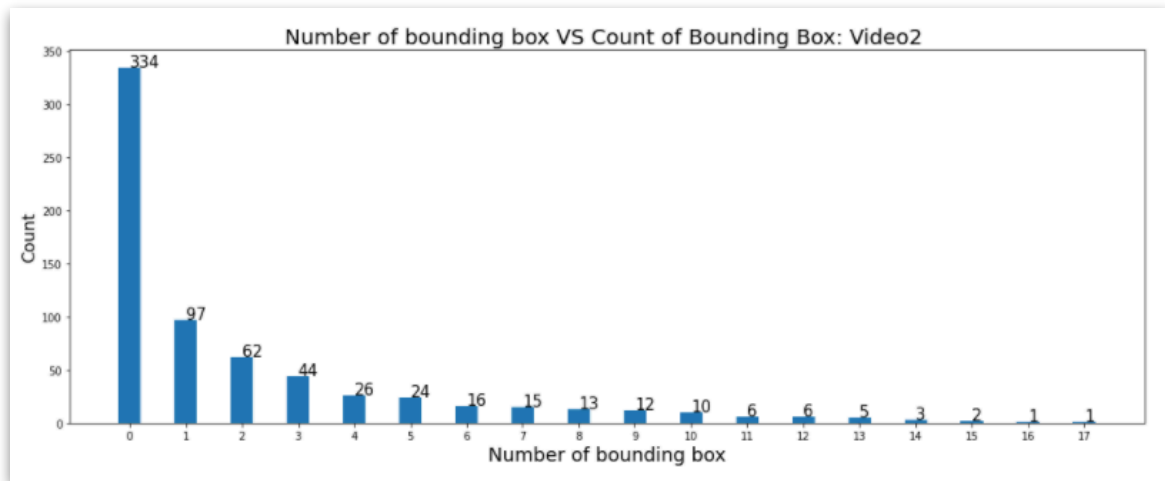
- Suivre les best practices mises en places par la communauté
- YOLO étant la famille la plus utilisée sur Kaggle, nous aurons beaucoup de Kernels à notre disposition
- Les méthodes YOLO sont plus rapide que les méthodes avec R-CNN

Nous allons notamment utiliser le framework Ultralytics qui met en place une implémentation de YOLOv5, ce qui nous permet d'avoir des outils clés en main prêts à être utilisés.

Dataset

Notre dataset contient des photos qui proviennent de 3 vidéos. Nous remarquons que les vidéos ne sont pas homogènes en termes de nombre d'identifications. C'est la conclusion des 3 graphs ci-dessous:

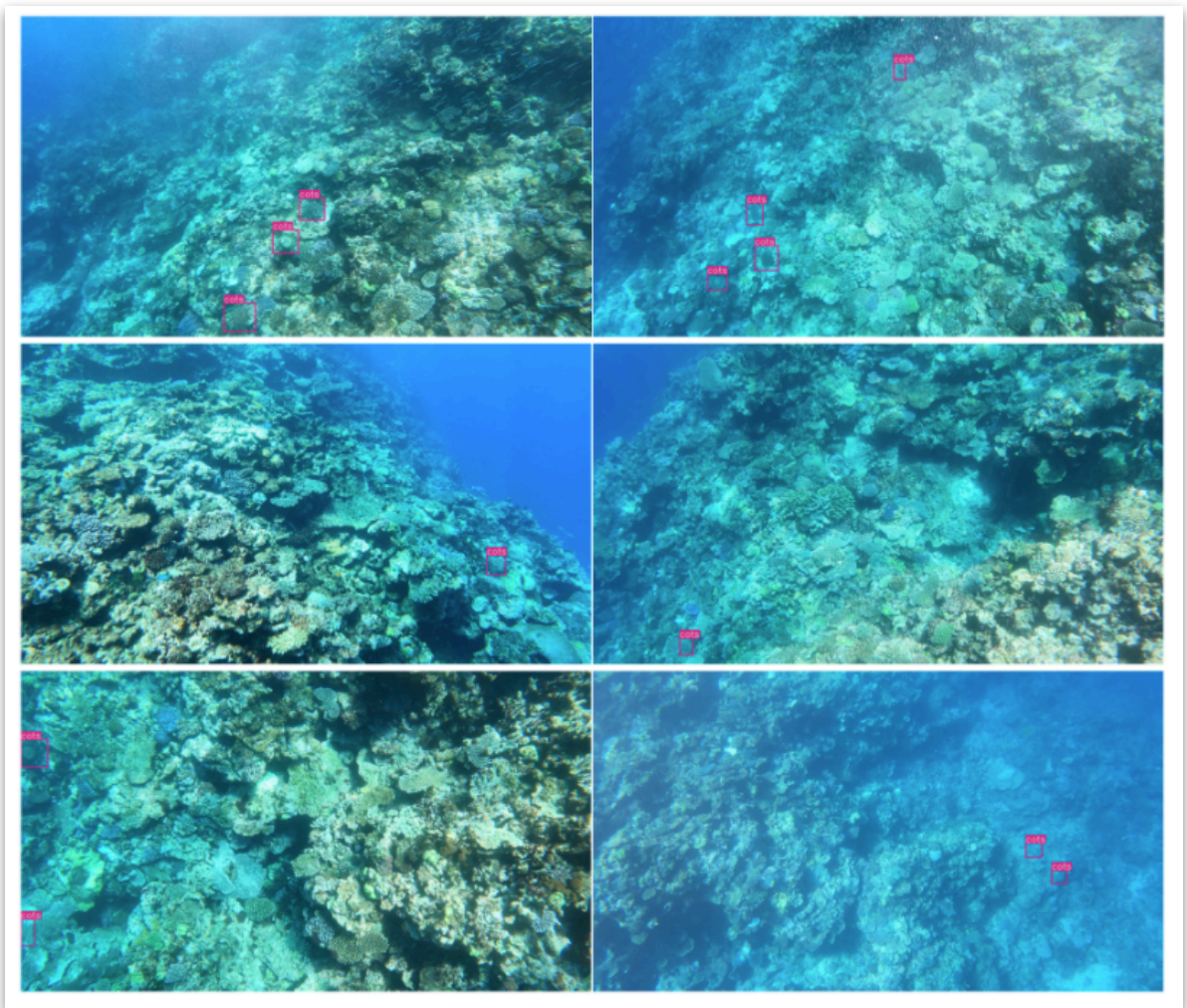




Notre base de données contient au total 23k images. Seul 5k images semblent contenir des COTS. Nous décidons donc retenir uniquement cette échantillon d'images afin d'accélérer les temps d'entraînement. Etant donné que nous avons un seul type d'objet à détecter, nous pensons que cela fait sens de prendre cette décision.

Un autre point important avant d'aller vers l'entraînement est le passage au format YOLO de notre base de données. En effet, les algorithmes YOLO nécessitent d'avoir les détections "ground truth" d'être fournies dans un format définis. Heureusement nous avons déjà des outils pré-existants qui vont nous permettre d'accélérer ce processus de mise au format.

Nos coordonnées ground truth se trouvent à l'heure actuelle au format CoCo, dont le nom fait allusion à une base de données d'objets très utilisé dans la communauté pour entraînement d'algorithmes de computer vision.



L'entraînement se fait de manière relativement simple, à l'instar des kernels Kaggle déjà présents.

A noter que nous avons pris la décision de diviser notre kernel en deux: un kernel d'entraînement et un kernel d'inférence. Cette décision est motivée par la volonté de limiter le temps d'exécution sur notre kernel d'inférence lors de la soumission de résultats. Cela nous permet aussi de pouvoir isoler le travail de d'optimisation des paramètres d'inférence.















Résultats

A noter que nous avons soumis deux types de résultats dans cet exercice. Dans notre première soumission, nous avons utilisés les poids calculés par le créateur du kernel dont nous nous sommes inspirés.

En effet, ces poids ont été entraînés en utilisant des plateformes cloud (AWS, GCP, Azure, etc...) permettent d'avoir accès à des cartes graphiques avec des ressources en mémoire RAM plus élevés que les cartes graphiques de Kaggle (16GB). Nous remarquons que nous atteignons rapidement les limites en ressources de mémoire de notre carte graphique.

Dans une notre deuxième lot de soumissions, nous utilisons nos propres poids entraînés via notre kernel d'entraînement. Nous remarquons que nos résultats sont légèrement inférieurs, toutes choses égales par ailleurs. Notre travail sera d'essayer de modifier les paramètres principaux pour essayer d'aller chercher de la performance.

A noter que nos performances sont mesurées avec le score F2, qui est une variante de notre score F1 qui donne plus d'importance au recall qu'à la précision. Ce choix semble logique vis à vis de notre objectif de protection des coraux. On préfère avoir des faux positifs plutôt que d'avoir des faux négatifs.

Submission and Description	Status	Private Score	Public Score	Use for Final Score
Great-Barrier-Reef: YOLOv5 [infer]  (version 7/7) 14 hours ago by Efkan Turedi Notebook Great-Barrier-Reef: YOLOv5 [infer]  Version 7	Succeeded 	0.562	0.571	<input type="checkbox"/>
Great-Barrier-Reef: YOLOv5 [infer]  (version 6/7) a day ago by Efkan Turedi Notebook Great-Barrier-Reef: YOLOv5 [infer]  Version 6	Succeeded 	0.577	0.500	<input type="checkbox"/>
Great-Barrier-Reef: YOLOv5 [infer]  (version 5/7) a day ago by Efkan Turedi Notebook Great-Barrier-Reef: YOLOv5 [infer]  Version 5	Succeeded 	0.582	0.592	<input type="checkbox"/>
Great-Barrier-Reef: YOLOv5 [infer]  (version 4/7) 2 days ago by Efkan Turedi Notebook Great-Barrier-Reef: YOLOv5 [infer]  Version 4	Succeeded 	0.313	0.477	<input type="checkbox"/>
Great-Barrier-Reef: YOLOv5 [infer]  (version 3/7) 17 days ago by Efkan Turedi Init test : YOLOv5 [infer]  Version 3	Succeeded	0.588	0.627	<input type="checkbox"/>

Annexes

Liens vers les kernels que nous avons utilisés:

Kernel 1: <https://www.kaggle.com/awsaf49/great-barrier-reef-yolov5-train>

Kernel 2: <https://www.kaggle.com/awsaf49/great-barrier-reef-yolov5-infer>

Kernel 3: <https://www.kaggle.com/soumya9977/learning-to-sea-underwater-img-enhancement-eda>