CentraleSupélec

OPENCLASSROOMS

Développez une preuve de concept

Adrian Rodriguez - Ingénieur Machine Learning

1. Thématique

Un rapport sur les erreurs de classification des races de chiens a été soumis à l'association de protection des animaux. Dans ce rapport, il est recommandé d'implémenter une solution de détection d'objet pour améliorer la qualité des photos. Cette solution doit isoler l'objet à classifier en procédant à un rognage.

- Dataset : Stanford Dogs Dataset
- Méthode baseline : Modèle Xception 2017, et Rapport de classification
- Méthode mise en oeuvre :
    - Etat de l'art
    - Phase 1 : Intégration et mesure du détecteur d'objet en production
    - Phase 2 : Si résultat de la phase 1 insuffisant, apprentissage du modèle existant avec photos recadrées et mesure des résultats.

2. Etat de l'art

J'ai procédé à un état de l'art pour connaître ce qu'il se fait de mieux en matière de détection d'objet fin 2020.
La figure 1 ci-dessous décrit simplement les performances des détecteurs d'objets existant.

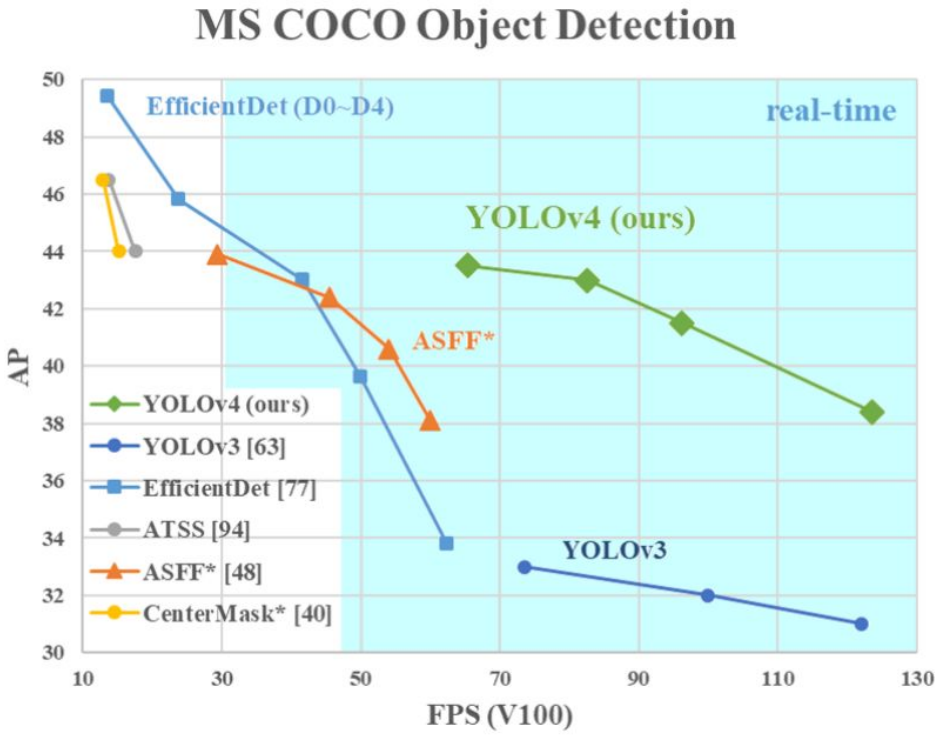


Figure 1 : Etat de l'art

Selon les auteurs de ce papier sorti le 23 avril 2020, YOLOv4 fonctionne deux fois plus vite qu'EfficientDet avec des performances comparables. Il améliore les résultats de son prédécesseur, YOLOv3, d'environ 10 %.

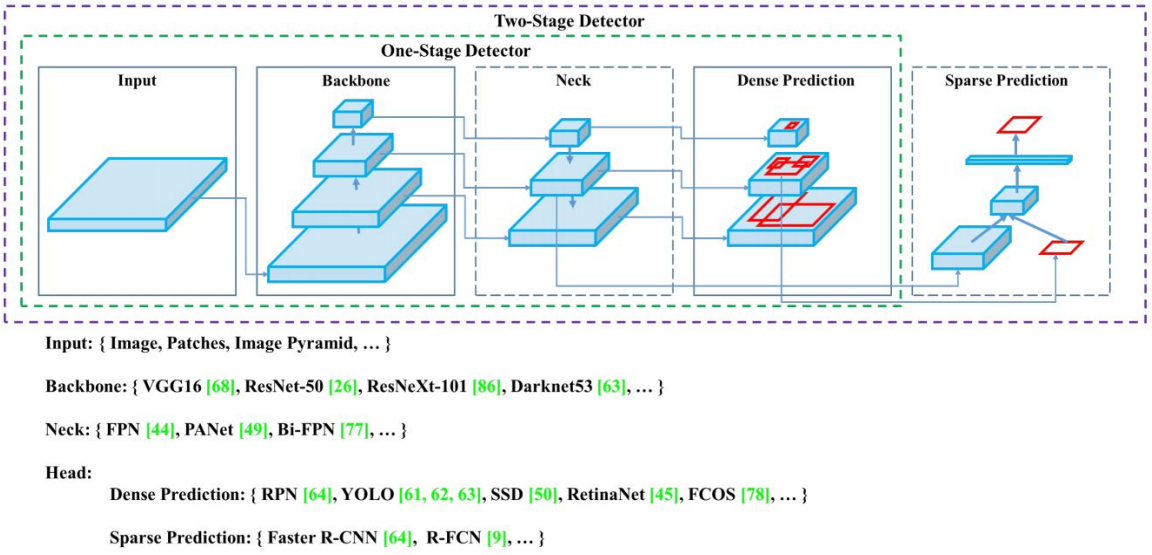


Figure 2 : Architecture de YOLOv4

YOLO dans sa version originale (décembre 2016) est le premier détecteur d'objets à combiner le problème de dessin de boîtes, et le problème de classification comme le fait actuellement Xception en production.

Dans sa version la plus récente, YOLOv4 (avril 2020) met à jour son réseau dorsal (backbone) avec CSPDarknet53, CSPResNext50, et EfficientNet-B3. Ces réseaux sont préformés sur imagenet, c'est à dire que ses poids sont déjà adaptés pour identifier les caractéristiques pertinentes d'une image.

YOLOv4 ajoute un module Neck, qui n'existait pas avant. C'est un module d'agrégation qui va mixer et combiner les features extraites du backbone avant de les envoyer dans les couches de classification ou de détections. YOLOv4 conserve les même couches de têtes que YOLOv3.

C'est avec YOLOv4 que je nettoierai les images et que j'isolerai les objets à identifier.

Après détection, il suffit d'extraire les coordonnées de l'objet à identifier afin de créer un nouvelle image propre.

Image 1 : Détection d'objet avec YOLOv4 sur une image extraite dataset

3. Phase 1 : Intégration en production

3.1. Implémentation

Actuellement, le bénévole de l'association charge une photo dans un programme qui lui retourne la race prédite. Avant de procéder aux opérations de classification, j'opère à une extraction de l'objet à identifier.

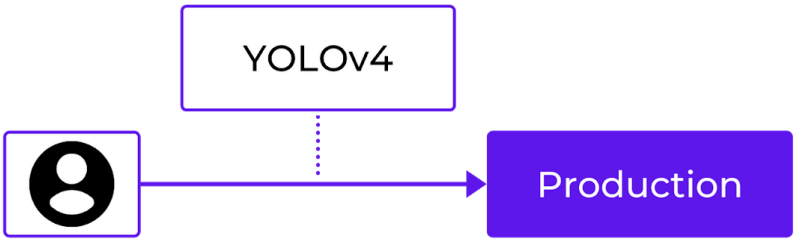


Figure 3 : Implémentation de YOLOv4 avant classification

3.2. Mesure des résultats

Comme mentionné dans le rapport initial, nous avons 2 types d'erreurs :

- Erreurs liées à la classification manuelle des photos,
- Erreurs liées à la présence d'objets autres que la race du chien à identifier.

C'est ce dernier cas que je cherche à résoudre avec le nouvel algorithme.

Sur les données de test, de 4116 individus, j'ai 517 individus en erreur de prédiction, soit une précision de 87.4 %.

En appliquant l'algorithme uniquement sur les 517 individus en erreur, je les réduis de 35 %. Ce qui à première vue peut paraître convaincant, il est nécessaire de généraliser l'algorithme YOLOv4 sur l'ensemble de l'échantillon de test et pas seulement sur les erreurs ; Il n' y a pas de différenciation en production.

Après généralisation sur le fichier de test, j'obtiens une précision 87.5 % avec implémentation de YOLOv4, soit seulement 0.1 % de plus que sans le nouvel algorithme. Les résultats attendus se situent aux alentours des 91 %.

Cela signifie que YOLOv4 a induit des erreurs qu'il n'y avait pas sans l'intervention de cet algorithme. Je vais tenter de comprendre comment cela est possible.

3.3. Compréhension des erreurs induites

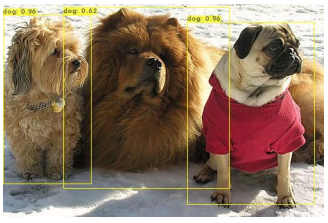
Cas	Photo originale	Résultat crop auto	Objets identifiés YOLOv4
1			
2			
3			

Tableau 1 : Cas d'erreurs induites

Dans le premier cas, le chien au premier plan est identifié comme un cheval, et le détecteur n'a retenu que le chien en arrière-plan. Même si pour l'oeil humain les deux animaux semble de la même race, il manque des éléments à la machine pour identifier la bonne race.
Dans le second cas, le grillage en premier plan fait croire à l'algorithme qu'il s'agit d'un mouton. De fait, aucun rognage n'est appliqué, car aucun chien identifié.
Dans le dernier cas, 3 races de chiens sont présentes et YOLOv4 extrait l'objet comme étant le chien le plus probable. Cependant, ce n'est pas le chien dont la race a été identifiée.

4. Phase 2 : Apprentissage d'un modèle

4.1. Implémentation avant apprentissage machine

Suite à ces résultats mitigés, j'entraîne le modèle à partir des photos nettoyées par YOLOv4.
L'objectif de cette étape est d'apprendre à la machine, les objets que le détecteur d'objet aura découvert par lui même. De cette manière, l'algorithme de classification sera plus en phase avec le travail de restitution de YOLOv4 en production.

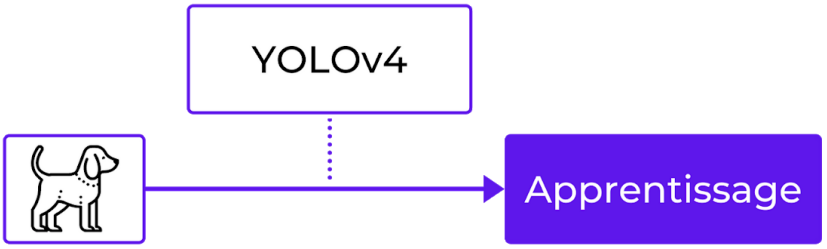


Figure 4 : Implémentation de YOLOv4 avant apprentissage

Les photos ont été recréées dans un nouveau dossier, puisque selon mes recherches, il n'est pas encore possible d'effectuer un rognage dans les paramètres d'un data generator, méthode utilisée pour la data augmentation en apprentissage machine.

4.2. Mesure des résultats

En procédant de cette manière, la précision de l'ensemble YOLOv4 + Xception atteint 91.3 %, soit 3.9 % d'amélioration de performance.

En observant le rapport de classification global par classes, nous trouvons Eskimo dog avec une précision de 49 % et en suivant, Miniature poodle avec une précision de 67 %. Cet écart significatif me laisse penser qu'il existe beaucoup d'erreurs de classification humaine en amont. D'après cette source, l'Eskimo dog n'est pas celui qu'on croit dans les données. Cela me laisse penser à une erreur majeure sur cette race, bien au delà de la simple erreur de classification que l'on peut rencontrer de manière éparse.

Image 2 :
Eskimo dog selon source externe

Image 3 :
Eskimo dog selon données

Au-delà de cette hausse significative de la performance et des restes d'erreurs de classification, toujours perfectibles, je mesure la performance de YOLOv4 dans sa capacité à détecter les bons objets d'une photo.

5. Evaluation de la performance de YOLOv4

5.1. Principe d'évaluation d'un détecteur d'objet

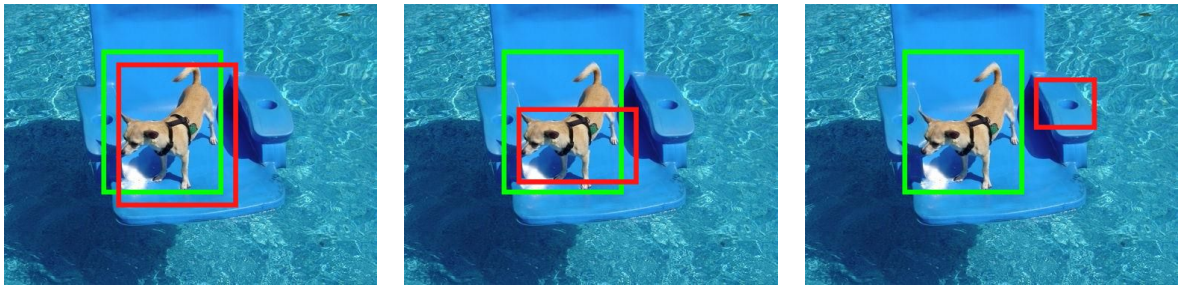


Image 2 : Boîtes délimitantes

Dans l'image ci-dessus, en vert la boîte annotée par les données et en rouge la boîte prédite par un détecteur.
Le principe d'évaluation d'un détecteur d'objet se base sur la comparaison de positionnement et de taille des boîtes délimitantes annotées aux boîtes délimitantes prédites.

5.2. Métrique custom

Pour comprendre les “points faibles” des détecteurs d’objets, j’ai dissocié dans un premier temps les métriques de positionnement et de taille des boîtes délimitantes.

Selon les résultats du tableau ci-dessous, le centre de l’objet est plutôt assez bien détecté. 79 % des objets détectés ont une erreur inférieure à 5 % sur le centre d’un objet. En revanche, seulement 49 % des objets détectés ont une erreur inférieure à 5 % sur le format de la boîte délimitante. Cette valeur remonte relativement assez lorsque je suis plus tolérant sur l’erreur.

Le détecteur aura donc tendance à trouver le bon objet, ce sera moins précis sur le contour de l’objet. C’est aussi un cas d’appréciation en l’humain, qui aura annoté le contour de la boîte, et la machine.

treshold	metric_center	metric_shape	metric_global
0.05	79.0	49.0	48.0
0.1	87.0	75.0	74.0
0.2	90.0	87.0	86.0
0.5	93.0	93.0	92.0
1.0	100.0	100.0	100.0

Tableau 1 : Métriques custom

5.3. Métrique Intersection over Union (IoU)

J’ai ensuite appliqué une métrique plus commune pour l’évaluation de la détection d’objets; Intersection over union. Cette technique consiste à calculer le rapport entre la surface de chevauchement et la surface d’ensemble entre les boîtes délimitantes, comme l’illustre la figure ci-contre.

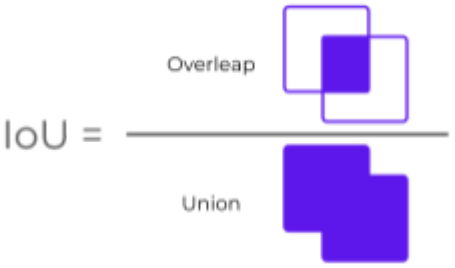


Figure 5 : Intersection over Union

Les résultats figurent dans le tableau 2 suivant :

treshold	metric_iou
0.05	12.0
0.1	46.0
0.2	79.0
0.5	89.0
1.0	97.0

Tableau 2 : Métrique IoU

Dans ce cas, seulement 12 % des prédictions présentent une erreur inférieure à 5 % et ce chiffre monte à 46 % pour les erreurs une erreur inférieure à 20 %.

YOLOv4 présente une précision moyenne de 78.7 % sur l'ensemble du jeu de données.

6. Perspectives

Pour aller plus loin dans la résolution de la problématique, nous pourrions aborder les points suivants :

- Toujours apporter des photos propres (à minima, 1 race par photo)
- Ré-entraîner les poids de YOLOv4 à notre problématique (uniquement des chiens à détecter, sur les photos de l'asso).
- Tester le classifieur YOLOv4 au lieu de Xception, qui est un ensemble de modèle.
- Eventuellement tester un pipeline détecteur/classifieur dans YOLOv4.