# Projet 8 – Proof of Concept Amélioration de l'indexation automatique d'images avec Mask R-CNN

Parcours Data Scientist

Julien Alvarez

14 Novembre 2019

# Table des matières



# Table des illustrations

# 1 Présentation du projet

Nous avons étudié dans le projet 7 comment indexer automatiquement des images au moyen d'un Réseau de Neurones Convolutif (CNN). Nous allons ici essayer d'améliorer les performances obtenues en prétraitant les images. L'idée de départ de cette expérimentation est que l'on va "focaliser" l'apprentissage de notre modèle sur le chien que l'on cherche à classer, en supprimant de l'image tous les éléments que l'on considère comme parasites : personnes, lieu où se situe le canidé, autres animaux, etc…

La méthode choisie, elle-même basée sur des réseaux de neurones, vise à détecter et classifier les objets présents sur une image. Il s'agit d'une méthode de segmentation d'instance, appelée Mask R-CNN. Nous étudierons dans le présent document le principe de cet algorithme en suivant les évolutions qui ont mené à sa conception, et commenterons sa mise en œuvre sur le projet de classification de races de chien qui nous servira de baseline pour juger les performances obtenues.

# 2 Principe du Mask R-CNN

Le Mask R-CNN est issu de l'évolution de modèles de segmentation d'instance. Voici une forme pseudo-chronologique des méthodes qui y ont abouti.

## 2.1 Region Proposal Network (RPN)

Le réseau de proposition de région (RPN) est un réseau de neurones utilisé pour décider «où» regarder afin de réduire les exigences de calcul du processus d'inférence global. Le RPN analyse rapidement chaque emplacement (en balayant l'image d'entrée avec des fenêtres de différentes dimensions) afin de déterminer si un traitement supplémentaire doit être effectué dans une région donnée. Pour ce faire, il édite k propositions de boîtes de sélection, chacune avec 2 scores représentant la probabilité d'objet ou non à chaque emplacement. Une fonction d'activation (softmax) permet de déterminer si une région est conservée ou non.

## 2.2 Region-based CNN (R-CNN)

Le Réseau de neurones de convolution par région (R-CNN) est utilisé pour réaliser une détection d'objet. Il définit une limite autour de chaque objet détecté dans l'image traitée. Pour ce faire, l'algorithme procède en deux étapes :

1. Étape de proposition de région (RPN),
2. Étape de classification. L'étape de classification consiste en l'extraction de vecteurs de caractéristiques et d'un ensemble de SVM linéaires.

Pour résoudre le problème de la sélection d'un grand nombre de régions, une recherche sélective est utilisée pour extraire k régions de l'image (soit k propositions de région). Cela permet, au lieu d'essayer de classer un grand nombre de régions, de plus simplement travailler avec k régions. L'algorithme de recherche sélective est exécuté suivant les étapes:

1. On génère une sous-segmentation initiale (plusieurs régions candidates),
2. Un algorithme combine de manière récursive des régions similaires,
3. On utilise les régions générées pour produire les propositions de régions finales.

Ces régions proposées sont ensuite introduites dans le réseau de neurones convolutif et produisent en sortie un vecteur de caractéristiques (un vecteur pour chaque région), qui est ensuite utilisé en tant qu'entrée dans l'ensemble de SVM produisant une étiquette de classe.

L'algorithme prédit également quatre valeurs de décalage pour augmenter la précision du cadre de sélection.

Cette approche vise donc à déterminer un nombre raisonnable de régions (objet) candidates, et évaluer des réseaux de convolution indépendamment sur chaque RoI (Région of Interest).

Le principal problème de R-CNN est qu'il nécessite un temps d'apprentissage élevé.

## 2.3 Fast R-CNN

Afin d'améliorer R-CNN, Fast R-CNN utilise également l'algorithme de recherche sélective mais résout le problème de la lenteur de R-CNN en partageant le calcul des couches de convolution entre différentes propositions de régions.

L'image est donnée en entrée au CNN qui génère une carte de caractéristiques convolutionnelle en sortie.

Les dimensions des régions proposées (RoI, pour Région of Interest) sont variées. Il est donc compliqué de pouvoir les utiliser telles quelles afin de déterminer celle que l'on souhaite conserver. C'est pourquoi il est d'usage d'utiliser une couche appelée RoIPooling. Cette opération consiste à effectuer un Max Pooling sur chacune des dimensions de RoI. Ainsi, si on détermine k dimensions différentes, on obtient un ensemble de valeurs de dimension k.

Les couches utilisées pour mapper la carte des caractéristiques de taille fixe sur un vecteur de caractéristiques sont de type Fully Connected (FC). Enfin une couche softmax prédit une classe pour la région proposée et une régression donne des valeurs de décalage pour le cadre de sélection.

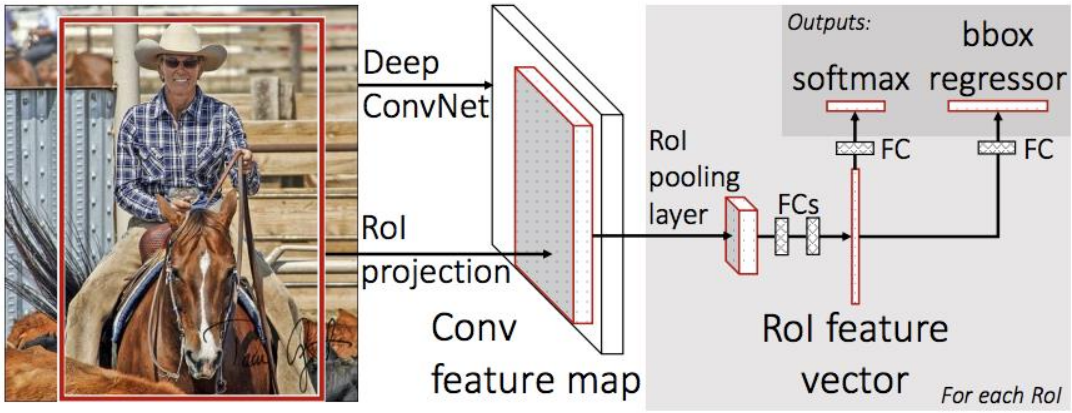


FIGURE 1- FRAMEWORK DU FAST R-CNN

## 2.4 Faster R-CNN

Faster R-CNN comporte un composant supplémentaire après la dernière couche convolutive appelée réseau de proposition de région (RPN).

Dans cette méthode, l'image est donnée en entrée au CNN qui fournit une carte de caractéristiques convolutionnelle. Au lieu d'utiliser un algorithme de recherche sélective sur la carte de caractéristiques, un RPN est utilisé pour prédire les propositions de région. Les propositions de région sont ensuite remodelées à l'aide d'une couche de regroupement RoI,

qui est ensuite utilisée pour prédire la classe de la région proposée et les valeurs de décalage pour les cadres de sélection.

Faster R- CNN a l'avantage d'être flexible et robuste aux améliorations, ce qui en fait un framework majeur sur de nombreux critères.

- Deux niveaux:
    - RPN propose des cadres aux objets candidats
    - Fast R-CNN extrait les features de chaque cadre avec RoIPooling et réalise une classification et la régression du bounding-box

  Les features utilisées par les deux niveaux peuvent être partagées pour une inférence plus rapide

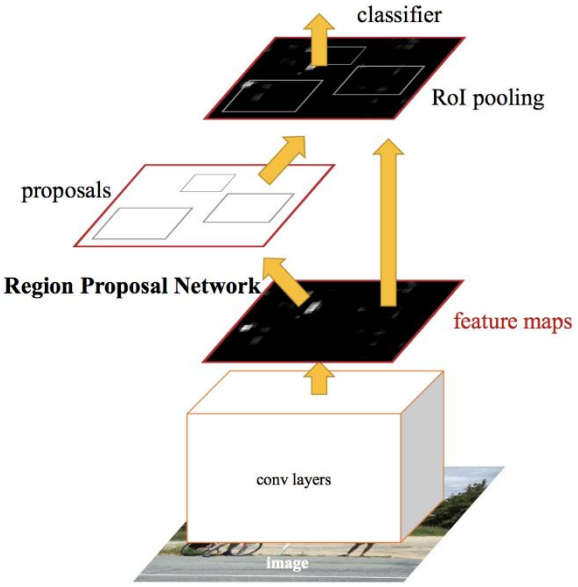


FIGURE 2- FRAMEWORK DU FASTER R-CNN

## 2.5 Mask R-CNN

Le masque R-CNN est une technique de segmentation d'instance qui localise chaque pixel de chaque objet de l'image, au lieu des cadres de sélection. Il comporte deux étapes : les propositions de région, puis la classification des propositions et la génération de cadres de sélection et de masques. Pour ce faire, il utilise un réseau supplémentaire entièrement convolutionnel au-dessus d'une carte de caractéristiques basée sur un CNN avec en entrée une carte de caractéristiques et fournit en sortie une matrice avec des 1 à tous les emplacements où le pixel appartient à l'objet et 0 ailleurs.

Cette approche détecte donc les objets d'une image, tout en générant un masque de segmentation pour chaque instance. La méthode reprend le Faster R-CNN en y ajoutant une branche pour prédire le masque d'un objet en parallèle à la branche existante qui réalise la classification et la reconnaissance de la zone de délimitation (Bounding box).

La branche dédiée au masque est un FCN (Fully Convolutional Network) appliqué à chaque RoI qui prédit un masque de segmentation de façon pixel par pixel.

Les auteurs du Mask R-CNN ont déterminé que Faster R-CNN n'était pas conçu de manière à réaliser un alignement au pixel près des entrées et sorties du réseau. Mask R-CNN introduit par conséquent la couche RoIAlign (à la place du RoI Pooling) dont l'objectif est de corriger ce décalage, en conservant les positions exactes des instances.

Un autre aspect important de la méthode repose sur l'indépendance entre le masque et la classification. On prédit un masque binaire de chaque classe indépendamment, sans compétition entre les classes, et on s'appuie sur la branche de classification du RoI pour prédire la catégorie.

J'ai choisi d'utilser le Mask R-CNN car d'après [1], les résultats obtenus par la méthode dépassent ceux des modèles ayant remporté des compétitions de détection d'images et de segmentation d'objet.

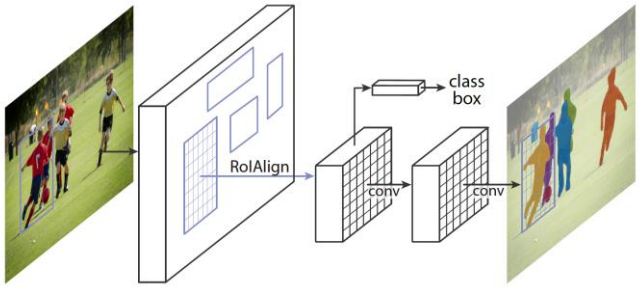


FIGURE 3- FRAMEWORK DU MASK R-CNN

# 3 Mise en œuvre

## 3.1 Code du Mask R-CNN utilisé

Nous utilisons le code développé par Matterport, et disponible sur Github [5]. Le modèle disponible est pré-entrainé sur le jeu de données MS COCO (Common Objects in COntext - http://cocodataset.org/#home), et permet de détecter un panel varié de classes d'objet (personnes, batiments, bornes incendie, chien, chat, vache, etc..).

Nous l'utilisons tel quel, et effectuons les traitements simples décrits ci-après afin d'utiliser les résultats qui nous intéressent.

## 3.2 Séquence de la modélisation

Nous avons un modèle de classification déjà mis en œuvre sur le jeu d'images d'origine. Un moyen simple de constater l'influence sur les performances de la réduction des images à l'objet que l'on cherche à classifier, est d'insérer dans l'algorithme existant une étape de prétraitement des images avant leur utilisation pour entrainer notre modèle.

Nous allons donc appliquer Mask R-CNN à chacune des images, et enregistrer le résultat dans des jeux de données distincts.

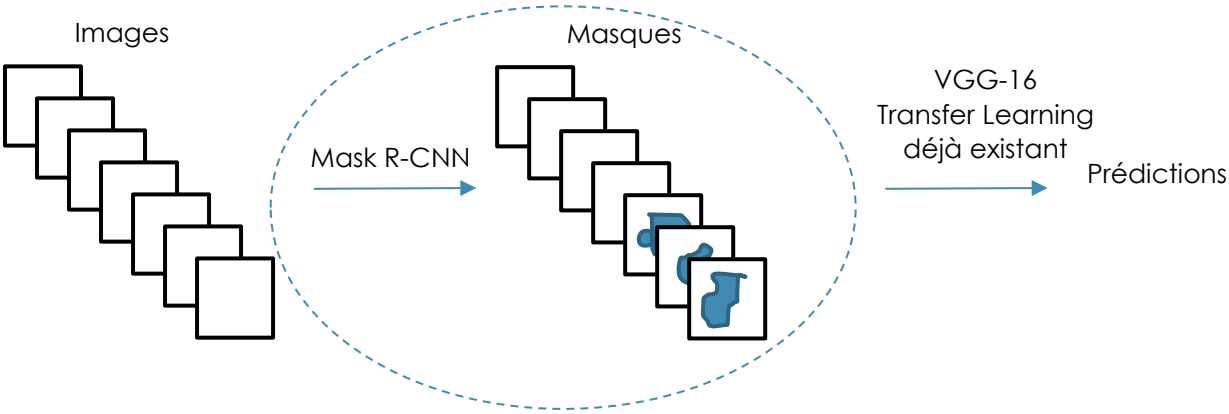


FIGURE 4- INSERTION DE LA DÉTECTION D'OBJET DANS LE MODÈLE D'INDEXATION EXISTANT

## 3.3 Détection de chien dans une image

L'algorithme Mask R-CNN détecte et classifie des instances d'objets prédéfinis dans un jeu d'entrainement (Personne, automobile, chat, chien, borne incendie, etc...). Or nous souhaitons, dans notre cas, effectuer une classification de races de chiens. Il est par conséquent nécessaire de filtrer les résultats selon la classe de l'instance détectée. Après quelques essais, nous avons opté pour le fait de conserver les classes "animales", afin de conserver un résultat, même lorsque le chien a été labélisé "chat". De plus, les images du jeu de données sont centrées sur un chien à classifier, mais il y a parfois plusieurs chiens sur une même image. Nous avons donc décidé de conserver le chien ayant la plus grande probabilité en sortie du Mask R-CNN.

Lorsque le Mask R-CNN ne détecte pas de chien sur une image, deux cas de figure sont possibles :

- Un autre animal est détecté : on traite alors l'objet comme si c'était un chien (fig. 5). C'est le cas pour 4,2 % des images traitées.

- Aucun objet de type animal n'est détecté : on conserve l'image entière(fig. 6). C'est le cas pour 2,5 % des images traitées.

Figure 5- Chien classé en tant que chat

Figure 6- Livre détecté mais pas de chien !

Nous utilisons les sorties de l'algorithme du Mask R-CNN pour sauvegarder plusieurs versions de chaque image (voir fig. x): une version rognée selon le cadre correspondant à l'instance détectée, une autre avec le masque (on met tous les autres pixels à 0), et une dernière en combinant les deux (cadre + masque).

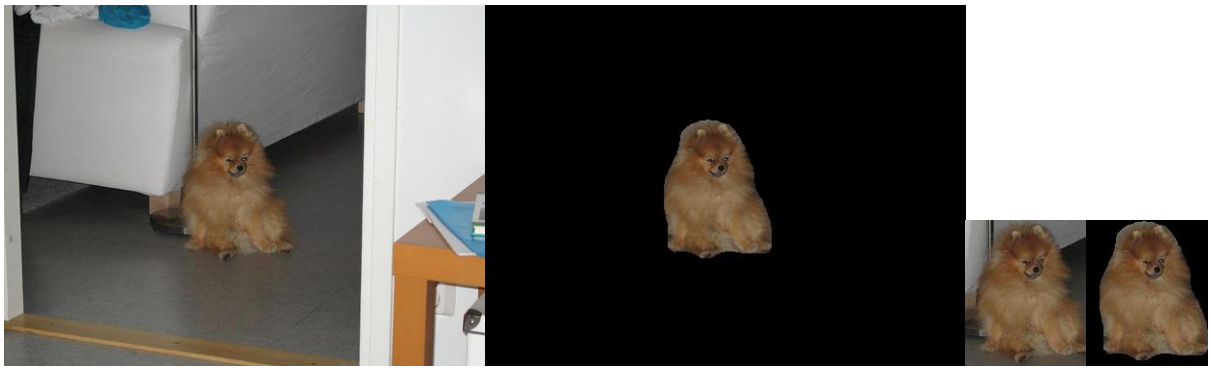

Figure 7- Image prétraitée avec les sorties du Mask R-CNN (originale, masque, box, masque + box)

## 3.4 Remarques sur les résultats du Mask R-CNN

Outre le fait que certains chiens ne sont pas détectés, ou détectés en tant qu'autre animal, on constate que le masque obtenu n'est pas toujours fidèle à l'animal. Il a en effet parfois tendance à être plus petit que le chien, et l'application du masque nous fait alors perdre certains éléments dont on pourrait s'attendre à ce qu'ils permettent de caractériser une race (extrémités des oreilles, par exemple, tel que l'image de droite de la Figure 8).

Cependant, lorsque l'on parcourt les images traitées, on s'aperçoit que les masques ont de manière générale un rendu très précis. (image de gauche sur la Figure 8)

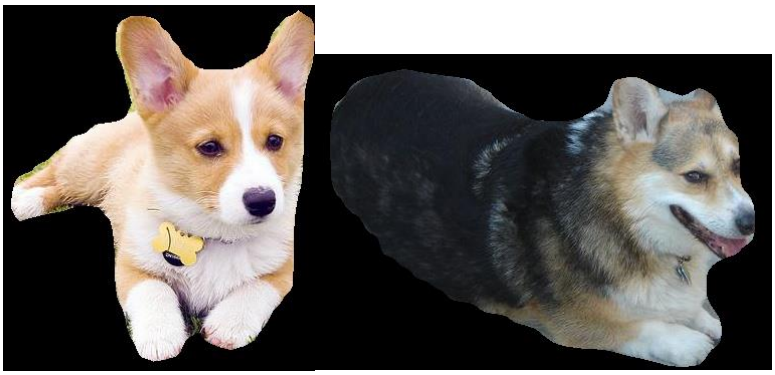

Figure 8- Certains masques, comme ici à droite, sont un peu "serrés"

# 4 Protocole expérimental

## 4.1 Méthode

Nous souhaitons vérifier l'influence éventuelle sur les performances de la classification de l'utilisation du Mask R-CNN. Nous allons donc pour cela dans un premier temps réaliser plusieurs jeux de données d'images avec des prétraitements différents :

- Images d'origine,
- Images recadrées suivant l'ancrage déterminé par l'algorithme Mask R-CNN,
- Images avec les dimensions d'origine, sur lesquelles on applique le masque issu de l'algorithme,
- Images sur lesquelles on applique le masque et redimensionnées selon l'ancrage de l'objet détecté.

Nous allons ensuite entrainer le même modèle, avec les mêmes caractéristiques d'apprentissage, sur ces différents jeux de données, préalablement réparties à l'identique selon des jeux d'entrainements, validation et test.

## 4.2 Observation des résultats

Les performances obtenues sont les suivantes :

| Prétraitement | Aucun | Bounding box | mask | mask + box |
|---|---|---|---|---|
| 3 races | 87,4 % | 91,6 % | 91,6 % | 92,6 % |
| 6 races | 85,2 % | 90,1 % | 86,2 % | 87,2 % |

On constate que, comme on pouvait l'espérer, le fait de restreindre l'information contenue dans une image à l'objet que l'on souhaite classifier améliore les performances. Le fait que l'application du masque n'apporte pas d'amélioration par rapport au rognage selon le cadre définit autour de l'instance détectée peut s'expliquer par le fait que la modélisation redimensionne toutes les images à une taille de 224x224 pixels. Par conséquent appliquer le masque sans recadrage entraine une réduction de l'image plus importante que si elle est rognée. Le fait d'appliquer le masque et de recadrer l'image donne de meilleurs résultats, car cela permet de diminuer le facteur de réduction appliqué à l'image en entrée du réseau de classification.

Le masque, même s'il épouse le plus souvent l'objet de façon relativement juste, manque parfois de précision tel que remarqué précédemment et ne contient pas toujours le contour exact de l'objet. Il serait intéressant de réaliser des essais supplémentaires, dans lesquels on pourrait agrandir le masque par homothétie, afin de voir si cela permettrait d'améliorer encore

les résultats. Le rapport de l'homothétie pourrait alors être ajusté afin d'améliorer le score de la prédiction.

# 5 Conclusion

Nous avons pu vérifier que le fait de supprimer d'une image les pixels qui n'appartiennent pas à l'objet que l'on souhaite classifier permettent d'améliorer les performances de notre modélisation. Et ce de façon relativement simple, puisque nous avons pu (dans notre cas) utiliser un modèle déjà entrainé.

Les résultats obtenus permettent également de supposer des pistes supplémentaires d'amélioration, par exemple en ajustant la façon dont on modifie les images avant la classification.

# 6 Références


[1] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, Xiangyu Zhang. Bounding Box Regression with Uncertainty for Accurate Object Detection (arXiv:1809.08545v3).

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN (arXiv:1703.06870v3).

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (arXiv:1506.01497v3)

[4] Priya Dwivedi. Building a Custom Mask RCNN model with Tensorflow Object Detection. (https://towardsdatascience.com/building-a-custom-mask-rcnn-model-with-tensorflow-object-detection-952f5b0c7ab4)

[5] Matterport. Github Mask R-CNN for Object Detection and Segmentation. (https://github.com/matterport/Mask_RCNN)

[6] Aditi Mittal. Instance segmentation using Mask R-CNN. (https://towardsdatascience.com/instance-segmentation-using-mask-r-cnn-7f77bdd46abd)