Université Sorbonne
Campus Pierre et Marie Curie Jussieu

Filière IMA
PIMA
Rapport final

Path Aggregation Network

Etudiant :
Hakim Amraoui

Encadrant :
Vannary Mesas-Yedad

22 mai 2022

SORBONNE UNIVERSITÉ

UPMC SORBONNE UNIVERSITÉS

Table des matières

Table des figures

1 Introduction

La segmentation d'image fait partie des plus importants enjeux de la vision par ordinateur. Ce processus est séparé en deux parties principales, d'une part la segmentation sémantique et d'autre part la segmentation d'instance. La première réfère au fait de regrouper des pixels d'une image en classes ou objets concrets tels qu'une voiture ou le fond de l'image. La deuxième partie quant à elle, consiste à identifier et classifier divers objets d'une image. C'est une des plus complexes tâches pour ce qui est de la détection d'objet.

On peut trouver des applications dans de nombreux domaines, tels que la vision des véhicules autonomes ou des analyses médicales.

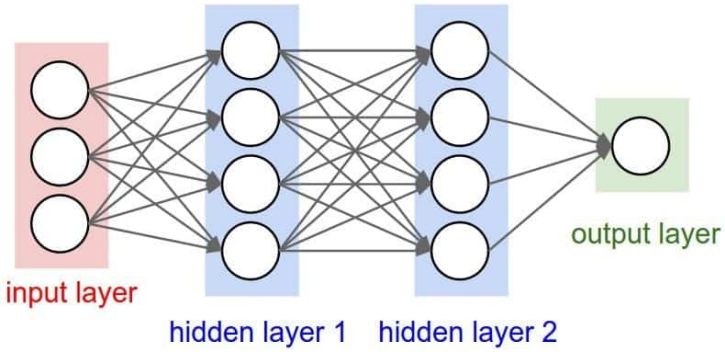


FIGURE 1 – Schéma d'un réseau de neurones

Les réseaux de neurones sont la technologie la plus utilisée dans ce domaine. Un réseau prend en entré une image convertit en liste pour chaque pixel, c'est l'input layer, on applique ensuite une série d'opérations sur ses entrées, ce sont les hidden layers, et pour finir, l'opération finale regroupe tous les neurones de la dernière couche en un neurone, sa sortie est une valeur entre 0 et 1, ce qui représente sa probabilité d'appartenance à la classe recherchée. Dans le cas du multi classe, il y aura ainsi un neurone de sortie par classe.

Ce type de réseau pose un problème, les pixels adjacents ne sont pas considérés les uns par rapport aux autres, mais comme des pixels individuels. Le motif de connexion entre les neurones dans un réseau de neurones convolutifs, ou CNN, est inspiré par le cortex visuel des animaux. Les neurones du cerveau agissant sur la vision se chevauchent et s'influe les uns les autres.

L'architecture d'un CNN [1] consiste en 3 couches principales :

1. **La couche convolutionnelle :** La couche convolutionnelle utilise des filtres et des noyaux pour scanner l'entrée $\boldsymbol{I}$ en appliquant des opérations de convolution. Ses paramètres, tels que la taille du filtre $\boldsymbol{T}$ ou le stride $\boldsymbol{S}$, sont ajustables. Avec cette opération, une carte de caractéristiques (feature map) est extraite.
2. **Couche de pooling :** Cette opération de sous-échantillonage vient souvent après une couche convolutionnelle. Le max pooling et l'average pooling sont les types de pooling les plus populaires, les valeurs max, respectivement moyennes, sont prises.
3. **Couche de fully connected :** L'entree $\boldsymbol{I}$ de cette couche est aplatie en une liste où chaque entrée est connectée à tous les neurones. Cette couche est principalement la composante finale des architectures de CNN et sert à optimiser les scores comme les scores de classification.

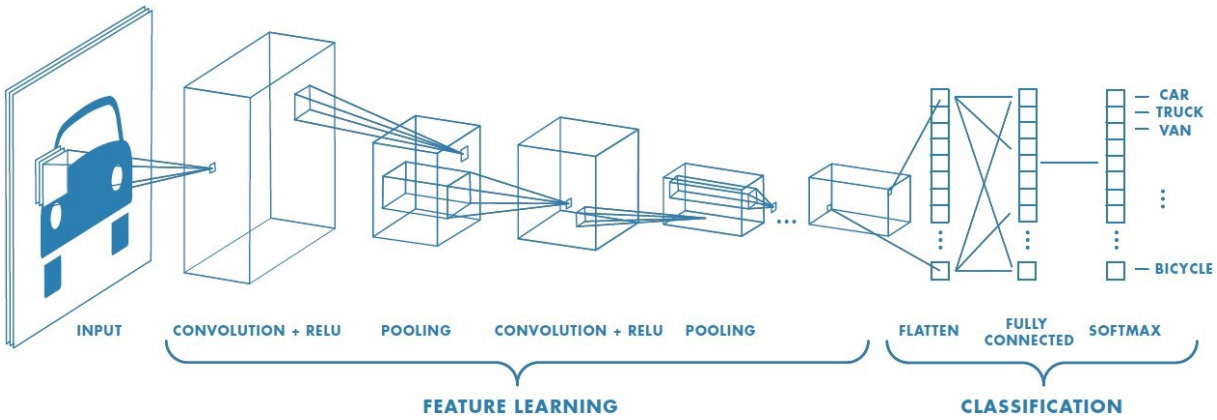


FIGURE 2 – Concept d'un réseau de neurones convolutif

De nombreux types de réseaux dérivent des CNN que nous allons voir plus tard.

Pour entraîner ces réseaux, une base de données est nécessaire, certaines très utilisées sont MNIST dataset pour les chiffres manuscrits ou CIFAR10 pour des objets communs.

Mon but est de m'inspirer de PANet [2] pour implémenter un modèle réalisant de la segmentation d'instance sur des images cellulaires issues de laboratoire.

2 Les différentes architectures

Dans le domaine de la détection d'objet, un algorithme est considéré comme le meilleur, YOLO, (You Only Look Once). Dans ses versions précédentes, YOLO utilisait les FPN (Feature Pyramid Network), et à partir de la version 4 a changé d'approche en utilisant Path Aggregation Network.

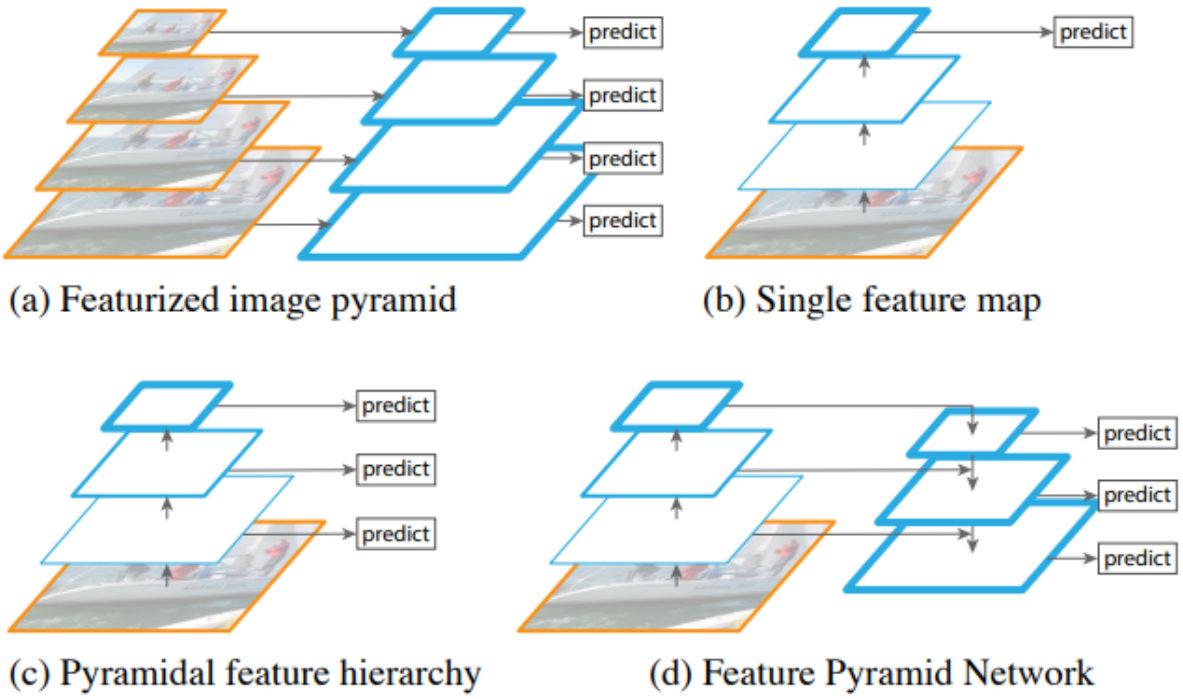


FIGURE 3 – Schéma des évolutions apportées par les FPN

(a) La méthode utilisée de base, chacune des caractéristiques sont extraites à différentes échelles indépendamment les unes des autres, ce qui est très lent. Ce modèle permet cependant d'être invariant à l'échelle. (b) Afin d'améliorer la rapidité, le calcul d'une couche utilise les informations issues des couches inférieurs et non l'image de base à différentes échelles. (c) Les couches permettent maintenant de prédire et pas seulement pour calculer les couches supérieures. (d) Un FPN , chaque couche sont améliorées par les caractéristiques extraites des couches inférieures.

2.1 R-CNN

Un R-CNN [**?**], pour Region-Base Convolution Neural Network, est une architecture de CNN utilisé dans la vision d'ordinateur, spécialement pour ce qui est la détéction d'objet. Celle-ci est plus adaptée qu'un CNN lorsque plusieurs objets sont présent sur l'image.

L'image ci-dessous décrit correctement le concept d'un **Region-Based CNN (R-CNN)**. La première étape génère les boites autour des différents objets de l'image. Ensuite, le CNN est utilisé sur chacun des **Région d'Intérêt (RoI)** pour classifier les différentes régions de l'image avec les classes proposées.

Le R-CNN a été conçue pour résoudre les taches de détections d'images. Ainsi, l'architecture de ce dernier est donc la base des Mask R-CNN et a aussi été améliorée par les Faster R-CNN.

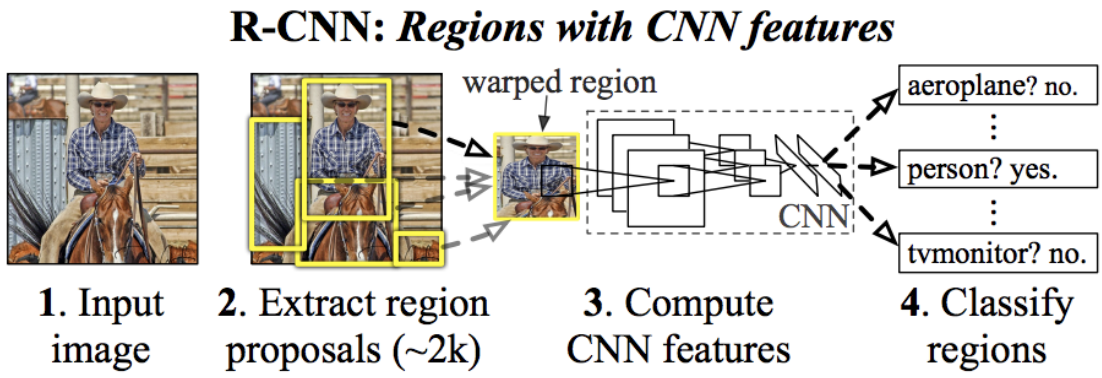


FIGURE 4 – Concept du R-CNN

2.2 Faster R-CNN

L'architecture des Faster R-CNN [**?**] améliorent celle des R-CNN en deux étapes.

1. L'utilisation d'un **Region Proposal Network (RPN)**, un simple réseau de neurone qui scanne chaque région et prédit si un objet est présent ou non.
2. **Fast R-CNN** en tant que détecteur. Les différentes caractéristiques sont extraites utilisant un RoIPool (Region of Interest Pooling) pour chacune des boites obtenues à l'étape précédentes, une classification est alors appliquée. Le RoIPool est appliqué pour extraire une carte des caractéristique à partir des région d'intérêt obtenues.

La raison pour laquelle les Fast R-CNN sont plus rapide que les R-CNN est parce qu'ils n'ont pas besoin de scanner toutes les propositions de régions mais ne le fais seulement qu'une fois par image, et une carte des caractéristiques est générée à ce moment là.

Les Faster R-CNN sont la version optimisé des Fast, ils sont conçus de telles sortes à grandement réduire le temps de calcul (Temps de prédiction : 40-50 sec pour le R-CNN contre 0.2 sec pour le Faster R-CNN).

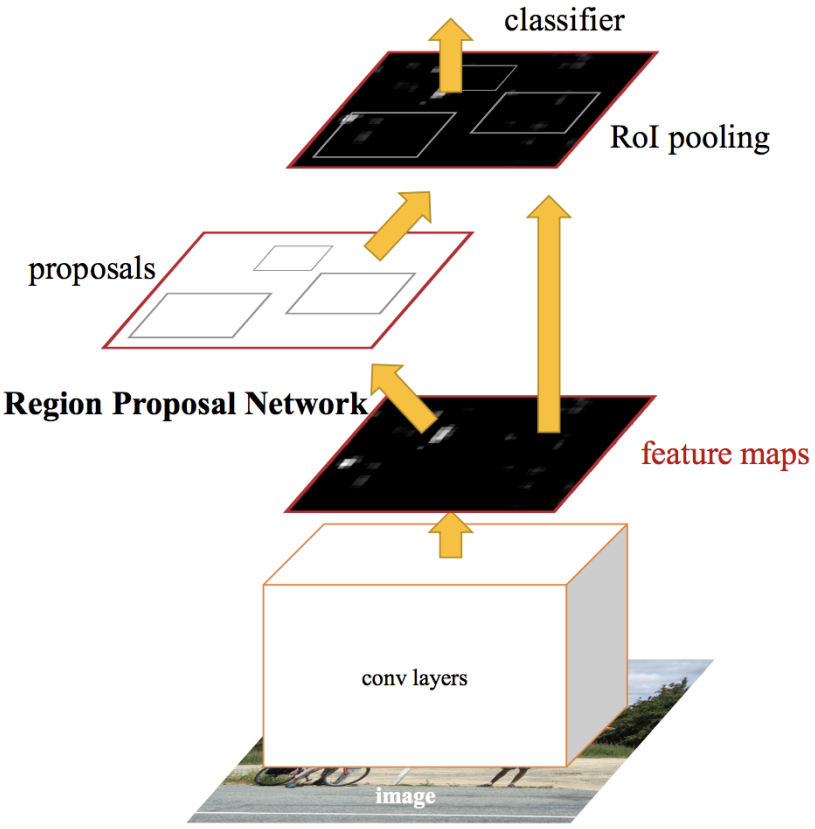


FIGURE 5 – Architecture du Faster R-CNN

2.3 Mask R-CNN

Le Mask R-CNN [?] est une architecture de réseau convolutif, qui en terme de segmentation d'image et d'instance est une des méthodes les plus employée. Cette méthode a été conçu en reprenant et améliorant l'architecture des Faster R-CNN. Alors que le Faster R-CNN n'a que deux sorties pour chaque objet candidat, un nom de classe et un contour de boite, le Mask R-CNN possède en plus des ces deux la une troisième branche qui a pour sortie le masque. La sortie du masque est distinguée des sorties de la classe et des contours.

Le Mask R-CNN est une extension du Faster R-CNN et ajoute à ce dernier une branche pour la prédiction d'un masque sur les RoI en parallèle avec la branche de prédiction de contours.

Les avantages d'un Mask R-CNN :

— **Simplicité :** Un modèle très simple à entraîner.

— **Performance :** Le Mask R-CNN performe mieux que tous les autres modèles à entrés uniques sur toutes les tâches.

— **Flexibilité :** Le Mask R-CNN peut facilement être utilisé pour d'autres tâches telle que la détection de la posture d'un homme

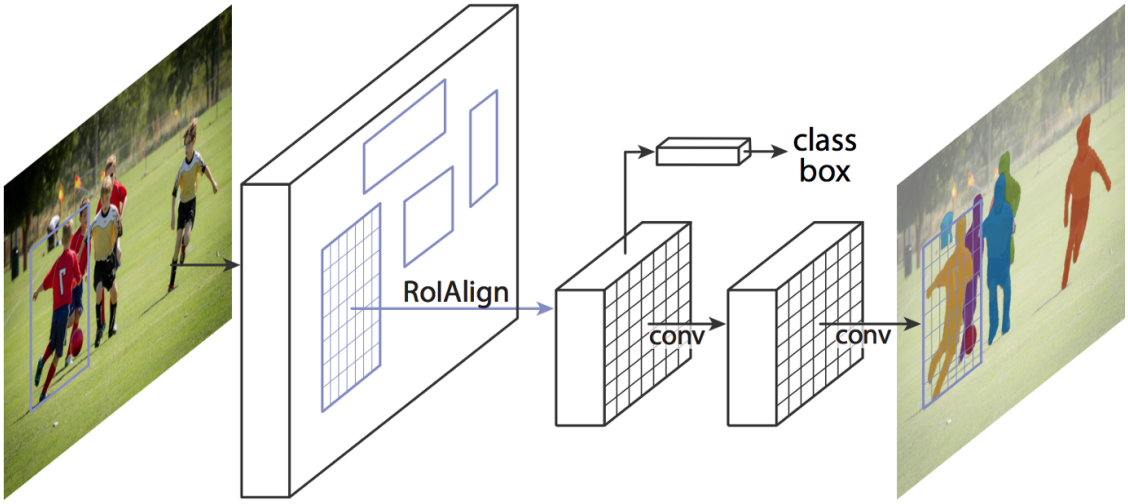


FIGURE 6 – Architecture du Mask R-CNN

2.4 PANet

L'architecture de Path Aggregation Network a pour but d'améliorer la transmission d'information grâce à sa capacité conserver les informations spatiale des pixels ce qui aide pour la formation des masques.

Les propriétés importantes permettant à PANet d'être si précis sont :

— un chemin de bas en haut (**bottom-up path augmentation**) en plus du chemin de haut en bas (top-down) des FPN

— l'utilisation de l'AFP (**Adaptive Feature Pooling**) afin d'utiliser les informations provenant des tous les niveaux

— l'utilisation des **Fully-connected layers** pour augmenter la prédiction de masque

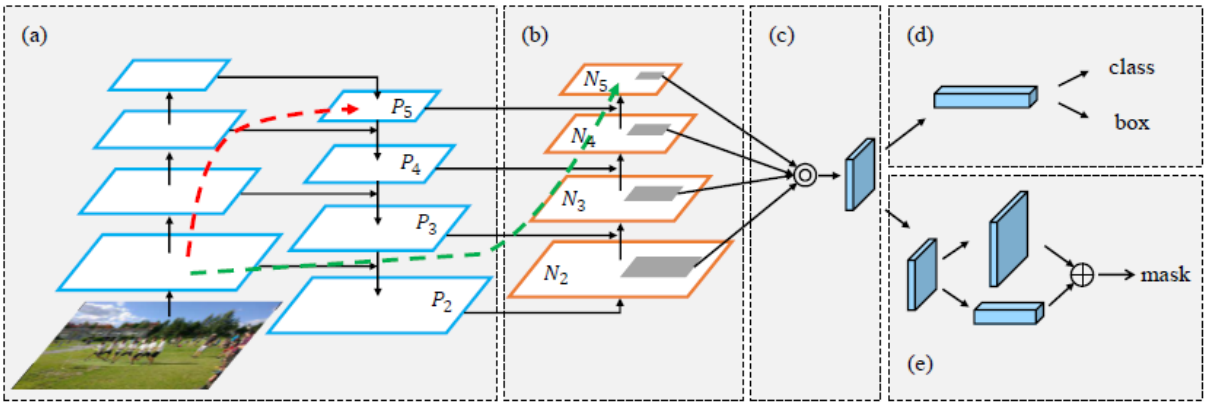


FIGURE 7 – Représentation du modèle PANet

2.4.1 Bottom-up Path Augmentation

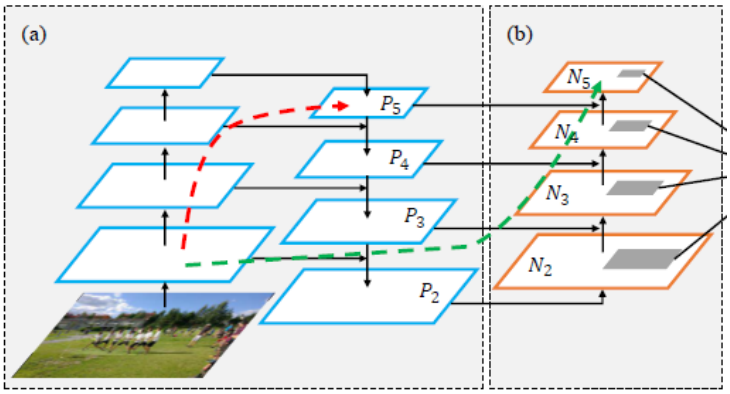


FIGURE 8 – (a) FPN backbone et (b) Bottom-up augmentation

Puisqu'une image passe à travers de nombreuses couches du réseau de neurones, la complexité des caractéristiques s'accroît alors que la résolution de l'image décroît. Ainsi, appliquer un masque avec seulement les informations des couches les plus supérieures est impossible. Le FPN utilisé dans YOLOv3, implémente un top-down path pour combiner les informations et pouvoir localiser les masques plus précisément. Mais cette tâche devient assez longue du à la centaine de couche du FPN. Un bottom-up path est ainsi utilisé dans PANet afin de propager les informations des couches inférieurs plus facilement en faisant des connections latérales avec le top-down path. Le raccourci utilisé est seulement composé d'une dizaine de couches.

2.4.2 Adaptative Feature Pooling

Les algorithmes précédents tels que les Mask-RCNN utilisaient les informations d'une seule couche pour prédire un masque. Un **ROI** (Region Of Interest) **Align Pooling** est utilisé pour extraire les informations des couches les plus hautes. Cependant bien que suffisamment précis, cela peut parfois conduire à des résultats non optimaux. Deux régions avec 10 pixels de décalage peuvent être assignées à deux régions différentes, alors qu'elles sont en fait assez similaires. C'est pour éviter cela que PANet utilise les informations des toutes les couches et le réseau choisit lesquelles sont utiles. Après avoir extrait les

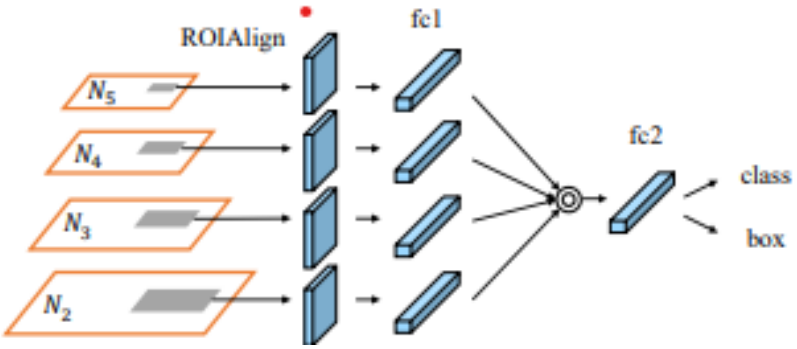


FIGURE 9 – Adaptative feature pooling

informations, l'opération *max* est utilisée pour fusionner les informations des différentes couches et le réseau peut les adapter.

2.4.3 Fully-Connected Layers

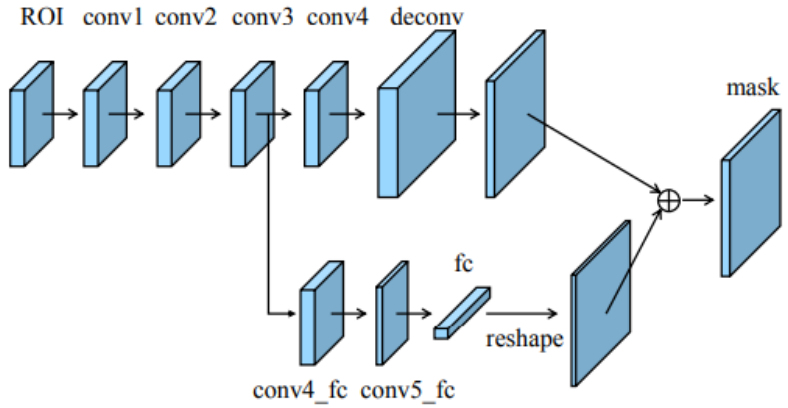


FIGURE 10 – Fully-connected fusion pour la prédiction de masque

Les Mask-RCNN utilisaient des fully-convolutional network plutôt que les fully-connected layers car elles préservent les informations spatiales et réduisent le nombre de paramètres du réseau. Pour ce qui est des *fcl*, ces derniers sont sensibles aux emplacements et peuvent alors s'adapter.

3 Implémentation

3.1 Architecture

L'architecture retenue pour l'implémentassions a été le Mask R-CNN. Plutôt que de reprendre de zéro en utilisant les documents de recherche, j'ai décider d'utiliser la bibliothèque Mask R-CNN construite par Matterport dont le code est accessible ici : https://github.com/matterport/Mask_RCNN couplée à la librairie **Tensorflow**.

L'implémentation d'un Mask R-CNN utilisant cette bibliothèque étant très simple et rapide, je me suis donc tourné vers cette solution d'architecture. J'ai couplé cette

architecture au langage python en utilisant un jupyter notebook, ainsi que le service **Google Colab** qui offre des **GPU** puissant et gratuitement dans le cadre de la recherche dans l'apprentissage automatique.

Avant tout travail sur le Mask, la première étape afin d'entraîner un Mask R-CNN est de préparer les annotations qui vont servir aux masques. J'ai obtenu un dataset d'image cellulaire fournis par mon encadrante. 40 images ont été sélectionnées pour former le set d'entraînement et de validation du réseau. Une répartition de 90% pour l'entraînement et 10% pour la validation. Un ensemble d'image de taille suffisante pour un Mask R-CNN.

J'ai ainsi utilisé l'outil `https://www.makesense.ai/` pour générer pour chaque image, un masque pour chacune des cellules compris dans l'image.

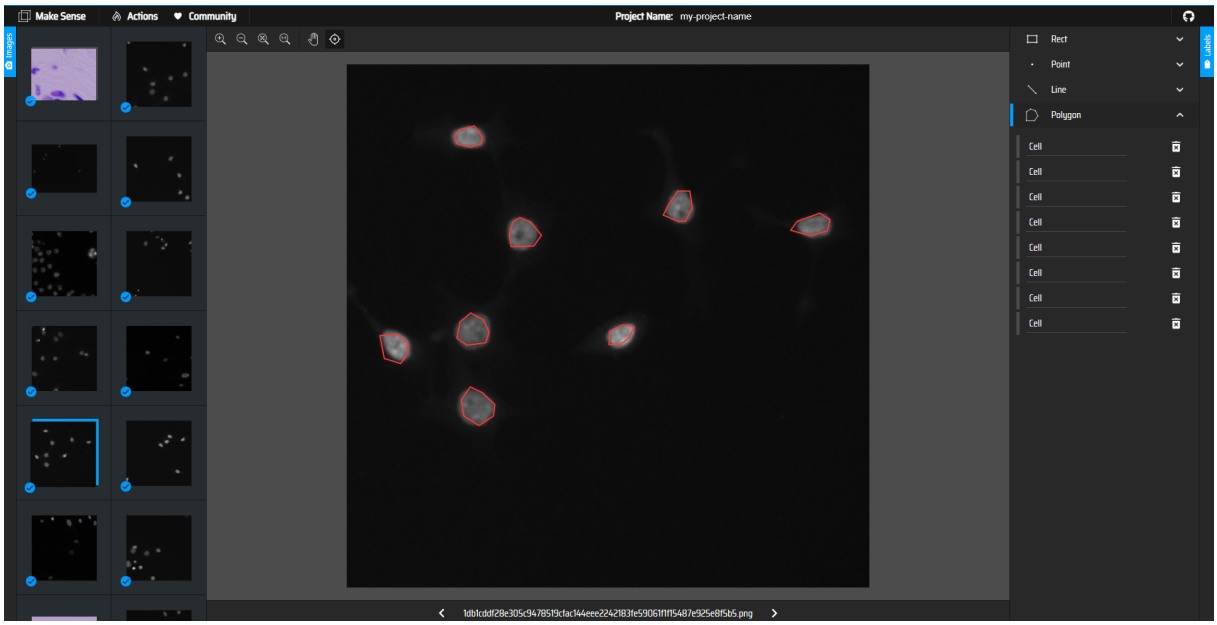


FIGURE 11 – Annotation des masques via l'outil **Make Sense**

Les annotations ont ainsi été stockées dans un fichier au format COCO JSON permettant d'être utilisé par le réseau.

Pour l'implémentation, j'ai procédé en plusieurs étapes. La première a été de charger la bibliothèque et a configurer le GPU de Google.
La seconde a été de charger le dataset. D'abord d'extraire le fichier .ZIP contenant les images, ensuite de les séparer en deux set avec une répartition 90/10.
La troisième étape est d'instancier le modèle, pour ce faire j'ai instancié la configuration par défaut et le modèle à partir de cette configuration. Le modèle est instancié à partir des poids COCO pré-entraînés.
La dernière étape est bien entendu l'entraînement du modèle. L'entraînement se fait sur 5 epochs, de 500 étapes chacun. Chaque epoch dure environs 20 min.

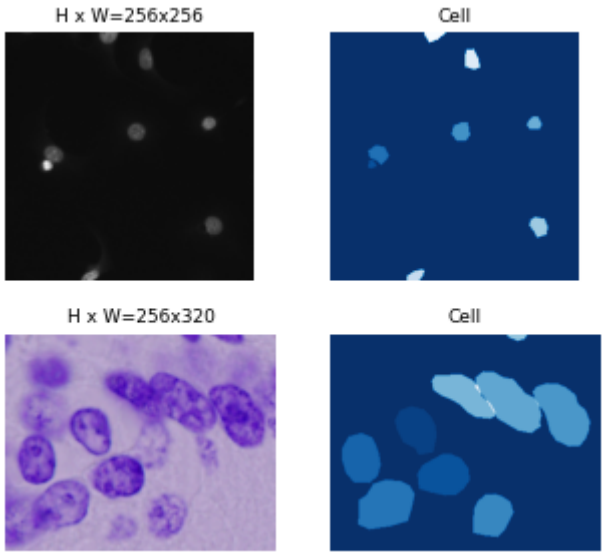
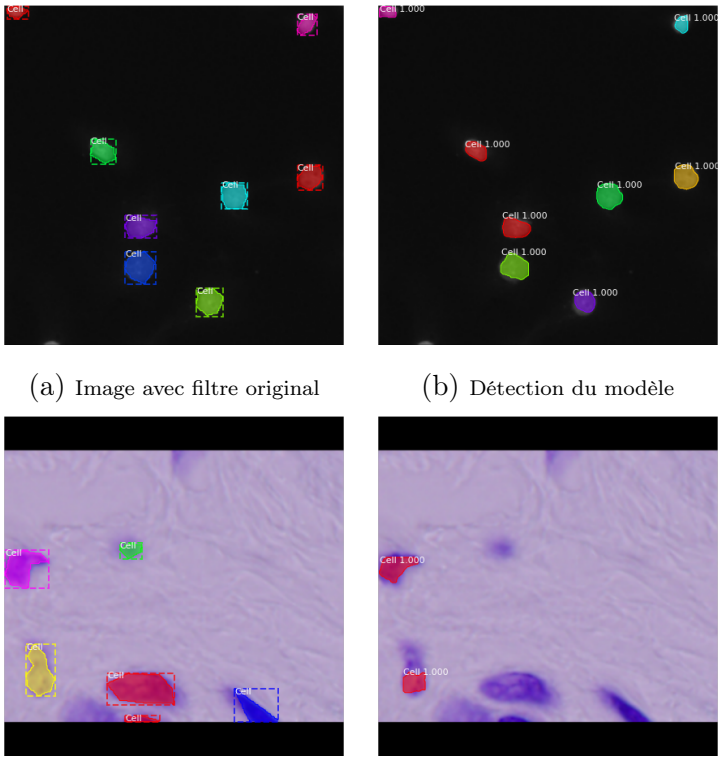


FIGURE 12 – Quelques échantillons d'images et de leurs masques.

3.2 Résultats

Une fois le modèle entraîné, on peut effectuer des détections sur des images. Comme mentionné précédemment, le set de validation contient 4 images.



(a) Image avec filtre original (b) Détection du modèle

(c) Image avec filtre original (d) Détection du modèle

FIGURE 13 – Masque d'origine et masque prédit par le réseau

Comme on peut le voir, les résultats sont bien mieux pour les images en noir et blanc, où le masque est proche du parfait, contrairement aux images en couleurs où là notre modèle manque de précision. Cela doit sûrement être dû au fait que notre set d'image est majoritairement en noir et blanc.

Pour tester la **mAP** (mean Average Precision) de notre modèle, j'ai sélectionné 4 images aléatoires du set, et obtenu une $mAP = 0.79$. Bien que cela ne représente que trop peu d'image, cela reste une précision fort raisonnable.

Lorsque l'on jette un oeil à quelques ROIs générées, on peut apercevoir des ROIs correspondantes à des cellules, mais d'autres ne correspondants à aucun objet de l'image. Ces dernières sont filtrées et ne seront pas conservées.

Les images des résultats sont disponible ici.

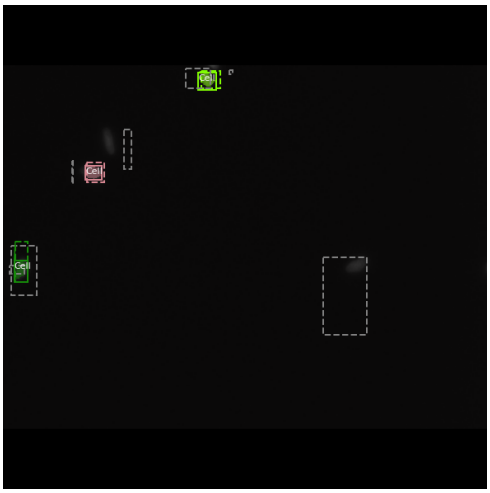


FIGURE 14 – 10 ROIs aléatoires parmi plus de 200.

4 Conclusion

Voici donc mon implémentation d'un réseau de neurone utilisant l'architecture Mask R-CNN [3] entraîné sur des images cellulaires donnant de bons résultats avec très peu d'image et pouvant concurrencer les autres architecture de ce domaine,

Mon modèle donne de bons résultats et nécessitant peu d'image, on pourrait toutes fois améliorer sa précision avec une plus grandes sélections d'images, notamment des images en couleurs.

5 Bibliographie

Références

[1] CS 230 - Pense-bête de réseaux de neurones convolutionnels. https ://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels.

[2] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation. *arXiv :1803.01534 [cs]*, September 2018. Comment : Accepted to CVPR 2018.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv :1703.06870 [cs]*, January 2018. Comment : open source ; appendix on more results.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4 : Optimal Speed and Accuracy of Object Detection. *arXiv :2004.10934 [cs, eess]*, April 2020.

[5] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep Learning in Medical Image Analysis. In M. L. Yarmush, editor, *Annual Review of Biomedical Engineering, Vol 19*, volume 19, pages 221–248. Annual Reviews, Palo Alto, 2017.

[6] Mohit Sewak. *Practical Convolutional Neural Networks*. Packt Publishing, 1st edition edition, 2018.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. *arXiv :1506.01497 [cs]*, January 2016. Comment : Extended tech report.

[8] Le Mask R-CNN pour la délinéation de parcelles : retour d'expérience – Séries Temporelles.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. *arXiv :1612.03144 [cs]*, April 2017.

[10] Jeremiah W. Johnson. Adapting mask-RCNN for automatic nucleus segmentation. *arXiv :1805.00500 [cs]*, 944, 2020. Comment : 7 pages, 3 figures.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv :1311.2524 [cs]*, October 2014. Comment : Extended version of our CVPR 2014 paper ; latest update (v5) includes results using deeper networks (see Appendix G. Changelog).

[12] Ross Girshick. Fast R-CNN. *arXiv :1504.08083 [cs]*, September 2015. Comment : To appear in ICCV 2015.

[13] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental Concepts of Convolutional Neural Network. In Valentina E. Balas, Raghvendra Kumar, and Rajshree Srivastava, editors, *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, volume 172 of *Intelligent Systems Reference Library*, pages 519–567. Springer International Publishing, Cham, 2020.