

SYS843-01 RÉSEAUX DE NEURONES ET SYSTÈMES FLOUS (H2019)

DÉPARTEMENT DU GÉNIE DE LA PRODUCTION AUTOMATISÉE

SYNTHÈSE DE LITTÉRATURE : DÉTECTION D'OBJETS

Soumis par

Danny SAUVAL

Département du Génie de la Production Automatisée

École de Technologies Supérieures

Montréal, QC

Soumis à

Ismail BEN AYED

École de Technologies Supérieures

Montréal, QC



Le génie pour l'industrie

Table des matières

1 Mise en situation	3
1.1 Domaine d'application	3
1.2 Problématique abordée	3
1.3 Objectifs du projet	4
1.4 Méthodologie	4
2 Approches pertinentes de la littérature	5
3 Description détaillée des techniques qui sont exploités dans ce projet	6
3.1 Algorithme de proposition de région (Region Proposal Algorithm)	6
3.1.1 Recherche sélective (selective search)	6
3.2 Algorithme de détection d'objet	8
3.2.1 R-CNN	8
3.2.2 Fast R-CNN	13
3.2.3 Faster R-CNN	15
3.2.4 SSD : Single Shot Detectors	17
4 Analyse critique	20
5 Conclusions	22

1 Mise en situation

1.1 Domaine d'application

Le fonctionnement de l'être humain est fascinant et passionnant. L'intelligence dont nous faisons preuve dès nos premiers jours nous permet de rapidement prendre conscience du monde qui nous entoure et de le comprendre. La détection d'objet est un bon exemple. Après avoir vu quelques fois un objet, nous sommes capables de facilement le détecter et de le reconnaître quand on le voit, et cela, même si l'objet est différent.



FIGURE 1: À gauche une chaise classique [1] et à droite une chaise de designer [2]

Mes propos peuvent être illustrés par la Figure 1. À gauche, on a une chaise classique : un dossier et un plateau supporté par 4 pieds. À droite, on a une chaise de designer qui ne correspond pas du tout à la définition d'une chaise. Pourtant notre cerveau est capable de l'identifier comme tel.

Le but de mon projet est de chercher à comprendre comment fonctionnent les méthodes actuelles pour la détection d'objet.

Les applications de ce type de systèmes sont très diverses, on peut par exemple citer l'utilité pour une voiture autonome qui pourrait reconnaître les objets présents sur la route et son environnement. Le système pourrait aussi être utilisé dans un robot d'assistance à la personne qui pourrait aller chercher un objet dans la maison.

Le but du projet est dans un premier temps d'être capable de détecter un objet dans une image (et plus tard une vidéo), et si les résultats sont concluants ajouter au système la possibilité de détecter plus d'objets.

1.2 Problématique abordée

La détection d'objet est un sujet largement abordé dans la littérature. La grande majorité des algorithmes sont basés sur l'utilisation d'un CNN, qui permet généralement de faire de la classification d'image et donne les meilleurs résultats. Si on prend en photo un chat, l'algorithme sera capable de classifier l'image comme étant celle d'un chat. Cependant, dans

ce projet on ne veut pas faire de classification, on veut identifier dans une image plusieurs objets. L'idée de base est donc de découper l'image de base en plusieurs régions, et d'envoyer chacune de ces régions à un CNN.

Dans cette revue de littérature, nous pourrons présenter 4 techniques parmi les techniques les plus utilisées : R-CNN, Fast R-CNN, Faster R-CNN et SSD.

1.3 Objectifs du projet

L'objectif du projet va être d'entraîner un réseau de neurones à détecter et reconnaître un objet dans une image. Par exemple, 80% d'une database contenant des vélos va être utilisé pour entraîner un réseau. On testera la précision de celui-ci à l'aide des 20% restant de la database. Si le résultat n'est pas suffisant, on entraînera à nouveau le réseau. On pourra éventuellement utiliser un réseau pré-entraîné tel que AlexNet ou MobileNet. Notre algorithme devra être capable d'apprendre à détecter et reconnaître de nouveaux objets. Il faudra également optimiser l'algorithme pour essayer de détecter et reconnaître les objets en temps réel. Cela permettrait de rendre l'algorithme compatible avec une voiture autonome, qui a besoin d'informations en temps réel.

Un objectif parallèle à mon projet et qui découle d'une contrainte est que le projet doit être capable de tourner raisonnablement sur mon ordinateur dont la configuration est la suivante : Windows 10, CPU Intel(R) Core(TM) i5-7300HQ, GPU Nvidia GeForce GTX1060 6GB Max-Q Design.

1.4 Méthodologie

Les outils que je compte utiliser seront Matlab ainsi que OpenCV combiné à Python ou bien au C++. Je connais bien ces outils ce qui va donc m'éviter d'apprendre à utiliser un nouvel outil.

Dans un premier temps, je chercherai à exécuter les codes des papiers étudiés afin de directement tester les algorithmes et de les découvrir. Une analyse profonde de ses algorithmes sera faite.

Je vais me concentrer sur l'utilisation d'une database, Pascal VOC : <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#devkit>. En cas de besoin, j'utiliserai la cocodataset qui est beaucoup plus grande que la Pascal VOC : <http://cocodataset.org/#download>.

2 Approches pertinentes de la littérature

Dans cette partie, nous allons rapidement aborder les approches pertinentes que l'on peut trouver dans la littérature. Notre problème de détection d'objet est similaire à celui de la classification d'une image :

1. Classification d'image : donnée une image en entrée, l'objectif est d'être capable de déterminer ce que l'image représente le plus. Par exemple si on a une photo de chien, un algorithme de classification d'image devra être capable de la classer comme correspondant à un "chien". Pour une image en entrée, on a une seule sortie, la classe de l'image.



FIGURE 2: Image d'un chien mignon qui devrait être classifié comme celle d'un chien

2. Détection d'objet : donnée une image en entrée, l'objectif est d'être capable de déterminer tout ce qui compose l'image. Pour une image en entrée, on a en sortie un ou plusieurs résultats.

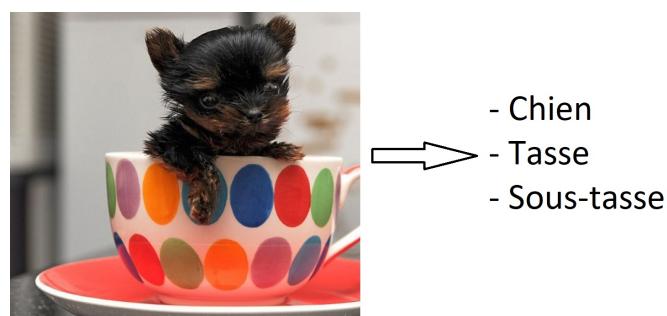


FIGURE 3: Image qui correspond à un chien, une tasse et une coupole

À noter que dans la plupart des méthodes utilisées (notamment dans celle que j'ai pu étudier), à une sortie donnée est associée sa position dans l'image.

Dans la littérature on peut trouver beaucoup de méthodes qui adaptent la classification

d'image à la détection d'objet. En effet, si on est capable de découper proprement l'image, le projet se résume à un problème de classification d'image.

Il existe beaucoup de techniques permettant de découper une image en plusieurs régions correspondant potentiellement à un objet, on peut notamment citer :

1. Recherche sélective (selective search)
2. Recherche exhaustive (exhaustive search) (fenêtre de sélection/fenêtre glissante/sliding window)
3. Segmentation
4. Objectivness
5. Constrained parametric min-cuts

Ces méthodes peuvent être couplées à des techniques qui ne seront pas abordés dans ce sujet, on peut notamment citer :

1. You Only Look Once (YOLO) : Cette technique utilise un seul CNN qui prédit la région d'intérêt et la classe associée à celle-ci. Nous ne travaillerons pas sur cette technique dans ce projet. Cette méthode est très rapide (40 à 155 fps sur une carte graphique Titan X) mais elle est moins précise que d'autres techniques.
2. SPP-net, technique très similaire à R-CNN que nous détaillerons plus tard.

3 Description détaillée des techniques qui sont exploités dans ce projet

3.1 Algorithme de proposition de région (Region Proposal Algorithm)

3.1.1 Recherche sélective (selective search)

Cette section est en grande partie inspirée par l'article de J.R.R. Uijlings et al. [3]. Le principe de la recherche sélective est inspiré de celui de la recherche exhaustive. Il fait partie des techniques que l'on peut appeler "alternative à la recherche exhaustive". On ne détaillera cette dernière technique que nous n'utiliserons pas dans notre projet. Cependant, il est à noter que cette technique est très lourde quant à la capacité de calcul requise. C'est une méthode de force brute qui, donnée une image en entrée, va essayer de chercher partout dans l'image les zones qui correspondent à des régions susceptibles d'être intéressantes. La sortie de la recherche exhaustive donne parfois un nombre énorme de régions qui sont plus ou moins précises.

L'objectif de la recherche sélective est de proposer une méthode similaire moins lourde en capacité de calcul, et donnant moins de régions en sortie. La méthode se veut également class-independant ce qui permet à l'algorithme d'être capable de trouver les régions de n'importe quel type d'objet dans l'image.

Pour écrire leur algorithme, J.R.R. Uijlings et al. sont partis de différents objectifs importants :

1. Taille des objets variables : les objets de l'image en entrée ont des tailles variables ⇒ cela a un impact sur la fenêtre de sélection qui aura une taille variable.
2. Diversification : les régions proposées initialement sont nombreuses. L'algorithme de recherche sélective doit donc fusionner celles qui correspondent potentiellement à la même zone. Pour cela, J.R.R. Uijlings et al. utilisent plusieurs caractéristiques, variation de la taille, couleurs, textures, et contours/forme.
3. Rapidité de calcul : l'algorithme ne doit pas être trop lourd à exécuter.

La fonctionne de l'algorithme de recherche sélective peut-être résumé comme suit :

1. Génération initiale d'une sur-segmentation, cela génère dans l'image beaucoup de régions. Le principe de la sur-segmentation est de générer beaucoup de régions, même trop et on le sait. Cela donne une première approche un peu grossière, mais qui fournit un bon point de départ. Dans le papier de J.J.R. Uijlings et al. cette partie de l'algorithme est basée sur la méthode de Felzenszwalb et Huttenlocher [4].
2. Utilisation d'un algorithme glouton qui va grouper/fusionner hiérarchiquement les régions similaires. Cela va donc fusionner les régions similaires en régions plus grandes. Le nombre de régions va donc diminuer. L'algorithme de fusion des régions utilise des caractéristiques comme la variation de la taille, couleurs, textures, et contours/forme.
3. L'étape 2 est répétée itérativement jusqu'à n'obtenir qu'une seule région.

L'image suivante issue du papier de J.R.R Uijings et al. est une bonne illustration de ce que l'on vient de voir.

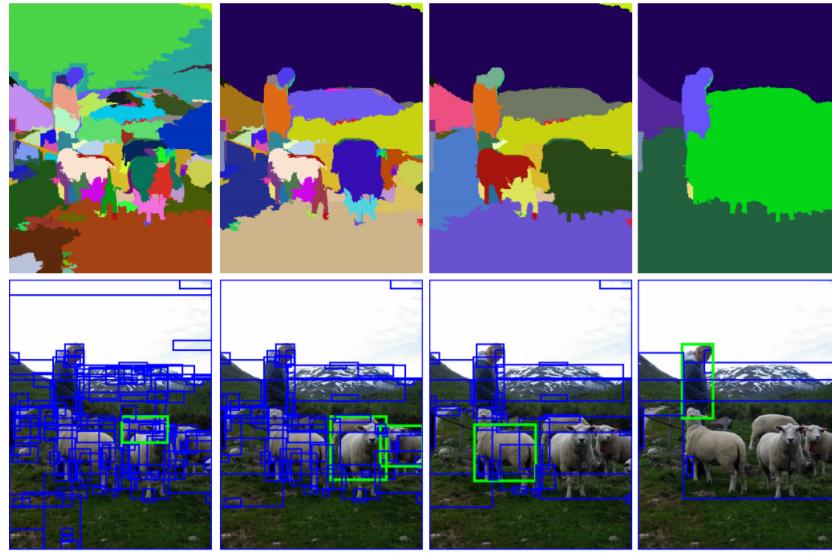


FIGURE 4: Illustration du déroulement de l'algorithme issue de [3]

On peut voir à gauche le grand nombre de segments, et plus l'on itère dans l'algorithme plus les classes sont fusionnées.

3.2 Algorithme de détection d'objet

3.2.1 R-CNN

R-CNN signifie Region-Convolutional Neural Network. Cette méthode a été créée par Ross Girshick et al [5]. Le principe est très simple :

1. Donnée une image en entrée
2. On utilise l'algorithme de recherche sélective pour proposer environ 2000 régions pour une image.
3. Les 2000 régions sont envoyées à un CNN, qui va calculer les caractéristiques CNN (CNN features). Le CNN permet donc d'extraire les caractéristiques des régions, variation de la taille, couleurs, textures, et contours/forme.
4. Ces caractéristiques CNN sont ensuite classifiées en utilisant une SVM.
5. Une fois une région classifiée, R-CNN va effectuer une régression linéaire afin d'optimiser la bounding box pour que celle-ci correspondent au mieux aux dimensions réelles de l'objet détecté

L'illustration issue du papier de Ross Girshick et al. illustre très bien le principe :

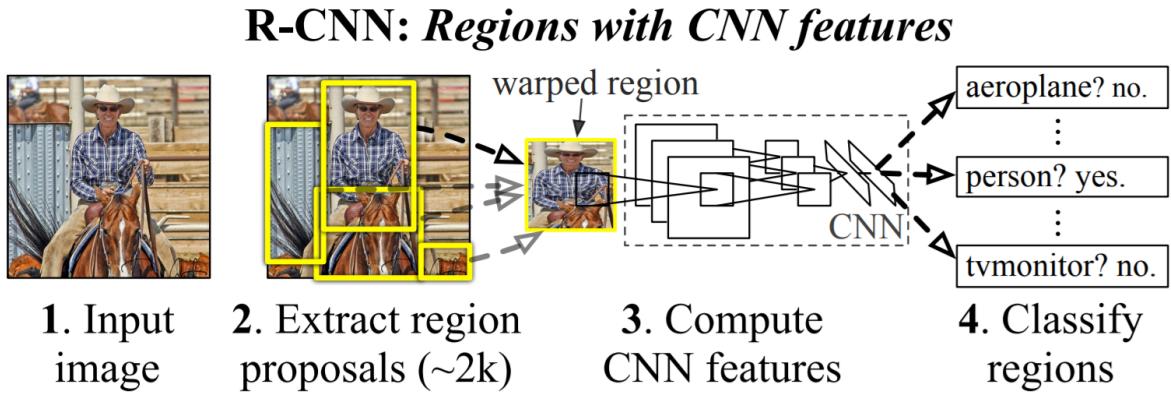


FIGURE 5: Illustration de l'algorithme R-CNN issue de [5]

1. L'extraction des régions avec la recherche sélective a été traitée dans la sous-sous-section 3.1.1
2. Pour chaque région, l'algorithme R-CNN extrait un vecteur de caractéristiques de taille 4096.

Pour cela Ross Girshick et al. [5] ont utilisé le CNN proposé par Yangqing Jia et al. [6]. Le code source de celui-ci est directement disponible sur leur Github (<https://github.com/BVLC/caffe/>). L'algorithme prend en entrée une image de taille 227x227, chaque région est donc déformée et redimensionnée en une image de taille 227x227. Le CNN agit ici comme un extracteur de caractéristiques. La partie suivante est en partie issue du cours de réseaux de neurones SYS843 proposé à l'ETS par Ismail Ben Ayed.

CNN est un réseau de neurones largement utilisé depuis un vingtaine d'années, notamment pour les problèmes de reconnaissance et traitement d'image. Il donne les meilleurs résultats actuellement. On peut résumer le fonctionnement d'un CNN comme suit :

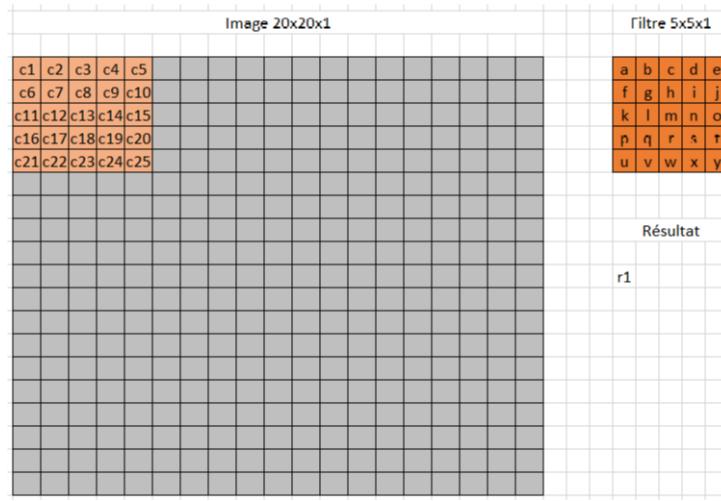


FIGURE 6: Illustration du principe de fonctionnement d'un CNN avec la première région

- (a) Donnée une image en entrée, par exemple une image 20x20x1 (niveau de gris)
- (b) On choisit un filtre, de taille 5x5x1 par exemple.
- (c) On va ensuite calculer la sortie de la couche de convolution en suivant l'algorithme suivant :
 - i. On place le filtre dans la première région comme illustré sur la Figure 7. On va effectuer une "multiplication par élément" entre la zone sélectionnée et le filtre. La première valeur de la matrice résultante sera donc $a*c1$, la seconde $b*c2$, $c*c3$, $d*c4$, ect.
 - ii. On additionne ensuite toutes les valeurs de la matrice résultante. Ce qui nous donne la valeur du premier pixel qui correspond à la première région.
 - iii. On recommence ensuite avec la région suivante.

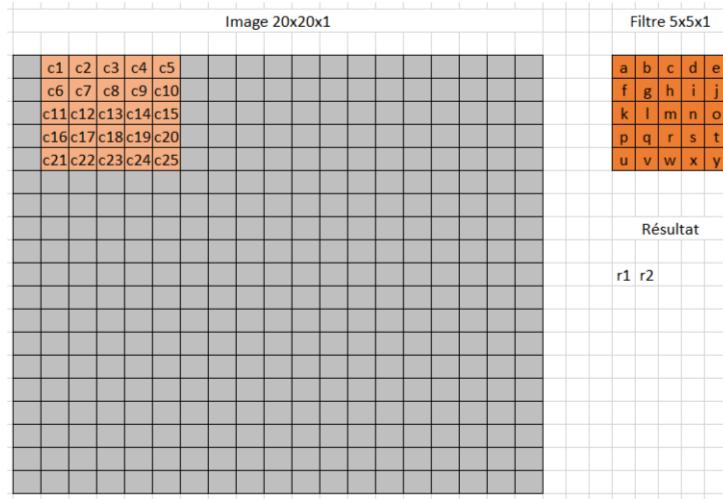


FIGURE 7: Illustration du principe de fonctionnement d'un CNN avec la seconde région

La sortie de l'algorithme est une nouvelle image, que l'on envoie à nouveau au CNN, jusqu'à obtenir un vecteur de caractéristique. A chaque couche, on peut appliquer une fonction d'activation (par exemple ReLU), qui va permettre de sélectionner l'information qui nous intéresse et réduire la lourdeur de calcul. On peut également utiliser des techniques de Pooling qui permettent de réduire la taille des images, par exemple en sous-échantillonnant l'image.

3. Des SVM (Support Vector Machines) sont utilisées pour classifier les données. C'est une technique d'apprentissage supervisée.

Le principe de base est le suivant :

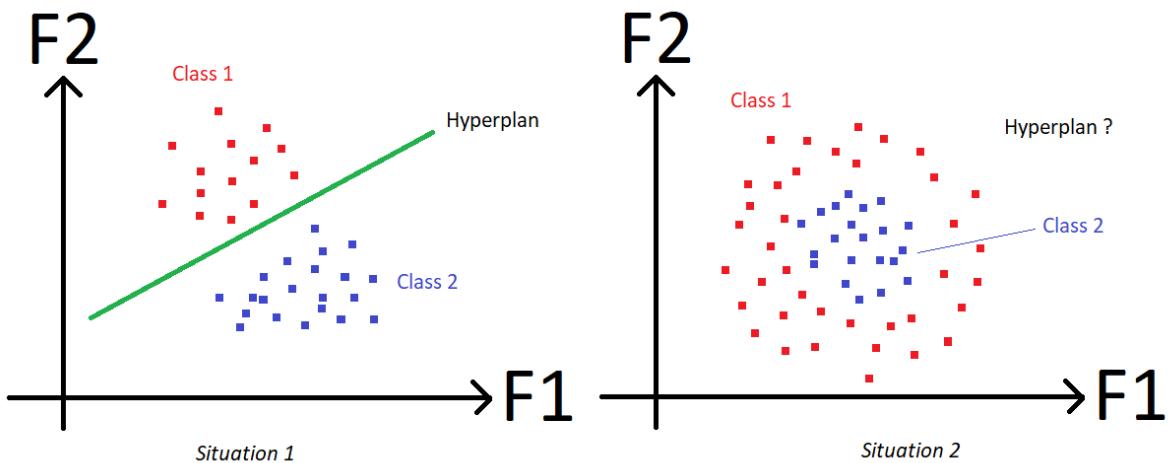


FIGURE 8: Exemple simple pour expliquer SVM

On a pu représenter rapidement sur Paint deux situations qui mettent en avant l'utilité de SVM. On a considéré deux classes, **Class 1** et **Class 2**. Les deux classes ont des données à 2 dimensions.

Le principe d'une SVM est de trouver l'hyperplan qui sépare au mieux les 2 classes de dimensions N. La dimension de l'hyperplan est toujours de N-1.

- (a) Dans le cas de la situation 1, cela n'est pas très compliqué. On peut facilement trouver l'hyperplan qui sépare les deux classes, une simple droite $y = ax + b$.
- (b) Dans le cas de la situation 2, comment séparer les deux classes? Ça n'est linéairement pas possible. Pour pallier ce problème, la SVM va ajouter aux données de chaque classe une dimension. Dans notre situation 2, on peut par exemple considérer que les points rouges de la **classe 1** auront une nouvelle dimension F3. Chaque point aura pour valeur $F3 = 1$. Tandis que pour la **classe 2** la valeur de F3 pour chaque point sera à 0. La SVM sera donc enfin capable de créer l'hyperplan qui sera un plan en 2 dimensions dépendant de F2 et F1.

Il est à noter que peu importe la répartition des données des **classe 1** et **classe 2**, la SVM sera toujours capable de trouver l'hyperplan optimal, celui qui sépare au mieux les deux classes.

De plus, dans notre cas on a plus d'une seule classe, comment la SVM va être capable de classifier les données alors que c'est un classificateur binaire? La SVM va appliquer le principe de "one-against-one". Donnée par exemple 4 classes, **classe 1**, **classe 2**, **classe 3** et **classe 4**, la SVM calcule le score entre 1 et 2, puis si 2 est le meilleur entre 2 et 3, puis si 3 est le meilleur entre 3 et 4. Il est ainsi capable selon chaque classe d'établir un score.

Ce sont donc les hyperplans qui sont capables de différencier les vecteurs de caractéristiques qui correspondent à une voiture de ceux qui correspondent à un vélo par exemple. Pour être capable de faire cela, il faut donc avoir entraîné la SVM, ou plus exactement il faut avoir laissé la SVM calculer l'hyperplan entre le vecteur caractéristique d'une voiture, et le vecteur caractéristique d'un vélo.

Conclusion R-CNN : Avant d'entrer dans les détails dans l'analyse critique, on peut soulever différents problèmes qui émanent de la méthode R-CNN :

1. Pour la phase d'entraînement : L'entraînement du réseau prend énormément de temps. Comme on a pu le voir précédemment, pour une seule image il faut classifier 2000 propositions de régions issues de la recherche sélective. Ces 2000 propositions passent

par le CNN pour l'extraction des caractéristiques, puis par la SVM pour la classification. Cela prend énormément de temps.

2. Pour la phase de test : L'algorithme au complet prend 49 secondes pour une seule image. Aucun espoir d'analyse en temps réel avec cet algorithme.

3.2.2 Fast R-CNN

Fast R-CNN est une méthode qui découle de l'itération de la méthode R-CNN. On a précédemment cité des points faibles de R-CNN. Ross Girshick, l'un des auteurs ayant créé R-CNN était conscient de ces points faibles et a cherché à améliorer la méthode. C'est ainsi qu'il a publié un article proposant une méthode plus rapide, Fast R-CNN [7].

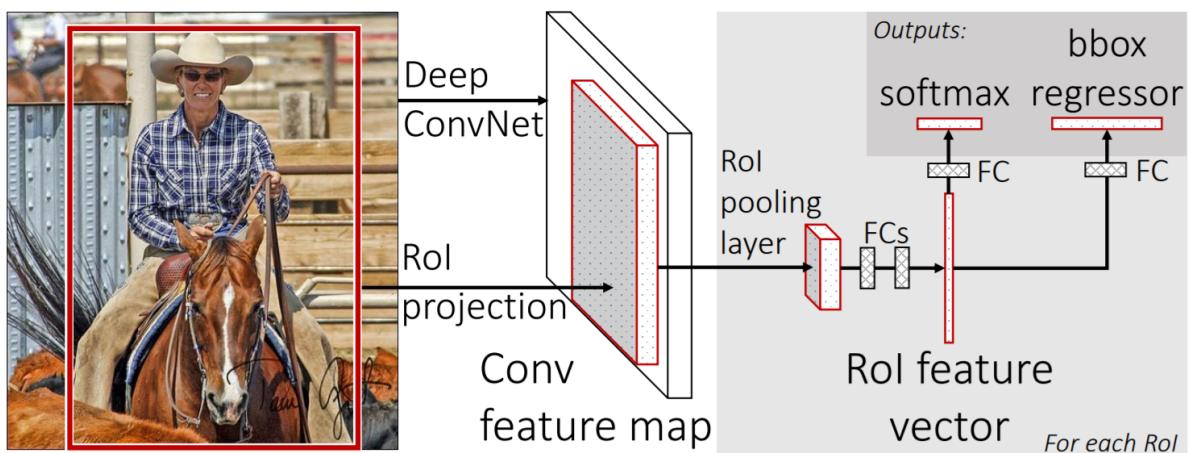


FIGURE 9: Architecture du Fast R-CNN

1. Donnée une image en entrée et un set de proposition de régions (issue de la recherche sélective dont nous avons précédemment parlé)
2. On envoie l'image dans un CNN alternant des couches de convolution et et des couches de max pooling. Cette étape permet de calculer la carte des caractéristiques (feature map).
3. Pour chaque région proposée par la recherche sélective, on extrait à partir de la feature map le vecteur de caractéristique qui correspond à la région proposée. L'extraction du vecteur de caractéristique est faite avec une couche appelée "region of interest (RoI) pooling layer", en français : couche de pooling de la région d'intérêt.
4. Chacun des vecteurs de caractéristiques obtenus à l'étape précédente est envoyé à une séquence de couches entièrement connectées. Celles-ci se terminent en deux branches :

- (a) La première donne la probabilité softmax estimée selon K classes, ainsi qu'une classe qui correspond à l'arrière-plan
- (b) La seconde retourne 4 valeurs qui sont la position et la taille de la bounding box liée à l'objet. Les 4 valeurs sont x , y , w , h . x et y correspondent aux coordonnées du point supérieur gauche de la bounding box, w correspond à la largeur de la bounding box, et h correspond à la hauteur.

Le principe de la "region of interest (RoI) pooling layer", que nous appellerons RoI pooling layer est un point clé essentiel de fast R-CNN, puisque cette partie n'était pas présente dans R-CNN. L'objectif de cette couche est, donnée une région valide, de convertir les caractéristiques de celle-ci en une carte de caractéristiques de taille fixe, $H \times W$. H et W sont appelés des hyperparamètres qui sont complètement indépendants des régions d'intérêt en entrée. Les ROI sont définis avec les mêmes paramètres que les bounding box, la coordonnée du point supérieur gauche (x, y), la largeur l et la hauteur h .

Plus précisément, le fonctionnement de la couche est relativement simple.

1. Donnée la ROI en entrée
2. On divise la ROI en une grille de taille $H \times W$ (la taille de la carte de caractéristique définie précédemment). Cette grille a des cellules de taille $h/H \times w/W$. On applique un max-pooling sur les valeurs de chaque cellule de la ROI et l'on remplit la valeur dans la grille en sortie.

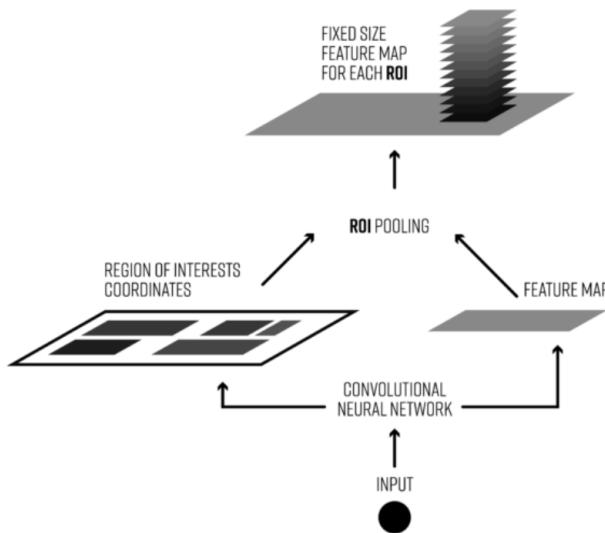


FIGURE 10: Illustration de la ROI pooling layer [8]

Lors de la phase d'entraînement, les ROI d'une même image ont tendance à effectuer les mêmes calculs. Fast R-CNN permet de partager ces différents calculs, afin d'éviter de les calculer plusieurs fois. On diminue ainsi fortement le cout en calcul.

Conclusion Fast R-CNN : L'algorithme est beaucoup plus rapide que R-CNN grâce à son efficience.

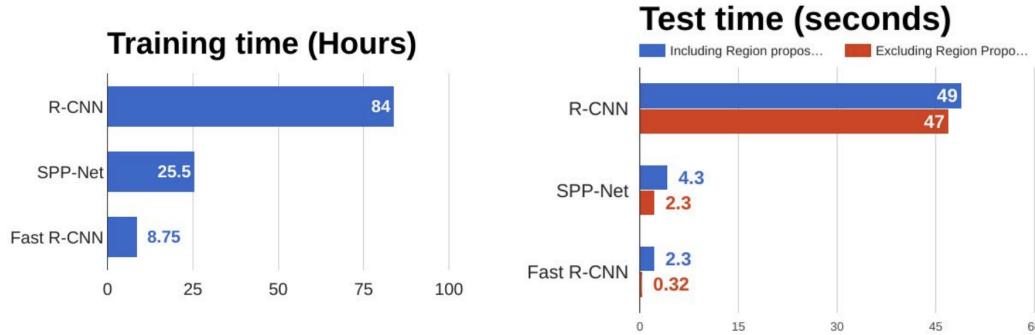


FIGURE 11: Comparaison entre SPP-Net, R-CNN et Fast R-CNN [9]

Il n'est cependant pas parfait et pourrait être amélioré, notamment au niveau de la recherche sélective.

3.2.3 Faster R-CNN

L'algorithme Faster R-CNN est une itération de fast R-CNN. L'objectif est de remplacer l'utilisation de la recherche sélective. Shaoqing Ren et al. [10] en sont les auteurs. Le principe de l'algorithme est davantage un complément à Fast R-CNN. Dans la partie précédente, on a vu que les entrées de Fast R-CNN sont une image ainsi que des propositions de ROI issues de la recherche sélective. Faster R-CNN propose une alternative à la recherche sélective, avant d'utiliser lui-même l'algorithme Fast R-CNN.

Le principe est le suivant :

1. Donnée une image en entrée
2. On envoie l'image à un réseau de neurones qui prédit les ROI. Le réseau de neurones est appelé un RPN, Region Proposal Network.
3. On envoie l'image et les ROI à Fast R-CNN.

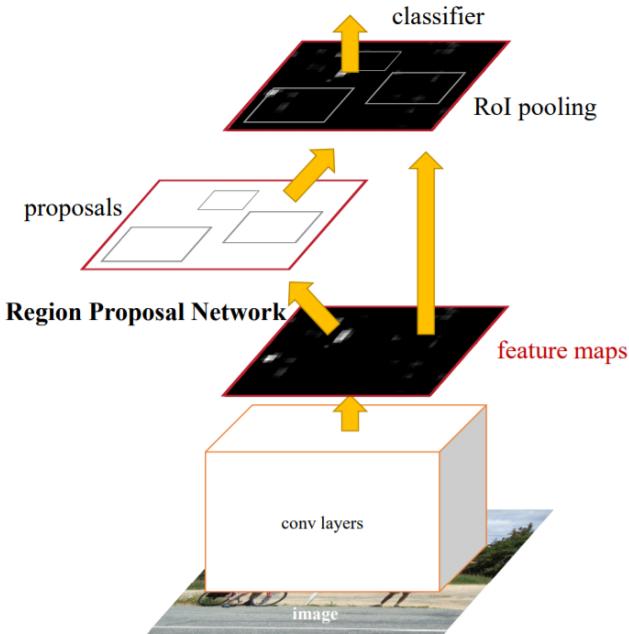


FIGURE 12: Architecture de Faster R-CNN [10]

Faster R-CNN est donc basé sur deux modules : RPN et Fast R-CNN.

Shaoqing Ren et al. se sont rendu compte qu'il était possible et bien plus efficace d'utiliser la carte des caractéristiques issue du CNN en entrée du RPN. L'objectif du RPN est donc de faire passer la carte de caractéristiques à travers d'autres couches.

RPN est en fait un CNN dont les couches sont entièrement connectées entre elles.

- Entrée du RPN : La carte de caractéristique du CNN
- Sortie du RPN : Bounding box d'un potentiel objet avec le score d'objectivité. Le score d'objectivité permet de savoir si la bounding box proposée correspond à plus probablement à un objet ou bien à un élément de l'arrière-plan.

Le déroulement de l'algorithme est le suivant :

1. On fait glisser un réseau sur la carte des caractéristiques du CNN. Ce réseau prend en entrée une fenêtre de taille $n \times n$ du réseau de neurones. Dans le papier [10] la taille est 3×3 .
2. La fenêtre est mappée dans un vecteur de caractéristique de dimension plus basse.
3. Le vecteur est ensuite envoyé à deux couches entièrement connectées jumelles :
 - (a) Une couche de box-regression -> permet d'optimiser la précision de la bounding box pour correspondre aux dimensions réelles de l'objet.

- (b) Une couche de box-classification -> permet de classifier le contenu de la bounding box

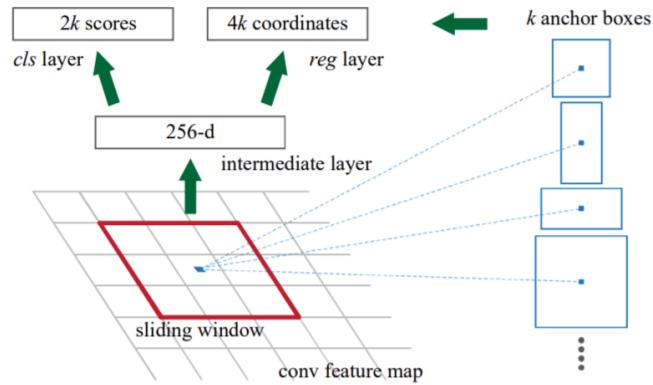


FIGURE 13: Illustration de l'architecture du RPN [10]

Conclusion Faster R-CNN : Faster R-CNN est une itération de Fast R-CNN, lui-même une itération de R-CNN. Son plus grand avantage est le gain de rapidité énorme par rapport à R-CNN et non négligeable par rapport à Fast R-CNN.

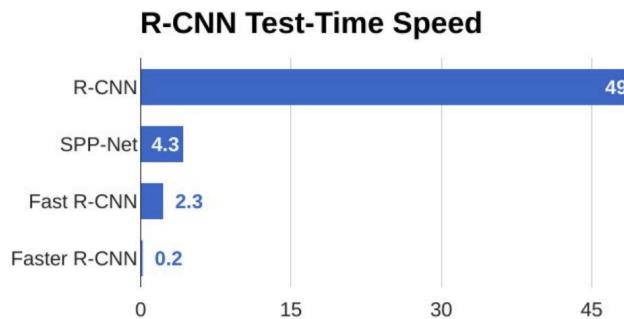


FIGURE 14: Comparaison des rapidités d'exécution [9]

L'inconvénient est que la méthode paraît en théorie complexe. Dans le prochain papier, nous aurons expérimenté et serons capables de plus clairement expliquer l'algorithme.

3.2.4 SSD : Single Shot Detectors

On a pu voir dans les algorithmes de la famille R-CNN qu'ils sont tous basés sur des propositions de régions. Le principe de SSD, Single Shot Detectors a été mis en place par Wei Liu et al. [11]. Cette technique utilise un seul et unique réseau de neurones. C'est une des techniques actuellement qui allie au mieux précision et rapidité. D'après Wei Liu et al., leur

technique est aussi précise que Faster R-CNN alors que beaucoup plus rapide, et beaucoup plus simple vis-à-vis de l'architecture.

Les deux fondements de SSD par rapport à d'autres systèmes de détection d'objet sont les suivants :

1. SSD n'utilise pas d'algorithme de proposition de régions comme la recherche sélective.
2. SSD ne rééchantillonne pas les données des pixels, ce qui est le cœur des algorithmes de la famille R-CNN par exemple.

On peut expliquer l'algorithme comme suit :

1. Phase d'entraînement : on a besoin en entrée d'une image, ainsi que des ground truth de celle-ci. Les ground truth sont généralement d'intervention humaine, et permettent de fournir à l'algorithme lors de la phase d'entraînement, des données sûres, des ground truth. Cette nécessité n'est pas une contrainte énorme, puisqu'on peut très facilement trouver des databases qui ont ce type d'images. On peut notamment citer Pascal VOC [12] qui est cité dans tous les papiers liés à la détection d'objet que j'ai pu lire. On peut également citer cocodataset qui propose même un outil de recherche en ligne de donnée :



FIGURE 15: Exemple du type de fichier dont on a besoin pour entraîner SSD

Donnée cette image en entrée, on la fait passer à travers des couches de convolutions qui vont permettre d'obtenir des cartes de caractéristiques. Les cartes de caractéristiques obtenues ont toutes une échelle différente, ce qui va permettre de prédire la localisation d'objets de tailles différentes.

Pour une carte de caractéristiques de taille $m \times n$ avec p canaux, l'élément de base permettant de prédire les paramètres d'une détection potentielle est de taille $3 \times 3 * p$. Cet élément produit soit un score d'appartenance à une catégorie, ou bien un offset relatif à la position de la bounding box de base. L'élément de base est appliqué à chaque localisation de la carte de caractéristiques. Cela va donner un certain nombre de bounding box où un objet est potentiellement présent.

La position des bounding box par défaut dépend des features map. Le nombre d'éléments d'une feature map correspond au nombre de localisations possible pour une bounding box. Par exemple si on a une feature map de taille 10×10 , on aura 100 localisations possibles. Selon que l'on cherche à détecter un être humain ou bien une voiture, le format (aspect ratio) de la bounding box sera différent. Probablement vertical pour un être humain, et horizontal pour une voiture. L'algorithme va donc proposer plusieurs formats de bounding box, et calculer le score de celle-ci, le principe est illustré sur la figure suivante :

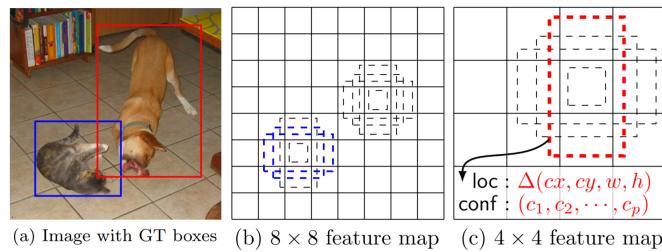


FIGURE 16: Exemple issue de papier de Wei Liu et al. [11]

La première image correspond à l'image dont les boundings box sont déjà connus (ground truth). L'image 2 est issue d'une feature map de taille 8×8 . Dans celle-ci on retrouve le chien et le chat. Différentes bounding box de différents formats ont dans cette feature map été proposés. Seule la bounding box qui donne le meilleur score est considérée comme une détection positive, les autres comme des détections négatives. Le score est évalué en fonction de la différence de position et d'échelle et de format entre la ground truth bounding box.

Résumé de l'architecture issue du papier de Wei Liu et al. [11] :

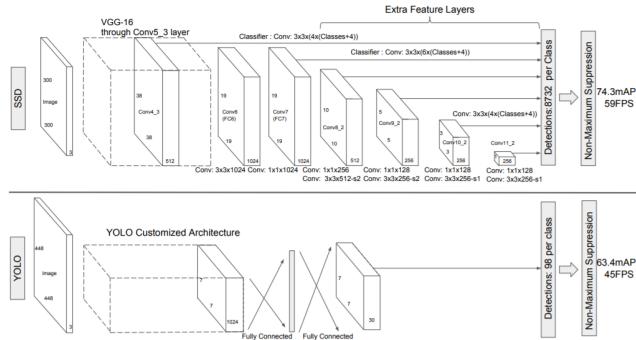


FIGURE 17: Architecture de SSD issue de papier de Wei Liu et al. [11]

Conclusion Single Shot Detector : Cette technique se présente comme une technique rapide et efficace. Un point attrayant de cette méthode est qu'elle n'est pas complexe contrairement à R-CNN par exemple qui se décompose en 3 modules. De plus, on peut facilement trouver des implémentations de cet algorithme sur github ainsi que des réseaux pré-entraînés, que l'on pourra utiliser dans la suite de notre projet.

4 Analyse critique

En complément des informations données dans la précédente partie, nous pouvons comparer analytiquement les précédents algorithmes.

Note : mAP (Mean Average Precision) évalue la précision du système. Plus la valeur est proche de 100% et plus le système se rapproche de la perfection.

Au début de ce papier nous avions parlé de l'importance de la légèreté et la rapidité de l'algorithme qui doit être exécuté sur la configuration de mon ordinateur (Windows 10, CPU Intel(R) Core(TM) i5-7300HQ, GPU Nvidia GeForce GTX1060 6GB Max-Q Design). La rapidité des algorithmes est donc un point essentiel.

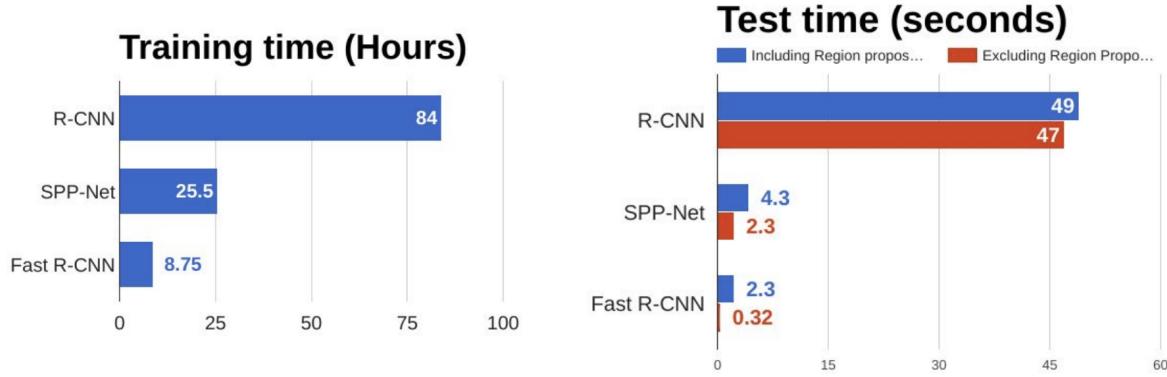


FIGURE 18: Comparaison du temps d'entraînement entre SPP-Net, R-CNN et Fast R-CNN [9]

Je n'ai pas pu trouver l'information pour le temps d'entraînement de Faster R-CNN. Cependant, on peut comparer la rapidité de la méthode proposition de région. Par image, la recherche sélective prend de 3 à 25 secondes, tandis que le RPN prend moins de 100 ms. Le gain de temps n'est donc pas négligeable. Pour SSD je n'ai pas non plus pu trouver l'information. Je pourrais obtenir cette information au cours de mon projet lorsque j'entraînerai moi-même un SSD.

Vis-à-vis du temps de test, R-CNN est le plus mauvais. Pour tester une image, l'algorithme prend 49 secondes, 0.02fps. Fast R-CNN est déjà beaucoup plus rapide et prend 2.3 secondes, 0.43fps pour une image. Faster R-CNN est comme son nom l'indique encore plus rapide. Pour une image, l'algorithme prend 0.2 seconde, 5fps. Avec SSD, on atteint du temps réel, 59 fps, 0.02 seconde, sur une Nvidia Titan X qui est entre 30 et 60% plus efficace que ma carte graphique Nvidia GeForce GTX1060 6GB Max-Q Design.

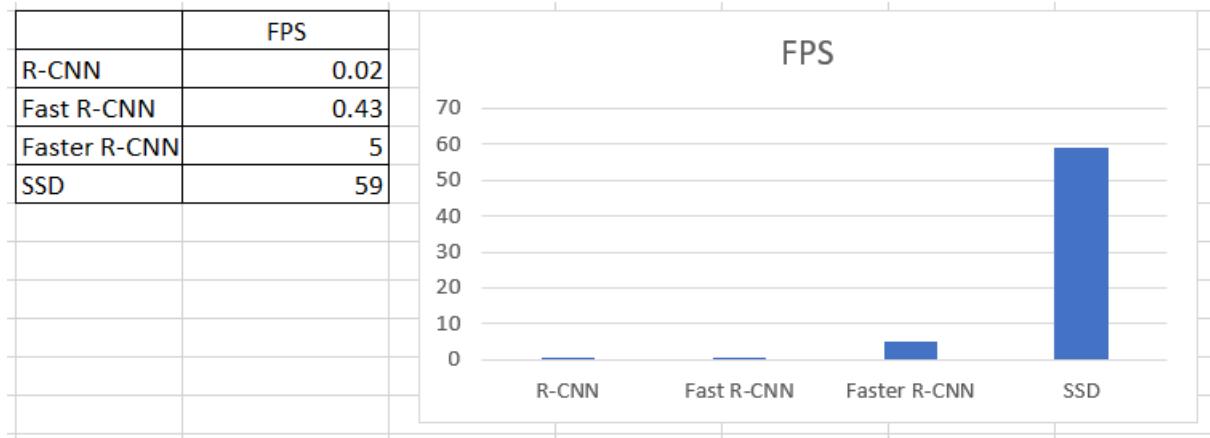


FIGURE 19: Comparaison de la rapidité de test de R-CNN, Fast R-CNN, Faster R-CNN et SSD (les valeurs sont issues de la partie résultat des papiers [11] et [10])

La précision des systèmes est un objectif majeur. Un système rapide, mais dont les résultats sont toujours faux est inutile.

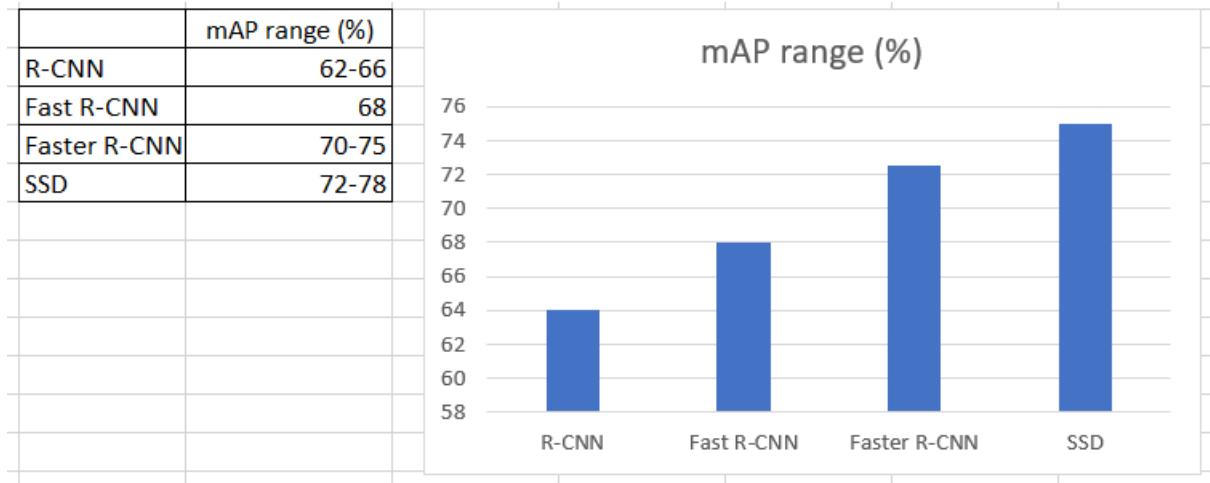


FIGURE 20: Comparaison de la précision de R-CNN, Fast R-CNN, Faster R-CNN et SSD (les valeurs sont issues de la partie résultat des papiers [11] et [10])

SSD et Faster R-CNN donnent les meilleurs résultats et sont les systèmes les plus rapides. Nous allons implémenter ces deux algorithmes en Python.

5 Conclusions

En conclusion, nous pouvons dire que les méthodes de détection d'objets sont nombreuses. On peut expliquer ce nombre par toutes les possibilités de combinaisons d'algorithmes. Beaucoup d'algorithmes de traitement d'image permettent de reconnaître des

objets en utilisant diverses caractéristiques comme Hog, la forme, la texture, ect. Du côté machine learning, beaucoup d'algorithmes permettent également de reconnaître les objets. Des méthodes comme la famille des R-CNN permettent d'allier machine learning à traitement d'image pour donner de bons résultats. Mais on peut toujours itérer pour changer des éléments comme on a pu le voir avec Fast R-CNN et Faster R-CNN.

Pour la suite du projet, on va chercher à synthétiser deux algorithmes, l'un de Faster R-CNN et un autre de SSD. L'objectif étant de comprendre et maîtriser ces algorithmes. Si le temps le permet, j'aimerais chercher à améliorer l'un de ces deux algorithmes.

Database à utiliser : <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#devkit> et <http://cocodataset.org/#home>

Table des figures

1	À gauche une chaise classique [1] et à droite une chaise de designer [2]	3
2	Image d'un chien mignon qui devrait être classifié comme celle d'un chien	5
3	Image qui correspond à un chien, une tasse et une coupole	5
4	Illustration du déroulement de l'algorithme issue de [3]	8
5	Illustration de l'algorithme R-CNN issue de [5]	9
6	Illustration du principe de fonctionnement d'un CNN avec la première région .	10
7	Illustration du principe de fonctionnement d'un CNN avec la seconde région .	11
8	Exemple simple pour expliquer SVM	11
9	Architecture du Fast R-CNN	13
10	Illustration de la RoI pooling layer [8]	14
11	Comparaison entre SPP-Net, R-CNN et Fast R-CNN [9]	15
12	Architecture de Faster R-CNN [10]	16
13	Illustration de l'architecture du RPN [10]	17
14	Comparaison des rapidités d'exécution [9]	17
15	Exemple du type de fichier dont on a besoin pour entraîner SSD	18
16	Exemple issue de papier de Wei Liu et al. [11]	19
17	Architecture de SSD issue de papier de Wei Liu et al. [11]	20
18	Comparaison du temps d'entraînement entre SPP-Net, R-CNN et Fast R-CNN [9]	21
19	Comparaison de la rapidité de test de R-CNN, Fast R-CNN, Faster R-CNN et SSD (les valeurs sont issues de la partie résultat des papiers [11] et [10])	22
20	Comparaison de la précision de R-CNN, Fast R-CNN, Faster R-CNN et SSD (les valeurs sont issues de la partie résultat des papiers [11] et [10])	22

Références

- [1] Callum - chaise de salle à manger. <https://www.habitat.fr/p/callum-chaise-de-salle-a-manger-naturel>. Accessed : 01-29-2019.
- [2] Chaise design original pin up par infiniti. <http://www.archiexpo.fr/prod/vitra/product-80422-684808.html>. Accessed : 01-29-2019.
- [3] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2) :154–171, 2013.
- [4] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2) :167–181, 2004.
- [5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe : Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
- [7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [8] Roi pooling layer illustration. <https://deepsense.ai/region-of-interest-pooling-explained/>. Accessed : 03-08-2019.
- [9] R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>. Accessed : 03-08-2019.
- [10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

- [12] Pascal voc database. <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#devkit>. Accessed : 03-08-2019.
- [13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [15] Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>. Accessed : 01-29-2019.
- [16] Adrian Rosebrock. Object detection with deep learning and opencv. <https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/>. Accessed : 01-29-2019.
- [17] Chien. <https://www.aufeminin.com/news-societe/le-plus-petit-chien-du-monde-mesure-5cm-et-il-est-trop-mignon-s1088519.html>. Accessed : 03-02-2019.