Séminaire de modélisation statistique

Segmentation simultanée et détection d'objets CNNs pour la détection et l'identification d'avions dans les images satellites

Antoine LELONG
Sirine LOUATI

ENSAE 2e année
Année scolaire 2021-2022

Table des matières

I Introduction

La détection et l'identification des objets dans les images satellites peut s'avérer être une tâche très difficile. En effet, les objets d'intérêt sont souvent très petits et leurs caractéristiques peuvent être difficiles à reconnaître, même en utilisant des images à très haute résolution. C'est pour cette raison qu'un arbitrage entre deux métriques très fréquemment utilisées pour ce genre d'application est nécessaire.

En effet, les deux métriques les plus utilisées dans ce secteur sont :

— la précision :

$$\frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Positifs}}$$

Cette métrique mesure la capacité du modèle à ne pas prédire trop de faux positifs parmi tous les positifs qu'il prédit.

— le rappel (recall) :

$$\frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Négatifs}}$$

Cette métrique mesure la fraction de positifs que le modèle arrive à prédire parmi tous les positifs de la base.

Ces deux métriques permettent de qualifier la qualité des algorithmes. Cependant, il est souvent nécessaire d'effectuer un compromis entre les deux afin d'obtenir des prédictions peformantes.
L'objectif de ce document s'inscrit ainsi dans ce contexte et présente une méthode permettant de maximiser ces deux métriques simultanément.

II Cadre général

Au cours de la dernière décennie, les images satellites à haute résolution sont devenues un outil incontournable utilisé pour les tâches de surveillance. Par exemple, lors de la surveillance des sites militaires, il est nécessaire de détecter et d'identifier automatiquement les objets d'intérêt.
Dans ce domaine, la reconnaissance des avions présente un intérêt particulier. En effet, chaque modèle d'avion a son propre rôle, et une variation du nombre d'un type spécifique d'avion à un endroit donné peut constituer une perspective très pertinente. Cette tâche de reconnaissance doit, par conséquent, être fiable pour permettre l'automatisation de l'analyse de site et la détection d'événements inhabituels.

Dans ce contexte, les CNN sont considérés comme l'un des meilleurs modèles pour analyser le contenu des images et sont la technique de Machine Learning la plus utilisée dans les applications de vision par ordinateur.

Les CNN sont un modèle d'apprentissage automatique et plus particulièrement d'ap-

prentissage profond qui consiste à apprendre des représentations de données d'entrée disponibles qui permettent d'approcher au mieux un résultat en sortie. Plus particulièrement en apprentissage profond, l'apprentissage se fait à travers des réseaux de neurones constitués d'une superposition de couches de convolution. Le contrôle de la sortie de ces réseaux de neurones se fait par le moyen d'une fonction de perte qui calcule un score de distance entre la prédiction et le résultat attendu.

Plus précisement, une architecture de type CNN est généralement composée de couches alternées de convolution, appelées encodeur, suivies d'un codeur qui peut comprendre une ou plusieurs couches entièrement connectées (pour la classification), un ensemble de convolutions de transposition (pour la segmentation) ou certaines branches de classification et de régression (pour la détection d'objets). L'agencement des composants CNN joue un rôle fondamental dans la conception de nouvelles architectures et permet ainsi d'améliorer les performances.

La couche de convolution constitue toujours la première couche d'un réseau de neurones. Elle permet de repérer l'existence de certaines caractéristiques (features) d'une image donnée en entrée. Un filtrage par convolution est alors réalisé : le principe est de faire passer un carré représentant la feature sur l'image, et de calculer le produit de convolution entre la feature et chaque portion de l'image selectionnée. Une feature est alors vue comme un filtre (ou encore un masque).
La couche de convolution reçoit ainsi plusieurs images en entrée, et calcule la convolution de chacune d'entre elles avec chacun des filtres, ces derniers correspondant ainsi aux les features dont nous souhaitons vérifier la présence dans l'image.
On obtient alors pour chaque image une carte de caractéristiques, appelée feature map, qui indique la localisation des features dans cette image.

La couche de pooling se trouve généralement entre deux couches de convolutions. Cette couche reçoit donc en entrée plusieurs features maps et leur applique une à une une opération de pooling. Cette opération consiste à retirer certaines informations tout en conservant les caractéristiques les plus importantes et ce afin de réduire la taille des images.
Ceci permet d'obtenir en sortie le même nombre de feature maps que celles obtenues entrée, mais leurs tailles sont réduites.
La couche de pooling permet alors de réduire le nombre de paramètres et donc de diminuer les calculs. Ceci permet ainsi d'améliorer l'efficacité du réseau et de réduire les risques de surapprentissage.

La couche de correction ReLU est basée sur la fonction réelle non linéaire suivante :

$$ReLU(x) = \max(0, x)$$

Cette couche permet ainsi de remplacer toutes les valeurs négatives reçues en entrée par des 0. Il s'agit donc d'une couche qui joue le rôle de fonction d'activation.

III Résumé de l'article de recherche

L'article étudié présente une méthode dédiée à la détection et à l'identification des avions, combinant deux réseaux de neurones convolutifs (CNN) très différents : un modèle de segmentation, basé sur une architecture U-net modifiée, et un modèle de détection, basé sur l'architecture RetinaNet.

Les résultats montrent que cette combinaison surpasse considérablement chaque modèle unitaire, réduisant considérablement le taux de faux négatif.

Nous décrivons chacune de ces deux architectures permettant, une fois combinées, d'obtenir la méthode présentée dans l'article.

III.A Le U-Net

Pour les tâches de segmentation, l'architecture U-net est largement utilisée.

L'architecture du réseau U-net est la suivante : le réseau se compose d'une partie contractante et d'une partie expansive, ce qui explique cette architecture en forme de «U».

La partie contractante, appelée aussi encodeur, consiste en un assemblage répété de couches de convolutions, chacune suivie d'une unité linéaire rectifiée (ReLU) et de couches de pooling maximum (ou maxpooling), permettant de retirer certaines valeurs des couches intermédiaires afin de réduire la taille de l'image et de diminuer le nombres de paramètres. Ceci permet ainsi de capturer le contexte de l'image et d'en créer une feature map.

La partie expansive, appelée aussi décodeur, combine les informations des caractéristiques géographiques et spatiales à travers une séquence de convolutions et de concaténations ascendantes.

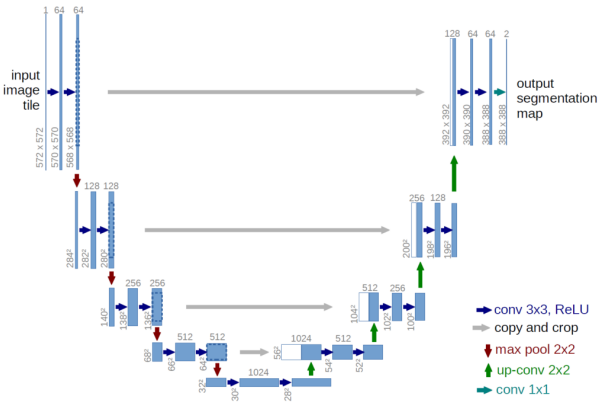


FIGURE 1 – Architecture U-net

L'intérêt d'utiliser des convolutions dans les réseaux de neurones traitant des images est qu'elles permettent de capter les relations de proximité des points d'une images, alors qu'une couche classique de neurones agirait comme si l'image était un vecteur.

Nous décrivons dans cette section plus en détail les différentes composantes de l'architecture U-net :

Les feature maps sont calculées en utilisant des masques, c'est à dire des carrés de poids, appliqués à chaque pixel de l'image et ses voisins. Le même masque s'applique à tous les pixels de l'image, mais on utilise plusieurs masques de manière conjuguée afin d'obtenir plusieurs features maps pouvant capter différentes relations dans l'image. Dans notre UNet, le nombre de feature maps est d'autant plus grand que la taille des images diminue dans la partie contractante.

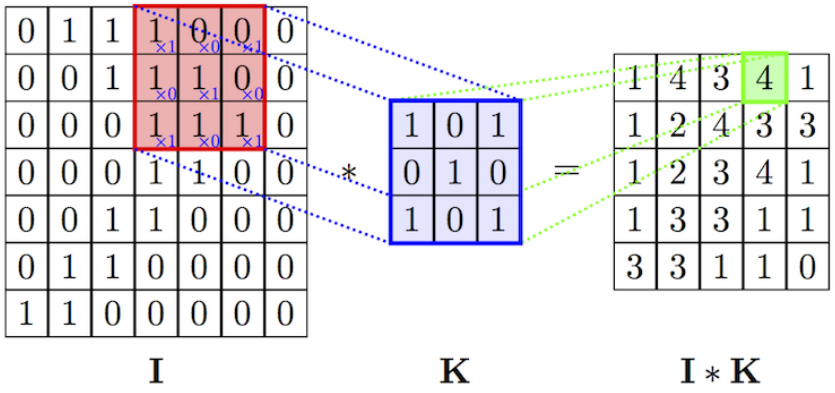


FIGURE 2 – Application d'un masque à une matrice

Cette opération réduit mécaniquement la taille des images, c'est pourquoi on compense cet effet en rajoutant des lignes et des colonnes de zéros sur les bords de l'image avant la convolution afin de conserver une taille d'image constante au cours d'une convolution. Cette compensation s'appelle le padding (de l'anglais rembourrage).

L'opération de maxpool est, comme évoqué precedemment, celle qui réduit la taille des images. Chacune des feature map est sectionnée et on ne retient de chaque section que la valeur maximale. Dans notre cas, les sections sont des carrés de 2×2 pixels, en garder un seul a donc pour effet de diviser par deux les dimensions de l'image. Les features maps auxquelles on applique le maxpool sont également gardées en mémoire pour la partie décodeur du UNet.

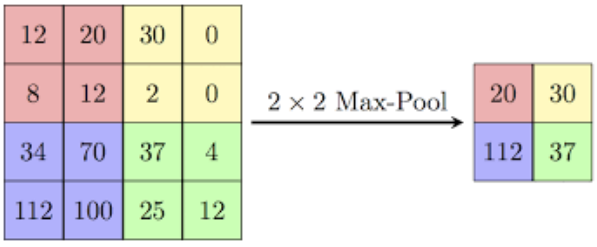


FIGURE 3 – Opération maxpool sur une matrice

Les images sont ensuite aggrandies à nouveau dans la partie décodeur du UNet en utilisant une couche appelée Upsample, qui remplace chaque pixel par un carré de 2×2 pixels contenant la même valeur.

Dans la partie décodeur, on rajoute aux feature maps obtenues par Upsample les feature maps de même taille gardées en mémoire pendant la phase encodeur, et on utilise à nouveau des convolutions pour combiner ces informations.

La sortie du UNet est une ou plusieurs feature maps de la même taille que l'image d'origine. L'entrainement du UNet consiste donc à rapprocher ces sorties des feature maps attendues pour chaque image. Les paramètres entrainables du UNet sont les poids des masques des couches de convolutions. Les couches de maxpooling et d'upsampling sont non paramétriques. Cependant, puisque les masques ne dépendent pas de la taille des images, un même UNet peut s'appliquer (au sens d'être capable de donner une sortie) sans problème sur des images de dimensions très différentes, dès lors que ces dimensions sont divisibles par 2 autant de fois qu'on applique le maxpooling dans la partie contractante. L'architecture classique du UNet divise les dimensions jusque par 16, imposant cette condition sur les images entrantes.

III.B La Retina-Net

Pour les taches de détection, l'architecture la plus performante est la RetinaNet.

RetinaNet est l'un des meilleurs modèles de détection d'objets denses et à petite échelle. Pour cette raison, il est devenu un modèle de détection d'objet populaire à utiliser avec l'imagerie aérienne et satellite.

RetinaNet a été formé en apportant deux améliorations par rapport aux modèles existants de détection d'objets à savoir Feature Pyramid Networks (FPN) et Focal Loss (FL).
Avant d'étudier l'architecture de RetinaNet, nous commençons par décrire le FPN.

Traditionnellement, dans la vision par ordinateur, les pyramides d'images visualisées ont été utilisées pour détecter des objets avec des échelles variables dans une image. Cela signifie que l'on prendrait une image et la sous-échantillonnerait en images de plus petite résolution et de plus petite taille (formant ainsi une pyramide). Des éléments fabriqués à la main seraient ensuite extraits de chaque couche de la pyramide pour détecter les objets. On peut donc imaginer que ce processus exige beaucoup de calcul et de mémoire.

La perte focale (FL) est une version améliorée de la perte d'entropie croisée (CE) qui permet de gérer le problème de déséquilibre de classe en attribuant plus de poids à des exemples difficiles ou facilement mal classés (comme par exemple des objets en arrière-plan avec une texture bruyante ou encore l'objet d'intérêt de l'étude) et en attribuant, en contrepartie, des poids faibles aux exemples faciles (comme par exemple des objets d'arrière-plan d'importance moindre dans l'étude).

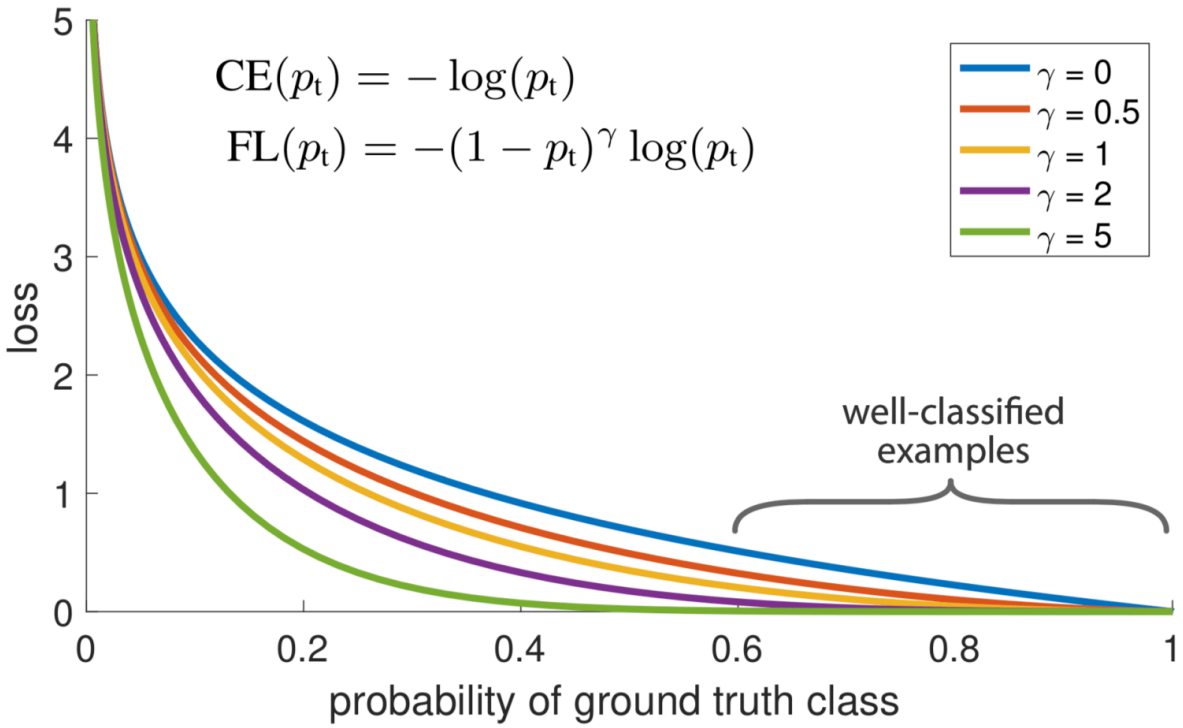


FIGURE 4 – Focal loss vs probability of ground truth class

Dans RetinaNet, à chaque couche de pyramide il peut y avoir des milliers de boîtes d'ancrage (rectangles de délimitations prédéfinies qui peuvent être considérés comme hyperparamètres à optimiser). Cependant, seules certaines de ces boites seront assignées à un objet "ground-truth" tandis que la grande majorité sera classée en arrière-plan ("background"). C'est donc ici qu'intervient la perte focale qui permet d'accorder plus d'importance à la correction des exemples mal classés et de réduire la contribution de perte des exemples faciles qui, même s'ils n'entrainent que de petites valeurs de perte, peuvent collectivement faire submerger le modèle.

RetinaNet consiste ainsi à intègrer le FPN et la FL et à ajouter des sous-réseaux de classification et de régression afin de créer un modèle de détection d'objet.

Il existe quatre composantes principales d'une architecture de modèle RetinaNet :

— **Bottom-up Pathway** : Le réseau de base (par exemple : ResNet) qui calcule les feature maps à différentes échelles, peu importe la taille de l'image d'entrée ou du réseau de base.

— **Top-down pathway and Lateral connections** : La voie du haut vers le bas renverse les feature maps spatialement plus grossières des niveaux pyramidaux supérieurs, et les connexions latérales fusionnent les couches du haut vers le bas et les couches du bas vers le haut avec la même taille spatiale.

Les feature maps de niveau supérieur ont tendance à avoir une petite résolution

bien que sémantiquement plus forte et sont donc mieux adaptées à la détection d'objets plus grands ; au contraire, les boîtes d'ancrage des feature maps de niveau inférieur ont une haute résolution et sont donc mieux adaptées à la détection d'objets plus petits.

Ainsi, avec la combinaison de la voie descendante et de ses connexions latérales avec la voie ascendante, qui ne nécessite pas beaucoup de calcul supplémentaire, chaque niveau des feature maps résultantes peut être à la fois sémantiquement et spatialement performant.

Par conséquent, cette architecture est à l'échelle invariante et peut fournir de meilleures performances en termes de vitesse et de précision.

— **Classification subnetwork** : Il prédit la probabilité qu'un objet soit présent à chaque emplacement spatial pour chaque boîte d'ancrage et chaque classe d'objet.

Un réseau entièrement convolutif (FCN) est rattaché à chaque niveau FPN pour la classification des objets. Comme le montre le diagramme ci-dessous , ce sous-réseau intègre 3*3 couches convolutionnelles avec 256 filtres suivis d'une autre couche convolutionnelle 3*3 avec des filtres K*A. Par conséquent, la feature map de sortie serait de taille W*H*K*A, où W et H sont proportionnels à la largeur et la hauteur de la feature map d'entrée et K et A sont nombre de classe d'objet et de boîtes d'ancrage respectivement.

La raison pour laquelle la dernière couche de convolution a des filtres K*A est expliquée par le fait que s'il y a un nombre A de propositions de boîtes d'ancrage pour chaque position dans la feature map obtenue à partir de la dernière couche de convolution, alors chaque boîte d'ancrage a la possibilité d'être classée dans K nombre de classes. Ainsi la feature map de sortie serait de la taille des K*A filtres.

— **Regression subnetwork** : Il régresse le décalage pour les boîtes limitantes des boîtes d'ancrage pour chaque objet ground-truth.

Le sous-réseau de régression est rattaché à chaque carte d'entités du FPN parallèlement au sous-réseau de classification. La conception du sous-réseau de régression est identique à celle du sous-réseau de classification, sauf que la dernière couche convolutive est de taille 3*3 avec 4 filtres, ce qui donne une feature map de sortie de taille W*H*4*A.

La raison pour laquelle la dernière couche de convolution a 4 filtres est que pour localiser les objets de classe, le sous-réseau de régression produit 4 nombres pour chaque boîte d'ancrage qui prédisent le décalage relatif (en termes de coordonnées centrales, largeur et hauteur) entre la boîte d'ancrage et la boîte de ground-truth. Par conséquent, la feature map de sortie du sous-réseau de régression comporte des filtres de taille 4*A.

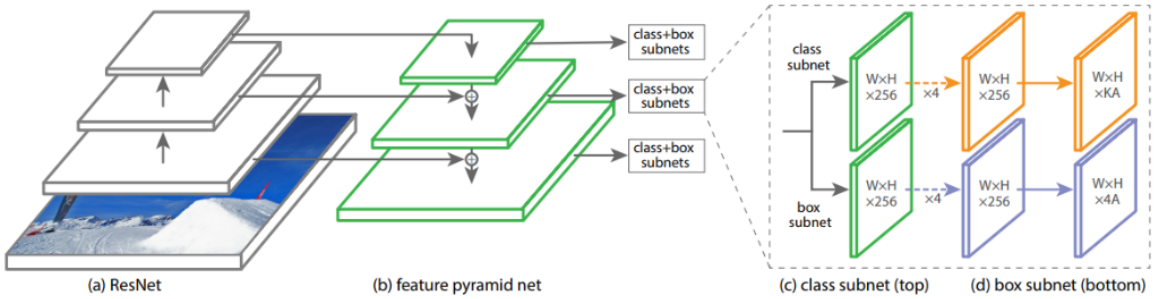


FIGURE 5 – Architecture Retinanet

III.C La SegRetina : L'approche simultanée

La solution hybride est basée sur différentes stratégies CNN : un modèle de segmentation basé sur l'architecture U-Net pour un meilleur taux de détection et un modèle de détection objet basé sur le RetinaNet, un détecteur rapide qui permet d'identifier et d'améliorer la précision.

L'approche simultanée permet de combiner les avantages des deux architectures.
En effet, l'architecture U-Net permet d'avoir un recall très élevé et de détecter un grand nombre d'avions sans les identifier. L'architecture RetinaNet assure quant à elle une précision élevée et permet d'identifier correctement les objets.
Cependant, l'apprentissage du modèle U-Net se fait sur les objets de l'avion, le modèle est alors bon pour localiser les objets mais ne l'est pas pour les séparer ou les reconnaître. L'apprentissage du modèle RetinaNet se fait, quant à lui, lors de l'identification du plus petit avion, le modèle est alors bon pour séparer et reconnaître les avions, mais a une très faible précision de localisation.

L'idée de l'approche simultanée est donc d'utiliser ces propriétés complémentaires pour améliorer les détections. Le processus du système est séquentiel et peut être résumé par les étapes suivantes :

— Appliquer le modèle de segmentation sur l'image inconnue pour extraire la valeur de prédiction pour chaque pixel. Il s'agit de l'étape de localisation.
— Appliquer le détecteur d'objets (uniquement) pour chaque zone positive de l'étape de localisation.
— (Optionnel) Étudier la maintenance des zones positive de la carte de prédiction pour augmenter le recall.

Ceci permet notamment de se focaliser sur les parties positives uniquement et par conséquent de réduire le temps de calcul. Ce processus s'effectue également en une seule fois et est très simple à implementer. On peut également jouer sur l'un des réseaux comme sur l'autre.

La suite de l'article traite d'un exemple de simulation de cette approche simultanée.

La méthode a été appliquée au problème de la reconnaissance des avions. Les ensembles de données comportent trois niveaux d'identification des avions : le premier est le type d'objet (« avion »), le deuxième représente la fonction de l'avion (« bombardier », « civil », « combat », « drone », « spécial » et « transport ») et le troisième niveau est l'identification de l'avion. Ce dernier niveau est actuellement composé de 61 classes (par exemple, « F-16 Fighting Falcon » est un troisième niveau de type « combat » et « Tupolev Tu-95 » un troisième niveau de type « bombardier »).

Comme prévu, la méthode simultanée a permis d'augmenter significativement les résultats de détection par rapport au modèle de segmentation ou au modèle de détection seul : les erreurs de chaque modèle sont corrigées par l'autre pour obtenir de meilleurs résultats. Les faux positifs obtenus avec le modèle de détection d'objet sont supprimés par la méthode simultanée.
On peut donc conclure que cette méthode permet effectivement de maximiser les deux métriques de précision de de recall simultanément et d'augmenter ainsi la performance du modèle.

IV Complément de la séance du 15 avril

Dans cette section nous présentons quelques remarques et explications pertinentes données lors de la présentation de Madame Bisot et de Monsieur Giruzzi.

- La convolution est basée sur des filtres qui aboutissent à des features map. L'algorithme choisit les filtres auxquels sont associés des paramètres à optimiser. La séléction des filtres est la base de la construction des réseaux.

- Le Max Pooling est un échantillonnage qui permet de retirer certaines valeurs des couches intermédiaires afin de réduire la taille des images tout en conservant les caractéristiques les plus importantes.

- L'approche SegRetina permet de combiner deux procédures :
  - la détection d'objets qui permet de faire des "boxs" (des contours) autour des objets.
  - la segmentation qui permet de classer chaque pixel en tant qu'objet ou pas.

- La segmentation selon l'architecture U-Net assure une bonne localisation mais est difficile à exploiter si on cherche à classifier les objets.

- La détection selon l'architecture Retina permet de mieux classifier les objets mais assure une moins bonne localisation.

- Le SegRetina est alors une combinaison de ces deux architectures. En effet, U-Net permet de détecter les objets et Retina opère par la suite uniquement sur les tuiles positives ce qui permet d'augmenter la précision et de diminuer le temps de calcul.

V Implémentation d'un UNet

V.A Présentation des données

Les jeux de données d'images satellites haute qualité d'avions identifiés sont difficiles à trouver, nous avons donc choisi à la place un jeu de données de photographies d'animaux appelé le "Hoofed Animals Dataset". Ce dernier présente l'avantage d'avoir 6 classes d'animaux différents et plusieurs animaux sur la plupart des images, contrairement à beaucoup de jeux de données classiques pour la reconnaissance d'objet, où un seul objet occupe généralement la quasi intégralité d'une image.

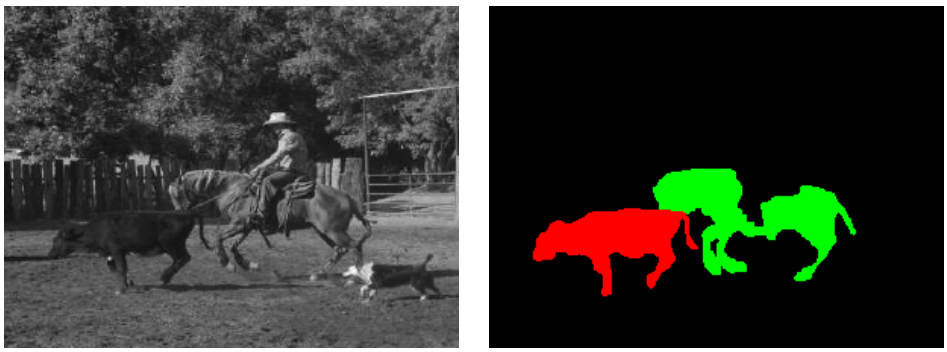


Image Masque

FIGURE 6 – Exemple de donnée

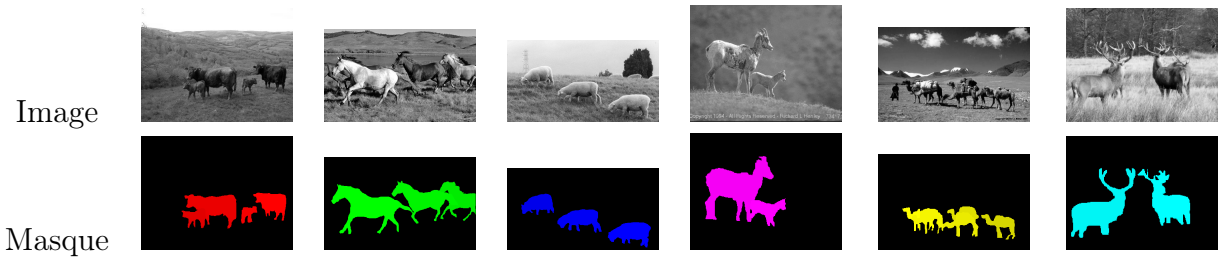


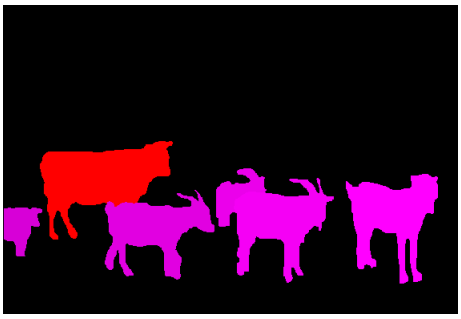
FIGURE 7 – Différentes classes d'animaux

Les différents types d'animaux sont identifiés par des couleurs différentes, et les différentes instances sont distinguées dans le masque en réduisant la valeur RGB des couleurs de 10 pour chaque nouvelle instance.

Ce jeu de données présente cependant quelques défauts. D'une part, les dimensions des images sont très variables, allant de 182×149 pixels pour les plus petites, jusqu'à 1353×910 pixels. Un deuxième défaut est qu'en cas de très nombreuses instances sur une image, seule les quelques plus visibles seront dans le masque objectif.

Enfin, dans la majorité des cas, les animaux sont très proches, ce qui rend la délimitation entre deux instances distinctes particulièrement compliquée.

Exemple d'image



Masque correspondant donné dans la base de données

FIGURE 8

V.B Entrainement du UNet

Preprocessing

Avant de pouvoir passer dans le UNet, il a fallu ajouter à chaque image un padding, c'est à dire des lignes et des colonnes de 0 afin que leurs dimensions soient des multiples de 16 (puisqu'il faut diviser cette dimension par deux à chaque étape de contraction du UNet).

L'entrainement du UNet étant également très lent en CPU, il a fallu également se restreindre aux images les plus petites afin que le calcul sur GPU puisse supporter la quantité de données.

Problème à optimiser

Le UNet est utilisé uniquement pour repérer les zones d'intérêt et pas pour faire la distinction entre les instances ou les différents types.

Les masques utilisés sont par conséquent simplement les masques retirant les distinctions.

Nous avons essayé deux fonctions de pertes pour entraîner le UNet. La première consiste à faire le ratio entre l'intersection et l'union des zones détectées et des zones attendues :

$$Loss(f(X), Y) = 1 - \frac{f(X) \times Y}{f(X) + Y - f(X) \times Y}$$

Cela revient à dire en termes de zones détectées :

$$Loss(f(X), Y) = \frac{\mathcal{A}\left(f(X) \cap Y\right)}{\mathcal{A}(f(X) \cup Y)}$$

La deuxième est une loss très utilisée avec les UNets appelée la Dice Loss :

$$DiceLoss_b\left(f(X), Y\right) = 1 - \frac{2f(X) \times Y + b}{f(X) + Y + b}$$

Cela revient à dire en termes de zones détectées :

$$DiceLoss_b(f(X), Y) = \frac{2\mathcal{A}\left(f(X) \cap Y\right) + b}{\mathcal{A}\left(f(X)\right) + \mathcal{A}(Y) + b}$$

Le paramètre b permet de lisser le gradient afin de faciliter la descente de gradient. Nous fixons arbitrairement $b = 1$ lors de notre entrainement, et on ne cherchera pas à optimiser l'entrainement selon ce paramètre, mais cela peut être une piste d'amélioration.

V.C Résultats

De la même manière que l'article étudié, le UNet a un rappel élevé, quitte à créer davantage de zones positives que nécessaire et à entourer grossièrement, car les zones positives permettront de corriger les manques de rappel d'un RetinaNet, qui pourra alors simplement trier les zones détectées.

Sur des images reformatées à la même taille que le jeu d'entraînement, à savoir 336×512 pixels, on obtient des résultats assez grossiers, mais satisfaisants, au sens que la grande majorité des instances sont correctement détectées. Ces sorties, couplées à l'image originale devraient permettre au RetinaNet de comprendre sur quelles zones se focaliser.

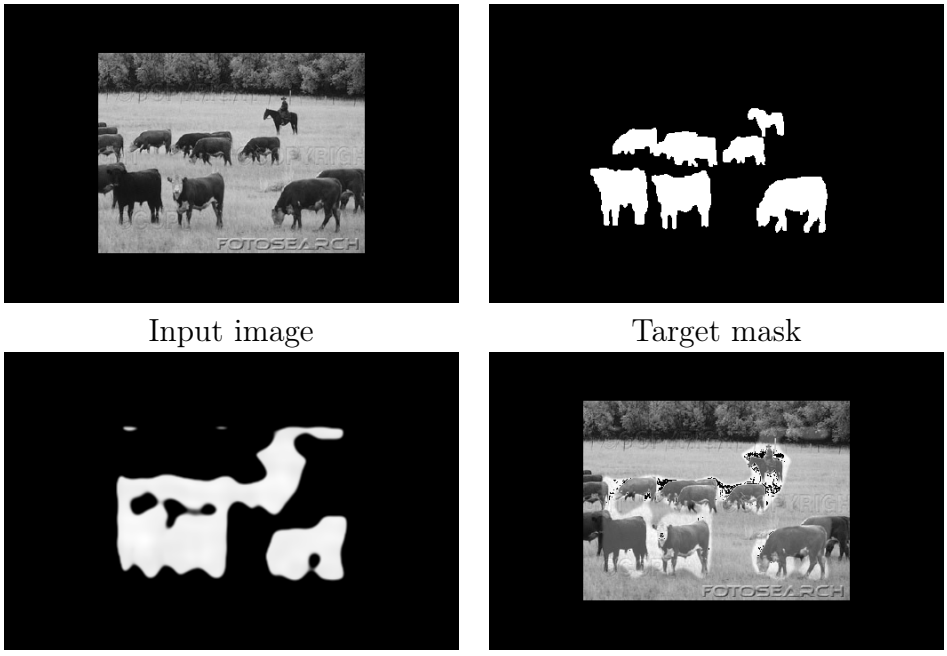
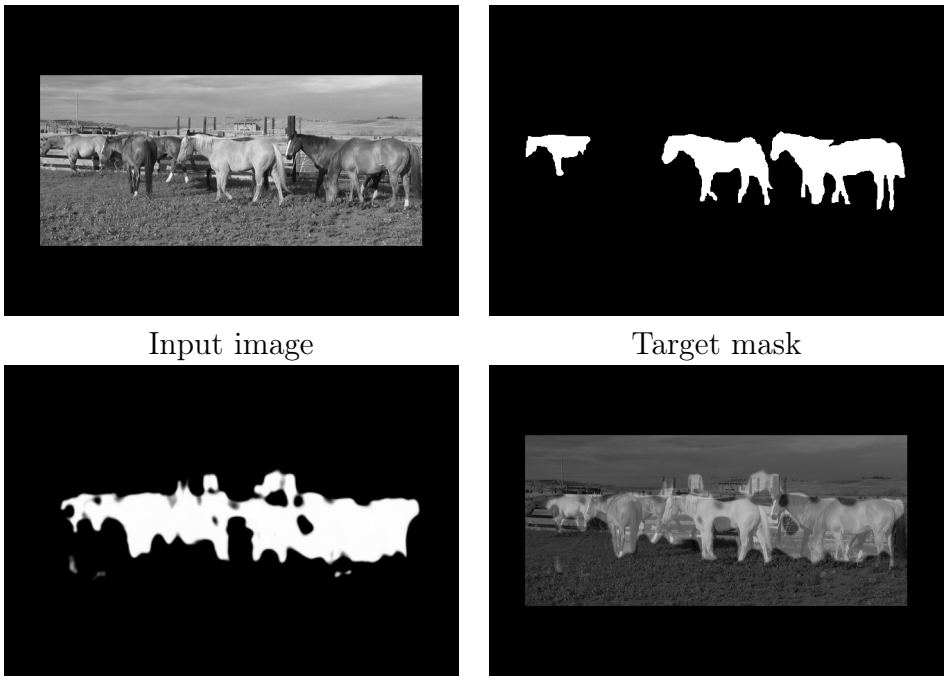


FIGURE 9 – Résultats du UNet sur une image de taille $\leq 336 \times 512$

De plus, le passage à des images de plus grande dimension (par conséquent calculées sur CPU) se passe relativement bien : les sorties pour les petites images aggrandies à la nouvelle taille de 448×656 sont identiques, et pour les nouvelles images, les résultats sont de bonne qualité, même si on remarque tout de même un effet de bord, empêchant sans un entrainement supplémentaire sur les images redimensionnées, de détecter les animaux en bordure d'image.

L'utilisation de la DiceLoss a permis une convergence plus rapide que la Loss naïve, pour des résultats quasiment identiques. L'entrainement dans les deux cas est néanmoins long car nous sommes partis de réseaux de neuronnes sans préentrainement.



Input image | Target mask

Predicted mask | Superposition

FIGURE 10 – Résultats du UNet sur une image de taille $\leq 448 \times 656$

On peut même remarquer sur la figure 10 que notre UNet détecte un cheval qui n'est pas présent dans le masque cible, pointant à la fois un manque de qualité des données déjà mentionné, mais surtout que le modèle a bien compris comment détecter en l'occurence les chevaux, et n'a pas (ou pas trop) surappris.

VI Conclusion

Par le biais l'article étudié, nous avons découvert des techniques de computer vision permettant de détecter, puis identifier des motifs dans des images, des modèles d'avions dans le cas de l'article, différentes espèces d'animaux dans notre implémentation. La combinaison de deux réseaux de neurones aux architectures différentes permet de compenser les faiblesses de chaque modèle tout en gardant l'intégralité de leurs qualités respectives, à savoir un fort rappel pour le UNet et une précision élevée pour le RetinaNet. Le UNet que nous avons implémenté a, en adéquation avec les résultats de l'article, un rappel élevé,

quitte à avoir beaucoup de faux positifs se manifestant sous la forme d'un entourage grossier des zones à détecter, mais discrimine malgré tout les différentes zones des images. L'entrainement de ce UNet n'a pas nécessité un grand nombre d'image et se généralise pourtant bien aux images de format très différent du reste de la base de données d'origine.

VII Annexe

VII.A Lien du repository GitHub

L'ensemble du code nous ayant permis de produire les résultats de notre étude est disponible dans le *repository* GitHub suivant :
https://github.com/AntoineLlg/Instance-Detection-Hoofed-Animals