République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'étude Master

Filière : Informatique

Option : Sciences et Technologie de l'Information et de la Communication

Thème :

Détection d'objets basé Faster R-CNN

Encadré Par :

Dr. Hallaci Samir

Présenté par :

Chouini Mohammed El Mounsif

Octobre 2020

Résumé

La détection d'objets est un sous-domaine de la vision par ordinateur qui est actuellement fortement basé sur l'apprentissage automatique. Au cours de la dernière décennie, le domaine de l'apprentissage automatique a été dominé par les réseaux de neurones dits profonds, qui tirent parti des améliorations de la puissance de calcul et de la disponibilité des données. Un sous-type de réseau neuronal appelé réseau neuronal convolutif (CNN) est bien adapté aux tâches liées à l'image. Le CNN est formé pour rechercher différentes caractéristiques, telles que les contours et les différences de couleur, à travers l'image et pour les combiner en des formes plus complexes. Pour la détection d'objets, le système doit à la fois estimer les emplacements des objets probables et les classer. Pour ce mémoire, nous avons passé en revue l'état actuel de la technique sur la détection d'objets par convolution et testé la possibilité d'implémentation de l'une des méthodes appelée Faster R-CNN. Grâce à la disponibilité gratuite des ensembles de données et des modèles pré-entraînés, il est possible de créer une implémentation fonctionnelle d'un réseau neuronal profond sans accès à du matériel spécialisé. Les réseaux pré-entrainés peuvent également être utilisés comme point de départ pour former de nouveaux réseaux, ce qui réduit le temps de formation coûteux.

Pour la partie expérimentale, nous avons implémenté Faster R-CNN et testé le modèle sur des images externes différents liés à la circulation routière. Nous avons constaté que Faster R-CNN est relativement précis avec un taux de précision **86%**.

Mots clés : Vision par ordinateur, Détection d'objets, Apprentissage automatique, réseaux de neurones convolutifs, Faster R-CNN.

Remerciement

Je veux exprimer par ces quelques lignes de remerciement notre gratitude envers tout d'abord mon encadrant, Monsieur HALLACI Samir pour ses conseils, son encadrement et son assistance tout au long de cette année.

Enfin, je tiens à remercier également toutes les personnes qui ont participé de près ou de loin au bon déroulement de ce projet de fin d'étude.

Dédicace

A

Mes parents

Les plus chers au monde qui ont œuvrés pour ma réussite et qui n'ont jamais cessé de m'encourager et de me soutenir. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent. Je vous souhaite une longue vie plein de bonheur.

A

Mes sœurs et mon frère

Qui ont toujours été là pour moi, des exemples de persévérance, de courage et de générosité, pour leurs encouragements continus.

A

Mes amis

Et tous ceux que j'aime et me sont chers et que j'ai omis de citer, tout en leur souhaitant la réussite dans tout ce qu'ils entreprennent.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les systèmes informatiques doivent comprendre une scène visuelle afin d'effectuer diverses tâches. Les humains, dotés de sens, peuvent interpréter efficacement et inconsciemment les informations visuelles pour prendre des décisions éclairées. Cependant, pour les systèmes informatiques, cela signifie franchir le trou sémantique entre les informations au niveau des pixels et la perception humaine des informations visuelles. La vision par ordinateur comble cet écart.

L'extraction d'informations sémantiques à partir d'images est l'un des problèmes les plus fondamentaux et les plus difficiles de la vision par ordinateur. La reconnaissance d'images, la détection d'objets et la segmentation d'instances sont les grandes catégories de la vision par ordinateur. La reconnaissance, également appelée problème de classification, cela est le processus d'identification et de validation des objets dans une image et de les classer dans certaines classes. La détection, cependant, est le processus d'identification et d'étiquetage de plusieurs objets pertinents dans une seule image, ainsi qu'une estimation approximative de l'emplacement et de la taille de cet objet. La segmentation d'instance est la combinaison de tâches de détection d'objet classiques avec une segmentation sémantique, où le but est de classer chaque pixel en un ensemble fixe de catégories sans différencier les instances d'objet.

Avant l'application des réseaux de neurones profonds pour les tâches de détection et de segmentation d'objets, les cadres de détection d'objets classiques étaient principalement basés sur l'extraction de descripteurs de fonctionnalités tels que les histogrammes de gradients orientés (HOG) (1), l'algorithme ViolaJones (2) et la transformation de caractéristiques visuelles invariante à l'échelle (SIFT) (3). Ces détecteurs d'objets classiques reposaient fortement sur des fonctionnalités codées à la main (hand-coded).

Avec des avancées significatives dans les méthodes d'apprentissage profond, la demande de systèmes de détection d'objets rapides et précis augmente. Au cours des dernières années, des efforts considérables ont été investis dans l'augmentation des réseaux de neurones convolutifs (CNN) pour proposer des régions qui encapsulent des objets d'intérêt dans une seule image, Ces réseaux sont appelés réseaux de neurones profonds basés sur la région (R-CNN). Ces réseaux sont constitués de trois modules de base: génération de proposition de région, classification et régression du cadre de délimitation (bounding-box). Les premiers

modèles de détection d'objets basé régions tels que le réseau neuronal convolutif basé région (R-CNN) (4) et le réseau neuronal convolutif basé sur Fast R-CNN (5) ont déployé des algorithmes externes pour la tâche de génération de propositions de région. Recherche sélective (6), propositions d'objets indépendantes de la catégorie (Category Independent Object Proposals) (7), Constrained Parametric Min-Cuts (CPMC) (8), regroupement combinatoire à plusieurs échelles (9), détection des boîtes de bord (Edge Box Detection) (10)etc. sont quelques exemples de ces algorithmes externes. Une fois que ces algorithmes proposent de telles régions, des réseaux de neurones convolutifs (CNNs) sont utilisés pour calculer des caractéristiques très discriminantes dans chaque région, puis les classifieurs sont entrainés pour étiqueter les objets pertinents dans ces régions.

Même si les premiers détecteurs d'objets basés sur la région étaient intuitifs, ils présentaient un goulot d'étranglement en termes de durée d'exécution. Avec l'augmentation du volume de données et des ressources de calcul, des techniques de détection plus rapides et robustes sont devenues une exigence. Par conséquent, un réseau de neurone convolutif Faster R-CNN (11) et un réseau neuronal convolutif Mask R-CNN (12) ont été développés. Ces modèles ne nécessitent aucun algorithme externe pour la tâche de génération de proposition de région. Au lieu de cela, ils utilisent des couches convolutives du réseau pour la génération de propositions de région, réduisant son coût marginal et rendant la détection encore plus rapide. L'objectif de ce mémoire est d'analyser l'état de l'art des performances de l'architecture réseaux de neurones convolutifs basé région. Le travail présenté vise également à trouver une vitesse-précision compromis afin de détecter et segmenter les objets avec un taux de détection et précision élevé.

Structure:

Le présent rapport est composé de trois chapitres structurés comme suit :

- Chapitre 1 fournit une brève introduction au contexte théorique de l'apprentissage automatique et explique les concepts fondamentaux associés à l'apprentissage profond pour la détection d'objets.
- Le chapitre 2 est consacré aux méthodes adoptées dans ce mémoire et les différentes méthodes classiques pour la détection d'objet.
- Le chapitre 3 fournit l'évaluation expérimentale ainsi qu'à l'analyse des résultats

Et enfin, nous terminerons ce mémoire par une conclusion générale et quelques perspectives.

Chapitre 1 : Détection d'objet et Deep learning

Le chapitre 1 fournit une brève introduction au contexte théorique de l'apprentissage automatique et explique les concepts fondamentaux associés à l'apprentissage profond pour la détection d'objets utilisant des architectures de réseaux de neurones convolutifs.

1.1 Machine Learning :

Le terme «Machine Learning» ou bien «apprentissage automatique» a été utilisé pour la première fois par Arthur Samuel en 1959. Il l'a défini comme «un domaine d'étude qui donne à l'ordinateur la possibilité d'apprendre sans être explicitement programmé». Selon cette définition, l'apprentissage automatique peut être considéré comme une pléthore d'algorithmes qui «apprennent» à reconnaître des modèles dans un ensemble de données d'entrainement sans être explicitement programmés. En général, l'apprentissage peut être de deux types: **supervisé** et **non-supervisé.** La différence principale entre les deux types est que l'apprentissage supervisé nécessite les données étiquetées pour entrainer les algorithmes, tandis que l'apprentissage non-supervisé ne nécessite pas les données étiquetées.
Goodfellow et al. (13) Définit l'apprentissage non supervisé comme une technique d'apprentissage implicite ou explicite de la distribution de probabilité $p(x)$ en observant plusieurs exemples d'un vecteur aléatoire$x$.

Réseaux de neurones artificiels (ANNs) :
Les réseaux de neurones artificiels (ANNs) sont la classe des algorithmes d'apprentissage automatique, qui sont fortement inspirés par les opérations du système nerveux biologique. Un ANN est composé d'un grand nombre de petites unités interconnectées (appelées neurones). Un neurone artificiel (figure 1.1), introduit pour la première fois par Warren McCulloch et Walter Pitts en 1943 (14), est l'unité de base d'un ANN. Le neurone est activé si la somme des entrées binaires est supérieure au seuil θ. Pour le seuil $\theta > 0$, la fonction d'activation f est définie comme

$$f(x_1, \dots, x_n) = \begin{cases} 1, \text{ if } \sum_{i=1}^{n} x_1 \geq \theta, \\ 0, \text{ sinon} \end{cases}$$

Ces neurones sont généralement segmentés en plusieurs couches. Une couche d'entrée prend les informations comme un vecteur multidimensionnel et les distribue à une série de couches cachées. Les informations sont ensuite transformées et diffusées (les poids et les biais) à travers les couches cachées. Enfin, la dernière couche génère certaines caractéristiques abstraites des informations d'entrée. Un ANN optimise ces caractéristiques entraînables grâce à une fonction de perte différenciable, dans le but d'améliorer la ressemblance des étiquettes cibles et les prévisions générées par l'ANN.

La structure de base d'un ANN est illustrée à la figure 1.1 :

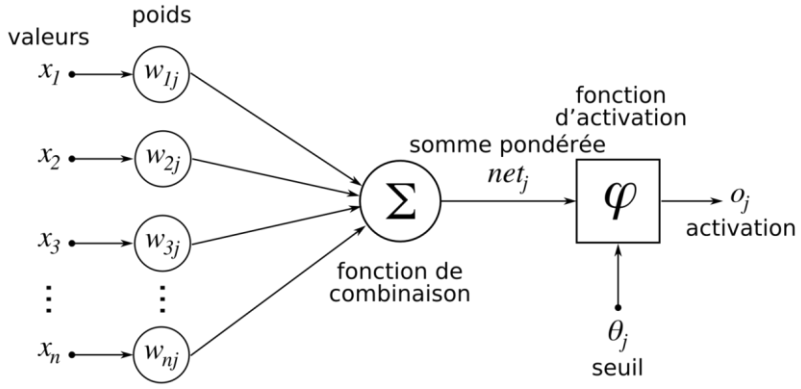


Figure 1.1: Un neurone de McCulloch-Pitts (15)

1.2 Deep Learning :

Le Deep Learning ou apprentissage profond peut être considéré comme faisant partie d'une famille plus large de Machine Learning, qui permet aux systèmes informatiques de transformer des concepts plus simples en concepts plus abstraits et complexes.

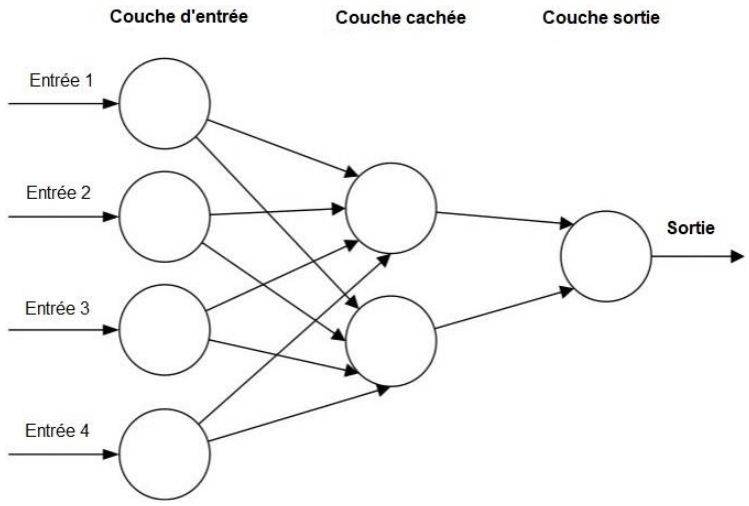


Figure 1.2 : ANN composé d'une couche d'entrée, cachée et d'une couche de sortie (16)

Les réseaux de neurones profonds multicouches existent depuis les années 1980, mais ces dernières années, les développements dans le calcul puissant et la disponibilité de grandes bases de données (Datasets) ont accru la popularité des réseaux de neurones profonds. Avec l'arrivée des unités de traitement graphique (GPU), l'entrainement de réseaux de neurones profonds avec une plus grande efficacité est devenu une réalité. Contrairement aux systèmes de reconnaissance classiques, l'apprentissage profond minimise considérablement le besoin de solutions d'apprentissage automatique fabriquées à la main (hand-crafted).

1.3 Réseaux de neurones convolutifs (CNNs) :

Les réseaux de neurones convolutifs (CNNs) sont la forme la plus impressionnante de réseau de neurones artificiels (ANNs). Le CNN est le composant clé de l'apprentissage profond, qui est principalement utilisé pour résoudre les tâches difficiles de reconnaissance des formes basées sur l'image (16).

L'idée de base de CNN a été inspirée par une caractéristique du cortex visuel animal appelée champ récepteur (17). Les champs récepteurs agissent comme des détecteurs sensibles à certains types de stimulus, par exemple les bords. En traitement d'image, le même type d'effets visibles peut être produit par filtrage par convolution (17)

Un CNN typique est construit par la répétition de trois types de couches de base, c'est à dire les couches convolutionnelles (convolutional layers), les couches de mise en commun (pooling layers) et les couches entièrement connectées (fully connected layers). Un réseau

neuronal profond empile un grand nombre de ces couches pour effectuer des tâches liées à la reconnaissance et à la détection de formes. Une architecture CNN simplifiée pour la classification d'images manuscrites MNIST (18) est illustrée à la figure 1.3.

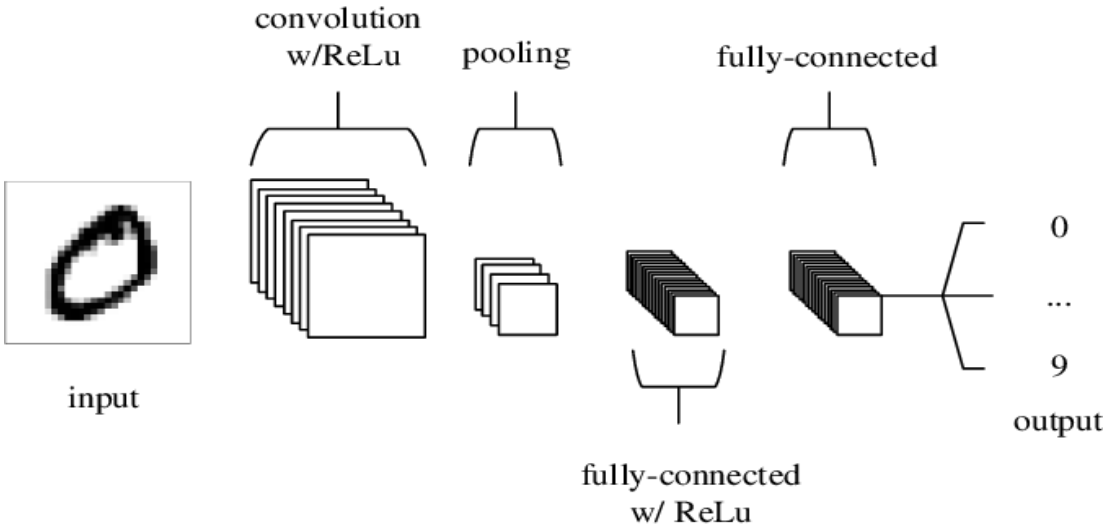


Figure 1.3 : Une architecture CNN, composée de cinq couches (16).

Même si les CNNs ont attiré beaucoup d'attention ces dernières années, leur histoire remonte aux années 1980. Le premier algorithme d'apprentissage supervisé utilisant l'algorithme du gradient (gradient descent) (19) a été créé par Rumelhart et al. en 1986, qui a ensuite été utilisé par LeCun et al. pour la reconnaissance des chiffres manuscrits (20). Cependant, cet algorithme a souffert de multiples problèmes de performances. L'un des problèmes concernait la faible invariance intégrée car elle ne permettait pas de gérer la variabilité des échantillons d'écriture manuscrite lorsqu'ils étaient soumis à des translations ou à des distorsions. Cela a conduit à la création d'un nouveau modèle avec une invariance de décalage plus forte qui répond aux hiérarchies des caractéristiques locales, Ce nouveau réseau a été appelé le réseau neuronal convolutif (CNN) (21). Les CNNs ont été abandonnés juste après en raison du manque de machines puissantes et remplacés par des machines à vecteurs de support (SVM). Cependant, le développement de GPUs puissants au cours de la dernière décennie a ravivé l'espoir dans les capacités de perception des CNNs.

Maintenant, les CNNs sont utilisés comme approche par défaut pour résoudre de nombreux problèmes de traitement d’image.

1.4 Fonctionnement d'un CNN :

Un CNN est composé de plusieurs couches d'opérations, chacune avec sa propre fonction. Le CNN prend une image en entrée et la transmet à la première couche. Les informations sur les entités sont propagées à travers les couches cachées. Pour chaque couche, les fonctions d'activation effectuent une activation élément par élément à la sortie produite par la couche précédente. La sortie de la couche finale est ensuite comparée à la sortie cible. Ce processus est appelé passage direct (forward pass).

1.4.1 Couche convolutionnelle (CONV) :

La couche convolutionnelle joue un rôle important dans le fonctionnement des CNN. Les paramètres de cette couche se concentrent sur de petites fenêtres $n \times n$ appelées noyaux (kernels) ou simplement filtres. La couche convolutive convolutionne chaque filtre ou noyau à travers la dimensionnalité spatiale de l'entrée pour produire une carte d'entités 2D (feature map) ou une carte d'activation (activation map) après le calcul de la fonction d'activation.

Habituellement, l'opération de convolution est définie comme une intégrale qui exprime la quantité de chevauchement d'une fonction g lorsqu'elle est décalée sur une autre fonction f, donnée par :

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau,$$

Où f et g sont les fonctions arbitraires continues dans le domaine t. Dans le cas d'une image, l'opération de convolution est définie comme une fonction discrète. L'opération de convolution discrète entre une image f et une matrice de filtre g est définie comme :

$$f[n] * g[n] = \sum_{m=-\infty}^{+\infty} f[m] g[n - m].$$

La carte des caractéristiques (feature map) est la matrice de sortie $f[n] * g[n]$ obtenue en alignant le filtre successivement avec chaque sous-image de f (de mêmes dimensions que g) centrée sur les coordonnées x, y.

La couche convolutionnelle d'un réseau de neurones est une combinaison de plusieurs filtres convolutionnels, dont les valeurs matricielles sont traitées comme des paramètres neuronaux. La répétition successive des couches convolutives avec certains autres types de couches comme la couche de mise en commun (pooling) se traduit par un réseau neuronal convolutif (CNN) (16).

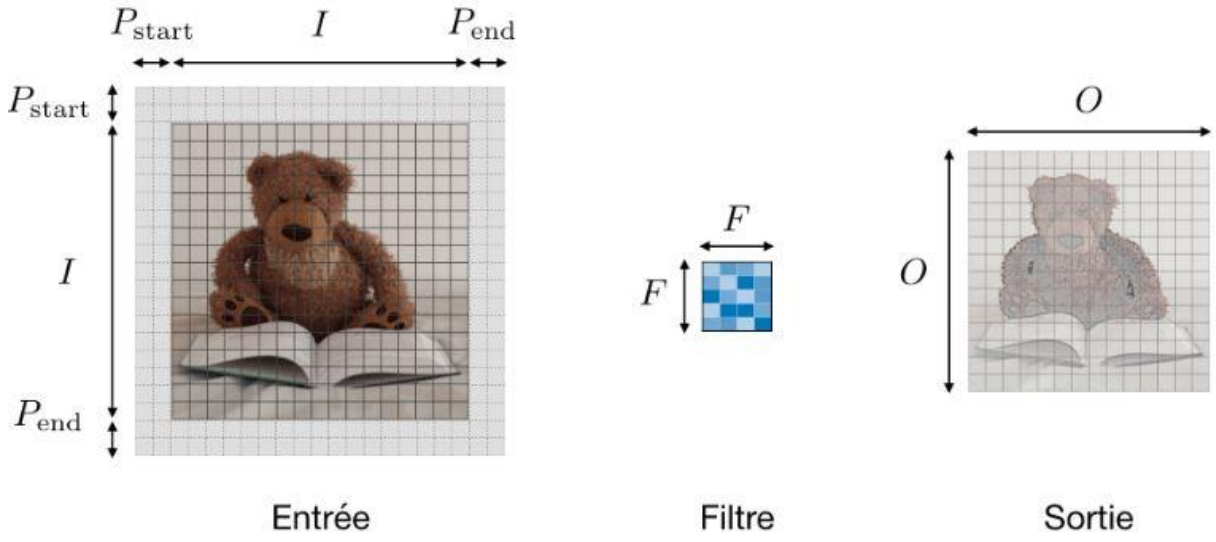


Figure 1.4: Feature map après filtre de convolution.

1.4.2 Couche de Pooling (POOL) :

En anglais pooling layer, pour ajouter une invariance spatiale et rendre le réseau plus facile à gérer pour le traitement d'image, il est utile de réduire la taille d'*activation map*. Par conséquent, l'opération de pooling d'un réseau de neurones est utilisée après une couche convolutionnelle pour réduire progressivement la dimension d'*activation map*. Les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises, respectivement. La couche de pooling, lorsqu'elle est appliquée, réduit le nombre de paramètres et la complexité de calcul du modèle (16). La méthode de pooling la plus couramment utilisée est la méthode max. Cette méthode génère la valeur maximale dans un voisinage rectangulaire d'*activation map* (13). La même chose pour la méthode average sauf qu'elle génère la valeur moyenne. La différence entre les deux types avec stride 2 (la stride est un paramètre qui dénote le nombre de pixels par lesquels la fenêtre se déplace après chaque opération) est illustrée à la figure 1.5.

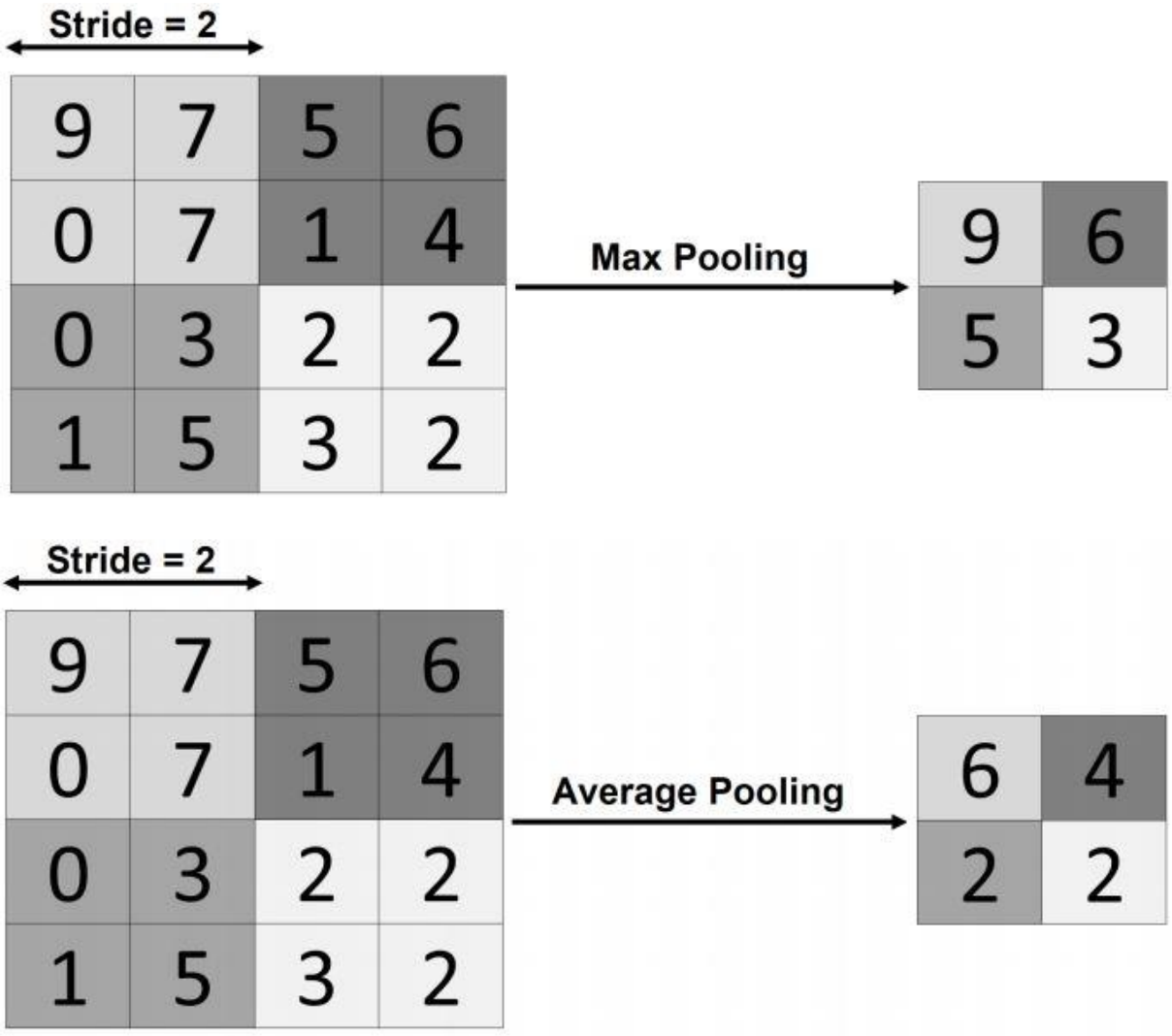


Figure 1.5 : Max-pooling vs average pooling avec stride 2.

1.4.3 La couche Fully Connected (FC) :

La couche de fully connected (en anglais fully connected layer) (FC) s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones. Les couches de fully connected sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe.

1.4.4 Fonctions d'activation

La fonction d'activation φ est appliquée à la sortie d'une couche de convolution pour limiter la sortie de chaque neurone et pour introduire des non-linéarités aux activations linéaires

générées par une couche de convolution. Quelques exemples de certaines des fonctions d'activation populaires utilisées dans les réseaux de neurones pour les tâches de classification et de détection sont donnés comme suit:

1.4.4.1 Unité linéaire rectifiée (ReLU)

La couche d'unité linéaire rectifiée (en anglais rectified linear unit layer) est l'une des fonctions d'activation les plus couramment utilisées dans l'apprentissage en profond, qui est calculée comme suit:

$$f(x) = \max(0, x).$$

Les fonctions d'activation ReLU sont très populaires pour la création d'un réseau non linéaire car il est plus facile de se différencier pour la rétropropagation du gradient. Même si la fonction ReLU n'est pas différenciable à zéro contrairement à la fonction d'activation sigmoïdale, qui a des dérivées lisses, elle converge toujours plus rapidement que la fonction tangente sigmoïde et hyperbolique.

1.4.4.2 Unité linéaire rectifiée Leaky (Leaky ReLU)

La fonction d'activation Leaky ReLU est similaire à la fonction ReLU avec une différence clé: elle permet un petit gradient positif lorsque l'unité est inactive. Un Leaky ReLU est calculé comme suit :

$$f(x) = \max(x, ax).$$

1.4.4.3 La fonction sigmoïde

La fonction Sigmoïde était la fonction d'activation la plus utilisée avant ReLU, qui prend les valeurs réelles en entrée et sort les valeurs dans l'intervalle$[0, 1]$. La fonction sigmoïde est définie comme suit :

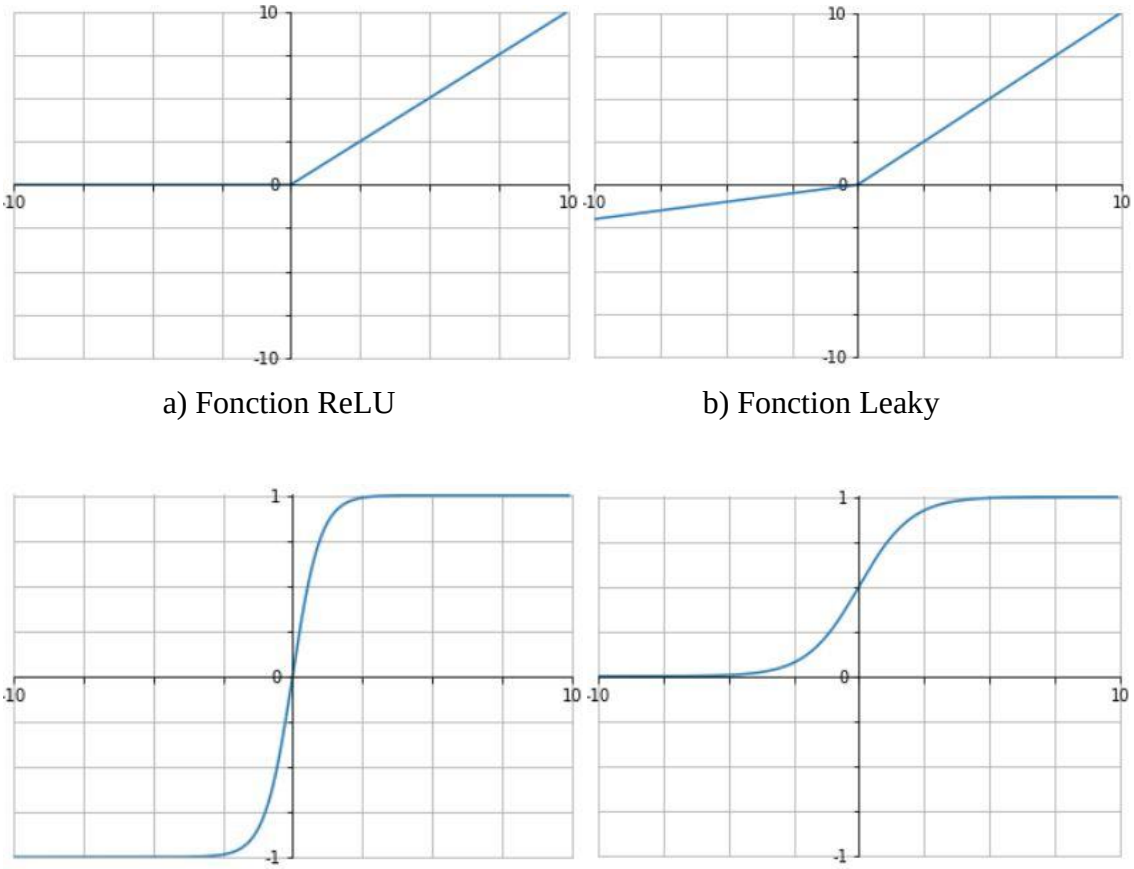
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Les valeurs d'entrée négatives plus grandes ont tendance à être plus proches de 0 tandis que les entrées positives plus grandes sont plus proches de 1. La fonction sigmoïde est rarement utilisée maintenant car elle souffre des problèmes de saturation de gradient et d'un calcul plus lent. En plus n'est pas centré en zéro.

1.4.4.4 La fonction tangente hyperbolique

La fonction tangente hyperbolique ou tanh est similaire à la fonction sigmoïde avec une différence de sortie, variant dans l'intervalle [−1, 1] au lieu de [0, 1]. Il est défini comme :

$$\tanh(x) = \frac{2}{1 + e^{-2x}}$$



a) Fonction ReLU b) Fonction Leaky

c) Fonction Tanh d) Fonction

Figure 1.6 : Différentes fonctions d'activation et leurs graphes.

1.5 L'entrainement d'un CNN :

L'entrainement d'un CNN pour une tâche de traitement d'image signifie le calcul d'une perte en comparant les prédictions de sortie avec les sorties ciblées. Pendant l'entraînement, le passage en arrière est calculé. Le passage en arrière peut simplement être décrit comme la sensibilité de la perte par rapport aux changements des paramètres du réseau. L'objectif principal de l'entrainement d'un CNN est de minimiser la fonction de perte dans l'ensemble de

données d'entrainement en modifiant les paramètres entraînables du réseau, tels que les poids et les biais, à l'aide de passage en arrière.

1.5.1 Fonction de perte (Loss Function)

Les fonctions de perte sont les fonctions différenciables utilisées pour guider le processus d'apprentissage d'un réseau neuronal. Pour s'entrainer un CNN pour une tâche de détection d'objets, les réseaux sont généralement optimisés pour les éléments de classification et de régression. Mathématiquement, la fonction de perte mappe la sortie du réseau y et la sortie ciblée $\hat{y}$ sur une valeur réelle représentant la pénalité pour l'inexactitude des prédictions. Les fonctions de perte les plus utilisées pour les problèmes de classification sont l'erreur quadratique moyenne totale (perte TMSE ou L2), entropie croisée (CEL) et la perte Softmax. Pour un problème de régression, les fonctions de perte peuvent varier d'un réseau à l'autre.

1.5.1.1 Erreur quadratique moyenne (TMSE)

L'erreur quadratique moyenne totale (TMSE) pour la sortie réseau y et la sortie cible $\hat{y}$ est définie comme suit

$$\text{TMSE}= -\frac{1}{n.l}\sum_{i=1}^{n}\|y_i - \hat{y_i}\|^2,$$

Où n et l représentent respectivement le nombre d'échantillons d'apprentissage et le nombre de neurones de sortie. Le terme $\|y_i - \hat{y_i}\|$ est la distance euclidienne entre y et $\hat{y}$.

1.5.1.2 La perte d'entropie croisée (CEL)

La perte d'entropie croisée pour la sortie réseau y et la sortie cible $\hat{y}$ est définie comme suit

$$\text{CEL}= -\frac{1}{N}\sum_{i=1}^{N}[\hat{y}\ logy_i + (1-\hat{y})\log(1-y_i)],$$

1.5.1.2 La perte Softmax

La perte Softmax est l'une des fonctions de perte les plus couramment utilisées dans les CNN. Il prend un vecteur N-dimensionnel de valeurs réelles et le transforme en vecteur de valeurs réelles dans un intervalle [0,1] de sorte que la somme soit égale à 1. Étant donné une matrice d'entrée $X_{n,k}$, la perte Softmax est définie comme suit :

$$\sigma(X_{n,k}) = \frac{e^{(X_{n,k})}}{\sum_{j=1}^{k} e^{(X_{n,j})}}$$

Et la perte est calculée comme :

$$\mathrm{L} = -\frac{1}{N}\sum_{n=1}^{N} \log\left(\frac{e^{(X_{n,k})}}{\sum_{k}^{j=1} e^{(X_{n,j})}}\right.$$

Où N est la taille du batch et K est le nombre d'identités (c'est-à-dire les étiquettes de classe).

1.6 Apprentissage par transfert

Avec les progrès dans le développement de CNN, il est rare d'entrainer un CNN à zéro, car il nécessite rarement de grandes quantités de données sur des clusters GPU pour être efficace. À partir de là, il est courant d'utiliser des réseaux pré-entrainés comme un extracteur de caractéristiques. La tâche de réutiliser un réseau pré-entrainé est connue sous le nom d'apprentissage par transfert et implique le transfert de connaissances d'un domaine à un autre. Le réseau de base est généralement entrainé sur un ensemble de données tel que MS COCO. Ce réseau sert alors de réseau de base utilisé dans l'apprentissage par transfert.

Lorsque le réseau pré-entraîné est utilisé comme extracteur de caractéristiques, la dernière couche entièrement connectée dans le CNN de base est supprimée et deux nouvelles couches d'adaptation sont ajoutées au réseau. Pendant l'entraînement, toutes les couches, à l'exception des deux nouvelles couches, restent fixes, et les seuls poids ajustés sont les poids de ces deux couches entièrement connectées.

1.7 Détection d'objets

La détection d'objets dans les images est l'un des problèmes les plus fondamentaux et les plus difficiles du domaine du traitement d'images. En plus d'identifier et de valider plusieurs objets dans une image et de les classer dans certaines classes, la détection d'objets traite également du problème de localisation, ainsi qu'une estimation approximative de leurs tailles. Par cette

définition, on peut conclure que la détection d'objets concerne deux sous-problèmes: la classification et la régression. Le problème de régression ou bien appelé régression de cadre de délimitation (bounding-box regression) concerne la régression linéaire de cadre de délimitation encapsulant des objets d'intérêt dans une image. Un cadre de délimitation idéal est un rectangle à axe parallèle qui contient toutes les parties d'un objet. Chaque boîte englobante d'une image est associée à un score de confiance qui estime la probabilité qu'un objet s'y trouve.

Typiquement, il existe trois étapes de base pour un cadre de détection d'objets basé région.

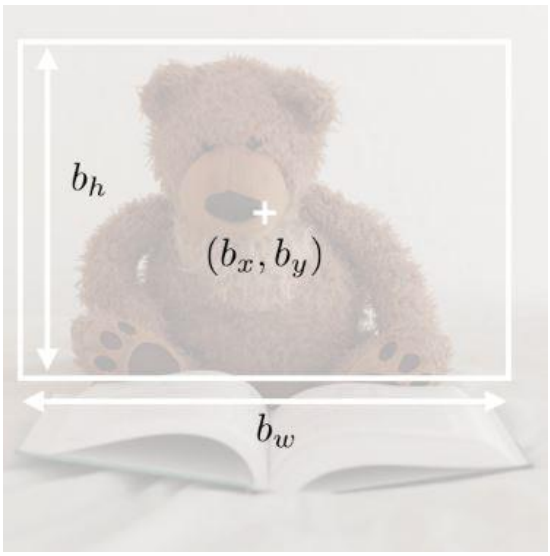
La première étape est la génération de régions d'intérêt (RoI) ou de propositions de régions. Un algorithme ou un modèle tel que RPN génère des boîtes de délimitation rectangulaires dans une image qui est le composant de localisation d'objet d'un détecteur d'objet. Après la génération des propositions de région, les caractéristiques visuelles sont extraites pour chacune des boîtes de délimitation à l'étape suivante. Ces caractéristiques visuelles décident si une boîte de délimitation contient un objet ou non. À l'étape finale, les boîtes qui se chevauchent sont combinées en une seule boîte de délimitation à l'aide d'un algorithme externe tel que NMS.

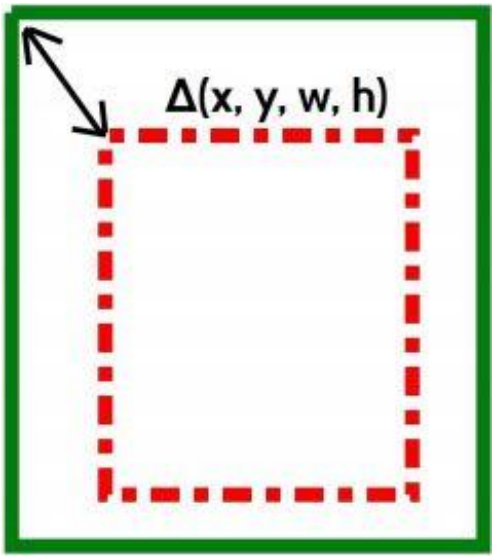
1.8 Les concepts importants de détection d'objets

Dans cette section, une brève introduction à certains des concepts importants utilisés dans la détection d'objets.

1.8.1 Région d'intérêt (RoI)

Une région d'intérêt, ou plus familièrement connue sous le nom de proposition de région ou proposition de boîte de délimitation, est une région rectangulaire dans une image d'entrée qui contient potentiellement un objet. Ces propositions peuvent être générées par certains algorithmes externes tels que la recherche sélective (6), la détection des boîtes de bord (Edge Box Detection) (10), ou par un réseau de propositions de régions (RPN). Un cadre de délimitation est représenté comme un vecteur 4×1 contenant son emplacement central, sa largeur et sa hauteur (x, y, w, h). Chaque boîte englobante dans une image est accompagnée d'un score d'objectivité ou de confiance de la probabilité que la boîte contienne un objet. La différence entre deux boîtes est généralement mesurée par la distance L2 de leurs représentations vectorielles (Figure 1.7).





a) Bounding-box encapsulant un objet d'intérêt

b) Décalage entre deux boite

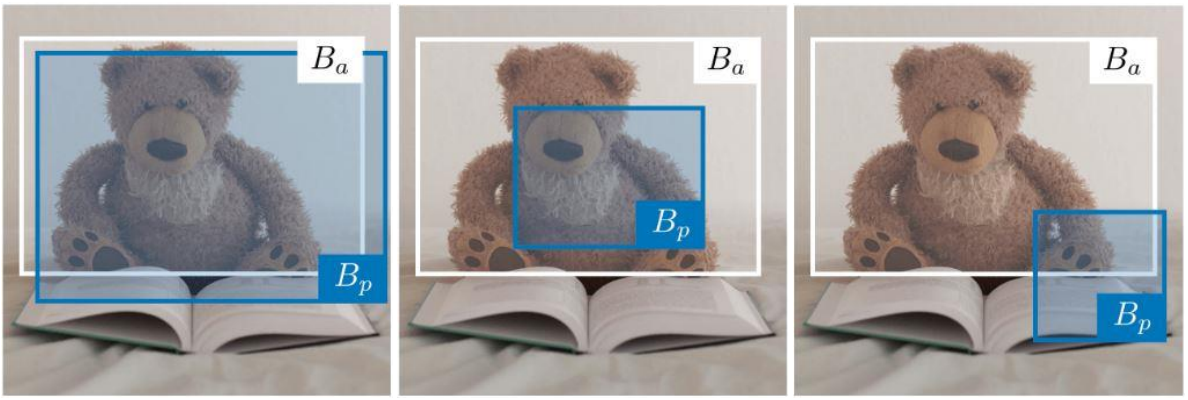
Figure 1.7: RoI dans une image.

1.8.2 Intersection sur Union (IoU)

Intersection sur Union (en anglais Intersection over Union), aussi appelé IoU, est une mesure basée sur l'indice Jaccard (22) qui quantifie à quel point la zone délimitante prédite B_p est correctement positionnée par rapport à la zone délimitante vraie B_a. Elle est définie de la manière suivante :

$$\mathbf{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

On a toujours IoU ∈ 0,1 par convention, la prédiction B_p d'une zone délimitante est considérée comme étant satisfaisante si l'on a $\mathbf{IoU}(B_p, B_a) \geq \mathbf{0.5}$.



$\mathbf{IoU}(B_p, B_a) \geq \mathbf{0.9}$ $\mathbf{IoU}(B_p, B_a) \geq \mathbf{0.5}$ $\mathbf{IoU}(B_p, B_a) \geq \mathbf{0.1}$

Figure 1.8 : IoU d’une image.

1.8.3 Suppression Non-Max (NMS) :

La technique de suppression non-max (en anglais non-max suppression) a pour but d'enlever des zones délimitantes qui se chevauchent et qui prédisent un seul et même objet, en sélectionnant les zones les plus représentatives. Après avoir enlevé toutes les zones ayant une probabilité prédite de moins de 0.6, il ya deux étapes répétées pour éliminer les zones redondantes. La première c’est de choisir la zone ayant la plus grande probabilité de prédiction, et la deuxième consiste à enlever toute zone ayant $\mathbf{IoU}(B_p, B_a) \geq \mathbf{0.5}$ avec la zone choisie précédemment.

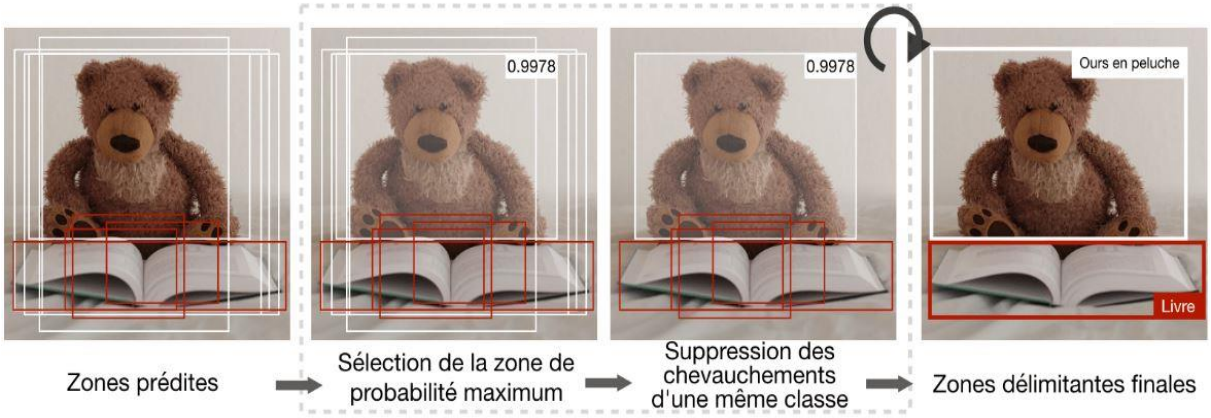


Figure 1.9 : Processus d’application NMS.

1.8.4 Bounding-box regression (raffinement de la boîte englobante)

La plupart des détecteurs d'objets modernes utilisent des régresseurs de boîte englobante qui sont entrainés pour prédire le décalage $\Delta(x, y, w, h)$ entre la boîte de région d'entrée et la boîte de vérité au sol. S'il existe un régresseur pour chaque classe d'objet, il est appelé régression spécifique à la classe (class-specific), et il est appelé indépendant de la classe (class-agnostic)

si un régresseur pour toutes les classes. Un régresseur de boîte englobante est souvent accompagné d'un classificateur de boîte englobante (score de confiance) pour estimer la confiance de l'existence d'un objet dans la boîte. Le classificateur peut également être spécifique ou indépendant de la classe.

1.9 Les architectures CNN

Les performances des réseaux de neurones profonds dans la détection d'objets reposent sur une grande quantité de données d'apprentissage ainsi que sur la puissance de calcul. La collecte et l'annotation d'un ensemble de données (Dataset) de taille suffisante est un processus mouvementé et qui prend du temps. Par conséquent, la plupart des recherches sur la détection d'objets sont effectuées sur des ensembles de données accessibles au public. Il existe plusieurs ensembles de données de ce type. Nous discutons brièvement de quelques architectures CNN les plus courantes et récentes.

1.9.1 LeNet-5

LeNet-5 est l'une des architectures les plus simples de LeCun et al. (21) en 1998. Il a 2 couches convolutionnelles (CONV) et 3 couches Fully Connected (FC) (d'où «5» - est le nombre de couches CONV et FC dont ils disposent). Cette architecture a environ 60 000 paramètres. Elle a été appliquée par plusieurs banques pour reconnaître les nombres manuscrits tels que les chèques. La capacité de traiter des images à haute résolution nécessite des couches plus grandes et plus convolutives, de sorte que cette technique est limitée par la disponibilité des ressources informatiques.

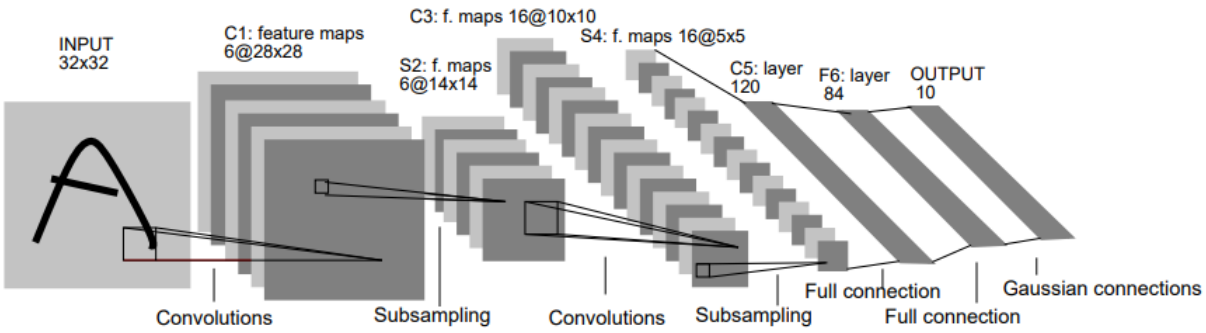


Figure 1.10 : Architecture LeNet-5 (21).

1.9.2 AlexNet

Alexanet (23) a été proposé par Krizhevsky et al. Cet article a été lauréat du défi de reconnaissance visuelle ImageNet 2012 (ILSVRC). Elle a notée le taux d'erreur le plus bas 15,4%. Le deuxième meilleur résultat avait le taux d'erreur de 26,2%. Ce fut une énorme amélioration des performances.

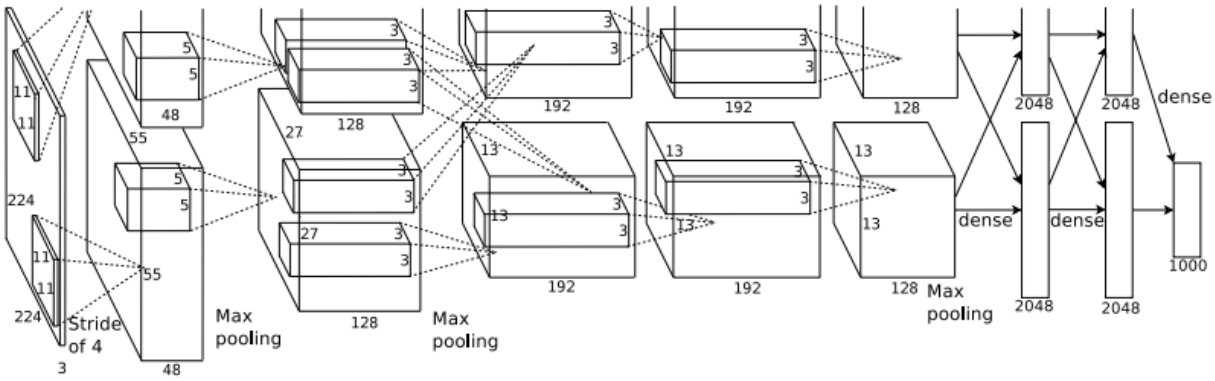


Figure 1.11 : Architecture AlexNet (23).

Avec 60M de paramètres, AlexNet a 8 couches - 5 convolutionnelles et 3 entièrement connectées. Le processus d'entrainement était coûteux en calcul par rapport à l'heure actuelle. Nous pouvons voir deux flux de réseaux convolutionnels. Une autre caractéristique d'AlexNet était que l'unité linéaire rectifiée était utilisée comme fonction d'activation non linéaire. Le modèle a été entrainé sur un dataset de 15 millions d'images en utilisant la méthode SGD sur deux GPU Nvidia Geforce GTX 580 pendant cinq à six jours ce qui explique pourquoi leur architecture est divisé en deux pipelines.

1.9.3 ZF Net

Le ZF Net (24) a été proposé par Zeiler et Fergus en 2013. Il n'est pas surprenant qu'il a remporté l'ILSVRC en 2013 avec le top 5 des taux d'erreur de 11,2%. Ce modèle n'a été entrainé que sur 1,3 million d'images, contre 15 millions d'images pour AlexNet. Ils ont également changé la taille du filtre de 11×11 dans AlexNet en 7×7 dans ZF Net. Ce modèle a été entrainé sur le GPU Nvidia Geforce GTX 580 mais pendant 12 jours.

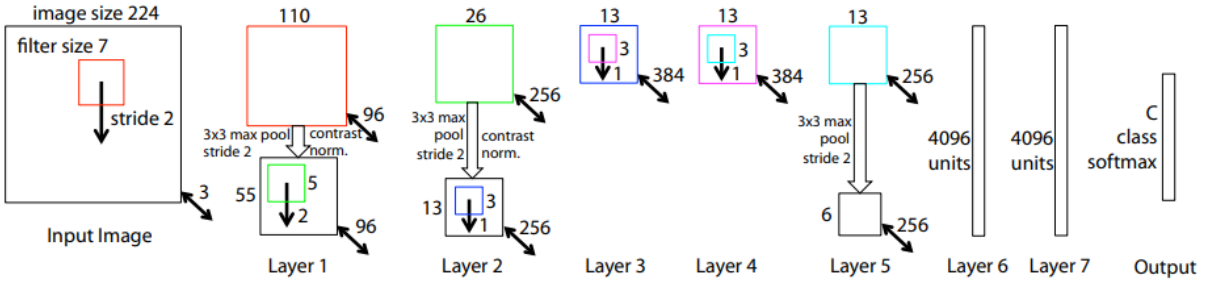


Figure 1.12 : Architecture ZF Net (24).

1.9.4 VGGNet

VGGNet (25) a été proposé en 2014 par Simonyan et al. L'équipe de Visual Geometry Group (VGG) ont inventés le VGG-16 qui a 13 couches convolutionnelles (CONV) et 3 couches Fully Connected (FC).Ils ont remplacés le filtre 11×11 AlexNet et le filtre 7×7 ZF Net par un filtre 3×3 dans leur modèle. Il se compose de 138 millions de paramètres et occupe environ 500 Mo d'espace de stockage. Ils ont également conçu un autre modèle plus profond, VGG-19. Cela représentait 7,3% du taux d'erreur dans le top 5 lors la compétition ILSVRC.

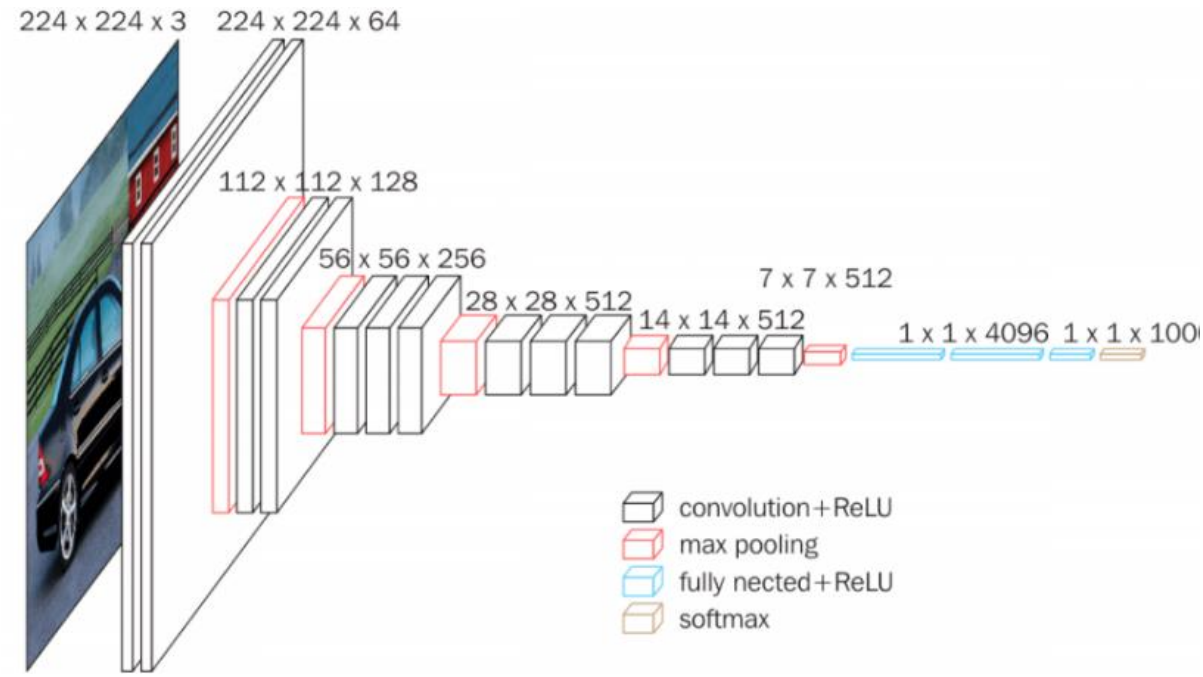


Figure 1.13 : Architecture VGGNet (25).

1.9.5 GoogLeNet/Inception

GoogLeNet / Inception-v1 a remporté l'ILSVRC 2014 avec un taux d'erreur de 6,7% dans le top 5 (26) Ceci a été développé par des chercheurs de Google en 2014. Cependant, GoogLeNet contient une pile de 22 convolutions et un nouveau module appelé Inception Module, cela est appelé Inception-v1. Ce modèle est composé de calculs parallèles de convolutions de différentes formes et de max-pooling. La sortie de tous ces modules est concaténée à la fin de chaque module Inception. Ils ont réduit le nombre de paramètres de 60 millions (AlexNet) à 4 millions.

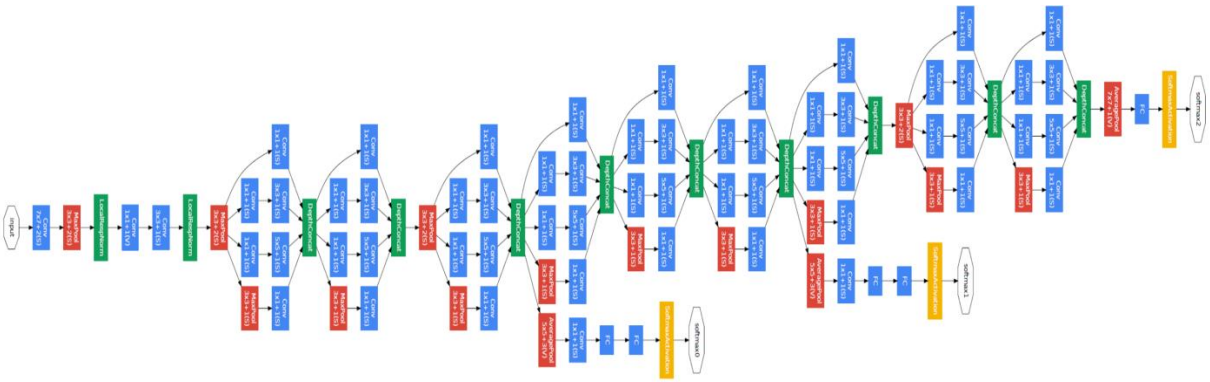


Figure 1.14 : Architecture GoogLeNet/Inception-v1 (26).

Autre modèle proposé en 2015 « Inception-V2 » est un successeur d'Inception-V1, c'était le prototype d'Inception-V3, avec 24M paramètres.

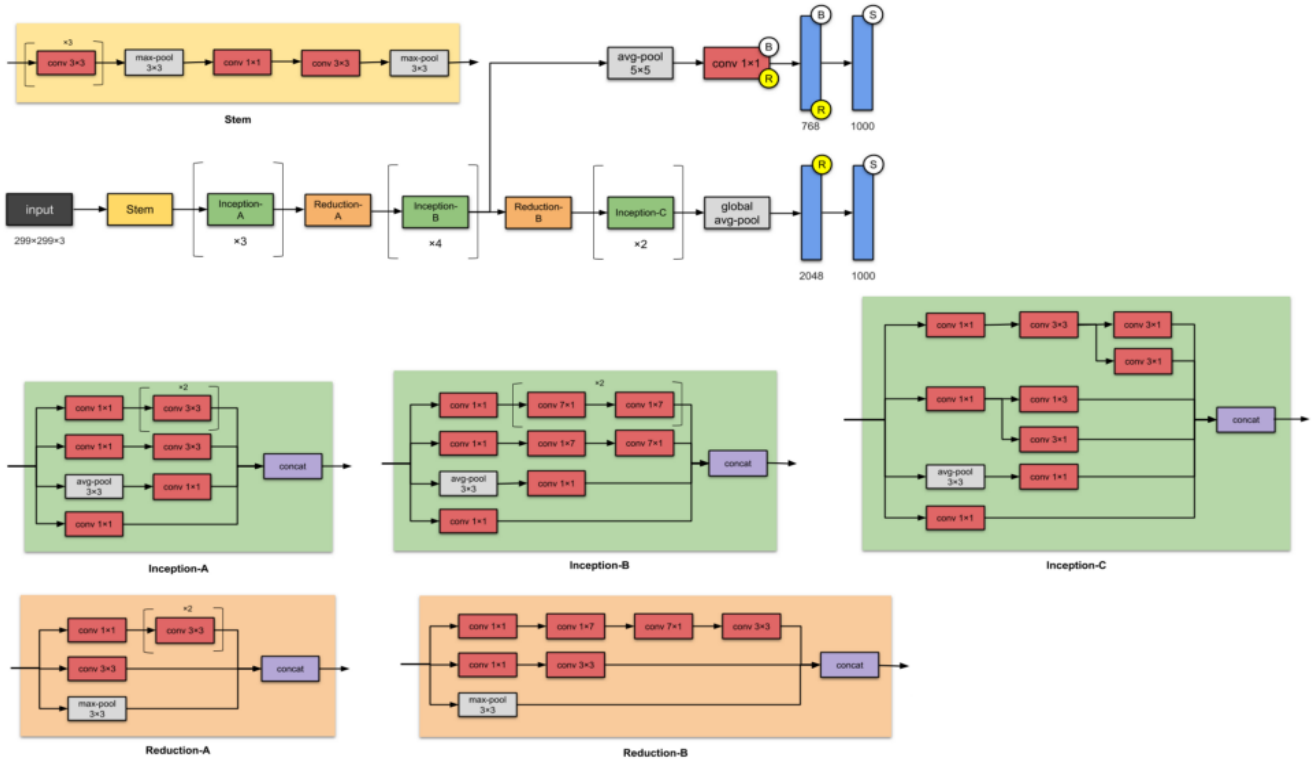


Figure 1.15 : Architecture GoogLeNet/Inception-v2 (27).

1.9.6 ResNet-50

ResNet (Residual Neural Network) est une architecture de réseau profond avec 152 couches (28). Ceci a été développé par Microsoft en 2015. Il a remporté l'ILSVRC 2015 avec un taux d'erreur de 3,6%, ce qui est considéré comme meilleur que la précision au niveau humain. Les couches résiduelles dans ResNet calculent les changements dans l'entrée. Ceci est ensuite ajouté à l'entrée pour produire la sortie. ResNet-50 est l'un des premiers à adopter la normalisation des batchs. Le ResNet est constitué de deux blocs CONV et Identité.

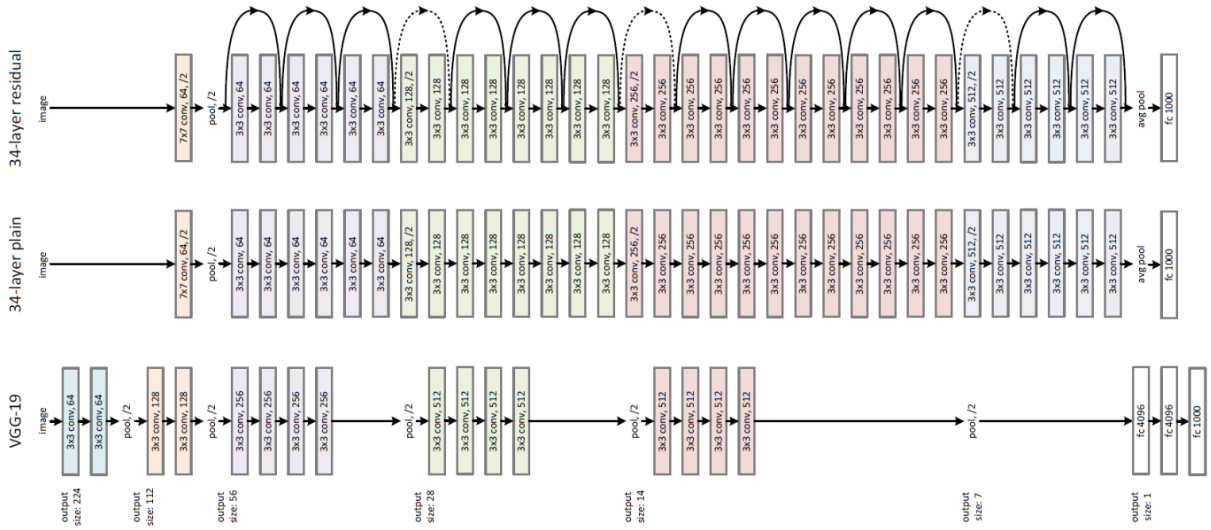


Figure 1.16 : Architecture ResNet-50 (28).

1.9.7 Xception

Xception est une adaptation d'Inception, où les modules Inception ont été remplacés par des convolutions séparables en profondeur. Il a également à peu près le même nombre de paramètres qu'Inception-v1 (23 M). Xception prend l'hypothèse Inception à un eXtrême (d'où le nom). Premièrement, les corrélations cross-canal (ou cross-feature map) sont capturées par convolutions 1 × 1. Par conséquent, les corrélations spatiales au sein de chaque canal sont capturées via les convolutions régulières 3 × 3 ou 5 × 5.

Pousser cette idée à l'extrême signifie effectuer 1 × 1 sur chaque canal, puis effectuer un 3 × 3 sur chaque sortie. Cela revient à remplacer le module Inception par des convolutions séparables en profondeur.

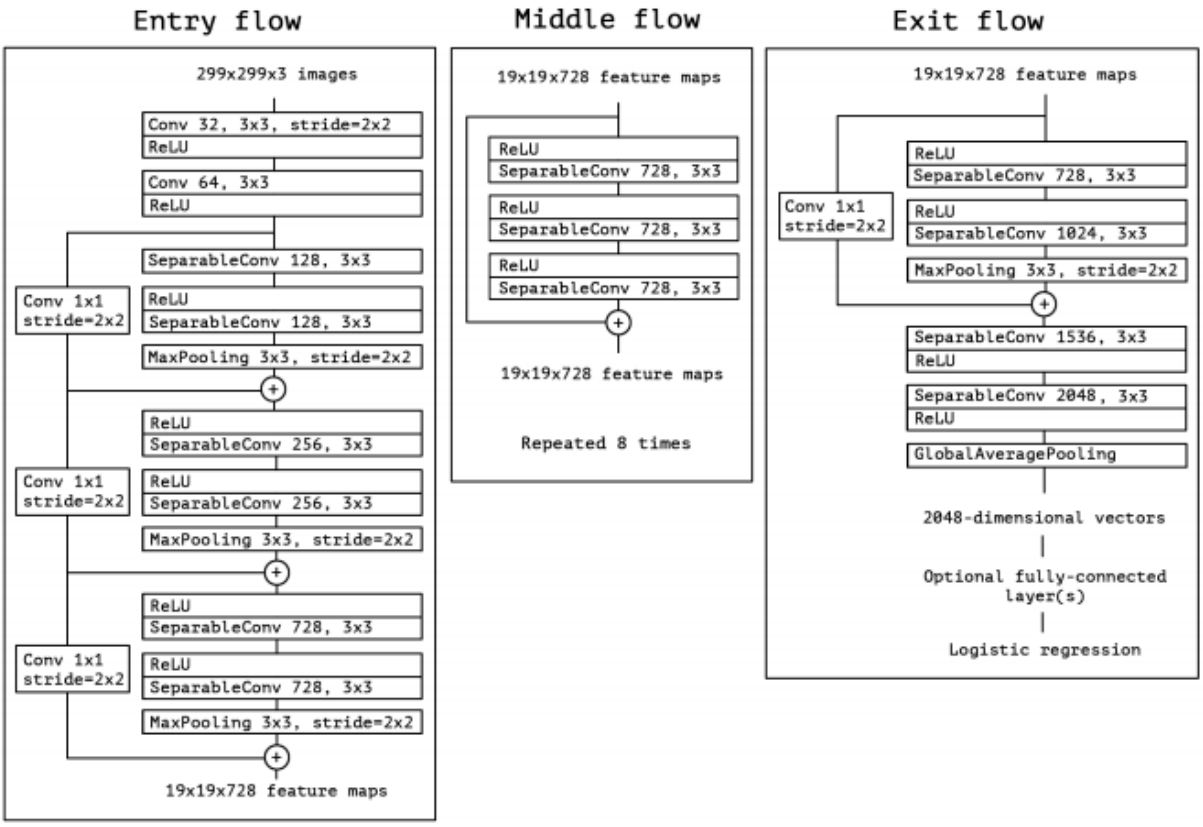


Figure 1.17 : Architecture Xception **(**29**)** .

1.9.8 ResNeXt-50

ResNext-50 a été proposé en 2017, similaire a ResNet-50 a 25M paramètres (ResNet-50 a 25,5M). La différence avec ResNeXts est l'ajout de tours / branches / chemins parallèles dans chaque module, comme indiqué ci-dessus par «32 tours au total».

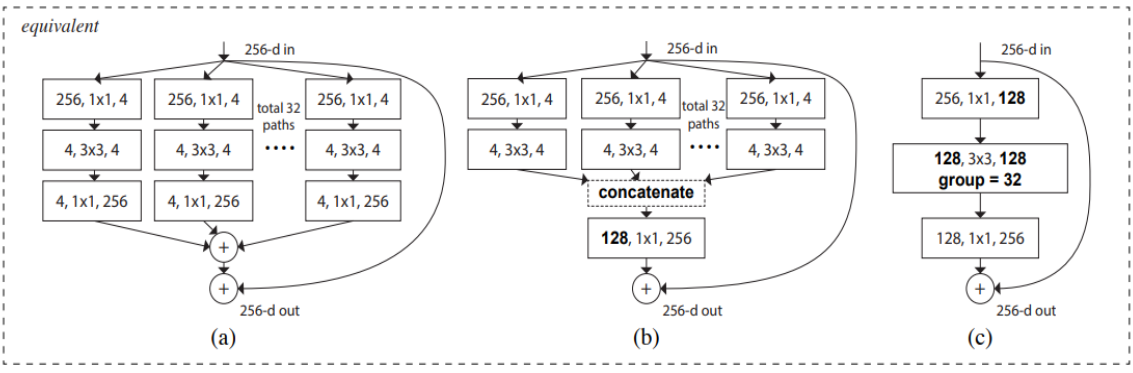


Figure 3. Equivalent building blocks of ResNeXt. (**a**): Aggregated residual transformations, the same as Fig. 1 right. (**b**): A block equivalent to (a), implemented as early concatenation. (**c**): A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

Figure 1.18 : Architecture ResNeXt-50 (30)

1.9.9 MobileNet-V2

Une architecture qui cherche à bien fonctionner sur les appareils mobiles. Il est basé sur une structure résiduelle inversée où les connexions résiduelles sont entre les couches de bottleneck. La couche d'expansion intermédiaire utilise des convolutions légères en profondeur pour filtrer les entités en tant que source de non-linéarité. Dans son ensemble, l'architecture de MobileNetV2 contient la couche initiale entièrement à convolution avec 32 filtres, suivie de 19 couches de bottleneck résiduelles.

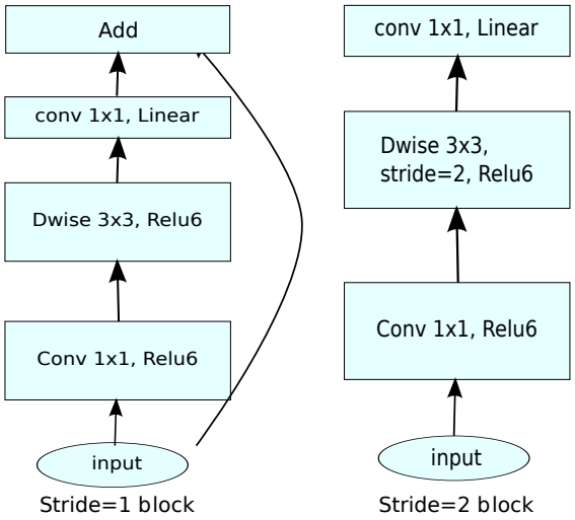


Figure 1.19 : Architecture MobileNet-V2 (31)

1.9.10 EfficientNet

EfficientNet (32) est une architecture et une méthode de mise à l'échelle qui met à l'échelle uniformément toutes les dimensions de profondeur/ largeur/résolution à l'aide d'un coefficient composé. Contrairement à la pratique conventionnelle qui met ces facteurs à l'échelle de manière arbitraire, la méthode de mise à l'échelle EfficientNet met à l'échelle uniformément la largeur, la profondeur et la résolution du réseau avec un ensemble de coefficients de mise à l'échelle fixes.

Le réseau de base EfficientNet-B0 est basé sur les blocs résiduels de goulot d'étranglement inversé de MobileNetV2, en plus des blocs de squeeze-and-excitation.

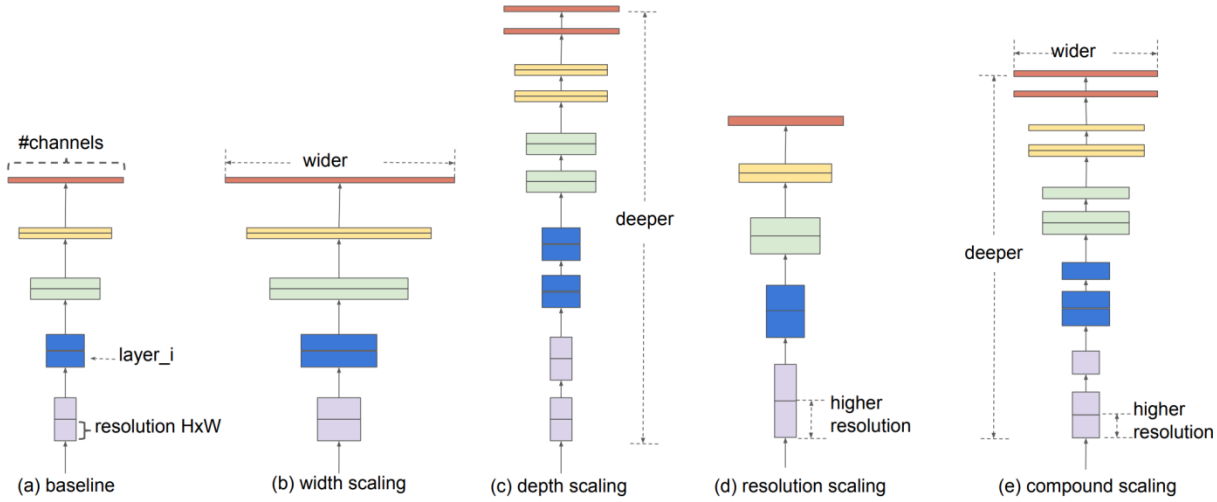


Figure1.20 : Architecture EfficientNet (32)

1.9.11 CSPDarknet-53

Cette architecture crée en 2020 est considérée comme un réseau neuronal convolutif et une backbone pour la détection d'objets qui utilise DarkNet-53 (33). Il utilise une stratégie CSPNet pour partitionner les feature maps de la couche de base en deux parties, puis les fusionne à travers une hiérarchie en plusieurs étapes. L'utilisation d'une stratégie de fractionnement et de fusion permet un flux plus dégradé à travers le réseau. Ce CNN est utilisé comme backbone pour YOLOv4.

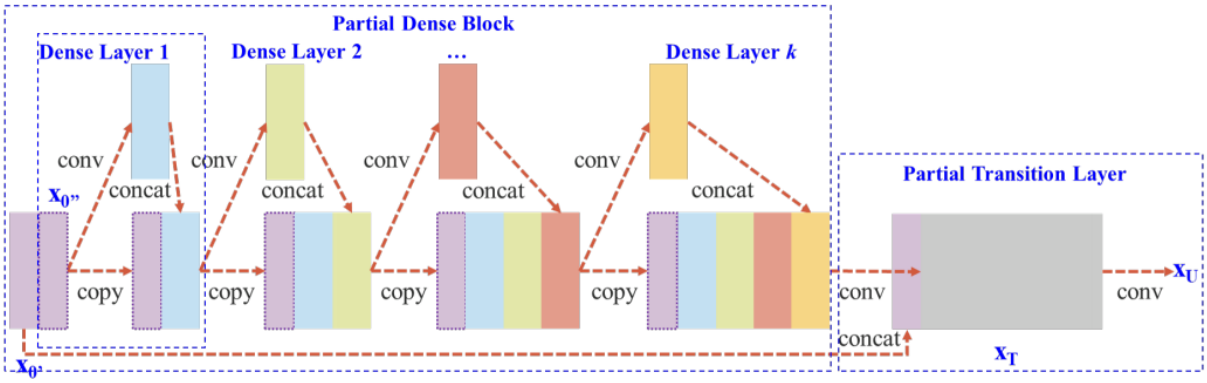


Figure 1.21 : Architecture CSPDarknet (33).

Conclusion

Dans cette partie, nous avons présenté les contextes théorique de l'apprentissage profond et explique les concepts fondamentaux de la détection d'objets, et à la fin du chapitre nous avons présenté les architectures CNN les plus courantes et récentes.

Dans le second chapitre, nous rentrerons plus en détail sur la détection d'objet utilisant un CNN.

Chapitre 2 : Travaux connexes

Ce chapitre fournit un bref historique des méthodes classiques de détection d'objets qui utilisent les réseaux neuronaux convolutionnels (CNN) ainsi que leur histoire moderne. En outre, les méthodes qui combinent les CNNs avec les générateurs de propositions de région seront brièvement discutées. La section 2.1 et la section 2.2 discutent des principes fondamentaux des frameworks de détection d'objets basés région, qui a inspiré le développement des approches présentées dans ce travail.

2.1 Méthodes classiques

L'une des tâches fondamentales de la vision par ordinateur est de permettre aux ordinateurs d'avoir une interprétation humaine pour des informations visuelles complexes. Une telle interprétation peut se produire à plusieurs niveaux dans une image, comme la détection et la segmentation d'objets. La détection d'objets est la tâche d'identifier la présence de toute instance d'une classe d'objets donnée dans une image tandis que la segmentation est la tâche d'étiqueter chaque pixel d'image avec la classe d'objets.

La détection et la segmentation d'objets sont les domaines de vision par ordinateur qui progressent le plus rapidement. Désormais, un ordinateur peut facilement dépasser une performance humaine moyenne en détection et segmentation d'objets, tout cela grâce aux technologies informatiques avancées et aux grands ensembles de données accessibles au public. Même si la popularité des applications de vision par ordinateur susmentionnées a considérablement augmenté au cours de la dernière décennie, son histoire remonte au début des années 1960.

Avant l’utilisation des réseaux de neurones profonds pour les tâches de détection et de segmentation d'objets, les frameworks de détection d'objets classiques étaient principalement basés sur l'extraction de descripteurs de caractéristiques, tels que les histogrammes de gradients orientés (HOG) (1) ou la transformation de caractéristiques visuelles invariante à l'échelle (SIFT) (3) avec classification ultérieure par un classificateur linéaire, tel que SVM. Dans cette section, ces méthodes classiques seront brièvement décrites.

2.1.1 Histogramme de gradient orienté (HOG)

HOG ou en anglais « Histogram of Oriented Gradients » est un descripteur de caractéristique dense pour les images, développé par Navneet Dalal et al. (1). L'idée importante derrière le descripteur HOG est que l'apparence et la forme locale d'un objet dans une image peuvent être décrites par la distribution de l'intensité du gradient ou la direction des contours.

En pratique, HOG divise d'abord une image en certain nombre de régions spatiales (ou blocs) qui se chevauchent. Ces blocs sont également appelés histogrammes de descripteurs de gradients orientés (HOG). Chaque bloc est ensuite divisé en $n \times n$ cellules plus petites, où chaque cellule contient un nombre fixe de cases d'orientation de gradient sur n nombre de pixels. Chaque pixel de la cellule vote alors pour une classe de l'histogramme, en fonction de l'orientation du gradient à ce point. Une fois les gradients d'intensité calculés pour chaque pixel de la cellule. Les orientations de gradient sont quantifiées en plusieurs cases.

Afin d'extraire des caractéristiques et de classer des régions à partir de sous-images à plusieurs échelles et proportions, HOG utilise une approche à fenêtre coulissante. La mise en mosaïque de ces fenêtres coulissantes avec une grille dense de blocs qui se chevauchent ou de descripteurs HOG, et l'utilisation du vecteur de caractéristique combiné dans un classificateur linéaire tel que les machines à vecteurs de support (SVM) aboutissent à la sortie finale (classification objet/non-objet).

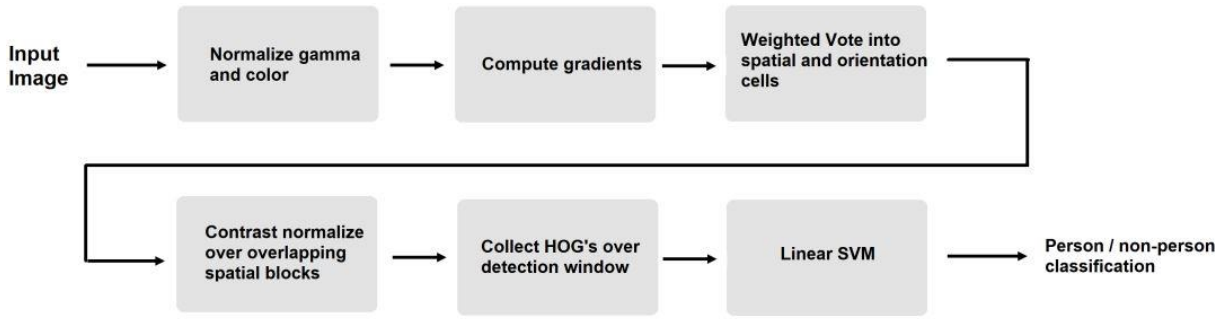


Figure 2.1 : Structure de descripteur HOG.

Étant donné que le grand HOG produisait des résultats de pointe au début des années 2000, il manquait de robustesse en ce qui concerne l'occlusion et les déformations.

2.1.2 Transformation de caractéristiques visuelles invariante à l'échelle (SIFT)

SIFT ou bien en anglais «Scale Invariant Feature Transform» développée à l'origine par David Lowe en 1999 (3), est un descripteur d'image, qui a été largement utilisé pour la localisation et la reconnaissance de la personne/visage. Le descripteur SIFT s'est avéré très utile en pratique pour la reconnaissance d'objets dans des conditions réelles en raison de son invariance aux translations, rotations et transformations d'échelle dans le domaine de l'imagerie. Le fonctionnement d'un descripteur SIFT peut être divisé en plusieurs étapes. La première étape consiste à extraire des points clés ou des points d'intérêt à partir d'images de niveau de gris étiquetées. Ces points clés sont situés dans un espace 2D où la variation du signal dépasse un certain seuil. SIFT extrait ces points clés en créant un espace d'échelle. Cet espace d'échelle est obtenu en construisant un ensemble d'images avec flou gaussien. Une fois les points clés obtenus, un vecteur caractéristique est calculé en trouvant les histogrammes des directions de gradient dans un voisinage local autour de chaque point clé, cette création d'un vecteur caractéristique comme un descripteur est la deuxième étape et un aspect unique de SIFT. Dans la troisième étape, les mauvais points clés tels que les contours à faible courbure et les régions à faible contraste sont éliminés et un tableau d'histogrammes d'orientation est calculé pour les points clés restants. Tout autre calcul est effectué par rapport à cet histogramme d'orientation qui annule efficacement l'effet de l'orientation, ce qui le rend invariant en rotation. Enfin, avec l'échelle et l'invariance de rotation en place, une transformée de Hough est effectuée pour identifier les groupes de correspondances, que l'on appelle clusters à partir d'un objet spécifique. Ensuite, la probabilité qu'un vecteur caractéristique particulier représente un objet dans l'image est calculée. La vérification est effectuée sous la forme d'une solution des moindres carrés, appliquée aux paramètres obtenus à partir de la transformation affine (3).

2.1.3 OverFeat (approche de fenêtre coulissante)

Les premières tentatives de détection d'objets à l'aide de CNN reposaient fortement sur le traitement des fenêtres coulissantes. De telles méthodes se sont avérées très efficaces dans plusieurs domaines, notamment la détection des piétons, visages et des textes. Un de ces réseaux appelé OverFeat (Sliding Window Approach) (34), a remporté la Compétition ImageNet de Reconnaissance Visuelle à Grande Échelle ImageNet 2013 (ILSVRC 2013)

(35). Les détecteurs d'objets basés région modernes doivent leur succès à OverFeat qui est considéré comme le modèle pionnier de l'intégration des tâches de classification et de régression d'images dans un réseau neuronal convolutif (CNN). L'idée principale derrière cela est d'effectuer une classification à différents endroits (ou régions) sur plusieurs échelles d'une image en mode fenêtre coulissante, c'est-à-dire en faisant glisser une fenêtre $n \times n$ sur la carte des caractéristiques (Feature Map) obtenue par la dernière couche convolutionnelle du CNN, et pour effectuer une régression de boîte englobante (expliquée à la sous-section 1.7.4) pour envelopper un objet plus étroitement.

Cette méthode entraine d'abord un modèle CNN pour la tâche de classification d'image. Ensuite, elle remplace les couches du classificateur supérieur par un réseau de régression spécifique et l'entraîner pour affiner les limites des boîtes englobantes à chaque emplacement spatial. Enfin, les scores de classe résultants et les boîtes englobantes régressées sont agrégés à l'aide d'un algorithme de glouton.

Malgré le succès de ces méthodes, de nombreux inconvénients subsistent. L'un des inconvénients majeurs était la redondance des fenêtres traitées (ne contenant que partiellement un objet), ce qui augmentait considérablement le temps de fonctionnement. Cependant, l'idée de localisation d'objet (combinaison de classification d'image et de régression) dans Overfeat a été étendue dans la tâche de détection d'objet moderne (où l'idée est de localiser les objets sur plusieurs régions d'images).

2.2 Méthodes basées CNN

Nous discutons maintenant des méthodes de localisation et de détection d'objets qui s'appuient sur les CNN. Comme indiqué précédemment, ceux-ci représentent un développement récent du problème de reconnaissance d'objets.

2.2.1 Réseau de neurones convolutif basé région (R-CNN)

En 2014, le premier détecteur d'objets basé région a été développé par Girshick et. al, est appelé le R-CNN (4).

Le modèle R-CNN se compose de trois modules. Le premier module génère des propositions de région en scannant l'image d'entrée à la recherche d'objets possibles à l'aide d'un algorithme externe. Il existe différentes méthodes pour la génération de propositions de

région, comme la recherche sélective (6), propositions d'objets indépendantes de la catégorie (Category Independent Object Proposals) (7), Constrained Parametric Min-Cuts (CPMC) (8), le regroupement combinatoire à plusieurs échelles (9) etc. La méthode de génération de proposition de région la plus couramment utilisée est l'algorithme de recherche sélective, qui crée près de 2000 RoI ou propositions de région indépendantes de la catégorie. Ces propositions de région sont ensuite déformées pour correspondre à l'entrée d'un grand réseau neuronal convolutionnel multicouche (CNN), qui est le deuxième module de R-CNN. Ce grand CNN extrait un vecteur d'entité de longueur fixe de chaque région d'une image. Le troisième module est un ensemble de machines à vecteurs de support linéaires (SVM) spécifiques à une classe. Dans ce module, les vecteurs de caractéristiques obtenus à partir du CNN sont fournis audit classifieur et l'ensemble des propositions de régions classées est réduit en utilisant la suppression non maximale (NMS). Le vecteur de caractéristiques est également introduit dans un régresseur linéaire, laissant les boîtes englobantes attendues des objets dans l'image d'entrée. Trois modules d'un R-CNN sont illustrés à la figure 2.2

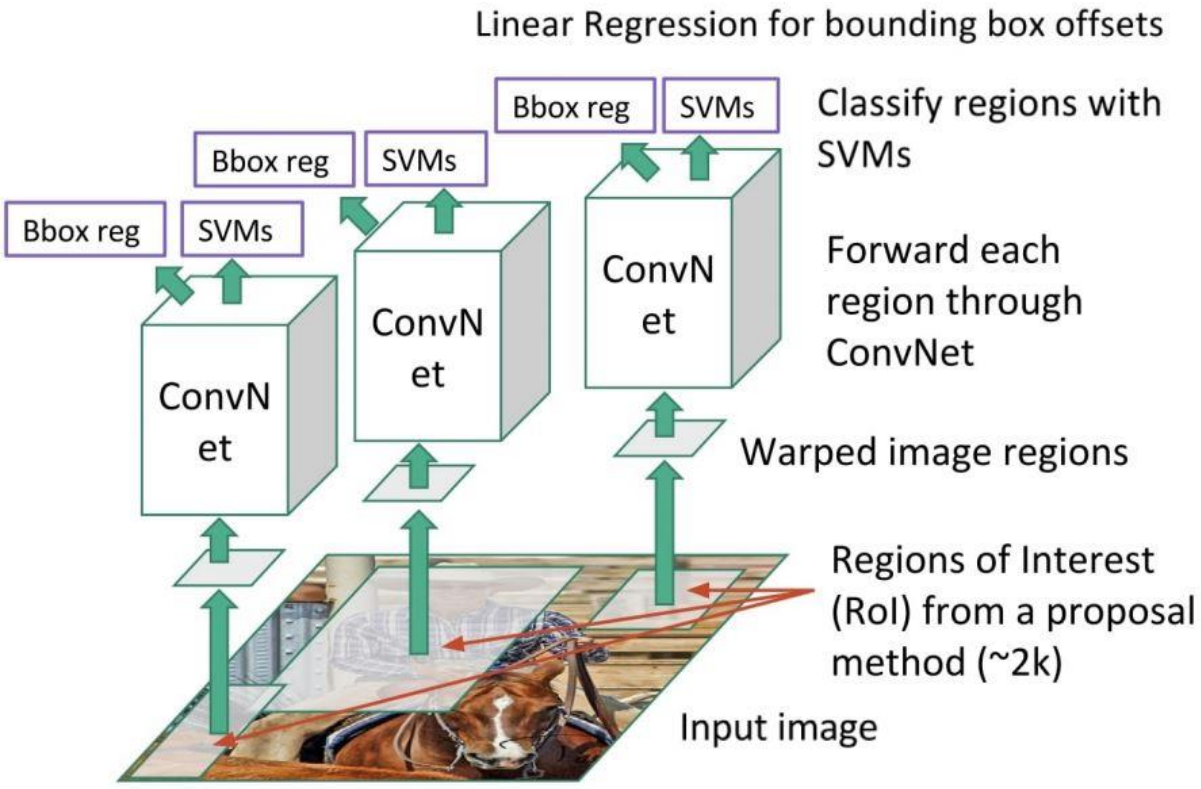


Figure 2.2 : Architecture R-CNN (4).

Bien que le R-CNN soit intuitif, il avait peu d'inconvénients. L'un des principaux inconvénients était la taille d'entrée fixe des propositions de région, dépassée par la déformation des régions d'image (indépendamment de sa taille, de son emplacement ou de son aspect) aux dimensions prédéfinies avant le traitement par le CNN. Un autre inconvénient majeur du R-CNN était son coût de calcul élevé; résultat du passage individuel de sous-images déformées à travers le CNN (avec environ 2000 régions générées au moment du test).

2.2.2 Réseau de neurones convolutif basé région rapide (Fast R-CNN)

Fast-R-CNN (5) est le premier successeur de R-CNN, conservant la plupart des notions fondamentales de R-CNN et introduisant peu de raffinements. Dans Fast R-CNN, le CNN principal avec plusieurs couches convolutives est appliqué à l'image pour la détection des caractéristiques avant de proposer des régions. Cela signifie qu'il ne nécessite pas plusieurs CNN sur plusieurs régions qui se chevauchent. Le SVM est également remplacé par une couche Softmax, étendant ainsi le réseau neuronal pour les prédictions au lieu de créer un nouveau modèle (Figure 2.3). Un autre raffinement a été la couche de mise en commun des régions d'intérêt (RoI).

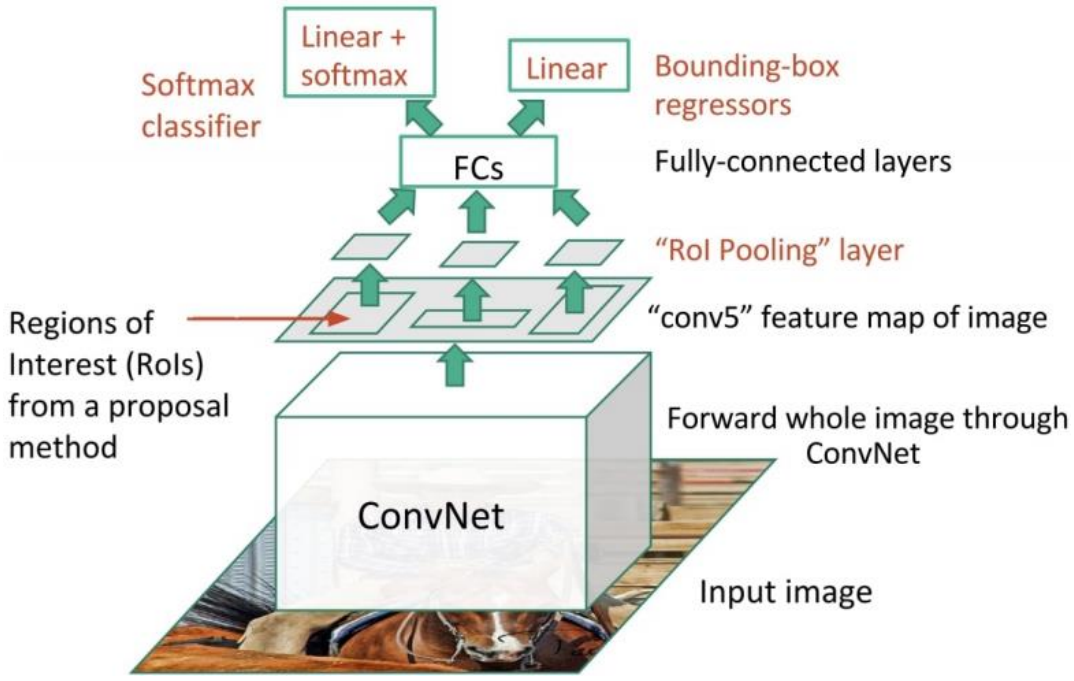


Figure 2.3 : Architecture Fast R-CNN (5).

Le Fast R-CNN rapide est presque 10 fois plus efficace et précis que le R-CNN en termes de calcul, car les propositions de région sont générées à partir de la carte des caractéristiques (Feature map) au lieu de l'image d'origine. Cependant, les propositions de région étaient toujours détectées avec la méthode de recherche sélective lente.

2.2.3 Réseau de neurones convolutif basé région plus rapide (Faster R-CNN)

Faster R-CNN développé par Shaoqing Ren et al. en 2016 (11), est le successeur de Fast R-CNN (5). Il est le plus couramment utilisé pour la détection d'objets tridimensionnels, segmentation, etc. Théoriquement, le Faster RCNN se compose de 3 réseaux neuronaux tels que le réseau de fonctionnalités (Feature Network), le réseau de proposition régionale (Regional Proposal Network) et le réseau de détection (Detection Network) (11). La figure 2.4 représente l'architecture de l'algorithme Faster RCNN.

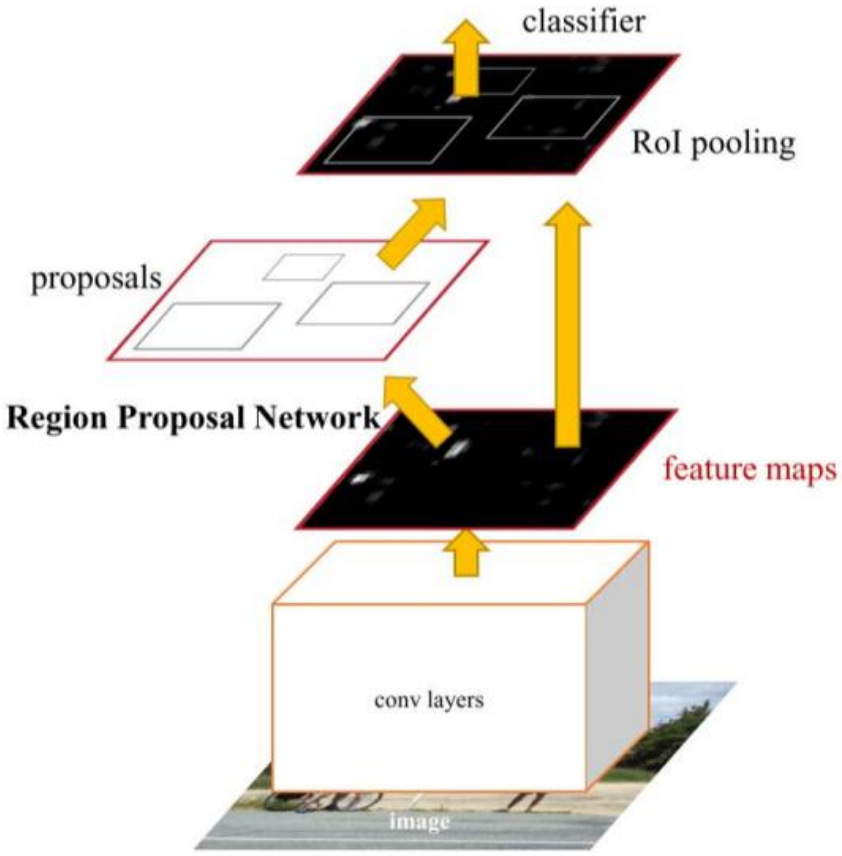


Figure 2.4 : Architecture Faster R-CNN (11).

2.2.3.1 Feature Network

Un feature network est un réseau de classification d'images. La fonction principale de ce réseau est d'identifier et de générer de bonnes caractéristiques à partir des images d'un ensemble de données (11). La sortie de cette couche contient des caractéristiques obtenues à partir des images qui sont ensuite utilisées comme entrées pour d'autres couches.

2.2.3.2 Regional Proposal Network (RPN)

Un réseau de proposition régional ou RPN est un réseau neuronal composé de 3 couches convolutives et l'une d'elles est une couche commune qui alimente en entrée les deux autres couches, à savoir la couche de classification et la couche de régression de boîte englobante (bounding box) (11). Le Réseau de proposition régional est responsable de générer des cadres de délimitation appelés Région d'intérêts. Ces régions d'intérêts (RoI) ont la plus grande possibilité de contenir l'objet que l'algorithme doit détecter. La sortie de ce réseau contient une série de valeurs telles que 1, 0 et -1 indiquant si un objet se trouve dans les boîtes englobantes générées, si l'objet n'est pas dans les boîtes englobantes ou si les boîtes englobantes générées peuvent être ignorées respectivement. Les cartes d'entités obtenues (feature maps) à partir du réseau d'entités peuvent être de tailles différentes et il est difficile de travailler sur des entités de tailles différentes. Pour résoudre ce problème, la mise en commun des régions d'intérêts est utilisée pour réduire les cartes d'entités générées à la même taille.

2.2.3.3 Detection Network

Le but de ce réseau est de générer un cadre de délimitation (bounding box) final en tenant compte des entrées du réseau d'entités (feature network) et du réseau de proposition régional (Regional Proposal Network) (11). Il se compose de 4 couches entièrement connectées qui sont à leur tour interconnectées à la couche de régression de boîte englobante (bounding box) et à la couche de classification qui aident à générer des détections finales.

2.2.5 Mask R-CNN

Le Mask R-CNN peut être considéré comme une extension du Faster R CNN (12). Il ajoute une petite surcharge au Faster R-CNN en ajoutant une branche supplémentaire pour prédire

un masque d'objet en parallèle avec les branches existantes pour la classification et la régression de la boîte englobante. Voir figure 2.5

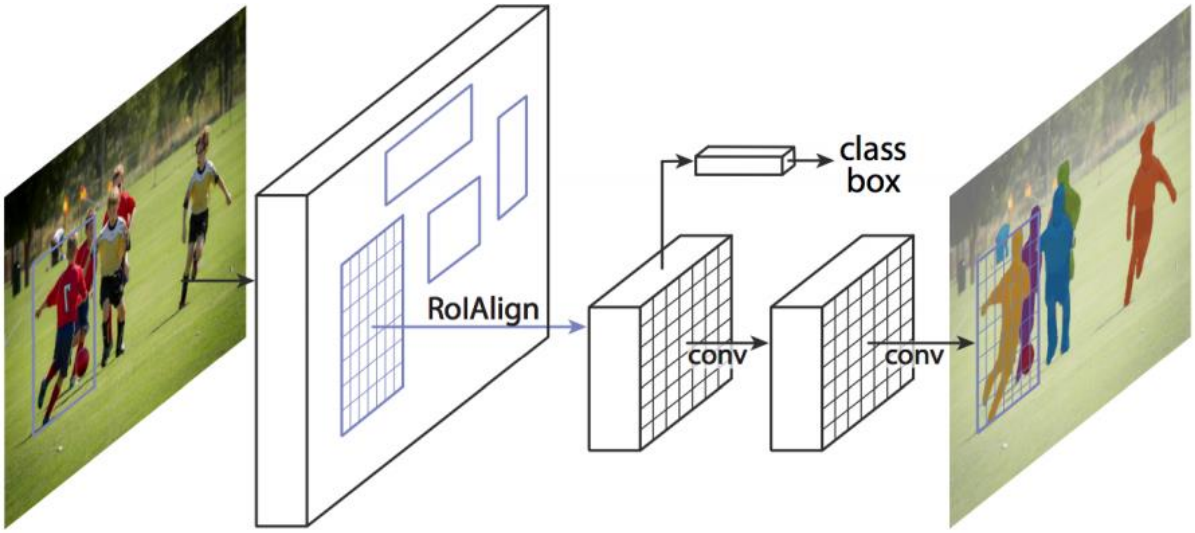


Figure 2.5 : Architecture Mask R-CNN (12).

Le Mask R-CNN utilise la couche RoiAlign au lieu de la couche RoIPool dans Faster R-CNN, cela augmente considérablement la précision de localisation.

2.2.6 SSD

Le SSD en anglais Single Shot MultiBox Detector présenté par Liu et al (36), qui est capable de fournir des performances en temps réel avec une grande précision. Ce réseau n'utilise pas la méthode de proposition régionale. Dans ce réseau, la localisation et la classification des objets sont effectuées en un seul passage aller du réseau tout en utilisant une technique connue sous le nom de «MultiBox» pour effectuer la régression de la boîte englobante (bounding box). Le SSD est donc capable d'effectuer des calculs end-to-end.

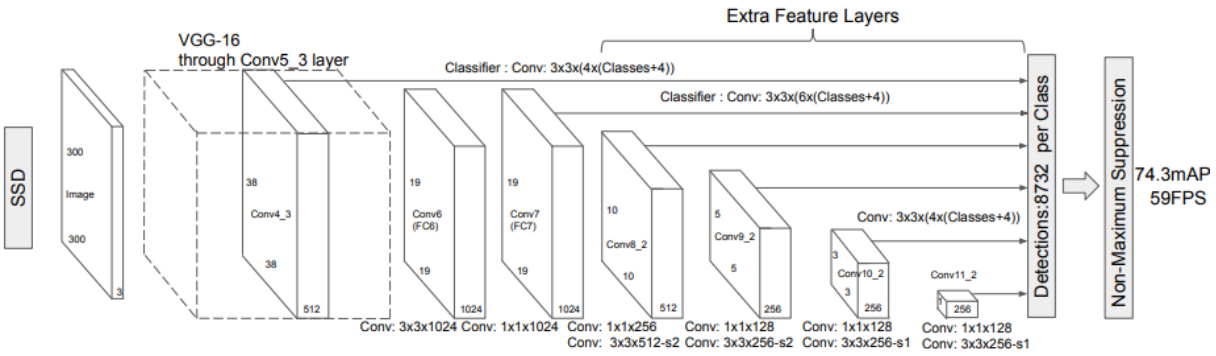


Figure 2.6 : Architecture SSD (36).

2.2.7 RetinaNet

RetinaNet (37) est un détecteur d'objet à un seul stage avec un backbone et deux sous-réseaux, un pour la classification et un pour la régression. Le backbone génère les feature map convolutives sur une image d'entrée entière similaire à Faster R-CNN. RetinaNet utilise un Feature Pyramid Network (FPN) construit sur ResNet comme backbone. Le subnet de classification est chargé de prédire la probabilité de présence d'objet à chaque position spatiale pour chacune des ancres A et K classes d'objets. Il prend une carte d'entités d'entrée avec des canaux C à partir d'un niveau de pyramide et applique quatre couches de convolution 3x3, chacune avec des filtres C et chacune suivie par des activations ReLU. Enfin, les activations sigmoïdes sont attachées aux sorties.

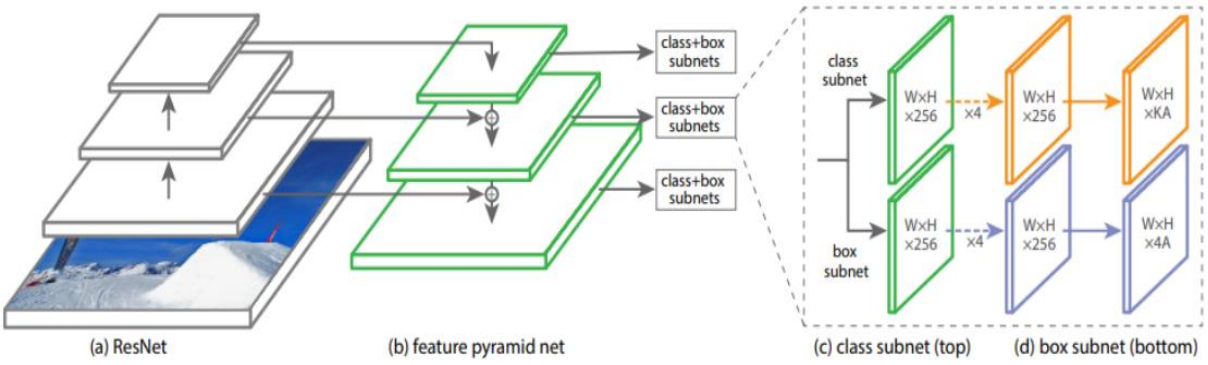


Figure 2.7 : Architecture RetinaNet (37).

2.2.7 YOLO V4

Les trois premières versions de YOLO ont été publiées en 2016, 2017 et 2018 respectivement. Cependant, en 2020 une version principale de YOLO a été publiée, nommées YOLO V4. Il prend l'influence de l'état de l'art BoF (Bag of Freebies) et plusieurs BoS (Bag of Specials). Le BoF améliore la précision du détecteur, sans augmenter le temps d'inférence. Ils ne font qu'augmenter le coût de la formation. D'un autre côté, les BoS augmentent légèrement le coût de l'inférence, mais ils améliorent considérablement la précision de la détection des objets. La figure 2.8 montre l'architecture générale de cette méthode de détection.

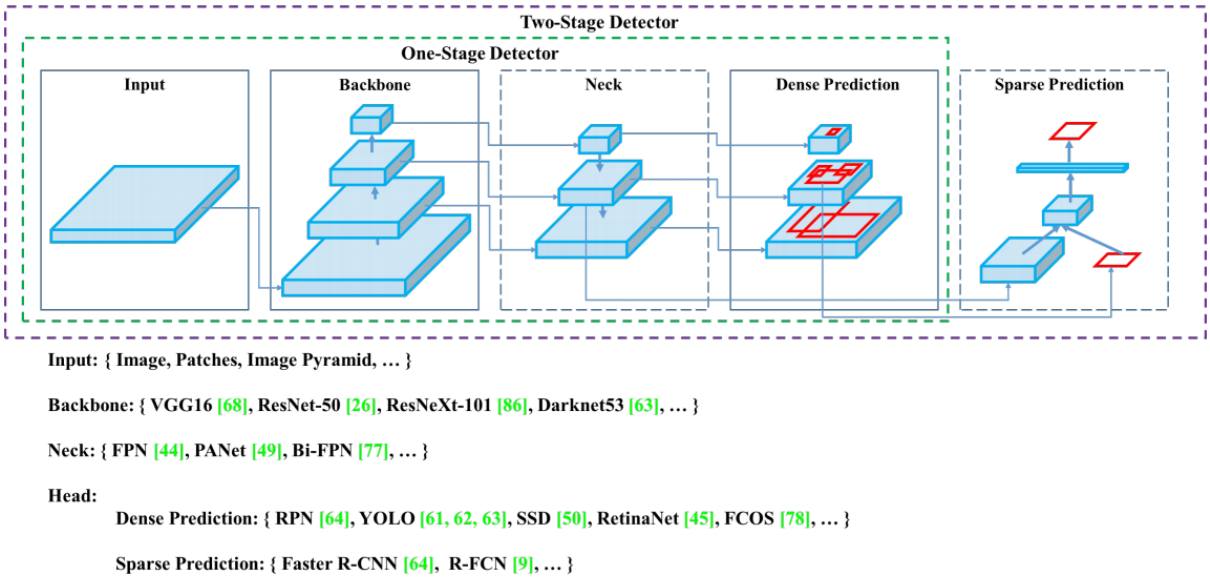


Figure 2.8 : Architecture globale YOLO v4 (38).

Conclusion

Dans ce chapitre, nous avons présenté les méthodes classiques de détection d'objets ensuite on a focalisé notre attention sur les méthodes de détection basé sur CNN et leur fonctionnement, et ses différentes architecture. On a présenté les méthodes récentes et les plus performantes, parmi eux Faster R-CNN qui sera implémenté dans notre système.

Le chapitre suivant explique l'idée d'utilisé cette méthode pour un la détection d'objet dans une voiture autonome.

Chapitre 3 : Implémentation et résultat expérimentaux

L'objectif de ce chapitre est de présenter les étapes de l'implémentation de l'approche proposée dans le cadre d'une méthode de détection d'objets basée Faster R-CNN pour une voiture autonome. On mettra en avant notre conception, puis nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement que nous avons utilisé. Puis les étapes de la réalisation du modèle et on termine par les tests effectués. Ce chapitre est composé de deux parties, l'implémentation de la méthode et les résultats expérimentaux des tests.

3.1 Conception

Avant d'entrer dans l'implémentation, on va tout d'abord détailler la méthode Faster R-CNN qui fait généralement référence à un pipeline de détection qui utilise le RPN comme algorithme de proposition de région et Fast R-CNN comme « Detection Network ». Puis on va expliquer la méthode de travail pour ce mémoire.

3.1.1 Architecture RPN

C'est un petit réseau de neurones glissant sur la dernière carte de caractéristiques des couches de convolution et prédire s'il y a un objet ou non et également prédire la boîte englobante de ces objets.

- Le réseau de proposition de région (RPN) commence par l'introduction de l'image d'entrée dans le CNN. L'image d'entrée est d'abord redimensionnée de telle sorte que son côté le plus court soit de 200 pixels et le côté le plus long ne dépasse pas 1000 pixels.
- Les fonctionnalités de sortie du backbone (indiquées par H x L) sont généralement beaucoup plus petites que l'image d'entrée en fonction de pas (stride) du backbone.
- Pour chaque point de la carte des caractéristiques en sortie, le réseau doit apprendre si un objet est présent dans l'image d'entrée à son emplacement correspondant et estimer sa taille, cela se fait en plaçant un ensemble d '«ancres» sur l'image d'entrée pour chaque emplacement sur la carte des caractéristiques « Feature map » de sortie du backbone. Ces ancres indiquent des objets possibles de différentes tailles et

proportions à cet emplacement. La figure ci-dessous montre 9 ancres possibles dans 3 rapports d'aspect différents et 3 tailles différentes placées sur l'image d'entrée pour un point A sur la carte des caractéristiques en sortie. Les ancres utilisées ont 3 échelles de zone de boîte 128^2, 256^2, 512^2 et 3 rapports d'aspect de 1: 1, 1: 2 et 2: 1.

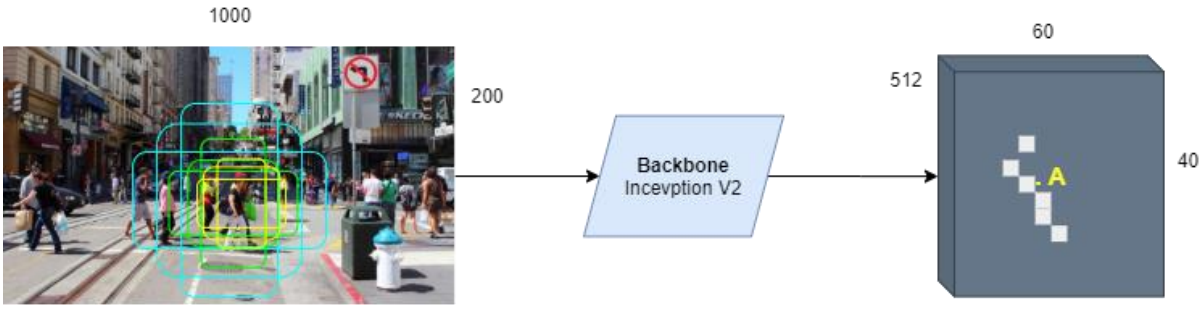


Figure 3.2: Les ancres possibles dans la carte des caractéristiques.

- Au fur et à mesure que le réseau se déplace à travers chaque pixel de la carte des entités (Feature map) en sortie, il doit vérifier si ces k ancres correspondantes couvrant l'image d'entrée contiennent réellement des objets, et affiner les coordonnées de ces ancres pour donner des cadres de délimitation en tant que «Object proposals» ou régions d'intérêt.
- Tout d'abord, une convolution 3 x 3 avec 512 unités est appliquée à la carte des caractéristiques de backbone, comme le montre la figure 1, pour donner une carte des caractéristiques de 512-d pour chaque emplacement. Ceci est suivi de deux couches frères: une couche de convolution 1 x 1 avec 18 unités pour la classification des objets et une convolution 1 x 1 avec 36 unités pour la régression de la boîte englobante (bounding-box).
- Les 18 unités de la branche de classification donnent une sortie de taille (H, W, 18). Cette sortie est utilisée pour donner des probabilités de savoir si chaque point de la carte des caractéristiques (Feature map) de backbone (taille: H x L) contient un objet dans les 9 ancres à ce point.
- Les 36 unités de la branche de régression donnent une sortie de taille (H, W, 36). Cette sortie est utilisée pour donner les 4 coefficients de régression de chacun des 9 ancres pour chaque point de la carte des caractéristiques de backbone (taille: H x L). Ces

coefficients de régression sont utilisés pour améliorer les coordonnées des ancres qui contiennent des objets.

3.1.2 La méthode Faster R-CNN

Après la conception de RPN on peut dire que l'architecture Faster R-CNN se compose du RPN comme algorithme de proposition de région et du Fast R-CNN comme réseau de détection. Voici les gros points pour cette méthode :

- L'image d'entrée est d'abord passée à travers le backbone pour obtenir la carte des entités (taille des entités: 60, 40, 512). Outre l'efficacité du temps de test, une autre raison clé d'utiliser un RPN comme générateur de proposition est logique: les avantages du partage de poids entre le backbone RPN et le backbone du détecteur Fast R-CNN.
- Ensuite, les propositions de boîte englobante du RPN sont utilisées pour regrouper les entités à partir de backbone de la carte des caractéristiques. Ceci est fait par la couche de regroupement ROI. La couche de mise en commun du retour fonctionne essentiellement en : a) prenant la région correspondant à une proposition à partir de backbone de la carte des caractéristiques; b) diviser cette région en un nombre fixe de sous-fenêtres; c) Effectuer un pooling maximal sur ces sous-fenêtres pour donner une sortie de taille fixe.
- La sortie de la couche de regroupement ROI a une taille de (N, 7, 7, 512) où N est le nombre de propositions de l'algorithme de proposition de région. Après les avoir passés à travers deux couches entièrement connectées, les entités sont introduites dans les branches jumelles de classification et de régression.
- Notez que ces branches de classification et de détection sont différentes de celles du RPN. Ici, la couche de classification a des unités C pour chacune des classes de la tâche de détection (y compris une classe d'arrière-plan fourre-tout). Les caractéristiques sont passées à travers une couche SOFTMAX pour obtenir les scores de classification - la probabilité qu'une proposition appartienne à chaque classe. Les coefficients de la couche de régression sont utilisés pour améliorer les boîtes englobantes prédites. Ici, le régresseur est indépendant de la taille (contrairement au

RPN) mais est spécifique à chaque classe. Autrement dit, toutes les classes ont des régresseurs individuels avec 4 paramètres correspondant chacun à des unités de sortie C * 4 dans la couche de régression.

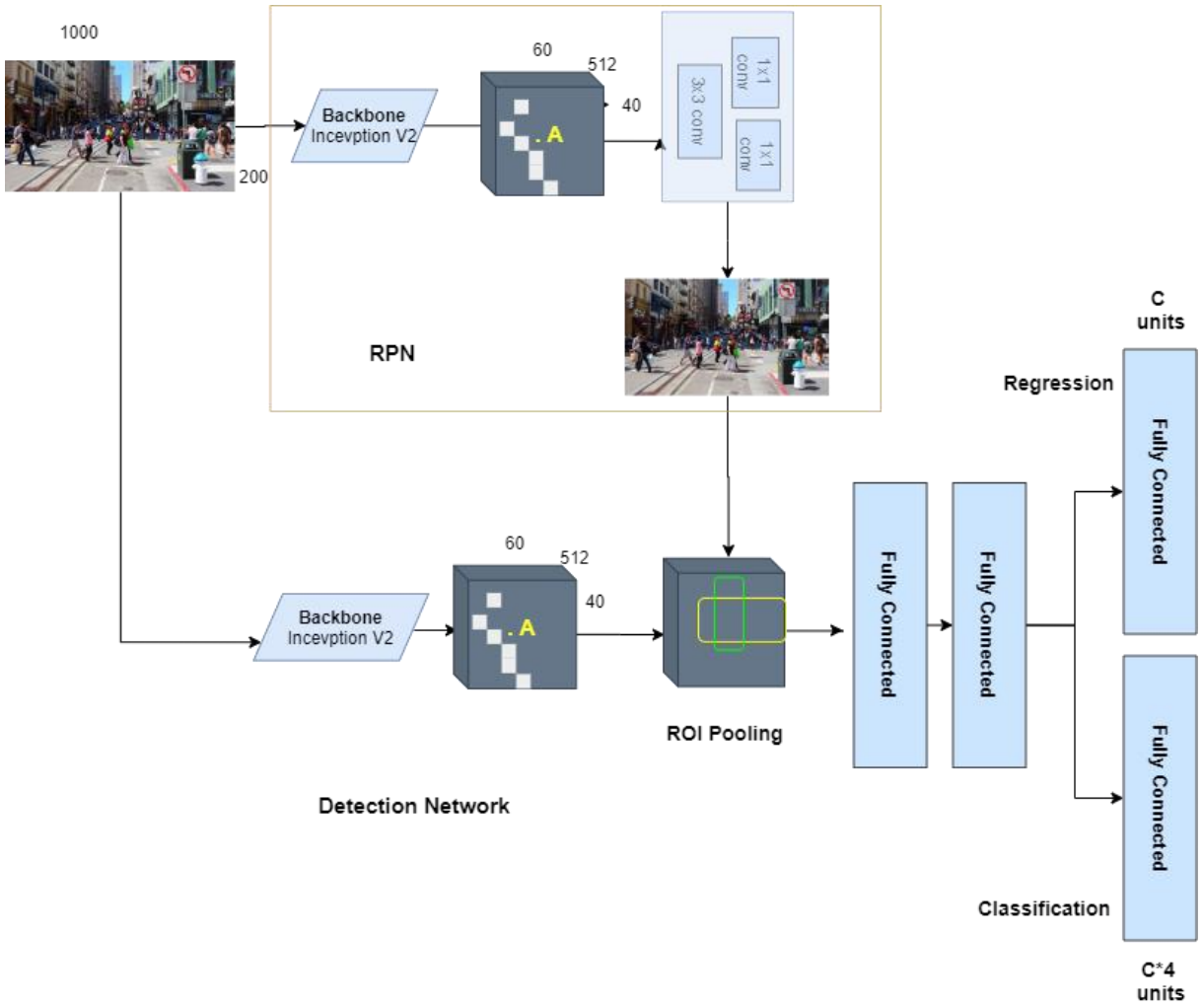


Figure 3.3: Pipeline du Faster R-CNN

3.1.3 Méthode de travail

Après connaitre le pipeline de Faster R-CNN, on a essayé d'utiliser un modèle pré-entrainé a partir de Model ZOO qui serait équilibré, c'est-à-dire on n'a pas concentré sur le plus rapide ni le plus précis mais on a essayé de choisir l'un qui convient ces deux facteurs. Donc on a

choisit Inception v2 pré entrainé sur COCO. Au même temps on a collecté l'ensemble de donnée d'entrainement à partir d'Open Images Dataset V4, de façon que ce modèle soit hybride c'est-à-dire entrainé sur deux grandes benchmark. Par la suite on va commencer de collecter le dataset puis le prétraiter par le redimensionner en 200x200 et le convertir en TFRecord, puis le nettoyer par l' élimination de quelque images car peuvent affecter négativement sur les résultats de détection, donc le dataset est nettoyé pour une détection plus précise.

Avant de commencer le processus d'entrainement, on remplace la dernière couche entièrement connectée dans le backbone du modèle par deux nouvelles couches d'adaptation. Durant la phase d'entraiment on mesure les performances du modèle pour les discuter plus tard. A la fin d'entrainement, on fait un test d'inférence sur une image ou une vidéo pour avoir les résultats obtenus.

3.2 Environnement software

Python

Python est un langage de programmation de haut niveau simple à apprendre et polyvalent. La syntaxe simple mais élégante de Python en fait l'un des langages de programmation idéaux pour le développement rapide d'applications (39). Certains des systèmes renommés développés en Python sont YouTube, Google Search, Bit Torrent, etc.

Python est largement considéré comme un langage préféré pour le développement d'algorithmes d'apprentissage automatique et d'apprentissage profond. Par conséquent, Python est utilisé comme langage principal pour implémenter les algorithmes d'apprentissage profond et également pour les raisons suivantes:

1. Python est plus simple à apprendre et à coder par rapport à d'autres langages de programmation comme C, C ++ et Java.
2. Python a des bibliothèques intégrées pour une implémentation facile des algorithmes.
3. C'est un langage de programmation open source.

Dans ce travail, les bibliothèques Python suivantes ont été utilisées pour implémenter les algorithmes d'apprentissage profond:

Tensorflow : Il s'agit d'une plate-forme open source end-to-end fournissant des bibliothèques, des outils et des ressources flexibles qui aident les développeurs à créer et déployer facilement des applications d'apprentissage automatique (40) .

Cython : C'est un compilateur statique qui facilite l'écriture d'extensions C pour Python. Il permet aux développeurs d'écrire du code Python qui peut communiquer avec d'autres programmes écrits en C et C ++ (41).

Pillow : Il s'agit d'une bibliothèque Python qui fournit de puissantes capacités de traitement d'image à l'interpréteur Python et prend également en charge divers formats de fichiers (42).

Lxml : C'est une simple bibliothèque Python utilisée pour traiter HTML et XML en Python (43).

Jupyter : C'est une application Web open source permettant aux développeurs ou aux chercheurs de partager des documents contenant du code, des visualisations de données, du texte et des équations. Cette bibliothèque Python est généralement utilisée pour la visualisation des données, le nettoyage des données, la transformation des données, l'apprentissage automatique, etc. (44).

Matplotlib : Il s'agit d'une bibliothèque de traçage Python capable de générer des graphiques, des histogrammes, des graphiques à barres, des nuages de points, des cartes thermiques, etc. (45).

Pandas : Il s'agit d'une bibliothèque open source pour Python fournissant des outils d'analyse de données faciles à utiliser pour Python (46).

OpenCV : Il s'agit d'une bibliothèque open source de vision par ordinateur et d'apprentissage automatique pour Python. Il est livré avec de nombreux algorithmes optimisés qui peuvent être utilisés pour détecter, identifier des objets, suivre des objets en mouvement, créer des cadres de délimitation (bounding box), etc. (47)

3.3 Environnement hardware

L'environnement matériel du système sur lequel le modèle a été formé est présenté ci-dessous dans le tableau 3.1.

Modèle de la machine	ASUS VivoBook S15 S510UN
CPU	Intel Core i7
GPU	Nvidia MX150
RAM	8GB
Système d'exploitation	Windows 10

Tableau 3.1: Environnement matériel

3.4 Dataset

L'ensemble de données utilisé dans ce travail contient presque 2000 images collectés à partir de Google Open Images V4 et qui comprenait cinq classes: personne, voiture, vélo, arbre et panneau de signalisation. Les classes ont été collectées de telle manière que cette détection soit pour une voiture autonome.

3.5 Prétraitement des données

Pour le Faster R-CNN, Il est suggéré que l'image d'entrée soit redimensionnée de manière à ce que le côté le plus court de l'image soit d'environ 200 pixels tandis que l'autre côté ne dépasse pas 1000 pixels. Toutes les images collectées sont modifiées pour avoir une résolution maximale de 200 x 200 car plus les images sont grandes, plus il faudra de temps pour entraîner le modèle. La taille de l'image d'entrée dépend principalement du réseau neuronal convolutif du squelette dans lequel l'image est introduite. Un script Python est écrit pour redimensionner les images de l'ensemble de données.

3.6 Étiquetage de dataset

Avant de commencer le processus d'entrainement, il est nécessaire d'étiqueter les images dans l'ensemble de données. Dans ce travail, l'outil LabelImg est utilisé pour étiqueter l'ensemble de données manuellement. Les étapes suivies sont:

1. Chargez le répertoire de l'ensemble de données dans l'outil LabelImg.

2. Ouvrez chaque image pour dessiner des cadres de délimitation (bounding box) autour des objets désirés puis étiqueter par exemple «Car» comme illustré à la figure 3.4

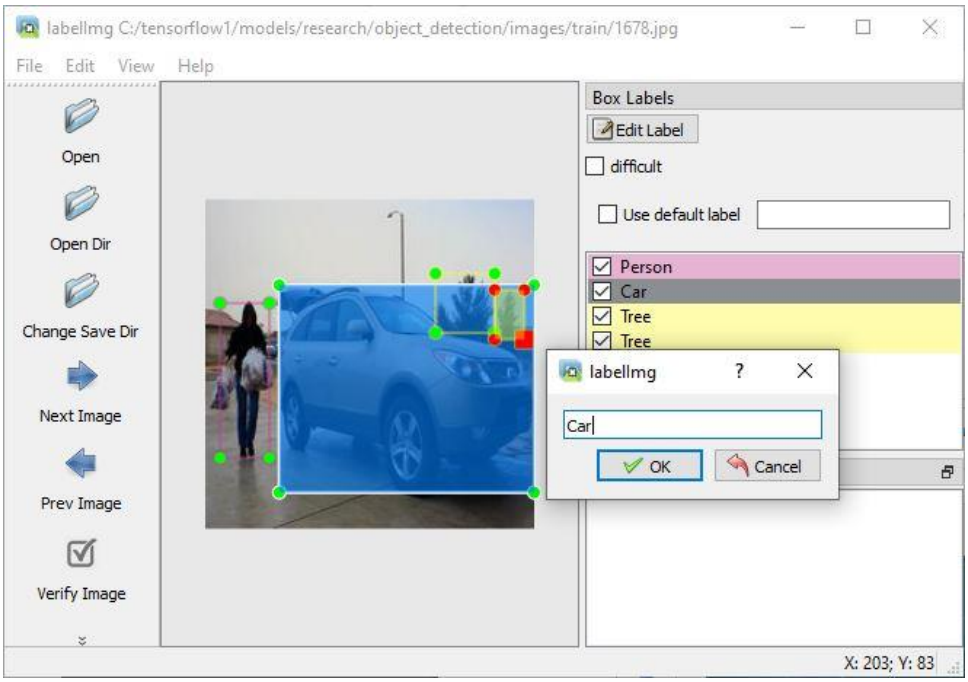


Figure 3.4: Étiquetage des images dans le dataset.

3. Toutes les étiquettes des images sont stockées au format XML. Des fichiers XML séparés sont générés pour chaque image de l'ensemble de données. Ces fichiers XML, comme illustré à la figure 3.10, contiennent le nom de fichier de l'image, le chemin de l'image, le libellé de l'image et les coordonnées des cadres de délimitation (bounding box) de l'image. Ces fichiers sont appelés fichiers d'annotations qui sont utilisés pour entraîner les algorithmes de détection d'objets.

3.7 Implémentation

La procédure d'implémentation suivie dans ce mémoire est celle illustrée dans la figure 3.5.

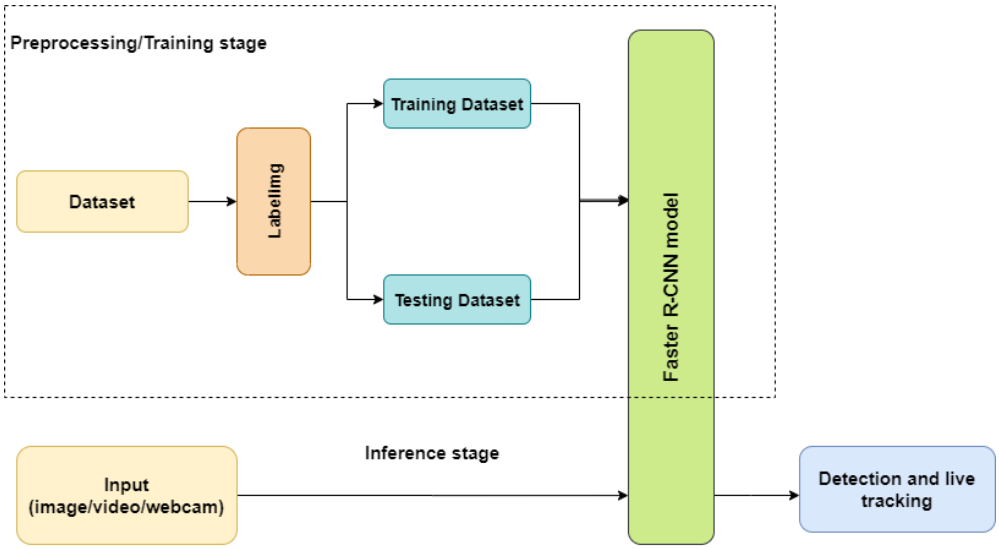


Figure 3.5 : Procédure d'implémentation.

3.7.1 Tensorflow Object Detection API

Il s'agit d'un framework open source construit sur Tensorflow qui facilite le développement, la formation et le déploiement d'algorithmes de détection d'objets. L'API Tensorflow Object Detection possède une vaste collection de modèles/algorithmes appelés zoo modèle. Dans ce mémoire, l'API Tensorflow Object Detection est utilisée pour implémenter la méthode Faster R-CNN.

3.7.2 Conversion des données d'entrainement

Depuis la version Tensorflow 1.4, il est recommandé d'utiliser le format TFRecord comme entrée pour réduire le temps de lecture des données ainsi pour une performance optimale. Le TFRecord est un format de stockage binaire créé et optimisé pour les applications Tensorflow qui lit simultanément les annotations et les images, puis il les mélange et les écrit dans des fichiers TFRecord. Il facilite la combinaison de plusieurs ensembles de données et s'intègre de manière transparente avec la fonctionnalité d'importation et de prétraitement de données fournie par la bibliothèque.

3.7.3 Configuration

Plusieurs modifications ont été apportées aux fichiers de configuration par défaut du Faster R-CNN fourni par l'API de détection d'objets TensorFlow, comme :

1. **Nombre de classes :** Les classes ici ne sont rien d'autre que le nombre d'objets que Faster RCNN devraient apprendre et détecter après l'entraînement. Dans ce cas, la méthode est chargée de détecter cinq classes, à savoir personne, voiture, vélo, arbre ou panneau de signalisation.
2. **Taux d'apprentissage:** un taux d'apprentissage par défaut de 0,0002 a été utilisé pour entraîner le modèle.
3. **Carte des étiquettes (Label map):** Une carte d'étiquettes indique ce qu'est chaque objet dans une image, en mappant les noms de classe aux numéros d'identification de classe. Pour que l'algorithme sache ce qu'est l'objet, la carte d'étiquettes est écrite comme indiqué dans la figure 3.11.
4. **Architecture CNN (Backbone):** Dans ce travail, le réseau neuronal convolutif Inception-V2 a été utilisé à l'intérieur pour l'entrainement de Faster RCNN. L'architecture est indiquée dans la figure 1.26.
5. **Taille du lot (Batch size):** la taille du lot fait référence au nombre d'échantillons d'apprentissage qui peuvent être utilisés par l'algorithme en une itération. Parce que la taille du lot est directement proportionnelle à la consommation de VRAM, compte tenu des spécifications matérielles du système comme indiqué dans le tableau 3.1. La taille du lot pour est définie par 1 pour un processus plus efficace.
6. **Fractionnement de dataset (Dataset splitting) :** L'ensemble de données utilisé dans ce mémoire est divisé de telle manière que 80% des images sont utilisées pour l'entraînement tandis que les 20% restants sont utilisés pour les tests d'évaluation. Cette règle générale suivie par divers chercheurs lors de la fragmentation de l'ensemble de données (48) (49) (50).
7. **Fonction d'optimisation:** SOFTMAX
8. **IoU sommet :** 0.6
9. **Nombre de pas**: 7200

3.8 Entrainement du modèle

Après avoir terminé toutes les étapes mentionnées ci-dessus et effectué les modifications nécessaires dans le fichier de configuration, le processus d'entrainement est initialisé. Le nombre d'étapes et la perte de classification dans chaque étape peuvent être vus à l'écran, comme le montre la figure 3.6. On peut noter que la perte de classification commence à une

valeur vraiment élevée et diminue progressivement à mesure que l'algorithme apprend au fur et à mesure que les itérations progressent. Ceci a été visualisé sous la forme d'un graphe, à l'aide de TensorBoard montré dans la figure 3.8. «Global_step» représente le nombre d'itérations ou le numéro de lot en cours de traitement. La valeur de «Loss» fait référence au cout payé pour l'inexactitude des prévisions.

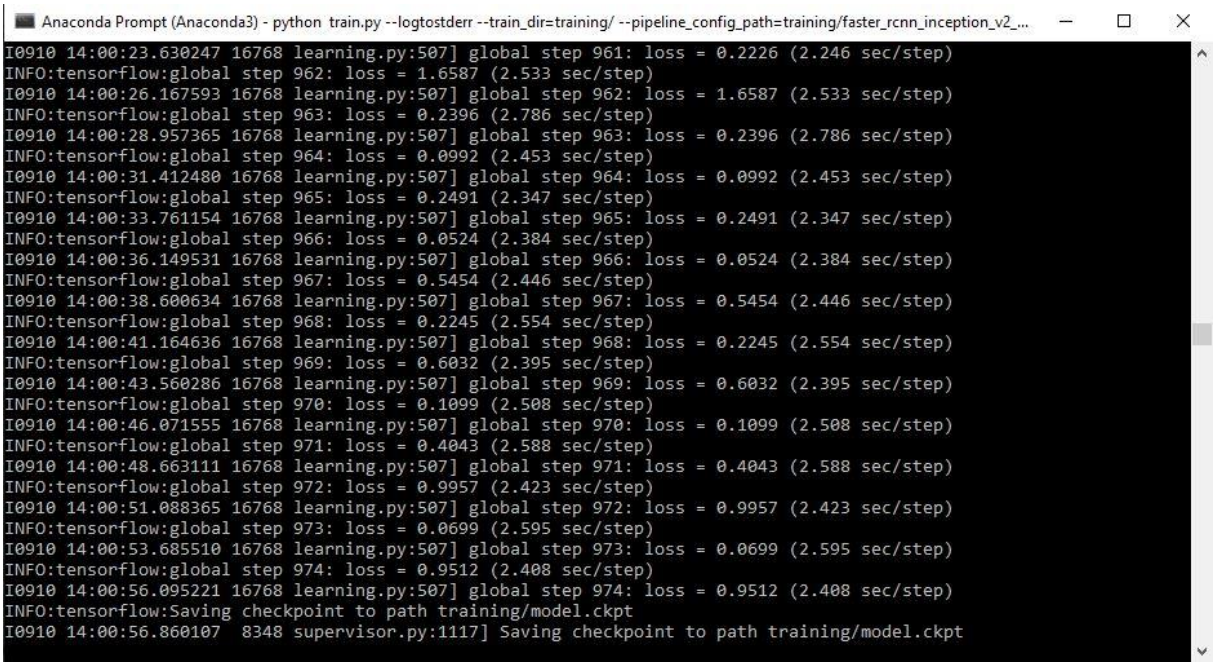


Figure 3.6 La perte affichée après chaque étape d'entrainement.

Pendant le processus d'entrainement, cette perte de classification continue de diminuer à mesure que l'algorithme continue à apprendre à chaque étape. Par conséquent, ce processus se poursuit jusqu'à ce que la perte soit constante. En général, moins de perte implique un meilleur entrainement du modèle. «sec/step» est le temps nécessaire pour traiter cette étape correspondante.

3.9 Les métriques de performance

La métrique d'évaluation la plus utile et la plus couramment utilisée pour la détection d'objet est la moyenne de la précision moyenne «*mean Avergae Precision*» (mAP). Cette métrique d'évaluation est utilisée par la plupart des frameworks modernes de détection d'objets et de segmentation d'instances pour comparer leurs performances. Pour comprendre mAP, il faut

d'abord avoir une intuition de sensibilité et de spécificité, qui sont les mesures statistiques de la performance d'un test de classification binaire.

La sensibilité est également appelée le vrai taux positif ou le rappel(Recall). Il mesure la proportion de positifs réels correctement identifiés. Pour la détection d'objet, la déclaration des vrais positifs (TP) et des faux positifs (FP) dépend d'un seuil IoU prédéfini qui est généralement défini sur 50%.

La spécificité est également appelée le vrai taux négatif. Pour mieux comprendre on a :

Vrai Positif (TP): Il y a une personne, et l'algorithme détecte comme une personne.

Faux positif (FP): Il n'y a pas de personne, mais l'algorithme détecte la personne.

Faux négatif (FN): Il y a une personne, mais l'algorithme ne détecte pas la personne.

Vrai négatif (TN): Il n'y a pas de personne et rien n'est détecté.

Accuracy : L'exactitude est une mesure qui indique si un modèle/algorithme est correctement entraîné et comment il exécute. L'Accuracy est calculée à l'aide de la formule suivante (51).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Precision : La précision est définie par le nombre de vrais résultats positifs divisé par le nombre total de résultats positifs prédits par le classificateur.

$$Precision = \frac{TP}{(TP + FP)}$$

Recall : Le rappel est défini par le nombre de vrais résultats positifs divisé par la somme des vrais positifs et des faux négatifs.

$$Recall = \frac{TP}{(TP + FN)}$$

Matrice de confusion : Est une technique permettant de résumer les performances d'un algorithme de classification. Une matrice de confusion $M_{i,j}$est égal au nombre d'observations connues comme appartenant au groupe i et prédites comme appartenant au groupe j.

3.10 Résultats expérimentaux

Une fois le processus d'apprentissage terminé, nous allons discuter les résultats obtenus :

- La figure 3.7 est la collection d'images montrant les résultats de détection obtenus en exécutant Faster RCNN sur des images pris dans des voitures.

Figure 3.7 Résultats de détection de Faster RCNN.

- On peut observer que Faster RCNN était capable de détecter des différents objets comme voitures, personnes, bicyclette, arbre et panneau de signalisation avec un niveau de confiance maximal de 99% même lorsque ces objets sont vus sous différents angles, postures et conditions d'éclairage.

- Le nombre de vrais positifs, de faux positifs, de faux négatifs et de vrais négatifs est obtenu en utilisant une matrice de confusion. La classe « rien » ca sert à une classe pour détection vide (IoU < 0.6). Les résultats ont été représentés par le tableau ci-dessous:

		Valeurs prédites					
Valeurs actuelles		**Vélo**	**Voiture**	**Panneau_s**	**Personne**	**Arbre**	**Rien**
	Vélo	**426**	0	0	3	1	49
	Voiture	0	**562**	6	1	12	67
	Panneau_s	0	7	**447**	2	26	215
	Personne	1	6	4	**537**	7	158
	Arbre	2	2	11	0	**432**	254
	Rien	65	197	153	259	752	**3**

Tableau 3.2 : Valeurs calculées de Faster R-CNN.

- D'après la matrice de confusion, on peut obtenir la précision et le rappel (recall) pour chaque classe qui ont été mentionnés dans le tableau 3.3

Classe	**Précision@0.5**	**Recall@.5**
Vélo	0.86	0.89
Voiture	0.73	0.87
Panneau_s	0.72	0.64
Personne	0.67	0.75
Arbre	0.35	0.62

Tableau 3.3 : Taux de précision et rappel.

A partir de graphe de figure 3.8, nous présentons le taux de perte obtenu qui est **0.4**

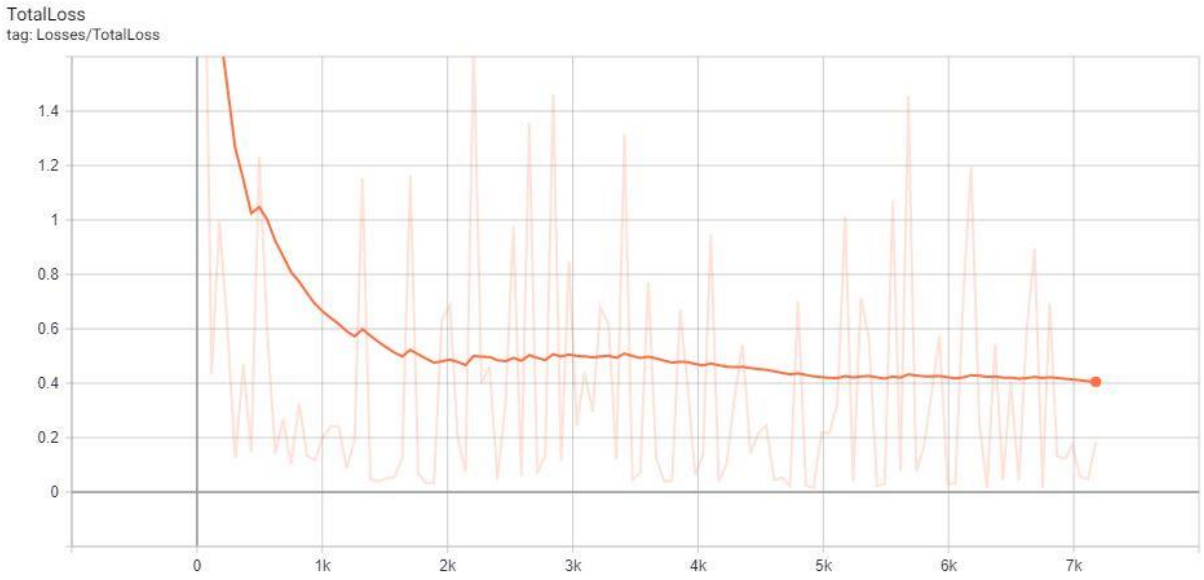


Figure 3.8: Graphe de taux de perte.

- Le Faster R-CNN a terminé le traitement et la détection dans un environnement urbain (plus complexe) en **4.81** secondes.

Figure 3.9 : Affichage du temps de détection.

3.11 Discussion :

D’après le tableau 3.2 et tableau 3.3 on observe que Faster R-CNN est très précis dans la pluparts des classes et qui atteint un taux de précision de **86%** dans la classe « Vélo », par contre on voit un taux de **35%** dans la classe « Arbre », et cela est dû au fait que le nombre de l’ensemble d'entraînement pour cette classe n'était pas suffisant. Nous avons réduit le taux de perte de **3.2** jusqu’à **0.4** après 7200 étapes en environ de 5 heures. La figure 3.9 montre que la détection été très précise et variante malgré l’environnement complexe d’objet (objet qui cache d’autre objet). Un temps de détection **4.81** secondes sur une image et **5** FPS sur une vidéo comme indiqué dans la figure 3.9, ce qui est considéré comme une détection assez lente en raison de l'environnement matériel. Les résultats obtenus sont relativement bons compte tenu l’environnement matériel et la taille réduite d’images de l’ensemble de données.

3.12 Conclusion :

Nous avons présenté dans ce chapitre l’implémentation de l’approche de détection des objets basée Faster R-CNN dans un environnement urbain pour qui peut aider une voiture autonome. L'approche proposée pour cette détection utilise une architecture particulière de réseau de neurones convolutionnels « Inception_V2 ». Les tests ont été effectués étaient sur des images externes prises depuis une voiture.

Conclusion générale

Dans ce mémoire, la méthode Faster R-CNN est identifiés comme un algorithme d'apprentissage profond approprié et il est entraîné avec un ensemble de données contenant des cinq différent classes « vélo, voiture, panneau de signalisation, personne, arbre » dans divers scénarios, visant à détecter et suivre en temps réel le maximum de ces objet dans un environnement complexe, par conséquent, réduire le nombre de décès dus aux accidents de la circulation routière. Le modèle entraîné a été évalué sur plus de 2000 images collectées à partir d'Open Images Dataset V4 et les résultats obtenus sont collectés et analysés à l'aide de mesures d'exactitude, précision, rappel, temps d'entraînement et de vitesse de détection. Après avoir analysé les résultats, il s'avère que Faster RCNN a montré des bons résultats par rapport à le facteur de précision. D'autre part, des métriques telles que le temps d'apprentissage, la vitesse de détection sont nécessaires si d'autres chercheurs reproduisent des travaux similaires, car elles les aideront à acquérir des connaissances sur le temps nécessaire pour entraîner les modèles ainsi que sur la vitesse de détection des entités ou des objets requis. Autre méthode peut être préférée si la vitesse de détection est plus importante. Mais comme ce mémoire vise à résoudre un problème de monde réel, on peut conclure que Faster R-CNN est la méthode de choix pour détecter et suivre les objets en temps réel et efficacement dans un environnement urbain.

Perspective :

Très peu de recherches ont été faites au sujet des voitures autonomes, de là nous envisageons comme perspectives de recherches futures :

- Inclure plusieurs classes comme les ralentisseurs de route « dos-d'âne », passage piéton, trottoir… etc. ce qui nécessitent un grand ensemble de données.
- Améliorer l'architecture Faster R-CNN de manière à ce qu'elle puisse détecter plus rapidement.
- Essayer de travailler sur un environnement matériel plus puissant.

Références

1. **Dalal, Navneet et Triggs, Bill.** Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition.* 2005, Vol. 1, pp. 886–893.

2. **Viola, Paul, Jones, Michael J et Snow, Daniel.** Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision.* 63, 2005, 2, pp. 153–161.

3. **Lowe, David G.** Object recognition from local scale-invariant features. *The proceedings of the seventh IEEE international conference.* IEEE, 1999, Vol. 2, pp. 1150–1157.

4. **Girshick, Ross, et al.** Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014, pp. 580–587.

5. **Girshick, Ross.** Fast R-CNN. *Proceedings of the IEEE international conference on computer vision.* 2015, pp. 1440–1448.

6. **Uijlings, Jasper RR, et al.** Selective search for object recognition. *International journal of computer vision.* 104, 2013, 2, pp. 154–171.

7. **Endres, Ian et Hoiem, Derek.** Category independent object proposals. *European Conference on Computer Vision.* Springer, 2010, pp. 575–588.

8. **Carreira, Joao et Sminchisescu, Cristian.** CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis & Machine Intelligence.* 2011, 7, pp. 1312–1328.

9. **Arbeláez, Pablo, et al.** Multiscale combinatorial grouping. *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014, pp. 328–335.

10. **Zitnick, C Lawrence et Dollár, Piotr.** Edge boxes: Locating object proposals. *European conference on computer vision.* Springer, 2014, pp. 391–405.

11. **Ren, Shaoqing, et al.** Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems.* 2015, pp. 91–99.

12. **He, Kaiming, et al.** Mask R-CNN. *Computer Vision (ICCV).* IEEE, 2017, pp. 2980–2988.

13. **Goodfellow, Ian, et al.** Deep learning. *MIT press Cambridge.* 2016, Vol. 1.

14. **McCulloch, Warren S et Pitts, Walter.** A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics.* 5, 1943, 4, pp. 115–133.

15. **Kawaguchi, Kiyoshi.** The McCulloch-Pitts Model of Neuron. [En ligne] [Citation : 13 09 2020.] http://wwwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node12.html.

16. **O'Shea, Keiron et Nash, Ryan.** An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458.* 2015.

17. **Fukushima, Kunihiko.** Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks.* 1, 1988, 2, pp. 119–130.

18. **Deng, Li.** The mnist database of handwritten digit images for machine learning [best of the web]. *IEEE Signal Processing Magazine.* 29, 2012, 6, pp. 141–142.

19. **Rumelhart, David E, Hinton, Geoffrey E et Williams, Ronald J.** Learning representations by back-propagating errors. *Nature.* 323, 1986, 6088, p. 533.

20. **LeCun, Yann, et al.** Backpropagation applied to handwritten zip code recognition. *Neural computation.* 1, 1989, 4, pp. 541–551.

21. **LeCun, Yann, et al.** Gradient-based learning applied to document recognition. *Proceedings of the IEEE.* 86, 1998, 11, pp. 2278–2324.

22. **al., Lieve Hamers et.** Similarity measures in scientometric research: The jaccard index versus salton's cosine formula. *Information Processing and Management.* 25, 1989, 3, pp. 315–18.

23. **Krizhevsky, Alex, Sutskever, Ilya et Hinton, Geoffrey E.** ImageNet Classification with deep convolutional neural networks. *Communications of the ACM.* 60, 2012, 6, pp. 84–90.

24. **Zeiler, Matthew D. et Fergus, Rob.** Visualizing and understanding convolutional networks. *ECCV.* LNCS 8689, 2014, pp. 818–833.

25. **Hassan, Muneeb ul.** VGG16 – Convolutional Network for Classification and Detection. *NeuroHive.* [En ligne] 2018. [Citation : 21 10 2020.] https://neurohive.io/en/popular-networks/vgg16/.

26. **Szegedy, Christian, et al.** Going deeper with convolutions. *arXiv preprint arXiv:1409.4842.* 2014.

27. **Karim, Raimi.** Illustrated: 10 CNN Architectures. *Towards Data Science.* [En ligne] 2019. [Citation : 21 10 2020.] https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d.

28. **He, Kaiming, et al.** Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385.* 2015.

29. **Chollet, François.** Xception: Deep Learning with Depthwise Separable Convolutions. *arxiv preprint arXiv:1610.02357.* 2017.

30. **Xie, Saining, et al.** Aggregated Residual Transformations for Deep Neural Networks. *arXiv preprint arXiv:1611.05431.* 2017.

31. **Sandler, Mark, et al.** MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv preprint arXiv:1801.04381.* 2019.

32. **Mingxing, Tan et Quoc, V. Le.** EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946.* 2019.

33. **Wang, Chien-Yao, et al.** CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN. *arXiv preprint arXiv:1911.11929.* 2019.

34. **Sermanet, Pierre, et al.** Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229.* 2013.

35. **Russakovsky, Olga, et al.** ILSVRC-2013. [En ligne] [Citation : 09 09 2020.] http://www.image-net.org/challenges/LSVRC/2013/results.php.

36. **Wei, Liu, et al.** SSD: Single Shot MultiBox Detector. *arXiv preprint arXiv:1512.02325.* 2016.

37. **Tsung-Yi, Lin, et al.** Focal Loss for Dense Object Detection. *arXiv preprint arXiv:1708.02002.* 2017.

38. **Bochkovskiy, Alexey, Wang, Chien-Yao et Liao, Hong-Yuan Mark.** YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934.*

39. What is Python? *Python For Beginners.* [En ligne] [Citation : 13 09 2020.] https://www.pythonforbeginners.com/learn-python/what-is-python/.

40. **Tensorflow.** . [En ligne] https://www.tensorflow.org/.

41. Cython: C-Extensions for Python. [En ligne] https://cython.org/.

42. Overview — Pillow (PIL Fork) 7.2.0 documentation. [En ligne] https://pillow.readthedocs.io/en/stable/handbook/overview.html.

43. lxml - Processing XML and HTML with Python. [En ligne] https://lxml.de.

44. Project Jupyter. [En ligne] https://www.jupyter.org.

45. Matplotlib: Python plotting — Matplotlib 3.3.1 documentation. [En ligne] https://matplotlib.org.

46. Pandas: powerful Python data analysis toolkit — pandas 1.1.2 documentation. [En ligne] https://pandas.pydata.org/pandas-docs/stable/.

47. OpenCV. [En ligne] https://opencv.org/about/.

48. **Ananth, Shilpa.** Faster R-CNN for object detection. *Towards Data Science.* [En ligne] 2019. [Citation : 13 09 2020.] https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46.

49. **Elhoseiny, M., Bakry, A. et Elgammal, A.** Multiclass object classification in video surveillance systems-experimental study. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 2013, pp. 788–793.

50. **Wang, D. Z. et Posner, I.** Voting for voting in online point cloud object detection. *in Robotics: Science and Systems.* 607, 2015, Vol. 1, 3, pp. 10–15.

51. **Google developers.** Classification : justesse, Cours d'initiation au machine learning. [En ligne] 2020. [Citation : 13 09 2020.] https://developers.google.com/machine-learning/crash-course/classification/accuracy.

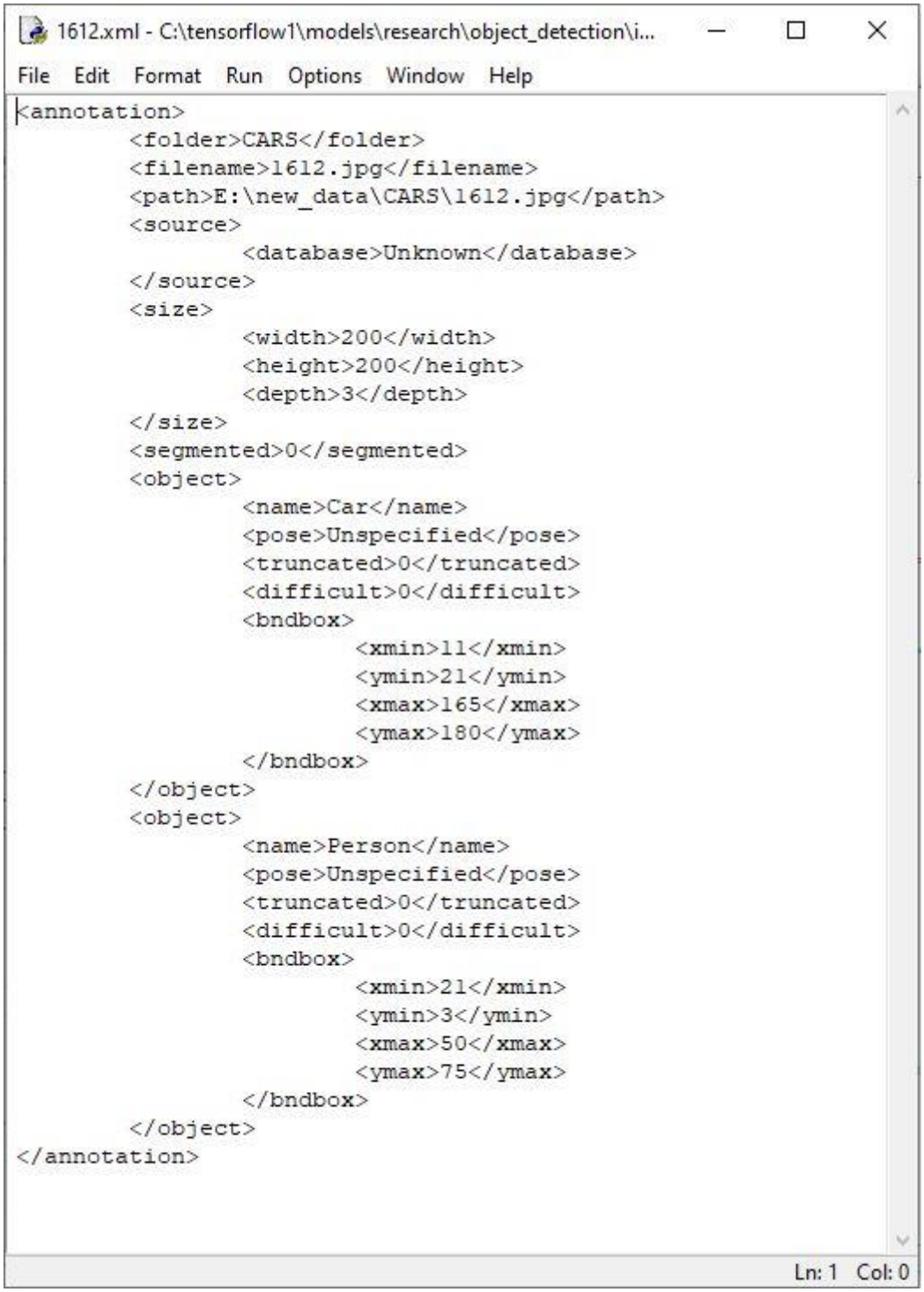


Figure 3.10: Fichier d'annotation XML généré par l'outil LabelImg.

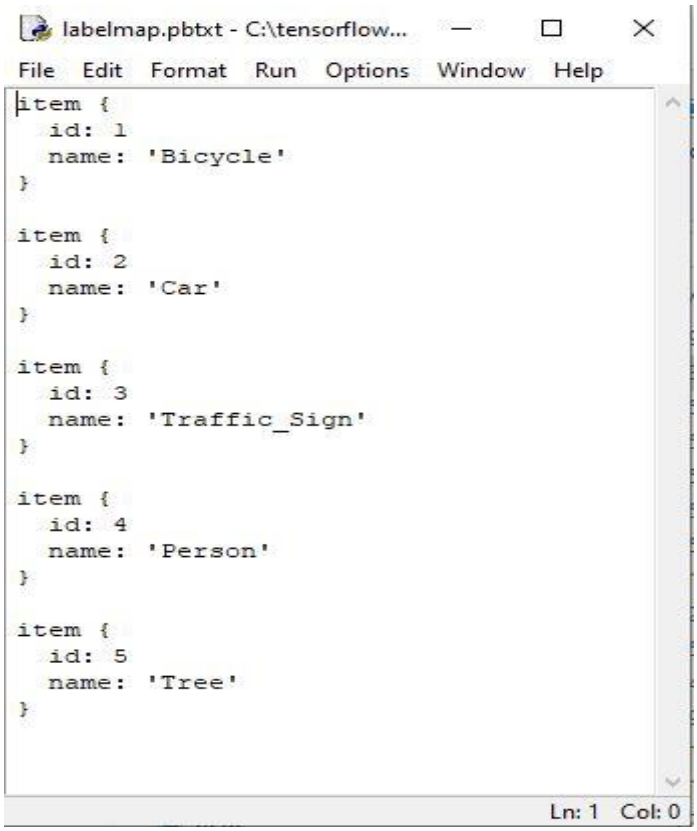


Figure 3.11 : Carte des étiquettes

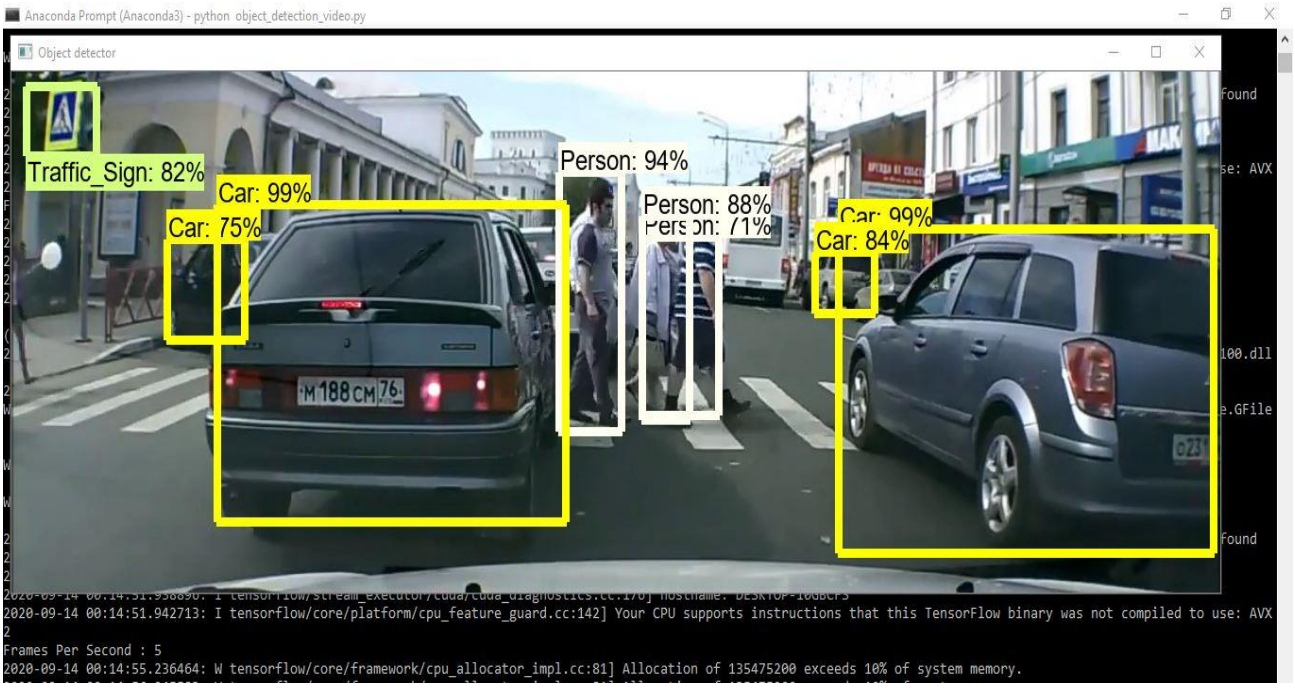


Figure 3.12: Détection en temps réel.

Faculté des Mathématiques, d'Informatique et des Sciences de la matière,
Université de 8 Mai 1945 – Guelma, Algérie