****

**Détection d’objet dans une image par apprentissage profond**

Implémentation d’un R-CNN

**Binôme : Thème proposé par :**

KADEM Cherifa Pr .Nicolas Thome

RAHMOUNI Imad

Sommaire

[1. Introduction : 3](#_Toc510095759)

[2. « Région Proposal » : 4](#_Toc510095760)

[2.1. Les méthodes de proposition de région 4](#_Toc510095761)

[2.1.1. La recherche sélective : 4](#_Toc510095762)

[3. Les Réseaux de Neurones Convolutionnels (CNN / ConvNets) 6](#_Toc510095763)

[3.1 Définition 6](#_Toc510095764)

[3.2 Architecture : 7](#_Toc510095765)

[4. R-CNN : 7](#_Toc510095766)

[4.1 Proposition de région : 8](#_Toc510095767)

[4.2 Extraction de vecteur de caractéristique 8](#_Toc510095768)

[4.3 Apprentissage du modèle : 9](#_Toc510095769)

[5. Travail effectué : 9](#_Toc510095770)

[5.1. Présentation des données et environnement de travail : 9](#_Toc510095771)

[5.1.1. Les données : 9](#_Toc510095772)

[5.1.2. Outils et environnement de travail : 10](#_Toc510095773)

[5.2. Implémentation du modèle 10](#_Toc510095774)

[5.2.1. Région Proposal : 11](#_Toc510095775)

[5.2.2. Labélisation et région d’intérêt : 12](#_Toc510095776)

[5.2.3. Extraction du vecteur de caractéristiques « Deep Features » : 13](#_Toc510095777)

[*5.2.3.1.* *Choix du CNN :* 13](#_Toc510095778)

[5.2.4. Apprentissage du classifieur : 14](#_Toc510095779)

[5.2.5 Evaluation du modèle : 15](#_Toc510095780)

# Introduction :

Avec la montée en puissance des véhicules autonomes, la vidéosurveillance intelligente, la détection faciale, la robotique et diverses applications de comptage de personnes, les systèmes de reconnaissance visuelle sont de plus en plus demandés.

Ces systèmes impliquent non seulement la reconnaissance et la classification de chaque objet dans une image, mais la *localisation de* chacun d'eux en dessinant la zone de délimitation appropriée autour d'elle et donc d'être capable de percevoir le monde qui nous entoure.

**Contexte :**

L’évolution des performances dans le domaine de détections automatique d’objet dans les images à quelque peu stagné au début des années 2010 où des techniques basées sur les histogrammes orientés telles que SIFT ou HOG[[1]](#footnote-1) étaient principalement utilisées. D’autre part, les réseaux de neurones à convolutions (CNN) dans le domaine de l’apprentissage profond ont largement perdu en popularité avec l'essor des machines à vecteurs support (SVM).

En 2012, *Krizhevsky et al* **[8]** ont ravivé l'intérêt pour les CNN en enregistrant une excellente performance dans la classification d’images lors du challenge de vision par ordinateur à grande échelle (ILSVRC[[2]](#footnote-2)). Leur succès a résulté de l’utilisation d’un CNN entraîné sur une base de données de grande taille de plus de 1,2 million d'images étiquetées.

Une question s’est donc soulevée dans la communauté de l’intelligence artificielle et vision par ordinateur : Dans quelle mesure les résultats de la classification par CNN se généralisent-ils aux résultats de détection des objets ?

*Ross Girshick et al* **[3]** ont répondu à cette question en comblant le fossé entre la classification des images et la détection d’objet. Leur document est le premier à montrer qu'un CNN peut présenter des performances sur la détection d’objet nettement supérieurss aux systèmes basés sur des caractéristiques tirées de techniques d’histogrammes orientés plus simples. Leur modèle consistait à tirer de l’image un certain nombre de régions, les passer à travers un CNN pour en tirer les caractéristiques puis de classifier ces caractéristiques selon les classes d’objets qu’elles représentent. Leur algorithme combinant les « régions » et les « CNN », ils l’ont baptisé « R-CNN »

Nous allons donc dans ce document étudier dans un premier temps le modèle R-CNN, expliqué le concept et les différents aspects de l’algorithme puis nous allons procéder à l’implémentation d’un modèle d’apprentissage inspirée par le R-CNN.

# « Région Proposal » :

Les algorithmes de proposition de région identifient des objets potentiels dans une image en utilisant la segmentation. Dans la segmentation, nous regroupons les régions adjacentes qui sont similaires en fonction de certains critères tels que la couleur, la texture, etc.

Une propriété importante d'une méthode de proposition de région est d'avoir un très **haut rappel.** C'est juste une manière de dire que les régions qui contiennent les objets que nous recherchons doivent figurer dans notre liste de propositions de régions. Pour ce faire, notre liste de propositions de régions peut finir par avoir beaucoup de régions qui ne contiennent aucun objet. En d'autres termes, il est acceptable que l'algorithme de proposition de région produise beaucoup de faux positifs tant qu'il capture tous les vrais positifs. La plupart de ces faux positifs seront rejetés par l'algorithme de reconnaissance d'objet dans notre cas le CNN.

## Les méthodes de proposition de région

Il existe Plusieurs méthodes de proposition de région telles que

1. [Objet](http://groups.inf.ed.ac.uk/calvin/objectness/)
2. [Min-Cuts paramétrés contraints pour la segmentation automatique d'objets](http://www.maths.lth.se/matematiklth/personal/sminchis/code/cpmc/index.html)
3. [Propositions d'objets indépendants de catégorie](http://vision.cs.uiuc.edu/proposals/)
4. [Primaire aléatoire](http://www.vision.ee.ethz.ch/~smanenfr/rp/index.html)
5. [Recherche sélective](http://koen.me/research/selectivesearch/)

Parmi ces méthodes de proposition de région, la recherche sélective sera utilisée dans notre projet.

### La recherche sélective :

La recherche sélective **[2]** est un algorithme de proposition de région utilisé dans la détection d’objet. Il est conçu pour être rapide avec un rappel très élevé. Il est basé sur le calcul du groupement hiérarchique de régions similaires en fonction de la compatibilité des couleurs, des textures, des tailles et des formes.

La recherche sélective commence par sur-segmenter l'image en fonction de l'intensité des pixels en utilisant une méthode de segmentation. La sortie de l'algorithme est montrée ci-dessous. L'image sur la droite contient des régions segmentées représentées en utilisant des couleurs unies



Figure 1 : La méthode de segmentation appliquée sur une image

L'algorithme de recherche sélective prend ces over-segments comme entrée initiale et effectue les étapes suivantes :

1-Ajouter toutes les boîtes de délimitation correspondant aux parties segmentées à la liste des propositions régionales

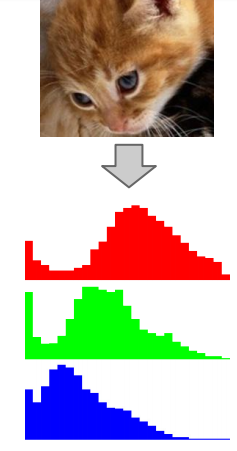
2-Regrouper les segments adjacents en fonction de la similarité

3-Aller à l'étape 1

A chaque itération, de plus grands segments sont formés et ajoutés à la liste des propositions de région. Par conséquent, l’algorithme crée des propositions régionales à partir de segments plus petits vers des segments plus importants selon une approche ascendante.

* + - 1. **Similarité :**

La recherche sélective utilise 4 mesures de similarité basées sur la compatibilité des couleurs, des textures, des tailles et des formes.

**Similitude de couleur**

Un histogramme de couleur de 25 cases est calculé pour chaque canal de l'image et les histogrammes de tous les canaux sont concaténés pour obtenir un descripteur de couleur résultant en un descripteur de couleur 25 × 3 = 75 dimensions.

La similitude de couleur de deux régions est basée sur l'intersection de l'histogramme et peut être calculée comme suit:

\[ s_{color}(r_i, r_j) = \sum_{k=1}^n min(c^k_i, c^k_j) \]

c^k_i est la valeur de l'histogramme pour k^{th} bin dans le descripteur de couleur

Figure 2:Histogramme de couleur

**Similitude de texture**

Les caractéristiques de texture sont calculées en extrayant des dérivées gaussiennes à 8 orientations pour chaque canal. Pour chaque orientation et pour chaque canal de couleur, un histogramme de 10 cases est calculé, résultant en un descripteur d'entités 10x8x3 = 240 dimensions.

La similarité de texture de deux régions est également calculée en utilisant des intersections d'histogrammes.

\[ s_{texture}(r_i, r_j) = \sum_{k=1}^n min(t^k_i, t^k_j) \]

 est la valeur de l'histogramme pour k^{th} bin dans le descripteur de texture



**Similarité de taille**

La similarité de taille encourage les petites régions à fusionner tôt. Il s'assure que les propositions de région à toutes les échelles sont formées à toutes les parties de l'image. Si cette mesure de similarité n'est pas prise en compte, une seule région continuera à engloutir une à une toutes les petites régions adjacentes et, par conséquent, des propositions régionales à plusieurs échelles seront générées à cet endroit seulement. La similarité de taille est définie comme :

\[ s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)} \]

où  size(im) est la taille de l'image en pixels.

**Compatibilité de forme**

La compatibilité de forme mesure à quel point deux régions ( r_i et r_j ) s'intègrent l'un dans l'autre. Si r_i rentre dans r_j nous aimerions les fusionner afin de combler les lacunes et s'ils ne se touchent pas, ils ne devraient pas être fusionnés. La compatibilité de forme est définie comme:

\[ s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)} \]

Où  size(BB{ij}) est une boîte englobante r_i et r_j .

Figure 3 comptabilisé de forme

**Similarité finale**

La similarité finale entre deux régions est définie comme une combinaison linéaire des 4 similitudes susmentionnées.

\[ s(r_i, r_j) = a_1s_{color}(r_i, r_j) + a_2s_{texture}(r_i, r_j) + a_3s_{size}(r_i, r_j)+ a_4s_{fill}(r_i, r_j) \]

Où r_i et r_j sont deux régions ou segments dans l'image et a_i \in {0, 1} indique si la mesure de similarité est utilisée ou non.

# Les Réseaux de Neurones Convolutionnels (CNN / ConvNets)

## 3.1 Définition

Un réseau neuronal convolutif (CNN) est constitué d'une ou plusieurs couches convolutives (souvent avec une étape de sous-échantillonnage), puis est suivi par une ou plusieurs couches entièrement connectées comme dans un réseau neuronal multicouche standard. L'architecture d'un CNN est conçue pour tirer parti de la structure 2D d'une image d'entrée (ou d'une autre entrée 2D telle qu'un signal de parole). Ceci est réalisé avec des connexions locales et des poids liés suivis d'une certaine forme de regroupement qui résulte en des caractéristiques invariantes de traduction. Un autre avantage des CNN est qu'ils sont plus faciles à former et ont beaucoup moins de paramètres que les réseaux entièrement connectés avec le même nombre d'unités cachées **[1].**

## 3.2 Architecture :

Un CNN consiste en un certain nombre de couches de convolution et de sous-échantillonnage éventuellement suivies de couches entièrement connectées. L'entrée d'une couche de convolution est un **m \* m \* r** image où m est la hauteur et la largeur de l'image et r est le nombre de canaux, par exemple une image RVB a r=3. La couche de convolution aura k filtres (ou noyaux) de taille **n x n x q** où n’est plus petit que la dimension de l'image et q peut être le même que le nombre de canaux r ou plus petit et peut varier pour chaque noyau. La taille des filtres donne lieu à la structure connectée localement qui sont chacune convoluées avec l'image pour produire k cartes de caractéristiques de taille **m−n+1**. Chaque carte est ensuite sous-échantillonnée avec un regroupement moyen ou maximal sur **p x p** régions contiguës où p est compris entre 2 pour les petites images (par exemple MNIST) et n'est généralement pas supérieur à 5 pour les plus grandes entrées. Avant ou après la couche sous-échantillonnage, un biais additif et une –linéarité sigmoïde sont appliqués à chaque carte de caractéristiques. Après les couches convolutives, il peut y avoir un nombre quelconque de couches entièrement connectées. Les couches densément connectées sont identiques aux couches d'un réseau neuronal multicouche standard.

Il existe plusieurs architectures dans le domaine des réseaux convolutifs. Dans ce projet on a utilisé le ResNet50. C’est un réseau résiduel développé par Kaiming He et al qui était le gagnant de ILSVRC 2015. Il comporte des connexions de sauts spéciaux et une utilisation intensive de [la normalisation par lots](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.fr&sl=en&sp=nmt4&u=http://arxiv.org/abs/1502.03167&xid=17259,15700022,15700043,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhi2mDRSLEdeVTmVdmJ5G-zWECccww) . L'architecture manque également de couches entièrement connectées à la fin du réseau **[1].**

# R-CNN :

R-CNN est l'abréviation de "Réseaux de neurones convolutionnels à base régionale". L'idée principale est axée sur deux concepts. En premier lieu, en utilisant la recherche sélective, on identifie un nombre gérable de régions candidates d'objet et de boîtes englobantes puis on extrait les fonctionnalités CNN de chaque région pour effectuer une classification.

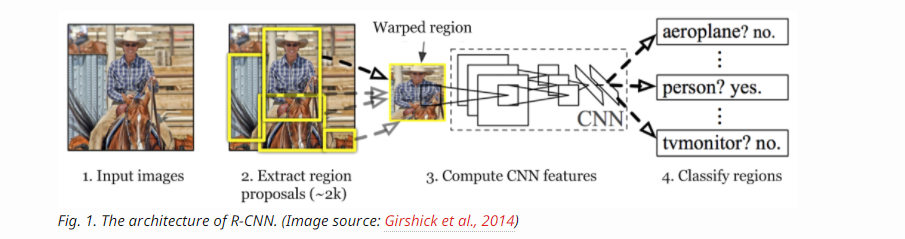


Figure 4 : Architecture du R-CNN (image source :Girshik et al.2014)

Le Modèle R-CNN se décompose principalement en 3 modules :

## Proposition de région :

Cette étape de l’algorithme consiste à extraire environ 2000 propositions des régions par recherche sélective, les régions peuvent contenir des objets cibles et sont de tailles différentes. Les régions candidates sont ensuite redimensionnées pour avoir une taille fixe comme requis par CNN prédéfinie par le Krizhevsky et al. **[8]**

## Extraction de vecteur de caractéristique

Le modèle extrait un vecteur de caractéristiques en 4096 dimensions de chaque proposition de région en utilisant le CNN décrit par Krizhevsky et Al. **[8].** Les caractéristiques sont calculées sur des images de 227 × 227 x 3 (En RVB) propagées à travers couches convolutives et de deux couches entièrement connectées.

**Transformation des images (warping) :** Afin d’extraire les caractéristiques d'une proposition de région, les données d’images de chaque région sont converties en une forme compatible avec le CNN (son architecture nécessite des entrées d'une taille fixe de 227 × 227 pixels). Parmi les nombreuses transformations possibles la plus simple est utilisée : Indépendamment de la taille ou du format de l'image, les régions candidates sont redimensionnées en la taille requise sans prendre en compte les proportions de l’image d’origine.



Figure 5 - Exemple de warping de régions candidates

## 4.3 Apprentissage du modèle :

**Préapprentissage supervisée du CNN :** le CNN est entrainé sur un grand ensemble de données (ILSVRC 2012)[[3]](#footnote-3) pour une tâche de classification (pas d'étiquettes de boîte englobante). Le modèle utilisé correspond presque à la performance de Krizhevsky et al. [8], avec un taux d'erreur du top-1 légèrement supérieur (2,2 points de plus) que celui de l'ensemble de validation ILSVRC 2012. Cet écart est dû à des simplifications dans le processus d’apprentissage.

**Réglage Fin (Fine tuning).** Pour adapter le CNN à la nouvelle tâche (détection) et le nouveau domaine (Pascal VOC), un réglage fin a été effectué sur le modèle en figeant tous les paramètres sauf ceux de la dernière couche qui ont été adaptés au nouveau domaine c’est-à-dire 21 classes de sortie.

**Entrainement du classifieur :** une fois le CNN pré-entrainé et affiné il est utilisé pour extraire les vecteurs de caractéristiques qui sont par la suite consommés par un SVM binaire formé pour chaque classe indépendamment. Les échantillons positifs sont des régions proposées avec un seuil de chevauchement IoU (intersection sur union)> = 0,3, alors que les échantillons négatifs ne sont pas pertinents pour la classifieur. Pour réduire les erreurs de localisation, un modèle de régression est formé pour corriger la fenêtre de détection prédite et son décalage par rapport aux boîtes englobantes.

# Travail effectué :

Dans cette partie nous allons procéder à l’implémentation d’un modèle R-CNN similaire à celui proposé par Ross Girshick *et al* ‎**[2]** que nous appellerons dans la suite « **modèle de référence** » ou « **modèle de base** » tout en apportant quelques modifications afin de l’adapter à notre cas d’utilisation.

## Présentation des données et environnement de travail :

### 5.1.1. Les données :

Nous allons travailler sur la même base d’images utilisée par le modèle de référence à savoir Pascal VOC 2007 [[4]](#footnote-4) Cette base se constitue comme suit :



Figure 6 - The PASCAL Visual Object Classes Challenge 2007

* 9 963 images entre l’ensemble d’apprentissage (training et validation) qui contient environ 5000 images et l’ensemble de test contient également environ 5000 images.
* Les images sont réparties en 20 classes (parmi les macro-catégories Person, Animal, Vehicle, Indoor), mais avec des étiquettes “multi-labels”. Par exemple, une image peut contenir à la fois une voiture et un vélo.
* 24 640 objets annotés dans les images avec la position des boîtes englobantes.
* Les images sont de tailles variables mais en moyenne de taille 500x300.

### 5.1.2. Outils et environnement de travail :

Nous avons effectué l’implémentation du modèle sur un environnement Python 3.6 doté des principaux outils de traitement de données et de Machine Learning (Numpy, Scikit-Learn et Tenseflow).

Pour les outils de Deep Learning nous avons opté pour la librairie Keras [[5]](#footnote-5) avec un Backend Tenserflow. Les calculs et visualisations des résultats se sont fait sur un document Jupyter Notebook que nous allons joindre à ce rapport et qui détaille toutes les étapes de la démarche.

|  |  |  |
| --- | --- | --- |
| Image result for python | Image result for keras | Image result for jupyter notebook |

Figure 7 - Technologies utilisées pour l'implémentation du modèle

## 5.2. Implémentation du modèle

Nous allons à présent détailler la démarche suivie dans l’implémentation du modèle R-CNN tout en précisant les modifications apportées par rapport au modèle de base et les motivations de ces divergences.

## **Région Proposal :**

Pour l’algorithme de proposition de régions nous avons opté pour le même choix à savoir le « Selective Search », et ce principalement pour sa rapidité mais aussi sa robustesse car il prend en compte les similarités de couleur, de taille et de texture des différentes parties de l’image. Nous avons utilisé une implémentation sur Python proposée par « AlpacaDB » **[4]** qui reprend l’algorithme de J.R.R. Uijlings et al *« Selective Search for Object Recognition »* **[5]** cité dans le modèle de référence.

Cet algorithme propose 3 différents paramètres :

* **Sigma :** qui est le paramètre écart-type du filtre gaussien utilisé pour atténuer les artéfacts de l’images sans pour autant altérer sa composition.
* **Min\_size :** Correspond à la taille minimale retrouvée dans les régions proposées.
* **Scale :** un paramètre définissant une échelle sur laquelle la comparaison des histogrammes est faites. Plus ce paramètres et grand plus on favorise des objets de taille importante dans l’image.

Les différents réglages de ces paramètres permettent de générer un nombre différent de régions pour chaque image. Nous rappelons que dans le modèle de base environs 2000 régions étaient générées par chaque images et ce dans un souci de scalabilités (s’adapter aux images de grande taille). .  
Comme dans notre cas nous allons nous limiter aux données de la base Pacal VOC2007, dont la taille moyenne des images et de 300x500 pixels nous avons opté pour le nombre de 200 régions par images ou chaque région est caractérisée par un rectangle ou 4 coordonnées qui correspondent à son positionnement dans l’image d’origine.

Ce choix nous semble judicieux car d’une part il permet de générer un nombre suffisant d’entrées pour l’apprentissage (pour 200 images nous obtenons 40.000 régions) et d’autre part le choix d’un nombre plus important engendrerai un temps de traitement extrêmement excessif notamment pour l’extraction des « Deep Features » que nous détaillerons dans la suite.

Ainsi, pour obtenir 200 régions par images nous avons effectué un « grid search » (recherche sur grille) des 3 paramètres définis précédemment. Nous avons donc gardé les paramètres suivants :

* **Sigma** : un écart type 0.2 qui représente la valeur par défaut préconisée par le concepteur.
* **Taille minimum** : 15x15 qui reste en adéquation avec les données traitées, car en fonctions des boîtes englobantes proposées par le dataset les plus petits objets sont à peu près de cette taille.
* **Scale** : en figeant les deux premiers paramètres une échelle fixée à 500 avait abouti au résultat escompté.

### Labélisation et région d’intérêt :

La deuxième étape pour préparer l’ensemble d’apprentissage consiste en la labellisation des régions, c’est-à-dire définir pour chaque région la classe d’objet qu’elle contient (éventuellement) parmi les 20 macro-classes de notre base de données sinon l’étiqueter en tant que « background » ou arrière-plan ce qui reviendra donc à créer une 21eme classe « background ».

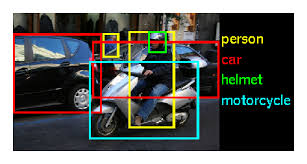


Figure 8 - Exemple de régions d'intêret et bounding box

Pour ce faire, nous disposons d’un fichier contenant pour chaque image les différents rectangles ou régions de l’image qui englobe un objet relatif à une classe.

Le but de cette étape est donc d’étiqueter les régions qui contiennent des objets, nous appellerons celles-ci « **régions d’intérêt** » ou « **fenêtres positives** », et par opposition les régions ne contenant aucun objet, donc arrière-plan, les « **fenêtres négatives »**.

Afin de définir les régions d’intérêt nous avons utilisé un critère de sélection défini comme IoU (intersection over union) ou l’intersection sur l’union qui consiste en la mesure de la superposition d’une région proposée sur une région « vraie » ou « groundtruth » c’est-à-dire qui est labelisée. Si la valeur de cette mesure est supérieure à un certain seuil (généralement 0.5) on peut considérer la région proposée comme étant une région d’intérêt.

**Exemple :**

|  |  |
| --- | --- |
| IoU>0.5 fenêtre positive région d’intérêt | IoU = 0 fenêtre négative arrière-plan |
|  |  |

Figure 9 - Exemple de mesure de IoU

### Extraction du vecteur de caractéristiques « Deep Features » :

La troisième étape de la démarche consiste en l’extraction des caractéristiques profondes à partir des entrées obtenues par « region proposal » en utilisant un CNN pré-entrainé pour une tache de classification (pas de détection d’objet) sur une large base d’image.

### ***Choix du CNN :***

Nous avons utilisé le réseaux ResNet50 **[6],** vainqueur du concours ILSVRC [[6]](#footnote-6) en 2015, le réseau avait atteint un taux d’erreur top-5 de 3.57% pour la tâche de classification sur la base ImageNet qui contient plus de 1,3 millions d’images pour 1000 classes différentes. De plus le vecteur de caractéristique correspondant à la dernière couche avant la couche de sortie du réseau est de taille relativement réduite de 2048. Pour rappel, le modèle R-CNN de référence utilise le réseau AlexNet **[7]** qui a atteint sur la même base d’images un taux d’erreur top-5 de 15% et se caractérise par un vecteur de caractéristique de dimension 4096.

Notre choix s’est donc porté donc sur le premier réseau de par sa performance et son efficience en mémoire pour notre cas d’utilisation.

#### *Extraction du vecteur de caractéristiques :*

Comme nous avons expliqué précédemment, nous avons opté pour la génération d’un nombre réduit de régions comparé au modèle de base et ce principalement pour des raisons de performance et coût de calcul.

En effet, afin de d’obtenir le vecteur caractéristique d’une seule région le CNN calcule plus de 23 millions de paramètres, pour 300 images nous obtenons 60000 régions et le temps de calcul devient rapidement trop important pour des machines non dédiées (absence de GPU).

Donc, pour chaque image, à l’aide de l’algorithme de « Selective Search » nous générons environs 200 région puis nous effectuons l’étiquetage des régions d’intérêt, ensuite chaque région subit les transformations nécessaires pour correspondre aux entrées du CNN (redimensionnement et centrage) afin d’extraire le vecteur d’entrée.

Une fois toutes les images traitées nous enregistrons dans un fichier tous les vecteurs caractéristiques obtenus avec les labels de classes correspondant.

En traitant les images d’apprentissage et de test nous obtenons donc deux fichiers qui correspondent à la base d’apprentissage et de test de notre classifieur.

Le schéma de cette étape se présente donc comme suit :

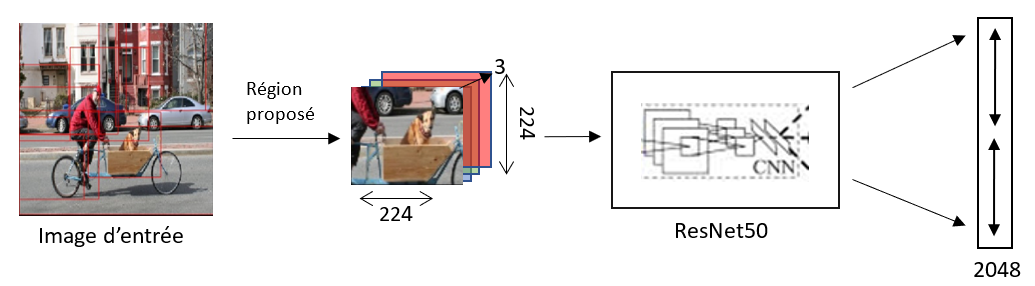


Figure 10 - Schéma de l'extraction des caractéristiques profondes

### Apprentissage du classifieur :

Cette dernière étape consiste à mettre en place un classifieur qui prend en entrée les vecteurs caractéristiques (correspondant à chaque régions) obtenus pendant l’étape précédente et en sortie les 20 classes de notre dataset en plus de la classe background.

#### **5.2.4.1 Architecture *du modèle :***

Pour le choix du modèle nous avons utilisé un modèle de régression logistique pour apprendre toutes les classes en même temps. Par définition pour chaque région correspond un seul label, cependant nous avons opté pour un modèle avec des sorties en multi-label et ce pour se rapprocher du modèle de base ou on utilise un SVM binaire par classe. L’architecture se résume comme suit:

* Un réseau de neurones entièrement connectées sans couches cachées : un vecteur d’entrée de dimension 2048 et une couche de sortie de taille 21, en tout nous disposons de 43 029 paramètres à estimer.
* Pour la fonction de coût nous avons utilisé l’entropie croisée binaire avec un seuil de 0.5 adaptée aux contexte multi-label.
* Une fonction d’activation sigmoïde en adéquation à l’usage de l’entropie croisée binaire.

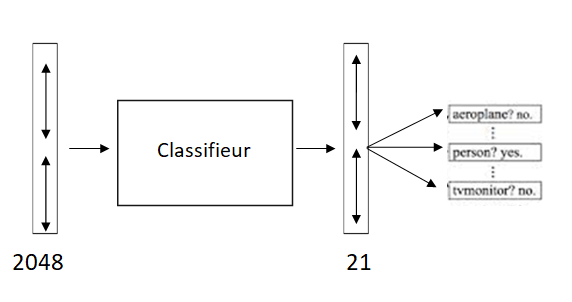


Figure 11 - Schéma global du classifieur

#### **5.2.4.2 Hyperparamètres *et validation :***

Pour l’apprentissage nous disposons d’un peu plus de 82000 exemples (générés à partir de 400 images), cependant la répartition de ces derniers est très déséquilibrée, en effet pour chaque image les fenêtres positives proposées sont extrêmement rares en comparaison aux fenêtres négatives d’arrière-plan. Nous disposions au final d’un ratio de 1/30 pour les exemples positifs et négatifs. Ce déséquilibre rend la convergence très lente voire impossible.

C’est pourquoi nous avons opté pour un apprentissage par mini-batch ou chaque mini batch est constitué pendant l’apprentissage en associant un vecteur de taille 32 comportant des exemples positifs et un autre de taille 96 comportant des exemples négatifs donnant ainsi lieu un batch de taille 128.

Cependant, cette technique risque d’engendrer un surapprentissage notamment des exemples positifs qui apparaissent plus souvent durant l’apprentissage, c’est pourquoi nous avons introduit un « Early stoping » calibré sur l’erreur d’un ensemble de validation d’environ 20000 exemples.

Enfin pour le pas d’apprentissage, nous avons utilisé « ADAM », une variante optimisée des SGD, en conservant les paramètres par défaut (0.01 pour le pas, et 0.8 pour le Momentum). Avec cette configuration nous avons remarqué une nette amélioration dans le temps de convergence.

### **5.2.5 Evaluation du modèle :**

Pour l’évaluation du modèle nous avons utilisé un ensemble de test comportant plus de 42.000 entrés avec un ratio d’exemples positif/négatif similaire à l’ensemble d’apprentissage.

La métrique qu’on a utilisée pour évaluer la performance par classe est la précision moyenne « average precision ». La performance globale du modèle est donnée par la moyenne des précisions de toutes les classes.

 La performance qu’on a obtenu sur l’ensemble de test était très mauvaise avec une **MAP = 12.3%,** nous rappelons que la mAP enregistré par le modèle de référence lors du Pascal VOC2012 challenge était de **53%**

Figure 12 – « Average Precision » par classe obtenues par le modèle

Nous pouvons expliquer cette mauvaise performance par :

1. Le nombre très réduit de données utilisées lors de l’apprentissage : en effet nous n’avons utilisé que 400 images parmi les 10000 disponibles dans la base, sur les 400 images utilisées seulement 1/30 constituait des régions d’intérêt. Nous avons cependant évoqué auparavant les raisons de cette limitation.

Nous pouvons vérifier cette hypothèse en consultant la répartition des objets par classe dans l’ensemble d’apprentissage, nous retrouvons que plus y’a d’exemples dans une classe donnée plus le résultat en test est (relativement) bon comme par exemple pour les classes « Person » et « Car » ou la performance atteint 33% et 44% pour 550 et 830 exemples respectivement. Pour les classes ou très peu d’exemples sont présents (44 exemples pour la classe « Sheep »), la performance en test avoisine les 1%.



Figure 13 - Nombe d'exemples par classe dans l'ensemble d'apprentissage

Il est donc possible d’atteindre de meilleures performances en présentant un nombre plus important d’exemple dans l’ensemble d’apprentissage. L’idéal étant naturellement d’utiliser les 5000 exemples disponibles dans la base d’images afin de générer environ 100.000 exemples de région.

1. Le fait d’avoir utilisé le CNN pré-entrainé sans avoir effectué de fine-tuning sur notre base d’images : Dans le modèle de référence un réglage fin a été effectué sur le CNN afin de l’adapter à la nouvelle tâche.

**Références :**

1. Cours de l’université de Stanford,[**http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/**](http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/)
2. Schuyler Smith , selective search for object recognition [**http://vision.stanford.edu/teaching/cs231b\_spring1415/slides/ssearch\_schuyler.pdf**](http://vision.stanford.edu/teaching/cs231b_spring1415/slides/ssearch_schuyler.pdf)
3. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. in ICCV, 2015
4. AlpacaDB: Selective Search Implementation for Python. <https://github.com/AlpacaDB/selectivesearch>
5. J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.
6. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition, ILSVRC, 2015
7. Very Deep Convolutional Networks for Large-Scale Image Recognition. <https://arxiv.org/abs/1409.1556>
8. A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012
9. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION : <https://arxiv.org/pdf/1412.6980.pdf>
10. A Beginner's Guide To Understanding Convolutional Neural Networks: <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>
11. Building powerful image classification models using very little data. <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>
12. Intersection over Union (IoU) for object detection. <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
13. Performance Metrics for Classification problems in Machine Learning. <https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b>
14. Duality and Geometry in SVM Classifiers. <http://www.robots.ox.ac.uk/~cvrg/bennett00duality.pdf>
15. How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras/ <https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/>

1. SIFT et HOG : <https://fr.wikipedia.org/wiki/Histogramme_de_gradient_orient%C3%A9> [↑](#footnote-ref-1)
2. Large Scale Visual Recognition Challenge 2012: [www.image-net.org/challenges/LSVRC/2012/](http://www.image-net.org/challenges/LSVRC/2012/) [↑](#footnote-ref-2)
3. Large Scale Visual Recognition Challenge 2012: [www.image-net.org/challenges/LSVRC/2012/](http://www.image-net.org/challenges/LSVRC/2012/) [↑](#footnote-ref-3)
4. PASCAL VOC2007. <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/> [↑](#footnote-ref-4)
5. Keras open source neural network library. <https://keras.io/> [↑](#footnote-ref-5)
6. ImageNet Large Scale Visual Recognition Competition: [www.image-net.org/challenges/LSVRC/](http://www.image-net.org/challenges/LSVRC/) [↑](#footnote-ref-6)