Cours de
Reconnaissance des Formes
Rapport du TP
Sujet:
Detection d'objets dans une image

IFI PROMTION 23
Redigé par:
Abdoul-Fatao Ouedraogo
OLEMBO Réel Devin Richmond

Février 2020

Contents

1 Introduction

A l'heure actuelle, les systèmes de vision sont de plus en plus répandus (webcam, caméscope.) et les caméras se sont installées partout dans notre quotidien. Elles sont utilisées pour réaliser de la vidéosurveillance (dans les magasins, rues ou aéroports), de l'aide à la conduite (aide au guidage ou détection d'obstacles), et bien d'autres applications encore. Pour l'être humain, voir est une tâche innée et nous ne mesurons souvent pas la difficulté pour obtenir les mêmes performances artificiellement. Malgré les avancées de la vision par ordinateur, les systèmes développés sont très loin d'égaler les performances de l'œil et du cerveau humains.

1.1 Contexte

Le but de la détection d'objets est de détecter toutes les instances d'objets d'une classe connue, tels que des personnes, des voitures ou des visages dans une image. Généralement, seul un petit nombre d'instances de l'objet sont présentes dans l'image, mais il existe un très grand nombre d'emplacements et d'échelles possibles où elles peuvent se produire et qui doivent être explorées d'une manière ou d'une autre.

Chaque détection est signalée avec une certaine forme d'informations de pose. Cela peut être aussi simple que l'emplacement de l'objet, un emplacement et une échelle, ou l'étendue de l'objet définie en termes de boîte englobante. Dans d'autres situations, les informations de pose sont plus détaillées et contiennent les paramètres d'une transformation linéaire ou non linéaire. Par exemple, un détecteur de visage peut calculer l'emplacement des yeux, du nez et de la bouche, en plus de la boîte englobante du visage. Un exemple de détection de vélo qui spécifie les emplacements de certaines pièces est illustré à la figure 1. La pose pourrait également être définie par une transformation tridimensionnelle spécifiant l'emplacement de l'objet par rapport à la caméra.

Les systèmes de détection d'objets construisent un modèle pour une classe d'objets à partir d'un ensemble d'exemples d'apprentissage. Dans le cas d'un objet rigide fixe, un seul exemple peut être nécessaire, mais plus généralement plusieurs exemples d'apprentissage sont nécessaires pour capturer certains aspects de la variabilité des classes.

1.2 Motivation

La résolution de la problématique sus-citée constituerait une grande avancée pour comprendre automatiquement les images, leur donner un sens. On ne peut qu'imaginer à quel point le champ d'applications est potentiellement très vaste que ce soit sur des thématiques de sûreté et de sécurité (vidéo-surveillance), de divertissement (par exemple au sein de jeux vidéo qui impliqueraient l'environnement réel du joueur), de santé, de logistique (gestion des stocks par exemple), ou de robotique (intelligence artificielle). L'intérêt pour ce sujet est très ancien mais

de véritables avancées n'ont été rendues possibles que par le développement de puissances de calculs suffisamment importantes pour traiter des données aussi complexes que les images, à partir des années 80. S'il est des tâches pour lesquelles les machines sont particulièrement indiquées (calculs, gestion de données), l'être humain se révèle pour le moment beaucoup plus efficace qu'elles pour reconnaître les objets au sein d'une image. Cela a d'ailleurs pu induire en erreur des chercheurs aussi chevronnés que Marvin Minsky, chercheur au MIT, qui au début des années 1980 pensait que la question de la vision était suffisamment simple pour demander à ses étudiants de la résoudre en tant que sujet d'étude pendant les vacances d'été ! L'on s'est bien sûr depuis rendu compte que c'était une question éminemment complexe, tant par la variété possible des conditions et des angles d'observation que la variabilité intra-classes (une personne assise et une personne debout sont par exemple différentes d'un point de vue graphique) ou encore par les différentes fonctions que peut occuper un même objet (un seau retourné doit-il être considéré comme une chaise ?) et qui dépendent du contexte.

1.3 Objectif

Les systèmes de détection d'objets ont pour objectif d'identifier un objet particulier dans une base de données image ou une vidéo. Cette identification est effectuée grâce à des attributs décrivant la forme, la couleur et/ou la texture. Notre projet consiste a prendre une image et identifier les différents objets que contient cette image, pour ce faire ,l'utilisateur donne une image d'entrée et l'application va identifier les objets de cette image.

Pour construire un modèle capable de détecter et de localiser des objets spécifiques dans les images.

Nous allons implémenter le Single Shot Multibox Detector (SSD) , un réseau populaire, puissant et particulièrement agile pour cette tâche.

1.4 Contributions

Nos contributions s'établissent à diffèrents niveaux :

- Au niveau de l'étude des systèmes existants, nous proposons une méthode permettant de révéler les différentes étapes nécessaires pour construire un algorithme classique pour la reconnaissance d'objets
- Sur un plan théorique, nous proposons une chaîne perpétuelle et optimale pour la reconnaissance d'objets, et proposons également un pipeline adapté pour traiter ce type de données en définissant des caractérisations propres à ces dernières afin d'obtenir une représentation proche de notre perception.
- Du point de vue expérimental, nous caractérisons l'influence de chaque étape de notre algorithme sur ces performance et son efficience. Les résultats obtenus montrent que notre système proposé peut être une première étape pour construire un système à la fois perceptuel et computationnel.

2 Analyse du sujet

2.1 Domaine d'application

Au delà d'être un problème général de la vision par ordinateur, la reconnaissance d'objets dans des catégories très variables est une étape indispensable pour de nombreuses applications spécifiques. On peut citer, parmi d'autres, quelques applications exemplaires pour illustrer les enjeux des travaux de recherche sur ce problème.

- La vidéo-surveillance : la reconnaissance et la détection automatique des personnes, des visages et des objets d'intérêt sur la très grande quantité d'images vidéo utilisées pour la surveillance.
- La recherche dans des bases d'images numériques : les systèmes actuels effectuent en général des requêtes à partir des mots clés associés aux images. On voudrait exploiter le contenu visuel des bases d'images sans avoir besoin de les indexer textuellement.
- La navigation des robots mobiles et des véhicules : on a besoin de reconnaître les obstacles, les objets divers dans l'environnement ou les autres véhicules pour la navigation automatique.

2.2 Problème visé

Aujourd'hui, la vision par ordinateur est utilisée par tous dans nos vies quotidiennes(application, vidéo surveillance,automates, etc..).Il est nécessaire pour nous d'avoir des systèmes de reconnaissance fiables capable de détecter et d'identifier les objets dans les images avec une grande précision possible.

2.3 Ce qu'on veux résoudre

Nous nous intéressons a la détection d'objets dans une image c'est a dire que pour une image on doit pouvoir identifier les différents types d'objets contenu dans cette images en indiquant le nom correspondant(voir figure ci-dessus).

2.4 Situation actuelle

Les modèles de reconnaissance d'objets se sont améliorés à pas de géant au cours de la dernière décennie, mais ils ont encore un long chemin à parcourir en matière de précision. C'est la conclusion d'une équipe conjointe du Massachusetts Institute of Technology et d'IBM, qui a récemment publié un ensemble de données - ObjectNet - conçu pour illustrer l'écart de performances entre les algorithmes d'apprentissage automatique et les humains.

Contrairement à de nombreux ensembles de données existants, qui contiennent des photos prises sur Flickr et d'autres sites de médias sociaux, les échantillons de données d'ObjectNet ont été capturés par des pigistes rémunérés. Des objets représentés comme des oranges, des bananes et des vêtements sont renversés sur le côté, tournés à des angles étranges et affichés dans des pièces encombrées - des scénarios avec lesquels même les algorithmes de pointe ont du mal à lutter. En fait, lorsque des modèles de détection d'objets de pointe ont été testés sur ObjectNet, leur taux de précision est passé d'un sommet de 97% sur le corpus ImageNet accessible au public à seulement 50% à 55%.

Il s'appuie sur une étude publiée par les chercheurs de Facebook AI plus tôt cette année, qui a révélé que la vision par ordinateur pour reconnaître les objets ménagers fonctionne généralement mieux pour les personnes dans les ménages à revenu élevé. Les résultats ont montré que six systèmes populaires fonctionnaient mieux entre 10% et 20% pour les ménages les plus riches que pour les ménages les plus pauvres, et qu'ils étaient plus susceptibles d'identifier les articles dans les maisons en Amérique du Nord et en Europe qu'en Asie et en Afrique.

3 État de l'art

3.1 Bilan des solutions existants

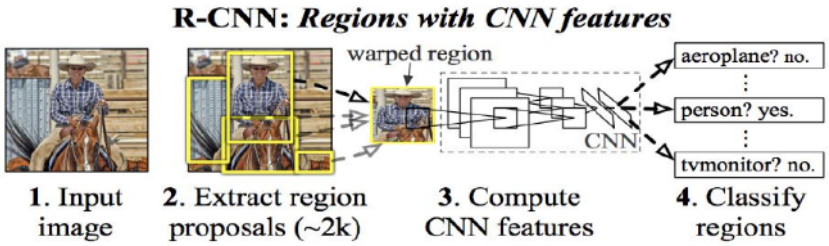
ResNet
Pour former le modèle de réseau de manière plus efficace, nous adoptons ici la

même stratégie que celle utilisée pour les DSSD (les performances du réseau résiduel sont meilleures que celles du réseau VGG). L'objectif est d'améliorer la précision. Cependant, le premier implémenté pour la modification a été le remplacement du réseau VGG qui est utilisé dans le SSD d'origine avec ResNet. Nous ajouterons également une série de couches d'entités convolutives à la fin du réseau sous-jacent. Ces couches d'entités seront progressivement réduites, ce qui a permis de prédire les résultats de détection à plusieurs échelles. Lorsque la taille d'entrée est donnée comme 300 et 320, bien que la couche ResNet – 101 soit plus profonde que la couche VGG – 16, il est connu expérimentalement qu'elle remplace le réseau de convolution sous-jacent du SSD par un réseau résiduel, et cela n'améliore pas sa précision mais le diminue plutôt

R-CNN

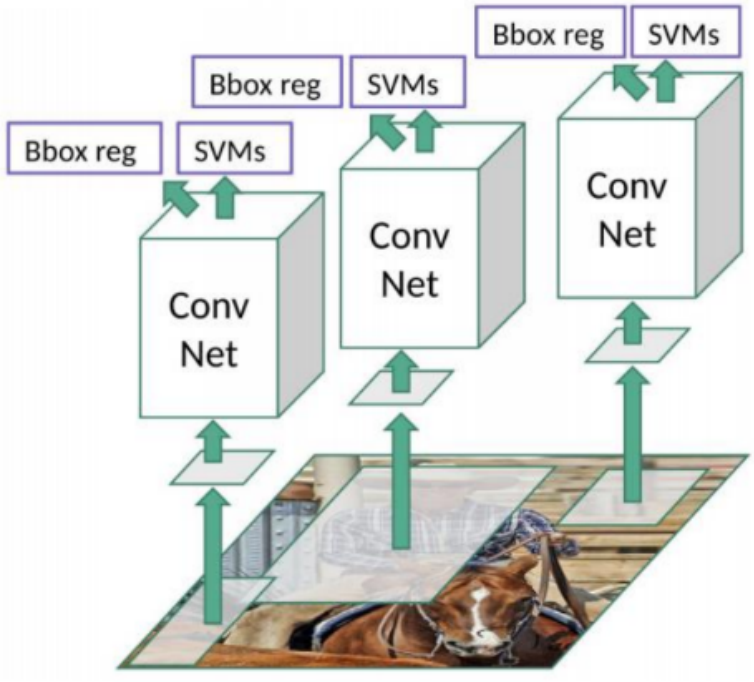
Pour contourner le problème de la sélection d'un grand nombre de régions, Ross Girshick et al. a proposé une méthode où nous utilisons la recherche sélective pour extraire seulement 2000 régions de l'image et il les a appelées propositions de régions. Par conséquent, au lieu d'essayer de classer le grand nombre de régions, vous pouvez simplement travailler avec 2000 régions. Ces propositions de 2000 régions sont générées en utilisant l'algorithme de recherche sélectif qui est écrit ci-dessous.

1. Générer la sous-segmentation initiale, nous générons de nombreuses régions candidates
2. Utilisez l'algorithme gourmand pour combiner récursivement des régions similaires en régions plus grandes
3. Utiliser les régions générées pour produire les propositions finales de régions candidates



Ces 2000 régions candidates qui sont des propositions sont déformées en un carré et introduites dans un réseau neuronal évolutif qui produit en sortie un vecteur caractéristique à 4096 dimensions. Le CNN joue un rôle d'extracteur d'entités et la couche dense de sortie se compose des entités extraites de l'image et les entités extraites sont introduites dans un SVM pour classer la présence de l'objet avec cette proposition de région candidate. En plus de prédire la présence d'un objet dans les propositions de région, l'algorithme prédit également quatre valeurs qui sont des valeurs de décalage pour augmenter la précision du cadre

de sélection. Par exemple, compte tenu de la proposition de région, l'algorithme aurait pu prédire la présence d'une personne, mais le visage de cette personne dans cette proposition de région aurait pu être réduit de moitié. proposition de la région.



R-CNN

Problèmes avec R-CNN

- La formation du réseau prend encore énormément de temps, car il faudrait classer 2000 propositions de région par image.

- Il ne peut pas être implémenté en temps réel car il faut environ 47 secondes pour chaque image de test.

- L'algorithme de recherche sélective est un algorithme fixe. Donc, aucun apprentissage ne se produit à ce stade. Cela pourrait conduire à la génération de mauvaises propositions de régions candidates.

Fast R-CNN

Le même auteur de l'article précédent (R-CNN) a résolu certains des inconvénients de R-CNN pour construire un algorithme de détection d'objet plus rapide et il a été appelé Fast R-CNN. L'approche est similaire à l'algorithme R-CNN. Mais, au lieu de transmettre les propositions de région au CNN, nous alimentons l'image d'entrée au CNN pour générer une carte des caractéristiques convolutives. À partir de la carte des caractéristiques convolutives, nous pouvons identifier la région des propositions et les déformer dans les carrés et en utilisant une couche de mise en commun RoI, nous les remodelons dans la taille fixe afin qu'elle puisse être introduite dans une couche entièrement connectée. À partir du vecteur d'entité RoI, nous pouvons utiliser une couche softmax pour prédire la classe de la région proposée ainsi que les valeurs de décalage pour le cadre de sélection.
La raison pour laquelle Fast R-CNN est plus rapide que R-CNN est parce que vous n'avez pas à alimenter à chaque fois 2000 propositions de région vers le réseau de neurones convolutionnels. Au lieu de cela, l'opération de convolution n'est toujours effectuée qu'une seule fois par image et une carte d'entités est générée à partir de celle-ci.

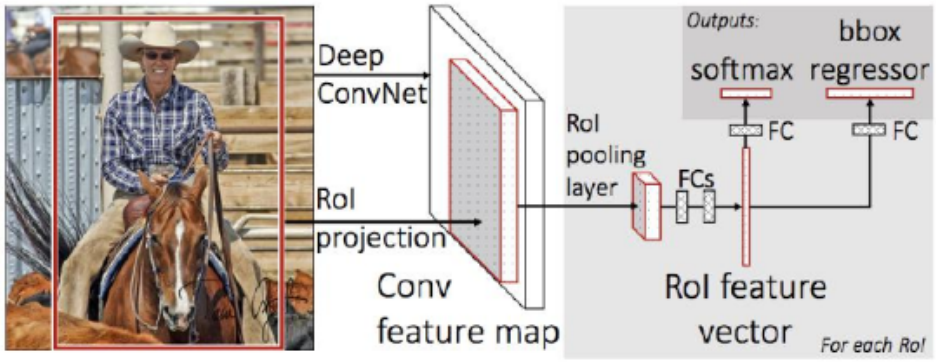


Figure 1: Fast R-CNN

4 Solution proposée

Nous présentons une méthode nommée **SSD(Single shot Detector)** qui est l'un des algorithmes les plus rapides dans le domaine actuel de détection d'objets, qui utilise un réseau neuronal entièrement convolutionnel pour détecter tous les objets à l'échelle dans une image. L'architecture du SSD se compose de 3 composants principaux:

- **Base network(Reseau de base)**
  Le réseau de base est essentiellement les couches initiales de tout réseau de classification d'images standard, pré-formé sur l'ensemble de données ImageNet.Le reseau de neurone convolutionnel VGG16 est utilise ici.Les couches entièrement connectées à la fin sont implémentées en tant que couches convolutives. La sortie finale du réseau de base est une carte de caractéristiques de taille 19 x 19 x 1024.
  Au sommet du réseau de base, 4 couches convolutionnelles supplémentaires sont ajoutées de sorte que la taille des cartes d'entités continue de diminuer jusqu'à ce qu'une carte d'entités finale de taille 1 x 1 x 256 soit obtenue.
- **Extra feature layers (Couches de fonctionnalités supplémentaires)**
- **Prediction layers(Couches de prédiction)**
  Les couches de prédiction sont un composant crucial du SSD. Au lieu d'utiliser simplement une carte d'entités pour prédire les scores de classification et les coordonnées du cadre de délimitation, plusieurs cartes d'entités, représentant plusieurs échelles, sont utilisées.

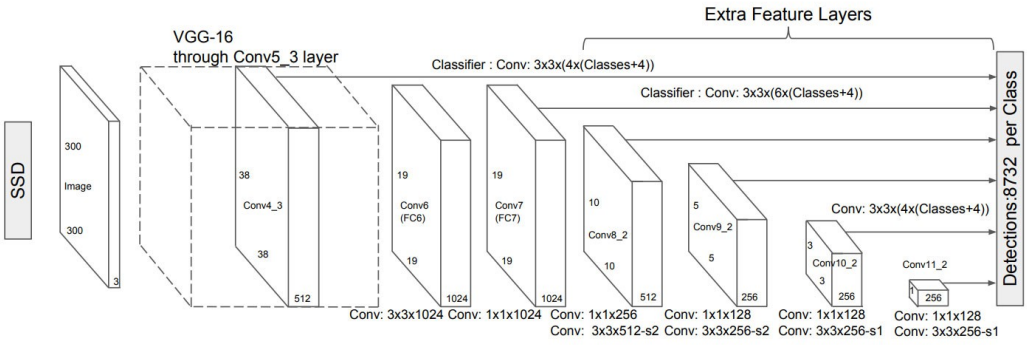


Figure 2: Architecture SSD

4.1 Données

Nous utiliserons les données Pascal Visual Object Classes (VOC) des années 2007 et 2012.
Ces données contiennent des images avec vingt types d'objets différents.

aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor.

Chaque image peut contenir un ou plusieurs objets de bounding box.
Chaque objet est représenté par

- Un cadre de délimitation en coordonnées limites absolues

Figure 3: Différents objets (images)

- Un label (l'un des types d'objets mentionnés ci-dessus)
- une difficulté de détection perçue (soit 0, ce qui signifie pas difficile , ou 1, ce qui signifie difficile)

Pour avoir le dataset il sera necessaire de cliquer sur:

2007 trainval (460MB)

2012 trainval (2GB)

2007 test (451MB)

Conformément a notre programme, les deux ensembles de données de trainval doivent être utilisés pour la formation, tandis que le test COV 2007 servira de données de validation et de test.
Et s'assurer d'extraire les données de test VOC 2007 trainval et 2007 au même endroit, c'est-à-dire de les fusionner.

Entrées à modéliser
Nous aurons besoin de trois entrées.

1. **Images**
   Puisque nous utilisons la variante SSD300, les images devraient être dimensionnées en 300, 300pixels et au format RVB. N'oubliez pas que nous utilisons une base VGG-16 pré-entraînée sur ImageNet qui est déjà disponible dans le module torchvision PyTorch.

2. **Boîtes englobantes d'objets**
   Nous aurions besoin de fournir, pour chaque image, les boîtes englobantes des objets de bounding box qui y sont présents en coordonnées de frontière fractionnaires (x_min, y_min, x_max, y_max).

Étant donné que le nombre d'objets dans une image donnée peut varier, nous ne pouvons pas utiliser un tenseur de taille fixe pour stocker les boîtes englobantes pour l'ensemble du lot d' Nimages.

Par conséquent, les boîtes de délimitation de la bounding box alimentant le modèle doivent être une liste de longueur N, où chaque élément de la liste est un Floattenseur de dimensions N_o, 4 , où N_o est le nombre d'objets présents dans cette image particulière.

3. **Labels d'objets**
   Nous aurions besoin de fournir, pour chaque image, les labels des objets de bounding box qui y sont présents.

   Chaque label devrait être codée sous la forme d'un entier de 1 à 20 représentant les vingt types d'objet différents. De plus, nous ajouterons une classe d' arrière-plan avec index 0, qui indique l'absence d'un objet dans une boîte englobante. (Mais naturellement, cette label ne sera en fait utilisée pour aucun des objets de bounding box dans l'ensemble de données.)

   Encore une fois, étant donné que le nombre d'objets dans une image donnée peut varier, nous ne pouvons pas utiliser un tenseur de taille fixe pour stocker les labels pour l'ensemble du lot de N images.

   Par conséquent, les labels de bounding box fournies au modèle doivent être une liste de longueur N, où chaque élément de la liste est un Long tenseur de dimensions N_o , où N_o est le nombre d'objets présents dans cette image particulière.

4.2 Outils

Les technologies et outils utlises que nous avons eu a utilise lors de ce travail sont:

- Détecteur Single Shot (SSD)
  Le SSD est un réseau neuronal purement convolutionnel (CNN) que nous pouvons organiser en trois parties -
  - **Convolutions de base** dérivées d'une architecture de classification d'images existante qui fournira des cartes d'entités de niveau inférieur.
  - Ajout de **convolutions auxiliaires**au-dessus du réseau de base qui fourniront des cartes d'entités de niveau supérieur.
  - **Des convolutions de prédiction** qui localiseront et identifieront les objets dans ces cartes de caractéristiques.

  Le document présente deux variantes du modèle appelé SSD300 et SSD512. Les suffixes représentent la taille de l'image d'entrée. Bien que les deux réseaux diffèrent légèrement dans la façon dont ils sont construits, ils sont en principe les mêmes. Le SSD512 est juste un réseau plus grand et offre des performances légèrement meilleures. Pour plus de commodité, nous traiterons du SSD300.
- Pytorch (Librairie de deep learning)
- Jupyter(Environnement de trvail)
- Python (langage de programmation)

4.3 Modèles conceptuels / Algorithmes / Pipeline de traitement

Comme nous le savons, nos données sont divisées en segments de formation et de test.
Analyser les données brutes
Voir create_data_lists() dans utils.ipynb.

Cela analyse les données téléchargées et enregistre les fichiers suivants

- Un fichier JSON pour chaque division avec une liste des chemins de fichiers absolus des I images , où I est le nombre total d'images dans la division.
- Un fichier JSON pour chaque groupe avec une liste de I dictionnaires contenant des objets de vérité au sol, soit des boîtes de délimitation en coordonnées aux limites absolues, leurs étiquettes codées, et des difficultés de détection perçus . Le i th dictionnaire de cette liste contiendra les objets présents dans la i th image du fichier JSON précédent.

- Un fichier JSON qui contient lelabel_map , le dictionnaire d'étiquette à index avec lequel les étiquettes sont encodées dans le fichier JSON précédent. Ce dictionnaire est également disponible utils.ipynb et directement importable.

PyTorch Dataset

Voir PascalVOCDataset dans datasets.ipynb. Il s'agit d'une sous-classe de PyTorch Dataset, utilisée pour définir nos ensembles de données de formation et de test. Il a besoin d'une __len__ méthode définie, qui renvoie la taille de l'ensemble de données, et d'une __getitem__ méthode qui renvoie la ie image, les cadres de délimitation des objets dans cette image et les étiquettes des objets dans cette image, en utilisant les fichiers JSON que nous avons enregistrés précédemment.

Vous remarquerez qu'il renvoie également les difficultés de détection perçues de chacun de ces objets, mais ceux-ci ne sont pas réellement utilisés dans la formation du modèle. Ils ne sont nécessaires qu'à l' étape d' évaluation pour calculer la métrique de précision moyenne moyenne (mAP). Nous avons également la possibilité de filtrer entièrement les objets difficiles à partir de nos données pour accélérer la formation au prix d'une certaine précision.

De plus, à l'intérieur de cette classe, chaque image et les objets qu'elle contient sont soumis à une multitude de transformations, comme décrit dans l'article et décrit ci-dessous.

Transformations de données

Voir transform()dans utils.ipynb.

Cette fonction applique les transformations suivantes aux images et aux objets qu'elles contiennent,

- **Ajustez au** hasard la **luminosité, le contraste, la saturation et la teinte,** chacun avec une chance de 50% et dans un ordre aléatoire.
- Avec 50% de chances, **effectuez une opération de zoom arrière** sur l'image. Cela aide à apprendre à détecter les petits objets. L'image agrandie doit être entre 1 et 4 fois plus grande que l'original. L'espace environnant pourrait être rempli avec la moyenne des données ImageNet.
- Recadrer l'image au hasard, c'est-à-dire **effectuer un zoom avant**. Cela aide à apprendre à détecter des objets volumineux ou partiels. Certains objets peuvent même être entièrement découpés. Les dimensions du recadrage doivent être comprises entre 0.3 et 1 fois les dimensions d'origine. Le rapport hauteur / largeur doit être compris entre 0.5 et 2. Chaque culture est réalisée de telle sorte qu'il y ait au moins une zone de délimitation restant qui a un chevauchement de l' une Jaccard 0, 0.1, 0.3, 0.5, 0.7, ou 0.9, choisi au hasard, avec l'image rognée. En outre, toutes les zones de délimitation restantes dont les centres ne sont plus dans l'image à la suite du recadrage sont supprimées. Il est également possible que l'image ne soit pas recadrée du tout.

- Avec 50% de chance, **retournez horizontalement** l'image.
- **Redimensionnez** l'image en 300, 300 pixels. C'est une exigence du SSD300.
- Convertissez toutes les cases des **coordonnées de limite absolues en fractionnaires**. À toutes les étapes de notre modèle, toutes les coordonnées de limite et de taille centrale seront dans leurs formes fractionnaires.
- **Normaliser** l'image avec la moyenne et l'écart type des données ImageNet qui ont été utilisées pour pré-former notre base VGG.

PyTorch DataLoader

Le Dataset décrit ci - dessus, PascalVOCDataset sera utilisé par un PyTorch DataLoader en train.ipynb pour **créer et lots d'alimentation de données au modèle** de formation ou de validation.

Étant donné que le nombre d'objets varie selon les différentes images, leurs boîtes englobantes, leurs étiquettes et leurs difficultés ne peuvent pas simplement être empilées ensemble dans le lot. Il n'y aurait aucun moyen de savoir quels objets appartiennent à quelle image.

Au lieu de cela, nous devons **passer une fonction de classement à l'argument collate_fn** , qui indique DataLoader comment il doit combiner ces tenseurs de taille variable. L'option la plus simple serait d'utiliser des listes Python.

5 Implémentation et expérimentation

5.1 Implémentation

Convolutions de base

Voir VGGBasedans model.ipynb.

Ici, nous **créons et appliquons des convolutions de base.**

Les couches sont initialisées avec les paramètres d'un VGG-16 pré-entraîné avec la load_pretrained_layers()méthode.

Nous sommes particulièrement intéressés par les cartes d'entités de niveau inférieur qui résultent de conv4_3 et conv7, que nous renvoyons pour une utilisation dans les étapes suivantes.

Convolutions auxiliaires
Voir PredictionConvolutions dans model.ipynb.

Ici, nous **créons et appliquons la localisation et de prévision de la classe convolution**s à la fonction de cartes conv4_3, conv7, conv8_2, conv9_2, conv10_2et conv11_2.

Ces couches sont initialisées d'une manière similaire aux convolutions auxiliaires.

Nous **remodelons** également **les cartes de prédiction résultantes et les empilons** comme indiqué. Notez que le remodelage dans PyTorch n'est possible que si le tenseur d'origine est stocké dans un bloc de mémoire contigu .

Comme prévu, la localisation empilée et les prédictions de classe seront de dimensions 8732, 4et 8732, 21 respectivement.

Mettre tous ensemble

Voir SSD300 dans model.ipynb.

Ici, les **convolutions de base, auxiliaires et de prédiction sont combinées** pour former le SSD.
Il y a un petit détail ici - les caractéristiques de niveau le plus bas, c'est-à-dire celles de conv4_3, devraient être sur une échelle numérique significativement différente par rapport à ses homologues de niveau supérieur. Par conséquent, les auteurs recommandent de normaliser L2, puis de redimensionner chacun de ses canaux par une valeur apprenable.

Training
Avant de commencer, il faudrais s'assurer de sauvegarder les fichiers de données requis pour la formation et la validation. Pour ce faire, il faudrai exécutez le contenu de create_data_lists.ipynb après l'avoir pointé vers les dossiers VOC2007et VOC2012 dans les données téléchargées.
Pour former le modèle à partir de zéro , exécutez le fichier python train.ipynb
Pour reprendre l'entraînement à un point de contrôle ,on pointe sur le fichier correspondant avec le checkpoint paramètre au début du code.

Notons que nous effectuons la validation à la fin de chaque période de formation.

Remarques
Dans un article de reference, ils recommandent d'utiliser la descente de gradient stochastique dans des lots d' 32images, avec un taux d'apprentissage initial de 1e3, un élan 0.9et 5e-4une décroissance de poids. Nous avns fini par utiliser une taille d'image 8 par lots pour une stabilité accrue. Si l'on constate que les dégradés explosent, on peut réduire la taille du lot, comme nous l'avons fait, ou écraser les dégradés.

Le document recommande une formation pour 80000 itérations au rythme d'apprentissage initial. Ensuite, il se désintègre de 90% pour 20000 itérations supplémentaires, deux fois . Avec la taille de lot du papier de 32, cela signifie que le taux d'apprentissage est diminué de 90% une fois à la 155e époque et une fois de plus à la 194e époque, et la formation est arrêtée à 232 époques.

En pratique, nous avons diminuer le taux d'apprentissage de 90% lorsque la

perte de validation a cessé de s'améliorer pendant de longues périodes. Nous avons repris l'entraînement à ce rythme d'apprentissage réduit à partir du meilleur point de contrôle obtenu jusqu'à présent, pas du plus récent.

Sur un TitanX (Pascal), chaque époque d'entraînement a nécessité environ 6 minutes. Notre meilleur point de contrôle était de l'époque 186, avec une perte de validation de 2.515.

5.2 Expérimentation: cas de test

Pour evaluer notre model il nous faut executer le fichier eval.ipynb

Les paramètres de chargement des données et de point de contrôle pour évaluer le modèle sont au début du fichier, on peut donc facilement les vérifier ou les modifier si l'on le souhaite. Pour commencer l'évaluation, exécutez simplement la evaluate() fonction avec le chargeur de données et le point de contrôle du modèle. Les prédictions brutes pour chaque image de l'ensemble de test sont obtenues et analysées avec la detect_objects() méthode du point de contrôle , qui met en œuvre ce processus . L' évaluation doit être fait à min_scoredes 0.01, un NMS max_overlapde 0.45 et top_k de 200 pour permettre de comparaison équitable des résultats avec le papier et d' autres mises en œuvre. Les prédictions analysées sont évaluées par rapport aux objets de bounding box. La métrique d'évaluation est la précision moyenne (mAP). Nous utiliserons calculate_mAP() à utils.ipynb cet effet. Comme c'est la norme, nous ignorerons les détections difficiles dans le calcul du mAP. Mais néanmoins, il est important de les inclure dans l'ensemble de données d'évaluation car si le modèle détecte un objet considéré comme difficile , il ne doit pas être considéré comme un faux positif. Les précisions moyennes par classe sont répertoriées ci-dessous.

Classe	Précision moyenne
avion	78,9
vélo	83,7
oiseau	76,9
bateau	72,0
bouteille	46,0
autobus	86,7
voiture	86,9
chat	89,2
chaise	59,6
vache	82,7
table à manger	75,2
chien	85,6
cheval	87,4
moto	82,9
la personne	78,8
plante en pot	50,3
mouton	78,7
canapé	80,5
train	85,7
tvmonitor	75,0

Nous pouvons voir que certains objets, comme les bouteilles et les plantes en pot, sont beaucoup plus difficiles à détecter que d'autres.

5.3 Analyse de résultats

Sur le premier résultat nous remarquons que notre programme a réussi a détecter tous les types d'objets avec précision se trouvant sur l'image.

Sur cette image nous remarquons que le programme réussi a détecter certains objets sur l'image mais pas toutes tel que les chaises ne sont pas détecter et nous pouvons voir que les cadres ne sont pas précis a cause de l'emplacement très rapprocher des objets.

Dans cette image nous remarquons que le programme détecte deux personnes et et une motorbike alors que nous pouvons remarqué qu'il n'y a qu'une personne ceci est a cause de la forme de la statuette située a droite de l'image car les caractéristiques de ce dernier correspond a celui d'une personne c'est a dire la forme de la tête et du corps.


POTTEDPLANT
POTTEDPLANT
CHAIR
SOFA
DININGTABLE

6 Conclusion

Un système de détection d'objets précis et efficace a été développé qui permet d'obtenir des métriques comparables avec le système de pointe existant. Ce projet utilise des techniques récentes dans le domaine de la vision par ordinateur et du deep learning. Un ensemble de données personnalisé a été créé à l'aide de labelImg et l'évaluation était cohérente. Cela peut être utilisé dans des applications en temps réel qui nécessitent la détection d'objets pour le prétraitement dans leur pipeline. Une portée importante serait de former le système sur une séquence vidéo à utiliser dans les applications de suivi. L'ajout d'un réseau temporellement cohérent permettrait une détection fluide et plus optimale que la détection par trame.

References

- https://pytorch.org/tutorials/beginner/deep$_{l}earning_{6}0min_{b}litz.html$

- https://pytorch.org/tutorials/beginner/pytorch$_{w}ith_{e}xamples.html$

-https://venturebeat.com/2020/02/03/apple-details-ai-to-help-voice-assistants-recognize-hotwords-and-multilingual-speakers/

-https://www.researchgate.net/publication/337464355_OBJECT_DETECTION_AND_IDENTIFICATION