



INSTITUT
FRANCOPHONE
INTERNATIONAL

Travaux Personnels Encadrés

Promotion 24 – S.I.M.

RAPPORT FINAL

Detection et Reconnaissance des pièces détaché de voiture avec Deep Learning.

ACHILLE Peterson

Encadrant : Dr. HO Tuong Vinh

Date de Remise : Septembre 2020



Since 1906
VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

Table des matières

Partie Analyse du Sujet

Introduction	5
2.- Le contexte et problématique du sujet.....	5
3.- Les problèmes à résoudre	5
4.- Les objectifs à réaliser	6
5.- Résultat attendu	6
6.- Difficultés à prévoir.....	7

Partie Bibliographie du Sujet

Introduction	8
1. - État de l'Art	8
3.- La reconnaissance d'objets	8
4.- Reconnaissance d'objets vs. Détection d'objets	9
5.- Solutions Existantes.....	9
6.- Méthodes de classifications	9
7.- Les différentes méthodes de détection	10
8.- Réseau de neurones	10
8.1.- CNN : Réseau de Neurones Convolutifs	11
8.2.- R-CNN (Regional Convolutional Neural Network)	11
8.3. - Fast R-CNN	13
8.4.- Mask R-CNN.....	13
9.- Difficultés.....	14
10.- Défis de la détection	15
Conclusion.....	15

Partie Solution Proposée

Introduction	16
2.- Notre Objectives	16
3.- Description de la solution	17
4.- Nos stratégies de fonctionnement	18
5.- Les méthodes de détection de pièces.....	19
6.- Présentation des données	19
7.- Scénario de Test et de Train.....	20
8.- L'entrainement ou l'apprentissage	21
9.- La classification.....	21
10.- Le déploiement	22
11.- Plan de travail.....	23
12.- Outils à utilisés	24
Conclusion.....	25

Partie Conception et Implémentation

1.- Conception et Implémentation	26
1.1.- Préparation des données.....	26
1.2.- Implémentation sur Colab Pro	29
1.3.- Implémentation sur anaconda prompt	37

Partie Expérimentation et Résultat Obtenus

1.-Evaluation de notre système détection des pièces voiture	39
1.1.- Expérimentation avec des images de voiture.....	39
1.2.- Expérimentation avec une vidéo de voiture.....	50
1.3.- Expérimentation avec le Webcam.....	51
Conclusion et Perspectives.....	52
Références Scientifiques	53

Remerciements

Je tiens à exprimer mes remerciements à tous ceux qui ont rendu ce travail possible. Leurs aides précieuses, leurs conseils fructueux et leurs encouragements, tout au long de l'élaboration de ce projet, m'a permis de le réaliser dans la meilleure considération.

Je remercie en premier le professeur **Dr. HO Tuong Vinh** en tant qu'encadrant de mon travail, il m'a aidé, à guider et à proposer des solutions pour avancer.

Finalement, je voudrais remercier ma famille et mes amis de m'avoir soutenu durant ce projet ainsi que pour la relecture de ce rapport.

LIEN :

<https://github.com/IngH2020/Detection-et-reconnaissance-des-pieces-de-voiture-avec-Deep-Learning/upload/master>

Partie Analyse du Sujet

Introduction

Depuis plusieurs années, la classification d'image s'est beaucoup améliorée grâce à de meilleures utilisations du concept de Machine Learning. La création de modèles par apprentissage nous permet de rapidement détecter et classifier différents objets sans devoir écrire un algorithme précis.

Cependant sur le marché de l'automobile il existe une multitude de type de pièces détachées, toutefois en imagerie, ces performances en détection et classification sont obtenues pour des objets de grande taille dans des images RGB. De plus, l'apprentissage des paramètres des réseaux profonds nécessite d'avoir accès à des grandes bases de données annotées (plusieurs centaines de milliers ou plusieurs millions d'exemples).

2.- Le contexte et problématique du sujet

Une voiture classique est composée d'environ de 30.000 pièces détachées, à noter que ce chiffre peuvent être variées en fonction du modèle, du type, de l'année ou de la marque de la voiture. De ce fait, l'identification et l'emplacement de ces pièces peuvent être compliqués pour certains professionnels (mécaniciens, électroniciens etc.) dans le domaine surtout si la marque est récente. Ce qui par ailleurs, peut entraîner un mal interprétation aux pièces.

3.- Les problèmes à résoudre

Avec l'évolution de la technologie et les techniques et appliquées à l'automobile, les nouvelles voitures sont plus difficiles à identifier ces pièces. Le problème à résoudre consiste à implémenter un système intelligent qui, à partir d'un modèle entraîner, pouvant aider à identifier et reconnaître des pièces d'une voiture et le système va vous donner toutes les informations par rapport aux pièces détecter.

4.- Les objectifs à réaliser

Tout en ayant conscience de la situation à laquelle nous sommes exposés l'objectif principal c'est de fournir un système informatique doté d'une détection d'objet remarquable, fiable, facile à utiliser et performant capable identifier et reconnaître des pièces d'une voiture en temps réel.

❖ Théorie

- Théoriquement l'objectif Consiste à l'implémenter d'un système informatique capable de détecter et reconnaître des pièces d'une voiture à partir d'une image, une vidéo et webcam.
- Former un modèle d'identification des pièces voiture sur l'ordinateur à base de l'algorithme proposé
- Déployer le modèle former sur une plateforme mobile et permettre l'identification des pièces avec les téléphones portables

❖ Pratique

Comme objectif Pratique pour la réalisation de ce projet nous allons procéder aux:

- La configuration et mise en place de la structure Deep Learning
- La collection et l'étiquetage des images nécessaires.
- L'entraînement du modèle
- Expérimenter notre modèle avec image, vidéo et webcam
- Créer une Application mobile
- Différent test sur le fonctionnement du système.

5.- Résultat attendu

A la fin de ce travail personnel encadré (TPE), nous aurons à construire un prototype permettant d'identifier des pièces voiture, d'abord sur un ordinateur et ensuite le déployer dans une application mobile pour permettre l'identification des pièces avec les téléphones portables. C'est-à-dire, avec un smartphone, on peut prendre des photos d'une voiture et l'analyser par notre application qui doit être capable de détecter les pièces.

6.- Difficultés à prévoir

Aux regards de nos résultats attendus, nous pouvons prévoir quelques difficultés qui sont entre autres

- Maîtrise de l'apprentissage profond ;
- Maîtrise du langage de programmation python ;
- Obtention des différents articles sur détection avec des téléphones portables ;
- Maîtrise du développement mobile.

Partie Bibliographie du Sujet

Introduction

Cette seconde partie qu'est la Recherche Bibliographique, exigera de nous une connaissance précise et spécialisée du sujet, de comprendre les techniques existantes, de trouver les solutions possibles au sujet et enfin d'établir une comparaison des différentes approches du sujet.

1. - État de l'Art

La reconnaissance des objets et leur détection dans une image sont des éléments importants de la recherche courante en vision ordinateur. Cette discipline cherche à détecter dans une image ou vidéo les différents objets, leur classe ainsi que leur position dans l'image.

Une architecture souvent utilisée ces dernières années est **Régions avec Réseau Neuronal Convolutif (R-CNN)**. Ce type d'architecture de Machine Learning s'est montré comme très performant dans plusieurs domaines pour résoudre la détection d'objet

Dans cette revue de la littérature, nous ferons un tour des algorithmes et principes utilisés pour détecter les pièces de voiture dans un ensemble d'image données ainsi qu'une revue de différents modèles de réseaux de neurones qui pourraient améliorer ce domaine.

3.- La reconnaissance d'objets

La reconnaissance d'objets est une technique de computer vision utilisée pour l'identification d'objets présents dans des images et des vidéos. La reconnaissance d'objets est le produit d'algorithmes de Deep Learning et de Machine Learning. Lorsqu'un être humain observe des photos ou regarde une vidéo, il est en mesure de percevoir immédiatement les personnes, les objets, les scènes et les détails visuels qu'il a sous les yeux. Le but est d'apprendre à un ordinateur à réaliser ce dont les humains sont naturellement capables, et à acquérir un niveau de compréhension approprié de ce que contient l'image.

4.- Reconnaissance d'objets vs. Détection d'objets

La détection d'objets et la reconnaissance d'objets sont des techniques similaires pour identifier des objets, mais qui diffèrent dans leur mise en œuvre. La détection d'objets est le processus qui s'emploie à rechercher des instances d'objets dans les images. Dans le cas du Deep Learning, la détection d'objets est un sous-ensemble de la reconnaissance d'objets, où l'objet est non seulement identifié mais également situé dans l'image. Cela permet l'identification et la localisation d'objets multiples dans une même image.

5.- Solutions Existantes

Il existe différents types de solutions d'apprentissage automatique pour la classification d'images. Mais le meilleur et le plus précis est CNN - Convolution Neural Network. Pour comprendre comment cela fonctionne, parlons de la convolution elle-même. C'est un processus au cours duquel deux fonctions intègrent la production d'un nouveau produit. En ce qui concerne les images, nous devons penser à une image comme une matrice de pixels. Chaque pixel a sa propre valeur mais est intégré à d'autres pixels et génère un résultat - une image.

- TensorFlow peut vous aider à créer des modèles de réseau neuronal pour reconnaître automatiquement les images. Il s'agit généralement de réseaux neuronaux convolutionnels (CNN). Il existe deux approches pour la reconnaissance d'image TensorFlow:
 - **Classification** - entraînez le CNN à reconnaître des catégories comme les chats, les chiens, les voitures ou toute autre chose. Le système classe l'image dans son ensemble, en fonction de ces catégories.
 - **Détection d'objets**: plus puissante que la classification, elle peut détecter plusieurs objets dans la même image. Il marque également les objets et montre leur emplacement dans l'image.

6.- Méthodes de classifications

Les méthodes de classification les plus communes peuvent être séparées en deux grandes catégories : les méthodes de classification supervisée et les méthodes de classification non supervisée.

- Méthode supervisé : Les classes sont définies à partir des besoins de l'utilisateur et correspondent à des unités sémantiques de l'image, ce qui nécessite une étape d'apprentissage préalable à la classification .Donc la classification supervisé est un processus inductif où l'utilisateur considère un ensemble d'exemples étiquetés préalablement : la cible à apprendre est connue (classe d'appartenance)

- Méthode non supervisé : La classification non supervisée désigne un corpus de méthodes ayant pour objectif de dresser ou de retrouver une typologie existante caractérisant un ensemble de n observations, à partir de p caractéristiques mesurées sur chacune des observations. Dans le cas non supervisé, l'opérateur n'a besoin d'aucune information a priori sur la scène à classifier. Soit il existe une collection universelle de classes déjà définies indépendamment de l'image, soit le nombre de classes et leurs caractéristiques sont définies automatiquement durant le processus de classification.

7.- Les différentes méthodes de détection

Les algorithmes de détection objet dans une image fixe peuvent être rangés selon les trois catégories suivantes :

- **Modélisation colorimétrique :**

Une segmentation en composantes connexes fondée sur un modèle de couleur est opérée. Les régions d'intérêt sont ensuite validées par un algorithme de reconnaissance ou un modèle d'apparence. Ces méthodes sont les plus rapides mais aussi les moins robustes aux variations des conditions d'éclairage.

- **Modélisation géométrique :**

Les contours de l'image sont analysés par une approche structurelle ou globale. Ces méthodes sont généralement plus robustes que celles photométriques parce qu'elles traitent le gradient de l'image, et peuvent traiter des images en niveaux de gris

- **Méthodes avec apprentissage :**

Un classifieur (cascade, SVM, réseaux de neurones) est entraîné sur une base d'exemples. Il est appliqué sur une fenêtre glissante qui parcourt l'image à plusieurs échelles. Ces méthodes combinent géométrie et photométrie mais peuvent être une étape coûteuse en temps de calcul.

8.- Réseau de neurones

Dans cette section, nous parlerons de plusieurs concepts reliés aux réseaux de neurones artificiels importants à la compréhension de ce rapport. "Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur

élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau."

Cette définition explique que chaque neurone est une fonction basique. Un neurone peut par exemple recevoir un nombre, le multiplier par deux puis l'envoyer aux neurones suivants.

8.1.- CNN : Réseau de Neurones Convolutifs

Revenons aux concepts de base. Les réseaux de neurones sont des algorithmes d'intelligence artificielle inspirés du fonctionnement des réseaux de neurones du cerveau humain. Parmi ces algorithmes, les réseaux de neurones convolutifs (CNN) permettent l'analyse des images. Ils constituent à ce jour l'une des tentatives les plus abouties de doter une IA de capacités s'approchant de la vision humaine.

Ils sont constitués de plusieurs couches, elles-mêmes constituées de plusieurs neurones. Chaque couche reçoit les informations de la couche précédente, traite ces informations et les renvoie à la couche suivante. Chaque neurone d'une couche est donc relié aux neurones de la couche suivante. Ces liens sont pondérés par des poids et, à l'image du signal électrique émis par les neurones biologiques, vont influencer la propagation de l'information à travers le réseau de neurones convolutifs.

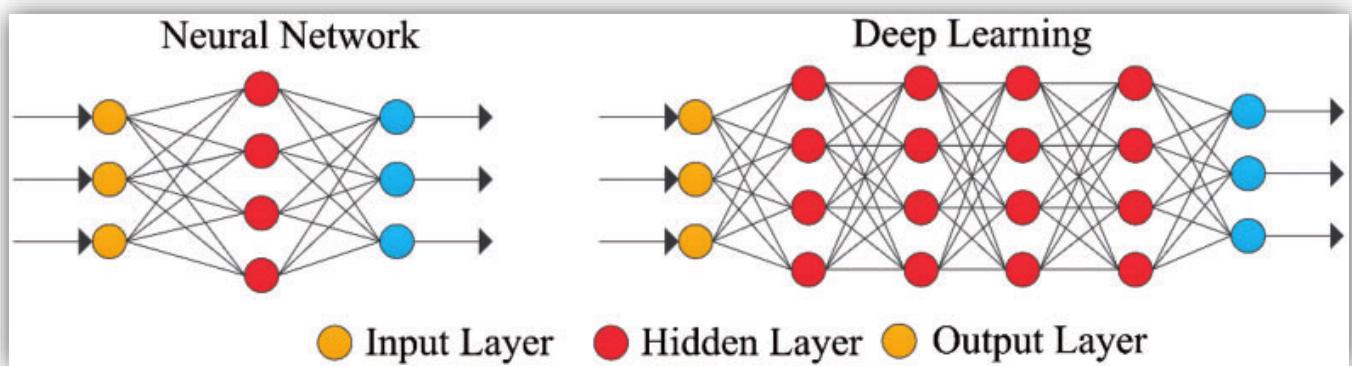


Figure: Schéma CNN

8.2.- R-CNN (Regional Convolutional Neural Network)

Les modèles R-CNN sont une famille de réseaux de neurones convolutifs conçus pour la détection d'objets, développés par Ross Girshick.

A partir d'une image donnée en entrée, le modèle va extraire de l'image les régions les plus susceptibles de contenir un objet. On parle de **zones d'intérêts**. Pour chacune de ces zones d'intérêts, un ensemble de

boîtes englobantes (*bounding box*) va être générée. Ces boîtes sont classifiées et sélectionnées en fonction de leur probabilité à contenir l'objet. 2000 propositions de régions sont ainsi extraites. Ces régions sont ensuite reçues en entrée par le CNN. Le modèle peut alors détecter dans une image un (ou plusieurs) objet(s), pouvant appartenir à des classes différentes.

Les avantages sont donc de traiter l'image par morceau et non pas toute l'image comme pour un CNN simple et de pouvoir localiser plusieurs objets dans une image. C'est un traitement plus rapide et moins coûteux en puissance-machine.

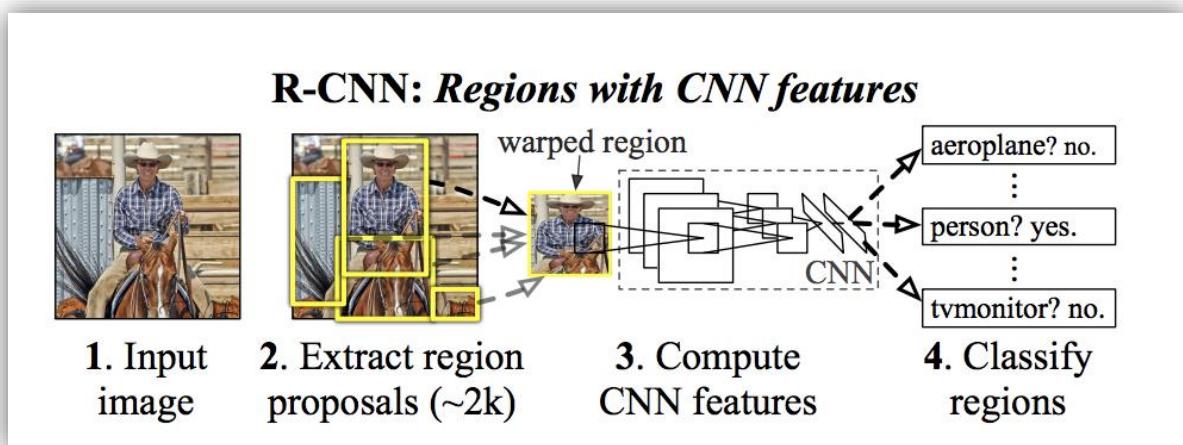


Figure: Schéma R-CNN

Les CNN - réseaux de neurones à convolution sont axés principalement sur des applications comportant des motifs répétitifs dans différents domaines de l'espace de modélisation, notamment dans la reconnaissance d'image et vidéo.

- ❖ **La couche de convolution** fait passer les images d'entrée par un ensemble de filtres consolutifs, chacun de ces filtres activant certaines caractéristiques des images.
- ❖ **La couche ReLU** (Rectified linear unit) permet d'accélérer et d'optimiser l'apprentissage en mappant les valeurs négatives à zéro et en conservant les valeurs positives. Elle est parfois appelée couche d'activation, car seules les caractéristiques activées sont transférées dans la couche suivante.
- ❖ **La couche de pooling** simplifie la sortie en effectuant un sous-échantillonnage non linéaire, réduisant ainsi le nombre de paramètres que le réseau doit apprendre.

Ces opérations sont répétées sur des dizaines ou des centaines de couches, et chacune de ces couches apprend à identifier des caractéristiques différentes.

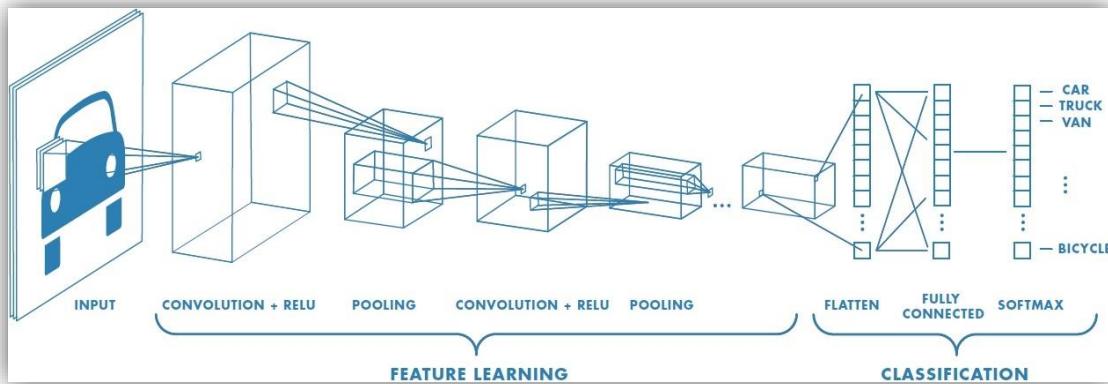


Figure: Schema R-CNN

8.3. - Fast R-CNN

Avec un modèle R-CNN, chacune des zones d'intérêts proposées est reçue en entrée par le CNN, ainsi l'image de départ subit une convolution par zone proposée. En utilisant un modèle Fast R-CNN c'est l'image de départ qui subit cette étape de convolution et c'est sur les features map générées que sont obtenues les propositions de régions susceptibles de contenir l'élément ciblé par le modèle. Cela constitue une amélioration en termes de temps d'exécution et de puissance requise.

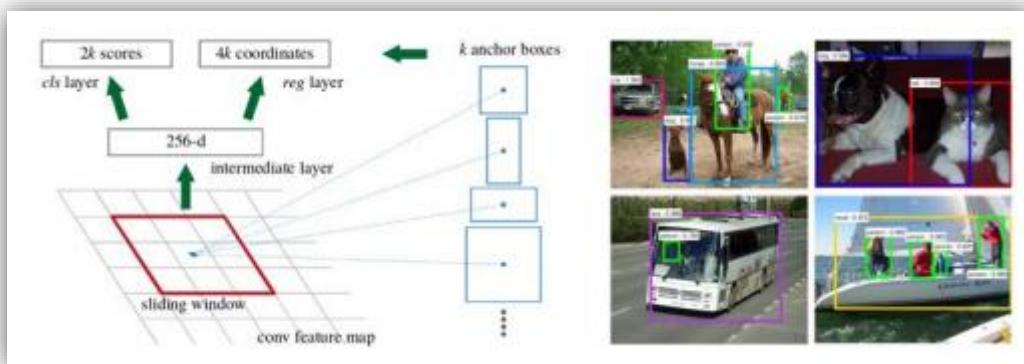


Figure: Fast R-CNN

8.4.- Mask R-CNN

Le modèle Mask R-CNN permet de **déetecter** des objets et de les **classifier**. Sa particularité est d'ajouter à cette tâche de détection la **segmentation d'instance**, c'est à dire que chaque pixel de l'image sera classé.

Ainsi, ce double “compétence” représente un avantage par rapport à des modèles de détection, elle vient affiner le résultat proposé. De plus, contrairement à la segmentation sémantique, qui permet d'associer à chaque pixel un label, la segmentation d'instance associe un masque et un label à chaque objet, même si ces objets appartiennent à la même classe. Dans le cadre de notre étude où nous cherchons à extraire des pièces à partir d'images.

Mask R-CNN est une extension du modèle Faster R-CNN. Aux deux types de sorties générées par ce dernier, qui sont la classe de l'objet présent sur l'image et la boîte englobante associée, s'ajoute une troisième branche dont la sortie est le masque de l'objet.

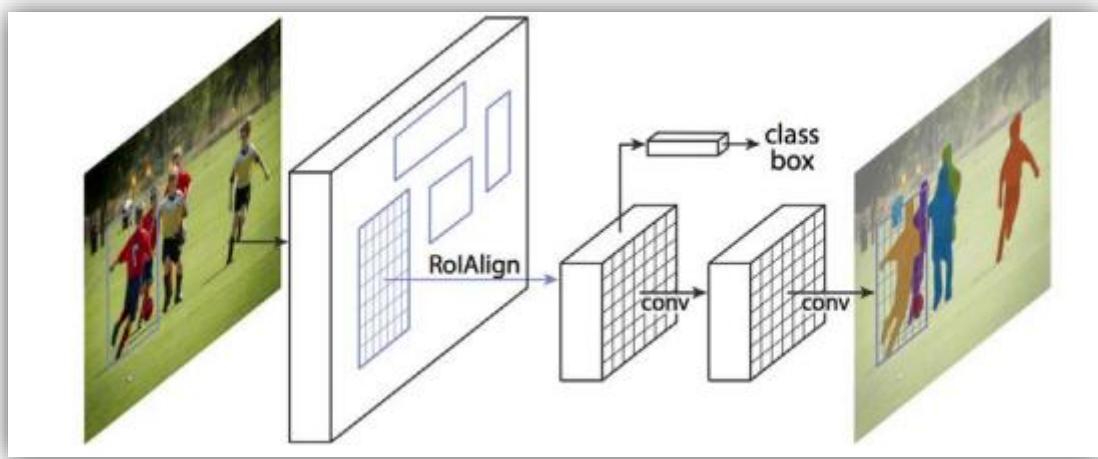


Figure : Schéma Mask R-CNN

9.- Difficultés

L'état des éventuelles difficultés techniques ou scientifiques rencontrées pour l'évaluation du projet Nous vous présentons un inventaire des difficultés auxquelles le projet a été confronté

- La principale difficulté rencontrée concerne la solution existante, hormis des sujets liés à la détection et reconnaissances d'objets, nous n'avons pas pu trouvé de sujets similaires à la nôtre c'est-à-dire pour les accessoires de véhicules.
- Pas d'analyse, ni de données disponibles sur place
- Nous avons eu des difficultés pour obtenir certaines données.
- Difficulté à appréhender les solutions possibles entre différentes échelles de façon explicite.

10.- Défis de la détection

Un détecteur d'objets doit faire face à plusieurs difficultés :

- La première est le temps de calcul : un bon détecteur est un détecteur qui maximise les performances tout en étant le plus rapide possible à exécuter. Cette notion de temps de calcul est particulièrement importante dans la conception de détecteurs multi-classes de par le plus grand nombre de classes d'objets à détecter.
- La seconde concerne l'apparence variable des objets. Celle-ci peut varier suivant plusieurs facteurs : l'occultation, le point de vue sous lequel les objets sont observés et la taille des objets dans l'image. Cette variabilité d'apparence est également une difficulté pour les systèmes de classification d'images.
- Une troisième difficulté est en lien avec la variation d'apparence des régions qui ne correspondent pas à des objets (le fond). Un détecteur doit être capable de décider si une région correspond à du fond, et cela, même dans des images provenant d'environnements complexes.

Conclusion

Étant donné que les technologies de détection, de reconnaissance et de classification des images n'en sont qu'à leurs débuts, on peut s'attendre à ce que de grandes choses se produisent dans un avenir proche. Le but de ce travail consistait à faire une étude détaillée sur les différents éléments utiles à notre projet.

Dans le prochain document nous aurons à présenter et retenir une solution que nous allons proposer pour une étude plus approfondie.

Partie Solution Proposée

Introduction

La reconnaissance d'objets est l'un des challenges les plus difficiles dans le domaine de la vision par ordinateur. Cependant, elle est considérée comme une étape primordiale dans de nombreuses applications comme médicales, industrielles, multimédia. . .

Par conséquent, ce domaine est depuis longtemps un objet d'intérêt pour la communauté scientifique. Différents objectifs et méthodes ont été proposés, depuis plus de 50 ans. En général, on peut les catégoriser en trois tâches en fonction de leur objectif :

- La catégorisation ou classification d'images qui consiste à donner un label à une image en fonction de la présence ou non d'un objet appartenant à une catégorie donnée.
- La détection d'objets qui désigne la tâche de localisation des objets d'une catégorie donnée.
- La segmentation de classes d'objets qui consiste à déterminer quels sont les pixels de l'image qui appartiennent à un objet d'une des classes d'intérêt.

2.- Notre Objectives

La reconnaissance d'objets, nous permet de détecter la présence d'une instance ou d'une classe d'objets, dans une image ou une scène naturelle. Selon Neisser, notre capacité à reconnaître un objet consiste en deux étapes :

- Un processus de sélection pour extraire les informations les plus pertinentes.
- Une chaîne complexe des processus pour identifier l'objet.

Pour reconnaître un objet, nous détectons les régions d'intérêts dans une image en nous basant sur certaines caractéristiques visuelles, comme la couleur, la texture. Chaque région détectée est ensuite représentée par un vecteur multidimensionnel. Ce concept peut être appliqué à la fois à un ensemble d'images et à une image requête. Après avoir calculé ces vecteurs, des mesures de similarité/distances entre les vecteurs représentant l'image requête et un ensemble des vecteurs sont calculées.

La reconnaissance est ainsi estimée à l'aide d'une des méthodes d'indexation existantes. Bien que cette approche ait eu une grande popularité dans le domaine de la reconnaissance d'objets, les algorithmes basés sur cette approche, s'appuient généralement sur l'analyse exhaustive du contenu de l'image au détriment d'une compréhension de plus haut niveau.

3.- Description de la solution

La première étape de notre étude constituera à trouver une base de donnée contenant plusieurs images de voiture. Une fois cela fait, l'utilisateur active l'application installée sur son téléphone et il prendra un cliché d'une partie des véhicules. Cette image sera stockée pour un traitement ultérieur (reconnaissance de chaque pièce sur l'image).

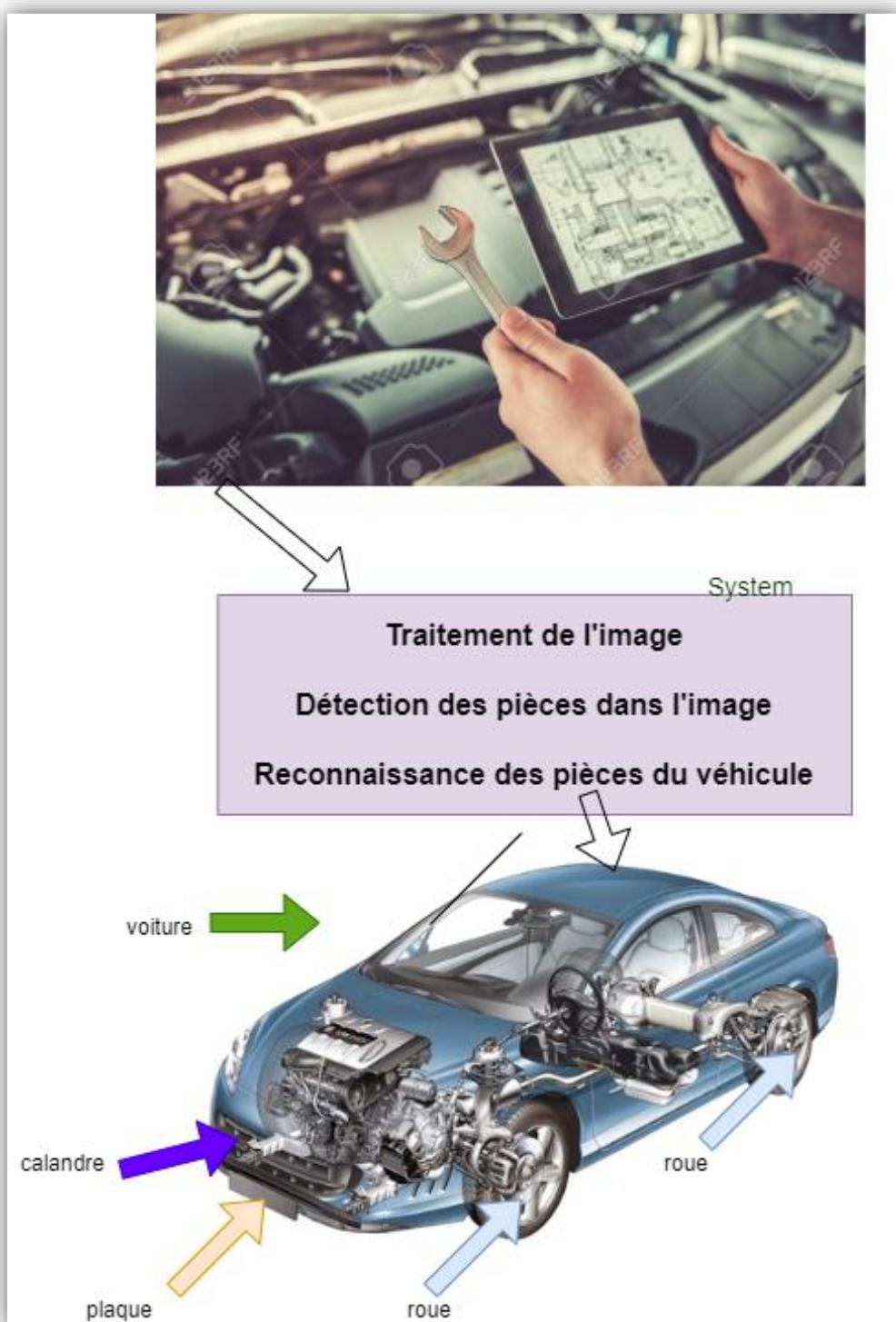


Figure : Schéma de notre solution proposée

4.- Nos stratégies de fonctionnement

La stratégie est constituée de deux modules principaux : module de détection et module de reconnaissance, comme représenté schématiquement ci-dessous.

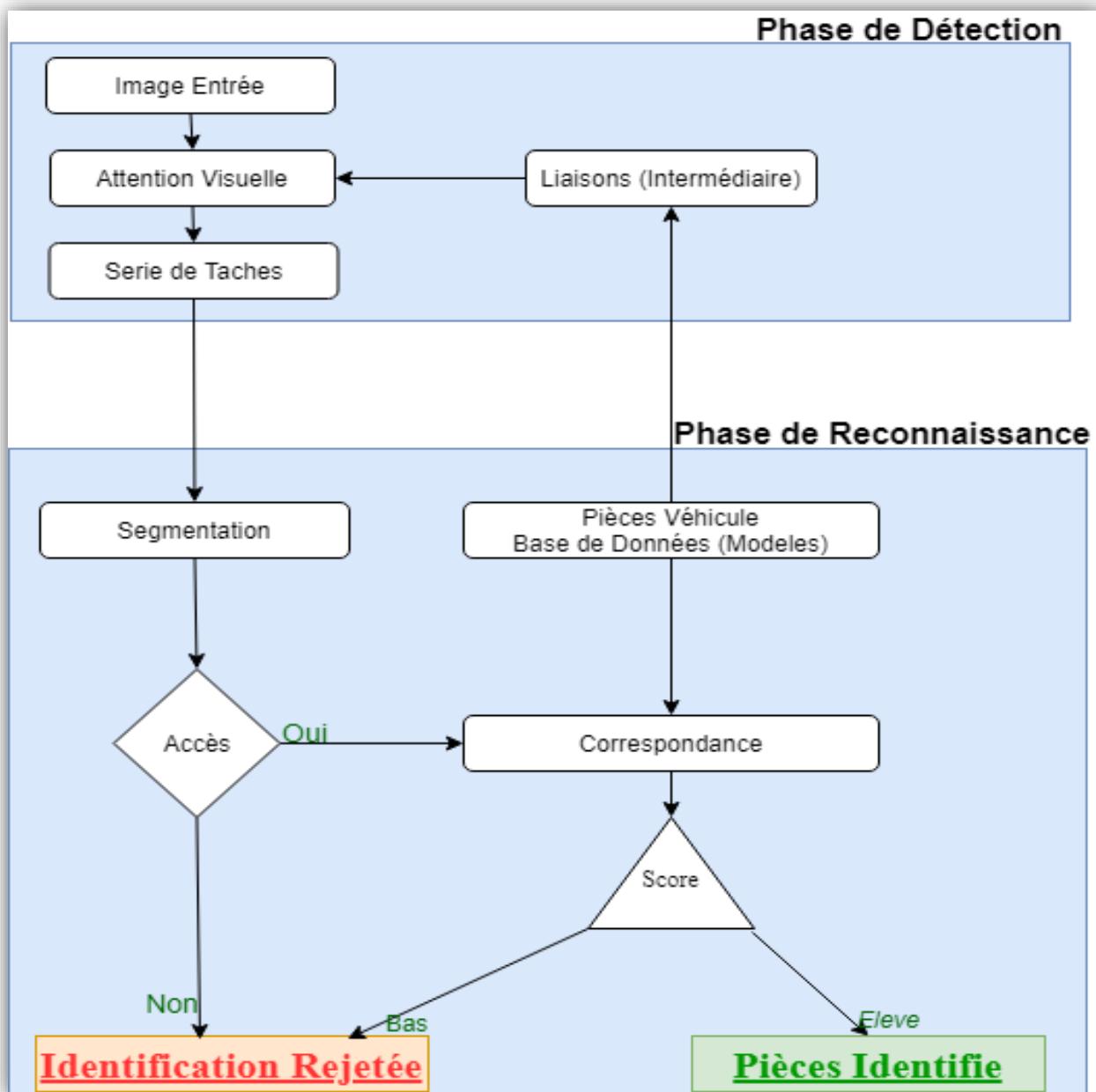


Figure: Schéma de stratégie d'identification des pièces de voiture

5.- Les méthodes de détection de pièces

Les algorithmes de détection de pièces dans une image fixe peuvent être rangés selon les trois catégories suivantes :

- **Modélisation colorimétrique :**

Une segmentation en composantes connexes fondée sur un modèle de couleur est opérée. Les régions d'intérêt sont ensuite validées par un algorithme de reconnaissance ou un modèle d'apparence. Ces méthodes sont les plus rapides mais aussi les moins robustes aux variations des conditions d'éclairage

- **Modélisation géométrique :**

Les contours de l'image sont analysés par une approche structurelle ou globale. Ces méthodes sont généralement plus robustes que celles photométriques parce qu'elles traitent le gradient de l'image, et peuvent traiter des images en niveaux de gris.

- **Méthodes avec apprentissage :**

Un classifieur (cascade, SVM, réseaux de neurones) est entraîné sur une base d'exemples. Il est appliqué sur une fenêtre glissante qui parcourt l'image à plusieurs échelles. Ces méthodes combinent géométrie et photométrie mais peuvent être une étape coûteuse en temps de calcul. Elles nécessitent la constitution d'une base d'apprentissage par type de pièces, étape fastidieuse lorsque le nombre d'objets à détecter est grand.

6.- Présentation des données

Le but des travaux de recherche en détection et reconnaissance des pièces de voiture. Cet outil devrait être capable de travailler en temps réel et devrait comporter un module dédié à l'acquisition du flux vidéo. Les travaux effectués se basent soit sur des images fixes, soit sur des séquences vidéo acquises en conditions réelles.

Lien : https://ai.stanford.edu/~jkrause/cars/car_dataset.html

Ce **dataset Cars** contient 16 185 images de 196 classes de voitures. Les données sont divisées en 8 144 images d'entraînement et 8 041 images de test, où chaque classe a été divisée à peu près à 50-50. Les classes sont généralement au niveau de la marque, du modèle, de l'année, par exemple Tesla Model S 2012 ou BMW M3 coupé 2012.

NB : Par contre au cours de la réalisation de ce projet pour minimiser le temps d'entraînement et en fonction des ressources nécessaire nous n'avons pas utilisé la totalité de ces données, nous avons utilisés 240 images pour l'entraînement et 60 pour test.

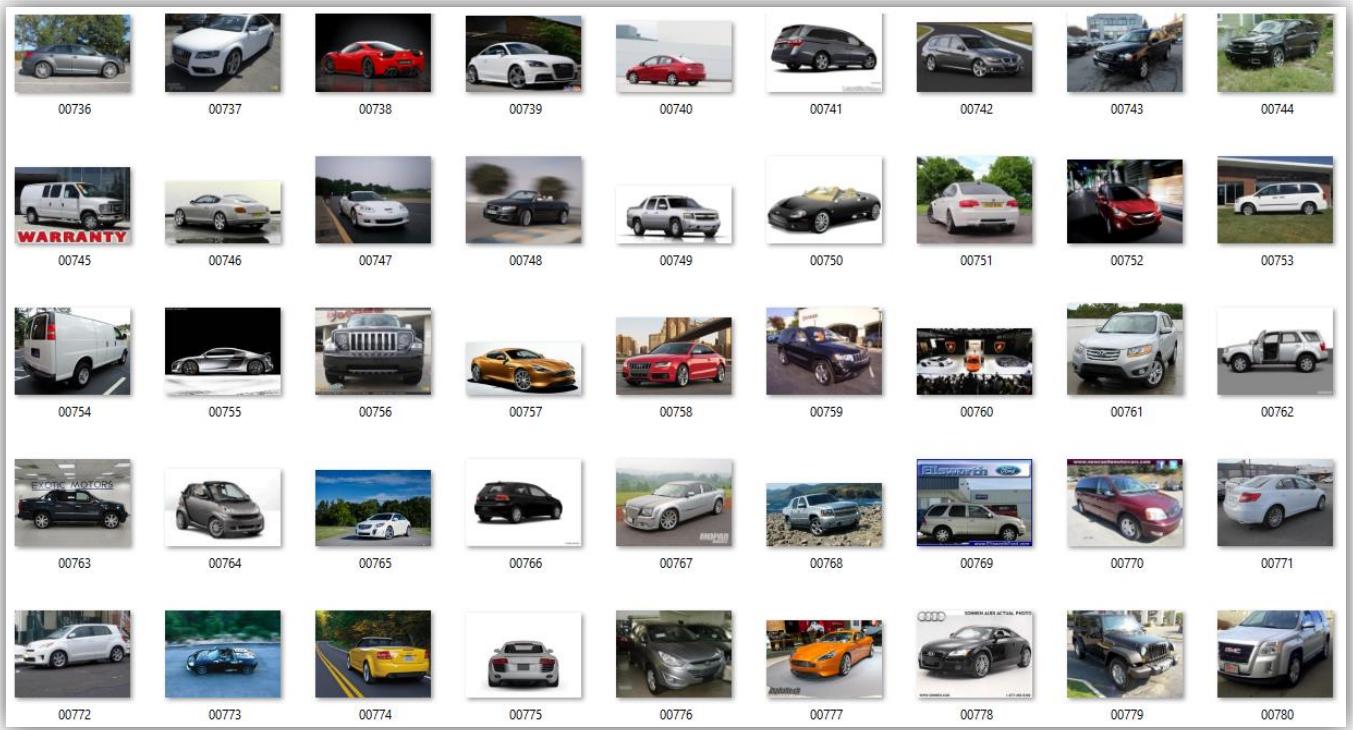


Figure : Le Dataset Cars

7.- Scénario de Test et de Train

Dans les projets de Détection, il est important de garder une partie des données en dehors de l'entraînement. En effet, il serait autrement impossible d'assurer que le modèle puisse généraliser en dehors des cas déjà présents ou non prévus.

Pour ce projet, nous avons séparé les données en 2 sets distincts : Test et Train.

Images	Train	Test
300	80%	20%

Tableau : Répartition de nos données pour le projet

Pour réaliser ces ensembles de données, nous avons sorti des images contigües du set d'entraînement dans chaque source de données. Les données devaient être contigües à cause du chevauchement lors

de la séparation des images et pour éviter que les mêmes données se retrouvent dans le set d'entraînement et les deux sets d'évaluation. Le set de test permet d'observer l'amélioration de la précision du modèle entre les différentes itérations. Le set d'évaluation permet d'avoir des données pour l'évaluation finale. En effet, il est important d'avoir des données non utilisée pendant les diverses itérations afin d'assurer que le modèle peut fonctionner dans des cas non prévus. L'évaluation m'étier sera aussi en partie réalisée sur des images en dehors du set de données et sans labels de vérité.

8.- L'entraînement ou l'apprentissage

Cette partie consiste en la formation d'un réseau de neurone convolutif à partir des images de voitures.

Trois (03) stratégies d'apprentissage par transfert peuvent être utilisées :

- **Apprentissage superficiel** : qui consiste à affiner uniquement les couches entièrement connectées, et le reste du réseau est utilisé comme extracteur de caractéristiques.
- **Stratégie profonde** : affine toutes les couches du réseau et commence l'optimisation de la rétro-propagation à partir du réseau préformé. En utilisant ces deux approches, le classificateur CNN essaie d'en savoir plus sur les caractéristiques spécifiques de la classification des pièces.
- la troisième stratégie consiste à entraîner le CNN à partir de zéro à partir d'une configuration aléatoire des poids.
- **Préformation** : consiste à former un CNN profond sur un grand ensemble de données comme ImageNet d'abord, avant la formation sur notre ensemble de données. Cette préformation est réalisée afin de préparer le CNN par le transfert

9.- La classification

Après l'apprentissage des caractéristiques dans plusieurs couches, l'architecture d'un CNN se poursuit par la classification.

L'avant-dernière couche est une couche entièrement connectée qui sort un vecteur de K dimensions, où K représente le nombre de classes que le réseau sera capable de prédire. Ce vecteur contient les probabilités de classement pour chaque classe d'une image.

La dernière couche d'une architecture de CNN utilise une couche de classification (softmax par exemple) pour fournir le résultat de classification.

10.- Le déploiement

Une fois les modèles formés à partir de l'apprentissage, ils peuvent être déployés sur les machines utilisateurs (ordinateurs et mobiles) et utilisés en deux modes à savoir le mode classification des pièces et le mode détection. Nous allons utiliser un convertisseur TensorFlow pour convertir les modèles formés par TensorFlow au format de fichier TensorFlow Lite (.tflite) qui sera utilisé sur les téléphones portables. TensorFlow Lite est la solution légère de TensorFlow pour les appareils mobiles et intégrés. Il permet une inférence d'apprentissage machine sur le périphérique avec une faible latence et une petite taille binaire. TensorFlow Lite prend également en charge l'accélération matérielle avec l'API Android Neural Networks.

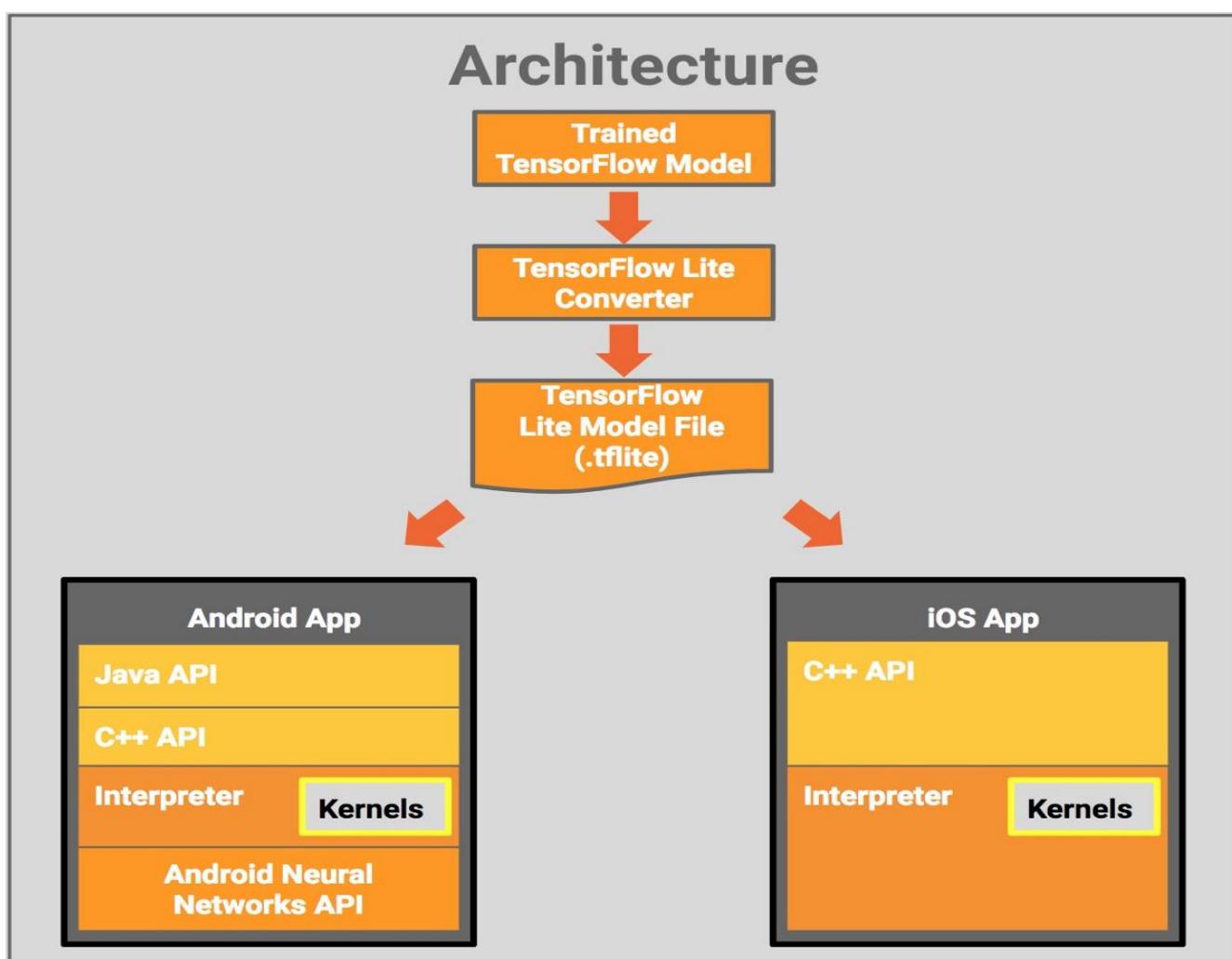


Figure : Architecture déploiement du model tensorflow

11.- Plan de travail

La figure ci-dessous présente le plan de travail que nous allons suivre pour implémenter notre Solution.

No	Taches	Durée
1	Installation des outils et différentes	1 Semaine
2	Configurations de la structure du répertoire de détection d'objets	1 Semaine
3	Acquisition des données	2 Semaines
4	Classification des données	2 Semaine
5	Créer une carte d'étiquettes et configurer la formation	2 Semaine
6	Apprentissage du réseau de neurones	3 Semaines
7	Exploration du graphique d'inférence	1 Semaine
9	Test & Mise en production	2 Semaines
10	Rapport Final	3 Semaines

12.- Outils à utilisés

Les outils à utiliser et leur description sont présentés dans le tableau suivant :

Outils	Description
Python	C'est un langage de programmation objet interprété multi-paradigme et multiplateformes
Tensorflow	C'est une bibliothèque open source de machine Learning. Cree par Google, permettant de développer et d'exécuter des applications de Machine Learning et de Deep Learning
Tensorflow Lite	Tensorflow Lite est la solution légère de Tensorflow pour les appareils mobile et intégrés. Il permet une inférence d'apprentissage machine sur le périphérique avec une faible latence et une petite taille binaire
Keras	C'est une bibliothèque open source écrite en python et permettant d'interagir avec les algorithmes de réseaux e neurones profonds et de machine Learning
Numpy	C'est une extension du langage de programmation Python. Destinée a manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux
Google Colaboratory Pro	C'est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud
Anaconda	Anaconda est une distribution <u>libre</u> et <u>open source</u> des langages de programmation <u>Python</u> et <u>R</u> appliquée au développement d'applications dédiées à la science des données et à l'apprentissage automatique
LabelImg	C'est un outil d'annotation d'images graphiques
Android Studio	Android Studio est un environnement de développement pour développer des applications mobiles Android
Google drive	
Bazel	Système de compilation créé à l'origine par Google

Tableau : Outils utilisé pour la réalisation de nos projets

Conclusion

En conclusion, dans ces travaux de recherches nous avons apporté plusieurs solutions à notre problème de détection et de reconnaissance d'Object, nous avons évoqué les différentes étapes nécessaires pour construire un système de reconnaissance d'objets classique.

Nous avons aussi présenté les différents travaux proposés pour détecter les points d'intérêts la première étape d'un système de reconnaissance d'objets.

LIEN :

<https://github.com/IngH2020/Detection-et-reconnaissance-des-pieces-de-voiture-avec-Deep-Learning/upload/master>

Partie Conception et Implémentation

1.- Conception et Implémentation

Toute l'implémentation de notre solution se fera sur un environnement d'exécution hébergé sur Google colaboratory PRO et Anaconda prompt.

1.1.- Préparation des données

LabelImg est un excellent logiciel libre qui facilite grandement le processus d'étiquetage. Il enregistrera des étiquettes XML individuelles pour chaque image que vous avez traitée. Nous avons labélisé tous les images de notre dataset

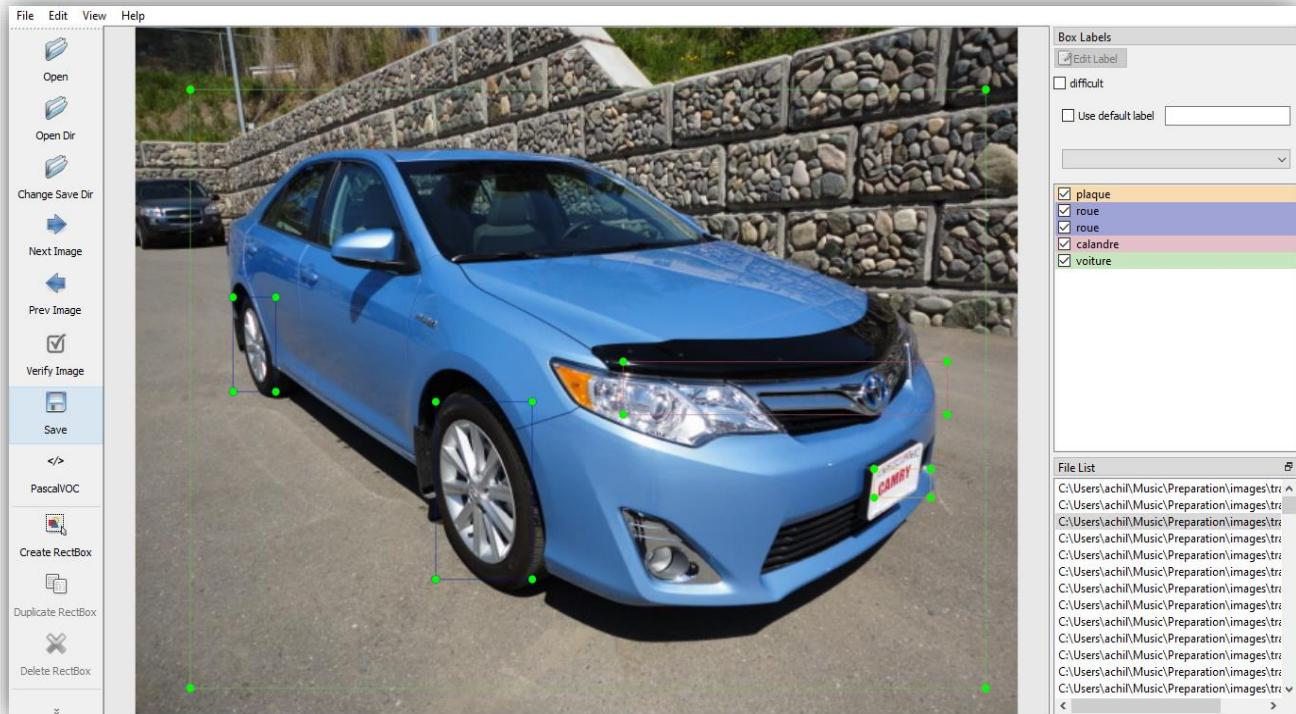


Figure : Etiquetage les images

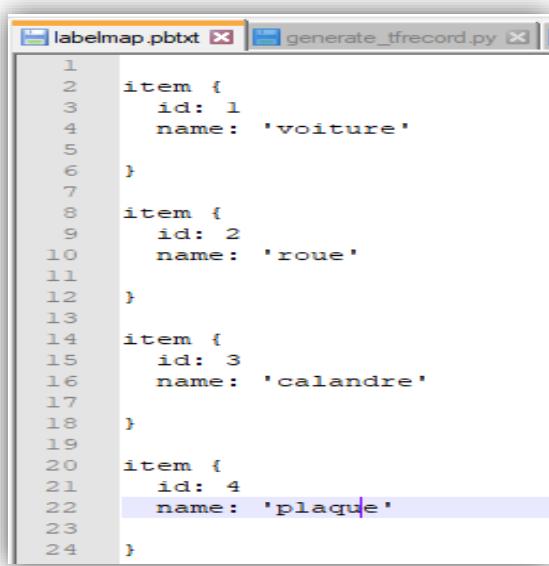
Les images et les fichiers xml de notre dataset :



Figure : Les images et les fichiers xml

❖ **Nous avons créé un dossier préparation contenant :**

- ✚ Un dossier image qui contient les images étiqueter test et train.
- ✚ Le fichier **faster_rcnn_inception_v2_pets.config**, nous avons modifié le nombre de classe et le nombre de photo qu'on a dans le dossier test
- ✚ Nous devons créer des TFRecords qui peuvent être utilisés comme données d'entrée pour l'apprentissage du détecteur des pièces de voiture.
- ✚ Un fichier **train.py**
- ✚ Nous avons créé un fichier **labelmap.pbtxt**, La carte d'étiquettes mappe un identifiant à un nom. Le **labelmap** de mon détecteur peut être vu ci-dessous.



```
1 item {
2   id: 1
3   name: 'voiture'
4 }
5
6 item {
7   id: 2
8   name: 'roue'
9 }
10
11 item {
12   id: 3
13   name: 'calandre'
14 }
15
16 item {
17   id: 4
18   name: 'plaque'
19 }
20
21 }
```

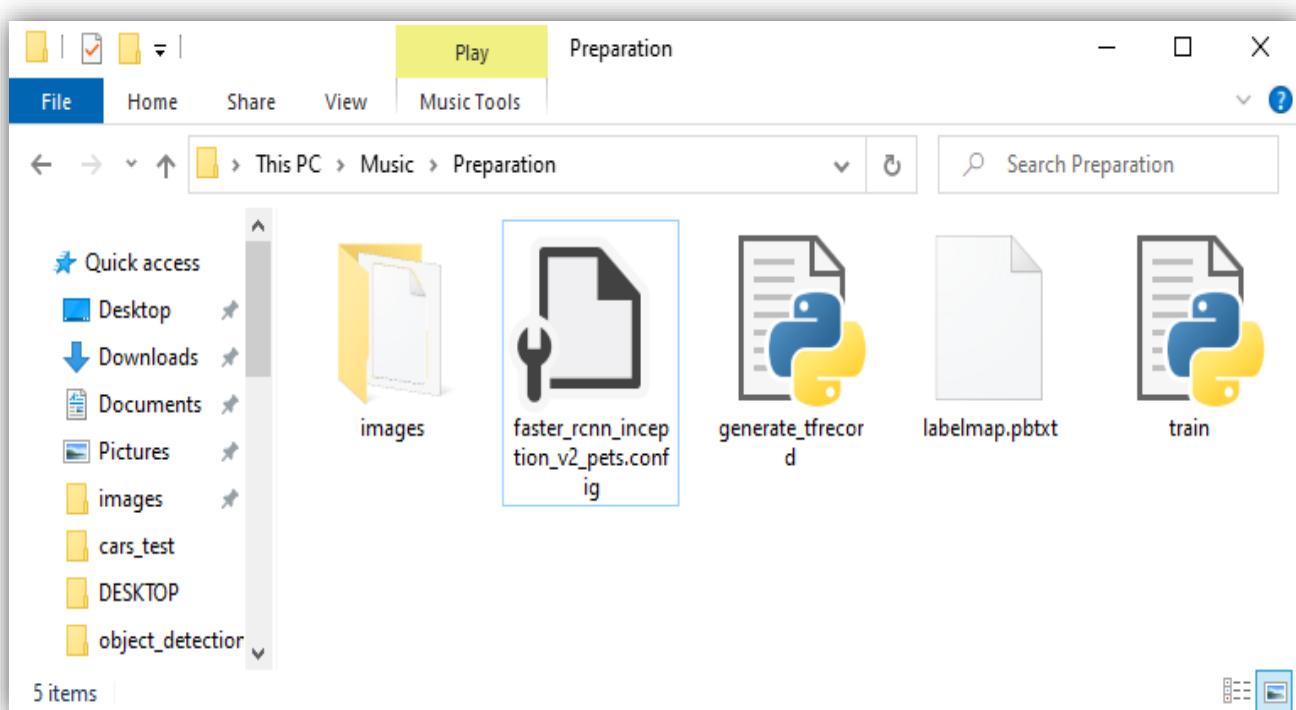


Figure : Le fichier Préparation

Apres avoir ajouté tous ces fichier dans le répertoire préparation, nous avons zippé le fichier et l'importer vers Google Drive pour pouvoir continuer sur Colab pro

1.2.- Implémentation sur Colab Pro

Nous avons utilisé COLAB PRO python3 avec 26GB FREE GPU pour l'implémentation de notre modèle de détections de pièces voitures

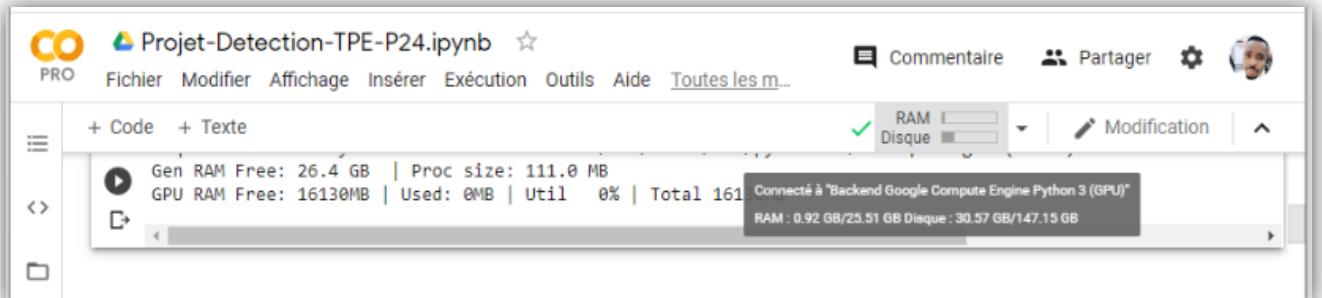


Figure : Capacité de Colab Pro

- 1) Pour commencer nous avons installer pip et protobuf

A screenshot of the Google Colab Pro interface showing a code cell execution. The command entered is "pip install --upgrade pip\npip install --upgrade protobuf". The output shows that both packages are already up-to-date or satisfied, with messages like "Requirement already up-to-date: pip in /usr/local/lib/python3.6/dist-packages (20.2.2)" and "Requirement already satisfied, skipping upgrade: six>=1.9 in /usr/local/lib/python3.6/dist-packages (from protobuf)".

Figure : Installation de pip et Protobuf

L'API de détection d'objets TensorFlow repose sur ce que l'on appelle des tampons de protocole (également appelés protobufs). Les protobufs sont une manière neutre de décrire les informations. Cela signifie que vous pouvez écrire un protobuf une fois, puis le compiler pour être utilisé avec d'autres langages, comme Python, Java ou C.

- 2) En seconde lieu nous avons installé tensorflow 1.15 et numpy

The screenshot shows a Google Colab notebook titled "Projet-Detection-TPE-P24.ipynb". In the code cell, the following commands are run:

```
%tensorflow_version 1.15
import tensorflow as tf

!pip install numpy
```

The output shows a warning about the `%tensorflow_version` command:

```
^%tensorflow_version` only switches the major version: 1.x or 2.x.
You set: `1.15`. This will be interpreted as: `1.x`.

TensorFlow 1.x selected.
```

Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (1.18.5)

Figure : Installation de Numpy et tensorflow 1.15

Numpy est destiné au calcul scientifique, toutes les opérations mathématiques peuvent être effectuées avec numpy. TensorFlow fournit toutes les fonctions intégrées pour l'apprentissage automatique, ce qui rend le code très court et facile. Avec numpy, vous devez résoudre les problèmes d'apprentissage automatique de manière mathématique, TensorFlow s'en charge.

- 3) Connection avec mon Google Drive pour pouvoir accéder avec mon fichier préparation que j'ai importé

The screenshot shows a Google Colab notebook titled "Projet-Detection-TPE-P24.ipynb". In the code cell, the following command is run:

```
from google.colab import drive
drive.mount('/content/drive')
```

The output shows the connection process:

```
^Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g
Enter your authorization code:
.....
Mounted at /content/drive
```

Figure : Connexion avec Google drive

- 4) Installation de fichier API object détection tensorflow depuis github

```

!wget https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects/master.zip
--2020-09-02 17:41:14-- https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-M...
Resolving github.com (github.com)... 140.82.118.3
Connecting to github.com (github.com)|140.82.118.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-M...
--2020-09-02 17:41:14-- https://codeload.github.com/EdjeElectronics/TensorFlow-Object-Detection-API-T...
Resolving codeload.github.com (codeload.github.com)... 140.82.121.10
Connecting to codeload.github.com (codeload.github.com)|140.82.121.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'master.zip'

master.zip          [=====] 57.26M 17.3MB/s   in 3.3s

```

5) Installez les outils et dépendances nécessaire

```

!apt-get install protobuf-compiler python-pil python-lxml python-tk
!pip install Cython
%cd /content/models-master/research
!protoc object_detection/protos/*.proto --python_out=.

import os
os.environ['PYTHONPATH'] += '/content/models-master/research/:/content/models-master/research/slim'

!python setup.py build
!python setup.py install

Extracting object_detection-0.1-py3.6.egg to /usr/local/lib/python3.6/dist-packages
Adding object-detection 0.1 to easy-install.pth file

Installed /usr/local/lib/python3.6/dist-packages/object_detection-0.1-py3.6.egg
Processing dependencies for object-detection==0.1
Searching for Cython==0.29.21
Best match: Cython 0.29.21
Adding Cython 0.29.21 to easy-install.pth file
Installing cygdb script to /usr/local/bin
Installing cython script to /usr/local/bin
Installing cythonize script to /usr/local/bin

Using /usr/local/lib/python3.6/dist-packages
Searching for matplotlib==3.2.2
Best match: matplotlib 3.2.2
Adding matplotlib 3.2.2 to easy-install.pth file

Using /usr/local/lib/python3.6/dist-packages

```

Figure : Installation de Protobuf, lxml et cython

Google Colab a préinstallé la plupart des logiciels Python, Tensorflow. Ce sont les packages dont nous avons besoin et ne seront pas préinstallés par défaut et nous avons compilé des proto buffers pour créer fichier name_pb2.py

6) Nous avons unzip et copier le fichier preparation dans le modèle de détection tensorflow

The screenshot shows a terminal window in Google Colab. The command entered is:

```
%cp -a /content/drive/'My Drive'/Preparation.zip /content/models-master/research/object_detection  
%cd /content/models-master/research/object_detection  
!unzip Preparation.zip
```

The output shows the contents of the Preparation.zip file being extracted into the /content/models-master/research/object_detection directory. The output text is as follows:

```
/content/models-master/research/object_detection  
Archive: Preparation.zip  
  creating: Preparation/  
  inflating: Preparation/Data Explained ModelTrainingOnColab_final_faster_rcnn.ipynb  
  inflating: Preparation/faster_rcnn_inception_v2_pets.config  
  inflating: Preparation/generate_tfrecord.py  
  creating: Preparation/images/  
  creating: Preparation/images/test/  
  inflating: Preparation/images/test/0.jpg  
  inflating: Preparation/images/test/0.xml  
  inflating: Preparation/images/test/00035.jpg  
  inflating: Preparation/images/test/00035.xml  
  inflating: Preparation/images/test/00037.jpg  
  inflating: Preparation/images/test/00037.xml  
  inflating: Preparation/images/test/00038.jpg  
  inflating: Preparation/images/test/00038.xml  
  inflating: Preparation/images/test/00091.jpg  
  inflating: Preparation/images/test/00091.xml  
  inflating: Preparation/images/test/00092.jpg  
  inflating: Preparation/images/test/00092.xml  
  inflating: Preparation/images/test/00096.jpg  
  inflating: Preparation/images/test/00096.xml  
  inflating: Preparation/images/test/00100.jpg  
  inflating: Preparation/images/test/00100.xml  
  inflating: Preparation/images/test/00104.jpg  
  inflating: Preparation/images/test/00104.xml
```

Figure: Unzip le fichier Préparation

7) Générer des données d'entraînement

Avec les images étiquetées, il est temps de générer les TFRecords qui servent de données d'entrée au modèle d'entraînement TensorFlow. Ce didacticiel utilise les scripts `xml_to_csv.py` et `generate_tfrecord.py` de l'ensemble de données Raccoon Detector de Dat Tran , avec quelques légères modifications pour fonctionner avec notre structure de répertoires.

```
%cd /content/models-master/research/object_detection/Preparation
!python xml_to_csv.py
```

```
/content/models-master/research/object_detection/Preparation
Successfully converted xml to csv.
Successfully converted xml to csv.
```

Figure: Création des fichiers .csv de test et de train

Cela crée un fichier train_labels.csv et test_labels.csv dans le dossier \ preparation \ images.

filename	width	height	class	xmin	ymin	xmax	ymax
00264.jpg	528	303	voiture	5	96	528	258
00264.jpg	528	303	roue	66	182	144	258
00264.jpg	528	303	roue	368	176	447	261
00017.jpg	640	424	voiture	15	50	631	368
00017.jpg	640	424	calandre	388	172	619	228
00017.jpg	640	424	roue	277	228	387	365
00017.jpg	640	424	roue	39	204	118	316
00001.jpg	276	182	voiture	16	44	250	162
00001.jpg	276	182	Roue	39	97	70	136
00001.jpg	276	182	Roue	144	108	178	146
00001.jpg	276	182	calandre	217	102	245	120
196.jpg	700	500	voiture	40	121	669	460
196.jpg	700	500	roue	336	293	474	455
196.jpg	700	500	roue	628	244	661	339
196.jpg	700	500	calandre	43	244	257	331
00082.jpg	160	120	voiture	12	9	147	101
00082.jpg	160	120	calandre	75	67	139	78
00082.jpg	160	120	plaque	104	84	128	87
00256.jpg	640	426	voiture	44	87	602	362
00256.jpg	640	426	calandre	404	212	584	262
00256.jpg	640	426	roue	73	230	156	326
00256.jpg	640	426	roue	300	258	369	346
176.jpg	700	500	voiture	41	51	650	434
176.jpg	700	500	roue	53	192	353	288
176.jpg	700	500	roue	372	250	476	433

Figure : Les fichiers train et test_labels.csv

Générez les fichiers TFRecord

```
[ ] %cd /content/models-master/research/object_detection/Preparation
!python generate_tfrecord.py --csv_input=images/test_labels.csv --image_dir=images/test --output_path=test.record
!python generate_tfrecord.py --csv_input=images/train_labels.csv --output_path=train.record --image_dir=images/train/

```

The code cell contains two commands using the `generate_tfrecord.py` script. The first command generates a test record from `test_labels.csv` in the `test` directory. The second command generates a train record from `train_labels.csv` in the `train` directory. Both commands output to their respective `test.record` and `train.record` files.

Figure: Création des fichiers .csv de test et de train

Ceux-ci génèrent un fichier `train.record` et un fichier `test.record` dans `object_detection`. Ceux-ci seront utilisés pour former le nouveau classificateur de détection d'objets.

8) Nous avons vérifié le temps GPU restant

```
+ Code + Texte
Vérifier le temps GPU restant

import time, psutil
Start = time.time() - psutil.boot_time()
Left= 12*3600 - Start
print('Time remaining for this session is: ', Left/3600)

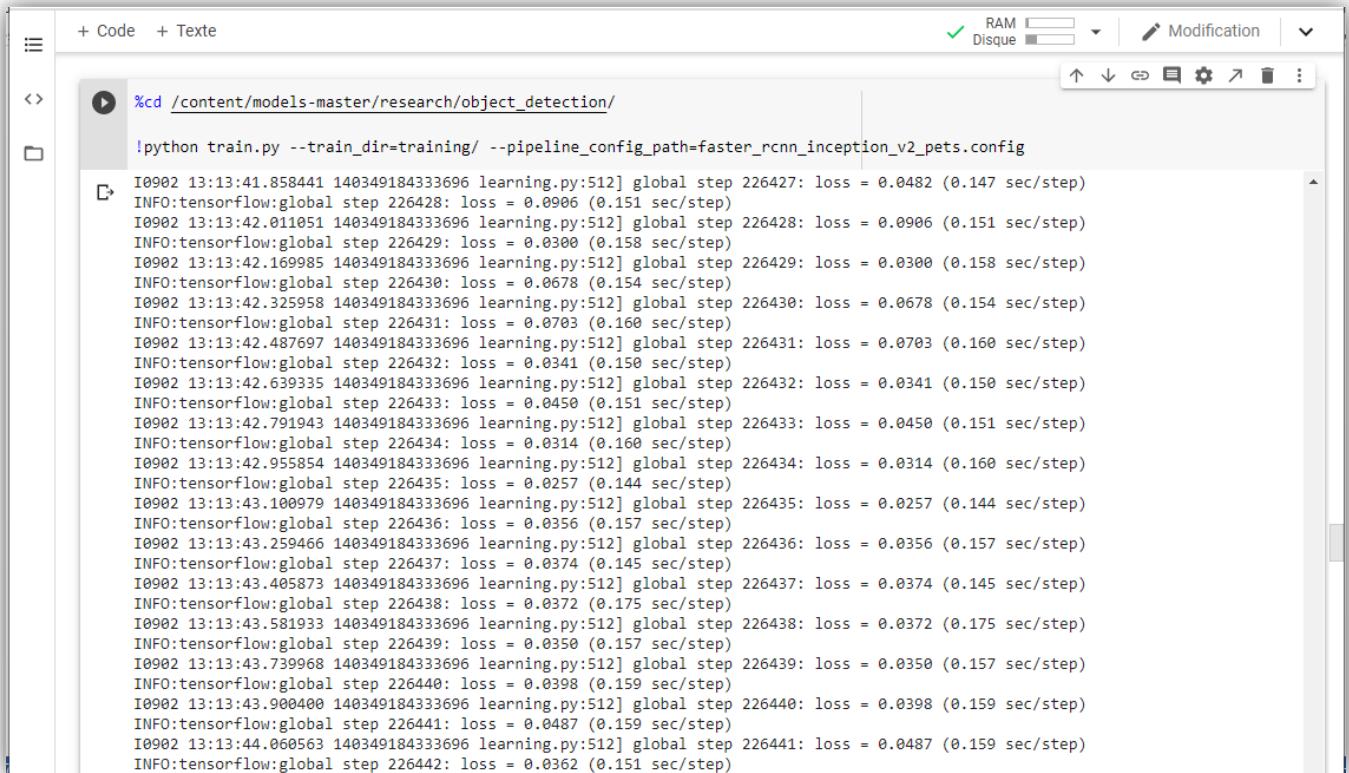
Time remaining for this session is:  9.412536661492453
```

The code cell imports `time` and `psutil`, calculates the start time, and prints the remaining time for the session. The output shows approximately 9.41 hours remaining.

Figure : Vérification du temps

Pour connaître les heures restantes dont nous disposons pour notre session colab

9) Nous avons lancé l'entraînement



The screenshot shows a terminal window with the following command and its output:

```
%cd /content/models-master/research/object_detection/  
!python train.py --train_dir=training/ --pipeline_config_path=faster_rcnn_inception_v2_pets.config
```

Output log:

```
I0902 13:13:41.858441 140349184333696 learning.py:512] global step 226427: loss = 0.0482 (0.147 sec/step)  
INFO:tensorflow:global step 226428: loss = 0.0966 (0.151 sec/step)  
I0902 13:13:42.011051 140349184333696 learning.py:512] global step 226428: loss = 0.0906 (0.151 sec/step)  
INFO:tensorflow:global step 226429: loss = 0.0300 (0.158 sec/step)  
I0902 13:13:42.169985 140349184333696 learning.py:512] global step 226429: loss = 0.0300 (0.158 sec/step)  
INFO:tensorflow:global step 226430: loss = 0.0678 (0.154 sec/step)  
I0902 13:13:42.325958 140349184333696 learning.py:512] global step 226430: loss = 0.0678 (0.154 sec/step)  
INFO:tensorflow:global step 226431: loss = 0.0703 (0.160 sec/step)  
I0902 13:13:42.487697 140349184333696 learning.py:512] global step 226431: loss = 0.0703 (0.160 sec/step)  
INFO:tensorflow:global step 226432: loss = 0.0341 (0.150 sec/step)  
I0902 13:13:42.639335 140349184333696 learning.py:512] global step 226432: loss = 0.0341 (0.150 sec/step)  
INFO:tensorflow:global step 226433: loss = 0.0450 (0.151 sec/step)  
I0902 13:13:42.791943 140349184333696 learning.py:512] global step 226433: loss = 0.0450 (0.151 sec/step)  
INFO:tensorflow:global step 226434: loss = 0.0314 (0.160 sec/step)  
I0902 13:13:42.955854 140349184333696 learning.py:512] global step 226434: loss = 0.0314 (0.160 sec/step)  
INFO:tensorflow:global step 226435: loss = 0.0257 (0.144 sec/step)  
I0902 13:13:43.100079 140349184333696 learning.py:512] global step 226435: loss = 0.0257 (0.144 sec/step)  
INFO:tensorflow:global step 226436: loss = 0.0356 (0.157 sec/step)  
I0902 13:13:43.259466 140349184333696 learning.py:512] global step 226436: loss = 0.0356 (0.157 sec/step)  
INFO:tensorflow:global step 226437: loss = 0.0374 (0.145 sec/step)  
I0902 13:13:43.405873 140349184333696 learning.py:512] global step 226437: loss = 0.0374 (0.145 sec/step)  
INFO:tensorflow:global step 226438: loss = 0.0372 (0.175 sec/step)  
I0902 13:13:43.581933 140349184333696 learning.py:512] global step 226438: loss = 0.0372 (0.175 sec/step)  
INFO:tensorflow:global step 226439: loss = 0.0350 (0.157 sec/step)  
I0902 13:13:43.739968 140349184333696 learning.py:512] global step 226439: loss = 0.0350 (0.157 sec/step)  
INFO:tensorflow:global step 226440: loss = 0.0398 (0.159 sec/step)  
I0902 13:13:43.900400 140349184333696 learning.py:512] global step 226440: loss = 0.0398 (0.159 sec/step)  
INFO:tensorflow:global step 226441: loss = 0.0487 (0.159 sec/step)  
I0902 13:13:44.060563 140349184333696 learning.py:512] global step 226441: loss = 0.0487 (0.159 sec/step)  
INFO:tensorflow:global step 226442: loss = 0.0362 (0.151 sec/step)
```

Figure : Entrainement de notre model

Chaque étape de la formation rapporte la perte. Il a commencé haut et diminuera de plus en plus au fur et à mesure que l'entraînement progressera. Pour ma formation sur le modèle Faster-RCNN-Inception-V2, il a commencé à environ 3,0 et est rapidement tombé en dessous de 0,8. Je recommande de permettre à votre modèle de s'entraîner jusqu'à ce que la perte tombe constamment en dessous de 0,05, ce qui prendra environ 400000 pas, soit environ 12 heures de temps d'entraînement.

Remarque: les nombres de pertes seront différents si nous avions utilisé MobileNet-SSD, il commence par une perte d'environ 20 et doit être formé jusqu'à ce que la perte soit constamment inférieure à 2.

10) Exporter le graphique d'inférence dans Google Drive

Maintenant que la formation est terminée, la dernière étape consiste à générer le graphe d'inférence figé (fichier .pb). Dans le dossier \ object_detection, exécutez la commande suivante, où «XXXX» dans «model.ckpt-XXXX» doit être remplacé par le fichier .ckpt portant le numéro le plus élevé dans le dossier de formation

```

+ Code + Texte
RAM Disque Modification
!python export_inference_graph.py --input_type image_tensor --pipeline_config_path faster_rcnn_inception_v2_pets.config
--trained_checkpoint_prefix training/model.ckpt-3440 --output_directory new_graph
2020-09-02 03:31:14.054005: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:14.054829: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:14.054856: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:14.054937: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had
2020-09-02 03:31:14.055570: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had
2020-09-02 03:31:14.056173: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1767] Adding visible gpu devices: 0
2020-09-02 03:31:14.056245: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:14.057790: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1180] Device interconnect StreamExecutor with stre
2020-09-02 03:31:14.057836: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1186]          0
2020-09-02 03:31:14.057853: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0: N
2020-09-02 03:31:14.057984: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had
2020-09-02 03:31:14.058632: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had
2020-09-02 03:31:14.059240: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:39] Overriding allow_growth setting because
2020-09-02 03:31:14.059287: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhost/re
INFO:tensorflow:Restoring parameters from training/model.ckpt-3440
I0902 03:31:14.061465 140670608271232 saver.py:1284] Restoring parameters from training/model.ckpt-3440
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/tools/freeze_graph.py:127: checkpoint_exists (from te
Instructions for updating:
Use standard file APIs to check for files with this prefix.
W0902 03:31:15.506398 140670608271232 deprecation.py:323] From /tensorflow-1.15.2/python3.6/tensorflow_core/python/tools/freeze_g
Instructions for updating:
Use standard file APIs to check for files with this prefix.
2020-09-02 03:31:16.295776: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS had
2020-09-02 03:31:16.296458: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1639] Found device 0 with properties:
name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0000:00:04.0
2020-09-02 03:31:16.296547: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:16.296574: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:16.296607: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:16.296632: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1
2020-09-02 03:31:16.296677: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library 1

```

Figure : Exportation du fichier interface_gragh

Cela crée un fichier new_inference_graph.pb dans le dossier \ Object détection \ inference_graph. Le fichier .pb contient le classificateur de détection d'objet.

11) Enregistrer le model entrainer dans Google Drive

```

+ Code + Texte
RAM Disque Modification
zip -r models-master.zip /content/models-master
!cp -a /content/models-master.zip /content/drive/'My Drive'

adding: content/models-master/research/efficient-hrl/run_eval.py (deflated 51%)
adding: content/models-master/research/efficient-hrl/agents/ (stored 0%)
adding: content/models-master/research/efficient-hrl/agents/_init__.py (stored 0%)
adding: content/models-master/research/efficient-hrl/agents/ddpg_agent.py (deflated 83%)
adding: content/models-master/research/efficient-hrl/agents/ddpg_networks.py (deflated 71%)
adding: content/models-master/research/efficient-hrl/agents/circular_buffer.py (deflated 72%)
adding: content/models-master/research/efficient-hrl/run_env.py (deflated 62%)
adding: content/models-master/research/efficient-hrl/eval.py (deflated 72%)
adding: content/models-master/research/efficient-hrl/environments/ (stored 0%)
adding: content/models-master/research/efficient-hrl/environments/assets/ (stored 0%)
adding: content/models-master/research/efficient-hrl/environments/assets/ant.xml (deflated 78%)
adding: content/models-master/research/efficient-hrl/environments/maze_env_utils.py (deflated 78%)
adding: content/models-master/research/efficient-hrl/environments/maze_env.py (deflated 76%)
adding: content/models-master/research/efficient-hrl/environments/_init__.py (stored 0%)
adding: content/models-master/research/efficient-hrl/environments/ant_maze_env.py (deflated 44%)
adding: content/models-master/research/efficient-hrl/environments/point_maze_env.py (deflated 44%)
adding: content/models-master/research/efficient-hrl/environments/point.py (deflated 57%)
adding: content/models-master/research/efficient-hrl/environments/create_maze_env.py (deflated 57%)
adding: content/models-master/research/efficient-hrl/environments/ant.py (deflated 62%)
adding: content/models-master/research/efficient-hrl/agent.py (deflated 78%)
adding: content/models-master/research/efficient-hrl/run_train.py (deflated 49%)
adding: content/models-master/research/efficient-hrl/cond_fn.py (deflated 79%)
adding: content/models-master/research/efficient-hrl/train_utils.py (deflated 68%)
adding: content/models-master/research/efficient-hrl/train.py (deflated 77%)
adding: content/models-master/research/efficient-hrl/configs/ (stored 0%)
adding: content/models-master/research/efficient-hrl/configs/train_uvf.gin (deflated 70%)
adding: content/models-master/research/efficient-hrl/configs/eval_uvf.gin (deflated 48%)
adding: content/models-master/research/efficient-hrl/configs/base_uvf.gin (deflated 71%)
adding: content/models-master/research/efficient-hrl/context/ (stored 0%)
adding: content/models-master/research/efficient-hrl/context/context.py (deflated 73%)

```

Figure : Enregistrement du model de détection tensorflow

1.3.- Implémentation sur anaconda prompt

Après l'enregistrement du modèle de détection, nous avons téléchargé les deux fichiers et pour l'expérimentation. Nous avons travaillé sur la configuration d'un environnement virtuel dans Anaconda pour tensorflow.

1.-Dans le terminal de commande qui apparaît, créez et activer un nouvel environnement virtuel appelé «tensorflow1»

- *conda create -n tensorflow1*
- *activate tensorflow1*

2.- Installez quelques packages nécessaires pour l'expérimentation

- `pip install tensorflow==1.15`
- `pip install pillow`
- `pip install matplotlib`
- `pip install opencv-python`

3.-Installer le protocol buffers 3.4.0, copier et coller le fichier **protoc** qui se trouve dans le fichier /bin, met le dans notre le fichier /research qui se trouve dans notre model

- `cd C:\tensorflow\models\research`
- `protoc object_detection/protos/*.proto --python_out=.`
- `Python setup.py build`
- `Python setup.py install`

This PC > Windows (C:) > model-detection-voiture > research			
Name	Date modified	Type	Size
a3c_blogpost	9/1/2020 11:32 PM	File folder	
adversarial_text	9/1/2020 11:32 PM	File folder	
attention_ocr	9/1/2020 11:32 PM	File folder	
audioset	9/1/2020 11:32 PM	File folder	
autoaugment	9/1/2020 11:32 PM	File folder	
build	9/3/2020 3:25 AM	File folder	
cognitive_planning	9/1/2020 11:32 PM	File folder	
cvt_text	9/1/2020 11:32 PM	File folder	
deep_speech	9/1/2020 11:32 PM	File folder	
deeplab	9/1/2020 11:32 PM	File folder	
delf	9/1/2020 11:32 PM	File folder	
dist	9/3/2020 3:25 AM	File folder	
efficient-hrl	9/1/2020 11:32 PM	File folder	
lfads	9/1/2020 11:32 PM	File folder	
lstm_object_detection	9/1/2020 11:32 PM	File folder	
marco	9/1/2020 11:32 PM	File folder	
nst_blogpost	9/1/2020 11:32 PM	File folder	
object_detection	9/3/2020 9:06 PM	File folder	
object_detection.egg-info	9/3/2020 3:25 AM	File folder	
pcl_rl	9/1/2020 11:32 PM	File folder	
rebar	9/1/2020 11:32 PM	File folder	
sequence_projection	9/1/2020 11:32 PM	File folder	
slim	9/1/2020 11:32 PM	File folder	
vid2depth	9/1/2020 11:32 PM	File folder	
<input checked="" type="checkbox"/> protoc	8/14/2017 3:24 PM	Application	4,315 KB
README.md	9/1/2020 11:32 PM	MD File	7 KB

Figure: Fichier de notre model et le setup app Protoc

Partie Expérimentation et Résultat Obtenus

1.-Evaluation de notre système détection des pièces voiture

L'évaluation des modules de détection et reconnaissance est effectuée sur un ensemble de 300 images de voiture contenant au total 240 pour l'entraînement et 60 pour le test.

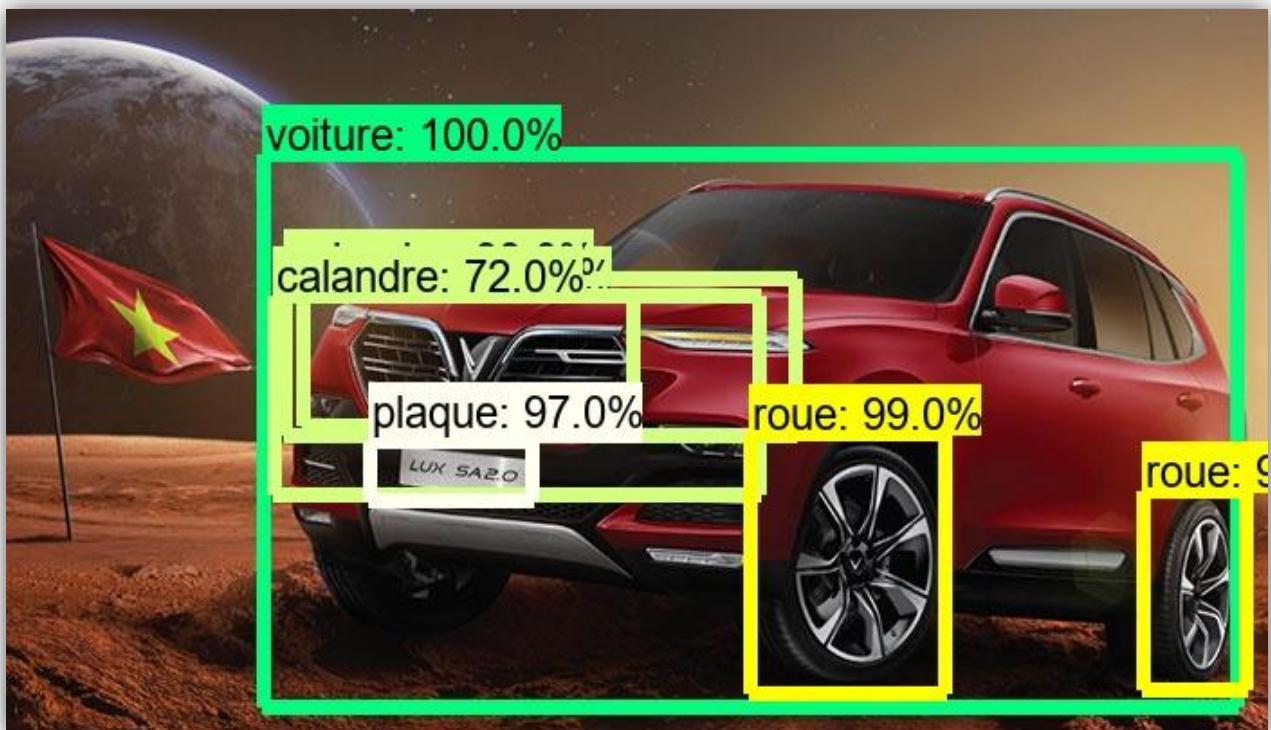
1.1.- Expérimentation avec des images de voiture

Nous avons fait le test avec quelques photos que nous avions pris dans l'université et quelqu'une que nous avions trouvé sur Google pour tester notre système

TEST 1

Les résultats de cette évaluation sont résumés dans les tableaux sous dessous:

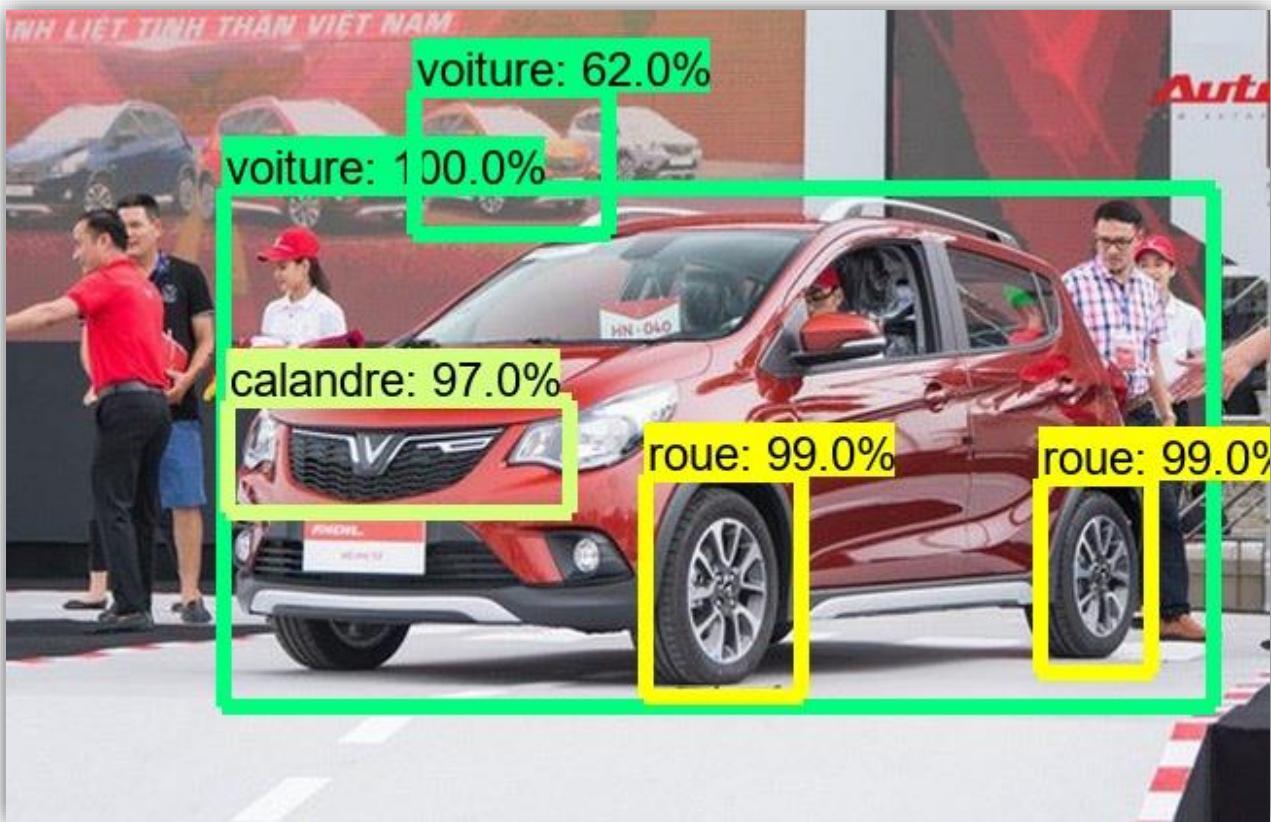
➤ *python Object_detection_image.py*



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	99%	72%	97

TEST 2

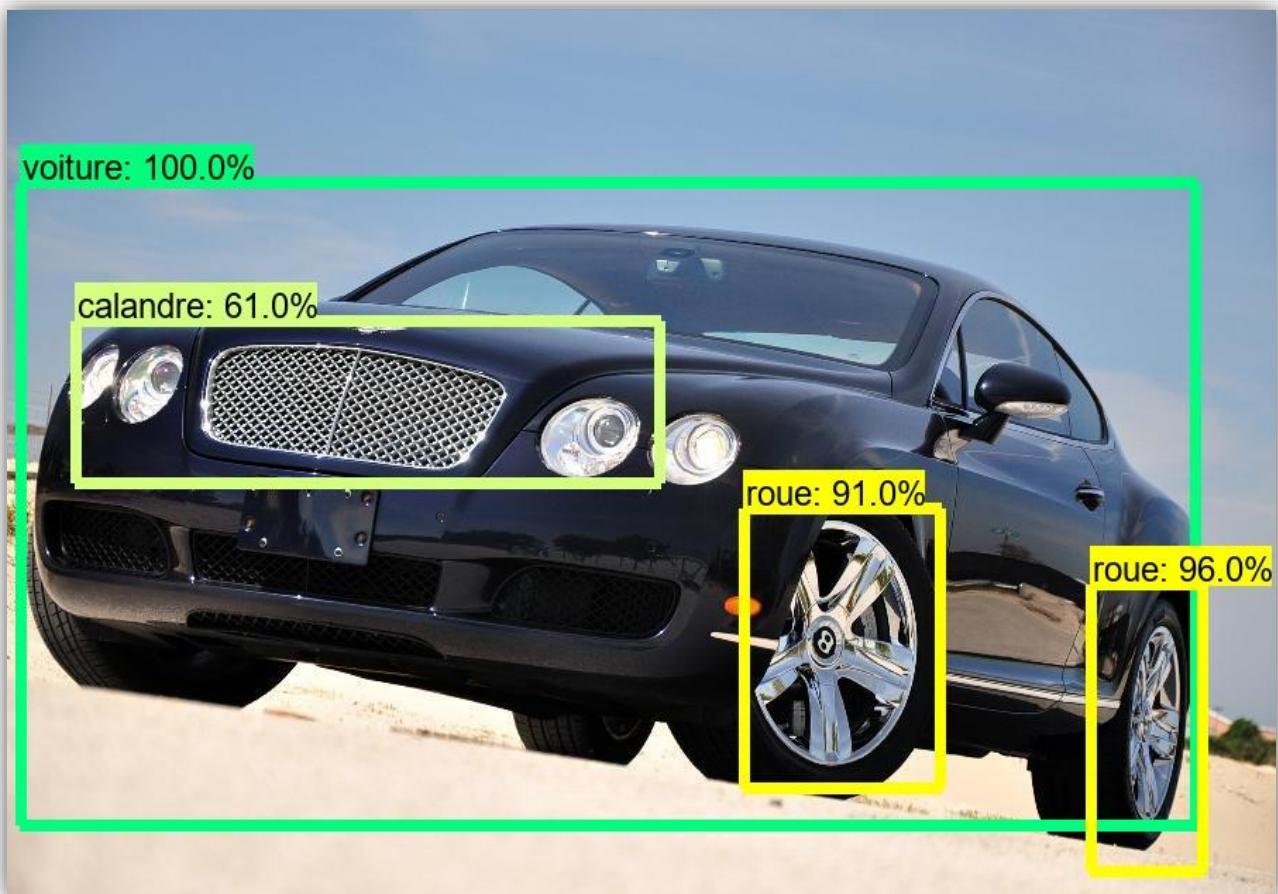
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	99%	97%	-

TEST 3

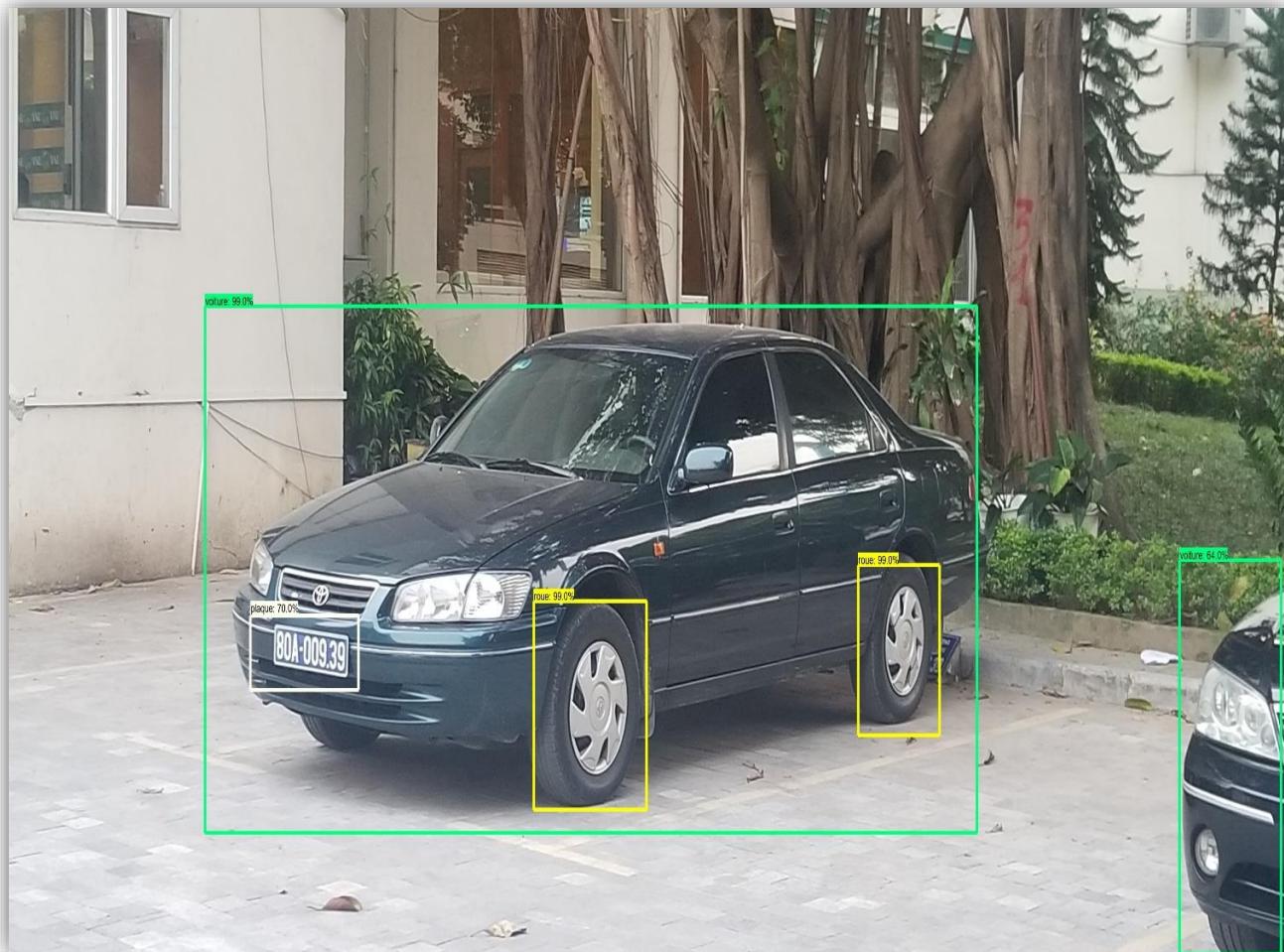
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	91 & 96%	61%	-

TEST 4

Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99%	99%	-	70%

TEST 5

Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99%	-	-	97%

TEST 6

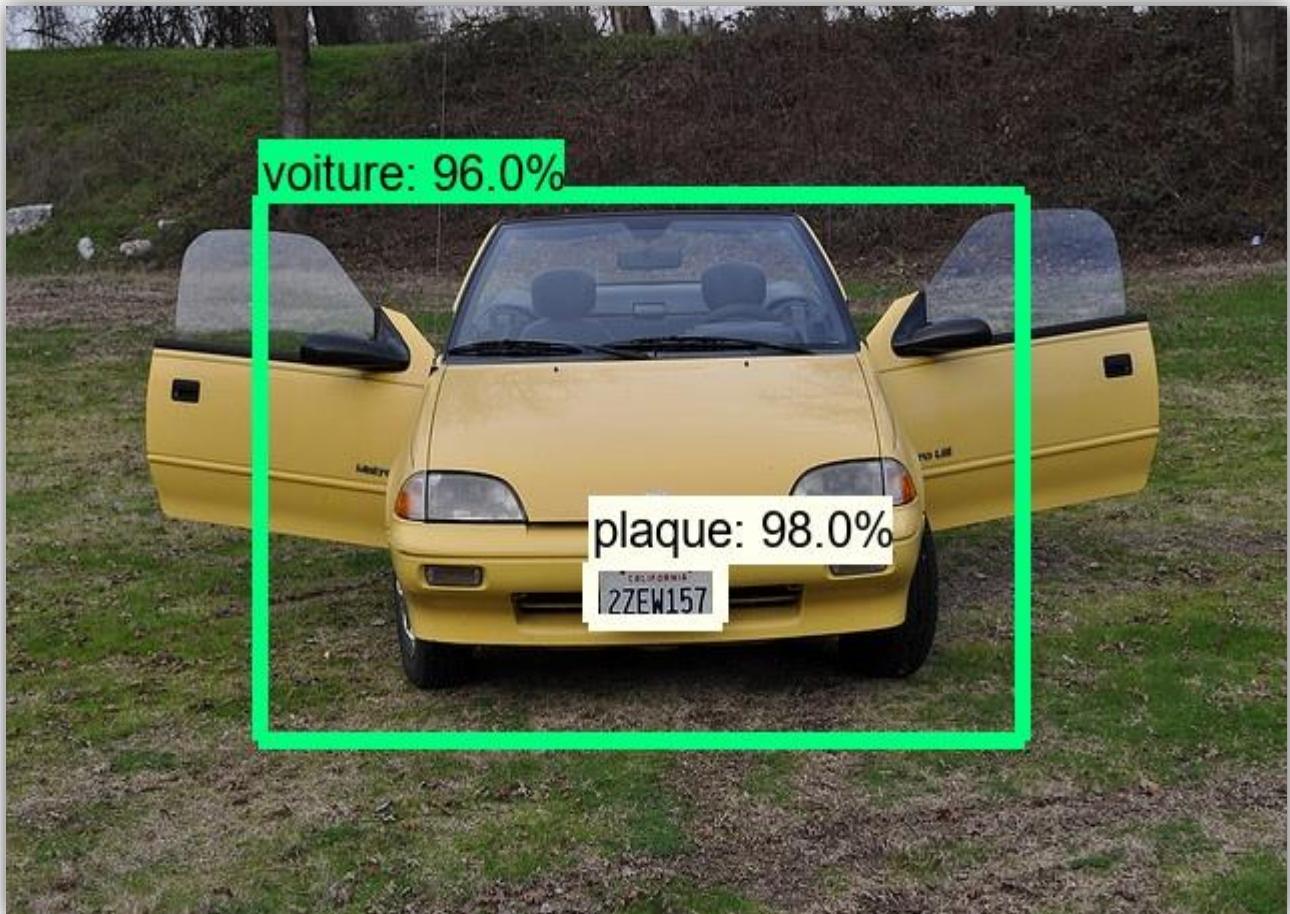
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99	96	9510	77

TEST 7

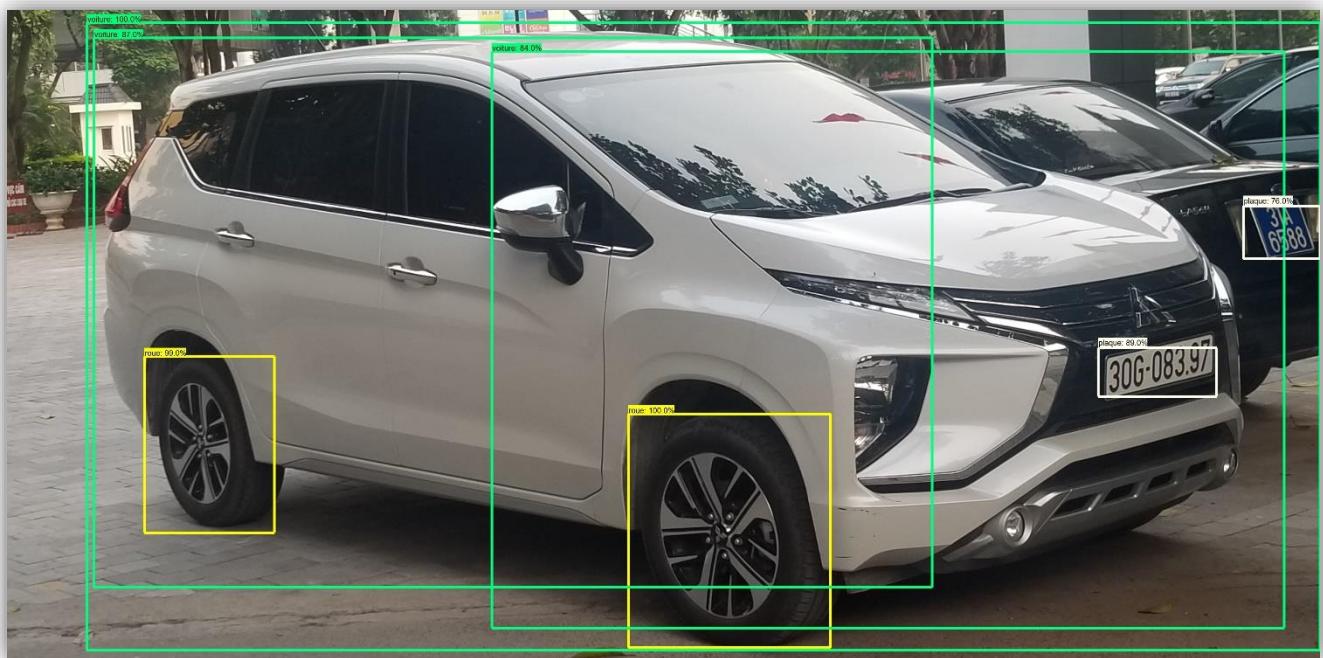
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	96%	-	-	98%

TEST 8

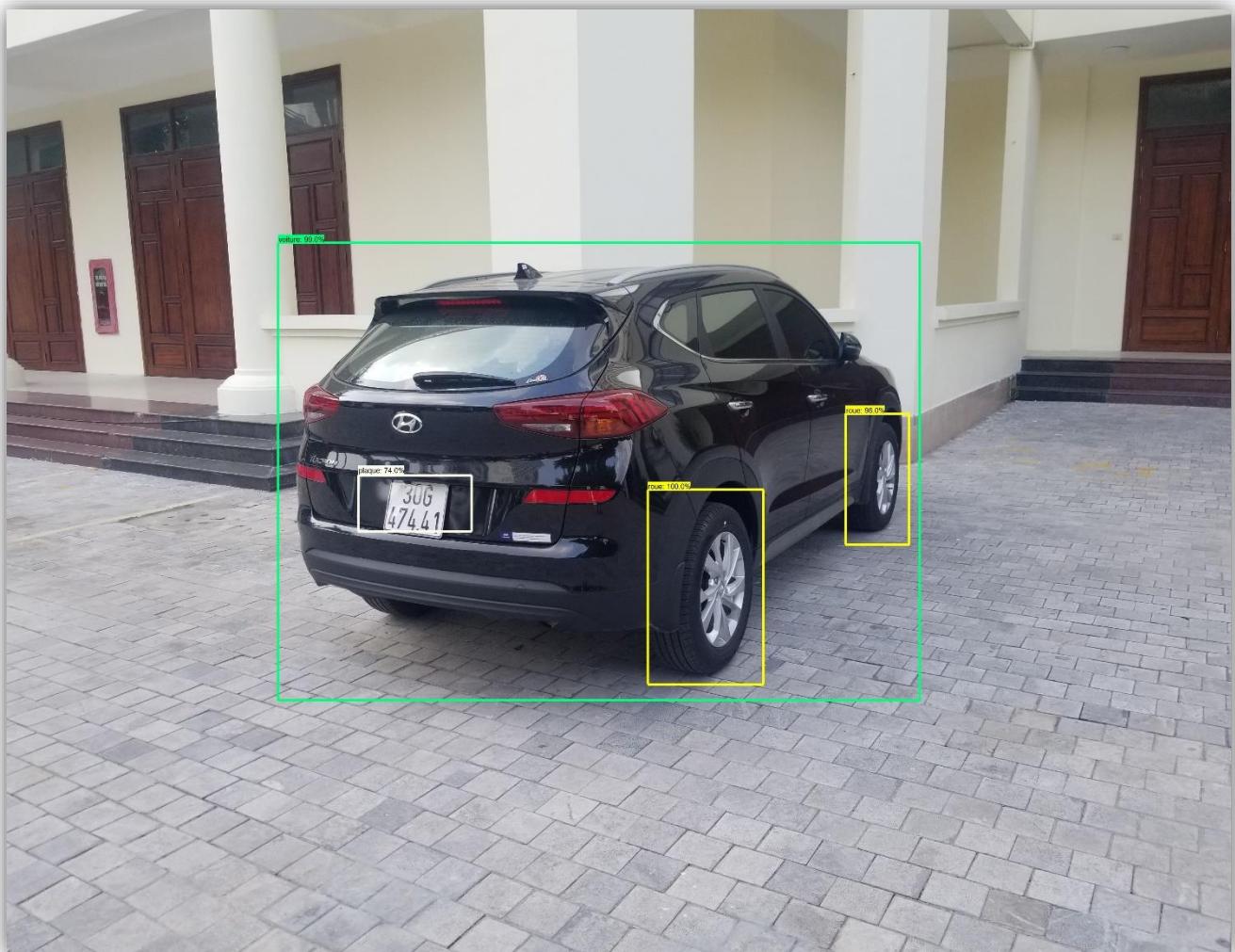
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	100%	-	89%

TEST 9

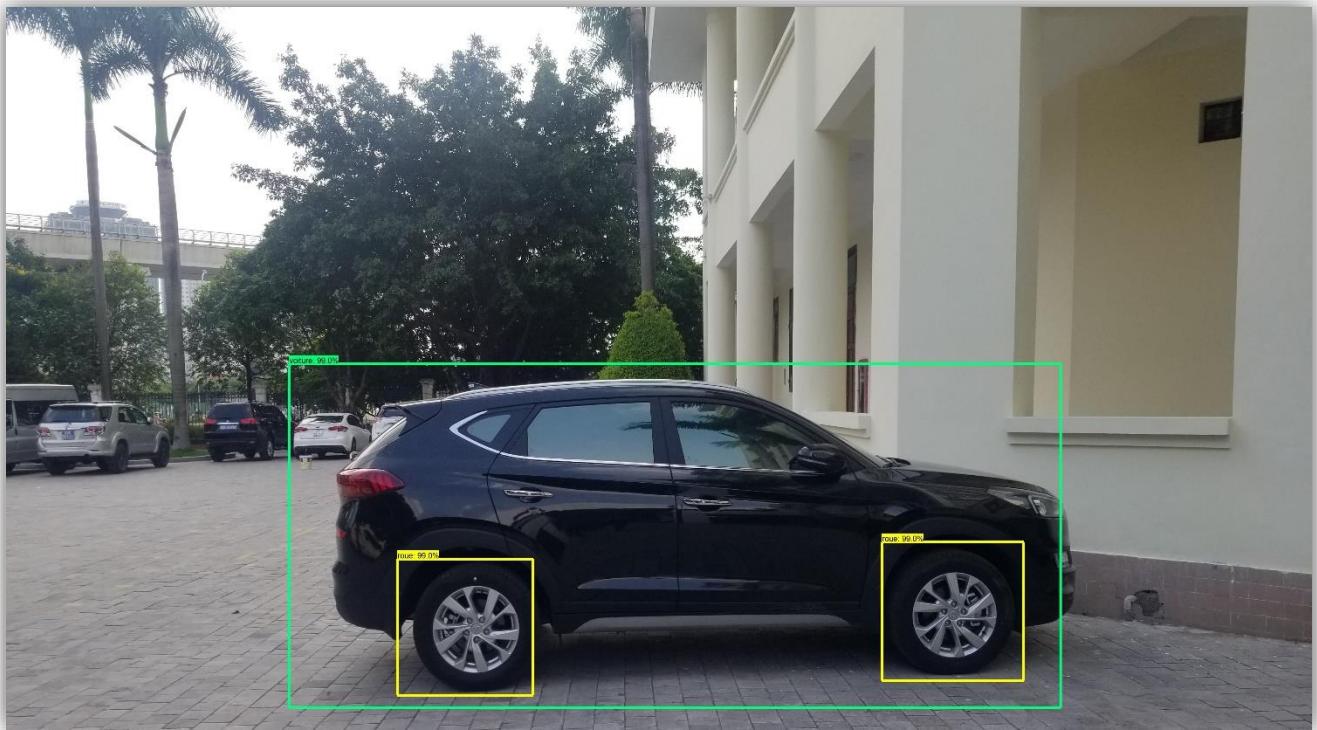
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99%	100%	-	74%

TEST 10

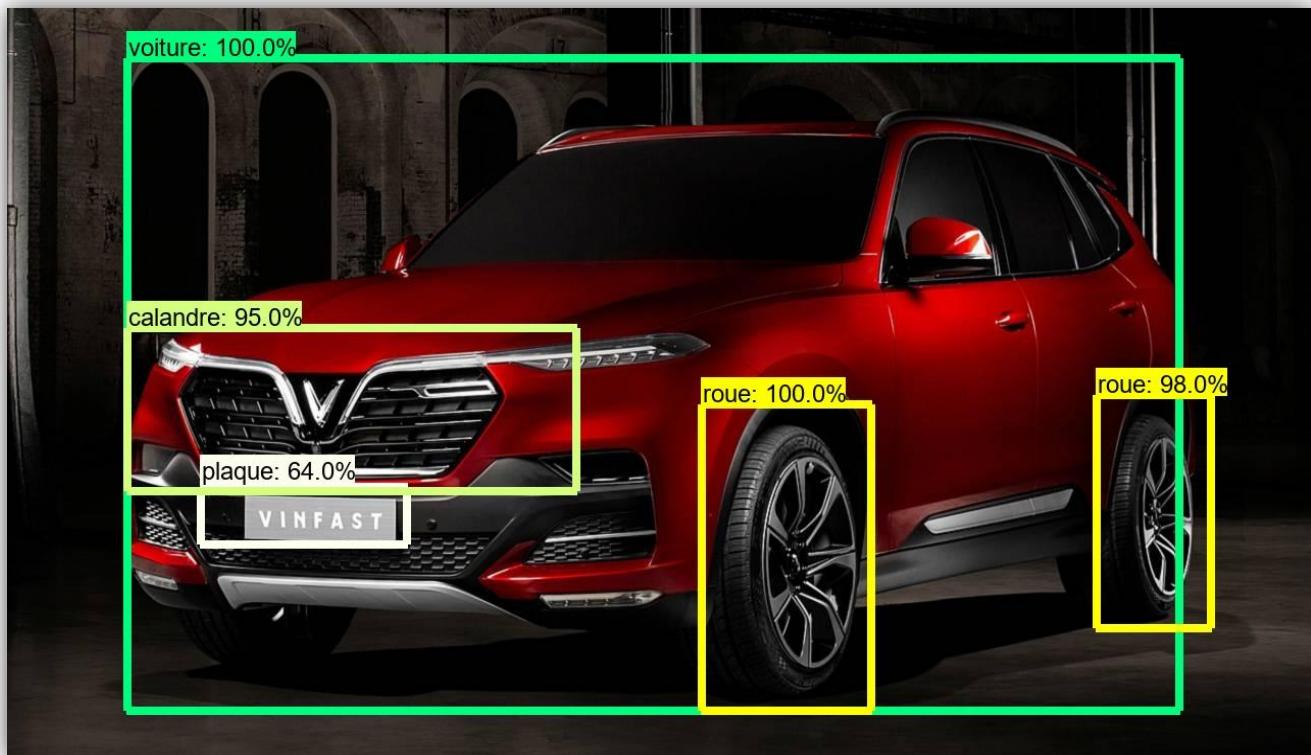
Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99%	99%	-	-

TEST 11

Les résultats de cette évaluation sont résumés dans les tableaux suivants:



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	100 & 98%	95%	64%

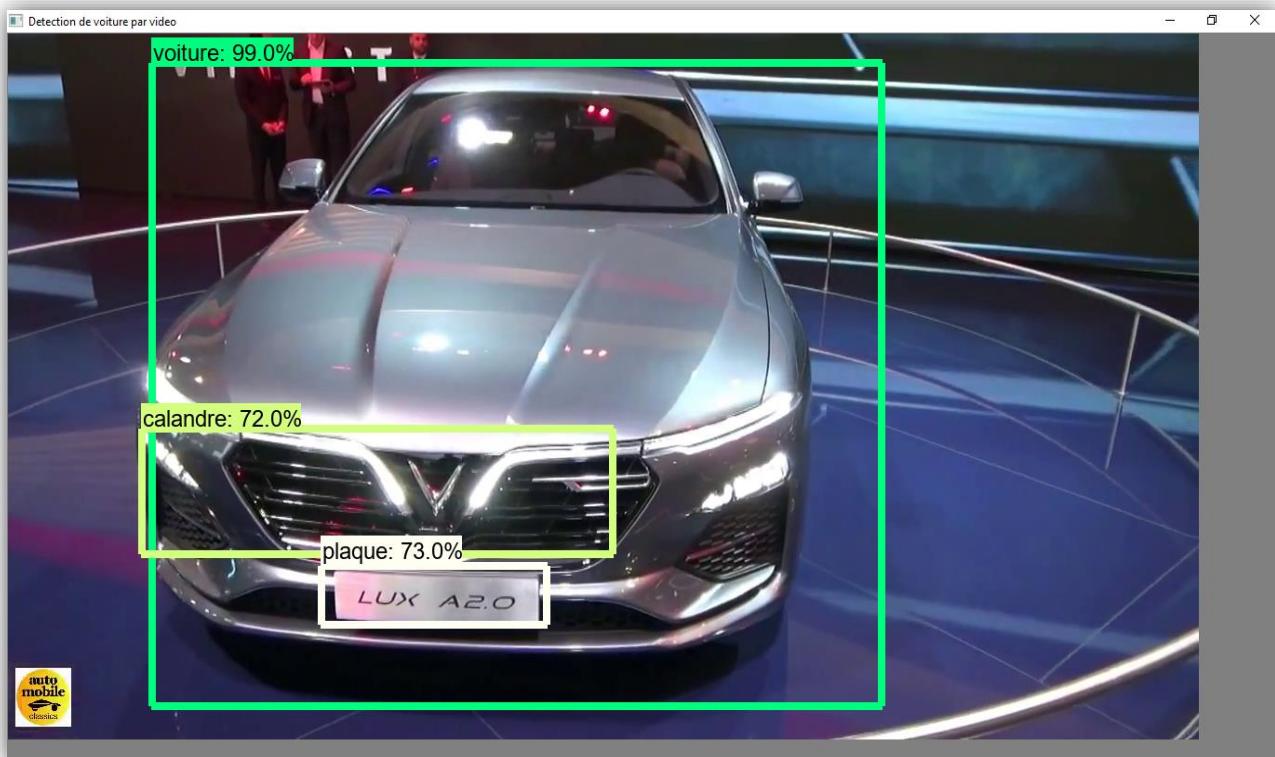
1.2.- Expérimentation avec une vidéo de voiture

Nous avons fait ce test avec une vidéo que nous avons téléchargé sur YouTube

Lien : <https://www.youtube.com/watch?v=3Ymx2BQ6WDk>

➤ *python Object_detection_video.py*

Test 12



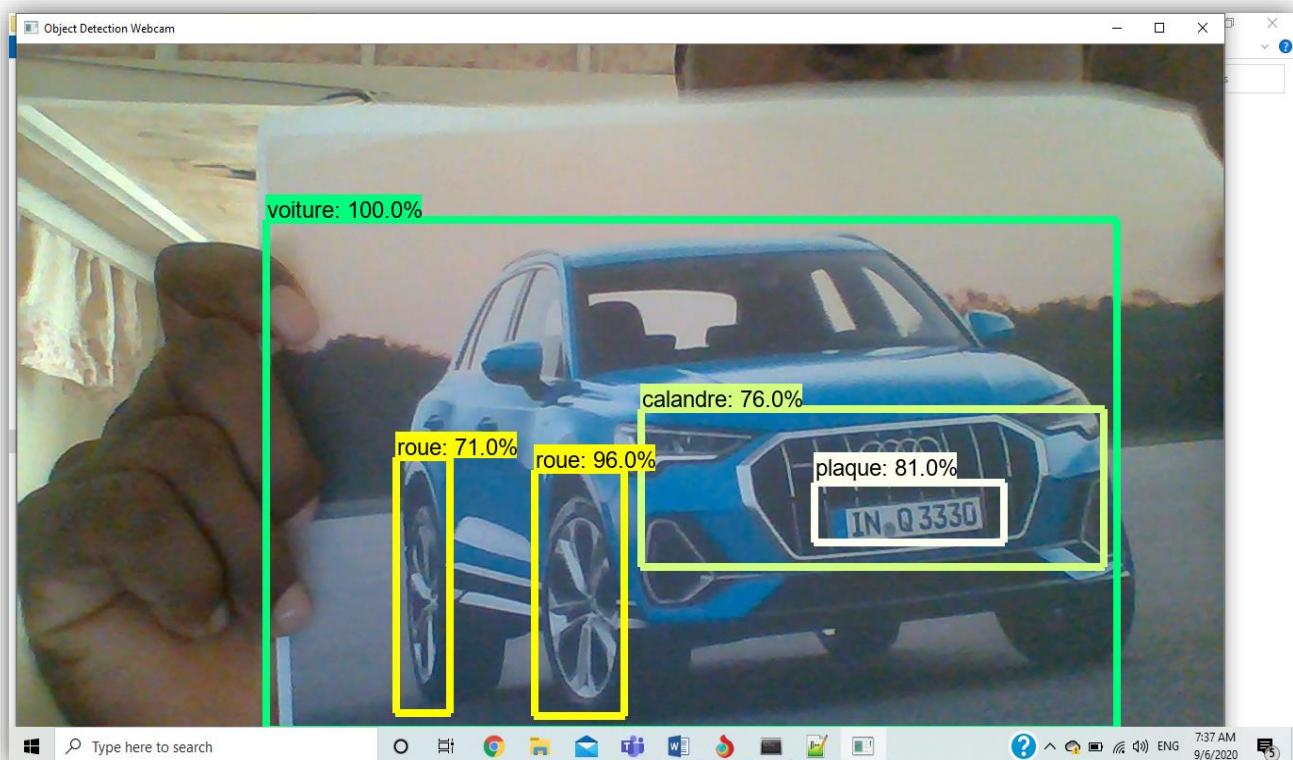
	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	99%	-	72%	73%

1.3.- Expérimentation avec le Webcam

Pour l'expérimentation avec la webcam, nous avons fait le test avec une photo de voiture devant webcam, et le système a pu détecter les différentes parties de la voiture

➤ `python Object_detection_webcam.py`

Test 13



	Voiture	Roue	Calandre	Plaque
Pourcentage de détection	100%	96%	76%	81%

Conclusion et Perspectives

Entraîner des modèles pour qu'ils aient des capacités semblables à celle de l'homme requiert énormément de ressources en termes de données et de temps. Pour optimiser cet apprentissage, il faut mutualiser les connaissances d'un modèle à l'autre en pratiquant le Transfer Learning. Principalement, dans les cas de traitement d'image, on réutilise les couches d'un modèle qui a déjà appris que l'on fixe, les dernières couches quant à elle se modifieront en fonction des données d'apprentissage et s'affineront en fonction des données en input.

Toutefois quelques améliorations peuvent être apportées à ce travail dans le futur :

- Continue avec l'implémentation de l'application mobile avec BAZEL pour l'extraction du fichier .apk ou en convertir le model tensorflow en tensorflow lite.
- Intégrer quelques classes pour détecter plus partie d'une voiture dans l'application
- Entrainer le model avec tous les images du dataset

Dans ce rapport, nous avons exposé les étapes de conception et de développement de notre application qui consiste à créer une application pour la détection et reconnaissance des pièces de voiture.

Notre travail s'est déroulé sur différents étapes. Nous avons commencée par une étude de l'existant, suivie de la proposition d'une solution adéquate.

Ce projet se situe en effet dans le cadre du projet de fin de la maîtrise 1 du cycle ingénieur en informatique. Ce projet était une véritable expérience de travail en collaboration, qui m'a permis de bien renforcer mes connaissances l'esprit de recherche ainsi que la qualité de mon travail.

Dans la dernière phase, nous avons évoqué les différentes technologies utilisées ainsi qu'implémentation du système.

Références Scientifiques

- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012
- Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
- <https://blog.tensorflow.org/2020/03/introducing-model-garden-for-tensorflow-2.html?m=1>
- <https://medium.com/analytics-vidhya/classification-of-images-based-on-fashion-mnist-dataset-bb11e4bcdefb>
- <https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-satellite-photos-of-the-amazon-rainforest/>
- <https://towardsdatascience.com/fastai-image-classification-32d626da20>
- <https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9>
- <https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c>
- <https://cognitiveclass.ai/courses/deep-learning-tensorflow>
- <https://www.udemy.com/course/le-deep-learning-de-a-a-z/lecture/8859522?start=0#overview>
- <https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/5869331-decouvrez-le-domaine-de-la-data-science>
- https://fr.wikipedia.org/wiki/D%C3%A9tection_d%27objet
- https://fr.wikipedia.org/wiki/Vision_par_ordinateur
- http://www.cogefa.com/resultat_par véhicule.html?rech=par ref&surtitre=&modele=&id_division=&id_rubrique=
- https://www.researchgate.net/publication/37796797_Reconnaissance_par_vision_du_type_d'un véhicule_automobile

- https://www.francetvinfo.fr/replay-radio/l-auto/penurie-de-pieces-detachees-sur-les-voitures-un-peu-agees_1762997.html
- <https://www.capital.fr/entreprises-marches/pieces-detachees-le-nouveau-logiciel-de-renault-cree-denormes-retards-1350516>
- <https://towardsdatascience.com/how-to-apply-machine-learning-ml-in-an-android-app-33e848c0dde6>
Consulté le 03-01-2020.
- https://www.hevs.ch/media/document/3/schule_vincent_master.pdf
- <https://www-master.ufr-info-p6.jussieu.fr/2015/Deep-learning-pour-la,29042>
- <http://dspace.univ-tlemcen.dz/bitstream/112/12583/1/Deep-Learning-pour-la-classification..pdf>
- <http://193.194.80.11/jspui/bitstream/123456789/2997/1/Doc3.pdf>
Consulté le 13-02-2020.
- <https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/>
Consulté le 03-01-2020.
- <http://di.univ-blida.dz:8080/jspui/handle/123456789/2023>
Consulté le 21-01-2020.
- <https://www.theses.fr/1998INPG0110>
Consulté le 13-01-2020.
- <https://rc.library.uta.edu/uta-ir/handle/10106/27411>
Consulté le 19-01-2020.
- <https://towardsdatascience.com/object-detection-using-deep-learning-approaches-an-end-to-end-theoretical-perspective-4ca27eee8a9a>
Consulté le 13-01-2020.
- <https://cv-tricks.com/artificial-intelligence/object-detection-using-deep-learning-for-advanced-users-part-1/>
Consulté le 13-03-2020.
- <https://arxiv.org/ftp/arxiv/papers/1804/1804.00429.pdf>
Consulté le 27-01-2020.
- https://www.researchgate.net/publication/323796590_Real-time_imagebased_parking_occupancy_detection_using_deep_learnin
Consulté le 13-01-2020.
- <https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/>
Consulté le 15-02-2020.

- <http://www.iri.upc.edu/files/academic/thesis/98-Research-Plan.pdf>
Consulté le 24-01-2020.
- <https://uk.mathworks.com/help/vision/object-detection-using-deep-learning.html>
Consulté le 03-03-2020.
- https://www.optimalplus.com/wp-content/uploads/2020/01/WhitePaper_APEX_DefectImgClass_US_11jan19_3_WEB-2.pdf
Consulté le 07-01-2020.
- <https://dzone.com/articles/beginners-guide-image-recognition-and-deep-learnin>
Consulté le 21-03-2020.
- <https://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf>
Consulté le 23-02-2020.
- <https://missinglink.ai/guides/tensorflow/tensorflow-image-recognition-object-detection-api-two-quick-tutorials/>
Consulté le 17-03-2020.
- <https://nowpublishers.com/article/Details/SIG-071>
Consulté le 18-03-2020.
- https://www.hevs.ch/media/document/3/schule_vincent_master.pdf
Consulté le 07-03-2020.
- <https://tel.archives-ouvertes.fr/tel-01808884/document>
Consulté le 03-03-2020.
- <https://www.aclweb.org/anthology/P05-2008.pdf>
Consulté le 03-04-2020.

Lien :

<https://github.com/IngH2020/Detection-et-reconnaissance-des-pieces-de-voiture-avec-Deep-Learning/upload/master>