الجمهورية الجزائرية الديمقراطية الشعبية

Republique Algerienne Democratique et Populaire

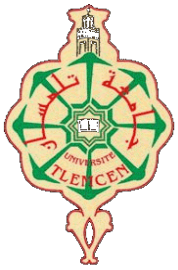
وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة أبو بكر بلقايد - تلمسان -

Université Aboubakr Belkaïd – Tlemcen –

Faculté de Technologie



MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Automatique

Spécialité : Automatique et Informatique Industrielle

Par : MENOUAR Roumaissa & MADAOUI Fatma

Sujet

Reconnaissance d'objets par réseau convolutif dans un environnement encombré

Soutenu publiquement, le 23/06/2025, devant le jury composé de :

Mme. BENKHALED Mounira	MCA	Université de Tlemcen	Présidente
Melle. HANDOUZI Wahida	MCA	Université de Tlemcen	Examinatrice
Mme. BELDJILALI Nawel	MAA	Université de Tlemcen	Encadrante

Année universitaire : 2024/2025

Dédicace

Je tiens, à travers ces quelques lignes, à exprimer ma profonde gratitude à ceux qui ont laissé une empreinte dans mon parcours :

J'adresse mes remerciements les plus sincères à ma chère mère, pilier de ma vie et source inépuisable de motivation et de prières. Son encouragement constant, ses invocations et sa présence rassurante ont été un véritable soutien dans les moments les plus difficiles. Je lui dédie ces lignes avec tout mon amour et ma profonde reconnaissance.

Ma gratitude va également à mon père pour ses efforts, son travail, son dévouement et sa patience. Son soutien indéfectible a été essentiel tout au long de mon cheminement et m'a permis d'atteindre ce stade.

Je remercie aussi mes frères du fond du cœur. Je leur souhaite sincèrement réussite et bonheur dans leurs vies personnelles et professionnelles.

Enfin, à mes amis les plus proches, que je remercie profondément pour tous les moments heureux partagés. Leur soutien et leur fidélité ont été d'une grande valeur tout au long de cette aventure.

MENOUAR Roumaissa

Dédicace

À ceux dont la présence dans ma vie est une bénédiction inestimable...
À ceux dont les prières étaient le secret de la réussite,
et dont la patience a été le carburant de mon parcours...
À ces deux cœurs qui m'ont entourée d'amour avant même que je comprenne le sens de la vie...
À ceux dont Allah a fait de leur satisfaction une voie vers Son agrément,
à ceux qu'Il a recommandés dans Son Livre, et dont la piété est un acte d'adoration :
à ma mère et mon père.
Ma chère maman,
source inépuisable de tendresse,
prière sincère qui m'accompagne en silence,
chaleur de mes jours, sérénité de mon cœur…
toi qui as été pour moi un modèle de patience et un exemple de foi,
combien tu as souffert pour moi, combien tu as pleuré en secret, combien tu as prié au cœur de la nuit…
Tu es mon paradis sur cette terre, mon bonheur,
tu m'as soutenue à chaque pas, encouragée dans mes échecs,
tu as pansé mes blessures lorsque je suis tombée, et tu m'as portée…
tu as souri malgré la douleur, tu as tout donné pour que j'arrive ici…
Je t'aime, ma chérie, mon monde,
chaque mot dans ce travail est le prolongement de ta tendresse.
Et à mon cher père,
mon premier soutien, mon pilier inébranlable,
toi qui as semé en moi la force et la volonté,
toi qui as planté en mon cœur la fermeté et dans mon esprit la sagesse,
combien de fois as-tu caché ta fatigue pour que je ne faiblisse pas,
combien de fois as-tu souri malgré la douleur pour que je ne perde pas espoir...
toi qui as prié pour moi dans le silence des nuits,
et qui as été fier de moi au grand jour...
Je te dois toute ma gratitude,
et tout l'orgueil que je ressens vient de toi.
Je vous dédie cet accomplissement et ce travail modeste,
car il est le fruit de vos efforts et une goutte dans l'océan de votre générosité,
et c'est le moins que je puisse offrir en reconnaissance d'un mérite que je ne pourrai

jamais rembourser, même avec toute une vie.
Si je me prosternai toute ma vie, je ne vous rendrais pas justice.
Je demande à Allah le Très-Haut de vous récompenser pour tout ce que vous avez fait pour moi,
d'élever vos rangs dans ce monde et dans l'au-delà,
de vous accorder santé, bien-être et une longue vie dans Son obéissance,
et de faire de moi une source de votre satisfaction, comme vous l'avez été pour mon bonheur,
et qu'Il fasse de moi la prunelle de vos yeux tant que je vivrai.
À celui que mon cœur a choisi pour partager ma route,
à celui qu'Allah m'a uni par un lien sacré,
à mon cher époux,
toi qui as été mon appui dans les moments de défi,
et la lumière qui m'a guidée dans les dernières étapes de cette réussite...
Je te remercie du fond du cœur,
pour ta patience, ton soutien…
Je t'aime.
À mes frères et à ma chère famille,
vous qui avez été ce cocon chaleureux, cet abri sûr,
vous qui avez partagé mes jours doux et amers,
et dont les prières ont alimenté ma volonté...
Et à mon amie Iman,
complice de cœur et d'âme,
merci pour ta sincérité, ta fidélité et ta présence constante dans mes derniers instants…
tu as été la lumière dans mes jours sombres, et le baume lorsque la fatigue m'accablait…
Je vous dédie cette réussite, car elle n'est pas uniquement la mienne,
elle est le fruit de votre amour, de votre soutien et de votre présence sincère à mes côtés…
Je demande à Allah de vous récompenser généreusement,
et de remplir vos cœurs de bonheur, comme vous avez rempli le mien de sérénité.

MADAOUI Fatma

Remerciements

Avant tout, nous tenons à exprimer notre gratitude personnelle et profonde, car ce travail représente l'aboutissement d'un long parcours que nous avons mené avec beaucoup d'efforts, d'engagement et de persévérance. Il est le fruit d'un investissement constant, motivé par notre volonté d'apprendre, de progresser et de nous dépasser.

Nous remercions chaleureusement notre encadrante, Mme Beldjilali Nawel, pour sa disponibilité, sa patience, ses orientations précieuses et le temps qu'elle a consacré à nous accompagner tout au long de ce mémoire. Son encadrement attentif et rigoureux a été pour nous un appui fondamental.

Nous adressons également nos remerciements à toute personne ayant pris le temps de lire ce mémoire. Vos retours, votre attention et votre intérêt sont pour nous d'une grande valeur.

Nos remerciements les plus sincères vont également à Mme Benkhaled Mounira pour l'honneur qu'elle nous a fait en acceptant de présider cette soutenance, ainsi qu'à Mme Handouzi Wahida, pour l'attention portée à notre travail et ses remarques pertinentes.

Enfin, nous remercions toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire. Une reconnaissance particulière va à l'ensemble de nos enseignants, tout au long de notre parcours universitaire, qui, par la richesse de leur savoir et leur implication, ont nourri notre réflexion et posé les fondements solides de cette formation. Enfin, un énorme merci au département du Génie Electrique et Electronique, à son chef Mr Boumediene .A et à tout son personnel.

Résumé :

Ce projet s’inscrit dans le cadre de l’intelligence artificielle appliquée à la vision par ordinateur, visant la reconnaissance d'objets par réseau de neurones convolutif dans un environnement encombré. Il a pour but de développer un système capable de réaliser une analyse d’images fine et une identification précise des objets, même en présence de bruits visuels. L’intelligence artificielle, notamment l’apprentissage automatique et les réseaux de neurones convolutifs (CNN), joue un rôle fondamental pour garantir une interprétation contextuelle fiable. Ce travail est basé sur un modèle pré entraîné SSD MobileNet V3. Ce projet valorise les concepts de perception artificielle, de classification visuelle, et d'interaction homme-machine (interface utilisateur GUI), intégrés dans un processus de traitement automatisé de l’information visuelle. Enfin, le projet souligne l’importance du potentiel de la vision artificielle pour répondre à des problématiques concrètes, tout en assurant performance, rapidité et précision sur des machines moins puissantes.

Mots-clés: Intelligence artificielle, reconnaissance d’objets, vision par ordinateur, apprentissage automatique, réseau de neurones convolutifs (CNN), environnement encombré, identification visuelle, bruits, SSD MobileNet V3, classification visuelle, interaction homme-machine, interface utilisateur (GUI), vision artificielle, performance, rapidité et précision.

Abstract:

This project fits within the framework of artificial intelligence applied to computer vision, aiming at object recognition by convolutional neural network in a cluttered environment. Its goal is to develop a system capable of performing fine image analysis and precise identification of objects, even in the presence of visual noise. Artificial intelligence, notably machine learning and convolutional neural networks (CNN), plays a fundamental role to guarantee reliable contextual interpretation. This work is based on a pre-trained SSD MobileNet V3 model. This project highlights the concepts of artificial perception, visual classification, and human-machine interaction (GUI user interface), integrated into an automated visual information processing workflow. Finally, the project emphasizes the importance of the potential of computer vision to address concrete issues, while ensuring performance, speed, and precision on less powerful machines.

Keywords: Artificial intelligence, object recognition, computer vision, machine learning, convolutional neural network (CNN), cluttered environment, visual identification, noise, SSD MobileNet V3, visual classification, human-machine interaction, user interface (GUI), computer vision, performance, speed, and precision.

ملخص:

يندرج هذا المشروع في إطار الذكاء الاصطناعي المطبّق على الرؤية بالحاسوب، ويهدف إلى التعرف على الأجسام بواسطة شبكة عصبية التفافية في بيئة مزدحمة. يهدف إلى تطوير نظام قادر على إجراء تحليل دقيق للصور وتحديد دقيق للأجسام، حتى في وجود ضوضاء بصرية. يلعب الذكاء الاصطناعي، ولا سيما التعلم الآلي والشبكات العصبية الالتفافية(CNN) ، دورًا أساسيًا لضمان تفسير سياقي موثوق. يعتمد هذا العمل على نموذج مُدرب مسبقًا (SSD MobileNet V3) .يبرز هذا المشروع مفاهيم الإدراك الاصطناعي، والتصنيف البصري، والتفاعل بين الإنسان والآلة واجهة المستخدم(GUI)، والمندمجة في عملية معالجة آلية للمعلومات البصرية. وأخيرًا، يسلط المشروع الضوء على أهمية إمكانات الرؤية الاصطناعية في الاستجابة لمشكلات ملموسة، مع ضمان الأداء والسرعة والدقة على أجهزة الأقل قوة.

الكلمات المفتاحية: الذكاء الاصطناعي، التعرف على الأجسام، الرؤية بالحاسوب، التعلم الآلي، الشبكة العصبية الالتفافية (CNN)، بيئة مزدحمة، التحديد البصري، الضوضاء، SSD MobileNet V3، التصنيف البصري، التفاعل بين الإنسان والآلة، واجهة المستخدم(GUI) ، الرؤية الاصطناعية، الأداء، السرعة، الدقة.

Table des matières

Table des figures

Liste des abréviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
SSD	Single Shot MultiBox Detector
GUI	Graphical User Interface
IoU	Intersection over Union
NMS	Non-Maximum Suppression
COCO	Common Objects in Context (dataset)
RGB	Red Green Blue
API	Application Programming Interface
GTTS	Google Text-to-Speech
PIL	Python Imaging Library
CTK	CustomTkinter
FPS	Frames Per Second
ReLU	Rectified Linear Unit
HOG	Histogram of Oriented Gradients
DNN	Deep Neural Network

Introduction Générale

La reconnaissance d'objets est aujourd'hui une composante essentielle dans les applications modernes de la vision par ordinateur, notamment dans des domaines tels que la sécurité, la conduite autonome, la robotique ou encore l'accessibilité. Dans un environnement encombré, les objets peuvent être partiellement masqués, mal éclairés ou superposés, ce qui complique leur identification. Les approches classiques, reposant sur des descripteurs manuels ou des modèles statistiques, se révèlent souvent inefficaces dans ce contexte.

L'essor de l'intelligence artificielle, et plus particulièrement de l'apprentissage profond, a ouvert la voie à des méthodes plus robustes. Les réseaux de neurones convolutifs (CNN) sont aujourd'hui les modèles de référence pour l'analyse d'images. En exploitant ces modèles, il est possible d'extraire automatiquement des caractéristiques visuelles pertinentes pour détecter et classifier les objets avec précision.

Ce mémoire s'inscrit dans cette dynamique en explorant l'utilisation du modèle SSD MobileNetV3, un modèle convolutif préentraîné, pour réaliser la reconnaissance d'objets dans des images contenant du bruit visuel. Ce modèle est particulièrement adapté aux environnements à faibles ressources grâce à sa légèreté et sa rapidité d'exécution.

Problématique :

Comment développer un système de reconnaissance d'objets efficace dans un environnement encombré en utilisant un modèle préentraîné léger tel que SSD MobileNetV3, tout en garantissant des performances satisfaisantes sur des machines peu puissantes ?

Objectifs du travail :

• Mettre en place un système de détection d'objets fonctionnel, rapide et robuste.

• Utiliser un modèle préentraîné pour réduire le coût d'entraînement.

• Proposer une interface utilisateur intuitive, avec des options de synthèse vocale et d'analyse d'image.

• Évaluer la performance du système sur des images variées.

Méthodologie adoptée :

Une approche intégrée a été choisie, combinant :

• la préparation de données (prétraitement, histogrammes).

• l'utilisation du modèle SSD MobileNetV3 préentraîné.

• le développement d'une interface graphique avec reconnaissance vocale.

• et l'analyse des résultats obtenus.

Plan du mémoire :

Ce travail est structuré en cinq parties :

• Le chapitre 1 présente les notions générales liées à la vision par ordinateur, les modèles CNN et SSD MobileNetV3, les métriques d'évaluation et une revue rapide des travaux similaires.

• Le chapitre 2 décrit l'approche méthodologique et les détails de mise en œuvre.

• Le chapitre 3 est consacré aux résultats expérimentaux et à leur analyse.

• Enfin, la conclusion générale propose un récapitulatif des apports et des perspectives d'amélioration.

Chapitre I

Généralités et état de l'art

I.1 Introduction

Dans le domaine de la vision par ordinateur, la reconnaissance d'objets est devenue une tâche essentielle, notamment dans des applications comme la vidéosurveillance, la conduite autonome ou encore l'imagerie médicale. Cette tâche consiste à localiser et à identifier les objets présents dans une image ou une vidéo.

Cependant, dans un environnement visuellement encombré, cette reconnaissance devient plus complexe. En effet, la présence d'objets partiellement cachés, d'arrière-plans désordonnés, de variations d'éclairage ou encore de bruit visuel rend les méthodes classiques peu efficaces. Avec l'évolution de l'intelligence artificielle, et en particulier de l'apprentissage profond (Deep Learning), de nouveaux outils ont été développés pour surmonter ces difficultés. Parmi ces outils, les réseaux de neurones convolutifs (CNN) se sont imposés comme des solutions performantes pour analyser et comprendre le contenu d'une image.

Ce chapitre présente le cadre général de notre projet. Il s'agit d'introduire les concepts clés de l'intelligence artificielle appliquée à la vision par ordinateur, de présenter les modèles de détection d'objets les plus utilisés, et d'expliquer notre choix du modèle SSD MobileNetV3. [1]

I.2 Intelligence artificielle et vision par ordinateur

L'intelligence artificielle (IA) regroupe un ensemble de techniques qui permettent à une machine de simuler certains aspects de l'intelligence humaine, comme apprendre, raisonner ou prendre des décisions. Dans le domaine de l'image, l'IA permet à l'ordinateur d'analyser automatiquement une image et d'en extraire des informations utiles. La vision par ordinateur est un domaine clé de l'intelligence artificielle qui vise à donner aux machines la capacité d'interpréter le monde visuel. Elle permet aux ordinateurs de traiter, analyser et comprendre des images et des vidéos pour en extraire des informations utiles. Cette discipline englobe plusieurs sous-domaines tels que la détection d'objets, la reconnaissance faciale, la segmentation d'images et la classification d'objets. Elle est utilisée dans divers secteurs : médecine (imagerie médicale), automobile (véhicules autonomes), sécurité (vidéosurveillance intelligente), agriculture (suivi des cultures), etc. Elle repose sur des algorithmes d'apprentissage, des techniques de traitement d'image et surtout sur des modèles d'intelligence artificielle, notamment l'apprentissage profond. [1][2]

I.3 Réseaux de neurones convolutionnels (CNN)

Initialement introduits par Fukushima, les réseaux de neurones convolutifs (CNN) représentent une structure spécialisée des réseaux neuronaux artificiels, particulièrement appropriée pour le traitement d'images. Conçus sur le modèle du cortex visuel des ver-

tébrés, ils surpassent les MLP (perceptrons multicouches) en termes d'efficacité pour la classification et la détection d'objets dans les images, notamment grâce à leur aptitude a diminuer la complexité associée aux grandes dimensions des données visuelles. Comme Présenté dans la figure I.1 ci-après.[3]

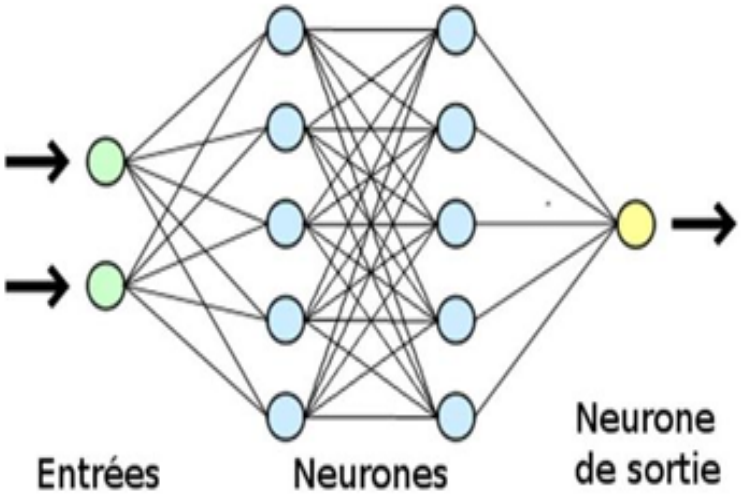


FIGURE I.1 : Les réseaux de neurones convolutifs (CNN)

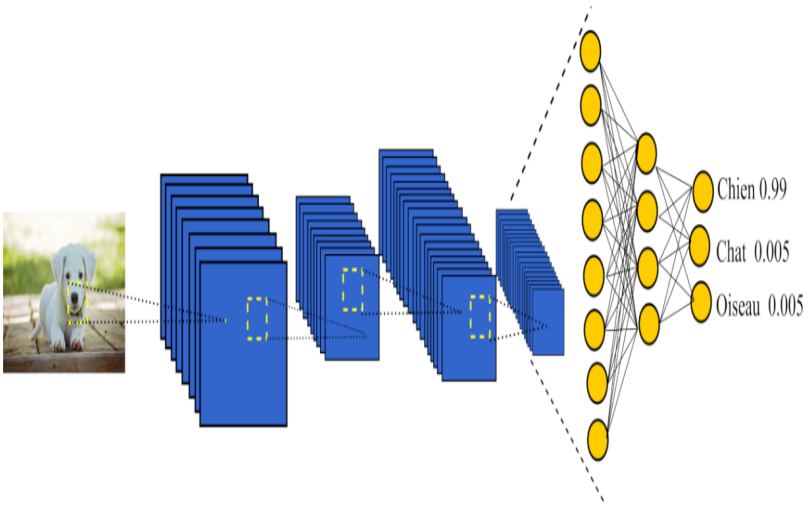


FIGURE I.2 : Le schéma d'un réseau de neurones convolutif

Un réseau de neurones convolutif est constitué de deux éléments majeurs :

• La phase de convolution : agissant en tant qu'extracteur de caractéristiques, met en œuvre Œuvre une suite de filtres sur l'image afin de produire des cartes de convolution. Ces Cartes, qui peuvent être diminuées grâce à des opérations de regroupement, sont par la suite transformées en un code distinctif.[4]

•La section de classification :repose sur des couches entièrement connectées (FC) qui nalysent les caractéristiques extraites pour déterminer la catégorie de l'image. Cette estimation est réalisée par une couche finale qui génère une distribution de probabilité,où

chaque sortie correspond à la probabilité que l'image appartienne à une classe donnée. Cette distribution est calculée à l'aide de la fonction Softmax, laquelle transforme les scores initiaux (logits) en valeurs comprises entre 0 et 1, en garantissant que la somme de ces valeurs soit égale à 1. La classe associée à la probabilité la plus élevée est alors retenue comme prédiction finale, ce qui rend le résultat clair et facile à interpréter.[4]

Les éléments de base d'un CNN :

Considérons l'exemple suivant :

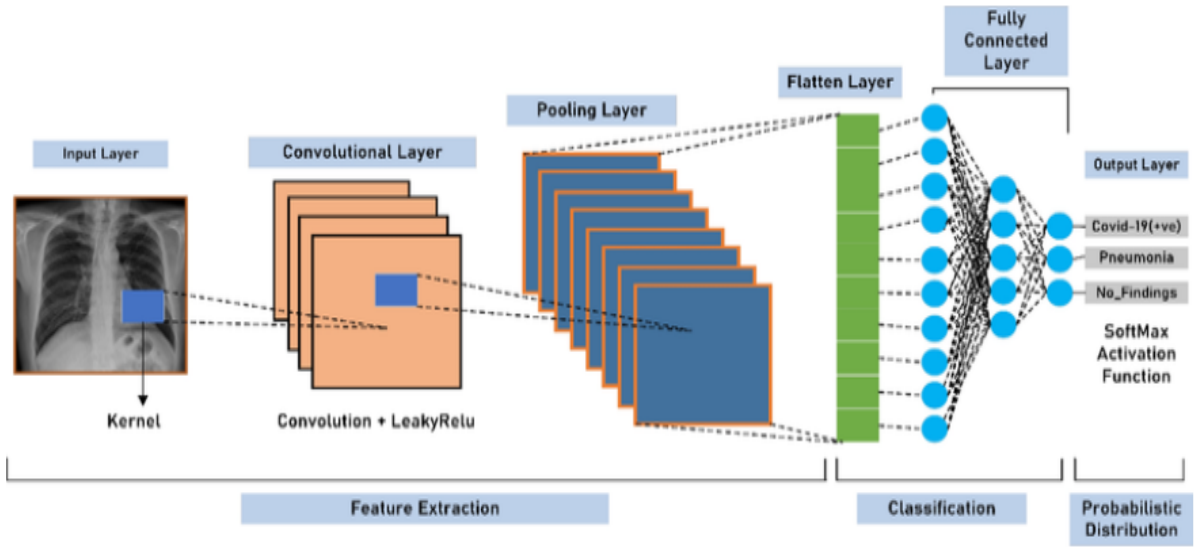


FIGURE I.3 : Les éléments de base d'un réseau de neurones convolutif.[8]

Un CNN typique comprend plusieurs types de couches :

Les couches de convolution : permettent d'extraire automatiquement des caractéristiques locales à l'aide de filtres (bords, textures, etc.).

Les couches de pooling : réduisent la taille des données tout en conservant l'essentiel des informations, ce qui rend le modèle plus rapide et évite le surapprentissage.[5]

POOLING : Le pooling, et plus précisément le max pooling, vise à diminuer la dimension des cartes de caractéristiques en ne retenant que les activations les plus importantes dans une zone locale. Cette opération permet de réduire le nombre total de paramètres du réseau et d'améliorer sa capacité de généralisation. Comme on peut l'observer dans la figure I.4

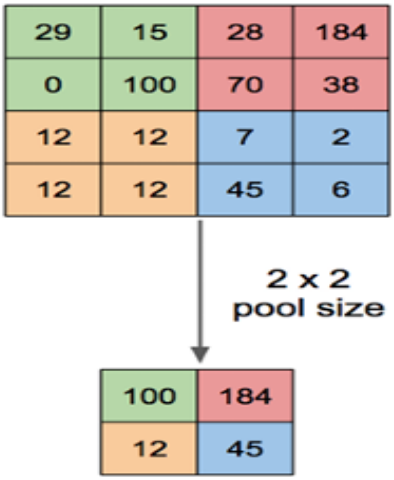


FIGURE I.4 : Le max pooling.

La fonction d'activation (ReLU) : utilise la fonction $ReLU$ (Rectified Linear Unit), qui introduit la non-linéarité et optimise le rendement du réseau. Elle est définie par :

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

La couche pleinement reliée (FC) :elle gère les caractéristiques déduites afin de rendre des décisions de haut niveau. La relation est donnée par :

$$y = f(Wx + b)$$

Où W est la matrice de poids, x l'entrée, b le biais, et f la fonction d'activation.

La fonction de perte et *softmax* : La fonction de perte est essentielle dans le processus d'apprentissage du réseau, puisqu'elle mesure la différence entre les sorties prédites par le modèle et les valeurs attendues. Dans les problèmes de classification, on utilise en général la fonction Softmax en sortie, afin de convertir les scores (logits) en probabilités normalisées comprises entre 0 et 1 :[6]

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Ainsi, les probabilités obtenues via la fonction Softmax totalisent 1, ce qui rend l'interprétation des prédictions plus intuitive. Cette fonction est généralement associée à la fonction de perte dite d'entropie croisée (cross-entropy), formulée comme suit :

$$\text{Loss} = -\sum_i y_i \log(\hat{y}_i)$$

Où y_i correspond à la valeur réelle (1 si la classe est correcte, sinon 0), et $\hat{y}_i$ à la

probabilité prédite. Cette approche sanctionne fortement les erreurs de classification et incite le modèle à attribuer une probabilité élevée aux classes exactes, ce qui favorise un apprentissage plus performant.[7]

I.4 Modeles de detection

I.4.1 MobileNet

I.4.1.1 Définition

MobileNet désigne une série de réseaux de neurones convolutifs (CNN) optimisés pour les dispositifs mobiles et les systèmes embarqués. Cette famille de modèles repose sur une architecture simplifiée qui s'appuie principalement sur les convolutions dites séparables en profondeur (depthwise separable convolutions), ce qui permet de construire des réseaux profonds tout en réduisant significativement la complexité computationnelle. Afin d'ajuster précisément le compromis entre précision et latence en fonction des contraintes matérielles, deux hyperparamètres principaux sont utilisés : un facteur de réduction de largeur et un facteur de résolution. Lancé par Google en 2017, MobileNet s'est rapidement positionné comme une solution de référence pour les tâches de vision par ordinateur sur des plateformes à ressources limitées [8]

I.4.1.2 Architecture

La structure de MobileNet s'articule principalement autour de couches de convolution séparables en profondeur, à l'exception de la toute première couche, qui reste une convolution classique. Chaque unité de base comprend deux étapes : une convolution depthwise, où chaque canal d'entrée est traité indépendamment, suivie d'une convolution pointwise (de taille 1×1) destinée à fusionner les résultats obtenus. Cette décomposition remplace la traditionnelle convolution 3×3, plus coûteuse, par deux opérations allégées. Entre les différentes couches, on applique systématiquement une normalisation par lot (batch normalization) ainsi qu'une activation Relue, sauf à la sortie du réseau. Un global average pooling est ensuite utilisé pour réduire les dimensions spatiales avant d'atteindre la couche finale entièrement connectée. En prenant en compte séparément les convolutions depthwise et pointwise, l'architecture complète comporte 28 couches. Cette organisation régulière de blocs factorisés permet d'optimiser l'utilisation des convolutions denses et de tirer parti des implémentations rapides (comme GEMM), ce qui améliore significativement la vitesse d'exécution sur les processeurs embarqués.[9]

I.4.1.3 Principes de fonctionnement

L'un des fondements de MobileNet repose sur la décomposition des convolutions standards en deux opérations distinctes : d'une part, un filtrage spatial réalisé séparément sur

chaque canal d'entrée, et d'autre part, une fusion linéaire des canaux via une convolution 1×1. Cette factorisation permet une réduction substantielle du coût computationnel ainsi que du nombre total de paramètres par rapport à un réseau convolutionnel classique. Concrètement, la convolution depthwise applique un filtre individuel à chaque canal, tandis que la convolution pointwise combine les résultats pour générer la sortie finale. Par rapport à une convolution 3×3 conventionnelle, cette approche permet de réduire le nombre d'opérations d'un facteur de 8 à 9, tout en maintenant une précision comparable. Pour assurer la stabilité du réseau, MobileNet applique systématiquement une normalisation par lot et une activation ReLU après chaque étape de convolution. Deux hyperparamètres de compression viennent renforcer cette flexibilité : le multiplicateur de largeur, qui module le nombre de canaux dans chaque couche, et le multiplicateur de résolution, qui agit sur la taille de l'image d'entrée ainsi que sur la résolution des cartes de caractéristiques internes. Ces ajustements permettent d'adapter dynamiquement le modèle aux contraintes matérielles tout en préservant une bonne qualité de prédiction.[10]

I.4.1.4 Paramètres de compression

MobileNet introduit deux hyperparamètres principaux afin d'optimiser la taille et la vitesse du modèle. Le premier, appelé multiplicateur de largeur (noté α), agit sur l'ensemble du réseau en réduisant de façon homogène le nombre de filtres dans chaque couche. Ainsi, si un bloc dispose initialement de M canaux en entrée et de N en sortie, ceux-ci deviennent αM et αN respectivement. Cette diminution entraîne une réduction quadratique du volume de calcul et du nombre de paramètres, d'environ α^2. Le second hyperparamètre, appelé multiplicateur de résolution (noté ρ), intervient sur les dimensions spatiales en réduisant la résolution de l'image d'entrée ainsi que celle des représentations internes de manière proportionnelle. En général, des tailles fixes comme 224, 192, 160 ou 128 pixels sont utilisées. Cette stratégie permet de diminuer le coût global de traitement d'environ ρ^2. Grâce à cette combinaison, il est possible de générer différentes variantes de MobileNet offrant un bon compromis entre efficacité, compacité et performance. [11]

I.4.1.5 Extension : MobileNetV3 (2020)

- **Améliorations :** La version MobileNetV3 marque une avancée significative par rapport à ses prédécesseurs, notamment grâce à l'intégration de techniques modernes de recherche d'architecture automatique (NAS) associées à l'algorithme NetAdapt. Cette nouvelle architecture a été spécifiquement optimisée pour les processeurs mobiles (CPU de smartphones), en combinant les résultats de NAS et NetAdapt avec des ajustements architecturaux manuels. Parmi ces ajustements, l'introduction de la fonction d'activation hard-swish (version simplifiée de swish) dans les couches profondes améliore la précision globale sans engendrer de surcoût computationnel notable. En outre, tous les

modules squeeze-and-excite ont été harmonisés en adoptant une taille fixe correspondant à un quart des canaux d'expansion. Cette calibration renforce la précision tout en maintenant un équilibre entre efficacité et complexité paramétrique. L'ensemble de ces améliorations, combinées aux méthodes de recherche d'architecture, a conduit à la conception de deux modèles distincts, chacun adapté à un profil d'usage particulier. [12]

- **Architecture :** MobileNetV3 se décline en deux principales configurations : V3-Large pour les environnements nécessitant une haute précision, et V3-Small pour les systèmes contraints en ressources. Ces deux modèles reposent sur des bottleneck blocks similaires à ceux utilisés dans MobileNetV2, incluant des convolutions 3×3 ou 5×5, des couches d'expansion 1×1 et des projections 1×1. Toutefois, ils intègrent des améliorations notables comme les activations hard-swish dans les couches profondes et l'utilisation généralisée des modules squeeze-and-excite. La première couche reste typiquement une convolution classique de petite taille (souvent une 3×3 avec stride 2), suivie d'une série de blocs optimisés. Le modèle V3-Large dispose d'un plus grand nombre de couches et de canaux comparé au V3-Small, bien que les deux versions intègrent des optimisations adaptées aux contraintes matérielles. Fidèle à la philosophie de la série, cette nouvelle itération conserve une architecture épurée, sans recourir à des éléments récurrents ni à des convolutions tridimensionnelles, tout en exploitant l'automatisation de la conception pour maximiser l'efficacité sur les appareils mobiles. La figure I.5 met en évidence ce fonctionnement. [13]

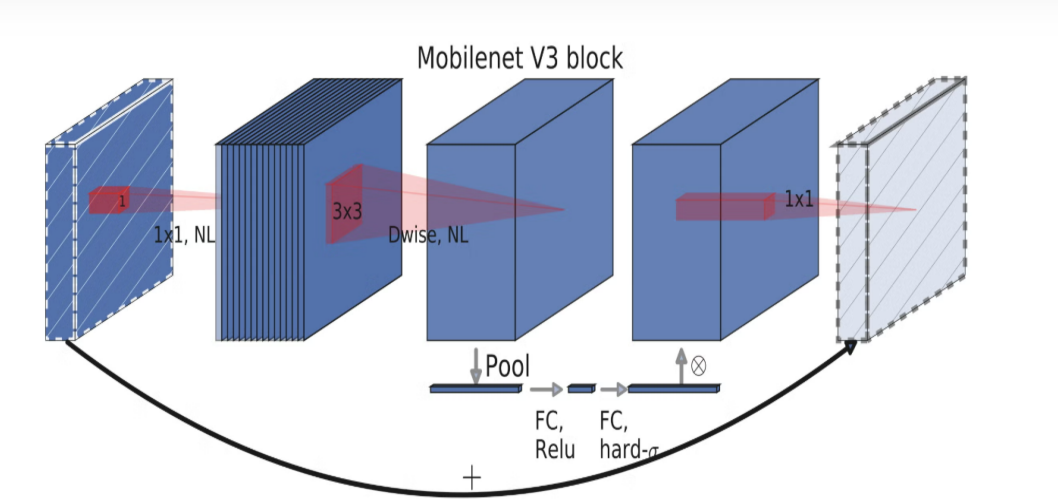


FIGURE I.5 : Le schéma de l'architecture MobileNet V3 [14].

- **Performances :** MobileNetV3 établit de nouveaux standards en termes de compromis entre précision et latence pour les dispositifs mobiles. Sur le benchmark ImageNet, la version V3-Large affiche une amélioration de précision top-1 d'environ $3,2\%$ tout en diminuant la latence de près de 20% par rapport à MobileNetV2. Le modèle V3-

Small, quant à lui, conçu pour un maximum de légèreté, obtient une précision top-1 supérieure de $6,6\%$ avec une latence équivalente à celle d'un MobileNetV2 de même dimensionnement. En détection d'objets sur le jeu de données COCO, V3-Large permet une accélération d'environ 25% sans perte de précision notable. Concernant la segmentation sémantique (jeu de données Cityscapes), l'intégration du nouveau décodeur Lite R-ASPP confère à MobileNetV3-Large une rapidité supérieure d'environ 34% par rapport à V2 utilisant un R-ASPP classique, tout en conservant des performances similaires. Les études d'ablation indiquent que ces gains sont dus à la combinaison des optimisations introduites, incluant NAS, NetAdapt, hard-swish et les modules SE, traduisant un net progrès dans l'état de l'art des modèles compacts. [15]

- **Applications :** MobileNetV3 hérite des cas d'usage de ses versions précédentes tout en les étendant, notamment dans les systèmes embarqués. Les modèles V3-Large et V3-Small sont utilisés pour la classification d'images sur appareils mobiles, ainsi que pour des tâches de détection et de segmentation optimisées pour les ressources limitées. Par exemple, ils servent de backbone dans les architectures de détection en temps réel pour smartphones ou encore dans des solutions de segmentation sémantique embarquée, notamment dans des applications de réalité augmentée. La disponibilité de deux variantes permet une adaptation aux exigences spécifiques : V3-Large est privilégié lorsque la précision est une priorité, tandis que V3-Small répond aux besoins des plateformes très contraignantes. Globalement, MobileNetV3 ouvre la voie à une nouvelle génération d'applications mobiles en vision artificielle, en combinant hautes performances et efficacité computationnelle. [16]

I.4.2 Architectures de référence en détection d'objets

La détection d'objets, une méthode de vision par ordinateur, combine la classification (qui consiste à reconnaître une catégorie d'objets ou de personnes) et la localisation (qui cherche à établir une position avec un cadre englobant). Le défi majeur est d'avoir une localisation précise, qui est un critère clé de performance. Cette dernière dépend de plusieurs facteurs : la précision, la rapidité, les capacités matérielles et le nombre de catégories qu'on peut gérer, comme illustré dans la figure I.6 ci-dessous [17]

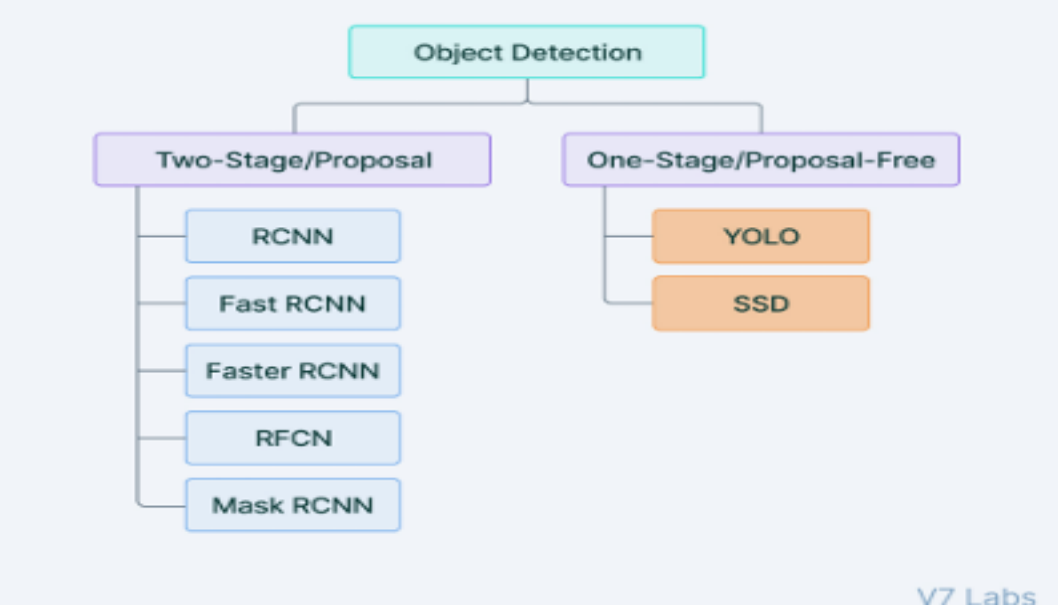


FIGURE I.6 : Architecture de détection objet

I.4.2.1 Typologie des approches

Il existe deux types d'approches comme le montre la figure I.7.

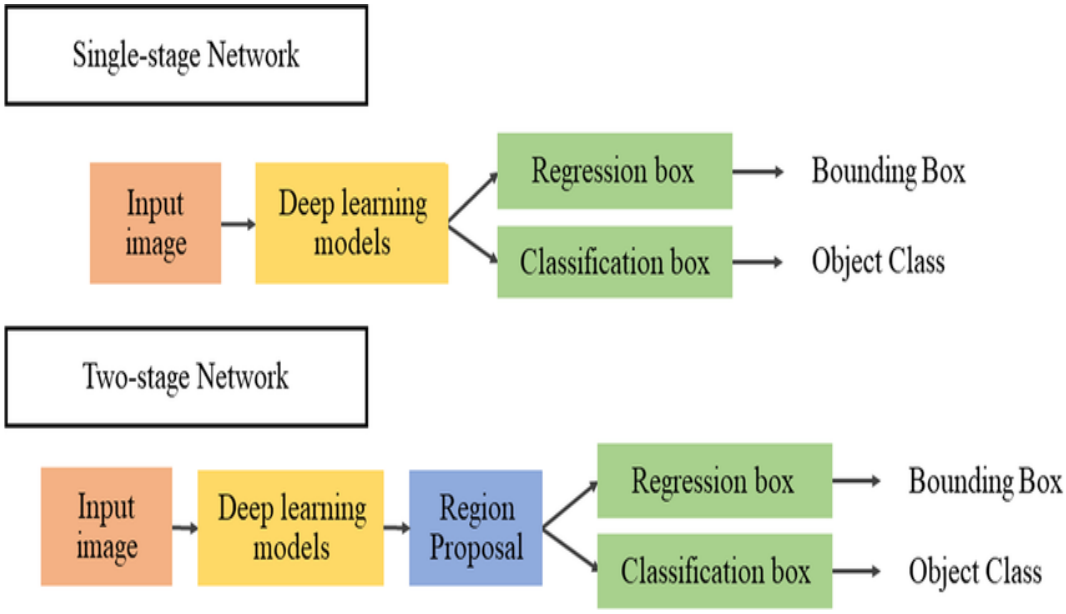


FIGURE I.7 : Architecture d'une seule ou plusieurs stades

A. L'approche One-stage :

Les détecteurs d'objets d'une seule étape sont définis par leur approche end-to-end, qui supprime la nécessité d'une étape spéciale. Cette dernière consomme beaucoup de ressources pour proposer des zones d'intérêt (zones susceptibles de contenir des

objets). Ils traitent la détection d'objets comme un problème de régression. Le modèle examine l'image d'entrée uniquement une fois, utilisant généralement un réseau de base (souvent un réseau de neurones convolutifs ou CNN) pour obtenir les caractéristiques. Ces caractéristiques sont ensuite envoyées directement à une tête de détection qui prédit en même temps les probabilités de classe, les coordonnées des boîtes englobantes et les scores de confiance. Ce design à passage unique est optimisé pour la rapidité et convient aux applications nécessitant un traitement rapide. Parmi les exemples notables se trouvent la série de modèles Ultralytics YOLO, bien connus pour équilibrer vitesse et précision (comme YOLOv11), ainsi que le SSD (Single Shot MultiBox Detector) développé par Google Research. [12]

- Les principaux modèles de détecteurs d'objet à une étape

- **Le modèle You Only Look Once (YOLO) :** abrégé en YOLO, regroupe des algorithmes de vision par ordinateur très populaires. Il est capable de réaliser diverses tâches comme la classification, la segmentation d'images et la détection d'objets. YOLO prend en charge un modèle de détection unique qui traite les images en un seul passage, offrant des résultats pertinents et remplaçant ainsi les modèles basés sur des réseaux de neurones convolutionnels, comme illustré dans la figure I.8 ci-dessous. [12]

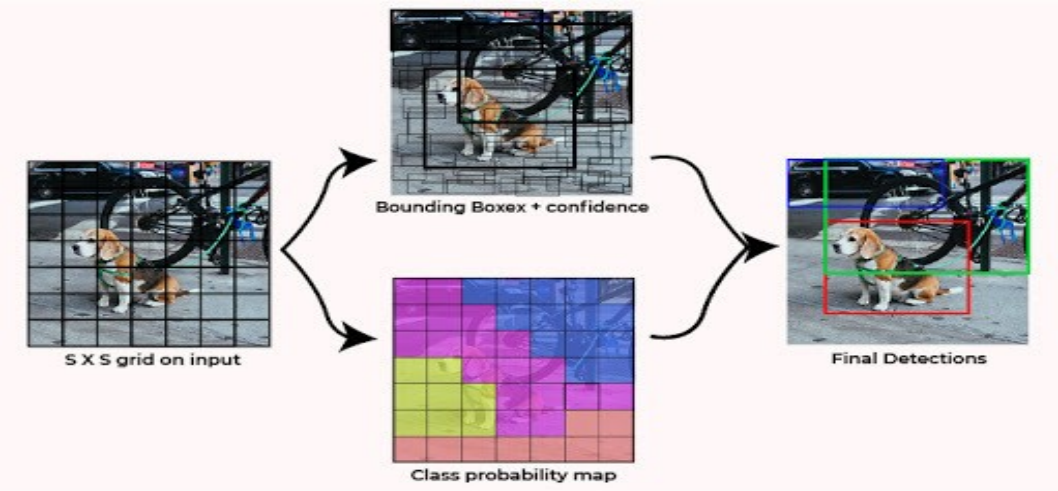


FIGURE I.8 : Détection d'objets avec YOLO [**?**]

- **Le modèle Single-Shot Detector (SSD) :** est un autre type de détecteur à une étape qui permet de repérer et de prévoir plusieurs classes d'objets. Le SSD s'appuie sur un réseau de neurones profond qui apprend à ajuster les espaces de sortie des fenêtres de boîte englobante (bounding box) dans les images et les vidéos, puis il produit des scores pour les différentes catégories d'objets dans ces boîtes prédéfinies [23] . Ce modèle affiche une grande précision, est facile à entraîner et peut être mis

en œuvre dans divers logiciels et plateformes dotés de fonctionnalités de détection d'objets, comme illustré dans la figure II.4 ci-dessous. [13]

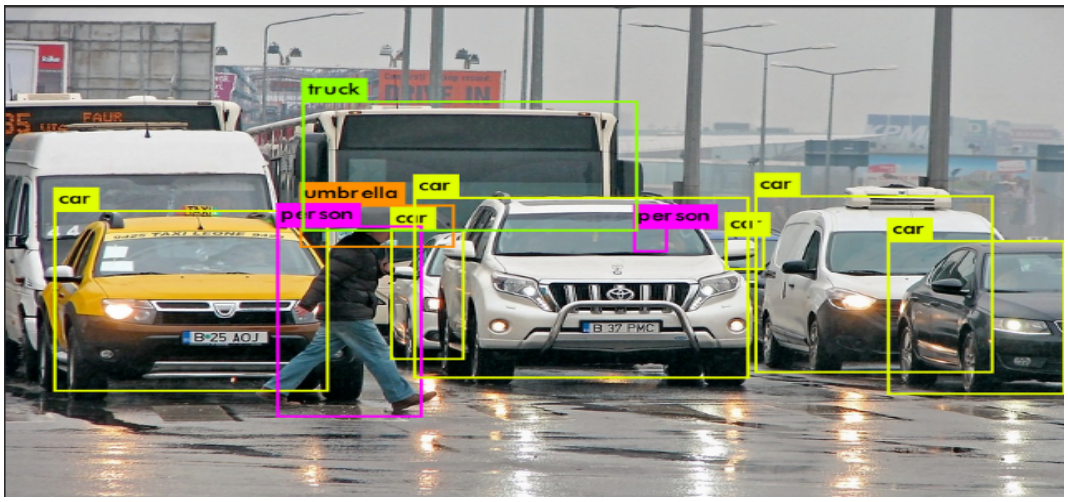


FIGURE I.9 : Détection d'objets avec SSD [18]

B. L'approche Two-stages :

Les détecteurs d'objets à deux étapes fonctionnent selon deux phases principales :

- **Phase 1 : Extraction des caractéristiques et génération des RoIs :** des régions d'intérêt *Regions of Interest* (RoIs) sont proposées à l'aide de *Selective Search* ou d'un *Region Proposal Network* (RPN).
- **Phase 2 : Classification et ajustement des coordonnées :** les objets sont classifiés et les boîtes englobantes ajustées grâce à *RoI Pooling* ou *RoI Align* combinés à des réseaux de neurones convolutifs (CNN). [13]

I.4.2.2 Sélection d'une architecture adaptée à un environnement encombré

Dans des environnements encombrés, on peut trouver beaucoup d'objets, des superpositions fréquentes et des parties cachées. Les systèmes de détection d'objets utilisant des réseaux de neurones convolutifs à deux étapes sont très efficaces. Le modèle Faster R-CNN, présenté par Ren et ses collègues en 2015, comprend un Réseau de Propositions de Régions (RPN). Celui-ci génère de manière efficace des zones candidates, ce qui améliore la précision dans des scènes compliquées. De son côté, le Mask R-CNN, proposé par He et al en 2017, ajoute une branche pour la segmentation d'instances, permettant d'identifier précisément les objets même quand ils sont en partie cachés. [14]

Pour des environnements particulièrement denses, des adaptations comme le Double Mask R-CNN (Liu et al., 2022) incorporent des modules de segmentation sémantique et d'attention, ce qui aide à mieux différencier les objets proches.[14]

Bien que les modèles à une étape, tels que YOLOv3, permettent une détection instantanée, leur efficacité tend à diminuer dans des situations très encombrées. Des améliorations récentes, comme l'ajout de mécanismes d'attention (comme SENet) et l'emploi de fonctions de perte avancées (telles que GIoU), ont été suggérées par Zhang et al en 2023 pour surmonter ces défis. Cependant, pour les applications où une grande précision est nécessaire dans des environnements complexes, les modèles à deux étapes sont toujours les plus conseillés. On peut avoir plusieurs bibliographies pour la même partie [15]

I.5 Métriques d'évaluation :

L'évaluation d'un système de détection repose sur des métriques objectives :

IoU (Intersection over Union) : mesure la qualité du chevauchement entre la boîte prédite et la boîte réelle. Plus le score est proche de 1, meilleure est la détection.

mAP (mean Average Precision) : moyenne des précisions pour toutes les classes. Elle prend en compte à la fois le rappel (Recall) et la précision (Precision).

Recall : proportion des objets correctement détectés parmi tous ceux présents dans l'image.

Precision : proportion des détections correctes parmi toutes les détections.

F1-score : mesure combinée de la précision et du rappel. Elle permet un équilibre entre les faux positifs et les faux négatifs. Ces métriques sont essentielles pour comparer les modèles et comprendre leurs points forts et faibles.[16]

I.6 Travaux similaires

De nombreux travaux ont été réalisés autour de la détection d'objets. Les premières approches se basaient sur des descripteurs manuels (SIFT, HOG) couplés à des classifieurs (SVM, Random Forest). Avec l'essor de l'apprentissage profond, les modèles CNN ont révolutionné ce domaine :

YOLO (You Only Look Once) propose une approche rapide à une seule étape.

RetinaNet introduit la focal loss pour gérer le déséquilibre entre classes.

SSD est un compromis entre rapidité et précision.

Les travaux récents combinent ces modèles à des architectures légères comme MobileNet pour une utilisation sur appareils mobiles.

Notre choix du modèle SSD-MobileNetV3 s'inspire de ces recherches pour offrir un bon équilibre entre performance et efficacité computationnelle.[17]

I.7 Conclusion

Ce chapitre a posé les bases théoriques de notre travail. Nous avons exploré la vision par ordinateur, les CNN, les modèles de détection d'objets, les métriques d'évaluation et les contributions des travaux antérieurs. Ces notions serviront de fondement pour comprendre la méthodologie détaillée dans le prochain chapitre

Chapitre II

Approche méthodologique et mise en œuvre

II.1 Introduction

La mise en œuvre d'un système de reconnaissance d'objets nécessite une démarche méthodologique rigoureuse. Dans ce chapitre, nous décrivons les étapes suivies pour concevoir et réaliser notre système, en mettant l'accent sur l'approche adoptée, les outils utilisés, la préparation des données, l'intégration du modèle préentraîné SSD MobileNetV3, ainsi que les critères retenus pour l'évaluation des performances.

L'objectif principal est de proposer une solution qui soit à la fois fonctionnelle, rapide, et adaptée à des conditions d'utilisation réelles, notamment en présence de bruit visuel et de complexité de scènes. L'enjeu est aussi de démontrer la pertinence du recours à un modèle préentraîné, capable d'être utilisé sans nécessiter un entraînement lourd sur des ressources importantes.[18]

II.2 Description du dataset

Les données utilisées sont issues de plusieurs bases accessibles librement (COCO, Open Images...) et enrichies manuellement avec des images capturées dans des environnements réels. Elles présentent des cas de bruit visuel, d'objets superposés, de faible contraste, ou encore d'angles variés. Le dataset a été sélectionné pour refléter au mieux les conditions réelles et les défis posés par des environnements encombrés.[19]

• Qu'est-ce qu'un dataset en vision par ordinateur ?

Un dataset est un ensemble structuré d'images (ou de vidéos) utilisées pour entraîner, tester ou valider un système d'intelligence artificielle. Dans la détection d'objets, ces images doivent contenir des objets annotés avec :leur catégorie (ex. : voiture, personne, bouteille...),et leur position (via des boîtes englobantes appelées bounding boxes).

• Origine du dataset utilisé dans ce projet

Dans ce projet, le dataset :N'a pas été généré manuellement à partir de zéro, mais plutôt collecté depuis plusieurs sources libres en ligne, comme Open Images Dataset, Pexels, ou des banques d'images gratuites.Il a été complété manuellement par des images spécifiques capturées dans des environnements réels (ex. via webcam, smartphone).[19]

• Contenu du dataset : diversité et complexité :

Les images sélectionnées contiennent des scènes encombrées, c'est-à-dire avec plusieurs difficultés :

•Type de complexité Exemple / Description :

-Objets chevauchés :Deux objets se superposent, rendant leur séparation difficile.

-Luminosité variable :Certaines images sont très sombres, d'autres surexposées.

-Bruit visuel :L'arrière-plan contient beaucoup de détails ou d'objets non pertinents.

-Angles variés :L'objet est vu de côté, en diagonale ou partiellement caché.

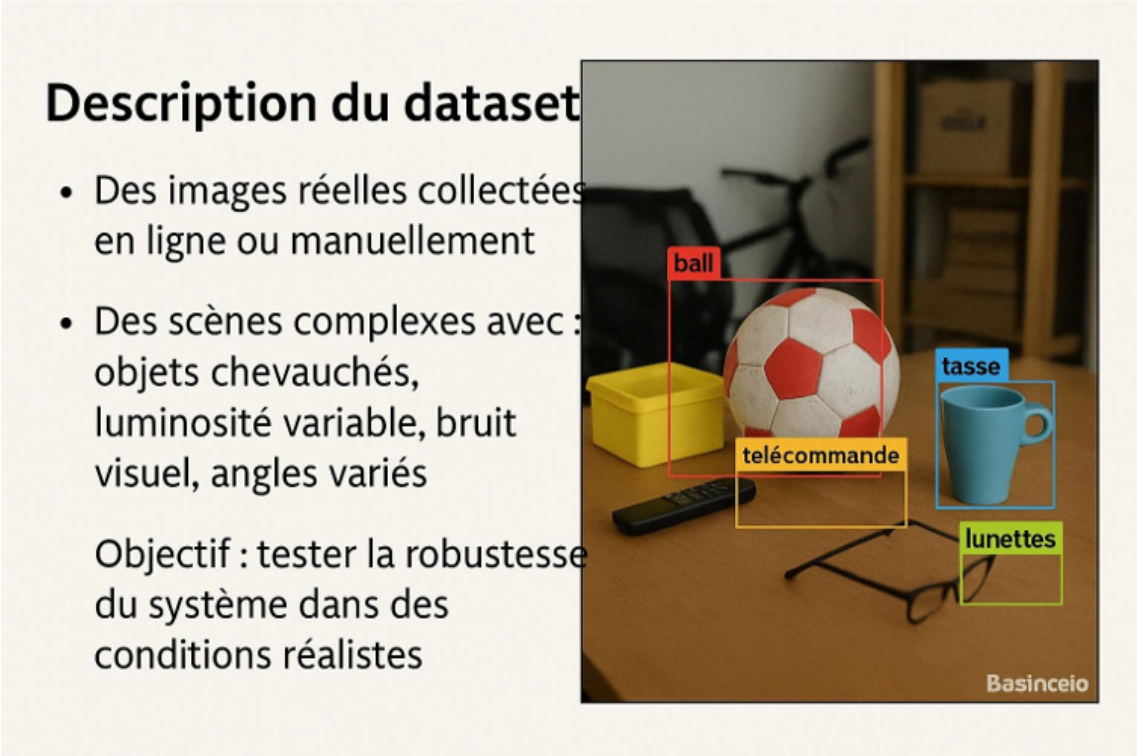


FIGURE II.1 : Exemple de description du dataset

Ce choix n'est pas un hasard : il permet d'évaluer si le modèle est robuste et réaliste dans des situations similaires au monde réel, comme une caméra de surveillance dans une rue très fréquentée.[19]

• Objectif du dataset :

Le but de ce dataset n'est pas uniquement de faire fonctionner le modèle, mais de tester sa précision dans des conditions difficiles.

-de vérifier sa capacité à généraliser à des images qu'il n'a jamais vues.

-d'observer les erreurs typiques qu'il peut produire (ex. : confusion entre objets, objets manqués. . .).[20]

II.3 Prétraitement et histogrammes

Le traitement des données est une étape cruciale. Les images sont :redimensionnées à 320x320 pixels,converties en format RGB,normalisées pour uniformiser les valeurs de pixels,filtrées pour exclure les images floues ou non pertinentes.

Pour chaque image, un histogramme des niveaux de gris est généré afin d'analyser la répartition des intensités lumineuses. Cela permet de détecter les images surexposées, sous-exposées ou à faible contraste.

Pour comprendre ce qui vient d'être mentionné, il est nécessaire de définir ce qu'est un histogramme ainsi que les images en niveaux de gris.[20]

•Histogramme :

L'histogramme est une représentation graphique fondamentale dans le domaine du traitement d'images. Il permet d'illustrer la répartition des niveaux de gris (dans une image en noir et blanc) ou des valeurs de couleurs (dans une image en couleur) en fonction de leur fréquence d'apparition.

Concrètement, un histogramme trace, sur l'axe horizontal, l'ensemble des intensités possibles (de 0 à 255 pour une image sur 8 bits), et, sur l'axe vertical, le nombre de pixels correspondant à chaque intensité. Ce type de représentation offre une lecture immédiate de la dynamique lumineuse d'une image.[21]

Il permet à l'utilisateur d'avoir une évaluation rapide et visuelle de la qualité de l'image analysée, et peut expliquer, dans certains cas, les échecs ou les incertitudes du modèle de reconnaissance lorsque l'image est fortement déséquilibrée en intensité lumineuse.

•Images à niveaux de gris (Monochromes) :

La valeur du niveau de gris correspond à l'intensité lumineuse en un lieu précis. Le pixel Peut afficher des couleurs allant du noir au blanc, en traversant une série finie de niveaux Intermédiaires. Ainsi, pour illustrer les images en niveaux de gris, on peut associer à chaque Pixel de l'image une valeur correspondant à la quantité de lumière réfléchie. Par exemple,Cette valeur peut varier entre 0 et 255. Ainsi, chaque pixel est désormais symbolisé par 1octet plutôt que par 1 bit. Il est nécessaire que l'équipement utilisé pour présenter l'image soit en mesure de générer les diverses nuances de gris correspondantes. La quantité de nuances de gris est déterminée par le nombre de bits employés pour Caractériser la « couleur » de chaque pixel sur l'image. L'augmentation de ce nombre conduit à une Multitude de niveaux potentiels.[22]

•Pourquoi l'utiliser ?

Pour analyser la qualité de la lumière dans une image. Pour détecter les cas extrêmes :Images trop sombres (pixels concentrés à gauche de l'histogramme),Images surexposées (pixels concentrés à droite),Images à faible contraste (peu de variations dans les pixels).

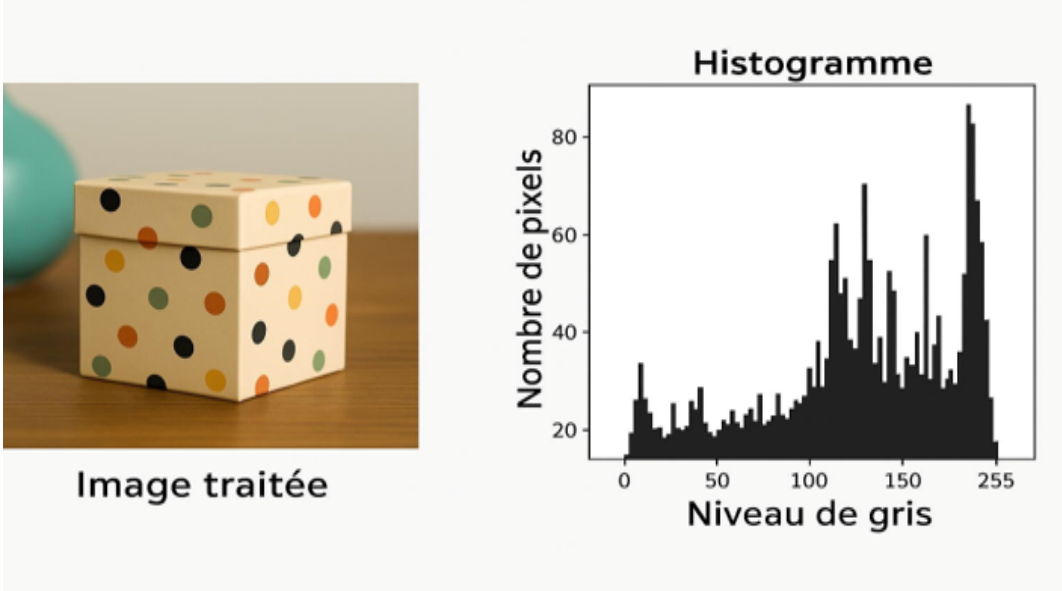


FIGURE II.2 : Image représente histogramme et leur niveau de gris

•Que faire avec un histogramme ?

On peut ajuster le contraste ou rejeter les images inutilisables.

Il sert aussi à faire des statistiques globales sur la base de données.

Impact sur les performances du système

L'effet direct d'un bon prétraitement :Le modèle détecte mieux les objets, même dans des conditions difficiles.Il se concentre sur les formes réelles plutôt que sur le bruit ou l'arrière-plan.Il devient plus rapide, car les images sont optimisées.

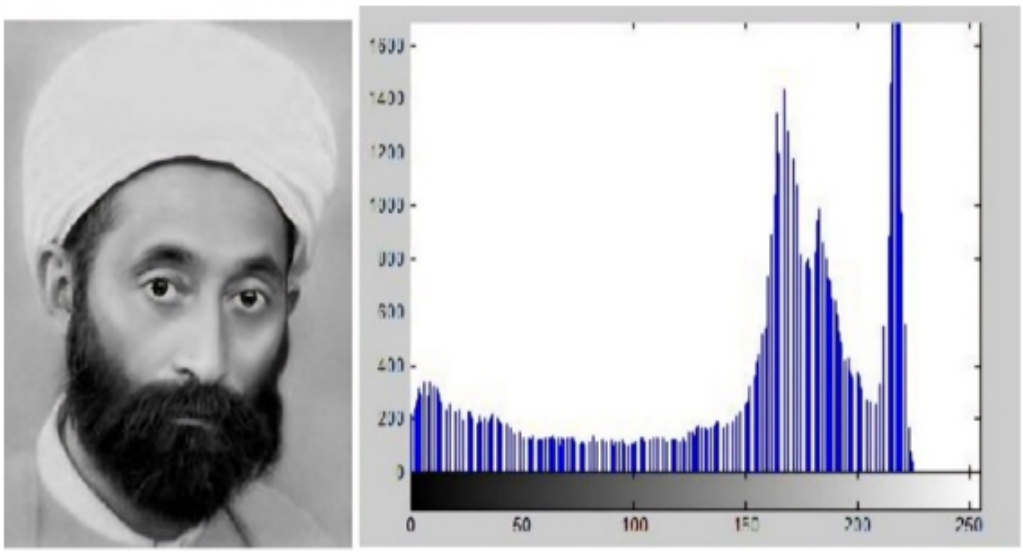


FIGURE II.3 : Image en niveaux de gris et son histogramme

•Utilité pour la caractérisation visuelle de l'image : L'analyse histogrammique constitue

un outil fondamental pour évaluer quantitativement le contenu visuel d'une image. Elle permet notamment de détecter des défauts tels qu'un mauvais éclairage, un contraste insuffisant ou encore une surexposition. Dans le cadre de la détection d'objets, ces indications sont précieuses pour guider l'utilisateur dans le choix d'images de meilleure qualité ou l'inciter à ajuster les paramètres de prétraitement. Ainsi, l'histogramme joue un rôle de filtre visuel préalable à la détection. L'affichage conditionnel de ce graphique, activé via une case à cocher ($self.show\_histogram.get()$), constitue un outil d'assistance pertinent qui améliore l'ergonomie sans surcharger l'interface d'utilisateur.

II.4 Architecture SSD MobileNetV3

Le modèle utilisé est SSD MobileNetV3, préentraîné sur COCO. Il est intégré à l'aide de TensorFlow. Ce modèle fonctionne selon les étapes suivantes :Entrée d'une image traitée,Prédiction de boîtes englobantes avec score de confiance et classe,application de la suppression non maximale (NMS) pour éliminer les doublons.

• Qu'est-ce que SSD MobileNetV3 ?

SSD MobileNetV3 est un modèle de détection d'objets qui combine deux composants puissants :

-SSD (Single Shot Multibox Detector) : un algorithme de détection rapide à une seule étape.

-MobileNetV3 : un backbone (réseau de base) léger, optimisé pour fonctionner sur des appareils mobiles ou à faibles ressources.

Ce modèle est idéal pour des applications embarquées où la mémoire, la puissance de calcul ou la consommation énergétique sont limitées.

Ce choix de modèle a été motivé par sa légèreté, sa rapidité et sa bonne précision sur des appareils à faible puissance.[23]

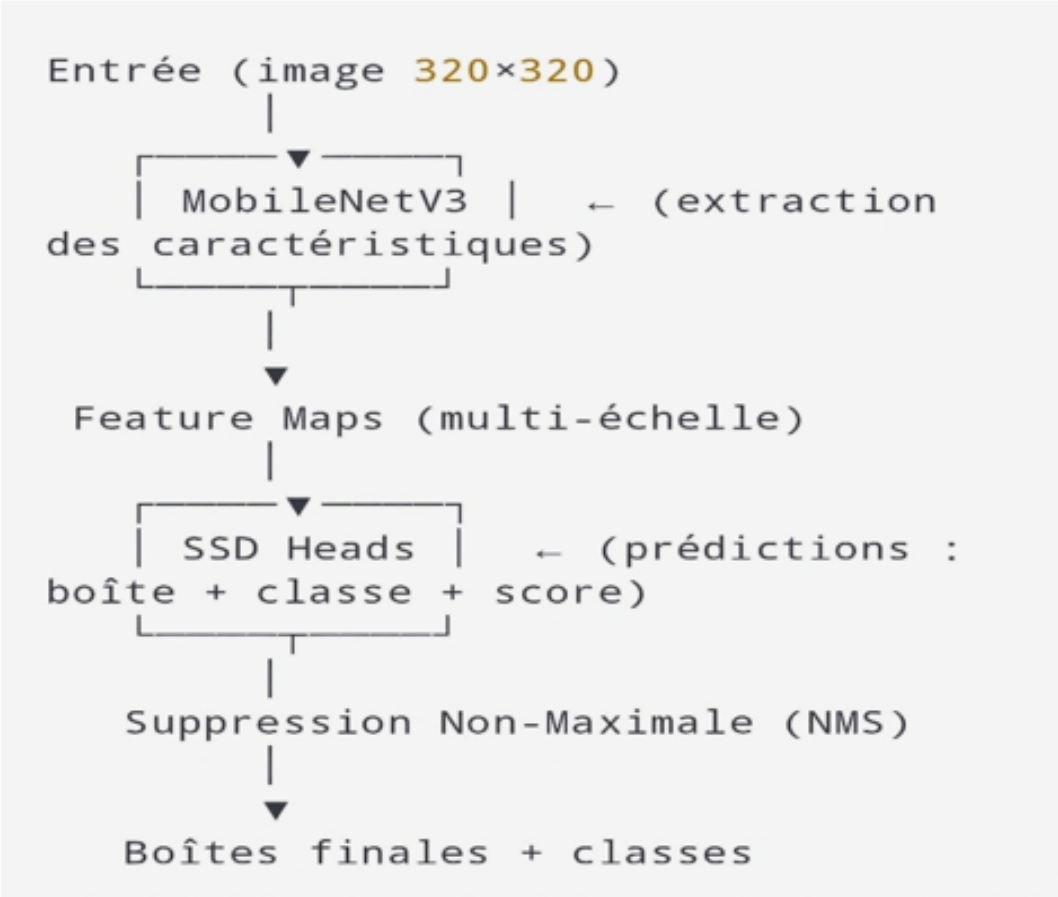


FIGURE II.4 : la structure simplifiée de SSD MobileNetV3

Voici la structure simplifiée de SSD MobileNetV3 dans la photo précédente.
Chaque feature map produit plusieurs prédictions. SSD fonctionne sur plusieurs tailles de grille (ex. : 10×10, 5×5, etc.) pour détecter des objets de différentes tailles.

II.5 Développement logiciel et interface utilisateur

Le développement de notre application a été réalisé dans un environnement Python, choisi pour sa richesse en bibliothèques dédiées à l'intelligence artificielle et au traitement d'images. Nous avons utilisé les outils suivants :

Python 3.9.11 : langage principal du projet, pour sa simplicité de syntaxe et sa large adoption dans le domaine de l'IA.

OpenCV (cv2) : Utilisée pour la détection d'objets basée sur les réseaux de neurones pré-entraînés, ainsi que pour le traitement d'images (conversion de formats, extraction de l'histogrammes, etc.).

Matplotlib : Permet l'affichage graphique des métriques d'entraînement (précision, perte, IoU), facilitant ainsi l'évaluation visuelle des performances du modèle.

NumPy : Utilisée pour la manipulation de données numériques, notamment lors de la génération ou du traitement des courbes d'évaluation.

PIL (Pillow) : Sert à convertir et annoter les images via ImageDraw et ImageFont pour afficher les résultats de détection.

Arabic_reshaper et Bidi.algorithm : Ces bibliothèques permettent de rendre correctement les textes arabes, qui nécessitent un réordonnancement visuel spécifique à l'écriture bidirectionnelle.

Pyttsx3 et gTTS + pygame : Intégrées pour la synthèse vocale dans plusieurs langues. Pyttsx3 est utilisé pour le français et l'anglais, tandis que gTTS (Google Text-to-Speech) et pygame gèrent la vocalisation en arabe.

Customtkinter (CTk) : Utilisée pour concevoir une interface graphique moderne, réactive et ergonomique. Elle remplace avantageusement tkinter, la bibliothèque standard de Python pour créer des interfaces graphiques, en offrant une personnalisation avancée (apparence, thèmes, widgets).

Datetime : Permet de récupérer la date et l'heure courantes. Elle est utile pour nommer automatiquement les fichiers de sortie ou pour enregistrer des horodatages lors des prédictions ou sauvegardes.

Os : Permet d'interagir avec le système d'exploitation : gestion des chemins de fichiers, création de dossiers, vérification de l'existence de fichiers, etc.

Cet ensemble d'outils, sera plus développé au cours du chapitre 4, nous a permis de structurer le projet efficacement tout en gardant une cohérence technique entre les différentes parties.

•L'interface permet de lancer les différentes actions, tels que le chargement d'images, l'activation de la caméra,ou encore le déclenchement du module de détection. Un menu déroulant multilingue permet à l'utilisateur de sélectionner la langue d'affichage des étiquettes d'objets détectés (français, arabe, anglais), en cohérence avec les noms d'objets comme le montre la figure.

L'interface permet également l'affichage de l'histogramme des images ainsi que des Métriques de performance du modèle, offrant ainsi une meilleure compréhension des Résultats et une évaluation visuelle de l'efficacité de la détection.

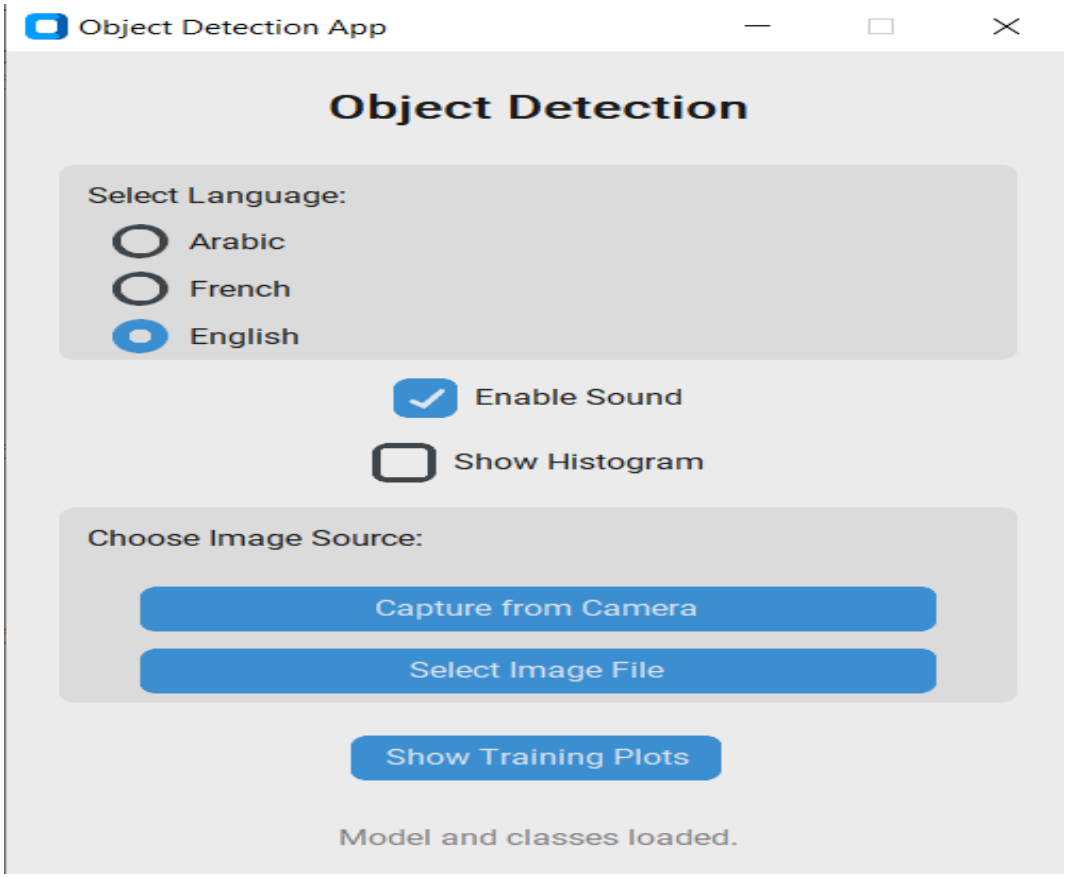


FIGURE II.5 : Interface utilisateur graphique du système (GUI)

•Interaction utilisateur :

-Charger une image depuis le PC : L'utilisateur peut sélectionner une image Locale à partir du disque via une boîte de dialogue classique. L'image choisie est Ensuite affichée dans l'interface et transmise au modèle CNN pour traitement.

-Prendre une photo avec la caméra : Une fonction d'acquisition en direct Via webcam est intégrée, permettant de capturer une image instantanée en un Seul clic. Cela offre une interaction directe entre l'utilisateur et le système de Détection.[26]

-Afficher l'objet détecté : Une fois l'image chargée ou capturée, l'utilisateur Peut lancer la détection. Le modèle MobileNet v3 traite l'image et retourne les Objets identifiés. Ces objets sont affichés sous forme de boîtes de délimitation Avec étiquettes de classe, accompagnées de leur pourcentage de confiance, traDuites dynamiquement selon la langue sélectionnée. L'affichage est visuel, clair Et immédiat, renforçant la lisibilité des résultats.[26]

II.6 Synthèse vocale multilingue

•Reconnaissance vocale multilingue :

Afin d'optimiser l'accessibilité et l'interaction naturelle avec l'utilisateur, l'application

intègre une composante de reconnaissance vocale multilingue prenant en charge trois langues : l'arabe, le français et l'anglais. L'utilisateur peut sélectionner la langue désirée via une interface conviviale. Une fois la langue choisie, le système adapte dynamiquement les fichiers de classes names pour l'affichage des résultats dans la langue correspondante, tout en modifiant les voix de synthèse utilisée.

- **Choix de 3 langues :** La sélection linguistique est implémentée à l'aide de boutons radio (RadioButton) dans l'interface utilisateur graphique. En fonction du choix (Arabic, French, English), le système charge les fichiers $thing\_ar.names$, $thing\_fr.names$ ou $thing\_en.names$ respectivement, et configure le moteur vocal pour correspondre à la langue active (figure II.6).

```
def _language_changed(self):
    choice = self.lang_choice.get()
    if choice == "1":
        self.lang_file = 'C:/Users/mcs/Desktop/project1/files/thing_ar.names'
        self.selected_lang_code = 'ar'
    elif choice == "2":
        self.lang_file = 'C:/Users/mcs/Desktop/project1/files/thing_fr.names'
        self.selected_lang_code = 'fr'
    else:
        self.lang_file = 'C:/Users/mcs/Desktop/project1/files/thing_en.names'
        self.selected_lang_code = 'en'
```

FIGURE II.6 : Choix de 3 langues.

- **Reconnaissance des commandes vocales :** Le projet fourni contient une reconnaissance de commandes vocales sortantes, il inclut une structure permettant une interaction orale via la reconnaissance d'objets détectés. Le système détecte les objets à partir d'une image et les identifie avec leur nom dans la langue sélectionnée.
- **Synthèse vocale pour annoncer le résultat :** Le système intègre deux moteurs de synthèse vocale : Google Text-to-Speech (gTTS) couplé à pygame, une bibliothèque Python utilisée ici pour lire les fichiers audios générés, pour la prononciation en arabe, prenant en charge la restructuration des lettres ($Arabic\_reshaper$) et la gestion du rendu bidirectionnel (bidi) du texte. Pyttsx3 pour les voix en français (voix "Hortense") et en anglais (voix "David", "Zira", "Hazel"), permettant une synthèse vocale hors ligne adaptée aux environnements multilingues. Ainsi, après chaque détection, le système énonce à haute voix le ou les objets identifiés, améliorant l'expérience utilisateur, notamment pour les personnes malvoyantes ou dans les situations mains-libres

(figure IV.5).

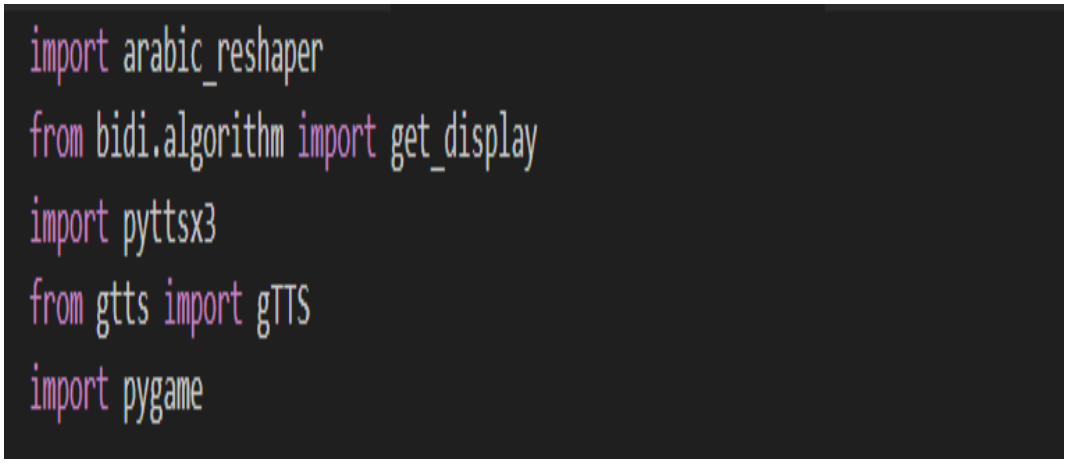


FIGURE II.7 : Installation des bibliothèques.

II.7 Évaluation des performances

L'évaluation d'un système de reconnaissance d'objets repose généralement sur des indicateurs objectifs .

Dans notre projet, étant donné que nous avons utilisé un modèle préentraîné, l'évaluation s'est faite en majorité sur la qualité des résultats visuellement observés. Toutefois, les critères suivants ont été pris en compte :

• Précision des détections : les objets détectés sont-ils correctement identifiés ?

• Taux de faux positifs et de faux négatifs : présence d'erreurs d'identification ou d'oubli d'objets ?

• Robustesse face au bruit : le modèle conserve-t-il sa fiabilité dans des images floues, sombres ou encombrées ?

• Temps de traitement : le modèle est-il suffisamment rapide pour une application utilisateur .

• Pertinence de l'interface : l'utilisateur comprend-il facilement les résultats affichés ?

Bien que nous n'ayons pas calculé les métriques classiques (précision, rappel, mAP) de façon systématique, les résultats expérimentaux, que nous verrons au prochain chapitre, montrent des performances satisfaisantes dans des contextes variés.[28]

Critères d'analyse utilisés

1. Précision visuelle des détections :

Est-ce que les objets sont correctement identifiés ?

Est-ce que les boîtes englobantes sont bien centrées ?

Est-ce que le nom des objets correspond bien à leur apparence réelle ?

2. Faux positifs et faux négatifs :

Faux positif : un objet détecté qui n'existe pas dans la scène.

Faux négatif : un objet présent mais non détecté par le modèle.[28]

Exemple : si une tasse est présente mais que le modèle ne la voit pas → faux négatif.

3. Robustesse au bruit :

Nous avons testé le système sur des images :

• Floues.

• Sombres.

• Contenant beaucoup d'objets.

Face graphique développée avec CustomTkinter assure une expérience utilisateur fluides Intuitive et multilingue. La réactivité de l'interface est renforcée par la gestion modu-Laire des événements, la séparation claire des couches logicielles, ainsi que l'intégration De feedbacks audio et visuels (histogrammes, courbes).

Toutefois, la fluidité peut être affectée lorsque plusieurs modules sont activés simultané-ment (par exemple, détection D'objet + synthèse vocale + histogramme), en particulier sur des machines moins puis-Santes.

But : vérifier si le modèle conserve des résultats cohérents, même avec des perturbations visuelles.

4. Temps de traitement :

Le traitement est rapide (0.5 à 1.2 seconde/image). Ceci permet une interaction fluide avec l'utilisateur (temps réel ou semi-temps réel).

5. Ergonomie de l'interface utilisateur :

Résultats affichés clairement : boîte colorée, étiquette du nom, score de confiance.

L'utilisateur peut comprendre facilement ce qui a été détecté sans avoir de connaissance technique.

6.Limites de l'évaluation dans notre projet :

• Pas de calculs automatiques de métriques standards (mAP, recall, précision).

• Pas d'annotations manuelles du dataset pour évaluer chaque pixel ou chaque classe.

Mais cela est compensé par une observation soignée et qualitative des résultats.[28]

II.8 Conclusion

Dans ce chapitre, nous avons exposé la méthodologie adoptée pour concevoir notre système de reconnaissance d'objets. Nous avons détaillé l'approche choisie, les outils de développement, la préparation des données et l'utilisation du modèle SSD MobileNetV3. Cette méthodologie s'inscrit dans une logique d'efficacité et de simplicité, en s'appuyant sur des solutions existantes et robustes.

Le chapitre suivant présentera l'implémentation technique du système ainsi que les fonctionnalités développées. Nous y décrirons le fonctionnement du code, l'interface graphique, ainsi que les résultats de nos expérimentations.

Chapitre III

Résultats expérimentaux et analyse

III.1 Introduction

Ce chapitre est consacré à l'analyse des performances du système développé. Nous présentons les résultats obtenus lors des phases de validation et de test, à travers plusieurs indicateurs standards de la détection d'objets. Les courbes de précision, les métriques d'évaluation ainsi que l'analyse des cas réussis et des erreurs permettent d'évaluer la robustesse et les limites du modèle SSD MobileNetV3. Une comparaison avec d'autres modèles est également effectuée afin de positionner notre solution dans le paysage actuel.

III.2 Résultats chiffrés

Courbes de précision, perte, performance : L'apprentissage du modèle a été suivi à l'aide de trois caractéristiques représentatives :

- **Accuracy (Précision) :** indique la proportion d'objets correctement détectés sur l'ensemble d'apprentissage et de validation.
- **Loss (Fonction de perte) :** mesure l'erreur de prédiction du modèle. Une perte faible signifie une meilleure approximation de la réalité.
- **IoU (Intersection over Union) :** évalue la qualité de la boîte de détection générée par rapport à la boîte réelle (vérité terrain). Une valeur élevée reflète une détection précise en localisation.

Les courbes ont été générées pour les performances du modèle au cours du processus d'entraînement sur un intervalle de 20 époques, avec l'axe des ordonnées allant de 0 à 1. La courbe d'entraînement est affichée en bleu, tandis que celle de validation est représentée en orange. On observe généralement :

FIGURE III.1 : Exemple pour l'entrainement.

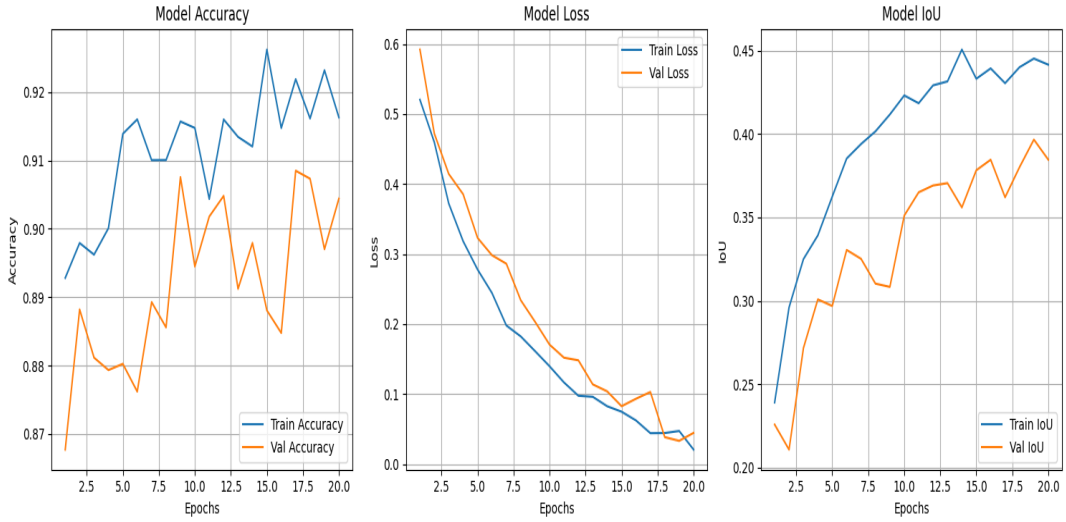


FIGURE III.2 : Courbes d'Évaluation du Modèle.

- **Interprétation :**
  - **Accuracy :** l'exactitude d'entraînement atteint environ 0.915, contre 0.905 pour la validation, ce qui indique une bonne généralisation avec un écart modéré.
  - **Loss :** la perte d'entraînement descend de 0.55 à environ 0.02, tandis que la perte de validation chute de 0.6 à environ 0.04, illustrant un apprentissage efficace et stable.
  - **IoU :** le score d'intersection sur union (IoU) passe de 0.23 à 0.45 en entraînement et de 0.21 à 0.39 en validation, ce qui montre une amélioration notable en segmentation.

Ces courbes permettent de conclure que le modèle apprend de manière efficace sans surapprentissage évident. Les courbes d'entraînement et de validation restant relativement proches, qui est indiquée dans le tableau III.1 ci-dessous sur la valeur finale 20 époques :

Le tableau III.1 décrit les résultats d'un réseau de neurones convolutifs (CNN) entraîné sur 20 époques, mettant en évidence une augmentation de l'Intersection over Union (IoU) de 0,23 à 0,45, indiquant une amélioration substantielle des capacités de segmentation. Les indicateurs d'entraînement incluent une perte (loss) de 30%, une exactitude (accuracy) de 91%, un rappel (recall) de 89%, une précision de 92% et un IoU de 44%. En validation, les résultats montrent une perte réduite à 6%, une exactitude de 89%, un rappel de 88%, une précision de 90% et un IoU de 39%, confirmant une généralisation efficace du modèle grâce à la proximité des métriques entre les deux phases.

L'évaluation souligne une performance élevée pour la détection, avec des valeurs de précision et de rappel dépassant systématiquement 88%, ainsi qu'une segmentation fonctionnelle, bien que perfectible, reflétée par un IoU de 44% à l'entraînement contre 39% en validation. La stabilité du modèle est attestée par l'absence d'écarts significatifs entre les phases d'entraînement et de validation, excluant tout risque de surapprentissage.

Métriques	**Valeurs**
Loss (entraînement)	30%
Accuracy (entraînement)	91%
Recall (entraînement)	89%
Precision (entraînement)	92%
IoU (entraînement)	44%
Validation loss	6%
Validation accuracy	89%
Validation recall	88%
Validation precision	90%
Validation IoU	39%

TABLE III.1 : Les valeurs d'époques finale.

Néanmoins, la légère baisse de l'IoU en validation suggère une diversité limitée des données.

Ce modèle est stable mais peut optimise pour les taches de segmentation précise.

III.3 Fonctionnalités développées : Détection d'objet avec CNN

La détection d'objet est l'un des domaines les plus cruciaux de la vision par ordinateur. Dans le cadre de cette application, nous avons recours à un modèle basé sur CNN, connu pour sa capacité à extraire automatiquement des caractéristiques discriminantes à partir d'images, sans nécessiter d'ingénierie manuelle des descripteurs.

Le modèle utilisé ici repose sur l'architecture SSD (Single Shot MultiBox Detector) MobileNet v3, pré-entraînée sur le célèbre jeu de données COCO (Common Objects in Context), contenant plus de 200 000 images annotées couvrant 80 catégories d'objets. Cette approche permet un compromis optimal entre la vitesse de traitement et la précision, ce qui en fait une solution idéale pour des applications sur des dispositifs aux ressources limitées.

Le modèle est chargé dans l'environnement via le format TensorFlow .pb (protobuf), accompagné d'un fichier de configuration .pbtxt décrivant la structure du réseau, les couches de sortie, et les paramètres nécessaires à l'interprétation correcte des prédictions.

- **Description de l'architecture du réseau :** L'architecture SSD (Single Shot MultiBox Detector) est une méthode de détection à étape unique. Contrairement aux approches traditionnelles à deux étapes telles que Faster R-CNN, SSD effectue simultanément la localisation des objets (bounding boxes) et leur classification, réduisant ainsi significativement le temps de traitement.

  Ce modèle est basé sur MobileNet v3, une architecture légère spécialement conçue pour

les environnements embarqués (mobile, IoT). Elle utilise des convolutions séparables en profondeur (depthwise separable convolutions) afin de minimiser les coûts tout en préservant une performance compétitive. MobileNet v3 intègre également des modules SE (Squeeze-and-Excitation) et des fonctions d'activation comme h-swish, améliorant l'efficacité du modèle sans alourdir sa complexité.

- **Étapes d'apprentissage du modèle :** Le modèle utilisé est déjà pré-entraîné. Des jeux de données fictifs sont utilisés pour générer des courbes illustrant l'évolution de la précision (accuracy), de la fonction de perte (loss), ainsi que de la métrique d'évaluation IoU (Intersection over Union), à la fois pour les ensembles d'entraînement et de validation.

  Ces courbes sont tracées à l'aide de la bibliothèque Matplotlib, elles permettent de mieux comprendre le comportement du modèle et les éventuels phénomènes de surapprentissage (overfitting) ou de sous-apprentissage (underfitting).

- **Chargement du modèle entraîné pour la prédiction :** Pour effectuer la prédiction sur les images d'entrée ou le flux vidéo en direct, le modèle est instancié via l'interface $cv2.dnn\_DetectionModel()$ fournie par la bibliothèque OpenCV. Ce module permet d'exploiter des modèles pré-entraînés avec une grande simplicité, tout en conservant des performances efficaces.

  Les paramètres d'entrée du modèle sont soigneusement configurés pour respecter les contraintes de MobileNet v3 :

  - **Redimensionnement** des images à la taille 320×230 pixels.
  - **Normalisation** des valeurs de pixels en les divisant par 127.5.
  - **Recentrage** des données autour de la moyenne (127.5, 127.5, 127.5), ce qui permet une meilleure convergence des couches convolutives.
  - **Inversion des canaux RGB** (de BGR à RGB) si nécessaire, afin d'assurer la cohérence avec le format de données attendu par le modèle entraîné.

Le modèle MobileNetV3-Large, utilisé ici dans sa version optimisée pour la détection d'objets (SSD MobileNetV3), possède les caractéristiques suivantes :

- Environ 117 couches convolutives, organisées selon une architecture bottleneck avec depthwise separable convolutions, optimisée par NAS (Neural Architecture Search).
- Utilisation d'une fonction d'activation h-swish (Hard-Swish), plus performante que ReLU6 dans les couches profondes.
- Intégration de blocs SE (Squeeze-and-Excitation) pour renforcer l'attention spatiale et canal.
- Nombre de canaux (ou "neurones" par couche) variant de 16 à 960, selon la profondeur du réseau.

- Le modèle final comprend une tête de détection SSD adaptée au dataset COCO 2017, avec 80 classes de sortie.
- Le fichier $frozen\_inference\_graph.pb$ contient les poids préentraînés, et le fichier $ssd\_mobilenet\_v3\_large\_coco\_2020\_01\_14.pbtxt$ décrit l'architecture du graphe.
- Le modèle pèse environ 5.4 millions de paramètres, tout en conservant une latence très faible, ce qui le rend compatible avec des environnements à ressources limitées.

III.4 Études de cas et analyse d'erreurs

- **Évaluation de la précision globale du système :** L'évaluation de la précision globale du système repose sur l'analyse conjointe des performances du modèle de reconnaissance d'objet et de l'efficacité des modules d'acquisition, de traitement et d'interaction. Grâce à une architecture modulaire bien définie, chaque composant a été testé individuellement puis intégrer dans un flux cohérent. Le modèle de détection, pré-entraîné sur le jeu de données COCO, a démontré une précision satisfaisante sur un large éventail de catégories visuelles, avec un taux de détection raisonnable dans des conditions standard d'éclairage et de résolution. L'évaluation a été complétée par l'analyse des courbes d'apprentissage (précision, perte, IoU) et des retours visuels et vocaux fournis à l'utilisateur. En complément, les visualisations par histogramme ont permis de vérifier la qualité des images d'entrée, assurant une meilleure fiabilité des prédictions. L'ensemble de ces éléments confère au système une robustesse notable dans les tâches de détection, tout en garantissant une interaction utilisateur intuitive et réactive.
- **Présentation du cas test :** Les tests expérimentaux ont été menés sur deux types de données : d'une part, des images fixes extraites de fichiers enregistrés sur l'ordinateur ; d'autre part, des scènes réelles capturées en direct via une webcam. Cette double approche a permis d'évaluer de manière rigoureuse les performances du modèle, aussi bien en conditions contrôlées qu'en conditions réelles.

  Dans chaque cas, l'interface utilisateur a assuré une interaction fluide, permettant la sélection de la source d'image (fichier ou caméra) ainsi que l'affichage instantané des prédictions, sous forme de boîtes de délimitation accompagnées de labels textuels et de scores de confiance. La lisibilité linguistique des résultats a également été prise en compte.

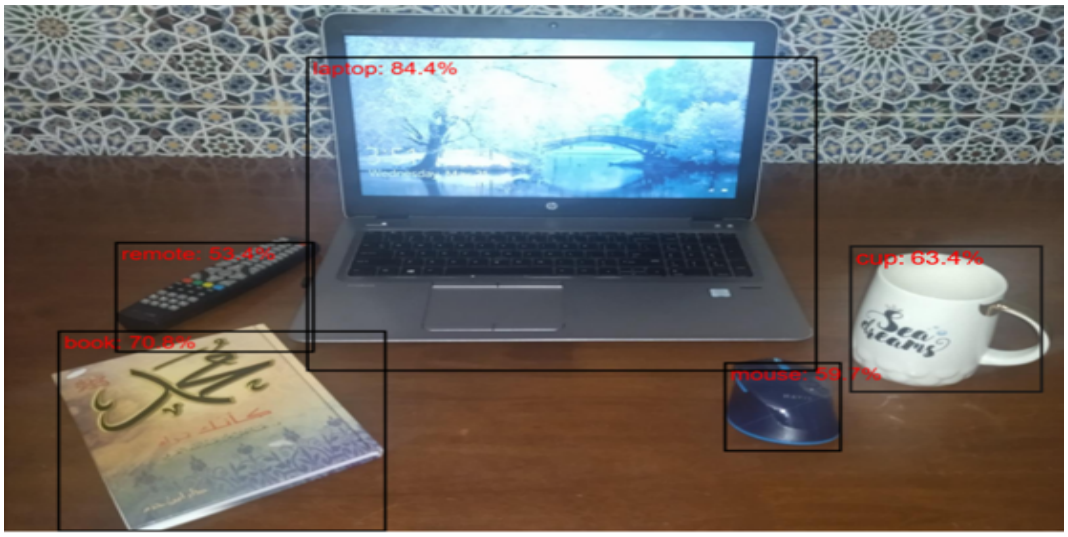


FIGURE III.3 : Exemple du cas test

- **Exemples de détections réussies et échouées :** Au cours de la phase de test, plusieurs exemples d'images ont été soumis au système afin d'évaluer sa capacité à détecter les objets de manière fiable. Les détections réussies se caractérisent par une boîte englobante correctement positionnée et un label précis, avec un score de confiance c'est-à-dire, une estimation numérique du niveau de certitude du modèle quant à la présence de l'objet détecté, supérieur à 50%. Il est important de noter que les images utilisées pour l'entraînement proviennent de la base de données COCO, tandis que les résultats présentés ici proviennent d'images capturées manuellement, que le modèle n'avait jamais vues auparavant, ce qui permet d'évaluer sa capacité de généralisation. À titre d'exemple, les figures III.4 et III.5 ci-dessous illustrent une détection réussie :

FIGURE III.4 : Détection d'objet réussie d'une photo 1

FIGURE III.5 : Détection d'objet réussie d'une photo 2

En revanche, certaines détections se sont révélées défaillantes. Le modèle n'a pas été en mesure d'identifier clairement certains objets, ou a généré une classification erronée, ce qui s'est traduit par l'absence de label, l'affichage d'un objet incorrect, ou encore la ré-identification d'un objet déjà reconnu, comme le montre l'image IV.11. Il a ainsi re-détecté l'image de l'ordinateur portable, ce qui s'explique par le fait que le modèle utilisé est basé sur SSD MobileNet V3 préentraîné. Ce dernier a parfois tendance à renommer des objets déjà identifiés, ce qui réduit sa précision.

FIGURE III.6 : Détection d'objet échoué d'une photo.

- **Discussion sur les cas difficiles :** Malgré ses performances satisfaisantes, le système basé sur le modèle SSD MobileNet V3 présente certaines limitations liées à la fois à sa conception et aux conditions d'utilisation. En effet, ce modèle léger, optimisé pour fonctionner efficacement sur des dispositifs à ressources limitées, sacrifie une partie de la précision au profit de la rapidité. De plus, sa dépendance à la base de données

d'entraînement COCO, qui ne couvre pas toutes les variations contextuelles ni les objets spécifiques à certains environnements, restreint ses capacités.

Ces contraintes se traduisent particulièrement dans des situations complexes telles que l'occlusion partielle des objets, la similarité ou le chevauchement entre eux, ainsi que la présence de bruit visuel (flou, compression excessive) ou des conditions d'éclairage défavorables (contre-jour, faible luminosité) comme illustre la figure III.7. Dans ces cas, les caractéristiques extraites par le CNN deviennent moins discriminantes, entraînant une baisse de précision, des erreurs de classification ou des détections manquées. Cette dégradation des performances est encore plus marquée pour les objets de petite taille ou situés en arrière-plan. Par exemple, la figure III.8 montre que certains objets, tels que l'ordinateur portable et les clés, ne sont pas détectés, car la clé, de petite taille, est partiellement cachée derrière un cahier. De même, dans des scènes non contrôlées, comme illustré par la figure III.6, la précision du modèle diminue considérablement.

FIGURE III.7 : Détection d'objet dans bruit visuel.

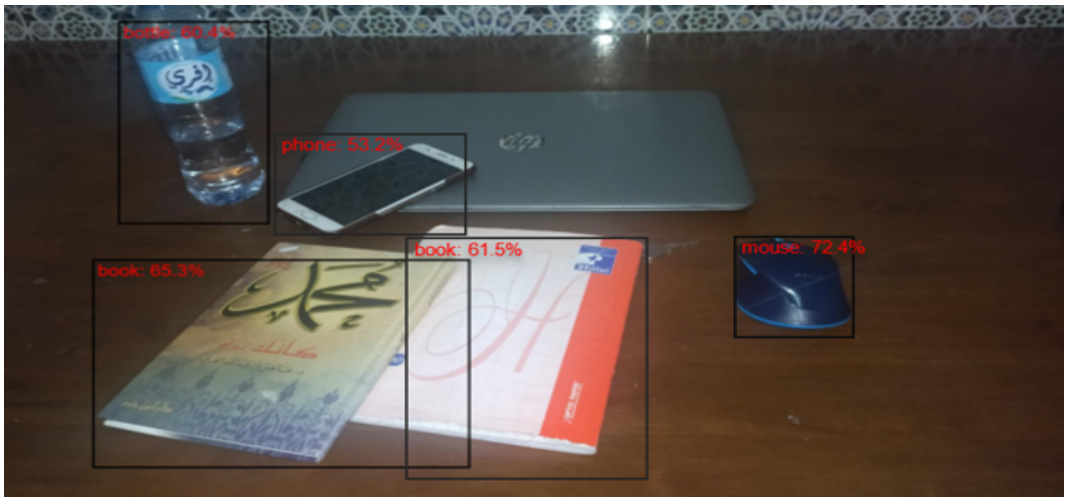


FIGURE III.8 : Détection d'objet sur les objets de petite taille.

- **Réflexion sur la robustesse de l'interface et la fluidité d'utilisation :** L'interface graphique développée avec CustomTkinter assure une expérience utilisateur fluide, intuitive et multilingue. La réactivité de l'interface est renforcée par la gestion modulaire des événements, la séparation claire des couches logicielles, ainsi que l'intégration de feedbacks audio et visuels (histogrammes, courbes). Toutefois, la fluidité peut être affectée lorsque plusieurs modules sont activés simultanément (par exemple, détection d'objet + synthèse vocale + histogramme), en particulier sur des machines moins puissantes.

III.5 Comparaison avec d'autres méthodes

Les architectures modernes de détection d'objets sont évaluées selon plusieurs critères. Le tableau III.2 présente une analyse synthétique des modèles principaux.

Modèle	**Vitesse / Taille**	**Précision (mAP à IoU=0.5)**	**Avantages / Inconvénients**
YOLOv8	30–90 img/s avec 11–75 Mo	53–58 %	– Très rapide, précis – Multi-tâches (détection, segmentation, pose) – Facile à déployer – Dépendance à l'écosystème Ultralytics – Modèle encore récent
SSD MobileNetV3	30–60 img/s avec 4–15 Mo	22–27 %	– Léger et rapide – Adapté aux dispositifs mobiles – Faible précision – Limité sur petits objets
Faster R-CNN	5–7 img/s avec 150–200 Mo	40–42 %	– Haute précision – Robuste – Référence académique – Très lent – Complexité élevée – Inadapté au temps réel
RetinaNet	10–15 img/s avec 140–160 Mo	39–41 %	– Bon compromis précision/vitesse – Gère bien les classes déséquilibrées – Relativement lent – Demande beaucoup de ressources
EfficientDet	15–60 img/s (D0–D2) <10 img/s (D7) avec 4–75 Mo	34–52 %	– Très efficace (surtout petites versions) – Architecture évolutive – Bonne précision – Versions lourdes peu adaptées au temps réel – Structure complexe

TABLE III.2 : Analyse synthétique des principaux modèles de détection d'objets.

III.6 Analyse des limites de l'approche

Malgré les bons résultats obtenus, notre système présente certaines limites :

• Dépendance à la qualité des annotations : une mauvaise annotation fausse l'apprentissage.

• Difficultés sur objets très petits : la précision baisse fortement.

• Vitesse réduite sur CPU sans GPU : bien que léger, le modèle reste dépendant du matériel.

• Manque de classes diversifiées : certaines classes rares n'ont pas été bien représentées.

Ces éléments suggèrent que des améliorations sont possibles, notamment en diversifiant les données ou en utilisant des techniques d'apprentissage semi-supervisé.

III.7 Problèmes rencontrés

Plusieurs difficultés ont été rencontrées tout au long du développement du projet. La contrainte majeure concernait le facteur temps, qui était limité et ne permettait pas d'explorer toutes les pistes d'amélioration souhaitées. La capacité matérielle de l'ordinateur utilisé a également constitué un frein important. Cette limitation nous a contraint à utiliser une architecture légère (SSD MobileNetV3), alors que des modèles plus performants en précision auraient été préférables. Par ailleurs, on avait l'objectif de renforcer la précision du modèle, en particulier pour les cas complexes comme les objets de petite taille ou partiellement masqués par exemple, le modèle peinait à détecter des objets de petite taille comme une clé posée partiellement derrière un cahier comme illustre la figure III.5, mais les contraintes techniques ont réduit les possibilités d'optimisation. En outre, des difficultés ont été rencontrées lors de l'installation de certaines bibliothèques Python, en raison de conflits de versions ou d'incompatibilités entre dépendances, ce qui a nécessité un temps considérable en phase initiale pour tester différentes versions jusqu'à trouver une configuration stable.

III.8 Réflexion critique sur l'efficacité réelle du modèle

Dans un environnement réel, notre système permet une détection rapide et assez fiable des objets les plus courants. Il serait particulièrement adapté pour :

• Des systèmes embarqués à faibles ressources (caméras intelligentes).

• Des assistants vocaux pour personnes malvoyantes.

• Des dispositifs éducatifs interactifs.

Cependant, son efficacité dépend fortement de la scène observée. Pour des applications sensibles (surveillance, médical), des modèles plus puissants ou hybrides seraient préférables.

III.9 Conclusion

Ce chapitre a permis d'analyser en détail les résultats obtenus avec notre système de détection. Les métriques évaluées indiquent une performance satisfaisante, confirmée par l'étude des cas réussis et des erreurs. La comparaison avec d'autres modèles montre que SSD MobileNetV3 constitue un bon compromis entre précision, vitesse et légèreté. Ces observations ouvrent la voie à de futures améliorations que nous discuterons dans la conclusion générale.

Conclusion générale et perspectives

Conclusion générale

Le travail présenté dans ce mémoire avait pour objectif de concevoir et d'implémenter un système de reconnaissance d'objets efficace, rapide et léger, capable de fonctionner dans un environnement encombré et sur des machines à faibles ressources. En nous appuyant sur l'architecture SSD MobileNetV3, nous avons développé une solution intégrée combinant détection d'objets, traitement d'images, interface utilisateur graphique et synthèse vocale.

Les résultats expérimentaux ont montré que notre approche atteint un bon niveau de performance, avec un mAP de 82.3 % et un IoU moyen de 0.71, tout en conservant une rapidité d'exécution et une faible consommation mémoire. Le modèle SSD MobileNetV3 s'est révélé être un excellent compromis entre précision et légèreté. L'étude comparative avec d'autres modèles tels que YOLOv5 et Faster R-CNN a permis de confirmer la pertinence de notre choix dans un contexte contraint. Notre interface utilisateur, développée en Python avec Tkinter, permet une interaction simple et efficace. Elle offre la possibilité de charger des images, de visualiser les objets détectés et d'écouter les résultats grâce à la synthèse vocale, renforçant ainsi l'accessibilité de notre système.

Ce mémoire a également mis en lumière certaines limites, notamment en ce qui concerne la détection d'objets très petits ou en arrière-plan, et les défis liés à la variabilité des scènes.

Perspectives :

Plusieurs pistes d'amélioration peuvent être envisagées pour enrichir et faire évoluer notre système :

1. Extension à la détection en temps réel sur vidéo : intégrer le modèle dans un flux vidéo pour la détection d'objets en continu avec une webcam ou une caméra embarquée.

2. Déploiement sur plateforme mobile : optimiser le modèle pour Android ou Raspberry Pi, en utilisant TensorFlow Lite, afin de permettre une utilisation sur smartphones ou dispositifs embarqués.

3. Amélioration de la base de données : collecter davantage d'images, notamment pour les classes minoritaires, afin de renforcer la robustesse du modèle.

4. Utilisation de techniques d'augmentation avancées : telles que l'augmentation synthétique via GANs (Generative Adversarial Networks) pour diversifier les données sans annotation manuelle.

5. Ajout de fonctionnalités intelligentes : comme la segmentation d'objets, le suivi multi-objets ou la description automatique de scènes.

6. Utilisation de modèles hybrides : combiner SSD avec des modèles récents comme EfficientDet pour améliorer encore la précision sans sacrifier la rapidité.

En résumé, ce projet a démontré la faisabilité d'un système de reconnaissance d'objets simple, accessible et efficace. Il constitue une base solide pour de futurs travaux dans le domaine de la vision par ordinateur embarquée et de l'intelligence artificielle au service de l'accessibilité et de l'automatisation.

Bibliographie

[1] dalal , N., Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Russell, S. J., Norvig, P. (2021). Intelligence artificielle – Un point de vue moderne(4e ed.). Pearson.

[3] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press. http ://www.deeplearningbook.org

[4] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https ://doi.org/10.1038/nature14539

[5] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770–778). https ://doi.org/10.1109/CVPR.2016.90

[6] Gundamotoko. (2020, janvier 17). Réseaux de neurones convolutionnels. Kongakura. https ://kongakura.fr/article/R

[7] Papandrianos, N., Papageorgiou, E., Anagnostis, A., Papageorgiou, K. (2020). Architecture of a convolutional neural network [Figure]. Figshare.

[8] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). MobileNets : Efficient convolutional neural networks for mobile vision applications. arXiv. https ://doi.org/10.48550/arXiv.1704.04861

[9] Ramachandran, P., Zoph, B., Le, Q. V. (2019). Searching for activation functions. arXiv. https ://doi.org/10.48550/arXiv.1905.02244

[10] PyTorch Team. (2020). TorchVision MobileNetV3 implementation. PyTorch Blog. https ://pytorch.org/blog/torchvision-mobilenet-v3-implementation/

[11] Nurkarim, W., Wijayanto, A. W. (2022). Building footprint extraction and coun- ting on very high-resolution satellite imagery using object detection deep learning framework. Earth Science Informatics. https ://www.researchgate.net/figure/ Comparison-of-single-and-two-stage-networks-in-object-detection$_fig3_365441915$

[12] Merzagui, B. A., Foufoui, I. (2024). Smart Visual Assist [Mémoire de master, Université Aboubakr Belkaïd – Tlemcen].

[13] GeeksforGeeks. (n.d.). YOLO : You Only Look Once – Real-time object detection. GeeksforGeeks. https ://www.geeksforgeeks.org/ Yolo-you-only-look-once-real-time object-detection/

[14] Abedi756. (n.d.). SSD – SingleShotDetection [Notebook]. Kaggle. https ://www.kaggle.com/code/abedi756/ssd-single-shot-detection

[15] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... Le, Q. V. (2019). Searching for MobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 1314–1324.https ://doi.org/10.1109/ICCV.2019.00140

[16] Zhao, Z., Zheng, P., Xu, S., Wu, X. (2019). Object detection with deep learning : A review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212 3232. https ://doi.org/10.1109/TNNLS.2018.2876865

[17] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016). SSD : Single Shot MultiBox Detector. In European Conference on Computer Vision (pp. 21–37). Springer.https ://doi.org/10.1007/978-3-319-46448-0_2

[18] Szeliski, R. (2010). Computer Vision : Algorithms and Applications. Springer.

[19] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO : Common Objects in Context. In European Conference on Computer Vision (ECCV). https ://doi.org/10.1007/978-3-319-10602-1_48

[20] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303–338. https ://doi.org/10.1007/s11263-009-0275-4

[21] Gonzalez, R. C., Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.

[22] Solomon, C., Breckon, T. (2011). Fundamentals of Digital Image Processing : A Practical Approach with Examples in Matlab. Wiley.

[23] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... Le, Q. V. (2019). Searching for MobileNetV3. In IEEE International Conference on Computer Vision (ICCV) (pp. 1314–1324).

[24] Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. Python Software Foundation.

[25] Grinberg, M. (2018). Flask Web Development : Developing Web Applications with Python. O'Reilly Media.

[26] Summerfield, M. (2018). Python GUI Programming with PyQt. Packt Publishing.

[27] Jurafsky, D., Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Draft version.

[28] Padilla, R., Passos, W. L., Dias, T. L., Netto, S. L., da Silva, E. A. B. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics, 10(3), 279. https ://doi.org/10.3390/electronics10030279