Université ABBES LAGHROUR Khenchela
Faculté des Sciences et de la Technologie
Département de Génie Industriel
جامـــعة عبـــاس لـــــغرور خـنـشـلة
كليـــة العلـــوم والتكنـــولـــوجيـا
قسـم الهنـدسة الصنـاعية

N° Série :.....

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master

Filière : Télécommunications
Spécialité : Systèmes des Télécommunications

THEME

Implémentation des méthodes d'apprentissage profond (deep learning) sur la carte NVIDIA Jetson Nano en temps réel

Réalisé par : *FRIHA Amel*

DJARAMOU Naziha

Soutenu le : 30/06/2022 ***Devant le jury composé de:***

Dr. KHALFAOUI Mehiou	***Président***	***Université Abbes Laghrour-Khenchela***
Dr. DOUAK Fouzi	***Encadreur***	***Université Abbes Laghrour-Khenchela***
Dr. MEDJALDI Malika	***Examinatrice***	***Université Abbes Laghrour-Khenchela***

Promotion 2021/2022

Je dédie ce modeste travail à:

Mes très chers parents qui m'ont soutenu durant toutes mes études, qui ont fait de moi ce que je suis aujourd'hui je leurs saurais éternellement reconnaissante.

A mes chers frères : Anouar et Oussama

A ma chère seule amie : Farida

A mon binôme Naziha et toute sa famille

Amel

Je dédie ce travail à ma chère mère, dont la présence à mon côté est la raison de mon bonheur dans cette vie

À ma famille

À tous mes professeurs

À ma chère amie qui on a travaillé ensemble sur ce mémoire de fin d'étude

À toutes les bonnes personnes que j 'ai rencontrées dans ma vie et à tous mes proches.

Naziha

Remerciements

*Nous remercions avant tout, **ALLAH** le tout puissant, de nous avoir donné la force et la volonté de réaliser à terme ce projet de fin d'études.*

*Nous adressons nos plus profonds remerciements, toute notre gratitude et notre reconnaissance à notre encadreur : Dr. **Douak Fouzi**, pour nous avoir dirigés durant ce travail et nous avoir permis de le réaliser dans les meilleures conditions. Nous tenons particulièrement à le remercier de la liberté d'action qu'il nous a accordée à chaque étape du mémoire. Nous espérons avoir été dignes de la confiance qu'il nous a accordée et que ce travail est finalement à la hauteur de son espérance.*

*Nous adressons nos plus sincères remerciements à Monsieur Dr. **Khalfaoui Mehiou**, enseignant à l'université de Khenchela, pour nous avoir fait l'honneur de présider le jury de notre mémoire. Nous remerciements vont aussi à Madame Dr. **Medjaldi Malika**, enseignante à l'université de Khenchela, qui a accepté d'examiner ce travail.*

*Nous n'oublions pas aussi de remercier chaleureusement notre enseignant et le responsable de la spécialité télécommunications Pr. **Bedra Sami,** ainsi tous nos enseignants qui nous permettent d'acquérir plus de connaissances durant notre parcours.*

Nous voudrions exprimer notre reconnaissance envers nos familles et amis qui nous ont apportés leur support moral et intellectuel tout au long de notre cursus.

Résumé

Les progrès réalisés dans le domaine de l'intelligence artificielle, en particulier des ‘‘réseaux de neurones profonds’’ ont encouragé le développement des différents systèmes de sécurité. L'opération de reconnaissance de formes constitue l'une des applications les plus connues dans l'intelligence artificielle. L'application d'IA est donc bien destinée à assurer la reconnaissance et la classification des objets.

Ce mémoire présente les travaux menés dans le cadre du domaine de la vision par ordinateur. Notre objectif est de développer des systèmes fonctionnant de façon indépendante et en temps réel dans les systèmes de sécurité intelligents comme les systèmes de surveillance et la sécurité routière. Pour cela, nous avons mis en place des méthodes de l'intelligence artificielle présentées en deux phases : Dans un premier temps, nous avons présenté la méthode de V-J (Viola and Jones) pour la reconnaissance faciale, ensuite nous avons implémenté et évalué les deux réseaux profonds YOLOv3-tiny et YOLOv4-tiny pour la détection et la reconnaissance de masques et de panneaux de signalisation. Cette phase nous a permet de comparer les deux modèles de la famille YOLO et constaté que YOLOv4-tiny est le modèle le plus performant. Dans un second temps nos approches étant validées à travers différentes expérimentations sur des images et des vidéos en temps réel en utilisant une unité puissante de calcul graphique «la carte NVIDIA Jetson Nano» qui permet d'accélérer les calculs pour le traitement des images.

Mots clés : Intelligence artificielle, systèmes de sécurité, NVIDIA Jetson Nano, détection et reconnaissance d'objets.

Abstract

The progress made in the field of artificial intelligence, in particular ‘‘deep neural networks’’, has encouraged the development of various security systems. The pattern recognition operation is one of the best known applications in artificial intelligence. The application of AI is thus well intended to ensure the recognition and classification of objects.

This project presents the work carried out in the field of computer vision. Our goal is to develop systems that work independently and in real time for the intelligent security systems such as surveillance systems and road safety. For this purpose, we have implemented artificial intelligence methods presented in two phases: First, we presented the V-J (Viola and Jones) method for face recognition, then we implemented and evaluated the two networks YOLOv3-tiny and YOLOv4-tiny for mask detection and traffic sign recognition. This phase allowed us to compare the two models of the YOLO family and found that YOLOv4-tiny is the better performing model. In a second phase, our approaches were validated through different experiments on images and videos in real time using a powerful graphics processing unit "the NVIDIA Jetson Nano" which allows to accelerate the calculations for the processing of images.

Keywords: Artificial intelligence, security systems, NVIDIA Jetson Nano, Object detection and recognition.

ملخص

شجعت التطورات في مجال الذكاء الاصطناعي ، ولا سيما "الشبكات العصبية العميقة" على تطوير أنظمة أمنية مختلفة. يعد التعرف على الأنماط أحد أشهر التطبيقات في مجال الذكاء الاصطناعي. لذلك فإن الغرض من تطبيق الذكاء الاصطناعي هو ضمان التعرف على الأشياء وتصنيفها.

تقدم هذه المذكرة العمل المنفذ في مجال الرؤية الحاسوبية. هدفنا هو تطوير أنظمة تعمل بشكل مستقل وفي الوقت الحقيقي لتطوير أنظمة الأمن الذكية مثل أنظمة المراقبة وتحقيق السلامة في الطرق. لهذا الغرض ، قمنا بتنفيذ طرق الذكاء الاصطناعي على مرحلتين: أولاً ، قدمنا طريقة V-J للتعرف على الوجه، ثم قمنا بتنفيذ وتقييم الشبكتين YOLOv3-tiny و YOLOv4-tiny للكشف والتعرف على الأقنعة وإشارات المرور. سمحت لنا هذه المرحلة بمقارنة نموذجي عائلة YOLO ووجدنا أن YOLO v4-Tiny هو النموذج الأكثر كفاءة. في الخطوة الثانية، يتم التحقق من صحة مناهجنا من خلال تجارب مختلفة على الصور ومقاطع الفيديو في الوقت الفعلي باستخدام وحدة معالجة قوية "بطاقة NVIDIA Jetson Nano" التي تسرع العمليات الحسابية لمعالجة الصور.

الكلمات المفتاحية: الذكاء الاصطناعي، أنظمة الأمان، NVIDIA Jetson Nano، اكتشاف و التعرف على الاجسام.

Liste des tableaux

Liste des figures

Liste des abréviations

IA: Intelligence artificielle.

ML: Machine learning.

DL: Deep learning.

NN: Neural network.

ANN: Artificial neuron network.

MLP: Multilayer perception.

CNN: Convolutional neural network.

ResNet: Residual neural network.

VGG: Visual Geometry group.

V-J: Viola and Jones.

Relu: Rectified linear unit.

GPU: Graphics processing unit.

CPU: Central processing unit.

SSD: Signal Short Detector.

GPIO: General-Purpose Input/Output.

OS: Operating System.

HDMI: High-Definition Multimedia Interface.

LAN: Local Area Network.

SD Card: Secure Digital Card.

USB: Universal Serial Bus.

SSH: Secure Shell.

SPI: Serial Peripheral Interface.

SVM: Support Vector Machine.

IEEE: Institute of Electrical and Electronics Engineers.

IP: Internet Protocol.

FN: False Negative (Faux négatif).

FP: False Positive (Faux positif).

RF: Random Forest.

PCA: Principal Component Analysis.

ILSVRC: ImageNet Large-Scale Visual Recognition Challenge.

TSR: Traffic Signs Recognition.

SCK: SPI Serial Clock.

FC: Fully Connected.

RN: Réseaux de neurones.

GND: Ground.

IP: Internet Protocol.

COVID-19: 'CO' stands for corona, 'VI' for virus, 'D' for disease and '19' for the year 2019.

IOS: iPhone Operating System.

CSI: Channel-State Information.

FSPL: free-space path loss.

Open CV: Open Source Computer Vision.

UART: Universal Asynchrones Receiver Transmitter.

Table des matières

Chapitre II

La carte de développement NVIDIA Jetson Nano

Chapitre III

Domaines d'application de l'apprentissage profond

Chapitre IV

Implémentation des algorithmes sur la carte Jetson Nano

Introduction générale

Introduction générale

L'intelligence artificielle (IA) est une technologie capable de produire des résultats similaires à ceux issus du cerveau humain. Il s'agit d'un outil informatique qui effectue des actions ou exécute des tâches. Cet outil repose notamment sur des algorithmes, c'est-à-dire des suites de formules mathématiques et de traitements statistiques [1].

La recherche en IA a permis de réaliser d'importants progrès dans la dernière décennie, et ce dans différents domaines: mathématiques, médecine, la pandémie de la Covid19, aérospatiale, militaire, télécommunication, etc. Les avancées les plus connues sont celles réalisées dans l'apprentissage automatique, grâce notamment au développement d'architectures d'apprentissage profond, des réseaux de neurones convolutifs multicouche dont l'apprentissage s'opère à partir de gros volumes de données sur des architectures de calcul intensif [2].

La vision par ordinateur est l'un des outils participants à une transition vers une évolution énorme dans les systèmes de sécurité intelligents comme les systèmes de surveillance et la sécurité routière, ces derniers requiert des moyens d'analyse afin de détecter des besoins, et des moyens d'action pour traiter les zones concernées. Ils peuvent tout à fait être fournis par l'humain au travers de son expérience dans son travail, éventuellement assisté d'outils. Néanmoins, le numérique et les évolutions récentes des matériels de calcul constituent une opportunité d'automatisation des tâches différentes plus spécifiquement, les avancées réalisées dans le domaine de la vision par ordinateur accélèrent la robotisation de nombreuses tâches dans le domaine de sécurité.

L'apprentissage profond, un sous-domaine de l'intelligence artificielle qui exploite l'apprentissage automatique via les réseaux de neurones, regroupe des méthodes algorithmiques démontrant un fort potentiel en vision par ordinateur qui commencent à être exploitées pour la sécurité. Ainsi, de nombreux domaines font l'objet de travaux de recherche et de développement dans cette direction.

La majorité des détecteurs d'objets basés sur CNN (Convolutional Neural Network) sont en grande partie applicables uniquement aux systèmes de recommandation. Par exemple la recherche de places de parking libres via des caméras vidéo urbaines et exécutée par des modèles lents et précis, alors que l'avertissement de collision de voitures est lié à des modèles rapides et imprécis. L'amélioration de la précision des détecteurs d'objets en temps réel permet de les utiliser non seulement pour les systèmes de recommandation générant des indices, mais aussi pour la gestion de processus autonomes et la réduction de l'intervention

humaine. Le fonctionnement des détecteurs d'objets en temps réel sur des unités de traitement graphique (GPU) conventionnelles permet leur utilisation massive à un prix abordable.

L'objectif de ce mémoire est d'étudier les éléments nécessaires à la réalisation des systèmes intelligents en temps réel utilisés pour des mesures de sécurité, ces systèmes sont examinés sur un appareil embarqué appelé NVIDIA Jetson Nano. Ce travail représente, le développement et l'implémentation de certains algorithmes d'apprentissage automatique dans trois domaines différents : (*i*) détection et reconnaissance faciale (application dans les systèmes de surveillance), (*ii*) détection du visage et port du masque (application pour Covid-19), (*iii*) détection des panneaux de signalisation (application dans les systèmes de transport intelligents).

Notre travail s'est inspiré de plusieurs travaux de recherche. Dans le domaine de la détection et la reconnaissance de visage, différentes approches ont été développées pour résoudre ce problème dans différents environnements et conditions [3, 4]. En 2004 avec la publication de la méthode de Viola et Jones [5], la première méthode capable à détecter des visages en temps réel. La méthode devient standard est reprise et améliorée par de nombreux chercheurs, telles que celles basées sur la représentation sparse [6], YOLO [7], CNN [8], etc. Dans ce contexte, plusieurs systèmes de sécurité ont été proposés dans la littérature scientifique, on peut citer les articles [9-12].

La deuxième application présentée dans notre travail, est l'implémentation des algorithmes d'IA sur la carte NVIDIA pour la détection de masque. Comme pour les systèmes de surveillance, ou de détection de visages, la question du contrôle automatisé du port du masque se pose. D'un point de vue technique, une véritable résolution s'est opérée ces dernières années notamment dans le domaine de la reconnaissance d'objets grâce à l'apprentissage profond [13-16].

D'autre applications intelligentes issues de l'avancé de l'intelligence artificielle est créditée de changer la vie ; notamment en véhicules autonomes. Une des applications vise à renforcer la prévention routière, et à munir nos voitures et nos routes de capacités permettant de rendre la route plus sécurisé (les informations sur le trafic, les accidents, les dangers, les déviations possibles et les informations météorologiques, …, etc.). Ce type d'application permet de définir le concept des systèmes de transports intelligents (ITS, Intelligent Transportation System), son rôle est d'améliorer la sécurité, l'efficacité et la convivialité dans les systèmes de transport routier, à travers l'utilisation des nouvelles technologies de l'IA [17]. Dans notre travail, nous avons réalisé une application liée à la sécurité du trafic.

Il existe plusieurs types d'applications selon l'information à transmettre et selon le contexte : l'application d'avertissement de la violation des feux de signalisation, l'application d'avertissement à l'approche d'un tournant, l'application d'avertissement coopératif d'une collision imminente, l'application d'avertissement de changement de voie, etc. En effet lorsqu'un véhicule reçoit une alerte ou un avertissement, le conducteur du véhicule peut réagir efficacement pour éviter un accident ou une situation dangereuse [18].

Le but commun de toutes ces applications est d'obtenir une meilleure identification de notre objet (visage, masque, panneaux de signalisations routière). Ce sont toutes ces considérations qui ont guidé ce travail. Pour résoudre cette problématique d'identification ou de classification, nous utiliserons des réseaux de neurones qui font une grande partie dans l'évaluation des recherches scientifiques, il en découle plusieurs sous-problématiques que nous décrivons dans ce qui suit :

- ✓ Les modèles des réseaux neurones doivent entrainer d'une manière suffisante et rapide.
- ✓ Les réseaux neuronaux modernes les plus précis ne fonctionnent pas en temps réel et nécessitent un grand nombre de GPU pour l'entraînement avec une grande taille de mini-batch.

Notre travail est divisé en deux parties : la première est de trouver les meilleurs paramètres permettant à nos modèles de classifier et d'identifier nos objets hors ligne (simulation), la seconde est de pouvoir évaluer les performances de ces modèles. Nous choisissons la carte NVIDIA Jetson Nano comme unité de traitement en temps réel, sur la base des paramètres obtenus dans la première phase.

Ce mémoire explore l'utilisation d'algorithmes de vision par ordinateur et d'apprentissage profond pour des applications en sécurité et en surveillance. Il nécessite des algorithmes d'analyse d'images permettant d'obtenir des informations très localisées sur les objets à traiter. Ce travail propose plusieurs approches de vision par ordinateur et d'apprentissage profond permettant d'une part de reconnaitre la personne dans des images ou un vidéo streaming et autres applications : détecter le masque sur le visage d'une personne et reconnaitre les panneaux de signalisation routière dans des séquences d'images ou vidéo en temps réel.

Afin d'atteindre le but souligné de notre travail, on a divisé ce mémoire en quatre (4) chapitres de la manière suivante :

Le premier chapitre ‘‘*Intelligence artificielle*’’, donne des définitions de la terminologie et des concepts du monde de l’apprentissage automatique, d’intelligence artificielle, les réseaux de neurones et leurs types avec explication; et nous abordons aussi les domaines d’application des systèmes utilisant l’IA.

Le deuxième chapitre ‘‘*La carte de développement NVIDIA Jetson Nano*’’, où nous abordons la définition de la carte NVIDIA Jetson Nano ainsi que sa configuration détaillée pour l’installation de son système d’exploitation. On explique aussi la composition matérielle de la carte, de ses entrées/sorties GPIO, caméras, carte micro-SD, etc. Puis on explique ce que Raspberry Pi et sa composition générale. Enfin, on parle de l’aspect logiciel en expliquant le langage de programmation *Python* et ses bibliothèques servant au développement des différents types des systèmes de sécurité.

Le troisième chapitre ‘‘*Domaines d’application de l’apprentissage profond”*, sera consacré pour l’explication de la méthode de Viola et Jones qu’on va exploiter dans la réalisation de système de reconnaissance faciale. Puis, l’implémentation des modèles pré-entrainés en utilisant l’apprentissage par transfert pour la détection de masque et la reconnaissance des panneaux de signalisation routière.

Le quatrième chapitre ‘‘*Implémentation des algorithmes sur la carte Jetson Nano*’’, exposera la réalisation des trois systèmes en utilisant l’intelligence artificielle et la carte NVIDIA en temps réel.

Enfin, la dernière partie de ce mémoire est consacrée à la conclusion générale et quelques perspectives.

Chapitre I

Intelligence artificielle

I.1. Introduction

L'intelligence artificielle (IA) étudie les moyens de créer des programmes et des machines intelligentes capables de résoudre des problèmes de manière créative, ce qui a toujours été considéré comme une prérogative humaine. L'IA vise à simuler la façon dont le cerveau humain prend des décisions. Donc implique la mise en œuvre d'un certain nombre de technologies pour permettre aux machines de simuler une forme d'intelligence réelle. L'IA comprend divers sous-domaines, dont les plus connus sont l'apprentissage automatique et l'apprentissage profond [19].

Le *Machine Learning* (apprentissage automatique) et le *Deep Learning* (apprentissage profond) sont les deux concepts les plus importants qui rendent l'intelligence artificielle possible [20]. On confond bien souvent ces deux termes, alors qu'ils désignent deux méthodes bien distinctes employées dans des champs d'application différents (vision par ordinateur, traitement du langage naturel, robotique, 5G, etc.).

L'apprentissage automatique (ML) est une branche de l'IA qui applique systématiquement des algorithmes pour synthétiser les relations sous-jacentes entre les données et les informations. Par exemple, les systèmes d'apprentissage automatique peuvent être entraînés sur des systèmes de reconnaissance automatique de la parole pour convertir les informations acoustiques d'une séquence de données vocales en structure sémantique exprimée sous la forme d'une chaîne de mots [21].

L'apprentissage profond utilise des réseaux neuronaux artificiels pour effectuer des calculs sophistiqués sur de grandes quantités de données. Il s'agit d'un type d'apprentissage automatique qui fonctionne sur la base de la structure et du fonctionnement du cerveau humain. Les algorithmes d'apprentissage profond forment les machines en apprenant à partir d'exemples. Des industries telles que les soins de santé, le commerce électronique, le divertissement et la publicité utilisent couramment l'apprentissage profond [22].

I.2. Types d'apprentissage automatique (ML)

I.2.1. Apprentissage supervisé

Où l'algorithme génère une fonction qui associe les entrées aux sorties souhaitées. Une formulation standard de la tâche d'apprentissage supervisé est le problème de classification : l'apprenant doit apprendre (ou approcher le comportement) d'une fonction qui fait correspondre un vecteur à l'une de plusieurs classes en examinant plusieurs exemples d'entrée-sortie de la fonction. La Figure (I.1) donne un exemple d'apprentissage supervisé.

Techniques d'apprentissage supervisé : Régression linéaire, Régression logistique, CART (Classification and Regression Tree), Naïve Bayes, KNN et apprentissage profond [21].

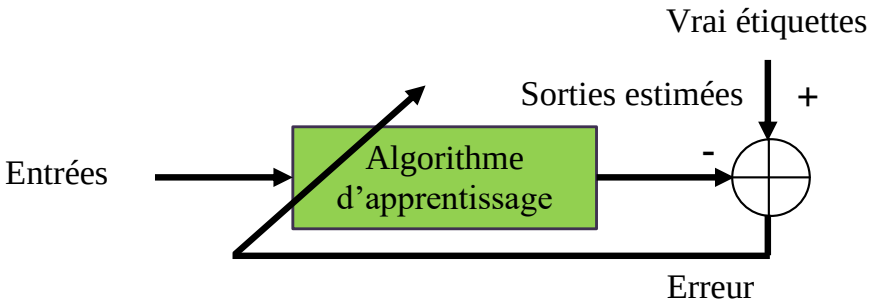


Figure I. 1. Apprentissage supervisé.

On distingue deux types de problèmes auxquels l'apprentissage supervisé est appliqué, comme représente la Figure suivante [23]:

Classification :

La classification est un type de Machine Learning supervisé dans lequel un algorithme «apprend» à classifier de nouvelles observations à partir d'exemples de données étiquetées.

Régression :

Les modèles de régression décrivent la relation entre une variable réponse (sortie) et une ou plusieurs variables prédictives (entrée). Ici on n'attribue pas une classe mais une valeur mathématique : un pourcentage ou une valeur absolue.

Classification binaire:

Où les données entrantes sont classées dans l'une des deux catégories possibles.

Classification multiclasse :

Dont les données sont divisées en plusieurs catégories possibles supérieures à deux. Ce type de classification est très utile pour la grande masse de données.

Figure I. 2. La classification et la régression.

I.2.1.1. Phase d'entraînement

Dans cette phase l'algorithme d'apprentissage reçoit en entrée des exemples d'apprentissage (les documents d'entraînement) étiquetés et produit un modèle de prédiction le plus

performant possible, c'est-à-dire le modèle qui produit le moins d'erreurs en prédiction comme présente la Figure I.3.

La prédiction est une tâche qui vise à prédire une ou plusieurs caractéristiques inconnues à partir d'un ensemble de caractéristiques connues. Par exemple : prédire la qualité d'un client en fonction de son revenue et de son nombre d'enfant [23].

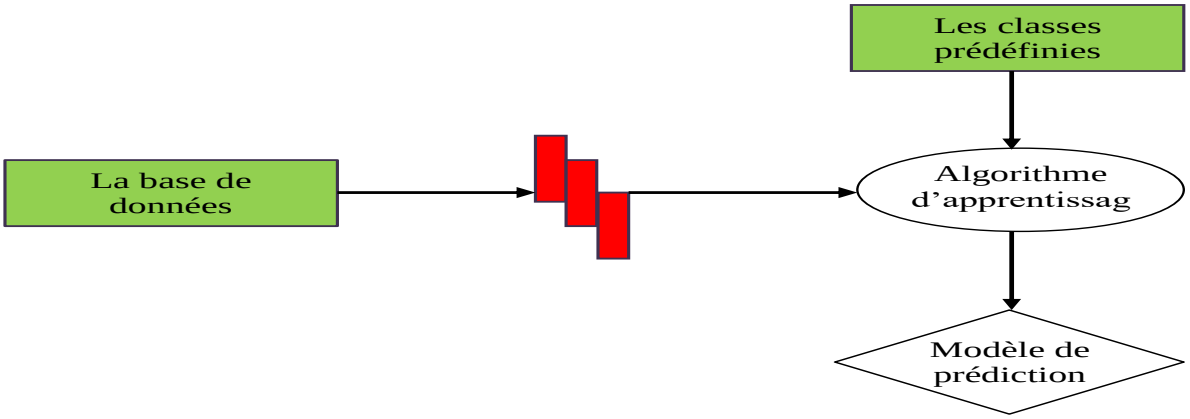


Figure I. 3. La phase d'apprentissage.

I.2.1.2. Phase de test

Dans cette phase le modèle obtenu lors de la phase d'apprentissage doit être capable de prédire l'étiquette d'un nouvel exemple en fonction des valeurs d'entrées, selon la Figure I.4.

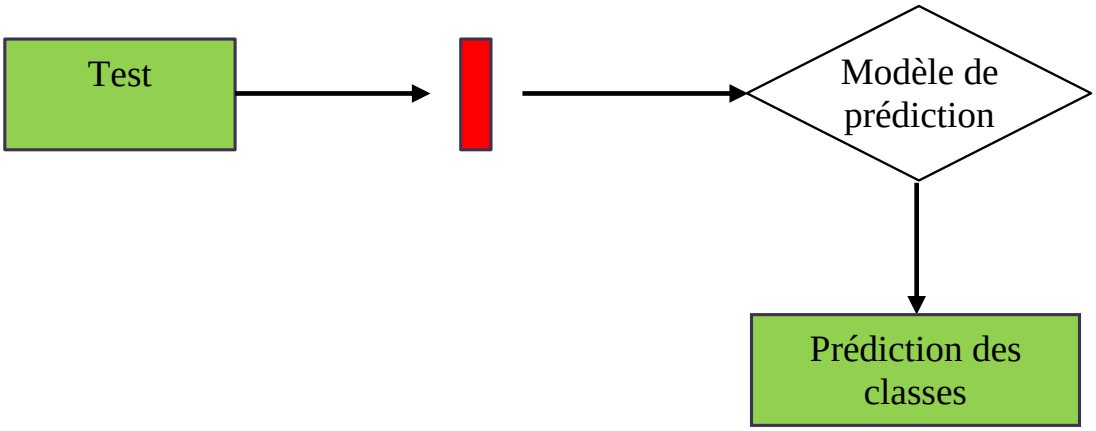


Figure I. 4. La phase de test.

I.2.2. Apprentissage non supervisé

Le modèle apprend par l'observation (X) et trouve des structures dans les données (Y). Une fois que le modèle reçoit un ensemble de données, il trouve automatiquement des modèles et des relations dans l'ensemble de données en créant des clusters, comme il est représenté dans la Figure I.5.

Techniques d'apprentissage non supervisé : Apriori, K-means, SVM et PCA [21].

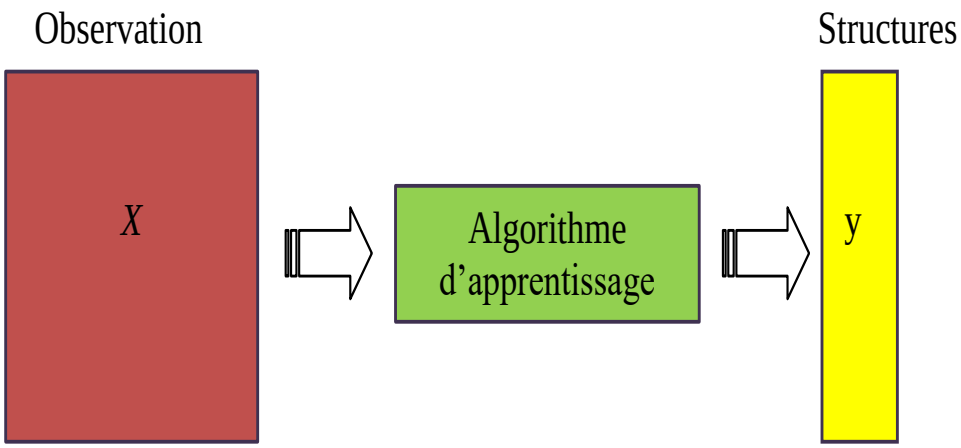


Figure I. 5. Apprentissage non supervisé.

L'apprentissage non supervisé comprend deux catégories d'algorithmes :

Algorithmes de regroupement et d'association.

Quelques méthodes qu'on peut citer en classification non supervisé :

- K-moyennes.
- Nuées dynamiques.
- Classification ascendante hiérarchique (analyse des données).

I.2.3. Apprentissage semi-supervisé

Cette approche d'apprentissage consiste à combiner les deux approches citées précédemment, elle combine une petite quantité de données étiquetées (exemple d'entrée et de sortie), avec une grande quantité de données non étiquetées. Cette technique d'apprentissage a permis à Amazon d'améliorer sensiblement les performances de l'assistant personnel intelligent : d'Alexnet [24].

I.2.4. Apprentissage par renforcement

Dans ce mode d'apprentissage le réseau va maximiser un indice de performance qui lui est fourni, appelé signal de renforcement. Le système étant capable ici, de savoir si la réponse qu'il fournit est correcte ou non, mais ne connaissant pas la bonne réponse, contrairement à l'apprentissage supervisé [25].

I.2.5. Apprentissage par transfert

L'apprentissage par transfert est une méthode d'apprentissage automatique dans laquelle un modèle développé pour une tâche est réutilisé comme point de départ d'un modèle sur une deuxième tâche. La Figure I.6, représente un transfert extrait de "transfert de l'apprentissage". C'est une approche populaire dans l'apprentissage en profondeur où des modèles pré-entraînés sont utilisés comme point de départ pour les tâches de vision par ordinateur et

d'apprentissage automatique étant donné les vastes ressources de calcul et de temps nécessaires pour développer des modèles de réseaux neuronaux sur ces problèmes et des énormes sauts de compétences [26].

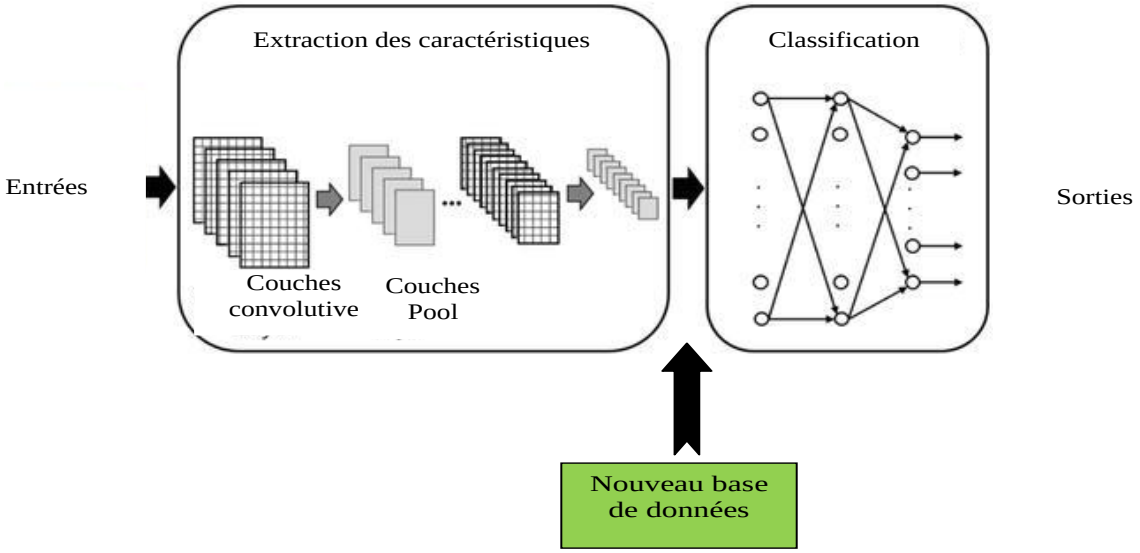


Figure I. 6. Représentation du transfert inductif extrait de ''transfert de l'apprentissage''.

I.3. Comparaison machine learning et deep learning

Tableau I. 1. Comparaison générale entre machine learning et deep learning.

Machine Learning	**Deep Learning**
L'apprentissage automatique utilise des algorithmes pour analyser les données, apprendre de ces données et prendre des décisions éclairées en fonction de ce qu'il a appris	L'apprentissage profond structure les algorithmes en couches pour créer un réseau neurone capable d'apprendre et de prendre des décisions intelligentes par lui-même
Peut s'entrainer avec moins de données d'entrainement	Nécessite de grands ensembles de données pour la formation
Prend moins de temps pour s'entrainer	Prend plus de temps pour s'entrainer
Trains sur CPU	S'entraine sur GPU pour une formation appropriée
La sortie est sous forme numérique pour les applications de classification et de notation	La sortie peut être sous n'importe quelle forme, y compris des éléments de forme libre tels que du texte et du son libres
Capacité de réglage limitée pour le réglage des hyper paramètres	Peut être réglé de différentes manières
Besoin de comprendre les fonctionnalités qui représentent les données	Pas besoin de comprendre la meilleure Caractéristique qui représente les données

I.4. Réseaux de neurones artificiels

Un réseau de neurones peut être considéré comme un modèle mathématique de traitement réparti, composé de plusieurs éléments de calcul non-linéaires (les neurones), opérant en parallèle et connectés entre eux par des poids.

I.4.1. Réseau multicouches (MLP)

Dans un perceptron multicouche représenté dans la Figure I.7, il faut tenir compte de l'influence de plusieurs couches dans le calcul du gradient. Les différentes étapes de cet algorithme sont citées ci-dessus :

1. Initialisation des poids à des valeurs aléatoires de faible grandeur,
2. Propager les entrées x_n à travers le réseau,
3. Pour chaque couche cachée (k), calculer l'évolution des poids au cours d'une itération :

$$\Delta w_{j,i}^{k} = -2\mu \sum_{l=1}^{N} e_{j,l}^{k} f'(w^{k} - y_{l}^{k-1}) y_{l}^{k-1} \quad \text{(I.1)}$$

4. Mettre à jour chaque poids synaptique du réseau

$$w_{j,i}(k+1) = w_{j,i}(k) + \Delta w_{j,i}(k) \quad \text{(I.2)}$$

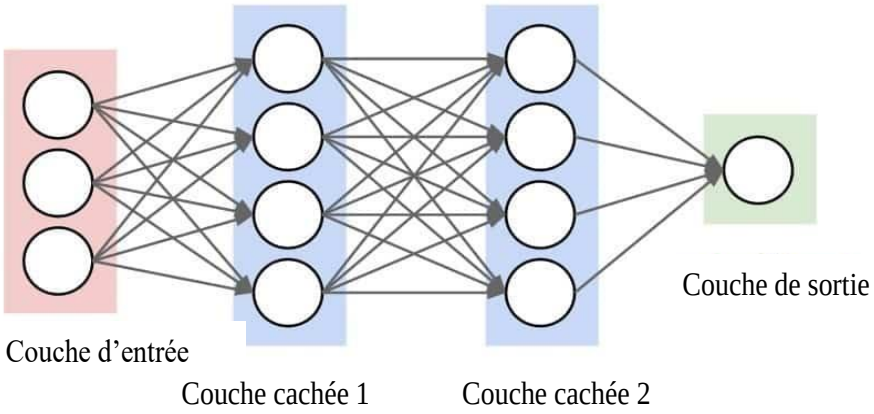


Figure I. 7. Architecture de perceptron multicouche.

I.5. Réseaux de neurones convolutifs

Les réseaux convolutifs sont une forme particulière de réseau neuronal multicouche dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères.

Leur conception suit la découverte de mécanismes visuels dans les organismes vivants. Ces réseaux de neurones artificiels sont capables de catégoriser les informations des plus simples aux plus complexes. Ils consistent en un empilage multicouche de neurones, des fonctions mathématiques à plusieurs paramètres ajustables, qui prétraitent de petites quantités

d'informations. Les réseaux convolutifs illustrés dans la Figure I.8, sont caractérisés par leurs premières couches convolution (généralement une à trois). Une couche convolutive, est basée comme son nom l'indique sur le principe mathématique de convolution, et cherche à repérer la présence d'un motif (dans un signal ou dans une image par exemple).

Pour une image, la première couche de convolution peut détecter les contours des objets (par exemple un cercle), la seconde couche de convolution peut combiner les contours en objets (par exemple une roue), et les couches suivantes (non nécessairement convolution elles) peuvent utiliser ces informations pour distinguer une voiture d'une moto. Une phase d'apprentissage sur des objets connus permet de trouver les meilleurs paramètres en montrant par exemple à la machine des milliers d'images d'un chien, d'une voiture ou d'un sport... L'un des enjeux est de trouver des méthodes pour ajuster ces paramètres le plus rapidement et le plus efficacement possible. Les réseaux neuronaux convolutifs ont de nombreuses applications dans la reconnaissance d'images, de vidéos ou le traitement du langage naturel.

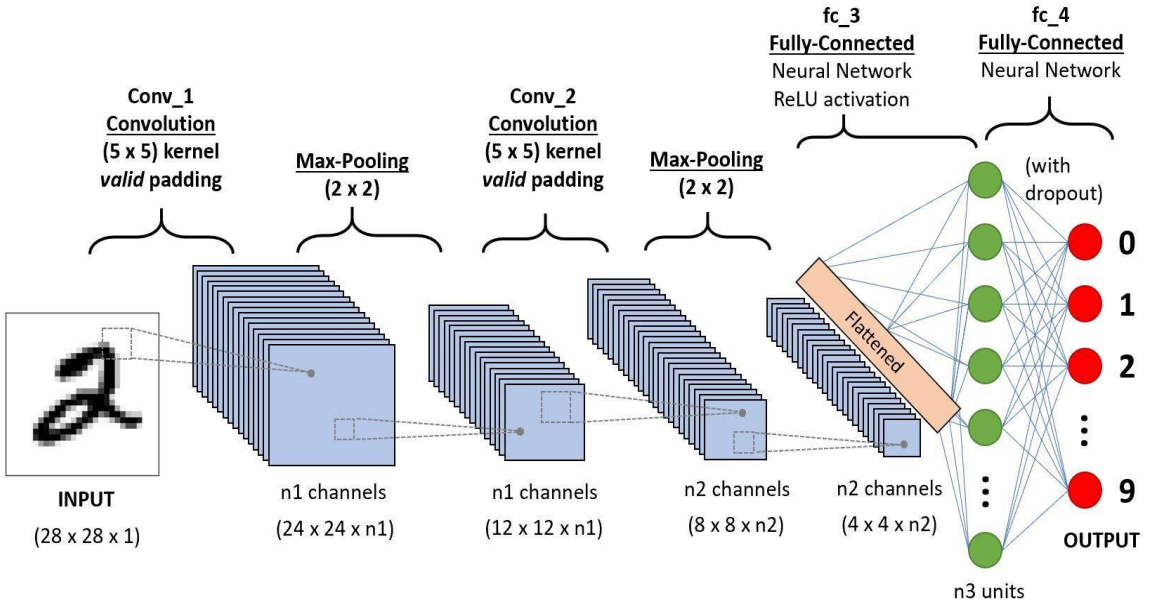


Figure I. 8. Réseau de neurones convolutifs.

I.5.1. Principe architectural d'un CNN

Un réseau neuronal convolutif n'est pas simplement un réseau neuronal profond comportant de nombreuses couches cachées. Il s'agit plutôt d'un réseau profond qui simule le fonctionnement du cortex visuel du cerveau pour reconnaître et classer des images ou des vidéos, et pour découvrir un objet ou même une partie d'une image. Le concept et le fonctionnement des réseaux neuronaux convolutifs sont différents des autres réseaux neuronaux, en effet un réseau neuronal convolutif a deux parties distinctes avec en entrée une

image sous la forme d'une matrice de pixels bidimensionnelle (à 2 dimensions, noir et blanc), ou une image couleur à 3 dimensions (couleurs : rouge, vert et bleu).

La première partie d'un réseau neuronal convolutif est la partie convolutive, qui sert à extraire les caractéristiques de l'image. Ensuite, l'image passe par le fichier de séquence de filtres, ou le noyau de bobinage, qui conduit à la création d'une nouvelle image appelée cartes de convolution. En général, les filtres intermédiaires réduisent la résolution de l'image. Ensuite, les cartes de caractéristiques sont aplaties en un vecteur de caractéristiques pour former les données d'entrée de la partie de la couche entièrement connectée. Le rôle principal de cette couche (entièrement connectée) est de combiner les caractéristiques contenues dans le vecteur de son entrée pour la classification des images. La sortie du CNN sera constituée de neurones qui représentent un neurone par classe et les valeurs obtenues sont généralement normalisées entre 0 et 1, La Figure I.9 montre l'architecture typique d'un CNN.

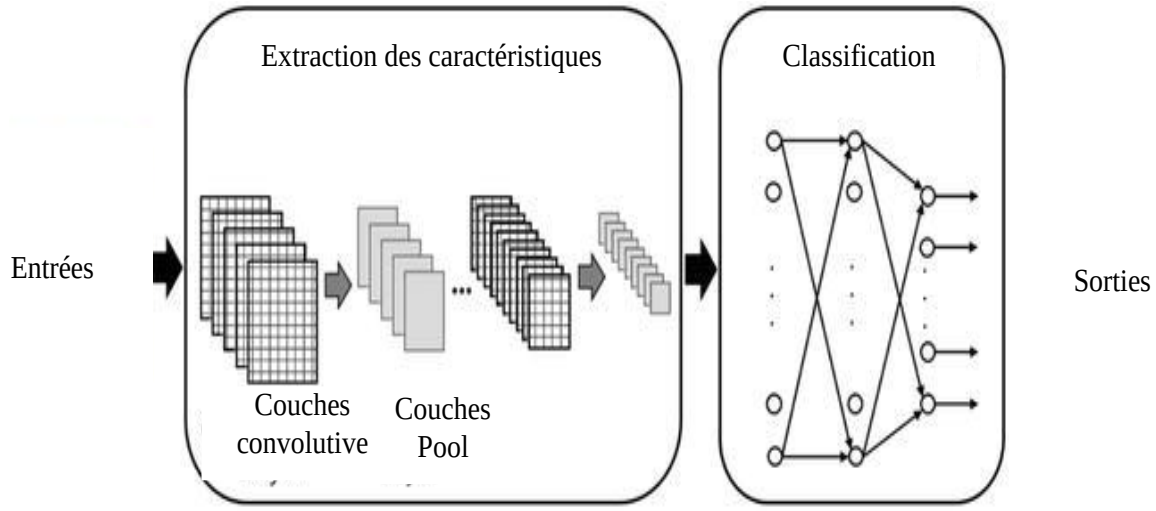


Figure I. 9. Représente l'architecture typique de CNN.

I.5.2. Les couches d'un réseau de neurones convolutif

I.5.2.1. Couche de convolution (CONV)

La convolution est la première couche à extraire des entités d'une image d'entrée. La convolution préserve la relation entre les pixels (invariance de translation). Selon la Figure I.10, il s'agit d'une opération mathématique pour calculer la taille spatiale du volume de sortie (la sortie de couche de convolution). Trois paramètres permettent de dimensionner le volume de la couche de convolution qui sont :

La profondeur de la couche: est le nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).

Le pas (stride): contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

La marge à zéro ou (zero-padding): est la taille du zero-padding. Au final, les cartes de convolutions sont mises à plat et concaténées en un long vecteur qui comprend les caractéristiques les plus pertinentes de l'image, appelé code CNN.

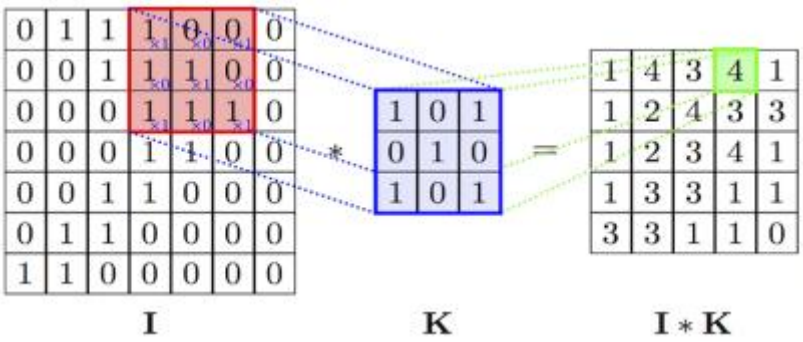


Figure I. 10. Couche de convolution.

I.5.2.2. Couche de correction (ReLU)

La couche de correction ReLU remplace toutes les valeurs négatives reçues en entrées par des zéros. En ne modifiant pas les données positives, ReLU n'impacte pas les caractéristiques misent en évidence par la convolution, au contraire : elle les met davantage en évidence en creusant l'écart (valeurs négatives) « entre » deux caractéristiques. La fonctionnalité ReLU est illustrée à la Figure I.11 [27].

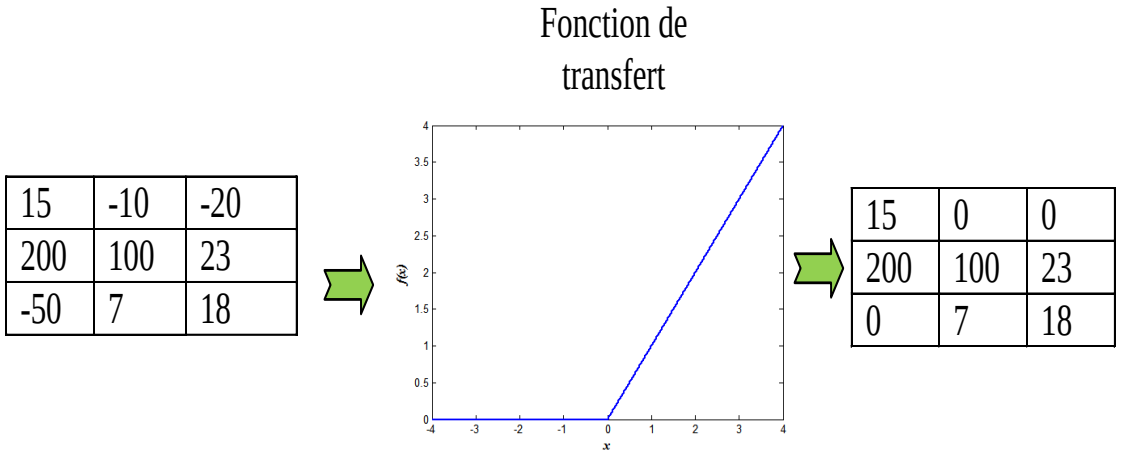


Figure I. 11. Représentation graphique de la fonctionnalité RELU.

I.5.2.3. Couche entièrement connectée (FC)

Ce type de couche reçoit un vecteur d'entrée et produit un nouveau vecteur de sortie. Pour cela, il applique une combinaison linéaire et éventuellement une fonction d'activation aux valeurs reçues en entrée.

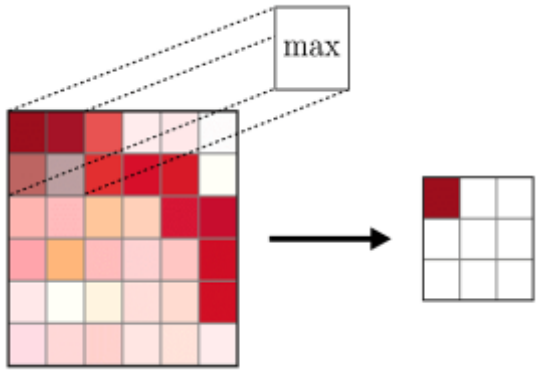
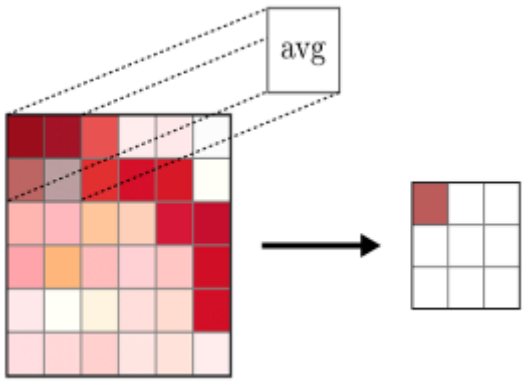
I.5.2.4. Couche de perte (LOSS)

C'est La dernière couche dans le réseau, spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La perte «Softmax» est utilisée pour prédire une seule classe parmi *K* classes mutuellement exclusives [28].

I.5.2.5. La couche de groupement (Pooling layer)

La couche de pooling en anglais (pooling layer) (POOL) est une opération de sous-échantillonnage typiquement appliquée après une couche convolutionnelle. En particulier, les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises, respectivement [26].

Tableau I. 2. Max et Moyenne Pooling.

Max Pooling	**Moyenne Pooling**
Chaque opération de Pooling sélectionne la valeur maximale de la surface.	Chaque opération de Pooling sélectionne la valeur moyenne de la surface.
	
➢ Garde les caractéristiques détectées ➢ Plus communément utilisé	➢ Sous échantillonne la feature map ➢ Utilisé dans LeNet

Elle permet la réduction de la taille de l'image en tenir en compte ses caractéristiques importantes. D'après la Figure I.12, le principe est de couper l'image en cellules régulières, puis nous conservons la valeur maximale dans chaque cellule par rapport au filtre utilisé (2×2

pixels, 3×3…). Donc des petites cellules carrées sont souvent utilisées afin de ne pas perdre trop d'informations.

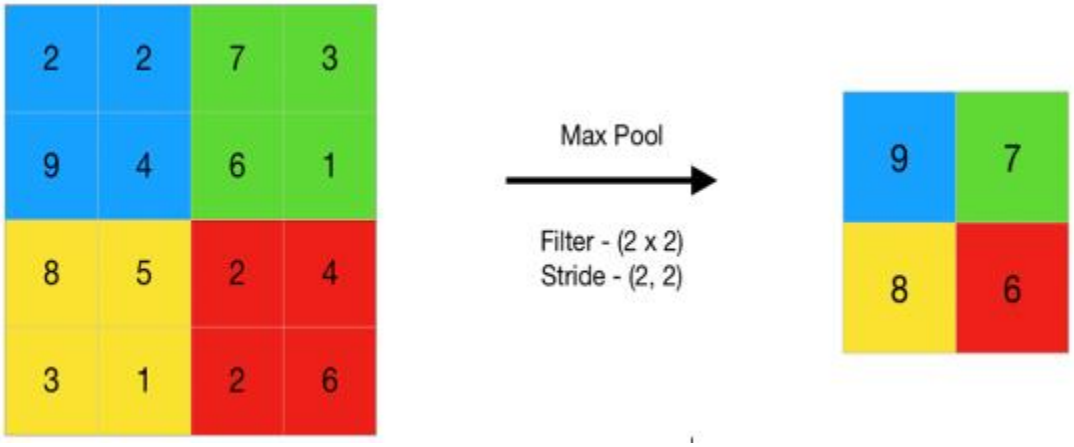


Figure I. 12. Max Pooling avec filtre 2×2 et un pas 2.

I.5.3. Fonctions d'activation

La fonction d'activation est une formule mathématique (algorithme) activée dans certaines circonstances. Comme illustré dans la Figure I.13, lorsque les neurones calculent la somme pondérée des valeurs d'entrée + le biais, elles sont transmises à la fonction d'activation, qui vérifie si la valeur calculée est supérieure au seuil requis. La fonction d'activation est activée et une valeur de sortie est calculée. Cette valeur de sortie est ensuite transmise aux couches suivantes ou précédentes (en fonction de la complexité du réseau), ce qui peut aider les réseaux de neurones à modifier le poids de leurs neurones.

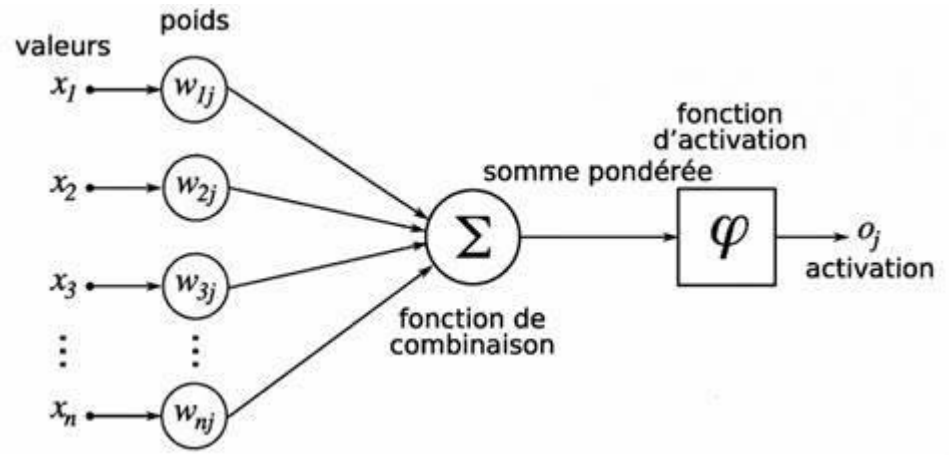


Figure I. 13. Illustration de la fonction d'activation.

Les fonctions d'activation introduisent la non-linéarité dans les réseaux de neurones, nécessaire pour résoudre des problèmes complexes.

Si nous traçons les sorties non linéaires produites par les fonctions d'activation, nous Obtiendrons une courbure. La pente de la courbe est utilisée pour calculer le gradient. Et le

gradient nous aide à comprendre le taux de changement et les relations entre les variables. À partir des relations, les algorithmes sont optimisés et les poids sont mis à jour [26].

I.5.3.1. Fonction Sigmoïde

La fonction Sigmoïde est une fonction continue. Comme le montre la Figure I.14, elle est utilisée lorsque les valeurs à prédire par le RN sont comprises dans l'intervalle [0, 1]. Son but est d'exprimer sa valeur de sortie sous forme d'une probabilité, si la valeur en entrée est un très grand nombre positif, la fonction convertira cette valeur en une probabilité de 1. A l'inverse, si la valeur en entrée est un très petit nombre négatif, la fonction convertira cette valeur en une probabilité de 0.

La fonction Sigmoïde est définie par :

$$f(x) = \frac{1}{1+e^{-x}} \tag{I.3}$$

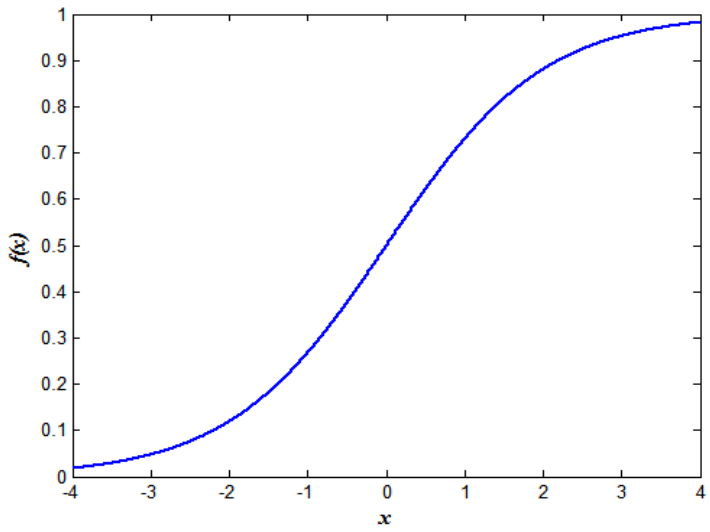


Figure I. 14. La fonction sigmoïde.

I.5.3.2. Fonction Relu / Leaky Relu

La fonction Relu (Unité de Rectification Linéaire), montrée sur la Figure I.15, est créée pour palier au problème de saturation (est la situation où le gradient reste presque nul après chaque période de temps durant le processus d'apprentissage) des deux fonctions (Sigmoïde et Tanh).

Relu désigne la fonction réelle non-linéaire interprétée par la formule :

$$f(x) = \max(0, x) \tag{I.4}$$

Si l'entrée est négative alors la sortie est 0, si l'entrée est positive alors la sortie est x. Cette fonction d'activation augmente considérablement la convergence du réseau et ne sature pas.

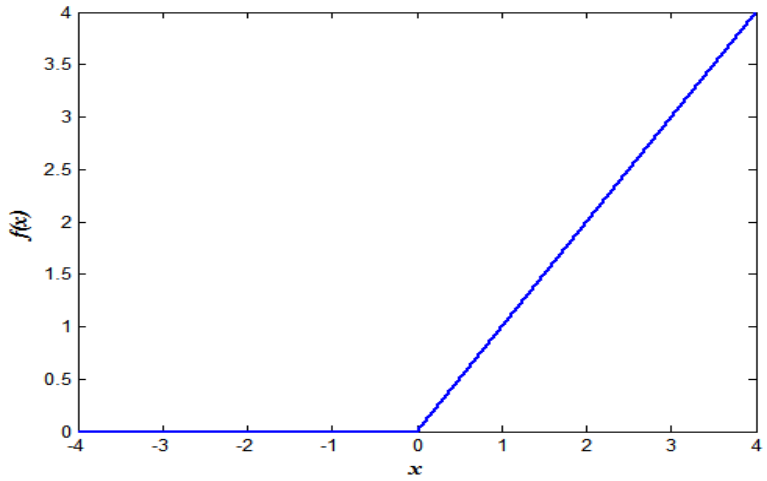


Figure I. 15. La fonction RELU.

L'inconvénient de la fonction Relu est que si la valeur d'entrée est négative, le neurone reste inactif, ainsi les poids ne sont pas mis à jour et le réseau n'apprend pas en raison des gradients de 0 dans la partie négative. Comme montrée la Figure I.16, la fonction Leaky Relu essaye de résoudre le problème de la fonction Relu lorsque l'entrée est négative.

Le concept est lorsque l'entrée est négative, il aura une petite pente négative de a=0,01 ou plus. Cette fonction élimine le problème d'inactivité de la fonction Relu pour les valeurs négatives.

La fonction Leaky RELU est donnée par la formule :

$$f(x) = \begin{cases} ax & si \quad x < 0 \\ x & si \quad x > 0 \end{cases} \tag{I.5}$$

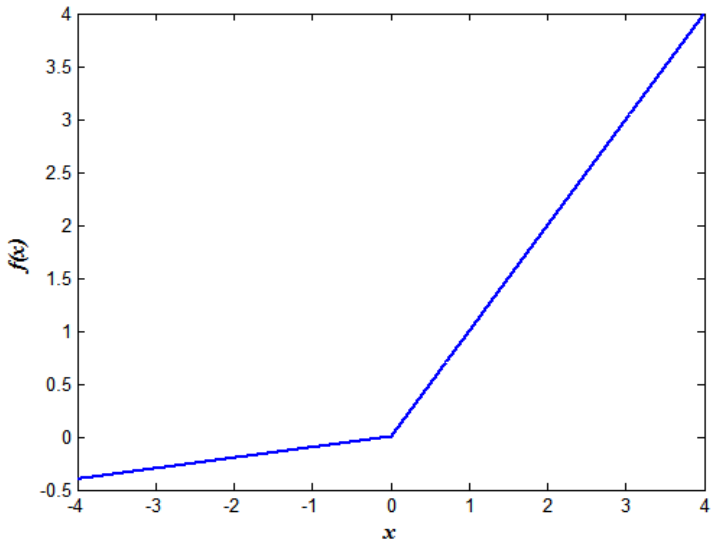


Figure I. 16. La fonction Leaky RELU.

I.5.3.3. Softmax

La fonction d'activation de softmax est habituellement utilisée dans la dernière couche du réseau, lorsque le problème de classification n'est pas binaire mais contient plusieurs classes, cette dernière est généralement utilisée, mais est également viable pour le cas binaire. Il s'agit d'une généralisation de la fonction logistique et n'est qu'un autre nom pour un modèle de classification multinomiale quand on suppose qu'il n'existe aucune hiérarchie parmi les classes. La fonction softmax est agréable car elle donne une approximation de la probabilité qu'une classe soit correcte. L'approche la plus simple consiste à simplement choisir la classe avec la probabilité la plus élevée et ignorer le reste. Mais étant donné qu'il s'agit d'une fonction probabiliste, il peut également être utilisé pour un modèle générateur [29].

La fonction softmax est représentée dans la Figure I.17, et interprétée par la formule :

$$f(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{k} e^{x_j}} \tag{I.6}$$

Où x est un vecteur des entrées du calque de sortie (si 10 unités de sortie, donc il y a 10 éléments dans x). j=1, 2, ..., k.

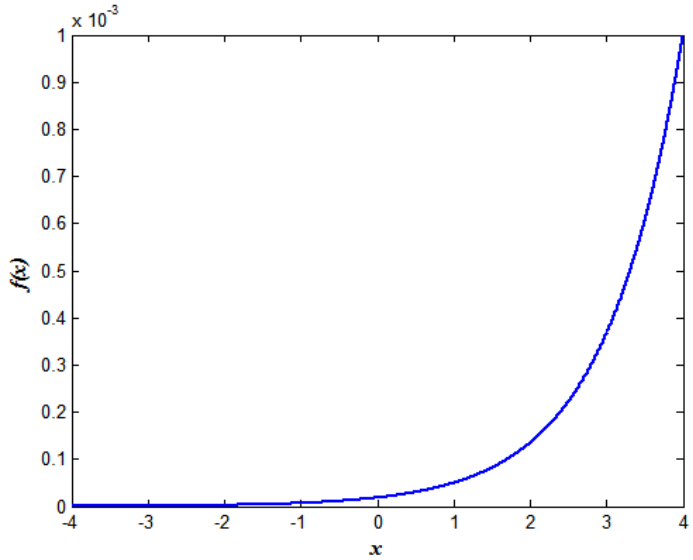


Figure I. 17. La fonction Softmax.

I.6. Architectures d'un réseau de neurones les plus courantes

I.6.1. AlexNet

Le premier travail qui a popularisé les réseaux convolutifs en vision par ordinateur est AlexNet, développé par Alex Krizhevsky, Ilya Sutskever et Geoff Hinton. AlexNet a été soumis au défi ImageNet LSVRC en 2012 et a largement surpassé le deuxième finaliste. Le réseau avait une architecture très similaire à LeNet, mais était plus profond, plus grand et

comportait des couches convolutionnelles empilées les unes sur les autres. L'architecture se compose de huit couches, cinq couches convolutives et trois couches entrainement connectées, comme il est représenté dans les Figures I.18 et I.19 [30].

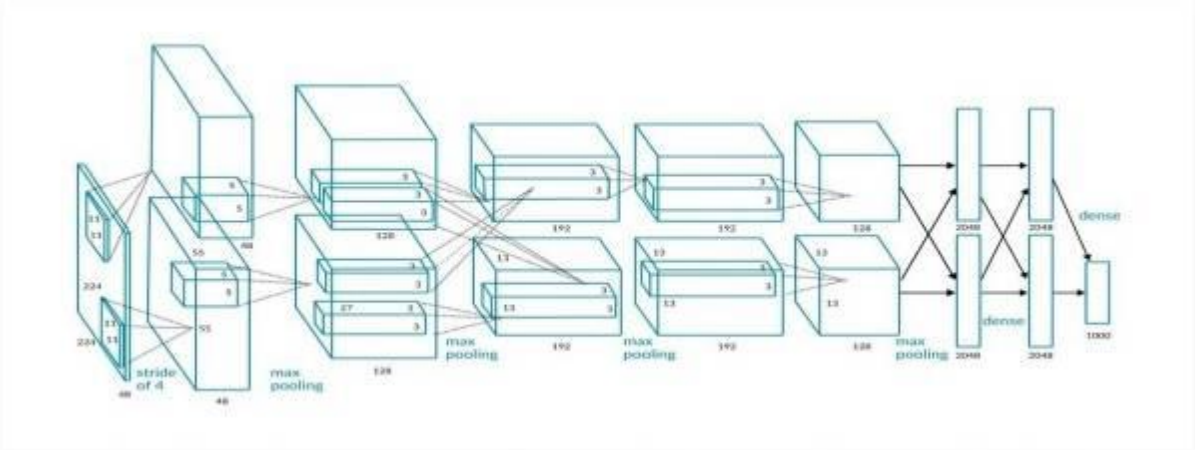


Figure I. 18. Architecture AlexNet.

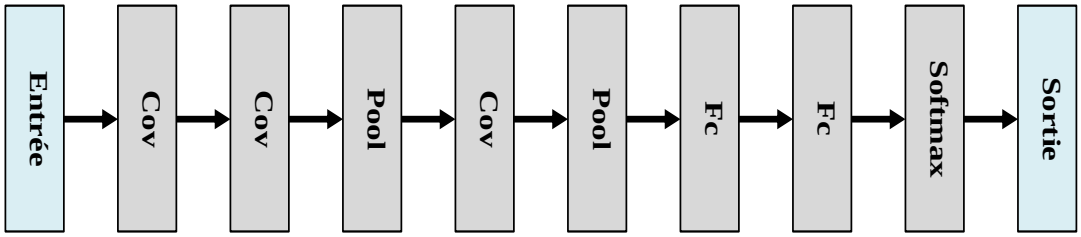


Figure I. 19. Représentation de l'architecture de AlexNet.

I.6.2. VGG

VGG est un algorithme connu en Computer Vision très souvent utilisé par transfert d'apprentissage pour éviter d'avoir à le réentraîner et résoudre des problématiques proches sur lesquelles VGG a déjà été entraîné.

a. VGG16

VGG-16 est constitué de plusieurs couches, dont 13 couches de convolution et 3 fully-connected. Il doit donc apprendre les poids de 16 couches.

Il prend en entrée une image en couleurs de taille 224×224 px et la classifie dans une des 1000 classes. Il renvoie donc un vecteur de taille 1000, qui contient les probabilités d'appartenance à chacune des classes [31].

Les deux figures I.20 et I.21 ci-dessous montrent l'architecture de VGG-16 et sa représentation en 3D.

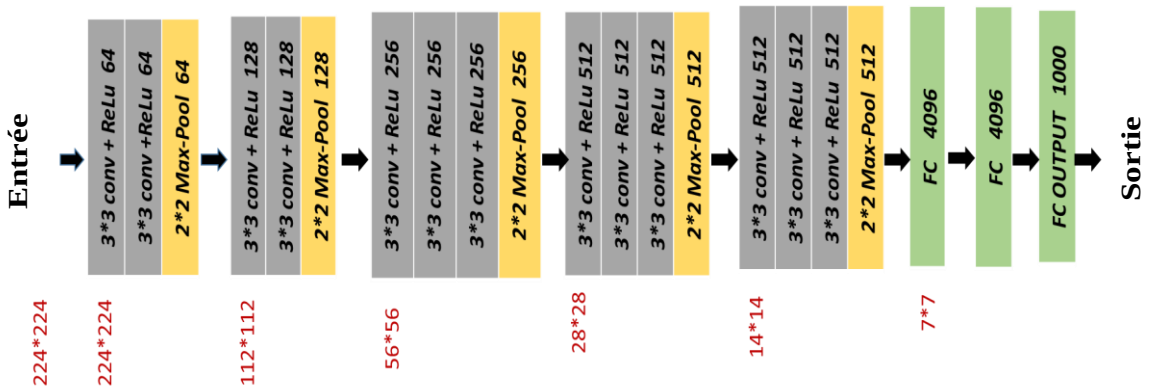


Figure I. 20. Architecture de VGG-16.

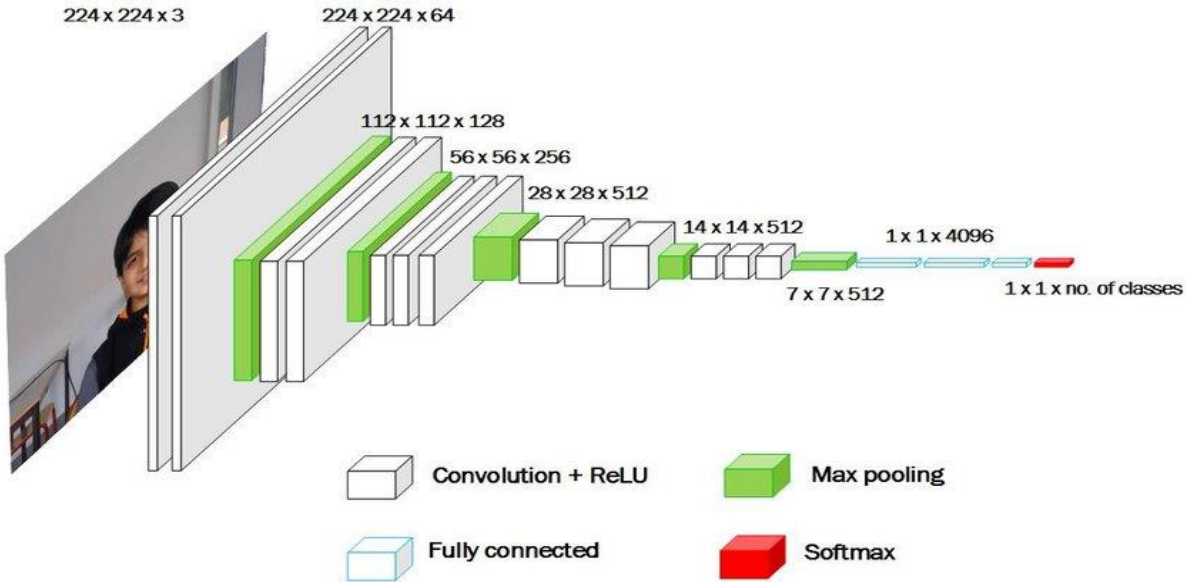


Figure I. 21. Représentation 3D de l'architecture de VGG-16.

Chaque couche de convolution utilise des filtres en couleurs de taille 3 ×3 px, déplacés avec un pas de 1 pixel. Le zero-padding vaut 1 pixel afin que les volumes en entrée aient les mêmes dimensions en sortie. Le nombre de filtres varie selon le "bloc" dans lequel la couche se trouve. De plus, un paramètre de biais est introduit dans le produit de convolution pour chaque filtre.

Chaque couche de convolution a pour fonction d'activation une ReLU. Autrement dit, il y a toujours une couche de correction ReLU après une couche de convolution.

L'opération de pooling est réalisée avec des cellules de taille 2×2 *px* et un pas de 2 *px*, les cellules ne se chevauchent donc pas.

Les deux premières couches fully-connected calculent chacune un vecteur de taille 4096, et sont chacune suivies d'une couche ReLU. La dernière renvoie le vecteur de probabilités de taille 1000 (le nombre de classes) en appliquant la fonction softmax. De plus, ces trois couches utilisent un paramètre de biais pour chaque élément du vecteur en sortie.

b. VGG19

VGG-19 est un réseau de neurones convolutifs de 19 couches de profondeur. Il est possible de charger une version pré-entraînée du réseau entraîné sur plus d'un million d'images de la base de données ImageNet. Le réseau pré-entraîné peut classer les images en 1000 catégories d'objets, telles que le clavier, la souris, le crayon et de nombreux animaux. En conséquence, le réseau a appris de riches représentations de caractéristiques pour un large éventail d'images. Le réseau a une taille d'entrée d'image de 224 par 224 [32].

Tableau I. 3. Configuration réelle de réseau.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
Input (224 × 224 RGB image)					
Conv 3-64	Conv 3-64 **LRN**	Conv 3-64 **Conv 3-64**	Conv 3-64 Conv 3-64	Conv 3-64 Conv 3-64	Conv 3-64 Conv 3-64
Maxpool					
Conv 3-128	Conv 3-128	Conv 3-128 Conv 3-128	Conv 3-128 Conv 3-128	Conv 3-128 Conv 3-128	Conv 3-128 Conv 3-128
Maxpool					
Conv 3-256 Conv 3-256	Conv 3-256 Conv 3-256	Conv 3-256 Conv 3-256	Conv 3-256 Conv 3-256 **Conv 1-256**	Conv 3-256 Conv 3-256 **Conv 3-256**	Conv 3-256 Conv 3-256 Conv 3-256 **Conv 3-256**
Maxpool					
Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512 **Conv 1-512**	Conv 3-512 Conv 3-512 **Conv 3-512**	Conv 3-512 Conv 3-512 Conv 3-512 **Conv 3-512**
Maxpool					
Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512	Conv 3-512 Conv 3-512 **Conv 1-512**	Conv 3-512 Conv 3-512 **Conv 3-512**	Conv 3-512 Conv 3-512 Conv 3-512 **Conv 3-512**
Maxpool					
FC-4096					
FC-4096					
FC-1000					
Soft-max					

I.6.3. ResNet

Le défi ILSVRC 2015 a été remporté en utilisant un réseau résiduel (ou ResNet), développé par ‘‘Kaiming He et al’’ [33] , qui a réalisé un taux d'erreur top-5 sous 3,6%, en utilisant un CNN extrêmement profond composé de 152 couches. Cela confirme la tendance générale: les

modèles deviennent de plus en plus profonds, avec de moins en moins de paramètres. La clé pour pouvoir former un réseau aussi profond est d'utiliser des connexions de saut (également appelées connexions de raccourci): le signal entrant dans une couche est également ajouté à la sortie d'une couche située un peu plus haut dans la pile. Lors de l'entraînement d'un réseau de neurone, l'objectif est de lui faire modéliser une fonction cible $h(x)$. Si vous ajoutez l'entrée x à la sortie du réseau (c'est-à-dire que vous ajoutez une connexion de saut), le réseau sera forcé de modéliser $f(x)=h(x)-x$ plutôt que $h(x)$. C'est ce qu'on appelle l'apprentissage résiduel (voir la Figure I.22). Lorsque l'initialisation d'un réseau de neurones normal, ses poids sont proches de zéro, donc le réseau ne sort que des valeurs proches de zéro. Si vous ajoutez une connexion de saut, le réseau résultant sort simplement une copie de ses entrées; en d'autres termes, il modélise initialement la fonction d'identité. Si la fonction cible est assez proche de la fonction identité (ce qui est souvent le cas), cela accélérera considérablement l'entraînement.

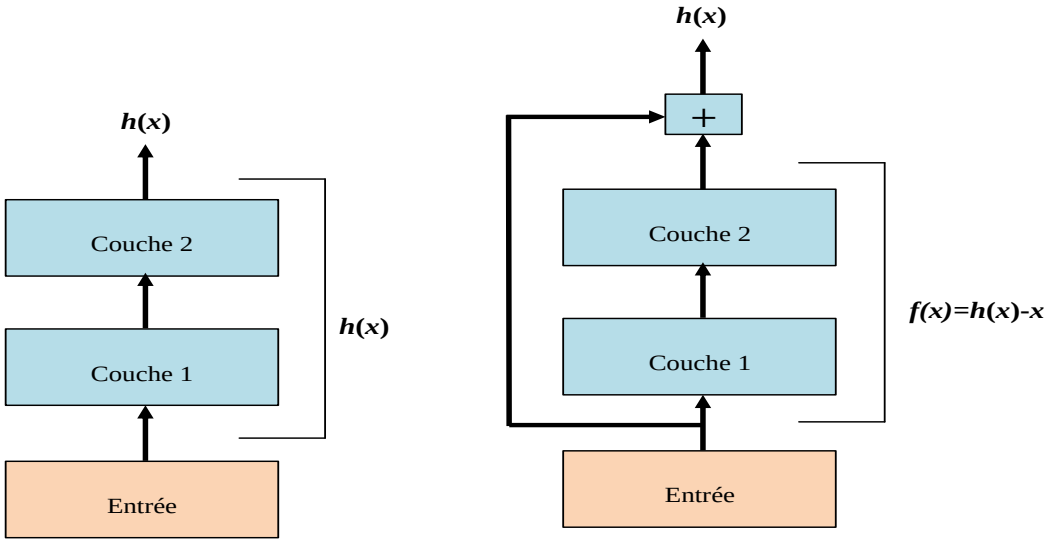
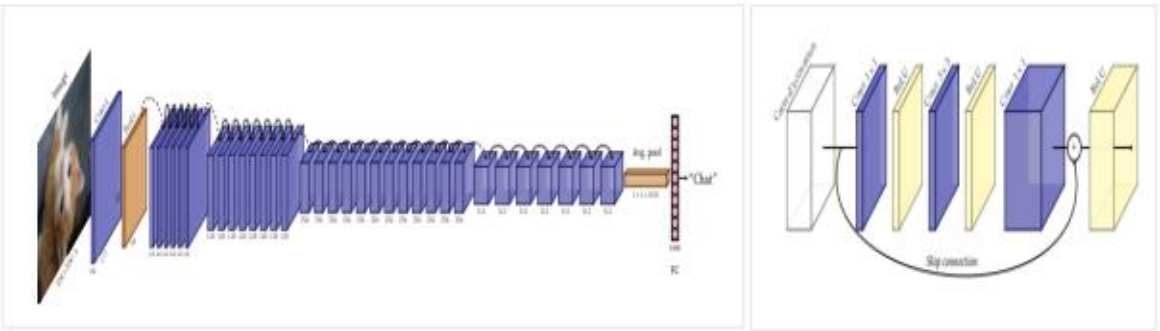


Figure I. 22. L'apprentissage résiduel.



a) ResNet-34 b) Bloc Résiduel

Figure I. 23. Architecture ResNet.

L'inconvénient de cette approche est le nombre élevé d'activation et de gradients intermédiaires à calculer rend les ResNet couteux en mémoire et peu pratiques sur des images de grandes dimensions. La Figure I.23, montre l'architecture de ResNet.

I.6.4. MobileNet

Une nouvelle architecture MobileNet est également disponible depuis Avril 2017, Cette architecture utilise des convolutions séparables pour réduire le nombre de paramètres. La convolution séparée est la même que Xception, montrée dans la Figure I.24, mais avec une grande réduction des paramètres. Ci-après la figure I.25 montre l'architecture complète du modèle [34]:

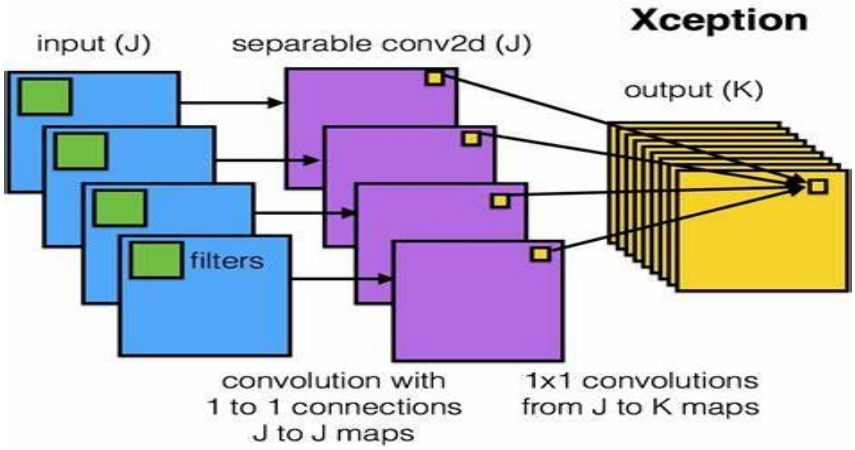


Figure I. 24. Le Module Xception.

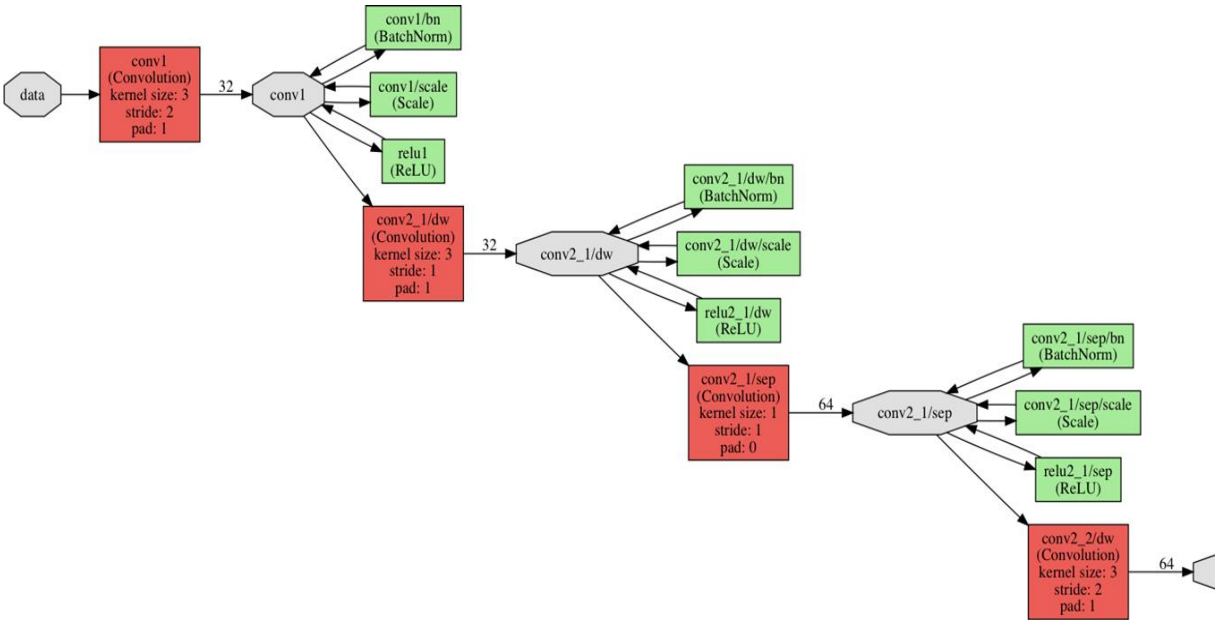


Figure I. 25. La représentation de l'architecture de MobileNet.

I.7. Domaines d'applications

Les méthodes de détection intelligente sont un domaine phénoménal avec diverses études ; dans notre travail nous nous concentrerons sur le domaine de la détection d'objets, domaines bien étudiés de la détection d'objets comprennent la détection des visages, la reconnaissance faciale et le transport intelligent [22].

Parmi ces applications on cite :

- La détection et la reconnaissance faciale :

  Ces applications comprennent: l'authentification d'identité, le contrôle d'accès, la surveillance, etc.

  - Sécurité: pour assurer un niveau de sécurité plus élevé en utilisant l'identification dans les lieux nécessitant un contrôle d'accès physique (bâtiments, aéroports et ports de mer sécurisés, Points de contrôle aux frontières, etc.…) et dans les systèmes d'information nécessitant un contrôle d'accès logique (ordinateurs, réseaux informatiques, bases de données sensibles, commerce électronique, distributeurs automatiques de billets, téléphones portables, etc.).
  - Surveillance: Recherche dans les lieux publics de criminels, de terroristes connus, de toxicomanes, d'enfants disparus, d'immigrants afin que les autorités puissent les informer de l'endroit où les trouver.
  - Vérification d'identité: telle que carte d'identité nationale, permis de conduire, sécurité sociale, passeport, etc.
  - Les interfaces homme-machine: On peut rendre la communication entre l'homme et la machine plus attractive, plus naturelle et plus confortable. Par exemple, s'il est possible d'estimer la direction de la tête, on peut diriger une caméra vers l'objet que l'individu regarde et acquérir des informations concernant cet objet.

- Applications pour la sécurité routière [35]:
  - Connaitre les panneaux de signalisations, etc.
  - L'application d'avertissement de la violation des feux de signalisation.
  - L'application d'avertissement à l'approche d'un tournant.
  - L'application d'avertissement coopératif d'une collision imminente.
  - L'application d'avertissement de changement de voie, etc.

- Prévenir les collisions et les travaux sur les routes, de détecter les obstacles (fixes ou mobiles) et de distribuer les informations météorologiques.

I.8. Conclusion

Ce chapitre a pour but de présenter le domaine de l'intelligence artificielle, nous avons commencé par les définitions (intelligence artificielle, apprentissage automatique et apprentissage profond) puis nous passons aux types d'apprentissage automatique et une comparaison entre l'apprentissage automatique et l'apprentissage profond, ensuite nous nous sommes penchés sur les réseaux de neurones artificiels, les fonctions d'activation et les réseaux de neurones convolutifs ''*CNN*'' puis nous avons décrit leur architecture en citant les plus actuels : *AlexNet*, *VGG*, *MobileNet*, *ResNet* et nous avons terminé par les domaines d'application.

Chapitre II

La carte de développement NVIDIA Jetson Nano

II.1. Introduction

Le dispositif NVIDIA Jetson Nano est un mini-ordinateur puissant, qui a été lancé par NVIDIA (NVIDIA Corporation, États-Unis) en mars 2019. Il est destiné aux applications d'intelligence artificielle embarquées comme l'edge computing et la robotique, et permet de mettre en œuvre des solutions lot grâce à la puissance de calcul de GPU. On peut fixer une caméra sur la carte NVIDIA Jetson Nano, on peut fabriquer un appareil intelligent ou réaliser un système de sécurité intelligent et fiable, comme on peut construire des programmes tels que la vision par ordinateur ou la vision d'intelligence en appliquant des bibliothèques d'apprentissage automatique sur la carte NVIDIA Jetson Nano [36].

Dans ce chapitre nous représentons la carte NVIDIA Jetson Nano, en citant ces caractéristiques matériels et tous ce qui concerne l'utilisation de cette carte telles que: la configuration de logiciel sur NVIDIA, l'accès à distance en penchant sur les ports GPIO puis le langage de programmation Python ainsi ces bibliothèques: Tensorflow, OpenCV, Google Colaboratory, Jupyter, …, etc.

II.2. Carte Nvidia Jetson Nano

Le NVIDIA Jetson Nano s'agit d'un petit ordinateur puissant qui permet d'exécuter plusieurs réseaux neuronaux en parallèle pour des applications afin de mettre en œuvre l'analyse d'image tel que la classification, la détection d'objets, la segmentation ou le traitement de la voix.

En outre, il dispose d'un connecteur d'entrée/sortie général à 40 broches entièrement configurable (GPIO) par lequel d'autres dispositifs peuvent être facilement contrôlés [37].

La carte est composée d'un processeur cadencé à 1,43 GHz et d'un GPU de 128 cœurs de la génération Maxwell. La Figure II.1-(a) présente le premier modèle de NVIDIA Jetson, il s'appelle NVIDIA Jetson Nano A02. Maintenant NVIDIA a sorti un nouveau modèle, le NVIDIA Jetson Nano B01 comme représenté sur la Figure II.1-(b). Techniquement les deux modèles possèdent le même CPU et GPU mais certains périphériques ont été modifiés [38].

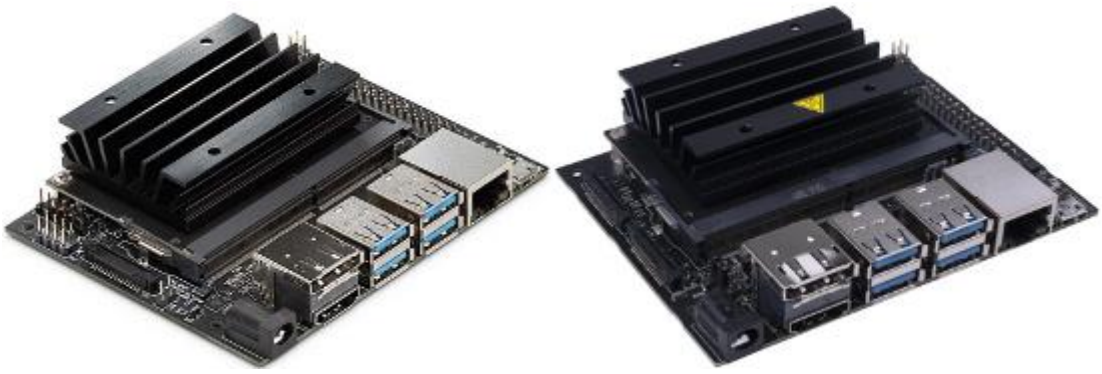


Figure II. 1. (a) NVIDIA Jetson NanoB01, (b) NVIDIA Jetson Nano A02.

II.2.1. Caractéristiques de la carte NVIDIA Jetson Nano

En général, la carte NVIDIA Jetson Nano possède les spécifications techniques indiquées dans le Tableau (II.1) ci–dessous:

Tableau II. 1. Spécifications techniques de la carte NVIDIA Jetson Nano.

Fonctionnalités	***Informations***
GPU	128-core Maxwell
GPU	Quad-core ARM A57 @ 1.43 GHZ
Mémoire	4 GB 64-bit LPDDR4 25.6 GB/s
Stockage	MicroSD
Encodage vidéo	4K @ 30 ; 4x 1080p @ 30 ; 9x 720p @ 30 (H. 264/H. 265)
Décodage Vidéo	4K @ 60 ; 2x 4K @ 30 ; 8x 1080p @ 30 ; 18x 720p @ 30 (H.264/H.265)
Caméra	1x connecteur MIPI CSI-2
Connectivité	Gigabit Ethernet et 802.11ac sans fil
Afficher	HDMI et port d'affichage
USB	1x USB 3.0 Type A, 2x USB 2.0 Type A, 1x USB 2.0 Micro-B
I/O	GPIO. I2C. I2S. SPI. UART
Mécanique	69 mm *45 mm. 260-pin connecteur de bord

II.2.2. Applications de la carte NVIDIA Jetson Nano

De plus l'utilisation de la carte NVIDIA Jetson dans le domaine de l'intelligence artificielle, elle peut être utilisée pour des activités informatiques quotidiennes.

Quelques applications de NVIDIA Jetson Nano :

- Naviguer sur Internet pour rechercher des informations.
- Rédaction de documents.
- La création de feuilles de calcul et l'impression de documents.
- Installer des applications bureautiques à partir de libre office.
- Joindre des caméras externes via l'interface CSI ou une caméra USB.
- Utiliser le NVIDIA Jetson Nano pour créer d'excellents programmes de vision par ordinateur

Dans le but de réaliser des applications de l'IA, le NVIDIA Jetson prend en charge plusieurs bibliothèques telle que Pandas, Numpy, Tensorflow, Keras et openCV qui fournit diverses bibliothèques de traitement d'images et de vidéos pour la vision par ordinateur, et peut être utilisée sur des programmes tels que C/C++ et Python. Pour obtenir des calculs optimisés, il

faut s'assurer que ces bibliothèques et les programmes prennent en charge les cœurs de GPU de la carte NVIDIA Jetson Nano.

II.3. Utilisation de la carte Nvidia Jetson Nano

II.3.1. Préparation du matériel

Le NVIDIA Jetson Nano ne dispose pas de stockage interne donc elle à besoin d'un stockage externe, elle ne prend en charge qu'une carte MicroSD pour le stockage interne. Ce périphérique de stockage sera utilisé pour stocker le système d'exploitation (OS) et les données. Elle a probablement besoin d'un lecteur de carte MicroSD pour permettre de travailler avec l'ordinateur pour lire et écrire des fichiers.

La figure II.2 montre une carte NVIDIA Jetson Nano, une carte MicroSD et un lecteur de carte MicroSD.

Figure II. 2. Carte MicroSD et NVIDIA Jetson Nano.

En général, on a besoin de matériel supplémentaire pour faire fonctionner le dispositif NVIDIA Jetson Nano comme suit :

- Une carte MicroSD avec une taille de stockage de 16 Go minimum.
- Lecteur de carte MicroSD pour écrire et lire des fichiers à partir d'un ordinateur.
- Souris avec fil USB.
- Clavier avec câble USB.
- Adaptateur électrique 5V 2A.
- Câble micro USB pour l'adaptateur d'alimentation.
- Moniteur avec connecteur HDMI.

Après avoir défini la configuration matérielle requise, on peut préparer le logiciel nécessaire au fonctionnement de la carte NVIDIA Jetson Nano.

II.3.2. Installation du système d'exploitation

Le NVIDIA Jetson Nano utilise son propre système d'exploitation pour faire tourner ses applications. Ce système est basé sur Ubuntu Linux, il permet d'effectuer n'importe quelle activité sur un NVIDIA Jetson Nano.

Les étapes de configuration de logiciel à suivre sont :

- Télécharger l'image de la carte SD du kit de développement Jetson Nano. (Disponible sur https://developer.nvidia.com/jetson-nano-sd-card-image).
- Flasher le fichier image NVIDIA sur la carte microSD selon la plateforme de système d'exploitation. On peut le faire en utilisant certains outils, tels que SD Memory Card Formatter pour Windows ou Etcher pour Windows, Linux et macos.
- Branchez votre carte microSD avec lecteur sur votre ordinateur.

On est maintenant prêt à flasher l'image NVIDIA Jetson Nano comme il est indiqué dans la figure II.3.

Figure II. 3. Flasher le fichier image NVIDIA sur la carte MicroSD avec Etcher.

On peut maintenant commencer à flasher le fichier image en suivant les étapes suivantes :

- Sélectionnez le Fichier image NVIDIA (fichier ZIP).
- Sélectionnez la carte microSD, si elle n'est pas reconnue par le système d'exploitation, il faut formater la carte avec le mode FAT. La table d'allocation des fichiers (FAT) est généralement utilisée sur un ordinateur personnel équipé du système d'exploitation Windows OS.
- Flasher l'image NVIDIA Jetson Nano sur la carte microSD en cliquant sur le bouton Flash et donner l'administrateur à flasher l'image, ce processus prend quelques minutes.

Après avoir terminé le flashage, une confirmation affiche sur l'écran pour débrancher la carte microSD et la placer sur le dispositif Nvidia Jetson Nano. La Figure II.4 montre la fin du flashage de l'image avec l'application Etcher.

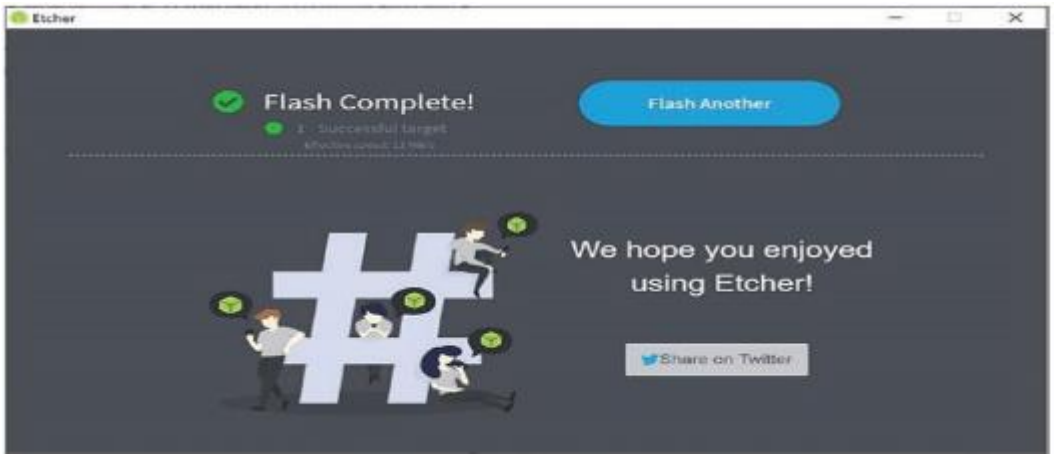


Figure II. 4. Flash de l'image de la carte SD du kit de développement Jetson Nano.

II.3.3. Configuration de la carte NVIDIA Jetson nano

Après avoir branché un adaptateur d'alimentation, on peut configurer le logiciel NVIDIA Jetson Nano pour la première fois. Il y a quelques paramètres qu'on doit compléter, utiliser le clavier et la souris. Tout d'abord, on obtient un formulaire d'accord comme illustré sur la figure II.5, accepter cet accord pour continuer à utiliser le périphérique NVIDIA Jetson Nano et cliquer sur le bouton ‘‘Continuer’’.

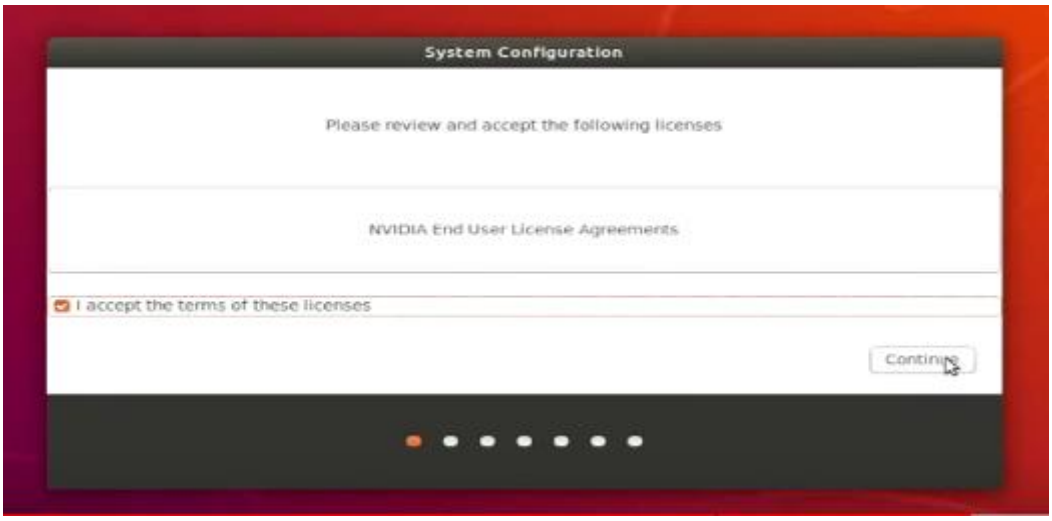


Figure II. 5. Accord de l'utilisateur sur Nvidia Jetson Nano.

Dans l'étape suivante, on doit sélectionner une langue pour tous les textes sur le NVIDIA Jetson Nano, également définir le type de clavier et le fuseau horaire pour la région locale.

Enfin, on crée un compte utilisateur pour le NVIDIA Jetson Nano en saisissant le nom complet, le nom d'utilisateur et le mot de passe; et on définit également le modèle d'authentification comme indiqué sur la Figure II.6.

Figure II. 6. Créer un compte utilisateur pour le NVIDIA Jetson Nano.

La carte NVIDIA Jetson Nano peut être connectée à internet via un câble LAN ou un module Wi-Fi, ce qui permet de mettre à jour le logiciel NVIDIA Jetson Nano. Après avoir effectué toutes les tâches, le bureau de la NVIDIA Jetson Nano va apparaitre.

Ce bureau est basé sur Ubuntu Linux. On peut effectuer des activités normales comme sur n'importe quel ordinateur, telles que créer et modifier des fichiers, naviguer sur Internet, chatter, etc. La Figure II.7 montre le bureau de système NVIDIA Jetson Nano.

Figure II. 7. Nvidia Jetson Nano Bureau.

II.3.4. Terminal

Puisque l'image NVIDIA Jetson Nano est construite à partir d'Ubuntu, on peut utiliser le logiciel Terminal pour effectuer des tâches d'administration, telles que la création et la modification de fichiers et dossiers ou compiler et exécuter des programmes.

On trouve le Terminal de NVIDIA Jetson en cliquant sur Rechercher en haut à gauche. Taper "Terminal" pour afficher l'application, après avoir cliqué sur l'icône Terminal, on obtient l'application Terminal comme illustré à la Figure II.8.

Figure II. 8. Application du terminal.

II.4. Accès à distance à la carte Nvidia Jetson nano

Le périphérique NVIDIA Jetson Nano dispose d'un module réseau intégré avec Ethernet. Il suffit de brancher un câble LAN sur le port Ethernet du périphérique NVIDIA Jetson Nano. Une fois que le périphérique de la carte est connecté à un réseau, on peut vérifier son adresse IP à l'aide de la commande ''ifconfig'' dans le Terminal. La figure II.9 montre le périphérique NVIDIA Jetson Nano avec un câble LAN UTP branché.

Figure II. 9. Connecter un câble LAN UTP à Nvidia Jetson Nano.

Dans le cas où il n'affiche pas l'adresse IP, cela veut dire que le réseau ne dispose probablement pas d'un serveur DHCP. Cependant, on peut configurer une adresse IP statique sur le NVIDIA Jetson Nano à l'aide de Terminal en modifiant le fichier ''/etc/network/interfaces'' en utilisant la commande suivante:

sudo nano /etc/network/interfaces

Ensuite, on définit une adresse IP statique. Par exemple, si on veut définir une adresse IP comme étant: 192.168.1.10 et l'adresse IP de la passerelle comme étant: 192.168.1.1, on écrit les scripts suivants dans le fichier interfaces ouvert par la commande précédente:

iface eth0 inet static

***address* 192.168.1.10**

***netmask* 255.255.255.0**

***getway* 192.168.1.1**

II.4.1. Connexion au réseau Wi-Fi

Le NVIDIA Jetson Nano n'est pas équipé d'un système Wi-Fi intégré, un dispositif supplémentaire est nécessaire si on veut accéder à un réseau Wi-Fi. Il existe deux options: soit d'utiliser un module Wi-Fi ou un dongle USB Wi-Fi.

II.4.1.1. Module de carte réseau Wi-Fi

Le NVIDIA Jetson Nano fournit un connecteur pour vous permettre de fixer votre carte réseau Wi-Fi sur la carte. On peut utiliser la carte Intel Dual Band Wireless-Ac 8265 w/Bluetooth 8265.NGWMG sur le NVIDIA Jetson Nano.

Pour attacher la carte réseau Wi-Fi, on doit ouvrir le module NVIDIA de la carte. Ensuite, insérez la carte réseau Wi-Fi dans le connecteur. La Figure II.10 illustre l'installation de ce module sur la carte NVIDIA.

Figure II. 10. Fixation du module Wi-Fi dans le NVIDIA Jetson Nano.

Maintenant, une liste de SSID Wi-Fi apparait sur le bureau et cela permet de sélectionner un SSID Wi-Fi et entrer la clé SSID si elle est disponible. Après avoir connecté à l'Internet par Wi-Fi, on peut accéder à des ressources telles que la navigation sur Internet, l'accès à des fichiers sur le réseau et envoyer des e-mails via le navigateur.

II.4.1.2. Dongle USB Wi-Fi

Pour accéder à un réseau Wi-Fi on peut utiliser un dongle USB Wi-Fi sur le NVIDIA Jetson Nano. Techniquement, on peut utiliser n'importe quel modèle de dongle Wi-Fi USB, mais le dongle Wi-Fi USB a un pilote de périphérique. Il existe des périphériques qui sont directement pris en charge par NVIDIA sans avoir besoin d'installer leur pilote. La Figure II.11 montre un dongle USB Wi-Fi Netgear connecté à un NVIDIA Jetson Nano. On peut connecter un SSID Wi-Fi à partir de la liste Wi-Fi SSID. Saisir la clé SSID si le SSID Wi-Fi nécessite une authentification.

Figure II. 11. Branchement de l'USB WIFI au NVIDIA Jetson Nano.

II.4.2. SSH

Le NVIDIA Jetson Nano peut être géré à distance en utilisant une application client SSH par exemple, ce serveur a été installé sur le NVIDIA Jetson Nano.

Pour accéder au serveur SSH sous Windows, on doit utiliser un logiciel Putty (Voir la Figure II.12) qui permet d'entrer sur l'interface de commande de NVIDIA, ce logiciel est accessible sur l'adresse (https://www.putty.org). Il suffit de saisir l'adresse IP du NVIDIA Jetson Nano, puis définir l'option SSH sur Putty et cliquer sur le bouton ouvrir.

Après avoir connecté à la carte NVIDIA Jetson Nano, on doit répondre à un test de sécurité, saisir le nom d'utilisateur et le mot de passe de compte enregistré sur le NVIDIA Jetson Nano. Si l'opération réussit, un terminal va apparaitre sur l'écran comme illustré dans la Figure II.13.

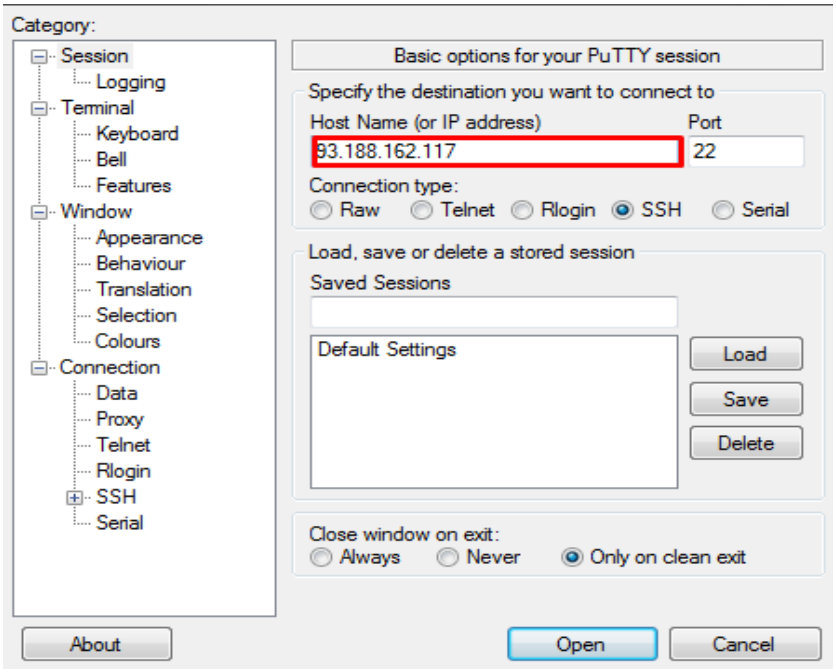


Figure II. 12. Accès à SSH à l'aide de PuTTY sous Windows.

Figure II. 13. Accès à la carte NVIDIA Jetson Nano à l'aide de PuTTYS.

Si on travaille sous Linux ou macOS, on peut accéder au NVIDIA Jetson Nano à l'aide de la commande SSH dans le terminal. Par exemple, on utilise ‘‘NVIDIA’’ comme nom d'utilisateur et « 93.188.162.117 » comme adresse IP du NVIDIA Jetson Nano. La commande sera donc la suivante :

SSH NVIDIA@**93.188.162.117**

La mise à jour de Windows 10 Avril 2018 ou ultérieure possède un client SSH intégré à l'invite de commande. On peut utiliser la commande **ssh** pour accéder au client SSH et par conséquence contrôler la carte NVIDIA Jetson Nano via SSH à distance. Maintenant on peut

gérer la carte, par exemple en installant et en mettant à jour les bibliothèques, on peut également exécuter des programmes via Terminal sur SSH. L'illustration Figure II.14 montre le client SSH dans l'invite de commande utilisé pour accéder à NVIDIA Jetson Nano.

Figure II. 14. Client SSH sur Windows 10 Invite de commande.

II.4.3. VNC Viewer

Virtual Network Computing (VNC) est une application de partage de bureau graphique qui nous permet de surveiller l'interface de bureau d'une machine avec un autre ordinateur ou un appareil mobile à distance. Le visualisateur VNC transmet au serveur VNC à l'aide d'une souris, d'un clavier ou d'un boîtier tactile, et reçoit en retour des mises à jour sur l'écran. Travailler directement sur le NVIDIA Jetson nano n'est pas toujours pratique.

Vous pouvez également inclure une télécommande d'un autre appareil pour travailler sur celui-ci. VNC utilise Real VNC, qui est utilisé avec le NVIDIA OS. Il comprend VNC Viewer, qui permet aux utilisateurs d'accéder à distance à une carte NVIDIA avec bureau, et un serveur VNC permet de surveiller la carte à distance. Ce dernier doit être activé en premier avant d'utiliser le serveur VNC. Le serveur VNC fournit aux utilisateurs une surveillance sans fil au bureau graphique du NVIDIA, ce qui permet la communication. Cependant, le serveur VNC peut être utilisé pour accéder au bureau graphique à distances [39].

II.5. Interfaces et modules de caméra

Techniquement on peut attacher le module caméra via l'interface CSI ou via USB. Le développement de la carte NVIDIA Jetson nano a donné des nouvelles caractéristiques pour chaque modèle, par exemple le modèle NVIDIA Jetson Nano A02 a une seule interface CSI pour caméra. Par contre le modèle NVIDIA Jetson Nano B01 dispose de deux interfaces CSI, comme il est illustré sur la Figure II.15.

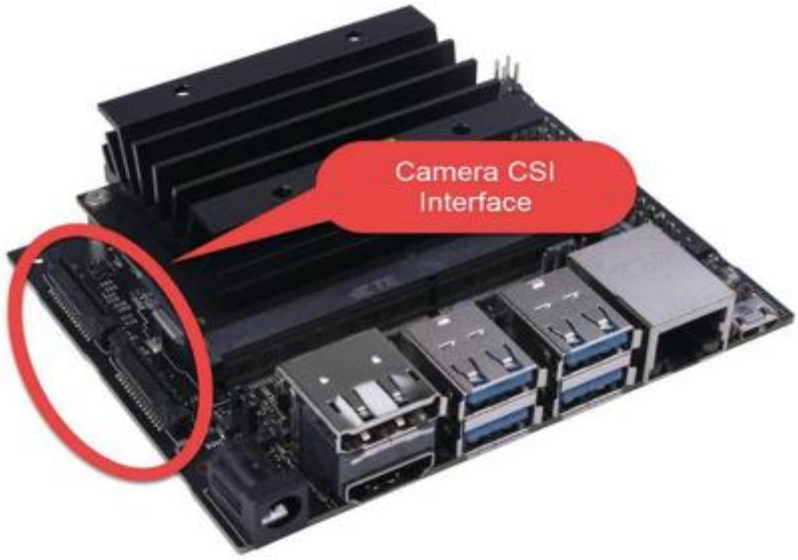


Figure II. 15. Camera CSI interface on NVIDIA Jetson Nano.

On peut utiliser la caméra Raspberry Pi Camera. Il existe également la caméra Raspberry Pi NoIR, qu'on peut voir sur la Figure II.16.

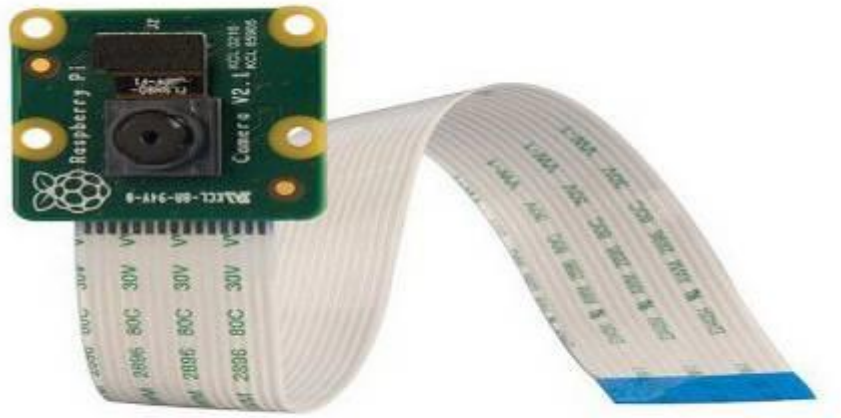


Figure II. 16. Raspberry Pi Camera.

Il est possible également d'utiliser une caméra USB, telle qu'une webcam. Le NVIDIA Jetson Nano possède quatre interfaces USB, donc c'est facile de connecter une caméra USB à l'une des interfaces USB de la carte [38].

II.6. Les ports GPIO

Les GPIO (General Purpose Input Output) sont des broches de port programmables par l'utilisateur et leur direction peut être définie comme d'entrée ou de sortie. Si la broche est programmée comme broche d'entrée, le microcontrôleur peut recevoir des données d'une source externe et si elle est programmée comme sortie le microcontrôleur peut envoyer des données au périphérique externe. De plus, ces broches GPIO ont les capacités suivantes [40]:

- GPIO peut être activé ou désactivé.
- GPIO peut être contrôlé au moment de l'exécution.
- Il peut être laissé non connecté, s'il n'est pas utilisé.
- La broche GPIO permet un haut niveau de réutilisation dans le circuit.
- Elle permet d'isoler le microcontrôleur des appareils bruyants.
- Elle est programmable par l'utilisateur.

Le NVIDIA Jetson Nano contient des broches GPIO sur le J41 comme indiqué par la flèche rouge dans la Figure II.17.

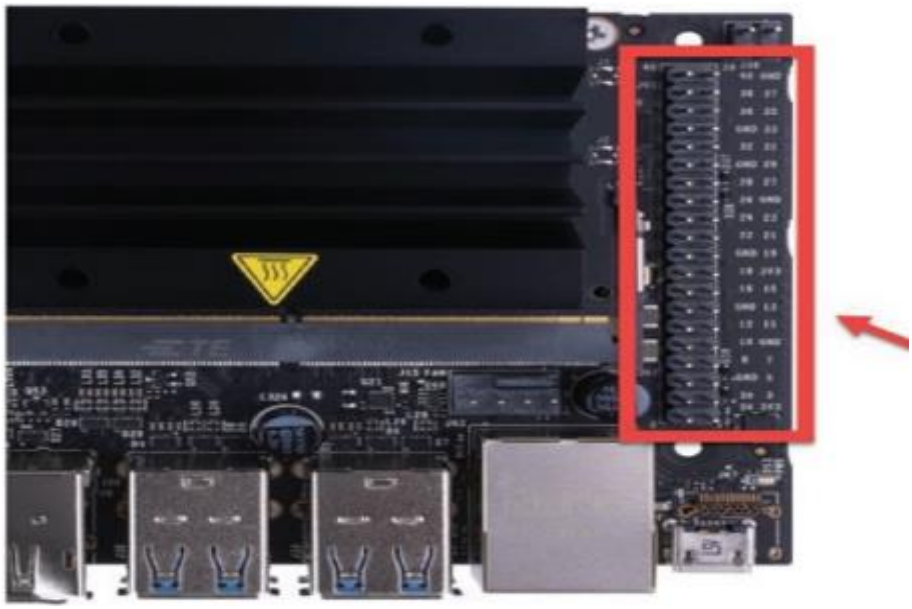


Figure II. 17. Brochage GPIO du NVIDIA Jetson Nano.

Le tableau II.2 représente les broches pour GPIO, UART, SPI, I2S et I2C. Le GPIO du NVIDIA Jetson Nano peut utiliser une tension comprise entre 1,8 et 3,3V. Par défaut, toutes les broches GPIO utilisent 3.3V.

Tableau II. 2. Broches GPIO sur le NVIDIA Jetson Nano.

Nom	***Port***	***Port***	***Nom***
3.3 VDC	1	2	5.0 VDC
I2C_2_SDAI2C Bus 1	3	4	5.0 VDC
I2C_2_SCLI2C Bus 1	5	6	GND
AUDIO_ MCLK	7	8	UART_2_TX/dev/tty THS 1
GND	9	10	UART_2_RX/dev/tty THS 1
UART_2_RTS	11	12	I2S_4_SCLK
SPI_2_SCK	13	14	GND
LCD_TE	15	16	SPI_2_CS1
3.3 VDC	17	18	SPI_2_CS0
SPI_1_MOSI	19	20	GND
SPI_1_MISO	21	22	SPI_2_MISO
SPI_1_SCK	23	24	SPI_1_CS0
GND	5	6	SPI_1_CS1
I2C_1_SDA	27	28	I2C_1_SCL
CAM_AF_EN	29	30	GND
GPIO_PZ0	31	32	LCD_BL_PVVM
GPIO_PE6	33	34	GND
I2S_4_LRCK	35	36	UART_2_CTS
SPI_2_MOSI	37	38	I2S_4_SDIN
GND	39	40	I2S_4_SDOUT

II.7. Raspberry Pi

Le Raspberry Pi est un petit ordinateur qui se connecte à un écran d'ordinateur ou à un téléviseur et fonctionne avec un clavier et une souris ordinaire, comme le montre la Figure II.18 [26]. C'est un petit gadget pratique qui vise à enseigner aux personnes de tous âges les langages de script comme Scratch et Python. Il peut remplir toutes les fonctions d'un ordinateur de bureau, comme surfer sur internet et visualiser des clips de plus grande définition, des feuilles de travail et jouer à des jeux.

Il est doté d'un chipset BCM2387 à quadruple cœur de 1,2 GHz avec un support GPU d'un coprocesseur multimédia à double cœur et vidéo et de GPU, qui comprend un Bluetooth 4.1 (Bluetooth et Bluetooth Classic). Avec le Bluetooth Low Energy (BLE) et le Wi-Fi BCM43143, le Raspberry Pi 3 offre une mise à niveau vers un nouveau processeur principal et un réseau amélioré. En outre, la gestion de l'alimentation de Raspberry 3 a été améliorée, avec une alimentation de 3,5 ampères qui peut gérer des périphériques USB externes plus puissants.

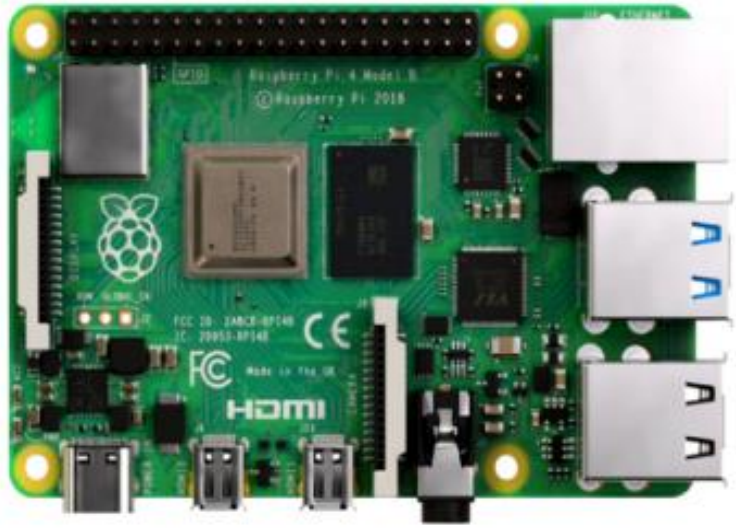


Figure II. 18. Raspberry Pi.

II.8. Aspects Logiciel

II.8.1. Python

Python est l'un des langages de programmation orientés objet open-source les plus populaires. Il possède un code simple et lisible, et il est facile à appliquer dans des processus complexe, des processus de développement de logiciels. Cela contribue à une approche de développement, ainsi que l'apprentissage automatique, l'analyse des données et la reconnaissance des formes [41].

De nos jours Python a été largement utilisé dans divers domaines. Il est utilisé comme frontal de bibliothèques d'apprentissage automatique, telles que TensorFlow et PyTorch et il est utilisé pour le développement de l'arrière-plan des applications Web, par exemple Flask, Tornado et Django [42].

II.8.1.1. Outils et bibliothèques

Os : c'est une bibliothèque python qui fournit des fonctions permettant d'interagir avec le système d'exploitation, en particulier, avec le système de fichiers. Elle fait partie des modules utilitaires standard de Python. Elle fournit un moyen portable d'utiliser les fonctionnalités dépendantes du système d'exploitation. Nous avons utilisé cette bibliothèque pour parcourir les répertoires contenant nos

collections une fois celles-ci construites et afin de les utiliser pour l'entraînement et l'évaluation [23].

Matplotlib : est une bibliothèque scientifique de données standard qui aide à générer des visualisations de données telles que des diagrammes et des graphiques bidimensionnels (histogrammes, diagrammes de dispersion, graphiques de coordonnées non cartésiennes). Matplotlib est l'une de ces bibliothèques de tracés qui sont vraiment utiles dans les projets de science des données – elle fournit une API orientée objet pour intégrer des tracés dans les applications.

C'est grâce à cette bibliothèque que Python peut visualiser avec des outils scientifiques comme MatLab ou Mathematica. Cependant, les développeurs doivent écrire plus de code que d'habitude en utilisant cette bibliothèque pour générer des visualisations avancées.

NumPy (pour Numerical Python) : est un outil parfait pour le calcul scientifique et la réalisation d'opérations de base et avancées avec des tableaux. La bibliothèque offre de nombreuses fonctionnalités pratiques permettant d'effectuer des opérations sur des tableaux (n-arrays) et des matrices en Python. Elle permet de traiter des tableaux qui stockent des valeurs du même type de données et facilite l'exécution d'opérations mathématiques sur les tableaux (et leur vectorisation). En fait, la vectorisation des opérations mathématiques sur le type de tableau NumPy augmente les performances et accélère le temps d'exécution [26].

Jupyter : est une application web gratuite et open-source permettant aux utilisateurs d'écrire des documents contenant du texte explicatif, des équations et des visualisations, ainsi que des codes en direct et leurs résultats d'exécution, est devenu extrêmement populaire de nos jours [43] pour le développement interactif de logiciels et l'analyse de données. Il est devenu un cadre standard de facto pour le développement d'applications dans plusieurs domaines tels que la science des données ou l'intelligence artificielle [44].

Les carnets Jupyter sont un moyen de rendre la science plus ouverte. Leur pertinence pour la communauté de la JCDL réside dans leur interaction avec de multiples composants de l'infrastructure des bibliothèques numériques, tels que les identifiants numériques, les mécanismes de persistance, le contrôle de version, les ensembles de données, la documentation, les logiciels et les publications [45].

TensorFlow : est une bibliothèque logicielle d'apprentissage automatique qui est open source et gratuite. Elle a été créée pour effectuer de grands calculs numériques sans tenir compte de l'apprentissage profond. Cet outil peut être utilisé pour une variété d'activités, mais il est principalement axé sur l'inférence et la formation de réseaux neuronaux profonds et il prend également en charge l'apprentissage automatique traditionnel. Cette bibliothèque Python est développée par Google permet de réaliser rapidement des calculs numériques. TensorFlow peut être utilisée comme une bibliothèque de base pour générer directement les modèles d'apprentissage profond, comme elle peut être utilisée pour simplifier le processus en utilisant les bibliothèques de wrapper de TensorFlow. Elle permet de créer des graphes et des structures de flux de données pour déterminer comment les données circulent dans le graphe, en recevant des entrées sous forme de tableau tensoriel multidimensionnel. Elle permet de construire un diagramme de flux pour ces entrées, qui est exécuté à une extrémité et est exécuté à l'autre [39].

OpenCV : est une bibliothèque de vision par ordinateur multiplateforme avec une source ouverte qui a commencé comme un projet de recherche chez Intel en 1998, disponible depuis 2000. OpenCV a pour but de fournir l'outil nécessaire pour résoudre les problèmes de vision par ordinateur, il peut être préparé par les langages C et C++, contient un mélange de fonctions de traitement d'image de bas niveau et d'algorithmes de haut niveau [46].

Dans l'interface Python, toutes les dernières avancées et algorithmes apparaissent. Il s'agit de la principale bibliothèque open-source de vision par ordinateur, d'apprentissage automatique et de traitement d'images. Il joue un rôle important dans l'activité en temps réel des systèmes actuels. Nous l'utiliserons pour traiter des photographies et des vidéos afin de reconnaître des êtres humains, des visages ou même des écritures manuscrites. Python est capable de traiter la structure de tableau OpenCV pour l'examiner, car il est combiné avec différentes bibliothèques telles que NumPy. Nous utilisons l'espace vectoriel pour reconnaître le motif de l'image et ses différentes caractéristiques et effectuer des calculs arithmétiques sur ces traits. Il est accessible sous Windows, Linux, iOS, etc, avec Python, C++, C et Java comme interfaces [47].

Google Colaboratory : également connu sous le nom de Colab, est un service en nuage basé sur Jupyter Notebooks pour diffuser l'enseignement et la recherche en apprentissage automatique. Il fournit un runtime entièrement configuré pour l'apprentissage profond et un accès gratuit à un GPU robuste [48].

L'apprentissage profond et l'apprentissage automatique peuvent être réalisés à l'aide de ressources basées sur le Cloud Computing pour exécuter TensorFlow, Keras, Caffe, PyTorch et d'autres codes basés sur Python. Microsoft Colaboratory: Cloud d'apprentissage profond et de Big data basé sur des GPU gratuits. Google propose des recherches approfondies et des calculs mathématiques avec le Free Cloud basé sur le GPU Tesla. Les services de cloud offrent 4992 CUDA et une bande passante mémoire de 480 Go/sec (240 Go/sec par GPU). Les codes Keras, Tensorflow et PyTorch peuvent être déployés directement sur le GPU Tesla K80 gratuit [49].

Que propose Colab ?

- Écrire et exécuter du code en Python.
- Documentez votre code qui prend en charge les équations mathématiques.
- Créer/Télécharger/Partager des blocs-notes.
- Importer/enregistrer des blocs-notes depuis/vers Google Drive.
- Importer/publier des blocs-notes depuis GitHub.
- Importez des ensembles de données externes, par exemple de Kaggle.
- Intégrer PyTorch, TensorFlow, Keras, OpenCV.
- Service Cloud gratuit avec GPU gratuit.

Colab utilisant implicitement Google Drive pour stocker vos blocs-notes. Nous ouvrons l'URL suivante dans notre navigateur: https://colab.research.google.com/. Notre navigateur affichera l'écran suivant (en supposant que nous sommes connectés à notre Google Drive).

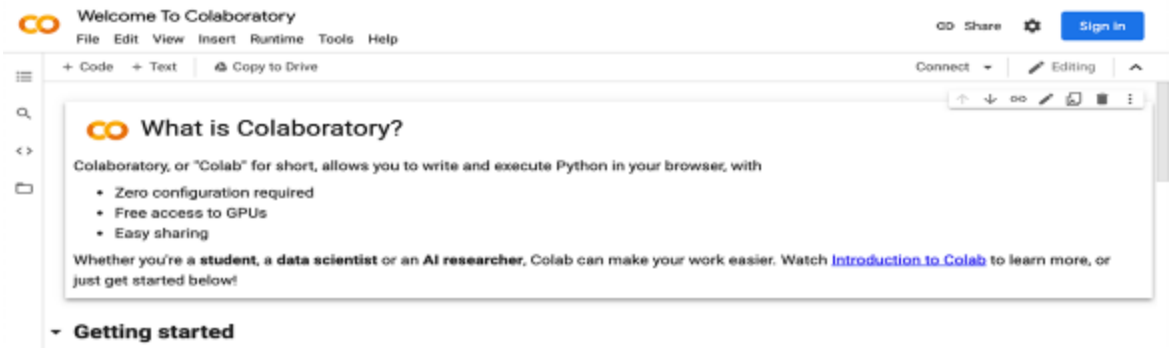


Figure II. 19. La page d'accueil de la plateforme.

- Cliquer sur fichier et choisissez nouveau notebook

Figure II. 20. Illustration de comment ouvrir un notebook dans la plateforme Colab.

- Lorsque nous ouvrons un nouveau notebook, nous serons prêts à écrire le code dans la cellule de code.

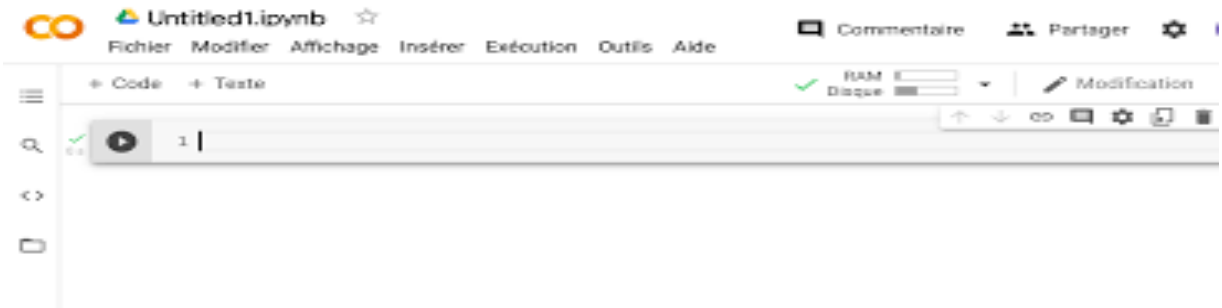


Figure II. 21. Illustration de nouveau notebook.

- si vous souhaitez ajouter une cellule de code, cliquez sur l'icône+code, ou sur insérer =>cellule de code

Figure II. 22. Illustration pour montrer comment ajouter une cellule de code.

- Pour exécuter une cellule particulière, on la sélectionne et on appuie sur les touches Ctrl+entrer ou bien l'icône correspondante, pour exécuter toutes les cellules exécution->tout exécuter.

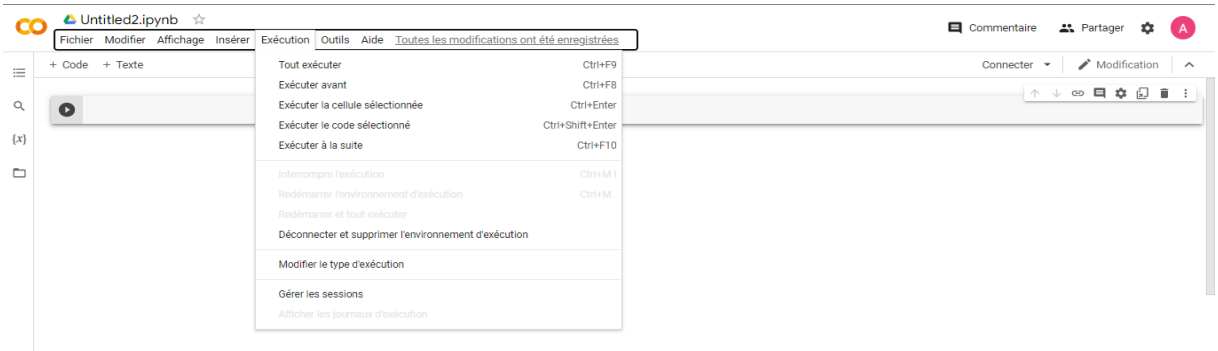


Figure II. 23. Illustration pour montrer comment exécuter.

Comme nous savons que les bibliothèques python couramment utilisées sont préinstallées et pour les nouvelles, nous pouvons les installer en utilisant les syntaxes suivantes :

- ***! pip install [nom du package]***

où

- ***! apt-get install [nom du package]***

Exemple :

```
!pip install numpy
```

```
!apt-gpt -qq install -y libfluidsynth1
```

Ensuite, Si nous voulons l'importer, nous utilisons la fonction ‘‘import’’ comme ci-dessus :

```
import numpy as np
```

Pour importer les fichiers locaux en utilise l'écriture ci-dessus :

- ***From google.colab import files***
- ***Uploaded=files.upload()***

On clique sur «sélect fichiers» et on importe le fichier souhaité

Figure II. 24. Illustration comment importer les fichiers.

Pour utiliser les fichiers depuis Google drive, on exécute la commande :

- ***From google. colab import drive***
- ***Drive.mount(‘/content/drive’)***

Ensuite, un lien sera généré, il faut cliquer sur le lien, autoriser l'accès puis copier le code qui apparait et le coller dans «enter your autorisation code». Maintenant, pour voir toutes les données de Google drive, on devra exécuter ce qui suit :

- ***!ls ‘‘/content/drive/Mydrive/’’***

Figure II. 25. Visualisation de contenu de notre Google Drive.

II.8.1.2. Entrainer sur GPU et TPU

GPU (en anglais ‘‘Graphics processing unit’’, ou en français processeur graphique) est une unité de calcul (circuit intégré ou puce) présent sur une carte mère, ou encore intégré développé par Google spécifiquement pour accélérer les systèmes d’intelligence artificielle par réseaux de neurones. Le TPU (en anglais ‘‘Tensor processing unit’’, ou en français ‘‘unité de traitement de tenseur’’) est un circuit intégré développé par Google spécifiquement pour accélérer les systèmes d’intelligence artificielle par réseaux de neurones. Pour passer en mode GPU ou TPU dans la barre des options choisir ‘‘exécution’’ puis ‘‘Modifier le type d’exécution’’ et mètre l’option accélérateur matériel en mode GPU ou bien TPU [26].

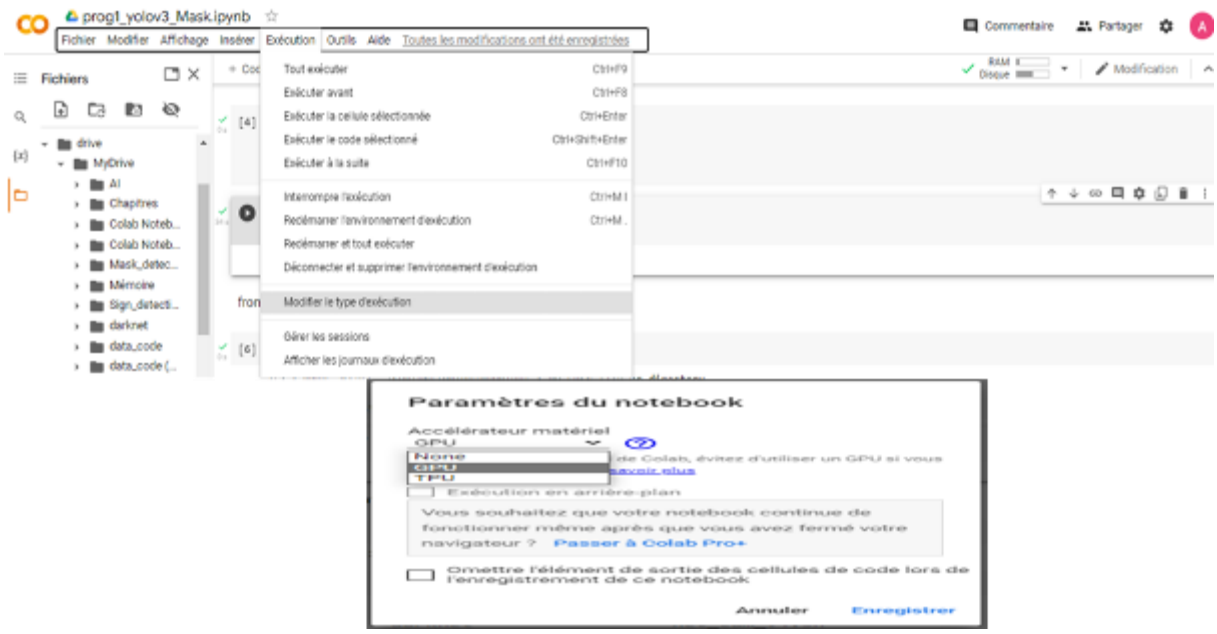


Figure II. 26. Illustration pour montrer comment configurer le type d’exécution.

II.9. Conclusion

Dans ce chapitre, nous avons présenté la carte NVIDIA Jetson Nano. Nous avons abordé les points suivants :

- Définition de la NVIDIA Jetson Nano.
- L'exploration des spécifications techniques du NVIDIA Jetson Nano.
- L'exploration des fonctionnalités du NVIDIA Jetson Nano.
- L'installation du système d'exploitation.
- Les caractéristiques de la carte NVIDIA Jetson Nano.
- La configuration de la carte NVIDIA.
- L'accès à distance à la carte NVIDIA.

Ensuite, nous avons présenté comment travailler avec une caméra sur la carte NVIDIA Jetson Nano. Puis nous passons parler de manière générale des ports GPIO et les aspects Logiciel comme Python.

Enfin, nous avons terminé par les bibliothèques d'apprentissage automatique sur la carte NVIDIA Jetson Nano tel que : Jupyter, Tensor Flow, OpenCV, Google Colaboratory.

Après avoir expliqué l'aspect matériel et logiciel de la carte NVIDIA Jetson Nano, nous mesurons que cela sera suffisant et pourra nous permettre de passer à la pratique.

En se basant sur les informations détaillées dans ce chapitre, on va développer nos modèles de détection de visage et d'autre modèles basés sur YOLOv3 et YOLOv4 en utilisant le transfert learning pour la détection de masque sur un visage et de panneaux de signalisation routiers dans le chapitre qui suit.

Chapitre III

Domaines d'application de l'apprentissage profond

III.1. Introduction

Ce chapitre est la partie implémentation de méthodes théoriques étudiées dans les chapitres précédents, où nous présentons trois applications principales, en s'appuie sur des algorithmes entre eux qui est classique; comme l'algorithme de viola and Jones pour la reconnaissance faciale; et d'autres intelligents qui se basent sur les méthodes d'intelligence artificielle comme YOLOv3-tiny et YOLOv4-tiny pour des applications de détection de masque et de panneaux de signalisation routiers. Dans un premier temps, nous expliquons le principe de fonctionnement de méthode de viola and Jones, puis nous présentons les deux différentes bases de données utilisées pour l'entraînement et l'optimisation des modèles YOLOv3-tiny et YOLOv4-tiny. Le premier ensemble de données sera utilisé pour la détection de masque sur le visage d'une personne présente sur une image, et le deuxième ensemble sera utilisé pour la détection des panneaux de signalisation routière.

Les étapes de développement de notre solution sera expliquée en détail en commençant de l'importation des bibliothèques nécessaires, les configurations essentielles jusqu'à l'évaluation des modèles entrainés et nous finissons par une comparaison des performances des modèles obtenus.

III.2. Systèmes de sécurité intelligents

L'intelligence artificielle et la sécurité sont -à bien des égards- faites l'une pour l'autre, et les approches modernes de l'apprentissage automatique semblent arriver juste à temps pour combler les lacunes des anciens systèmes de sécurité des données basées sur des règles [50].

Les problèmes pour doter les systèmes artificiels de capacités intelligentes sont, en fait, uniques. Nous voulons toujours atteindre un objectif général, qui est un meilleur fonctionnement du système intelligent que celui qui peut être accompli par un système sans composants intelligents. Il existe de nombreuses façons d'atteindre cet objectif et, par conséquent, nous avons de nombreux types de systèmes artificiels intelligents (IA). En général, il convient de souligner qu'il existe deux groupes distincts de chercheurs travaillant dans ces domaines : la communauté de l'IA et la communauté de l'intelligence informatique. Sur ce que nous nous intéressons dans ce chapitre est l'implémentation des systèmes de sécurité dans le domaine d'intelligence artificielle [51].

Ce mémoire a pour but d'implémenter plusieurs méthodes d'intelligence artificielle (classification, deep learning) sur la carte Nvidia Jetson nano, pour l'objectif de construire des systèmes de sécurité intelligents. Pour ce faire, nous avons mis en place un ensemble d'outils

nous permettant de collecter, stocker et analyser plusieurs type de données (visage, masque, signalisation). À partir de ces données nous avons fait trois contributions.

Dans une première contribution, nous présentons la méthode de reconnaissance faciale Viola-Jones V-J. Ce travail a pour but est de permettre à la carte Nvidia Jetson nano de comprendre ce qu'elle «voit» lorsqu'on la connecte à une ou plusieurs caméras. Avec la généralisation de l'utilisation des images numériques, l'analyse du mouvement dans les vidéos s'est révélée être un outil indispensable pour des applications aussi diverses que la vidéo surveillance pour les systèmes de sécurité, la robotique, l'interaction homme machine.

Ensuite, nous présentons une série de simulations basées sur des techniques de détection d'objet. Finalement, nous présentons l'implémentation de ces techniques sur deux domaines différents, à savoir la détection de masque et les systèmes de sécurité routière.

III.3. Détection de visage

La détection de visages est la première étape dans le processus de reconnaissance faciale. Son efficacité a une influence directe sur les performances du système de reconnaissance de visages. Il existe plusieurs méthodes pour la détection de visages, certaines utilisent la couleur de la peau, la forme de la tête, l'apparence faciale, alors que d'autres combinent plusieurs de ces caractéristiques. Les méthodes de détection de visages peuvent être subdivisées en quatre catégories [52]:

- Approches basées sur les connaissances acquises.
- Approches basées sur le «Template-matching».
- Approches basées sur l'apparence.
- Approches basées sur des caractéristiques invariantes.

III.3.1. Viola-Jones

L'algorithme de détection de visage Viola-Jones V-J (voir la Figure III.1) scanne une image avec une fenêtre à la recherche des caractéristiques d'un visage humain. Si ces caractéristiques sont trouvées et ont une valeur particulière comme un visage, alors la fenêtre particulière de l'image est estimée comme étant un visage. Pour résoudre le cas de visages de tailles différentes, la fenêtre est mise à l'échelle avec le processus répété pour chaque image. En réduisant le nombre de caractéristiques que chaque fenêtre doit vérifier, la fenêtre est passée par plusieurs étapes différentes, les premières étapes ayant moins de caractéristiques à vérifier et étant plus faciles à passer, tandis que les étapes ultérieures ont plus de caractéristiques et sont plus sélectives. Les caractéristiques calculées pour chaque étape sont ensuite accumulées, si la caractéristique accumulée ne passe pas le seuil, cela signifie que

l'étape est échouée et que l'on ne peut pas en tenir compte. La fenêtre actuelle est estimée ne pas contenir de visage. Pour faciliter la compréhension de l'algorithme, certains termes doivent être définis, notamment les caractéristiques, l'image intégrale et l'étape de classification cascade [53].

III.3.2. Principe

Afin de détecter la position des visages dans les images de sorte à obtenir une région d'intérêt sur laquelle l'extraction des vecteurs de caractéristiques pourra être accomplie, l'algorithme de détection de J-V est utilisé [5].

L'algorithme V-J se compose de quatre étapes principales qui sont:

- Sélection de fonctionnalités de type Haar.
- Créer une image intégrale.
- Entrainement d'Algorithme AdaBoost.
- Créer un ensemble de classificateur en cascade.

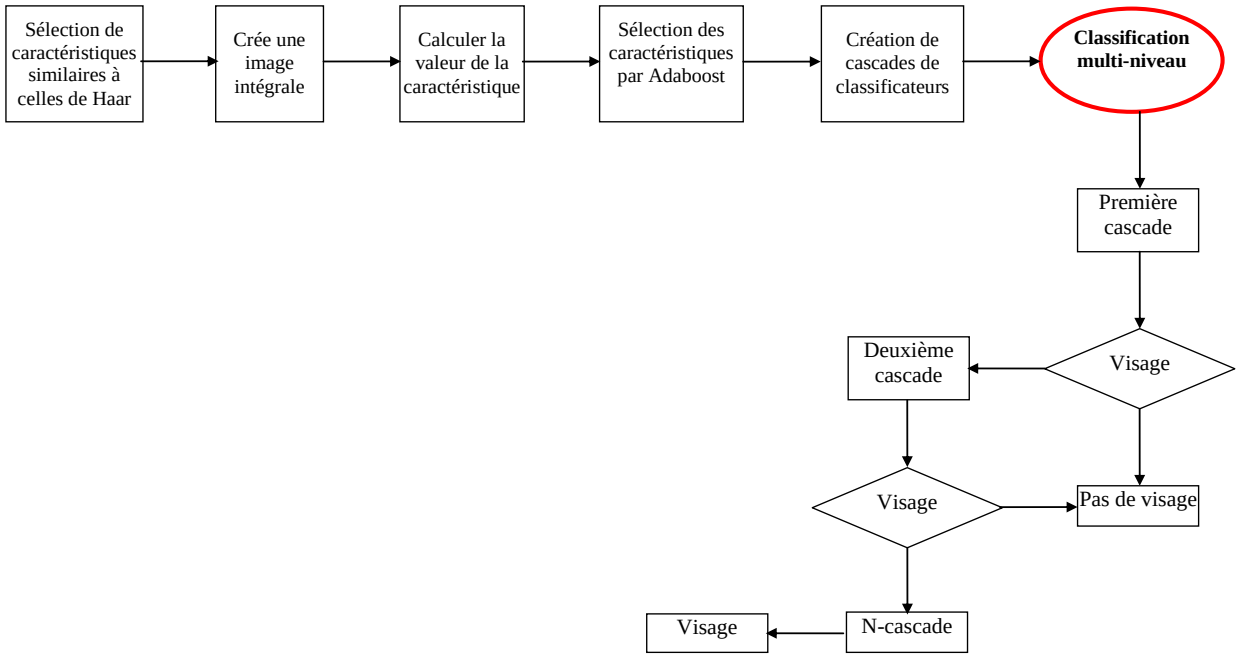


Figure III. 1. V-J en Général.

III.3.3. Sélection des caractéristiques

Des caractéristiques appelées classificateurs de Haar sont utilisées dans l'algorithme V-J pour détecter les caractéristiques d'un visage. Les caractéristiques Haar sont utilisées en vision par ordinateur pour classer l'intensité des pixels dans une région de manière traçable. Haar sont représentées sous forme de régions rectangulaires de l'image, et les classificateurs sont composés de deux ou trois caractéristiques rectangulaires pour rechercher en permanence les

caractéristiques dans la fenêtre. Un exemple de caractéristiques Haar est montré à la Figure III.2. Pour plus de détails sur ces caractéristiques, nous nous référons au travail original de Viola et de Jones [5].

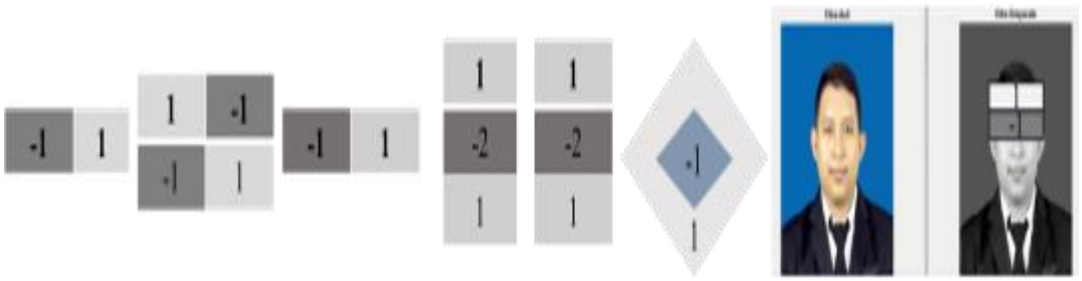


Figure III. 2. Les paramètres de Haar (Haar feature).

III.3.4. Extraction des caractéristiques

L'algorithme de V-J utilise des caractéristiques de type Haar, c'est-à-dire un produit scalaire entre l'image et des certains modèles de type Haar, plus précisément, laissons *I* et *P* désigner une image et un modèle, tous les deux de la même taille *N*×*N* (voir la Figure III.3).

La caractéristique associée au motif *P* de l'image *I* est définie comme suit :

Figure III. 3. Caractéristiques de type Haar.

Ici comme ci-dessous (Figure III.4), l'arrière-plan d'un modèle comme (b) est peint en gris pour mettre en évidence le support du modèle, seuls les pixels marqués en noir ou en blanc sont utilisés lors du calcul de la caractéristique correspondante est calculée.

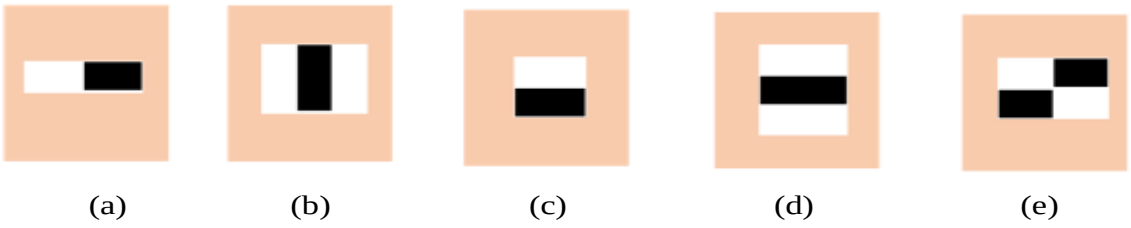


Figure III. 4. Cinq motifs de type Haar.

La taille et la position du support d'un motif peuvent varier à condition que ses rectangles noirs et blancs aient la même dimension, qu'ils se bordent les uns les autres et qu'ils gardent leur position relative, grâce à cette contrainte, le nombre de caractéristiques que l'on peut tirer d'une image est quelque peu gérable: une image 24×24, par exemple, possède 43200, 27600, 43200, 27600 et 20736 caractéristiques de catégories (a), (b), (c), (d) et (e), respectivement, soit 162336 caractéristiques en tout.

En pratique, cinq modèles sont considérés (voir la Figure III.4), et l'algorithme représenté sur la Figure III.1. Les caractéristiques dérivées sont supposées contenir toutes les informations nécessaires pour caractériser les visages étant par nature réguliers, l'utilisation de motifs de type Haar semble justifiée. Il existe cependant un autre élément crucial qui permet à cet ensemble de caractéristiques de prendre le dessus: l'image intégrale qui permet de les calculer à très faible coût, au lieu de faire la somme de tous les pixels à l'intérieur d'un rectangle [53].

III.3.5. Sélection par Boosting

Comme les dimensions des ensembles de données dans la modélisation prédictive continuent de croître, la sélection de caractéristiques devient de plus en plus pratique.

Les ensembles de données avec des interactions de caractéristiques complexes et des niveaux élevés de redondance représentent toujours un défi pour les méthodes de sélection de caractéristiques existantes.

Adaboost et la sélection de caractéristiques d'ensemble sont deux domaines actifs de la recherche sur les ensembles. Adaboost, communément appelé Boosting, est une technique efficace pour la construction d'ensembles, qui influence les différentes décisions du classificateur en utilisant la repondération des exemples. Son fonctionnement est étroitement associé à la théorie de la grande marge.

La sélection de caractéristiques d'ensemble construit un ensemble en utilisant différents sous-ensembles de caractéristiques pour chaque apprenant de base, cela fournit un moyen potentiellement plus actif de promouvoir la diversité des décisions par rapport au partitionnement ou à la repondération des exemples de formation.

Malheureusement, le succès de la construction d'un ensemble de cette façon est moins bien compris sur le plan théorique, la combinaison de ces deux techniques est intéressante pour deux raisons.

Premièrement, pour les cas où le Boosting est peu performant ou échoue, lorsque les données sélectionnées et la paire de classificateurs donnent une diversité d'erreurs inadéquate pour que l'assemblage soit bénéfique. La dynamisation échoue si l'erreur d'apprentissage est nulle en utilisant la description des caractéristiques. Cela se produit lorsque les données

d'entraînement sont petites et peuvent ne pas refléter la réalité, la véritable séparation des classes de l'ensemble de données, compte tenu de données suffisantes.

Deuxièmement, pour réduire le nombre de caractéristiques utilisées par chaque classificateur de base dans l'ensemble, les premiers exemples combinant Boosting et sélection de caractéristiques étaient Boosting Decision Stumps et Domain-Partitioning. Dans les deux cas, la motivation était d'améliorer l'ajustement du classificateur de base pour correspondre à celui d'un apprenant faible.

III.3.6. Classificateur en cascade

Le classificateur en cascade est le processus d'organisation d'un ensemble de caractéristiques dans une forme de classification à plusieurs niveaux. Il y a au moins trois classifications pour déterminer s'il existe ou non des caractéristiques faciales dans la zone de caractéristiques sélectionnée. Dans la première étape du filtre de classification, chaque sous-image sera classée en utilisant une caractéristique, si la valeur de la caractéristique sur le filtre ne répond pas aux critères attendus, elle sera rejetée. L'algorithme passe ensuite à la sous-fenêtre suivante et calcule la valeur de la caractéristique. Si les résultats sont conformes au seuil souhaité, il passe à la deuxième étape de filtrage jusqu'à ce que le nombre de sous-fenêtres qui passent la classification diminue jusqu'à ce qu'il s'approche de la limite inférieure. Ce qui se passe, la classification diminue jusqu'à s'approcher de l'image de visage détecté. La Figure III.5, représente un processus de filtrage qui est passé par chaque classificateur [53].

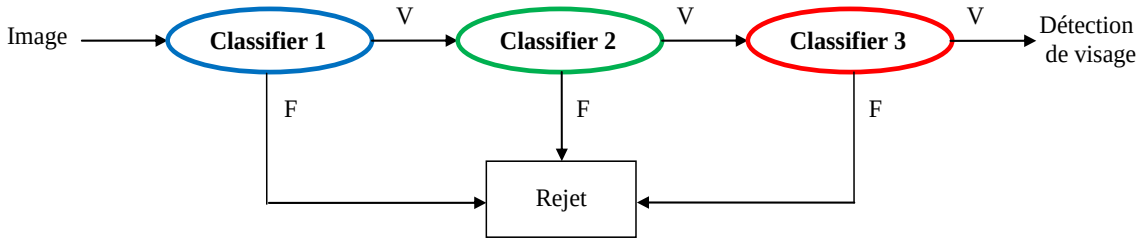


Figure III. 5. Cascade classifier.

III.4. Détection d'objet

La détection d'objets est une forme avancée de classification d'images dans laquelle un réseau neuronal prédit des objets dans une image et les signaler sous forme de boîtes délimitation (boxes).

La détection d'objets fait donc référence à la détection et à la localisation d'objets dans une image qui appartiennent à un ensemble prédéfini de classes (voir la Figure III.6).

Des tâches telles que la détection, la reconnaissance ou la localisation sont largement utilisables dans diverses applications utiles [54], telles que la détection des visages, la surveillance vidéo intelligente, l'automobile et le suivi d'objets, etc. [55], ce qui fait de la

détection d'objets (également appelée reconnaissance d'objets) un sous-domaine très important de la vision par ordinateur [54, 55].

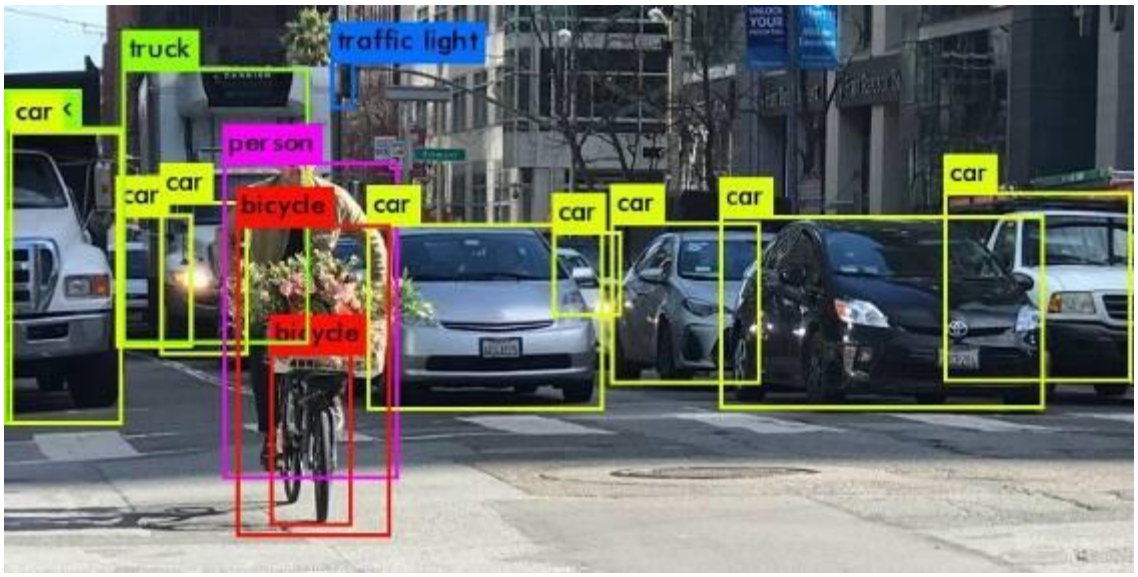


Figure III. 6. Un exemple de détection d'objet.

Les cadres de détection d'objets basés sur CNN peuvent être principalement divisés en détecteurs à une étape et à deux étapes.

Les détecteurs à une étape n'incluent pas de couche de proposition de région, ils extraient la caractéristique dans les images originales pour analyser la régression et la localisation des objets.

YOLO considère la détection d'objet comme une tâche de classification pour séparer les boîtes de délimitation spatialement et les probabilités de classe.

SSD traite les objets de tailles différentes en utilisant des boîtes de délimitation multi-échelles sur des cartes de caractéristiques [55].

La détection d'objets en deux étapes comme représente la Figure III.7, fait référence à l'utilisation d'algorithmes qui décomposent le problème de la détection d'objets en deux étapes:

- Détecter les régions d'objets possibles.
- Classification de l'image dans ces régions en classes d'objets.

Les algorithmes populaires en deux étapes comme Fast-RCNN et Faster-RCNN utilisent généralement un réseau de proposition de région qui propose des régions d'intérêt susceptibles de contenir des objets [54].

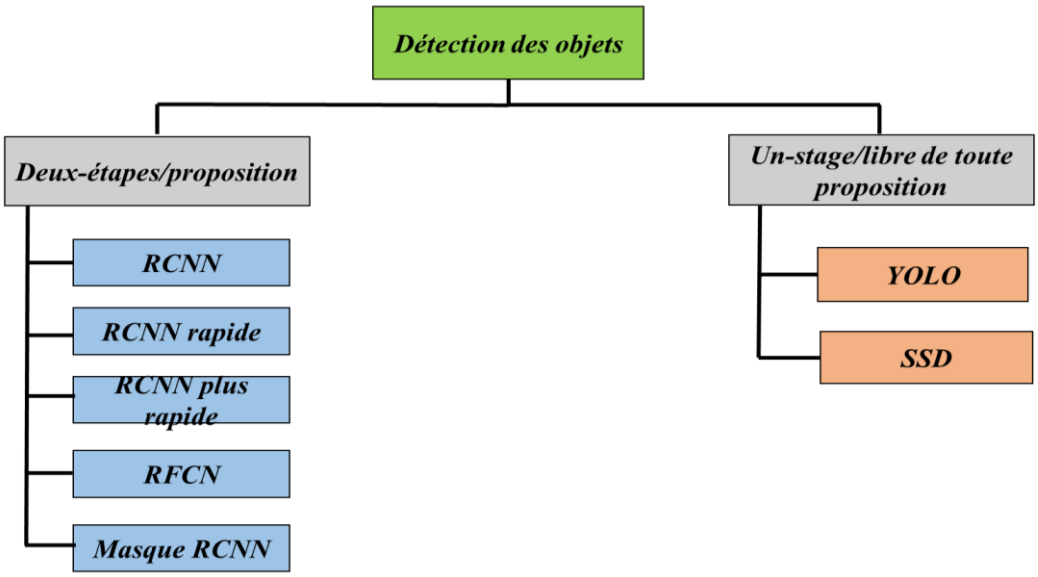


Figure III. 7. Détecteurs à un et deux étapes.

III.4.1. YOLO

YOLO You Only Look Once (Vous ne regardez qu'une fois) est un algorithme proposé par l'auteur Redmond et al. [56], dans un article de recherche publié à la conférence IEEE/CVF sur la vision informatique et la reconnaissance des formes (CVPR) en tant que document de conférence, et qui a remporté le prix OpenCV People's Choice Award.

Par rapport à l'approche adoptée par les algorithmes de détection d'objets avant YOLO, qui réutilisent les classificateurs pour effectuer la détection, YOLO propose l'utilisation d'un réseau neuronal de bout en bout qui effectue des prédictions des boîtes englobantes et des probabilités de classe en une seule fois.

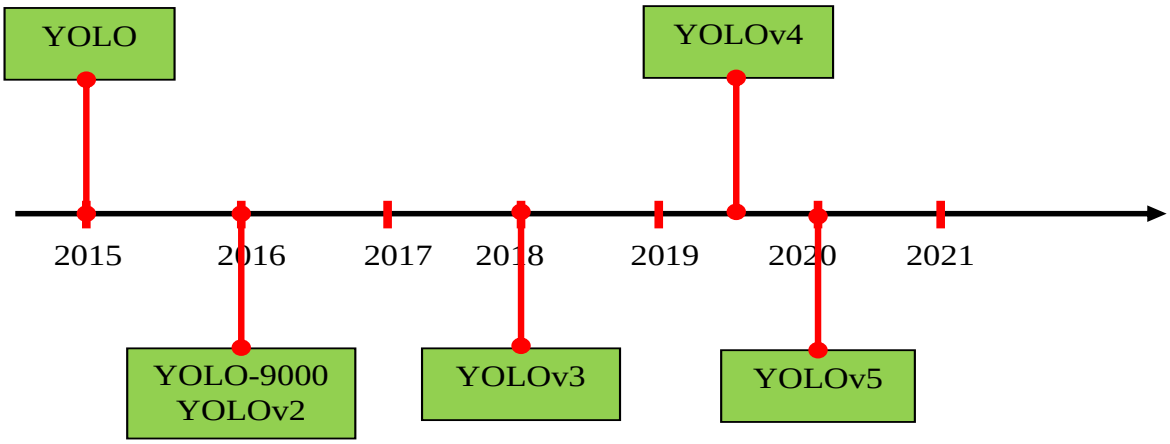


Figure III. 8. Chronologie de YOLO.

YOLO traite la tâche de détection comme une régression vers le bas, il a été largement utilisé dans les domaines du traitement de l'image.

Selon la Figure III.8, plusieurs versions de YOLO (par exemple, YOLOv1, YOLOv2 et YOLOv3) ont en outre été prévues. Par rapport à YOLOv1 et YOLOv2, YOLOv3 présente les avantages suivants :

- L'amélioration des performances de classification sur des ensembles de données complexes.
- La quantité accrue de données dans la carte des caractéristiques.
- Les couches de réseau plus profondes.

Le réseau YOLO est trop grand et sa structure est trop compliquée pour être implémentée sur des appareils embarqués ou mobiles. Afin de résoudre ce problème, les chercheurs ont proposé une série de YOLO légers, notamment :YOLOv2-tiny, YOLOv3-tiny et YOLOv4-tiny [57].

III.4.2. YOLOv3

Pour améliorer YOLO avec les CNN modernes qui utilisent des réseaux résiduels et des connexions sautées, YOLOv3 a été proposé. Voici comment il fonctionne, tel que présenté par Joseph Redmond.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure III. 9. Architecture de réseau YOLOv3.

Alors que YOLOv2 utilise le DarkNet-19 comme architecture de modèle, YOLOv3 utilise un DarkNet-53 beaucoup plus complexe comme colonne vertébrale du modèle. Un réseau neuronal à 106 couches comprenant des blocs résiduels et des réseaux de sur-échantillonnage, l'architecture réseau de YOLOv3 est illustrée à la Figure III.9 [54, 58].

La nouveauté architecturale de YOLOv3, permet de prédire à trois échelles différentes, les cartes de caractéristiques étant extraites aux couches 82, 94 et 106 pour ces prédictions.

En détectant des caractéristiques à 3 échelles différentes, YOLOv3 compense les lacunes de YOLOv2 et YOLO, en particulier dans la détection des petits objets. L'architecture permettant la concaténation des sorties des couches sur-échantillonnées avec les caractéristiques des couches précédentes, les caractéristiques à grain fin qui ont été extraites sont préservées, ce qui facilite la détection des petits objets [54].

YOLOv3 utilisera directement les images et les annotations de la saisie initiale pour l'encadrement, par conséquent, il économise les ressources informatiques. Dans la tâche de détection, une image est tout d'abord envoyée en entrée du réseau d'extraction, et des vecteurs de caractéristiques extraits qui sont ensuite envoyés à une structure comme le réseau de pyramide de caractéristiques, et donc la cellule de grille est obtenue à trois échelles.

En outre, chaque cellule de la grille prédit trois boîtes englobantes, et chaque boîte englobante prédit un vecteur *Y*, comme suit :

$$Y = (t_x + t_y + t_w + t_h) + Y_0 + (Y_1 + Y_2 + ... + Y_n) \quad \text{(III.1)}$$

avec :

$$Y_0 = \Pr(Object) \times IoU(truth, pred) \quad \text{(III.2)}$$

où t_x, t_y, t_w, t_h est que les coordonnées associées au la boîte englobante. *Pr*(*Object*) représente la probabilité que la chose se trouve dans la boîte de prédiction.

La suppression du non-maxima (NMS) est utilisée pour résoudre le problème du chevauchement des boîtes de délimitation dans la reconnaissance du même objet. Les NMS utilisent une fonction essentielle appelée Intersection sur Union, ou IoU. Ou l'illustration d'IoU est présentée à la Figure III.10 [58, 59].

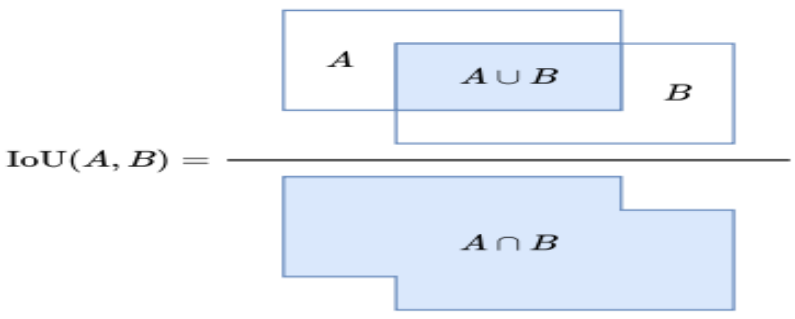


Figure III. 10. IoU.

III.4.3. Modèle YOLOv4

Le cadre du YOLOv4 se compose de trois parties principales: le cou, le tronc et la tête. La partie tronc est principalement composée de CSP Darknet53 comme montre la Figure III.11, qui est principalement composé du réseau résiduel à cinq couches Resblock_body. La taille de l'image d'entrée est de 416×416, où Resblock_body a une convolution spéciale pour réduire la résolution.

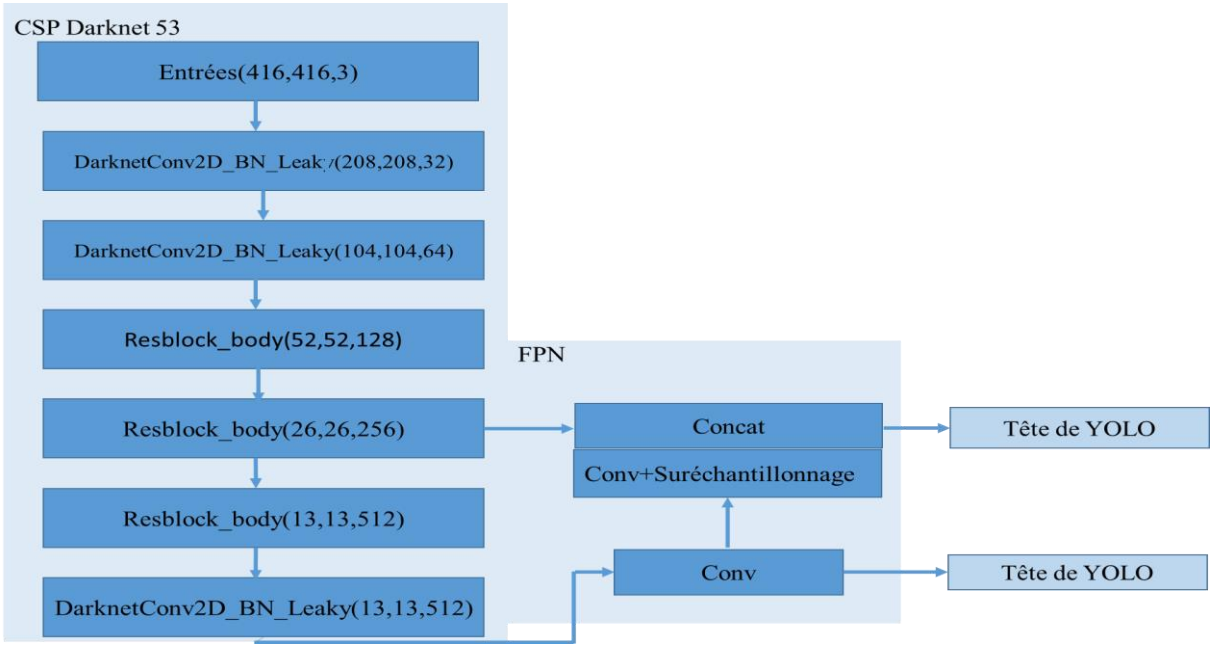


Figure III. 11. Architecture de CSP Darknet 53.

Le Resblock_body de chaque couche réduit progressivement le pixel par deux, et sa fonction principale est d'extraire les informations caractéristiques des données de l'image. Le cou est principalement composé de SPP et de PAN. La fonction principale de SPP est d'augmenter l'effet du champ réceptif. L'ensemble du processus de SPP n'a rien à voir avec la taille de l'entrée et peut traiter des boîtes de candidats de n'importe quelle taille. La fonction principale de PAN est de transformer les informations extraites en coordonnées, catégories et autres informations.

Il est principalement composé d'un échantillonnage vers le haut et d'un échantillonnage vers le bas. La tête continue de suivre la tête de détection de YOLOv3. Les principales fonctions des trois têtes de détection sont les suivantes:

Une fois la sortie, la fonction de perte est calculée en comparant avec l'annotation des données réelles. La fonction de perte est principalement composée de trois parties: perte de positionnement, perte de confiance et perte de classification [60].

III.4.4. Critères d'évaluation

Les critères d'évaluation des résultats des expériences sont principalement la précision d'identification *P*, le taux de rappel *R*, le volume du modèle *AP*, la précision moyenne de détection *mAP*, et le critère F1-score.

Le calcul de la précision *P* de l'identification est indiqué dans la formule suivante:

$$P = \frac{TP}{TP + FP} \tag{III.3}$$

TP : est le nombre d'échantillons positifs avec une classification correcte.

FP : est le nombre d'échantillons faux divisé par le nombre d'échantillons positifs.

Le taux de rappel *R* représente la proportion du nombre d'échantillons positifs, la formule de calcul est indiquée par l'équation suivante:

$$R = \frac{TP}{TP + FN} \tag{III.4}$$

FN : est le nombre d'échantillons positifs classés en tant que échantillons négatifs.

AP : est un indice d'évaluation important dans l'algorithme de détection de cibles. La formule est présentée par l'équation suivante :

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) P_i (r_i + 1) \tag{III.5}$$

$$mAP = \frac{\sum_{i=1}^{n} AP_i}{n} \tag{III.6}$$

mAP: La précision moyenne de détection, représente la valeur moyenne de plusieurs catégories ou classes *i*, où *n* est le nombre de classe.

F1-score est la moyenne harmonique du rappel et de la précision, donné par la relation suivante :

$$F1 - score = 2 \times \frac{P \times R}{P + R} \tag{III.7}$$

III.5. Détection de masque

Notre deuxième contribution à portée sur l'implémentation des techniques d'intelligence artificielle sur la carte Nvidia Jetson Nano est dans la détection de masque.

L'une des utilisations les plus courantes dans le domaine de la détection facial est l'identification des masques, en particulier pendant la crise dans le monde, est l'épidémie de coronavirus, qui a causé des pertes de plus de 501.669.076 cas de coronavirus à travers le monde (6.236.201 décès à 2022). Dans ce contexte, de nombreuses réalisations qui ont été

fournis comme des solutions pour aider à atténuer les dommages de coronavirus, y compris l'identification du masque qui était nécessaire à l'époque en raison de la nature et la vitesse de la transition corona par cette note. Nous avons travaillé pour compléter un système qui nous permet de savoir si la personne porte d'abord un masque, ou même s'il le met dans le mauvais sens par les modèles YOLOv3 et YOLOv4 qui sont généralement utilisés pour identifier des objets.

III.5.1. Description des bases de données

La base des données représente l'ensemble des images qu'on va l'utiliser pour entrainer et tester nos modèles, elle contient 1697 images étiquetées des gens qui portent le masque ou non, avec 3 classes. La taille d'une seule image est de 10 Ko à 48 Ko et de dimension 416×416 pixels. La taille totale des 848 images est de 20.4 Mo, ces derniers sont étiquetées selon 3 catégories : Avec_Masque, Sans_Masque, Masque_Mal_Porté.

Les 848 images sont divisées en deux parties :

- La partie d'entrainement : contient 90% (763 images) de l'ensemble de données.
- La partie de test : contient 10% (85 images) de l'ensemble de données.

Un échantillon des données est représenté sur la Figure III.12 ci-dessous.

Figure III. 12. Un échantillon des images de la base des données.

Chaque image est associée à un fichier d'annotation de format texte (.txt) qui contient les paramètres : (c, p_x, p_y, p_h, p_w). La Figure (III.13), représente un exemple d' une image avec son fichiers text. Les differents parametres et leurs descriptions sont présentés au tableau III.1.

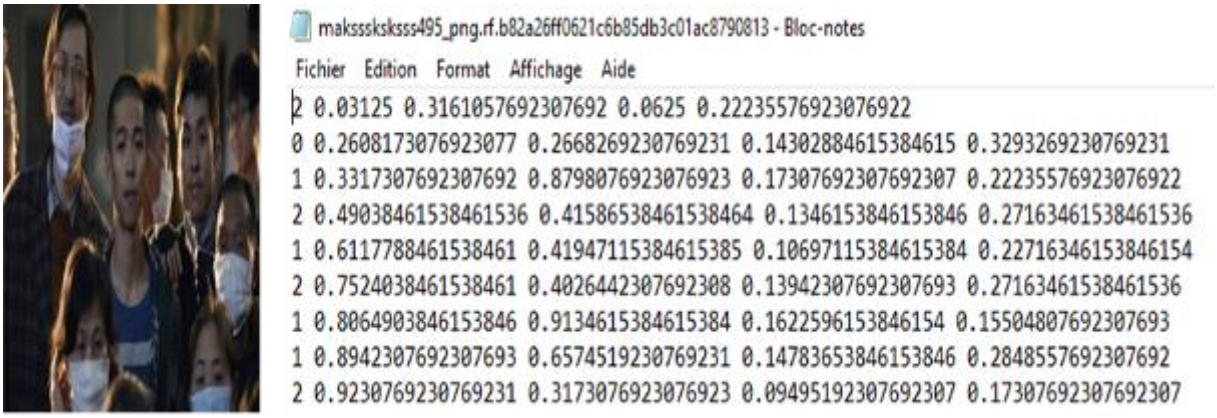


Figure III. 13. Un exemple d'image avec son fichier texte.

Tableau III. 1. La description des paramètres d'annotation d'image.

Paramètres	Description
c	Représente le type de classe qui compris entre 0 et 2 (nombre de classes 3) : 0 : Le masque est mal porté. 1 : Porté le masque. 2 : Sans masque.
p_x	La position x du centre de rectangle englobant par rapport à la cellule de grille à laquelle il est associé, elle est normalisée (divisée par la largeur de l'image).
p_y	La position y du centre de rectangle englobant par rapport à la cellule de grille à laquelle il est associé, elle est normalisée (divisée la hauteur de l'image).
p_h	Height : La hauteur du rectangle englobant normalisée, divisée par la hauteur de l'image
p_w	Width : La largeur du rectangle englobant normalisée (divisée par la largeur de l'image)

III.5.2. Description de processus de développement des modèles YOLO pour la détection de masque

YOLO est un modèle qui détecte les objets, il est déjà entrainé sur 80 classes, ce qui nous intéresse c'est de prédire si la personne porte ou non un masque, autrement dit notre objectif est de former un modèle d'apprentissage en profondeur personnalisé qui sera effectuée uniquement sur la prédiction de trois classes : ''Avec_Masque, Sans_Masque, Masque_Mal_Porté''. Après avoir obtenir les meilleurs résultats de prédiction, on passe à l'étape de tester le modèle.

Dans notre projet, on a travaillé sur deux modèles séparément : le modèle YOLOv3-tiny et le modèle YOLOv4-tiny. Le schéma (Figure III.14) représente la procédure de notre projet.

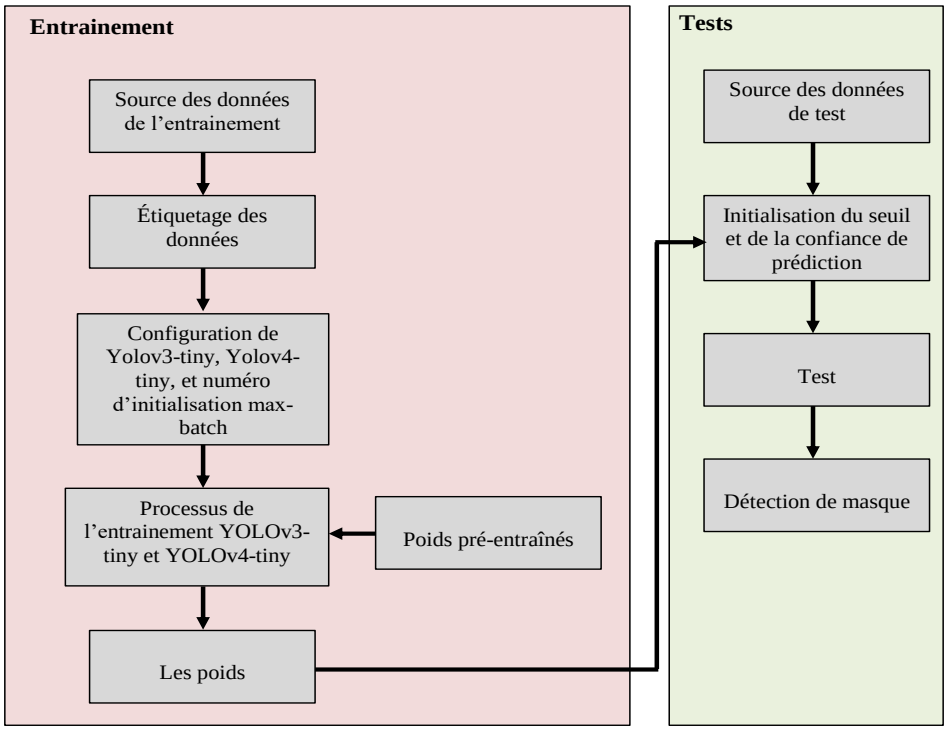


Figure III. 14. La procédure de détection de masque.

Le travail en général est divisé en trois étapes qui sont :

1. Préparation de l'environnement.
2. L'entrainement des modèles.
3. La discussion des résultats du test des modèles pré-entraînés.

Les étapes en détails pour construire des modèles YOLOv3 ou YOLOv4 :

1) Préparation d'environnement : Nous avons importé toutes les bibliothèques nécessaires.

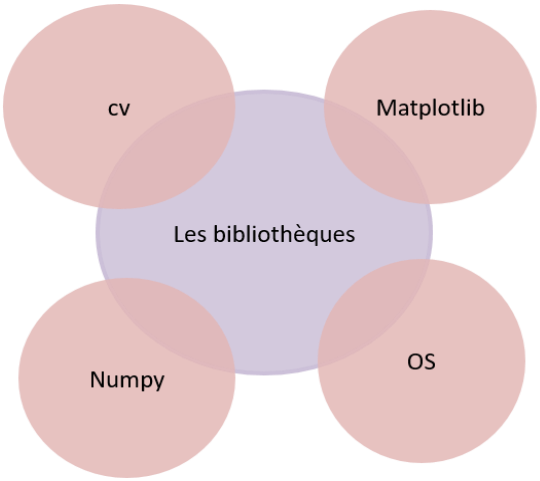


Figure III. 15. Bibliothèques.

2) Plateforme Darknetet: Télécharger les modèles pré-entrainés YOLOv3 et YOLOv4 pour détecter les objets en utilisant la commande :

!git clone https://github.com/AlexyAB/darknet.git

Dans cette étape, Darknet s'est téléchargé dans un fichier zip qui contient le framework d'apprentissage profond écrit en C qui facilite le travail et contient les codes python dont on a besoin.

Afin de pouvoir utiliser le Framework Darknet, il doit être compilé après la modification de son fichier Makefile pour lui permettre de prendre en charge les ressources nécessaires (GPU, OpenCV, …). La Figure III.16, montre la configuration qu'on a attribuée à Darknet avant sa compilation.

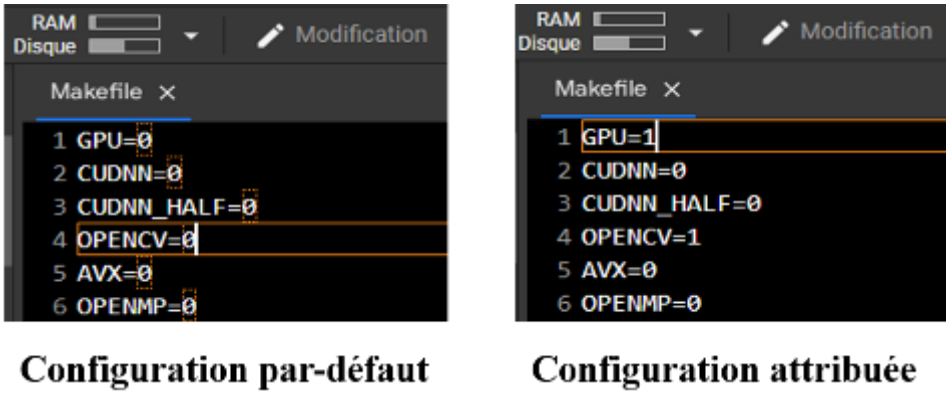


Figure III. 16. La configuration d'environnement Darknet.

Au lieu d'apprendre à partir de zéro, nous utilisons un modèle pré-entraîné qui contient des poids convolutifs entraînés sur ImageNet. En utilisant ces poids comme poids de départ, notre réseau peut apprendre plus rapidement. Nous allons maintenant le télécharger dans notre dossier darknet.

Dans le cadre de notre projet, on a travaillé avec les deux modèles YOLOv3-tiny et YOLOv4-tiny qui sont téléchargeable sur Google Colab par les commandes suivantes respectivement :

YOLOv3-tiny : ***!gdown --id 18v36esoXCh-PsOKwyP2GWrpYDptDY8Zf***

YOLOv4-tiny : ***git clone https://github.com/AlexeyAB/darknet***

3) Configurations des fichiers : Le framework Darknet s'exécute en utilisant des configurations simples par fichiers. Une fois que les fichiers d'étiquettes et les fichiers d'images sont préparés, nous devons créer deux fichiers texte contenant les liens des fichiers d'images d'entrainement et de test respectivement.

4) Etiquetage des classes : Les trois classes à prédire sont nommées dans un fichier .names, voir la Figure III.17.

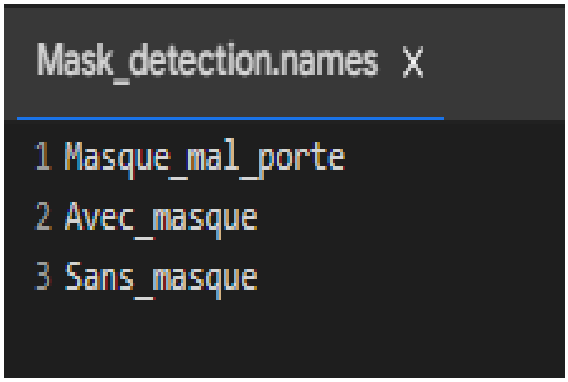


Figure III. 17. Les noms des classes.

5) Nouveau modèle entraîne : Le dossier backup c'est le répertoire des nouveaux poids ''weights'' obtenues aprés avoir entrainner le modèle.

6) Crée un fichier obj.data et modifier les paramétres : Ce fichier est très important dans le processus de développement de nos modèles, car il contient les liens des fichiers de configurations créés précédemment. La Figure III.18, montre le fichier obj.data.

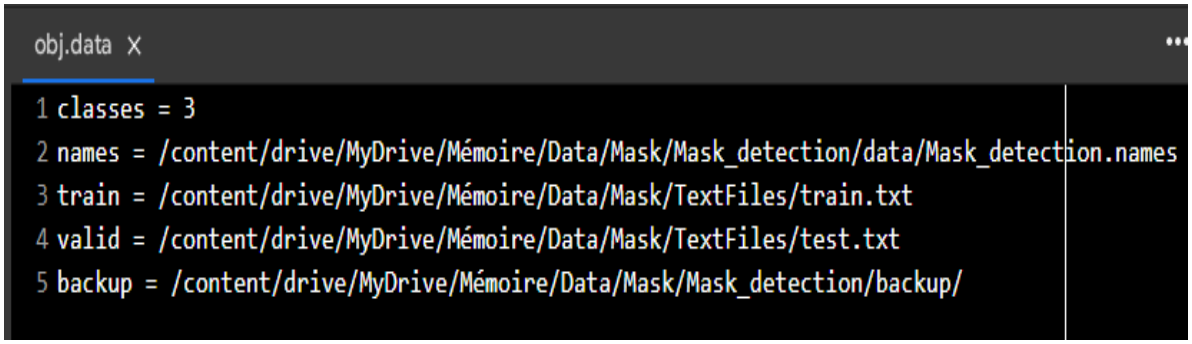


Figure III. 18. Fichiers obj.data après la configuration.

7) Paramètres : Des fichiers de configuration du réseau YOLOv3-tiny et YOLOv4-tiny avec les modifications nécessaires des paramètres suivants:

Cette section montre la configuration de YOLOv3-tiny pour 3 classes, nous apporterons plusieurs modifications au fichier dont nous avons besoin pour changer la configuration : ''batch'' en 64, cela signifie que nous utiliserons 64 images pour chaque étape de découpage, puis nous devons changer les subdivisions, ce que signifie que le lot sera divisé par 16 pour réduire les exigences du GPU, puis nous devons changer le max batch a au moins 2×nombre de classes=6000.

Le nombre de filtres est bien défini par l'équation suivante :

Filtres=(nombre de classe+5)×3

Nous devons également modifier leur nombre de classes qui représente le nombre de catégories que nous voulons détecter (Classes=3), ainsi le nombre de steps qui est :

Steps= (max-batch×80%)/100% ; (max-batch×90%)/100%.

Max-batch=6000

Steps=4800 ,5400

Classes=3

Filtres=24

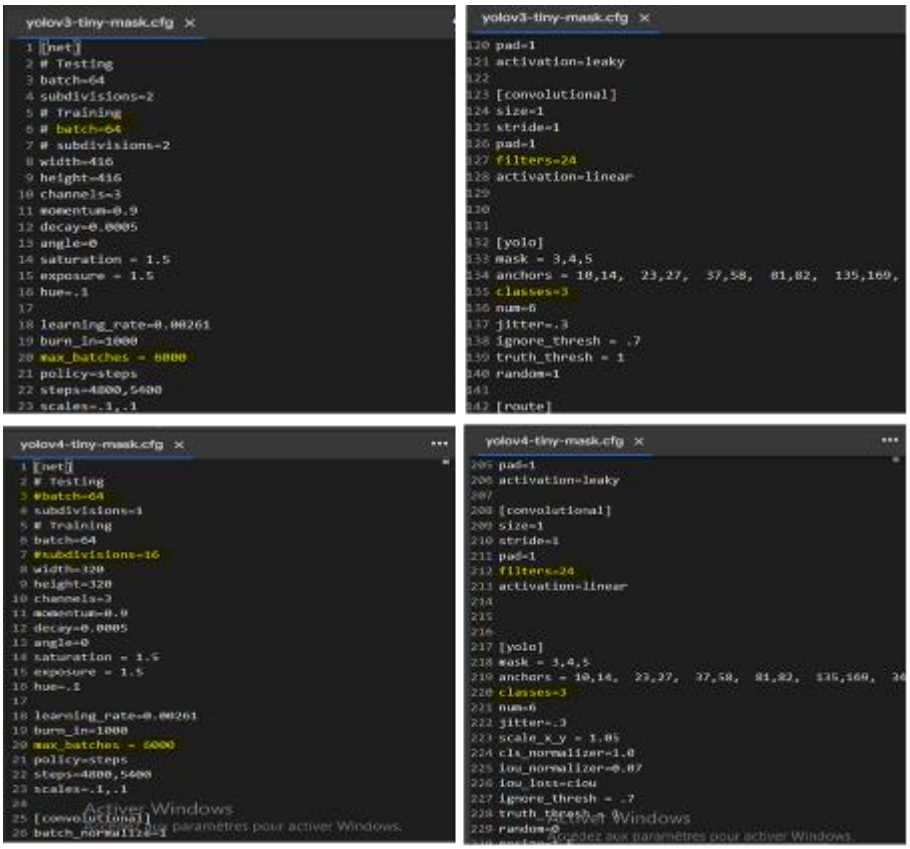


Figure III. 19. La configuration des fichiers.cfg YOLOv3-tiny et YOLOv4-tiny.

III.5.3. Entrainement

III.5.3.1. Entrainement de modèle YOLOv3-tiny

Lorsque nous avons entrainé notre propre détecteur de masque, nous avons exploité les modèles existants entrainés sur de très grands ensembles de données, même si ces derniers ne contiennent pas l'objet que nous essayons de détecter (le masque), il suffit de modifier les nouveaux poids de modèle de base YOLOv3-tiny et c'est exactement ce que nous avons fait. Nous avons évalué la performance de ce modèle sur les 3 classes, en changeant des paramètres (nombre de classe, nombre de filtre de la dernière couche, max batch et steps), au début nous avons utilisé un Max batch égale à 200, après nous avons essayé avec un nombre de max batch égale à 6000, c'est important à noter que on effectuer cette étape par l'activation de GPU. La Figure III.20, représente l'exécution de la procédure d'entraînement.

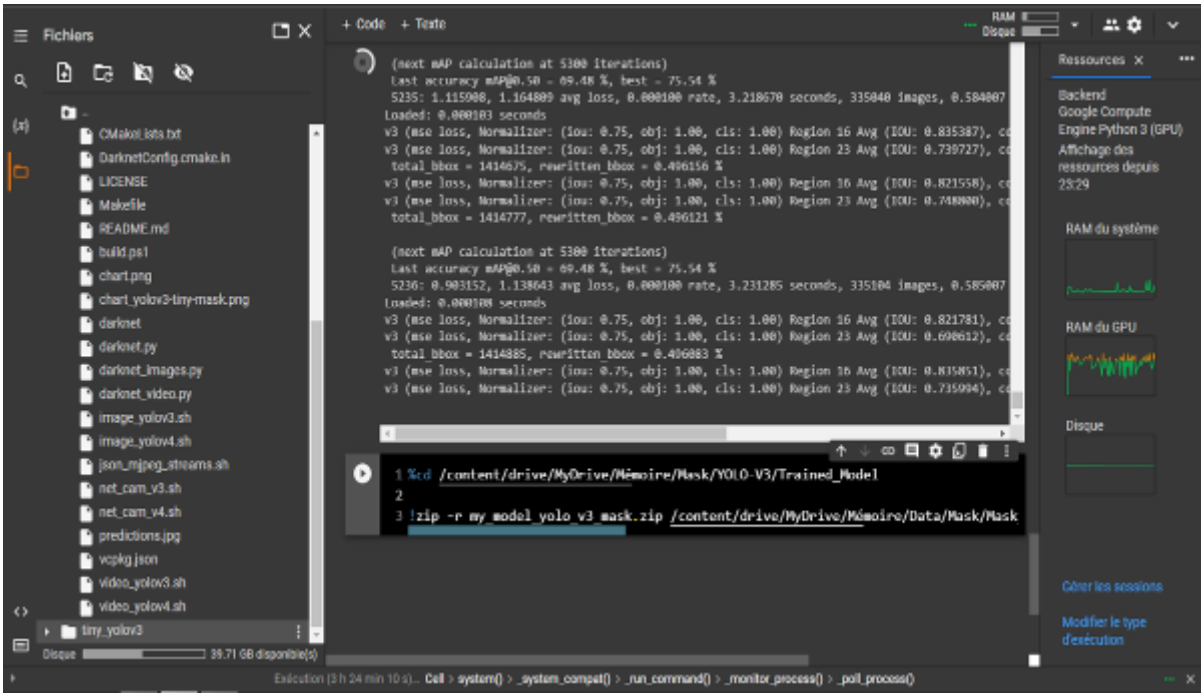


Figure III. 20. L'étape d'apprentissage de modèle YOLOv3-tiny.

III.5.3.2. Entrainement de modèle YOLOv4-tiny

Maintenant nous avons passé a l'entrainement de modèle YOLOv4-tiny, nous avons entrainné ce dernier sur 90% de la totalité de la base des données (763 images), exactement comme le modèle YOLOv3-tiny, en suivant les memes étapes de configuration pour 6000 iterations comme indique la Figure III.21.

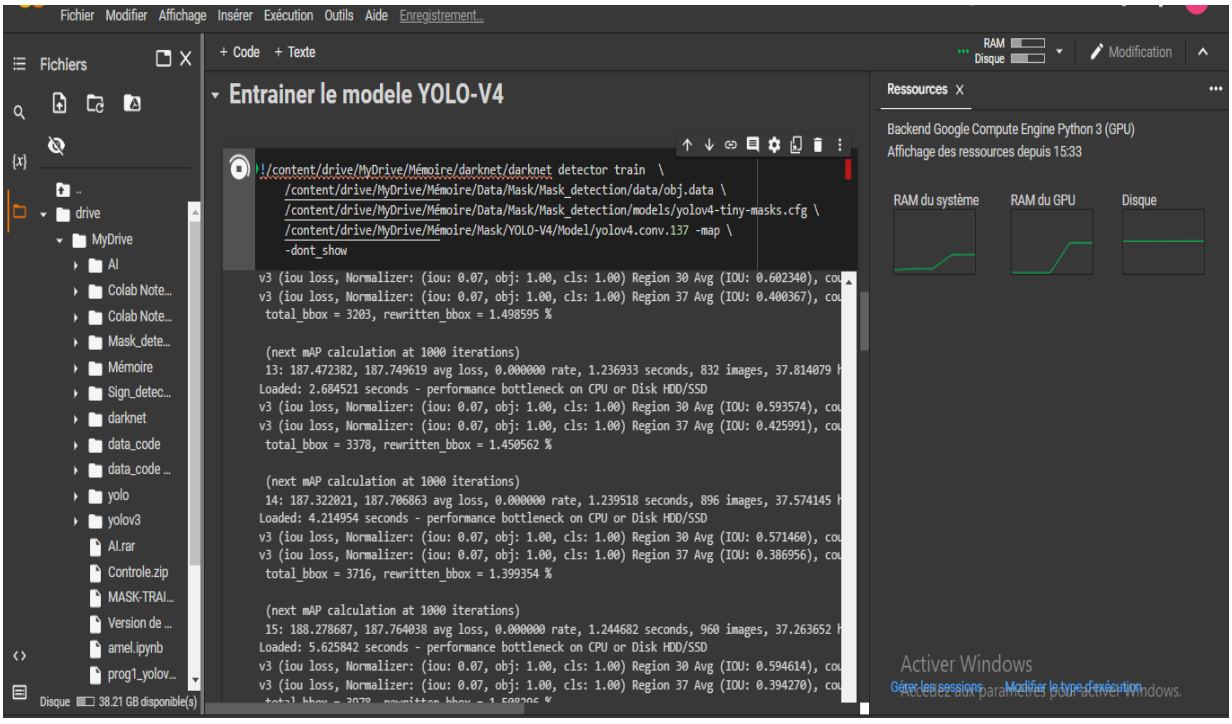


Figure III. 21. L'étape d'apprentissage de modèle YOLOv4-tiny.

III.5.3.3. Résultats et discussions

L'apprentissage de modèle YOLOv3-tiny et YOLOv4-tiny sur un nombre de Max-batch égale a 200 n'a rien affiché comme résultat et aprés le test il n'a rien détecté, par contre l'entrainemment de deux modéles séparement sur la meme base des donnes et pour un max de batch égale a 6000 a donné les meilleurs résultats représentés sur les Figures III.22 et III.23.

L'erreur de classification (Loss) est présentée en bleu (calculée sur le jeu d'entraînement) et la courbe de test utilisant le critère mAP (Mean Average Precision) est présentée en rouge. Elle montre une convergence stable qui sature rapidement au bout de quelques milliers d'itérations. En pratique, l'entraînement peut être stoppé dès que la métrique d'évaluation n'augmente plus de façon significative, d'une part la précision de modèle YOLOv3-tiny a pris une valeur stable de 70.7% autour de 4800 itérations, d'autre part nous constatons une précision de 78.4% autour de 1260 itérations pour le modèle YOLOv4-tiny. La durée de l'entraînement varie en fonction de la profondeur du réseau. Elle varie de 3 heures et 24 minutes pour YOLOv3-tiny à 5 heures et 49 minutes pour YOLOv4-tiny.

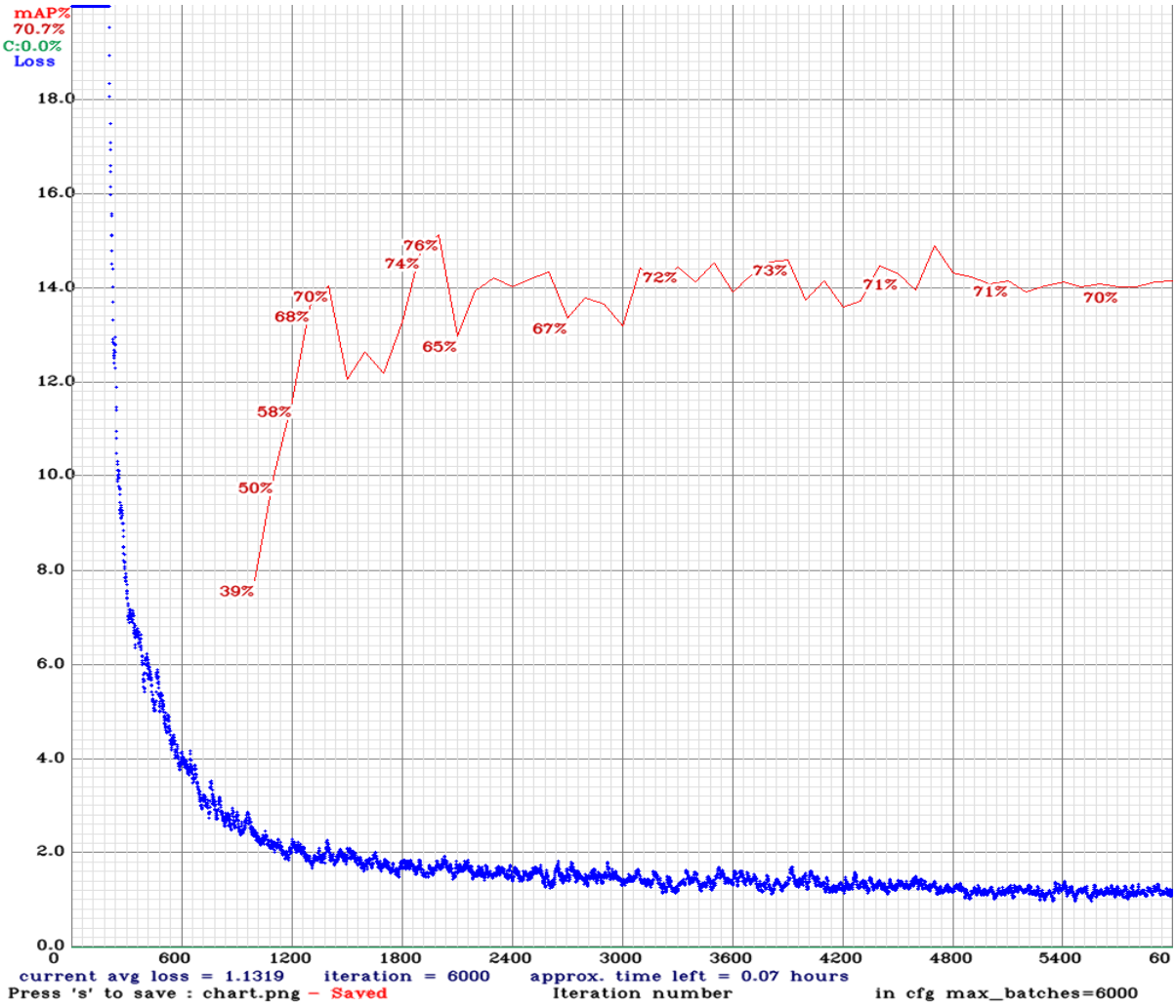


Figure III. 22. La courbe d'apprentissage de YOLOv3-tiny.

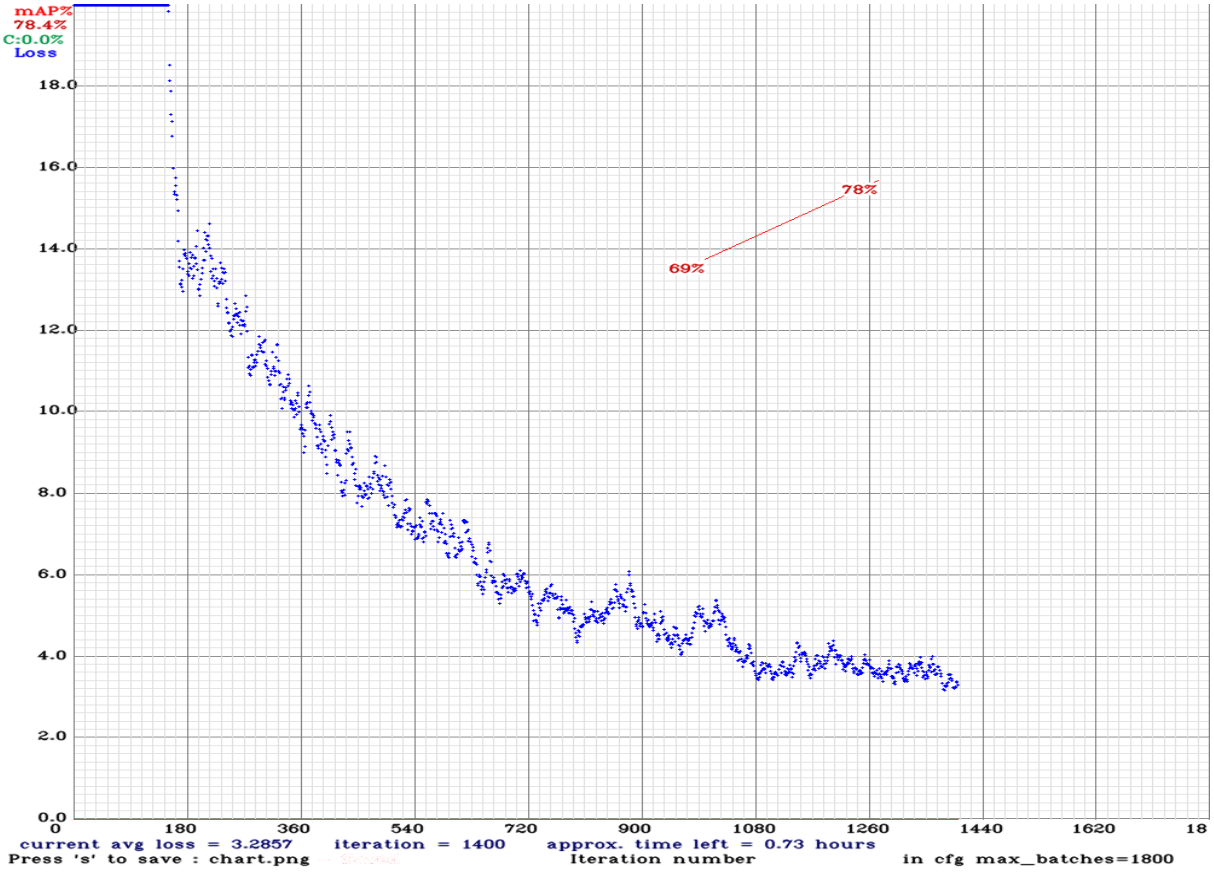


Figure III. 23. La courbe d'apprentissage de modèle YOLOv4-tiny.

Les deux modèles YOLOv3-tiny et YOLOv4-tiny ont pu atteindre des précisions globales de 70.7% et 78.4%, respectivement.

Tableau III. 2. Les métriques d'évaluation des modèles YOLOv3-tiny et YOLOv4-tiny pour la détection de masque.

Modèle	La précision pour l'ensemble de test (mAP)	Le temps écoulé pour l'entrainnement	Le nom de classe	ID de classe	AP	TP	FP	P
YOLOv3 -tiny	70.7%	3h 24 min 10 s	Masque mal porté	0	70.06	36	9	**0.80**
			Avec masque	1	45.52	2	3	**0.40**
			Sans masque	2	87.08	241	31	0.89
YOLOv4 -tiny	**78.4%**	5h 49 min	Masque mal porté	0	**89.27**	43	14	0.75
			Avec masque	1	40.70	2	3	**0.40**
			Sans masque	2	**96.00**	266	13	**0.95**

Le tableau III.3, illustre les différentes métriques d'évaluation en moyenne (P : précision, R : recall, F1-score, IoU) pour chaque modèle. Il est clair que pour YOLOv4-tiny le modèle donne une précision maximale.

Tableau III. 3. Les métriques d'évaluation en moyenne des modèles YOLOv3-tiny et YOLOv4-tiny pour la détection de masque.

Modèle	**YOLOv3-tiny**	**YOLOv4-tiny**
P	0.87	**0.91**
R	0.82	**0.91**
F1-score	0.84	**0.91**
IoU	68.63	**73.54**

Après l'analyse des valeurs obtenues dans les tableaux (III.2, III.3), nous avons constaté que YOLOv4-tiny donne des meuliers résultats que ceux de YOLOv3-tiny, pas seulement au niveau de paramètres d'évaluation (AP: Averege precision, TP: True Positive, FP : False Positive) mais plus précisément au terme de précision, où le réseau YOLOv4-tiny a pu atteindre une précision moyenne 91% supérieure à celle de YOLOv3-tiny 87%. Par contre YOLOv3-tiny a pris moins de temps pour l'entrainement ce qui fait de lui un modèle rapide.

Dans la suite, nous présentons les résultats sur les échantillons de tests des images qui représentent 10% (85 images) de base de données, nous avons choisi de présenter les résultats sur les Figures III.24 et III.25 qui illustre quelques séquences d'image et nous précisons uniquement celles qui contiennent les 3 catégories de classes. Nous avons fait exprès de choisir quelques images du test en commun pour les deux modèles, pour facilite la comparaison de leurs performances.

Figure III. 24. Echantillons d'images testés par le modèle YOLOv3-tiny.

Figure III. 25. Echantillon d'images testé par YOLOv4-tiny.

Nous avons testé les deux modèles YOLOv3-tiny et YOLOv4-tiny sur un ensemble d'images diverses. Dans lesquelles il y a une variété de positions de masque, ou il y a des images simples pour détecter le masque sur eux, et il y a des cas où la position du masque n'est pas très claire. Surtout pour le cas où le masque est mal porté. Dans cette phase nous avons travaillé seulement par le CPU, qui montre la faisabilité de combiner un détecteur rapide de type YOLO avec une unité de traitement graphique. Afin de traiter les opérations complexes dans un temps record pour la phase d'apprentissage.

Dans le premier groupe de résultat le modèle YOLOv3-tiny a montré des bons résultats, mais notamment la défaillance de ce dernier de détecter les cas ou le masque est mal porté, ou les cas ou la personne est un peu loin, par contre le modèle YOLOv4-tiny répond au mieux pour les cas difficiles que nous avons choisis (des personnes loin, image un peu flou, etc…).

D'après la comparaison des deux modèles YOLOv3-tiny et YOLOv4-tiny avec l'utilisation des critères numériques et des remarques subjectives, la méthode YOLOv4-tiny permet d'obtenir de meilleurs résultats que ceux de la méthode YOLOv3-tiny.

III.6. Systèmes de sécurité routière

Notre troisième contribution a portée sur l'implémentation des techniques d'intelligence artificielle sur la carte NVIDIA Jetson Nano dans les systèmes de transport intelligents (STI). Dans tous les pays, les panneaux de signalisation contiennent des informations essentielles pour les conducteurs sur la route, notamment la limitation de vitesse, l'indication de la

direction, l'information sur les arrêts, etc. Les systèmes de reconnaissance des panneaux de signalisation sont cruciaux pour de nombreuses applications dans le monde réel, telles que la conduite autonome, la surveillance du trafic, la protection et l'assistance au conducteur, le maintien du réseau routier et l'étude des perturbations du trafic. Deux sujets connexes importants pour la STI sont la détection et reconnaissance des panneaux de signalisation. La détection des panneaux de signalisation affecte directement la sécurité des conducteurs, en raison de leur ignorance, peut facilement causer des dommages. Les systèmes automatiques qui assistent les conducteurs peuvent améliorer les comportements dangereux au volant en se basant sur la détection et la reconnaissance des panneaux.

III.6.1. Description de base de données

La base des données de la reconnaissance des panneux de signalisation est composée de 699 images, qui sont divisées en 588 images (84%) pour l'entrainnement des modèles et 111 images (16%) pour le test. Chaque image a une dimension de 1360*800 pixel et une taille entre 212 a 516 octets. Les images sont divisées en quatre catégories qui conviennent aux propriétés de différents modèles de détection YOLOv3-tiny et YOLOv4-tiny avec des propriétés différentes. Un échantillon d'images d'entrainement et de test est représenté dans la Figure III.26.

Figure III. 26. Un Echantillon des images de la base des données pour la détection et la reconnaissance des panneaux routières.

Chaque image est associée à un fichier texte qui contient ces informations. A noter que nous avons travailler avec quatre catégories: panneaux de signalisation de danger, panneaux de signalisation d'obligation, panneaux de signalisation d'interdite, le reste des classes sont étiquettées par ''other''. La Figure III.27 ci-dessous montre un exemple d'une image de panneau de signalisation avec son fichier texte:

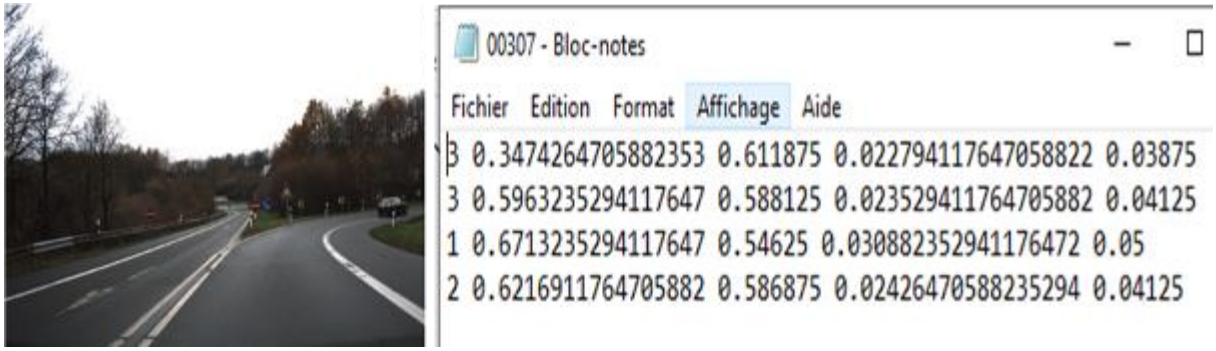


Figure III. 27. Un exemple d'une image de détection des panneaux routière avec son fichier texte.

III.6.2. Description de processus de développement des modèles YOLO

Dans cette section nous avons evalué la performance des modeles YOLOv3-tiny et YOLOv4-tiny pour la reconnaissance des panneaux routiers, les deux modèles sont déjà entrainnés pour la détection de 80 types d'objets, donc l'objectif de notre travail est de personaliser la reconnaissance des deux modèles. Le concept est le même que celle de la detection de masque, la seule différence est le jeu des données utilisé et le nombre des catégories sur lesquelles nous devont travaille.

Le schéma représenté sur la Figure III.28, montre les procédures principales de la détection des panneaux routières dans notre travail.

La préparation d'environnement que ce soit dans la détection de masque ou la détection des panneux routiers est la même, elle se base essentiellement sur le téléchargement des bibliothèques nécessaires que nous avons cité dans la partie consacrée pour le développement des modèles de masque et la configuration d'environnement Darknet, en activant le GPU et OpenCV ansi que la séparation des données d'entrainnement et de test.

Nous avons déjà détaillé toutes ces étapes dans la section de détection de masque, donc nous concentrons seulement sur le fichier qui contient les noms des classes et la configuration des modèles des modèles YOLOv3-tiny et YOLOv4-tiny. La Figure III.29, illustre les catégories à détecter.

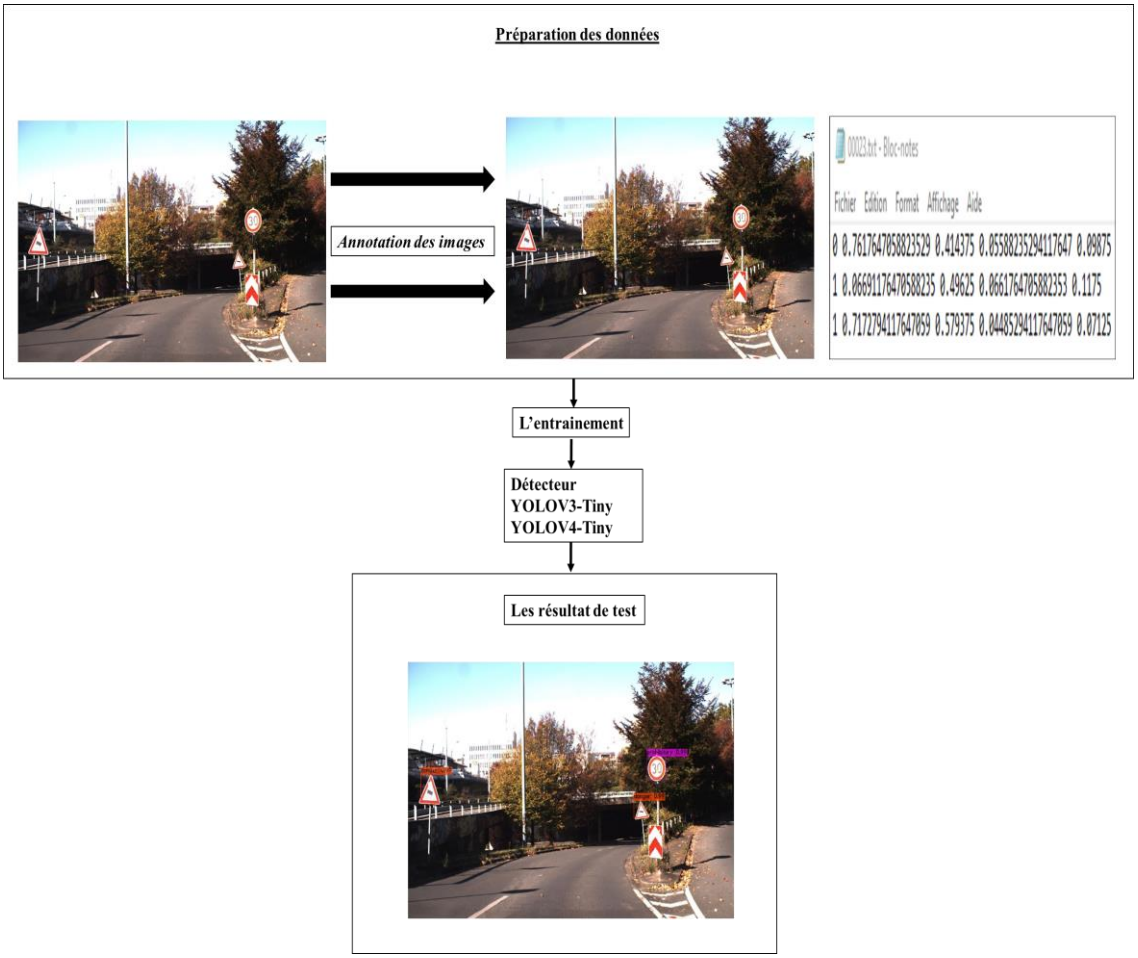


Figure III. 28. La procédure de développement de modèle de détection des panneaux routière.

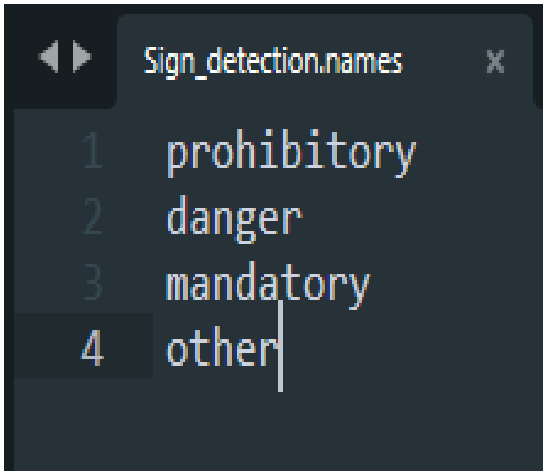


Figure III. 29. Les classes de détection des panneaux routière.

La Figure III.30, représente la configuration qu'on a attribué aux modèles (YOLOv3-tiny et YOLOv4-tiny), en changeant les valeurs de quelques paramètres de leurs fichiers de

configuration (yolov3_traffic_signs.cfg et yolov4_traffic_signs.cfg), respectivement. Dans cette partie nous citons les paramétres essentiels qu'on a changé :

Max-batch=8000

Steps=7600 ,8400

Classes=4

Filtres=24

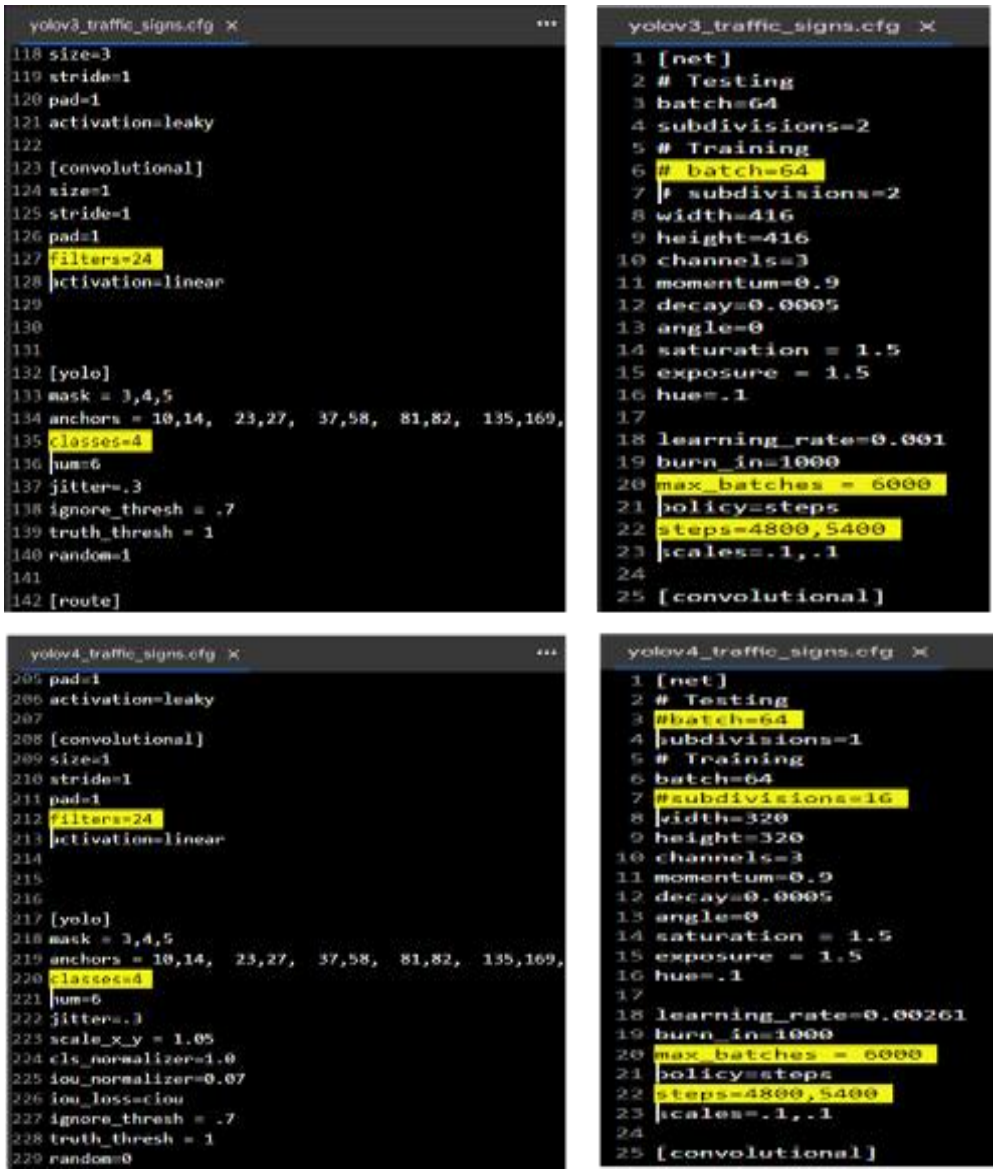


Figure III. 30. Configuration des fichiers. cnf pour YOLOv3-tiny et YOLOv4-tiny.

III.6.3. Entrainement

III.6.3.1. Entrainement de modèle YOLOv3-tiny

Après toutes les configurations des fichiers que nous avons appliqués. Maintenant nous pouvons déclencher l'entrainement de notre modèle YOLOv3-tiny on exécutant la commande d'apprentissage. La Figure III.31, montre l'architecture de modèle YOLOv3-tiny pendant

l'apprentissage. Une fois lancée le traitement peut durer des heures, selon le nombre de classes et de fichier d'entraînement, dans notre cas nous avons travaillé sur 4 catégories des panneaux routiers ‘‘prohibitory, mandatory, danger, other’’. Nous avons effectué la phase d'apprentissage pour plusieurs cas, en changeant le nombre d'itérations à chaque fois, nous avons commencé notre traitement par un nombre de max-batch minimal égale à 1700 puis nous avons augmenté le nombre a 6000 afin d'avoir une meilleur précision.

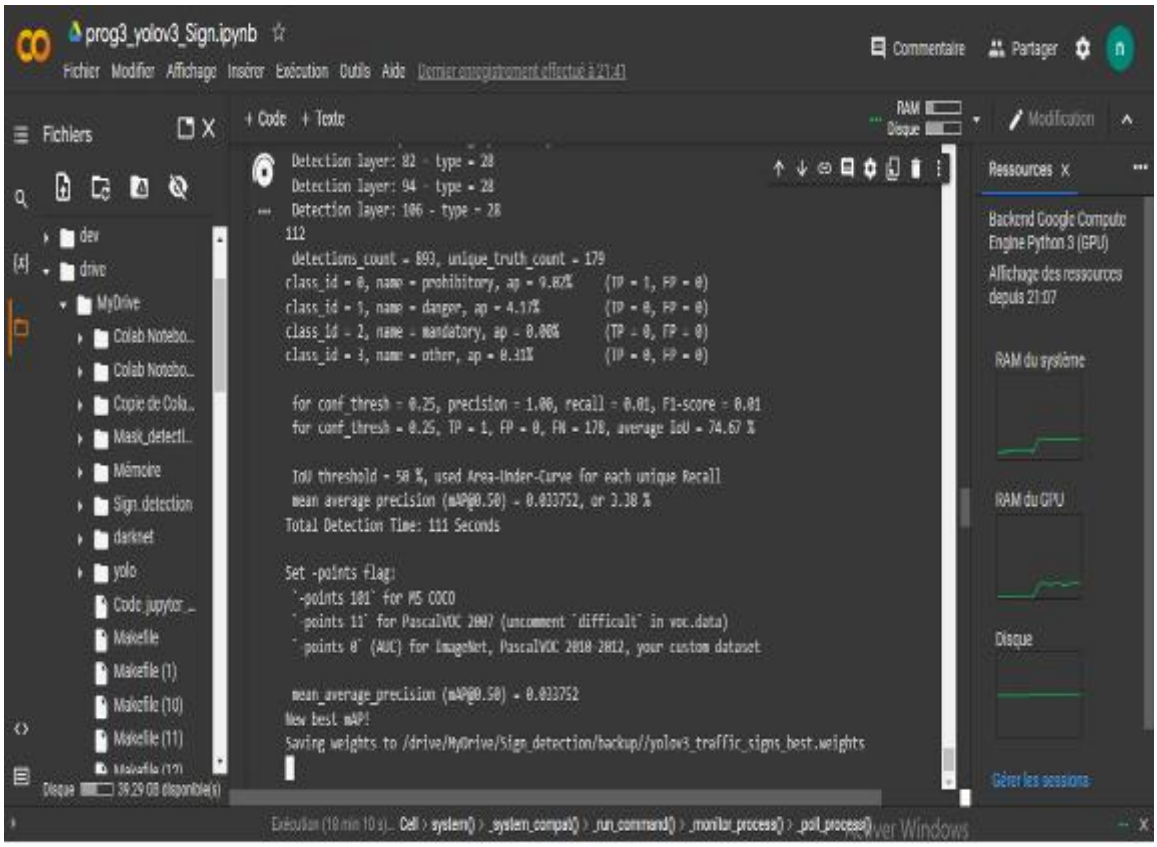


Figure III. 31. L'apprentissage de modèle YOLOv3-tiny.

III.6.3.2. Entrainement de modèle YOLOv4-tiny

Apres avoir effecter les mofdifications nessaiceres sur le fichier cfg de configuration de modèle YOLOv4-tiny, nous avons entrainné le système pour 1700 batch, puis pour 6000 batch. La Figure III.32, représente le processus d'apprentissage a la présence de GPU, qui economise le temps et facilite la phase d'entrainnement.

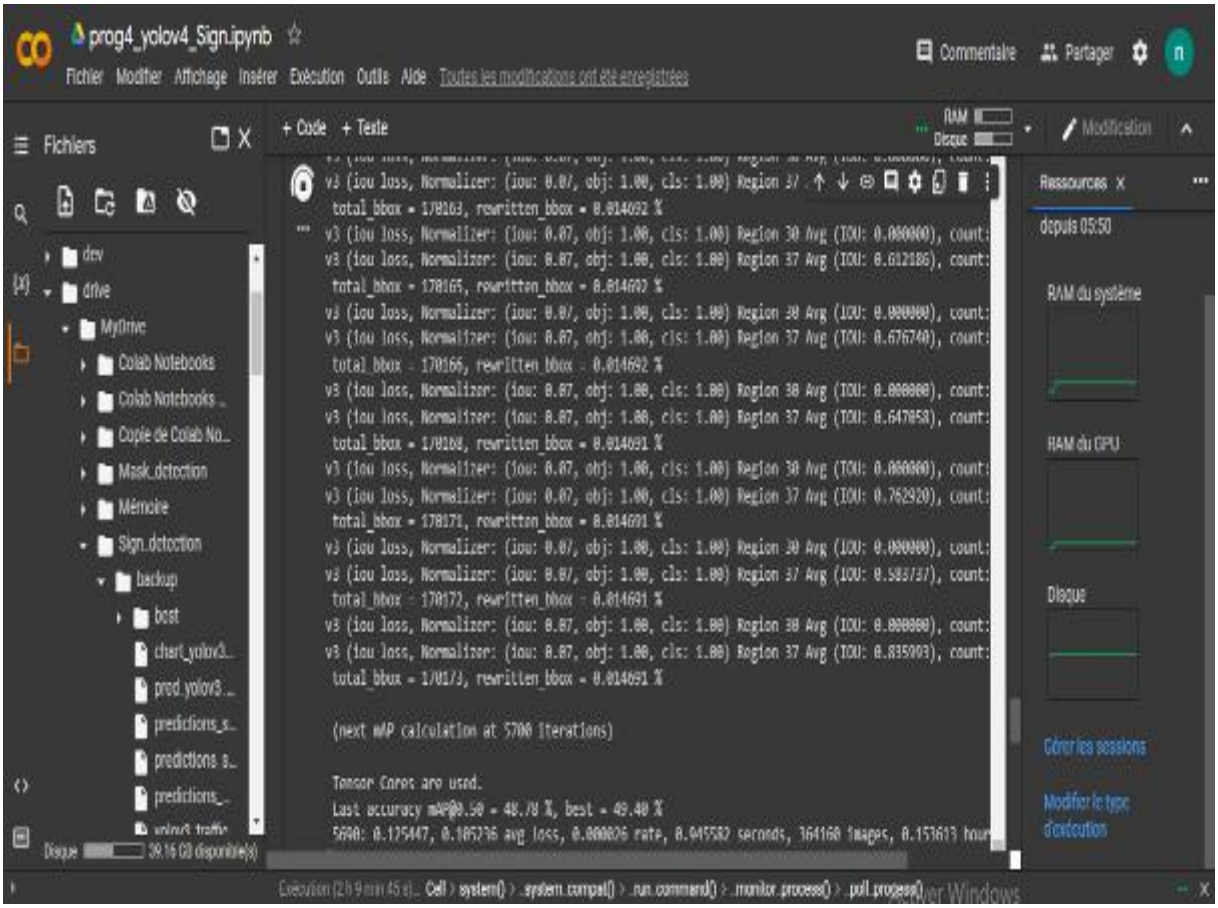


Figure III. 32. L'apprentissage de modèle YOLOv4-tiny.

A noter que Darknet effectue des sauvegardes régulièrement, en cas ou le temps de GPU est achevé et ceci pour modifier ces paramètres de backup automatiquement. Darknet propose aussi des fichiers de poids dont celui ayant le meilleur score (document_yolov4_best.weight) c'est le fichier qui contient les meilleurs paramètres possible qui permettre à nos modèles d'êtres fiables et performants.

III.6.3.3. Résultats et discussions

Dans cette partie, nous avons discuté l'entrainement réalisé pour les méthodes d'apprentissage profond de famille YOLO (les versions 3 et 4) sur la même base de données des panneaux de signalisation introduites précédemment. A noter que Les bases de données sont systématiquement divisées aléatoirement en un jeu d'entraînement comportant 80 % des données et un jeu de validation comportant les 20 % restant. Les données de validation servent à la recherche des hyper-paramètres des réseaux et pour l'évaluation des performances. Cette séparation en deux jeux de données, l'un pour l'entraînement et l'autre pour la validation, permet d'éviter le risque de sur-apprentissage (Over-fitting) des modèles d'apprentissage.

Les Figures III.33, et III.34 illustrent les meilleurs résultats d'apprentissage des modèles YOLOv3-tiny et YOLOv4-tiny pour un nombre d'itération de 6000 et 1700 respectivement, dont la courbe d'apprentissage utilisant le critère de *mAP* (mean averege precision) présentée en rouge et la fonction d'erreur en bleu. Tout d'abord nous observons que la courbe d'erreurs se diminue en fonction de nombre d'itération jusqu'à ce qu'elle atteint son minimum à la fin d'apprentissage, tandis que la précision d'apprentissage montre une évaluation progressive en fonction de nombre d'itération. Le modèle YOLOv3-tiny a pris une précision stable relativement 80% autour de 4200 batches, d'autre part le modèle YOLOv4-tiny a rien détecté au début jusqu'à 990 batches où il a commencé à apprendre pour atteindre une précision de 94% au bout de 1620 batches.

La durée de l'entraînement varie en fonction de la profondeur du modèle, elle varie de 3 heures pour YOLOv3-tiny à 2 heures pour YOLOv4-tiny.

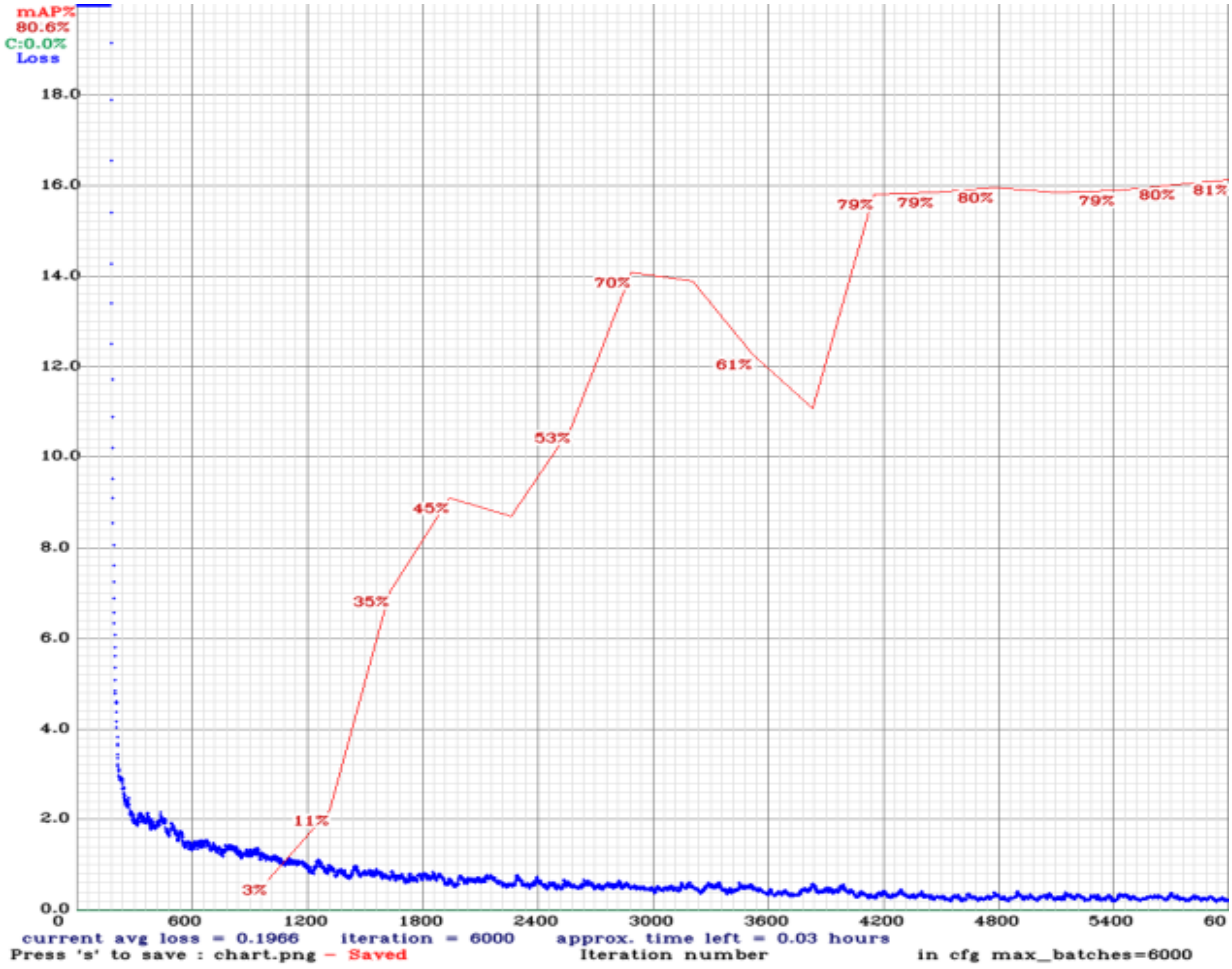


Figure III. 33. La courbe d'entrainement de YOLOv3-tiny a un max batch égale à 6000 batch.

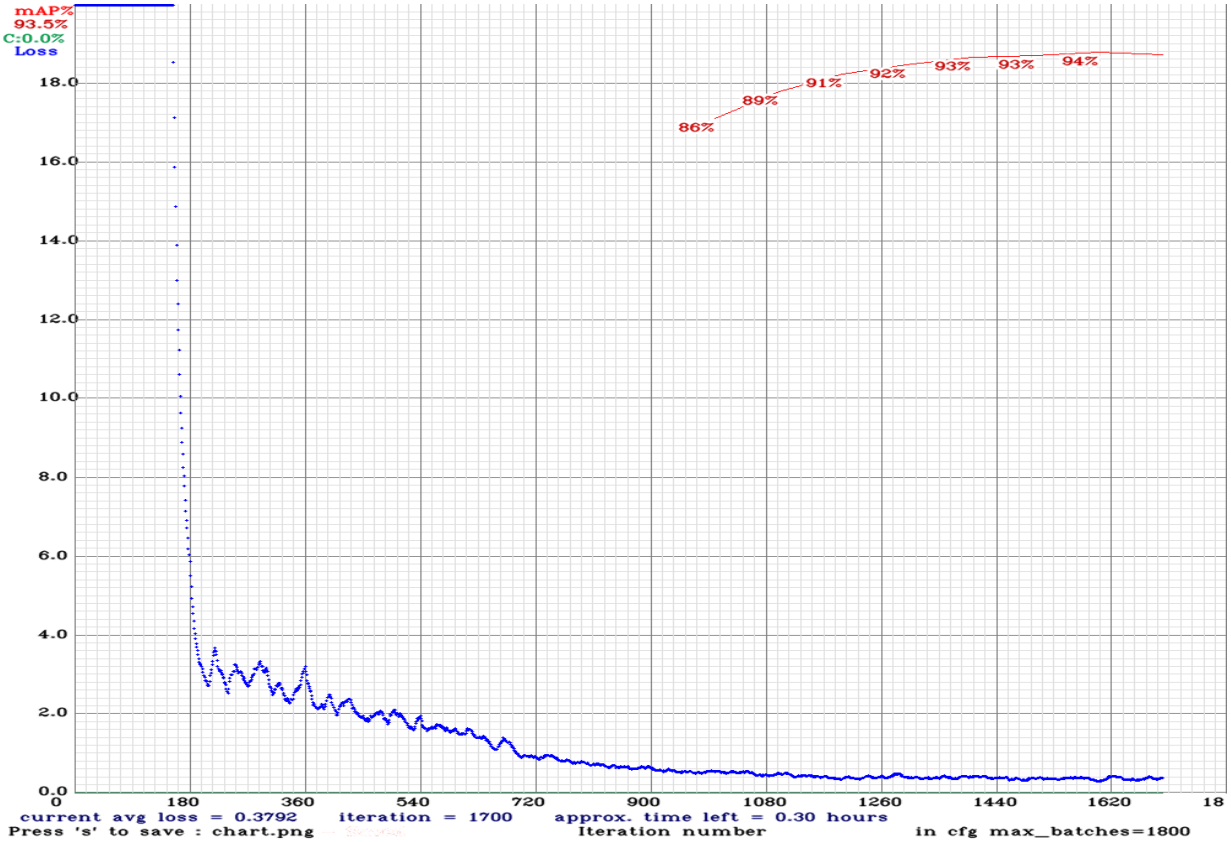


Figure III. 34. La courbe d'entrainement de YOLOv4-tiny a un max batch égale à 1700 batch.

Tableau III. 4. Les métriques d'évaluation des modèles YOLOv3-tiny et YOLOv4-tiny pour la détection et la reconnaissance des panneaux routiers.

Modèle	La précision pour l'ensemble de test (mAP)	Le temps écoulé pour l'entrainnement	Le nom de classe	ID de classe	AP	TP	FP	P
YOLOv3-tiny	80.6%	3h 00 min	Prohibitory	0	93.08	62	5	0.92
			Danger	1	**99.77**	36	6	0.86
			Mandatory	2	76.14	17	4	0.81
			Other	3	53.36	20	2	0.91
YOLOv4-tiny	93.5%	2h 00 min	Prohibitory	0	**96.87**	66	11	0.86
			Danger	1	99.49	35	3	**0.92**
			Mandatory	2	**89.61**	18	1	**0.95**
			Other	3	**88.25**	40	7	0.85

Tableau III. 5. Les métriques d'évaluation en moyenne des modèles YOLOv3-tiny et YOLOv4-tiny pour la détection et la reconnaissance des panneaux routiers.

Modèle	**YOLOv3-tiny**	**YOLOv4-tiny**
P	**0.89**	0.88
R	0.75	**0.89**
F1-score	0.82	**0.88**
IoU	71.80	**76.32**

Nous constatons que les valeurs de paramètres d'évaluation (P : Précision, R: Recall, F1-score: mesure F, IoU: Intersection sur Union), montrés dans les tableaux III.4 et III.5, sont assez similaires. YOLOv3-tiny et YOLOv4-tiny ont présenté les meilleurs résultats avec un taux de précision : 89% et 88%, respectivement. Concernant le temps d'exécution cette expérience a montré que pour un nombre d'itération égale à 6000 YOLOv3-tiny a fait 3 heures, tandis que YOLOv4-tiny a pris 2 heures pour un max batch de 1800.

Figure III. 35. Les résultats des images testées par YOLOv3-tiny.

Figure III. 36. Les résultats des images testées par YOLOv3-tiny.

Les Figures III.35 et III.36, illustrent des prédictions réussies de panneaux de signalisation, sur un échantillon des images de test diverse, YOLOv3-tiny et YOLOv4-tiny ont montré leurs capacité de détecter les panneaux de signalisation et les classifier selon 4 classes: ''Prohibitory, Danger, Mandatory, Other'', précisément les modèles ont reconnu les panneaux de signalisation même dans des cas difficile où les images ne sont pas claires à cause de l'effet d'éblouissement de la lumière, et les effets de l'ombre. Les captures que nous présentons ne sont pas les seules situations rencontrées.

L'entraînement de chaque modèle (YOLOv3-tiny, YOLOv4-tiny) sur deux bases des données différentes (Détection de masque, reconnaissance des panneaux de signalisation routière) ont mené à la constatation suivante :

YOLOv4-tiny est le modèle le plus performant et qu'il a donné une précision élevée avec les deux bases des données ce qui prouve ca capacité de généralisation. Tandis que YOLOv3-tiny a montré un comportement différent au niveau de précision, qu'était proche à celle de YOLOv4-tiny avec la base de données de panneaux de signalisation et inferieure avec la base de données de détection de masque.

III.7. Conclusion

Ce chapitre avait pour objectif de déterminer les meilleurs modèles qu'on pourra utiliser ultérieurement pour trois applications en temps réel, qui sont: la reconnaissance faciale, la détection de masque et la reconnaissance des panneaux de signalisation routiers.

Au premiers temps nous avons constaté la capacité de l'algorithme proposé par viola and Jones de détecter efficacement les visages. Tout d'abord, sa particularité repose sur le fait que les caractéristiques utilisées sont simple mais nombreuse ainsi la contribution et l'introduction des images intégrales qui permettent un calcul rapide de ces caractéristiques. Puis, nous avons présenté la partie implémentation de notre approche, d'où nous avons expliqué en détail les étapes de prétraitement et de préparation d'environnement. Enfin, nous avons discuté les résultats obtenus après l'entraînement de chaque modèle en se basant sur des considérations de vitesse d'exécution et de précision des prédictions.

D'après les résultats obtenus nous avons conclu que:

- L'apprentissage profond de détection d'objets peuvent être employés pour réaliser de la détection de points d'intérêt et permettent d'obtenir de bonnes performances.
- Un réseau plus profond est plus précis mais plus lent et un meilleur réseau est caractérisé par une meilleure précision et une bonne capacité de généralisation.
- La quantité et la qualité des données est aussi importante pour que les algorithmes d'apprentissage profond apprennent correctement.
- Les expérimentations montrent que YOLOv4-tiny est un modèle performant produit des résultats satisfaisants. Néanmoins, la contrainte du temps réel a le besoin de la présence d'unité de puissance de calcul (GPU).
- Le modèle YOLOv3-tiny est moins performant sur la métrique de précision mais plus rapide.

Chapitre IV

Implémentation des algorithmes sur la carte Jetson Nano

IV. 1. Introduction

Dans les chapitres précédents on a travaillé sur le côté théorique du projet, en revanche dans ce dernier chapitre on travaille sur le côté hardware. En effet, nous allons décrire les composants utilisés et leurs fonctions dans les différents systèmes proposés. La conception matérielle et logicielle de notre travail sera présentée et expliquée, ainsi que les différents tests effectués. Dans ce chapitre, on discute la partie la plus importante de notre mémoire, est comment développer et implémenter les différents systèmes de sécurité dans la carte NVIDIA en temps réel.

Dans le chapitre précédent nous avons évalué les performances des modèles YOLO, ils ont donné de bons résultats pour la détection de masque et la détection des panneaux de signalisation, ce qui montre la capacité d'un réseau de neurones à apprendre et d'atteindre de bonnes performances dans des tâches variées.

YOLOv4-tiny a donné les meilleurs résultats et a pu achever le but principal d'apprentissage d'une machine qui est la bonne capacité de généralisation. YOLOv4-tiny avait l'avantage qui autorise son emploi dans des situations diverses, sans toucher à son fonctionnement.

YOLOv3-tiny et YOLOv4-tiny sont des algorithmes qui bénéficient aujourd'hui d'accélérations matérielles TPU (processeur de tenseurs), GPU (processeur graphique) embarqué à basse consommation sur les cartes de développement comme le NVIDIA Jetson Nano, etc. Ces derniers permettent de les utiliser et les exécuter sur du matériel embarqué pour une application en temps réel. Ces considérations nous ont fait opter pour l'utilisation d'approches profondes dans les méthodes de vision par ordinateur développées dans ce mémoire pour des applications de la reconnaissance faciale et la détection de masque ainsi la détection des panneaux de signalisation.

Ce chapitre est divisé en quatre parties. La première partie couvrira la configuration matérielle de la carte NVIDIA. La deuxième partie se concentre sur le développement et l'implémentation d'un système de reconnaissance faciale en temps réel. Dans cette partie, nous décrivons plus particulièrement, l'interface graphique du notre système de détection et reconnaissance de visage. Dans la troisième partie nous avons présenté l'implémentation d'un système de détection de masque. Dans la dernière partie, est abordé le développement d'un système de sécurité routière en temps réel. Ce type d'application permet de définir le concept des systèmes de transports intelligents, son rôle est d'améliorer la sécurité, l'efficacité et la convivialité dans les systèmes de transport routier, à travers l'utilisation des nouveaux algorithmes d'intelligence artificielle.

IV. 2. Configuration matérielle

IV.2.1. NVIDIA Jetson Nano

La carte de développement NVIDIA Jetson Nano est un mini-ordinateur puissant qui permet d'exécuter plusieurs tâches en parallèles y compris des réseaux de neurones pouvant être utilisés dans différentes applications telles que la classification d'images, la détection d'objets, la segmentation et le traitement de la parole. Cette plate-forme est facile d'utilisation grâce à son système d'exploitation semblable à Linux, elle consomme très peu d'énergie (environ 5 watts). La carte «Jetson Nano» est très utile dans le domaine de l'intelligence artificielle et plus particulièrement d'apprentissage profond (Deep Learning) car elle possède toutes les fonctionnalités nécessaires pour développer des projets d'IA grâce aux composants suivants (voir Figure IV.1) :

- GPU : NVIDIA Maxwel à 128 cœurs.
- CPU : Quad-Core ARM® A57 à 1,43 GHz.
- Mémoire: 4 Go 64 bits LPDDR4.
- Stockage : microSD.
- Connectivité: Gigabit Ethernet et 802.11ac sans fil.
- Caméra : 2 connecteurs MIPI CSI-2.
- Affichage : HDMI.
- USB : 1x USB 3.0 Type A, 2x USB 2.0 Type A, 1x USB 2.0 Micro-B.

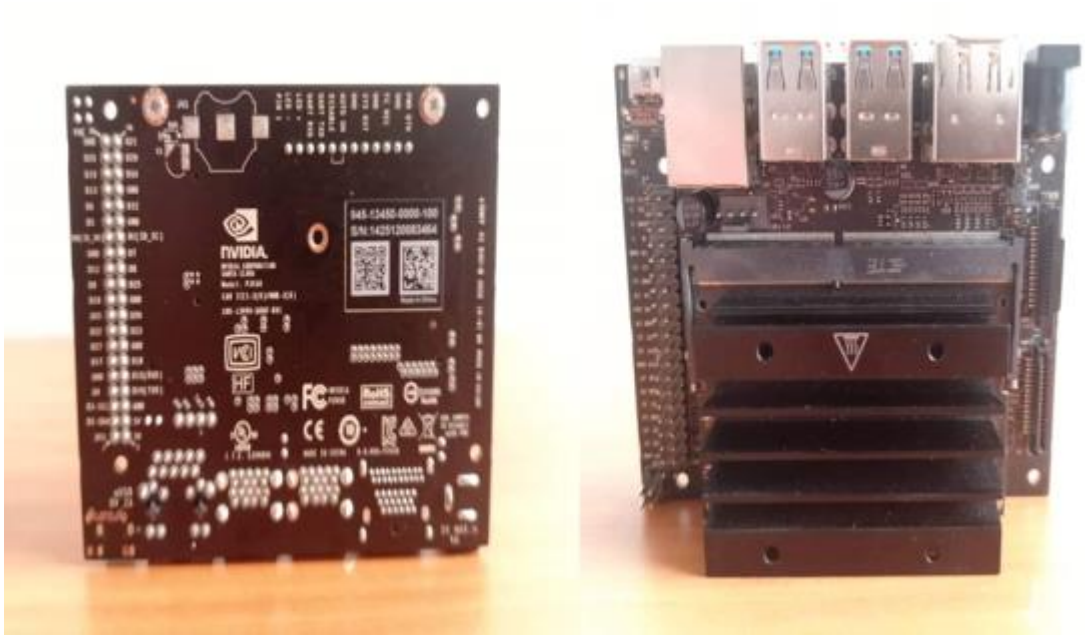


Figure IV. 1. La Carte NVIDIA Jetson Nano.

On peut accéder à la carte NVIDIA via le protocole SSH, VNC par l'intermédiaire d'un routeur WiFi, avec l'utilisation de l'adresse IP de la carte, comme la représente la Figure IV.2.

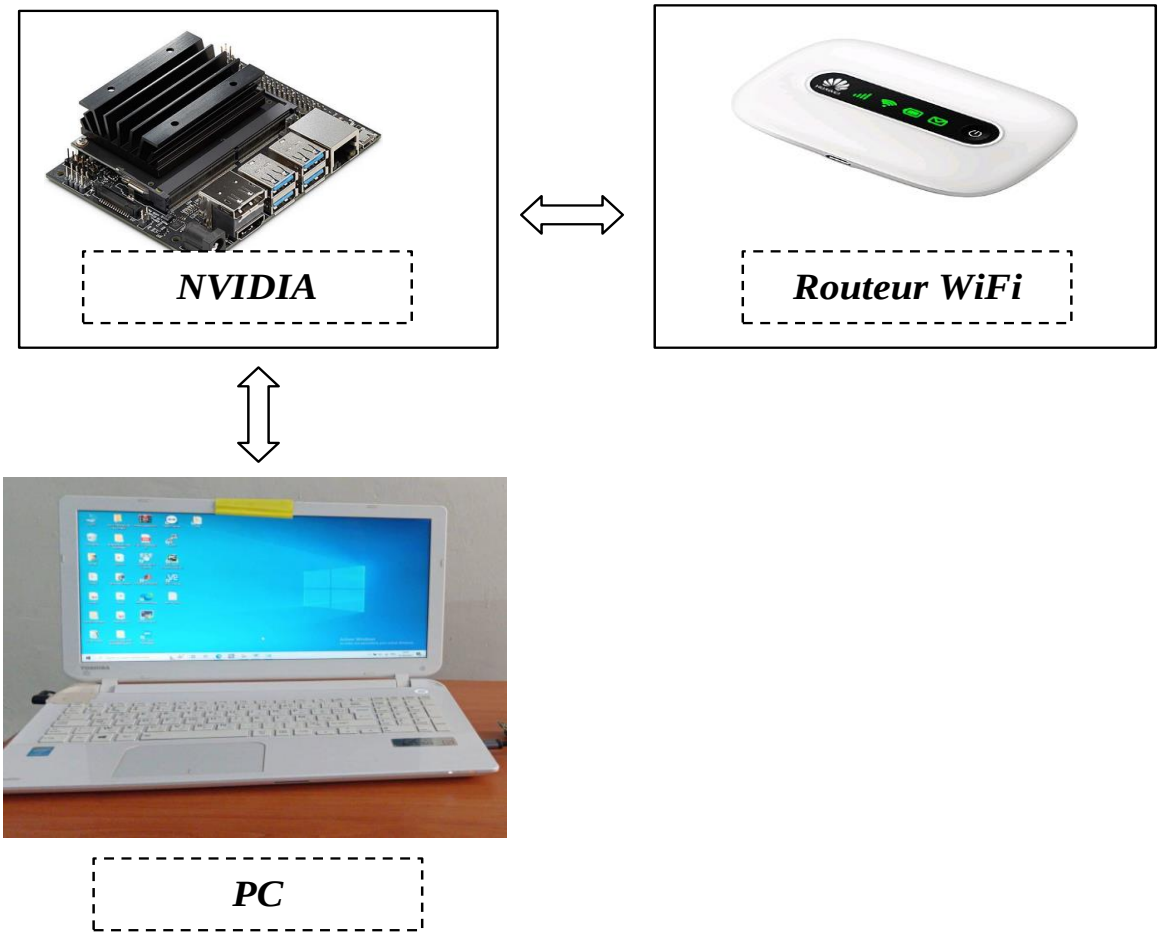


Figure IV. 2. Accès a distance de la carte NVIDIA.

IV.2.2. Caméra

Pour tester les performances de différentes méthodes proposées dans le chapitre précédent, nous avons utilisé deux types de caméra : ‘‘Havit-HV-N5080’’ et ‘‘Caméra JeWay-5152’’. L'utilisation de ces types de caméra, permet de transformer la carte NVIDIA en Webcam, caméra IP ou caméra de vidéosurveillance, comme les représentent les Figures IV. 3 et IV.4.

- **Caméra ‘‘Havit-HV-N5080’’**
  - 1 écran Full HD 1080 p.
  - Trépied de support convenant à l'affichage LCD ou CRT d'un ordinateur de bureau ou d'un ordinateur portable.
  - Prise en charge des ordinateurs courants supérieurs à XP, 2000, vista, Win7, Win8, Win10 et autres systèmes d'exploitation.
  - La longueur de câble 1.5m.

- Sensibilité : 830 V/lux-sec@550nm.
- Gamme dynamique : 68db@8X Gain.
- Rapport signal/bruit : 45db.
- Éclairage minimum : 0.01lux.
- Résolution supportée : 640*480 1280**720 1920*1080 2592*1944.
- Capteur d'image : COMS.
- Vitesse : 30 fps/s(VGA).
- Type de connexion : USB2.0 haute vitesse (Isochrone).
- Format de sortie : MJPEG/YUV2(YUYV).

Figure IV. 3. La Caméra Havit (Hv-N 5080).

- **Caméra JeWay-5152**

Jeway est une Webcam avec une fréquence d'images maximale de 30 ips. La webcam convient à toutes les applications Internet et est facile à connecter (USB) et à installer, elle est compatible avec les cartes électroniques NVIDIA et Raspberry, comme présenté dans la Figure IV.4.

- Résolution de l'écran :640*480.
- MPN (numéro de pièce du fabricant) : JW-5152.
- Modèle : JW-5152.
- Nombre de capteurs/pixels :0,3 mégapixel.
- Manuel de langue : Anglais.
- Sans fil : Non.
- Type d'aliment : USB.
- Type d'entrée/sortie : USB/USB 2.0 A.

- Longueur du câble : 1,3 mètre.

Figure IV. 4. Caméra JeWay-5152.

IV. 3. Développement d'un système de reconnaissance faciale en temps réel

La reconnaissance des visages est une application importante du traitement d'images en raison de son utilisation à de nombreuses fins. L'identification d'individus dans une organisation à des fins de présence est l'une de ces applications de la reconnaissance faciale. La tenue et le suivi des registres de présence jouent un rôle essentiel dans l'analyse de la performance de toute organisation. L'objectif du développement d'un système de gestion des présences est d'informatiser la méthode traditionnelle de prise des présences. Le système automatisé de gestion des présences effectue les activités quotidiennes de marquage des présences avec une intervention humaine réduite.

Il existe d'autre application, les plus importantes concernent la sécurité. La sécurité est le principal domaine d'application des systèmes de reconnaissance faciale.

Le système s'assure dans ce cas que l'utilisateur est bien un utilisateur valide avant de l'autoriser à accéder à un élément donné. Cela peut être utilisé dans un lieu public, par exemple : autoriser l'accès à une organisation, pour suivre et identifier une personne à partir des images d'un système de vidéosurveillance.

Bien que de nombreux algorithmes différents existent pour effectuer la détection des visages, chacun a ses propres faiblesses et forces. Certains utilisent les tons, d'autres les contours, et d'autres encore sont plus complexes, impliquant des modèles, des réseaux neuronaux. Ces algorithmes souffrent du même problème: ils sont coûteux en termes de calcul. Une image n'est qu'une collection de valeurs de couleur ou d'intensité lumineuse. L'analyse de ces

pixels pour la détection des visages prend du temps et est difficile à réaliser en raison des grandes variations de formes et de couleurs. Dans ce système, nous avons tendance à utiliser les classifeur de Haar fourni par OpenCV (Open source Computer Vision).

La détection d'objets en utilisant les descripteurs de Haar est une méthode efficace de détection d'objets proposée par Viola et Jones. Il s'agit d'une approche basée sur l'apprentissage automatique où une fonction en cascade est entraînée à partir d'un grand nombre d'images positives et négatives. OpenCV fournit à la fois des classifieurs et un détecteur, ce qui facilite l'intégration du système de reconnaissance des visages. Le système marquera la présence d'un individu dans une feuille Excel.

La figure suivante, montre le processus de développement des phases principales de système de reconnaissance faciale.

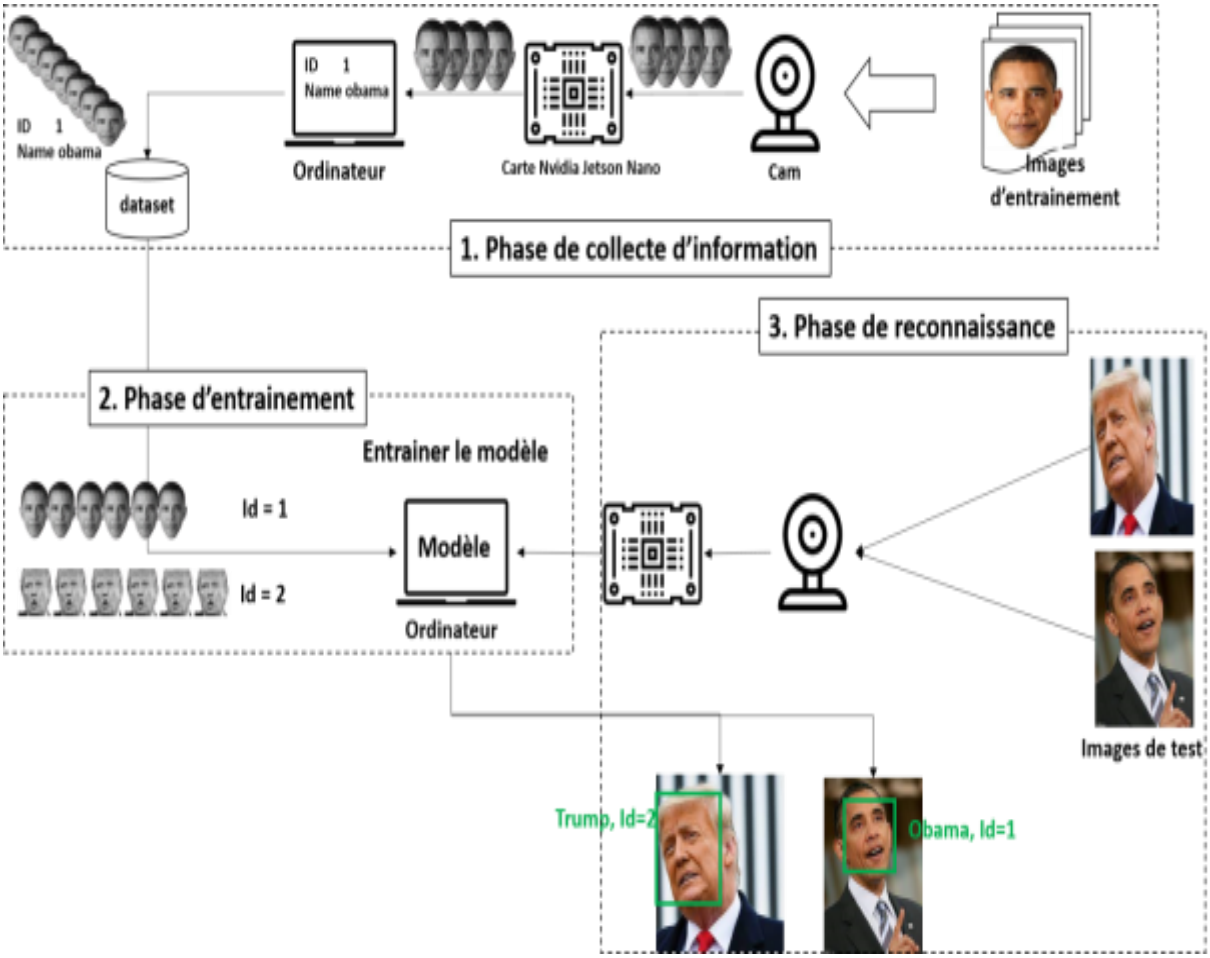


Figure IV. 5. Schéma fonctionnel du système de reconnaissance faciale.

L'interface de système est développée en utilisant la bibliothèque GUI (Graphic User Interface) de Python nommée : TKinter. Elle est simple à utiliser comme la montre la Figure IV.6.

Figure IV. 6. Interface du Système de reconnaissance facile.

IV.3.1. Description de l'interface graphique du système de reconnaissance faciale

L'interface de notre système se compose de trois fonctionnalités importantes :

- Le bouton «**Take Images**»: en introduisant un identifiant dans le champ ID sous format numérique, et une chaine alphabétique désignant le nom de la personne qu'on va être enregistrée dans le système, ce bouton permet de démarrer une caméra connectée à la carte NVIDIA Jetson Nano. Le flux de la vidéo est directement passé à un modèle pré-entrainé qui permet de détecter le visage d'une personne sur une image, et prend plusieurs captures faciales de la personne figurant dans le champ capté par la caméra, puis les enregistrer dans un dossier.
- Le bouton «**Train Images**»: après avoir pris des captures faciales d'un individu, le bouton **Train Images** permet d'entrainer le modèle de reconnaissance faciale sur l'ensemble des images contenues dans son répertoire des images prises par le premier bouton (**Take Images**).
- Le bouton «**TrackImages**»: ce bouton permet de visualiser le résultat de ce système de reconnaissance facile. En Cliquant sur le bouton **Track Images**, la caméra connectée au système démarre encore, mais cette fois, en utilisant les images enregistrées précédemment, et le modèle de reconnaissance faciale, elle va reconnaitre les visages des personnes enregistrées dans son ensemble de données.

Le bouton «**QUIT**» nous permet de quitter le système, autrement dit, de fermer la fenêtre de l'application.

L'interface de notre système contient deux champs permettant d'afficher des informations pour l'utilisateur :

- Le champ **Notifications**: quand l'entrainement du modèle de reconnaissance faciale termine son entrainement sur l'ensemble des images enregistrées, une notification s'affiche au niveau de ce champ.
- Le champ **Attendance**: des informations sur la personne reconnue par le système seront affichées au niveau de ce champ.

IV.3.2. Partie Expérimentale

Tout d'abord, parlons de la partie matérielle nécessaire pour réaliser notre système de reconnaissance faciale, Il existe deux modes pour connecter à la carte NVIDIA Jetson Nano, dans cette partie nous avons choisi d'utiliser le mode de fonctionnement sans tête ''headless'', d'abord on a placé le [J48] sur la connexion de périphérique série par le biais du port microUSB [J28] à côté de la prise d'alimentation, puis nous avons branché nos caméras Havit (HV-N5080) et JeWay (Jw-5152) aux ports USB3.0 de la carte. La Figure IV.7, montre à quoi ressemblera le Jetson lorsque nous avons connecté toutes les pièces ensemble.

Figure IV. 7. Matérielle nécessaire pour réaliser un système de reconnaissance facile.

Figure IV. 8. La Carte Nvidia Jetson Nano après l'alimentation.

Une fois que la carte NVIDIA Jetson Nano est alimentée par un adaptateur d'alimentation de (5v et 4A), selon la Figure IV.8, elle sera détectée comme un port COM dans le gestionnaire de périphériques. Ensuite, on a ouvert une session dans le Nano via le port COM en utilisant le logiciel de terminal virtuel ‘‘Putty’’.

Pour pouvoir se connecter à distance à notre NVIDIA, il faut le connecter à un réseau local et faire appel au protocole SSH. Le serveur SSH est activé par défaut sur la carte NVIDIA. Sous Linux, le client étant intégré dans la plupart des distributions. Par contre, sous Windows, nous devons installer «Putty» qui est un client SSH. Ensuite, il suffit d'entrer l'adresse IP du NVIDIA dans Putty pour pouvoir se connecter.

IV.3.3. Fonctionnement de système de reconnaissance faciale

Une fois que nous entrons sur le bureau de NVIDIA Jetson Nano nous pouvons ouvrir l'interface de la reconnaissance faciale à travers l'exécution de fichier nommé Face_recognation.py (voir la Figure IV.9).

Figure IV. 9. Le fichier Face_recognation

On peut marquer la présence d'une personne en faisant simplement face à la caméra. Quand nous lançons Face_recognation.py, une fenêtre s'ouvre et demande d'entrer l'identité et le nom de la personne. Après avoir entré le nom et l'identité, nous devons cliquer sur le bouton Take Images. En cliquant sur Take Images, la caméra de la carte NVIDIA commence à prendre un échantillon d'image de la personne. Cet ID et ce Nom sont stockés dans le dossier Student Details et le nom du fichier est StudentDetails.csv. Il prend 60 images comme échantillon et les stocke dans le dossier Training Image. Après avoir terminé, il notifie que les images ont été sauvegardées. Après avoir pris l'échantillon d'image, nous devons cliquer sur le bouton Training Image. Cela prend quelques secondes pour entraîner la machine sur les images qui sont prises en cliquant sur le bouton Take Image et crée un fichier Trainner.yml et le stocke dans le dossier Training Image Label. Toutes les configurations initiales sont maintenant terminées. En cliquant sur le bouton Track Image, la caméra de la machine en marche est à nouveau ouverte. Si le visage est reconnu par le système, l'identité et le nom de la personne sont affichés sur l'image. La figure au-dessous démontre des tests réalisés durant le développement de notre système.

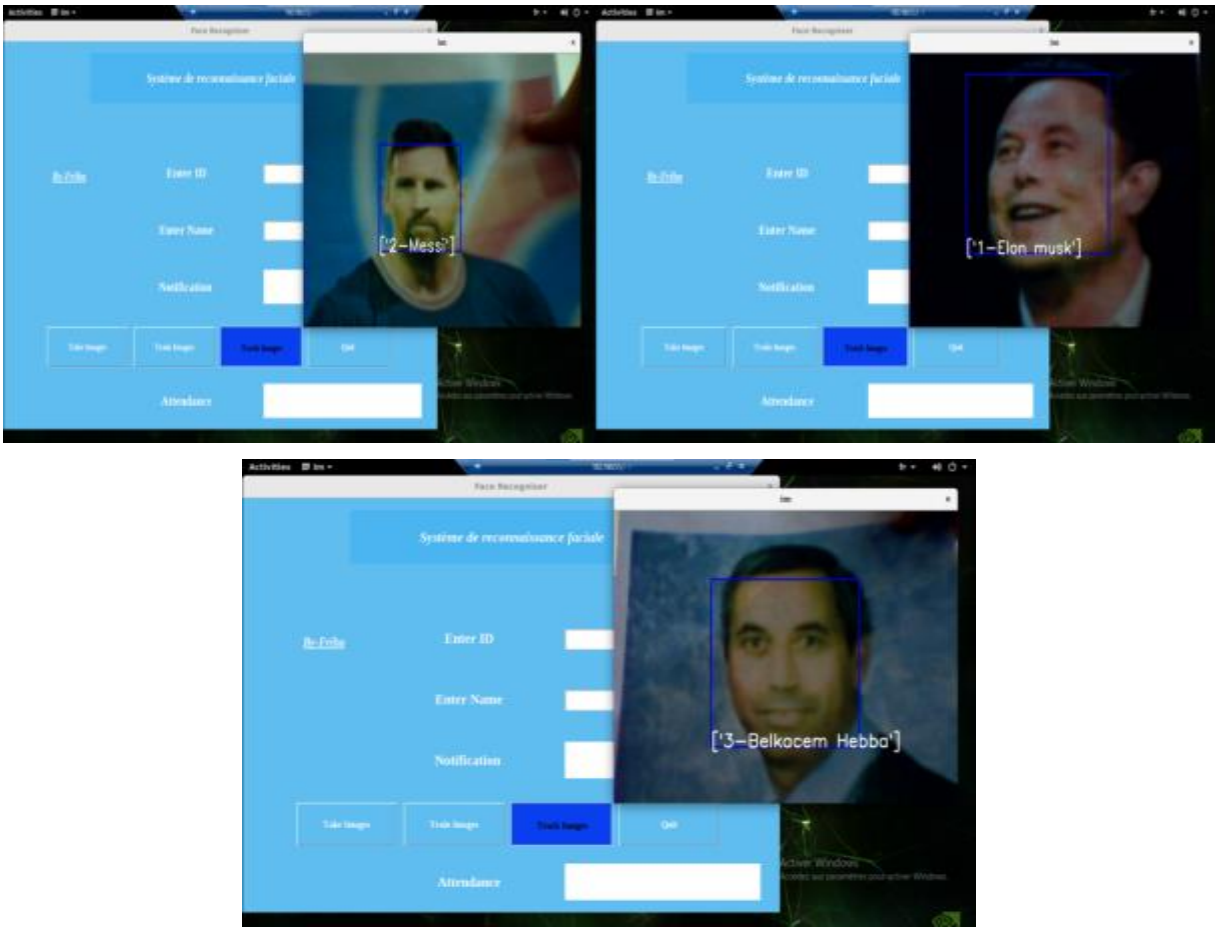


Figure IV. 10. Illustration montrant comment identifier des personnes via une application de reconnaissance faciale.

Appuyez sur Q (ou q) pour quitter cette fenêtre. Après avoir quitté cette fenêtre, les présences de la personne seront stockées dans le dossier Attendance sous forme de fichier CSV avec le nom, l'identifiant, la date et l'heure et elles sont également disponibles dans la fenêtre (Figure IV.11).

Figure IV. 11. Interface graphique après le test.

Les résultats obtenus quand nous avons utilisé la caméra Jeway et la caméra Havit étaient proches au terme de la reconnaissance correcte de la personne. La différence était remarquable dans le temps de reconnaissance qu'a pris la caméra Havit qui était élevé par rapport au temps pris par la caméra Jeway.

IV. 4. Fonctionnement de système de détection de masque

Avec l'objectif de réaliser un système intelligent basé sur l'apprentissage profond, nous avons travaillé sur la détection du masque qui peut être un moyen de protection contre certaines maladies et épidémies qui se propagent par l'air comme le (COVID-19) dans les hôpitaux et les lieux publics où le port d'un masque sera une obligation qui sert à protéger nos vies et celles des autres.

L'application est réalisée principalement par le détecteur YOLOv4-tiny qui est essentiellement un modèle pré-entraîné sur la détection d'objets. YOLOv4-tiny montre une bonne précision de performance et une bonne capacité de généralisation selon notre mise en œuvre dans la détection de masques et de panneaux routiers discutée dans le chapitre III.

Le travail est divisé en deux parties comme le montre la Figure IV.12, la première phase consiste à entraîner les modèles sur la détection de masque où nous avons exploité une étape

principale qui est l'apprentissage par transfert pour obtenir un modèle optimisé, et nous avons terminé cette étape en gardant les meilleurs modèles.

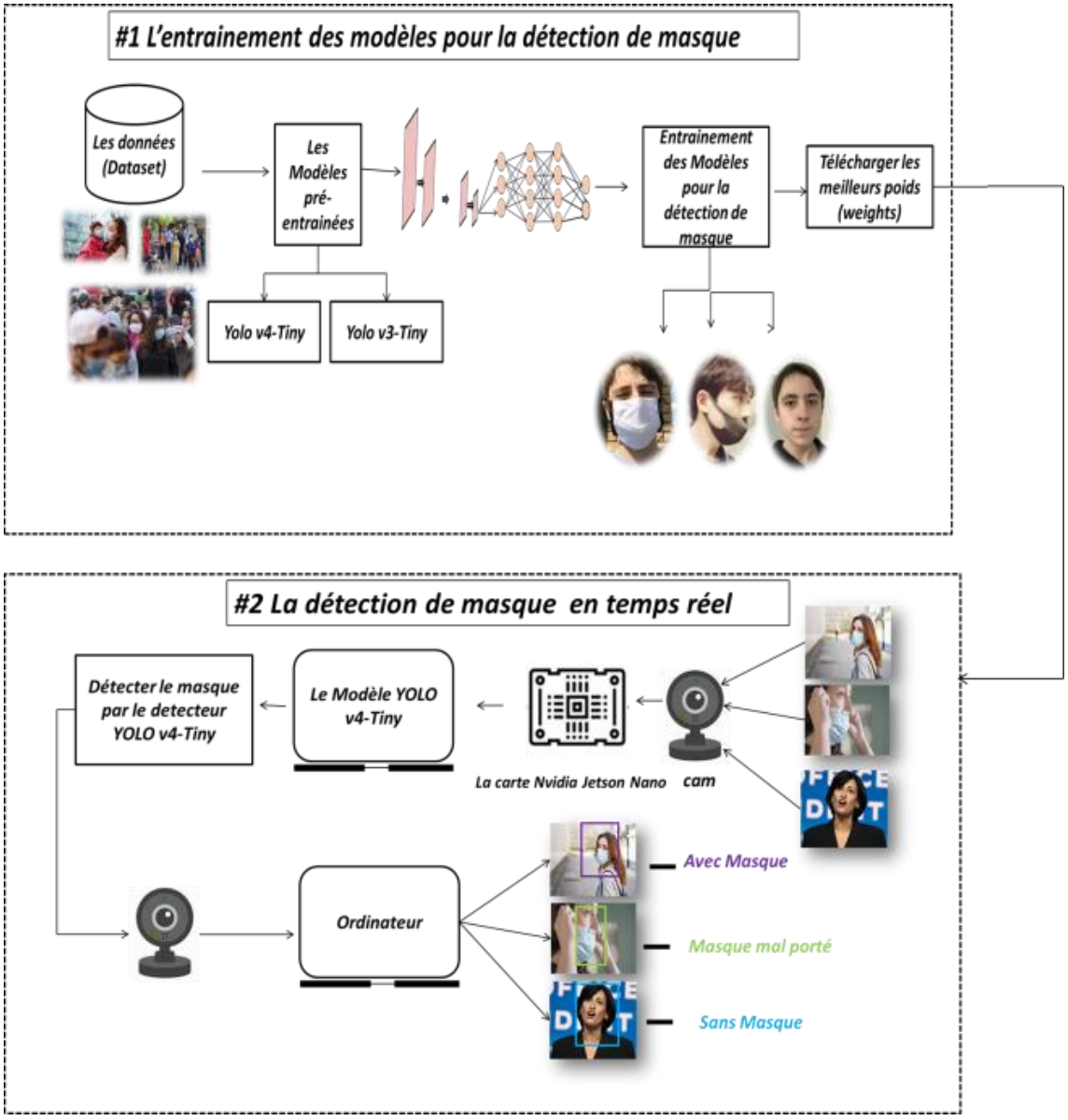


Figure IV. 12. Le fonctionnement de système de la détection de masque.

Nous avons choisi le modèle YOLOv4-tiny comme détecteur de masque en temps réel, pour la deuxième phase du projet car c'est le modèle le plus performant.

Les performances que nous avons pu obtenir de la Jetson Nano et l'exécution d'une ligne de traitement vidéo avec des modèles d'apprentissage automatique (voir La Figure IV.13). Le résultat est que la caméra peut suivre des visages masqués à 30 images par seconde en utilisant le modèle YOLOv4-tiny.

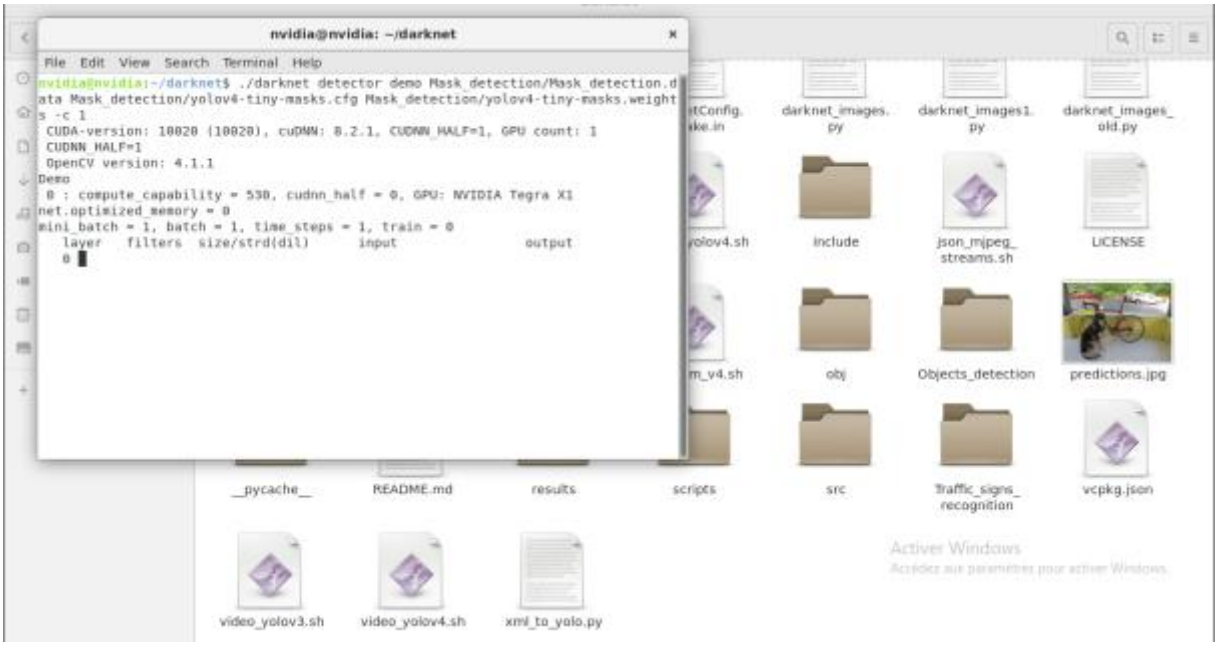


Figure IV. 13. Détection de masque par l'exécution de modèle YOLOv4-tiny.

Ce qui se passe sur l'ordinateur portable est une vue en direct de la caméra de détection de masque qui dessine des cases de détection sur les visages des personnes qui passent, elle utilise un modèle YOLOv4-Tiny pour identifier les visages et déterminer s'ils portent des masques ou non, l'algorithme suit les visages à travers l'image et fait plusieurs déductions sur les visages avant de dire si un visage est avec masque ou non et montre le pourcentage global de personnes portant des masques.

La caméra JeWay (Jw-5152) avec une résolution de (640×480) a montré une détection facile et rapide du masque avec des pourcentages de précision élevés, en particulier dans le cas où le masque est mal porté. D'autre part, la caméra Havit (HV-N 5080) avec une résolution de (2592×1944) qui est une caméra à haute résolution a eu des difficultés à détecter le masque et a pris beaucoup de temps pour détecter le cas d'un masque mal porté. Nous pouvons donc voir que le nombre de pixels a un impact direct sur le fonctionnement de notre système de telle sorte que lorsque la résolution est élevée effectivement le nombre de pixels est grand; donc il faut plus de temps au système pour détecter le masque est l'inverse pour une situation où la résolution n'est pas assez bonne donc le nombre de pixels est moins ce qui rend la détection du masque plus facile.

Donc, le nombre de frames par seconde (FPS) diffère d'une caméra à l'autre. La collection des figures IV.14 à IV.16 représente les résultats des tests du système de détection de masques en temps réel par les deux caméras Jeway (JW-5152) et Havit (HV-N5080).

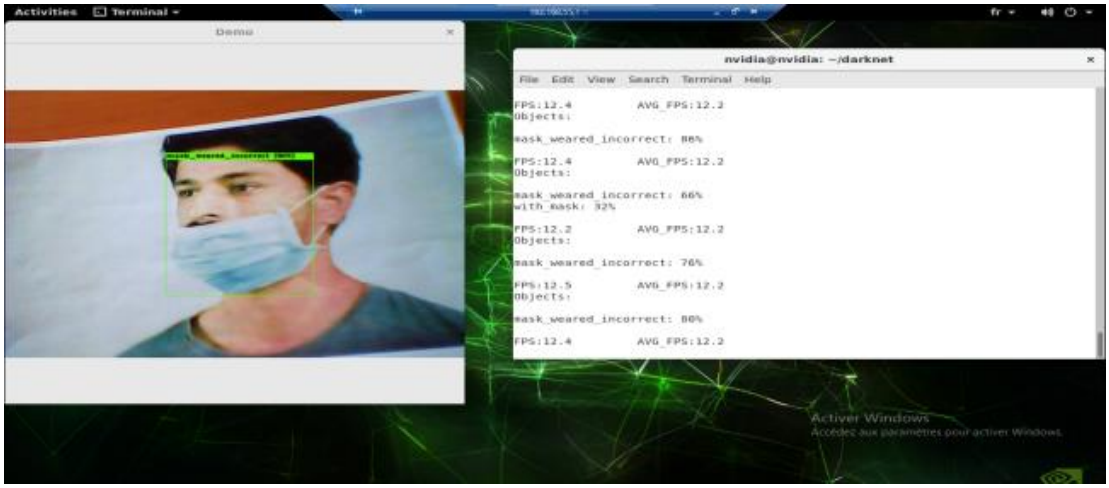


Figure IV. 14. Le résultat du test en utilisant la caméra Jeway (Jw-5152) dans le cas où la personne porte le masque de manière incorrecte.

Figure IV. 15. Le résultat du test de masque sur l'image d'une personne par la caméra Jeway (Jw-5152) et avec le pourcentage de précision.

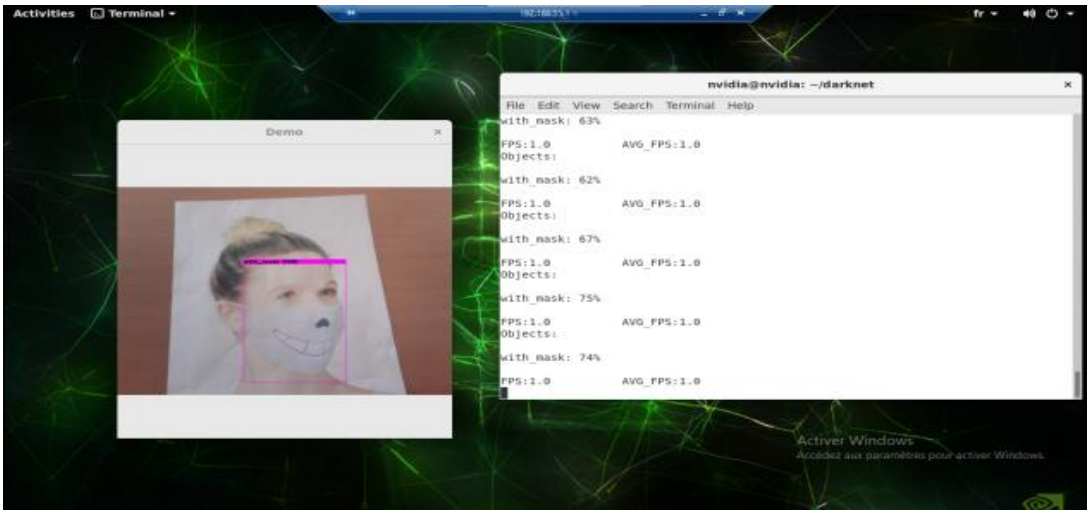


Figure IV. 16. Résultat du test de détection de masque sur l'image d'une femme à l'aide de la caméra Havit (HV-N 5080)

Nous pouvons constater que la moyenne de nombre de frames par seconde varie entre 12 à 15 pour la caméra Jeway et 1 frame par seconde pour la caméra Havit, cette différence dans le nombre d'images prises par les deux caméras se reflète directement sur le temps que le système met à détecter le masque; néanmoins la caméra Jeway est presque 15 fois plus rapide que la caméra modèle Havit.

IV. 5. Développement d'un système de sécurité routière en temps réel

L'être humain se déplace en permanence, de son domicile à son travail, pour bénéficier de services, ... etc., et pour ce faire, il utilise les moyens de transport qui sont importants dans notre vie quotidienne. Pour cela, l'intelligence artificielle est intervenue et a fait naître les systèmes de transport intelligents (STI) qui sont des systèmes dans lesquels les technologies de l'information et de la communication sont appliquées dans le domaine du transport routier, de l'infrastructure, de la gestion du trafic et de la mobilité.

Chaque année, plus de personnes meurent sur les routes, donc le STI est un système intégré de routes et de véhicules, conçu pour contribuer à améliorer le confort ainsi que la conservation de l'environnement par la réalisation d'un trafic fluide en soulageant la congestion du trafic, la sécurité routière, cette dernière est une partie essentielle de nos vies en tant que piétons, cyclistes, conducteurs ou passagers d'un véhicule, parmi ces moyens qui utilise: «les panneaux de signalisation».

Le but des panneaux routiers se diriger, de repérer ou encoure respecter le code de la route pour une conduite plus sure et en toute sécurité, Les signes peuvent partager des informations, donner des directions, nous rappeler de faire des choses ou nous avertir de certaines choses. Certains panneaux utilisent des mots, les panneaux sont identifiables à travers leur forme par exemple les panneaux de danger, présentent sur les panneaux triangulaires correspondant au danger, leur objectif avertir les usages d'un danger potentiel imminent, les panneaux d'interdiction pour limiter les vitesses, ...etc.

La détection des panneaux de signalisation en temps réel est une application importante qui traite une grande variété d'information lecture des panneaux, données du véhicule, …

IV.5.1. Fonctionnement de système de détection des panneaux routière

Notre méthode est divisée en deux parties comme le montrent les deux Figures IV.17 et IV. 18. La première partie : l'entrainement des modèles sur la détection des panneaux de signalisation routière, en utilisons l'apprentissage par transfert, nous avons terminé par le téléchargement des meilleurs poids pour le tester sur différent image qui contient 04 classes (prohibitory, mandatory, danger, other).

La deuxième partie : la détection des panneaux de signalisation routière en temps réel, dans cette partie on utilise, la carte NVIDIA Jetson Nano et les modèles pré-entrainé YOLO.

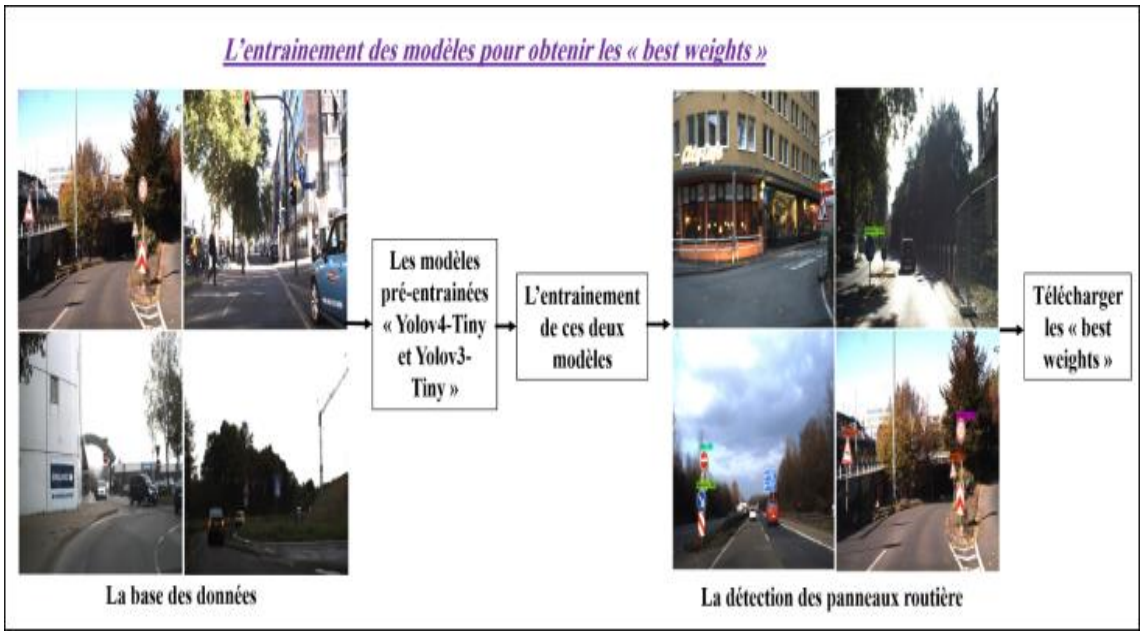


Figure IV. 17. Schéma de principe montrant l'entrainement des modèles dans le cas des systèmes de sécurité routière.

Figure IV. 18. Schéma de principe montrant la détection des panneaux routière en temps réel.

Afin de démontrer l'efficacité de notre méthode, un modèle YOLOv4-tiny a été développé pour montrer la détection en temps réel. La caméra Havit (HV-N 5080) à un nombre de pixels plus élevé que la caméra JeWay (Jw-5152). La caméra JeWay donne donc de bons résultats

de précision (entre 70 et 95%) lors de la détection des panneaux de signalisation routière, mais elle prend du temps, tandis que la caméra Havit ne prend pas du temps, mais donne des résultats de détection en temps réel moins précis (entre 20 et 70%). Les Figures IV.19 à IV.22, montrent plusieurs tests sur la carte NVIDIA en temps réel.

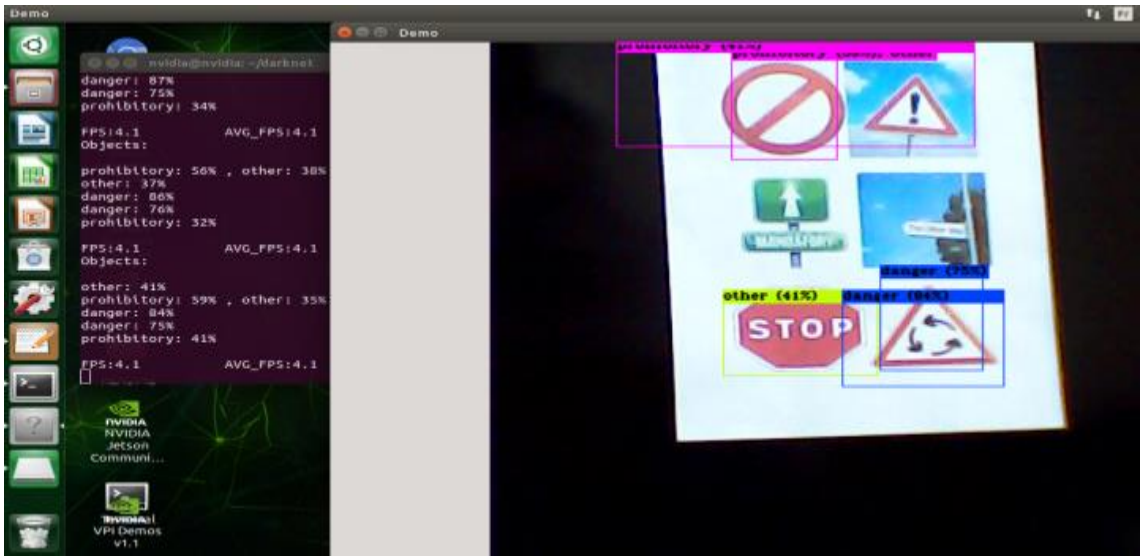


Figure IV. 19. Le résultat du test de détection sur plusieurs panneaux routiers par la caméra Jeway (Jw-5152).

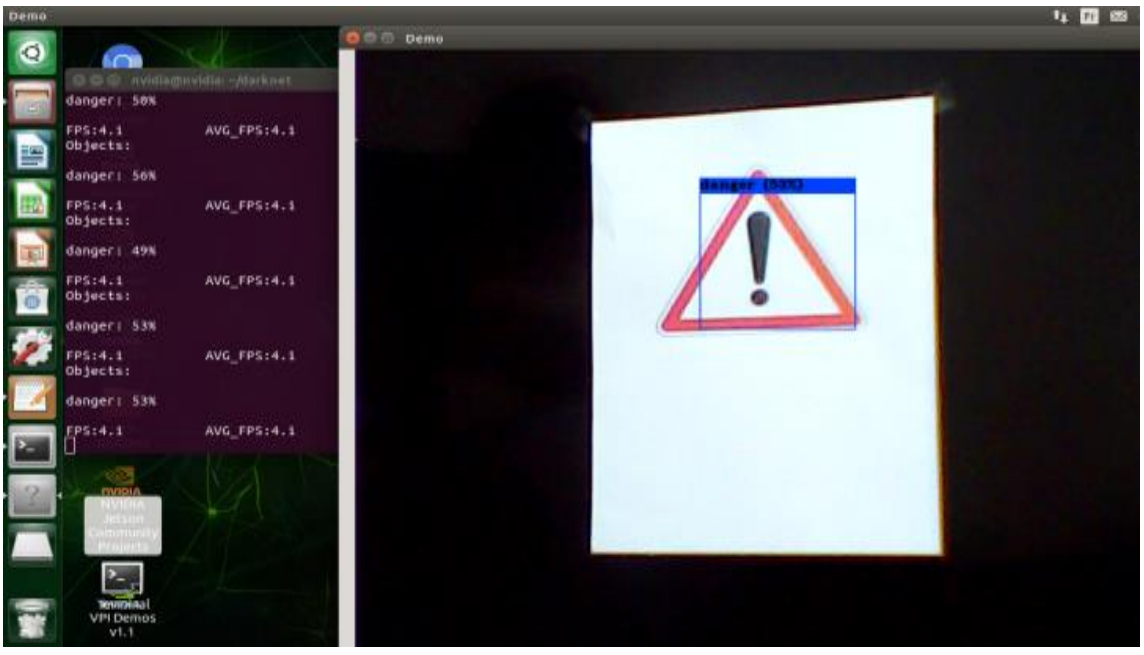


Figure IV. 20. Le résultat du test de détection sur le panneau "Danger" en employant la caméra Jeway (Jw-5152).

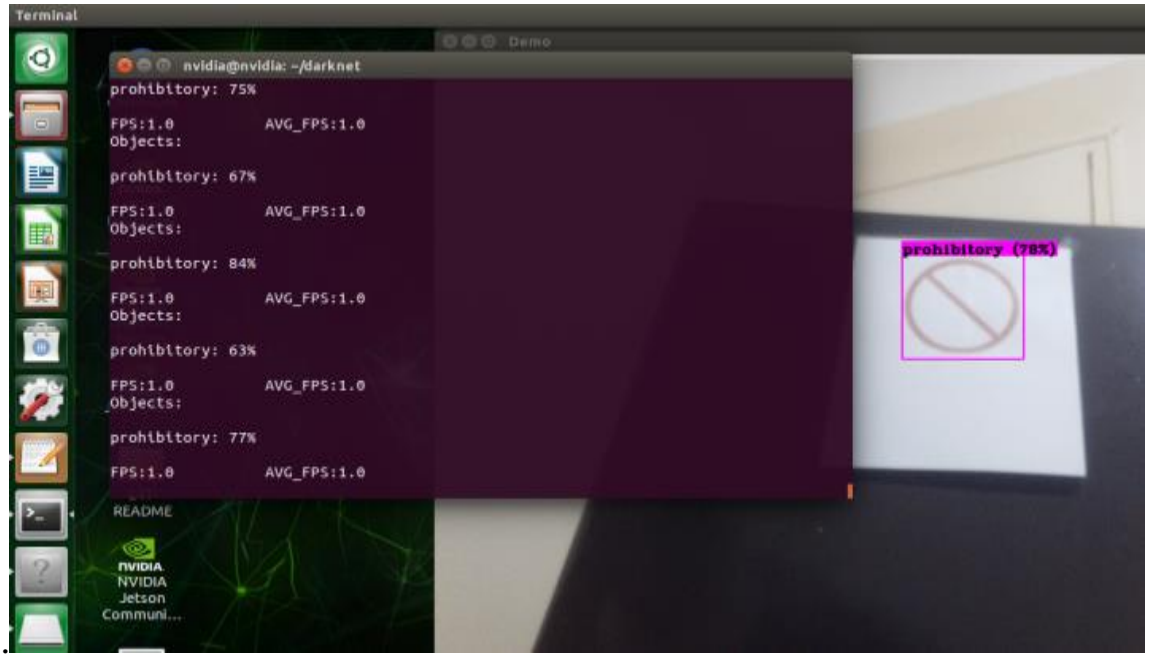


Figure IV. 21. Le résultat du test de détection sur le panneau d'interdiction en utilisant la caméra Havit (HV-N 5080).

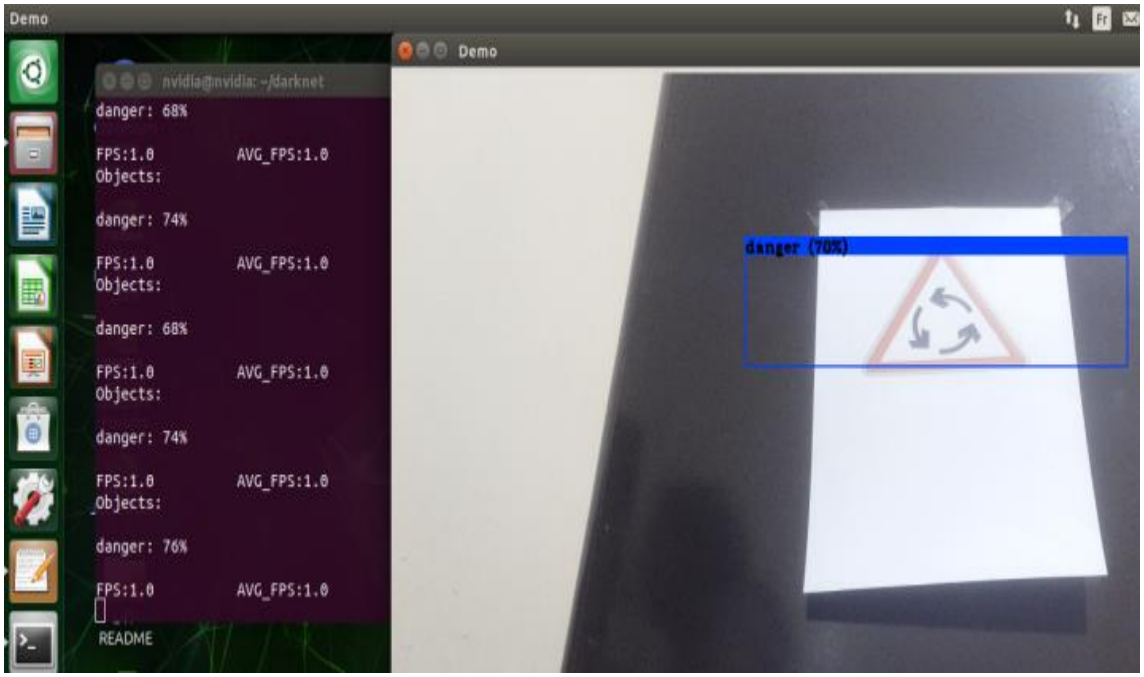


Figure IV. 22. Le résultat du test sur le panneau d'interdiction en utilisant la caméra Havit (HV-N 5080).

IV. 6. Conclusion

Dans ce chapitre, nous avons implémenté des algorithmes sur la carte NVIDIA Jetson Nona pour développer les applications suivantes : Système de reconnaissance faciale, Système de détection de masque, et enfin Système de sécurité routière avec des schémas blocs pour

montrer comment chacun fonctionne. Ensuite, on a présenté l'efficacité et les performances de nos systèmes implémentés sur la carte NVIDIA. Enfin, on a fait plusieurs tests afin d'éclairer mieux le fonctionnement des différentes méthodes utilisées dans notre travail.

Conclusion générale

Conclusion générale

L'objectif de ces travaux était d'explorer des méthodes de vision par ordinateur et d'apprentissage profond pour assister des tâches de détection et de classification des objets dans le but de développement des systèmes de sécurité intelligents.

La première contribution consiste en un algorithme de détection de visage, la méthode de Viola et Jones est l'une des méthodes les plus connues et les plus utilisées, en particulier pour la détection de visages et la détection de personnes, la méthode propose une architecture pour combiner plusieurs classifieurs en cascade, qui permet un net gain en temps de détection. Enfin la méthode de Viola et Jones prend un grand temps dans la phase d'entrainement mais elle est efficace pour la détection en temps réel.

La deuxième contribution concerne le choix et le paramétrage judicieux d'un réseau de neurones profond de détection d'objets employé pour le développement des systèmes de sécurité intelligents, les réseaux de neurones classiques et les réseaux de neurones convolutifs ont été présentés en détail afin de se familiariser avec le concept de base de ces notions. Nous avons opté les architectures YOLO pour la détection et la reconnaissance de masques et des panneaux de signalisation routière en se basant sur des considérations de vitesse d'exécution et de précision des prédictions. Le développement de cette approche a été fondé sur deux étapes. La première étape, est d'appliquer l'apprentissage par transfert à travers une proposition de l'architecture optimale pour les méthodes : YOLOv3-tiny et YOLO v4-tiny, pour atteindre notre objectif, nous avons utilisé deux types différents de bases de données qui sont bien annotées. A noter que le choix de la base de données a un impact direct sur la procédure d'apprentissage et par conséquents ses performances. Puis nous avons préparé l'environnement d'exécution Darknet qui contient les bibliothèques nécessaires pour la formation et l'apprentissage de nos modèles (YOLOv3-tiny et YOLOv4-tiny). Dans la deuxième étape nous avons testé les performances des méthodes d'intelligence artificielle développées, afin de les utiliser sur des scènes réelles. Les résultats obtenus sur la base de données de détection de masques avec le modèle YOLOv4-tiny étant meilleurs que ceux obtenus par le détecteur YOLOv3-tiny, tandis que les résultats sur la deuxième base de données de panneaux de signalisation sont assez similaires. Nous avons constaté que malgré YOLOv3-tiny a pris moins du temps en exécution, le modèle YOLOv4-tiny est le plus performant garce à sa précision élevée et sa capacité de généralisation et pour cela nous avons gardé les meilleurs paramètres de ce dernier qui sont utilisés dans notre système en temps réel avec la carte NVIDIA Jetson Nano équipée d'une caméra et une communication WiFi.

La troisième contribution est une réalisation de trois systèmes qui sont : une application représentée par une interface développée en utilisant la bibliothèque GUI (Graphic User Interface) de langage Python TKinter, pour la reconnaissance de visage qui se base sur les classifieurs de Haar en cascade, la procédure passe par deux phases, la première est la détection de visage, et la deuxième est l'identification de visage par la classification des caractéristiques. C'est à dire enregistrer l'image d'une personne dans une base de données. Ensuite, il faut que la personne passe devant la caméra qui va capturer son visage. Puis, le système de reconnaissance faciale va rechercher dans sa base de données l'image qui présente le plus de point commun avec l'image prise par la caméra. Ce système est utile pour plusieurs applications de sécurité comme la reconnaissance faciale par les autorités : recherche des criminels et contrôle aux frontières ou application de gestion de présence, …, etc.

Le deuxième système est pour la détection de masque dans une vidéo par le modèle YOLOv4-tiny, qui peut servir à la protection contre des maladies qui propagent dans l'air tel que la pandémie de la COVID 19.

Nous avons également ajouté un système qui reconnaitre les panneaux de signalisation utilisant le détecteur YOLOv4-tiny, qui peut utiliser comme une application sur véhicules autonomes, va améliorer la sécurité routière et réduire les accidents.

Le résultat intéressant est que les réseaux de neurones profonds de détection d'objets peuvent être employés pour réaliser de la détection de points d'intérêt et permettent d'obtenir de bonnes performances. Les expérimentations sur les réseaux de la famille YOLO montrent qu'ils sont robustes et performants. En effet, les performances dépendent peu des principaux paramètres tels que la définition du réseau ou la qualité et la quantité de la base de données. C'est la profondeur du réseau qui a le plus d'influence sur la performance, cependant, elle a un impact négatif sur la vitesse d'exécution. Le choix d'une architecture neuronale adaptée à une application donnée dépend donc de ce compromis et du matériel de calcul embarqué disponible.

Les perspectives

Ce travail nous a permis de traiter des problèmes d'ordre pratique et nous avons eu l'occasion de se rapprocher du monde de l'intelligence artificielle et découvrir l'importance de ce domaine actuellement. Grâce au travail continu, on a pu atteindre notre but et satisfaire le cahier de charge, mais cela ne veut pas dire qu'il est complet, nous proposons que le travail réalisé soit la base de toute série d'améliorations que nous n'avons pas eu la chance de les faire par manque de temps et de matériel. A noter que l'inconvénient majeur de ce type de

projet est le coût plus élevé de la carte NVIDIA (à partir de 40.000DA) et les caméras à haute résolution (à partir de 20.000DA).

Le nombre d'améliorations que peuvent être ajoutés sont:

- Etude approfondie de la phase d'algorithme d'apprentissage. La sélection de l'algorithme joue également un rôle crucial dans la précision et le temps requis pour le traitement.
- La réalisation de notre travail sur la carte NVIDIA Jetson Nano, qui à donnée des résultats très satisfaisants. Mais elle comporte des limites qui ont conduit à proposer d'utiliser d'autre carte comme NVIDIA Xavier.
- Améliorer notre système pour détecter les visages des personnes qui ne sont pas directement face à la caméra.
- Tester d'autre type de caméra à haute résolution.
- Dans les systèmes de transport intelligent, la réalisation d'une voiture intelligente est une tâche très difficile car en combinant plusieurs algorithmes complexes. Comme perspective, avec l'utilisation de l'intelligence artificielle, en peut intervenir dans plusieurs tâches telles que la planification de trajectoire, la surveillance des véhicules, la détection d'obstacles, etc.
- Détection de masque et identification de la distance sociale à l'aide de l'Internet des objets (IOT) et les méthodes d'intelligence artificielle.

Bibliographie

Bibliographie

[1] B. Barraud, "L'intelligence artificielle – Dans toutes ses dimensions," *Harmattan, hal-02327501v3,* 2019.

[2] B. Braunschweig, "Intelligence Artificielle: Les défis actuels et l'action d'INRIA" *Saclay, FR: INRIA Livre Blanc 1, Available via: https://hal.inria.fr/hal-01564589,* 2016.

[3] M. Khammari, "Détection et suivi de visages en temps réel sur flux vidéo," *Thèse de doctorat, Université de Badji Mokhtar, Annaba,* 2016.

[4] Y. Messai, K. Chara, F. Srairi, F. Douak, "Object Tracking Platform for Color Object Detection using Genetic Algorithm Optimization," *International Conference on Electrical, Communication, and Computer Engineering (ICECCE),* pp. 1-6, 2020.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision,* vol. 57, pp. 137-154, 2004.

[6] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, Y. Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence,* vol. 31, pp. 210-227, 2008.

[7] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 779-788, 2016.

[8] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE signal processing letters,* vol. 23, pp. 1499-1503, 2016.

[9] B. C. Lovell, S. Chen, A. Bigdeli, E. Berglund, C. Sanderson, "On intelligent surveillance systems and face recognition for mass transport security," *IEEE 10th International conference on control, automation, robotics and vision,* pp. 713-718, 2008.

[10] M. Owayjan, A. Dergham, G. Haber, Fakih N., A. Hamoush, E. Abdo, "Face recognition security system. ," *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering,* pp. 343-348, 2015.

[11] P. B. Balla and K. T. Jadhao, "IoT based facial recognition security system," *international conference on smart city and emerging technology (ICSCET),* pp. 1-4, 2018.

[12] H. F. Ng, "Pose-invariant face recognition security system," *Asian Journal of Health and Information Sciences,* vol. 1, pp. 101-111, 2006.

[13] A. Das, M. W. Ansari, R. Basak, "Covid-19 face mask detection using TensorFlow, Keras and OpenCV," *IEEE 17th India Council International Conference (INDICON),* pp. 1-5, 2020.

[14] S. Susanto, F. A. Putra, R. Analia, I. K. L. N. Suciningtyas, "The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam," *IEEE 3rd International Conference on Applied Engineering (ICAE)* pp. 1-5, 2020.

[15] Y. Said, "Pynq-YOLO-Net: An Embedded Quantized Convolutional Neural Network for Face Mask Detection in COVID-19 Pandemic Era," *International Journal of Advanced Computer Science and Applications,* vol. 11, pp. 100-106, 2020.

[16] S. Meivel, N. Sindhwani, R. Anand, D. Pandey, A. A. Alnuaim, A. S. Altheneyan, M. Y. Jabarulla, M. E. Lelisho, "Mask Detection and Social Distance Identification Using Internet of Things and Faster R-CNN Algorithm," *Computational Intelligence and Neuroscience, ID 2103975,* pp. 1-13, 2022.

[17] A. Daniel, A. Paul, A. Ahmad, S. Rho, "Cooperative Intelligence of Vehicles for Intelligent Transportation Systems (ITS)," *Wireless Pers Commun*, vol. 87, pp. 461-484, 2015.

[18] R. Engoulou, "Sécurisation des VANETs par la méthode de réputation des noeuds," *Thèse de doctorat. École Polytechnique de Montréal*, 2013.

[19] F. Z. Boukhalfa, "Deep learning approche pour Identifier/Detection Covid 19 Basse (IMAGE X-RAY)," *Mémoire de Master, Option : Système et multimédia, Universite Laarbi Tebessi, Tebessa*, 2021.

[20] R. Benbahria and A. F. Megri, "Prédiction Des Systèmes Par Apprentissage Automatique Dans Un Environnement Imprécis," *Mémoire de Master, Option : Systemes de Télécommunications, Université Mohamed Larbi Ben M'hidi, Oum el bouaghi*, 2021.

[21] A. Berra and E. ferhat, "Healthcare using deep learning Diabetic retinopathy detection using deep learning," *Mémoire de Master, Option :SIOD, Université Mohamed khider, Biskra*, 2020.

[22] M. Soltane and M. R. Laouar, "A Smart System to Detect Cheating in the Online Exam," *IEEE International Conference on Information Systems and Advanced Technologies (ICISAT)*, pp. 1-5, 2021.

[23] S. Houazene and S. Mansour, "Détection des scripts publicitaires a base d'apprentissage automatique profond," *Mémoire de Master, Option : Informatique, Universite Mouloud mammeri, Tizi-ouzou*, 2019.

[24] F. Abdi and A. Zenine, "Détection d'intrusion basée sur les méthodes d'ensemble de Machine Learning," *Mémoire de Master, Option : Informatique, Universite M'hamed Bougara. Boumerdès*, 2021.

[25] Y. Merabti, "Optimisation des réseaux de neurones MLP par l'algorithme hybride AG-RT pour le contrôle d'un système non linéaire," *Mémoire de Master, Option : Informatique Industrielle, Université Larbi Ben M'hidi, Oum El Bouaghi*, 2015.

[26] F. Abid and B. Boulares, "Détection de la rétinopathie diabétique avec le deep learning transfer learning,CNN,U-Net," *Mémoire de Master, Universite Saad Dahlab, Blida*, 2021.

[27] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, pp. 1-7, 2018.

[28] A. Oufar, "Diagnostic du cancer du sein-approches,techniques et applications: etude de cas : les "CAD": (computer aided diagnosis) basees sur la téchnique : deep learning," *Mémoire de Master, Option : Informatique, Universite Larbi Tébessi, Tébessa*, 2021.

[29] M. Wang, S. Lu, D. Zhu, "high-speed and low-complexity architecture for softmax function in deep learning," *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 223-226, 2018.

[30] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[31] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, A. Y. Ng, "Large Scale Distributed Deep Networks," *Advances in neural information processing systems*, vol. 25, pp. 1-9, 2012.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA*, 2015.

[33] K. He, X. Zhang, S. Ren, J. Sun, "Identity mappings in deep residual networks," *European conference on computer vision*, pp. 630-645, 2016.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, pp. 1-9, 2017.

[35] H. Mebarki and O. Ghedir, "Système de communication entre les véhicules (VNET) en utilisant la communication sans fil et les méthodes de l'intelligence artificielle," *Mémoire de Master, Option : Systemes des télécommunications, Universite Abbes Laghrour, Khenchela*, 2020.

[36] J. Y. Y. Zhang, Y. Chen, W. Yang, W. Zhang, Y. He, "Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application," *Computers and Electronics in Agriculture*, vol. 192, 2022.

[37] R. G.-C. A. García-Manso, C. J. García-Orellana, H. M. González-Velasco, M. Macías-Macías, "Towards selective and automatic harvesting of broccoli for agri-food industry," *Computers and Electronics in Agriculture*, vol. 188, pp. 1-8, 2021.

[38] A. Kurniawan, "IoT Projects with NVIDIA Jetson Nano: AI-Enabled Internet of Things Projects for Beginners," 2021.

[39] H. Y. B. Varshini, S. D. Pasha, M. Suhail, V. Madhumitha, A. Sasi, "IoT-Enabled smart doors for monitoring body temperature and face mask detection," *Global Transitions Proceedings*, vol. 2, pp. 246-254, 2021.

[40] R. Shrivastava and K. Sarawadekar, "Multi-Voltage GPIO Design and its Physical Implementation," *IEEE 4th International Conference for Convergence in Technology (I2CT)*, 2018

[41] T. W. S. R. Prasai, K. Mainali, H. Mathewson, H. Kafley, S. Thapa, D. Adhikari, P. Medley, J. Drake, " Application of Google earth engine python API and NAIP imagery for land use and land cover classification: A case study in Florida USA," *Ecological Informatics*, vol. 66, p. 101474, 2021.

[42] Q. Zhang, L. Xu, X. Zhang, B. Xu, "Quantifying the interpretation overhead of Python," *Science of Computer Programming*, vol. 215, p. 102759, 2022.

[43] J. Wang, L. Li, A. Zeller, "Better code, better sharing: on the need of analyzing jupyter notebooks," *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 2020.

[44] A. a. B. Teslyuk, S. Belyaev, A. Filippov, A. Izotov, K. Lyalin, I. Shitov, A. Yasnopolsky, V. Leonid Velikhov, "Architecture and deployment details of scalable Jupyter environment at Kurchatov Institute supercomputing centre," *Ivannikov Memorial IEEE Workshop (IVMEM)*, pp. 59-61, 2020.

[45] B. M. Randles, I. V. Pasquetto, M. S. Golshan, C. L. Borgman, "Using the Jupyter notebook as a tool for open science: An empirical study," *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 1-2, 2017.

[46] Z. Xu, X. Baojie, W. Guoxin, "Canny edge detection based on Open CV," *13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, pp. 53-56, 2017.

[47] S. A. Kumer, P. Kanakaraja, S. Areez, Y. Patnaik, P. T. Kumar, "An implementation of virtual white board using open CV for virtual classes," *Materials Today: Proceedings*, vol. 46, pp. 4031-4034, 2021.

[48] T. Carneiro, D. N. R. V. M., T. Nepomuceno, G. B. Bian, V. H. C. De Albuquerque, P. P. Reboucas Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, vol. 6, pp. 61677-61685, 2018.

[49] B. Prashanth, M. Mendu, R. Thallapalli, "Cloud based Machine learning with advanced predictive Analytics using Google Colaboratory," *Materials Today: Proceedings*, 2021.

[50] D. Faggella, "Artificial Intelligence and Security: Current Applications and Tomorrow's Potentials," *emerj*, 2019.

[51] R. Tadeusiewicz, "Introduction to intelligent systems," *The Electrical Engineering Handbook Series, Second edition, Boca Raton*, 2011.

[52] S. Guerfi, "Authentification d'individus par reconnaissance de caractéristiques biométriques liées aux visages 2D/3D," *Thèse de doctorat, Université d'Evry-Val d'Essonne*, 2008.

[53] C. Rahmad, R. A. Asmara, D. Putra, I. Dharma, H. Darmono, I. Muhiqqin, "Comparison of Viola-Jones Haar Cascade classifier and histogram of oriented gradients (HOG) for face detection," *IOP conference series: materials science and engineering*, p. 012038, 2020.

[54] H. Bandyopadhyay. (2022). *YOLO: Real-Time Object Detection Explained*. Available: https://www.v7labs.com/blog/yolo-object-detection

[55] Y. L. Y. Wang, X. Guo, L. Jiao, X. Liu, "CDANet: Common and Differential Attention Network for Object Detection and Instance Segmentation," *Pattern Recognition Letters*, vol. 158, pp. 48-54, 2022.

[56] J. Redmond, S. Divvala, R. Girshick, A. Farhadi, "Unified real-time object detection," *CoRR*, 2017.

[57] J. Y. Y. Zhang, Y. Chen, W. Yang, W. Zhang, Y. He, "Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application," *Computers and Electronics in Agriculture*, vol. 192, p. 106586, 2022.

[58] B. Setiyono, D. A. Amini, D. R. Sulistyaningrum, "Number plate recognition on vehicle using YOLO-Darknet," *Journal of Physics: Conference Series*, vol. 1821, p. 012049, 2021.

[59] L. LAC, "Méthodes de vision par ordinateur et d'apprentissage profond pour la localisation, le suivi et l'analyse de structure de plantes. Application au désherbage de précision," *Thèse de doctorat, Université de Bordeaux*, 2022.

[60] J. Liu and L. Liu, "Helmet Wearing Detection Based on YOLOv4-MT," *4th International Conference on Robotics, Control and Automation Engineering (RCAE)*, pp. 1-5, 2021.