

UNIVERSITE LIBRE DES PAYS DES GRANDS LACS
FACULTE DE SCIENCES ET TECHNOLOGIES
APPLIQUEES

DEPARTEMENT DE GENIE ELECTRIQUE ET INFORMATIQUE



BP. 368 GOMAs

www.ulpgl.net

CONCEPTION ET DÉVELOPPEMENT D'UN SYSTÈME DE
VIDÉOSURVEILLANCE INTELLIGENTE BASÉ SUR LA VISION
PAR ORDINATEUR POUR LE RENFORCEMENT DE LA
SÉCURITÉ DANS LES ZONES URBAINES ET RURALE : CAS DE
LA VILLE DE GOMA ET SES ENVIRONS

Par **KIKWAYA KASINDI Danny**

Travail présenté en vue de l'obtention du Diplôme de
Licence en Sciences et Technologies Appliquées

Option : Génie Electrique

Directeur : Prof. Dr. Ing. BARAKA MUSHAGE Olivier

Encadreur : Msc. Ing. KAMBALE WAMUHINDO
Abednego

ANNÉE ACADÉMIQUE 2024 - 2025

Epigraphe

« 80 % des effets proviennent de 20 % des causes. »

Vilfredo Pareto

Dédicace

A ma très chère tante Zawadi KIKWAYA

Remerciements

En premier lieu, je rends grâce à Dieu pour l'accompagnement constant qu'Il ne cesse de m'accorder depuis mon enfance et pour la force qu'Il m'a donnée tout au long de ce parcours.

Je tiens ensuite à exprimer ma profonde gratitude à mon directeur, le Professeur BARAKA MUSHAGE Olivier, ainsi qu'à mon encadreur, le MSc. Eng. KAMBALE WAMUHINDO Abdenego, pour leur encadrement, leur disponibilité et leurs précieux conseils qui ont grandement contribué à la réalisation de ce travail.

Mes remerciements les plus sincères s'adressent également à ma famille, en particulier à la famille KIKWAYA, pour son soutien moral, spirituel, financier et affectif, ainsi que pour les valeurs et l'encadrement qui ont favorisé ma croissance personnelle et académique.

Mes remerciements vont aussi au Réseau d'Innovation Organisationnel (RIO) ainsi qu'à Orange Ubora pour leur accompagnement et leur soutien tout au long de notre parcours académique.

Enfin, je remercie mes collègues et ami(e)s qui, de près ou de loin, ont contribué à la réalisation de ce travail. Je pense notamment à Seraphin, Alfred, Jeremy, Moïse, Wisdom, ainsi qu'à tous mes camarades de l'ULPGL, chacun ayant joué un rôle important dans l'aboutissement de ce projet.

Résumé

Ce mémoire porte sur l'apport de la vision par ordinateur au renforcement de la sécurité dans la ville de Goma, confrontée à des défis sécuritaires persistants. L'objectif principal a consisté à concevoir et développer un système de vidéosurveillance intelligente capable d'assister les acteurs de la sécurité dans l'identification automatique de situations potentiellement à risque à partir de flux vidéo. L'approche proposée combine les techniques d'intelligence artificielle et d'apprentissage profond afin d'analyser en temps réel les scènes capturées par des caméras installées dans des zones stratégiques de surveillance. Le système développé repose sur deux modules complémentaires : un module d'analyse permettant la détection d'objets assimilables à des armes ainsi que l'identification de scènes présentant des écarts par rapport aux comportements considérés comme normaux dans les données d'apprentissage, et un module de visualisation offrant un tableau de bord interactif pour le suivi des alertes et des indicateurs de sécurité. Les résultats obtenus démontrent la faisabilité technique de cette approche comme outil d'aide à la surveillance et à la prise de décision. Toutefois, les alertes générées constituent des indications nécessitantes, selon le contexte, une validation humaine, notamment pour l'interprétation des comportements observés et la distinction entre usages légitimes ou non des objets détectés. Cette étude met ainsi en évidence le potentiel des systèmes intelligents de vidéosurveillance pour renforcer la sécurité dans les zones couvertes par des dispositifs de captation vidéo.

Mots-clés :

Vision par ordinateur ; Intelligence artificielle ; Sécurité ; Python

Abstract

This thesis explores the contribution of computer vision to strengthening security in the city of Goma, which continues to face persistent security challenges. The main objective was to design and develop an intelligent video surveillance system capable of assisting security stakeholders in the automatic identification of potentially risky situations from video streams. The proposed approach combines artificial intelligence and deep learning techniques to analyze, in real time, scenes captured by cameras installed in strategic surveillance areas. The developed system is based on two complementary modules: an analysis module that enables the detection of objects resembling weapons as well as the identification of scenes that deviate from behaviors considered normal in the training data, and a visualization module that provides an interactive dashboard for monitoring alerts and key security indicators. The results obtained demonstrate the technical feasibility of this approach as a tool to support surveillance and decision-making processes. However, the alerts generated should be considered as indications that may require human validation depending on the context, particularly for interpreting observed behaviors and distinguishing between legitimate and illegitimate uses of detected objects. This study therefore highlights the potential of intelligent video surveillance systems to enhance security in areas covered by video monitoring devices.

Keywords:

Computer Vision; Artificial Intelligence; Security; Python.

Table des matières

0.	Introduction générale	1
0.1.	Contexte	1
0.2.	Identification et formulation du problème	1
0.3.	Questions de recherche	2
0.4.	Formulation des hypothèses.....	3
0.5.	Justification du choix du sujet et motivation	3
0.6.	Énoncé des objectifs de recherche	4
0.6.1.	Objectif général.....	4
0.6.2.	Objectifs spécifiques	4
0.7.	Méthodologie	4
0.8.	Subdivision du travail	4
Chapitre 1	Généralités sur la vision par ordinateur	6
1.1	Introduction.....	6
1.2	Présentation et définition	6
1.1.1.	La vision par ordinateur	6
1.1.2.	La vidéosurveillance	6
1.3.	Fondation du réseau des neurones convolutif.....	7
1.4	Fondements de l'intelligence artificielle appliquée à la vidéosurveillance	8
1.4.1	La convolution et son application en traitement d'images	8
1.4.2	Structure des réseaux des neurones.....	9
1.4.3	Transfert Learning	10
1.4.4	Techniques de régularisation et callbacks.....	11
1.5	Concept du réseau des neurones convolutif.....	12
1.5.1	Le deep Learning (Apprentissage approfondi)	13

1.6	Domaine d'application.....	15
1.7	Mode de vidéosurveillance	16
1.7.1	La vidéosurveillance analogique.....	16
1.7.2	La vidéosurveillance IP (numérique).....	17
1.7.3	Choix de l'utilisation de la caméra	18
1.8	Technologies, langages et outils de modélisation.....	19
1.8.1	Outils de développement pour la vision par ordinateur	20
1.9	Langages de programmation et Framework d'IA	21
1.10	Plateformes et matériels embarqués (Arduino, Raspberry Pi, Jetson Nano)	22
1.11	Outils de modélisation et simulation.....	22
1.12	Conclusion partielle	23
Chapitre 2	Conception du système de surveillance basé sur la vision par ordinateur.	24
2.1	Introduction.....	24
2.2	Architecture générale du système	24
2.3	Conception du système	26
2.4	Entraînement des modèles	28
2.4.1	Modél de classification	28
2.4.2	Modél de détection.....	30
2.4.3	Quelques difficultés rencontrées en utilisant l'environnement local	31
2.5	Présentation des résultats	33
2.5.1	La matrice de confusion normalisée issue des données des tests	33
2.5.2	Courbe Recall–Confidence	36
2.5.3	Courbe Precision–Recall (PR)	37
2.5.4	Courbe Precision–Confidence	37
2.5.5	Courbe F1–Confidence	38
2.5.6	Courbe d'apprentissage de l'autoencodeur	39

2.6	Conclusion partielle	40
Chapitre 3	Implémentation et Validation du Système de Surveillance	41
3.1.	Introduction.....	41
3.2	Schéma fonctionnel du système.....	41
3.3.	Configuration de l'environnement de l'expérience	43
3.3.1.	Configuration matérielle	43
3.4	Environnement de développement.....	46
3.4.1	Lancement de l'application.....	47
3.4.2	Interface du module de surveillance	47
3.4.3	Module de visualisation des activités	50
3.5	Conclusion partielle	52
Bibliographie.....		55
ANNEXES.....		58
	Code pour le module de visualisation.....	58

Liste des abréviations

CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HTTP	Hyper Text Transfer Protocol
IA	Intelligence Artificielle
LAN	Local Area Network
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
OpenCV	Open Computer Vision
ReLU	Rectified Linear Unit
SSD	Solid State Drive
UCSD	University of California San Diego
UML	Unified Modeling Language
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity
YOLO	You Only Look Once

Liste des figures

Figure 1 Diagramme schématique illustrant les interconnexions entre les couches dans le neocognitron , Kunihiro Fukushima [7]	7
Figure 2 Architecture du modèle de transfert Learning base sur un réseau de neurones convolutifs (CNN)	11
Figure 3 Exemple d'un réseau de neurones multicouche ayant 5 classes	14
Figure 4 configuration classique d'un système de vidéosurveillance analogique	17
Figure 5 Image d'une caméra digitale	18
Figure 6 Infographie médical du nerf optique	20
Figure 7 Schema bloc de l'architecture du système [28]	26
Figure 8 hyperparamètre du premier modèle	29
Figure 9 Code d'entrainement du modèle	30
Figure 10 Matrice de confusion normaliser pour les données de test.....	33
Figure 11 Figure illustrative des noms de classe du dataset	34
Figure 12 Graphique Recall confiance	36
Figure 13 Graphique précision-rappel	37
Figure 14 Graphique de la precision-confidence	37
Figure 15 Graphique F1 Confiance.....	38
Figure 16 Graphique du training curve de l'autoencodeur.....	39
Figure 17 Fonctionnement du système	41
Figure 18 pipeline du fonctionnement du système	44
Figure 19 Résultat de l'entrainement du modèle des données de test	45
Figure 20 Présentation de l'environnement anaconda.....	46
Figure 21 Interface du module de surveillance.....	48
Figure 22 Résultat de notre modèle pour la validation	50
Figure 23 Interface du module de visualisation	50

0. Introduction générale

0.1. Contexte

L'évolution technologique du 21^{ème} siècle a profondément transformé le système de traitement de l'information, en particulier grâce à l'essor de la vision par ordinateur, une branche de l'intelligence artificielle qui permet à une machine d'analyser, d'interpréter et parfois même de réagir à des images ou vidéos capturées dans le monde réel [1]. Plusieurs pays africains comme le Nigeria, l'Afrique du Sud ou l'Egypte intègrent déjà cette technologie dans des secteurs comme la sécurité, la sante ou le transport, avec des initiatives telles que les systèmes de reconnaissance faciale à Lagos, les cameras intelligentes à Johannesburg [2] ou les projets de vidéosurveillance urbaine au Caire. Néanmoins, dans de la ville de Goma, et notamment la République Démocratique du Congo (RDC), peinent encore à exploiter pleinement ces avancées technologiques. Dans un contexte marqué par l'insécurité persistante, le déplacement de la population, les cambriolages, les agressions ou les violences armées, la mise en place d'un système de surveillance intelligent capable de traiter en temps réel les flux vidéo représente une solution technologique prometteuse. Grace à la capacité des cameras connectées à des algorithmes de détection, il est désormais possible d'identifier des comportements suspects, d'émettre des alertes et d'assister les autorités dans la prise de décision. Le développement d'un système basé sur la vision par ordinateur vise donc à renforcer la sécurité dans les zones rurales et urbaines de la ville de Goma à travers une surveillance proactive, intelligent et contextuelle adaptée aux réalités locales.

0.2. Identification et formulation du problème

Depuis plus de trois décennies, la ville de Goma fait face à des problèmes persistants de tueries, viols, vols, pillages et massacres [3]. Malgré les efforts déployés par les services de sécurité gouvernementaux, l'insécurité demeure un défi majeur dans plusieurs zones, alimentée entre autres par le banditisme armé, les groupes terroristes, et parfois même par des éléments censés protéger la population.

Face à cette situation, ce travail propose une **approche technologique innovante**, centrée sur l'intégration de la vision par ordinateur comme outil d'aide à la sécurisation des populations, en particulier dans les cas de banditisme urbain et rural. Grâce à cette technologie, les forces de l'ordre pourront être informées plus rapidement de certaines situations potentiellement à risque, facilitant ainsi la prise de décision et l'organisation des interventions sur le terrain.

L'élément déclencheur de cette recherche a été l'observation des difficultés quotidiennes que rencontrent les habitants de la ville de Goma, notamment les cambriolages fréquents, les attaques de bandits armés souvent en moto, et l'impuissance de la population face à ces fléaux. Parfois, ces actes sont commis par des individus déguisés en membres des forces de l'ordre ou par des jeunes délinquants non identifiés.

Bien que le gouvernement tente de répondre à ces défis, il apparaît clairement que **les moyens traditionnels de sécurité sont insuffisants**. L'apport du numérique, notamment de la vision par ordinateur, représente une **solution stratégique et complémentaire** pour pallier ces limites et renforcer l'efficacité des interventions.

L'insécurité liée aux actes de violence, de vol, de viol et de pillage constitue un frein majeur au développement de la région. En effet, il est difficile pour un investisseur de s'implanter dans une zone où il existe un risque élevé de perte, faute de stabilité sécuritaire. Ainsi, proposer une **solution technologique basée sur la vision par ordinateur** apparaît comme une **réponse pertinente et moderne**, capable de réduire le taux d'insécurité.

0.3. Questions de recherche

Pour apporter une solution adaptée au problème de l'insécurité dans la ville de Goma, plusieurs questions de recherche se posent :

- Comment la vision par ordinateur peut-elle être exploitée pour détecter automatiquement des comportements suspects ou des intrusions dans un système de vidéosurveillance intelligente adapté aux environnements urbains et ruraux ?
- Dans quelle mesure l'intégration d'algorithmes d'intelligence artificielle permet-elle d'améliorer la précision, la rapidité et la fiabilité des systèmes de vidéosurveillance par rapport aux approches traditionnelles ?
- Quels sont les défis techniques et énergétiques liés au déploiement d'un système de vidéosurveillance intelligente en zones rurales, et comment une architecture optimisée (IoT, edge computing) peut-elle y répondre efficacement ?

0.4. Formulation des hypothèses

A partir de questions que nous venons de soulever, pour ce travail nous travaillerons sur les bases suivantes :

- La vision par ordinateur pourrait significativement contribuer à l'amélioration de la sécurité dans la ville de Goma en assurant une surveillance automatisée, rapide et précise des zones sensibles.
- Un système de vidéosurveillance intelligent, basé sur l'analyse d'images en temps réel, permettrait de détecter les comportements suspects, de générer des alertes automatiques, et ainsi de prévenir ou réduire les actes de banditisme, de violence et de pillage.
- Les données collectées par les caméras intelligentes pourraient être exploitées pour mieux comprendre les dynamiques sociales, les comportements individuels ou collectifs, et ainsi orienterait les prises de décision sécuritaires de manière plus efficace.

0.5. Justification du choix du sujet et motivation

Le choix de ce sujet découle d'un constat personnel et quotidien : l'insécurité grandissante dans la ville de Goma, notamment en RDC, constitue un véritable obstacle au développement socio-économique et à la stabilité sociale. En tant qu'étudiant passionné par les technologies émergentes et conscient des réalités locales, nous avons été motivés à explorer comment la vision par ordinateur, une branche innovante de l'intelligence artificielle, pourrait contribuer à améliorer la sécurité publique. Ce sujet s'inscrit également dans une dynamique scientifique actuelle, car de nombreux travaux internationaux s'intéressent déjà à l'application de l'IA dans les systèmes de surveillance intelligente. Toutefois, très peu d'études abordent ce thème sous l'angle des zones rurales et urbaines d'Afrique centrale, ce qui en fait un sujet à la fois original, pertinent et d'actualité. Sur le plan social, cette recherche vise à offrir des pistes concrètes aux décideurs politiques et aux acteurs de la sécurité pour moderniser les outils de surveillance, renforcer la protection des populations, et restaurer la confiance dans les institutions sécuritaires.

0.6. Énoncé des objectifs de recherche

0.6.1. Objectif général

Concevoir et développer un système de vidéosurveillance intelligente basé sur la vision par ordinateur, en vue de renforcer la sécurité dans la ville de Goma en RDC.

0.6.2. Objectifs spécifiques

Les objectifs spécifiques de ce travail consistent à :

- **Prétraiter et organiser** ces données pour les rendre exploitables par les algorithmes de vision par ordinateur ;
- **Entraîner** le modèle de vision par ordinateur capable de détecter des situations suspectes ou à risque ;
- **Intégrer** ce modèle d'IA dans un système embarqué fonctionnel incluant une ou plusieurs caméras ;
- **Tester et évaluer** les performances du système dans un environnement réel afin de vérifier son efficacité et sa fiabilité.

0.7. Méthodologie

Pour atteindre nos objectifs, nous avons utilisé une approche mixte. La méthode analytique a servi à étudier les systèmes existants de vidéosurveillance intelligente, afin d'en tirer des améliorations possibles. La technique documentaire nous a permis de consulter les travaux antérieurs sur le sujet. Enfin, la méthode expérimentale a été appliquée pour entraîner un modèle d'IA sous PyTorch, tester ses performances et l'intégrer dans une plateforme matérielle de démonstration.

0.8. Subdivision du travail

En dehors de l'introduction et de la conclusion générale, ce travail est structuré en trois chapitres principaux :

- Chapitre 1 : **Généralité sur la vision par ordinateurs**, ce chapitre présente les travaux antérieurs, les solutions existantes en matière de vidéosurveillance intelligente, ainsi que leur lien avec notre projet

- Chapitre 2 : **Conception du système de surveillance basé sur la vision par ordinateur**, Il est dédié à la modélisation du système, à la description de l'approche adoptée, et à la présentation des résultats issus de la phase de simulation.
- Chapitre 3 : **Réalisation, Implementation et Validation du Système de Surveillance**, Ce chapitre décrit le prototype final, ses composants, son fonctionnement, et les tests réalisés du système.

Chapitre 1

Généralités sur la vision par ordinateur

1.1 Introduction

La vidéosurveillance est, le plus souvent, implantée dans le cadre d'un programme de prévention de la criminalité ou de renforcement de la sécurité publique [5]. Dans ce chapitre, nous allons présenter des généralités sur les systèmes de vidéosurveillance en présentant le cadre théorique et conceptuel relatif aux systèmes de vidéosurveillance. Ce chapitre présente les définitions fondamentales, les techniques employées, les domaines d'application, les types d'équipements utilisés ainsi que les avantages et limites rencontrés. Cette analyse constitue le socle scientifique et technique de notre projet de conception d'un système de vidéosurveillance intelligente basé sur la vision par ordinateur.

1.2 Présentation et définition

1.1.1. La vision par ordinateur

La vision par ordinateur est une discipline rattachée à l'intelligence artificielle, dont l'objectif principal est le traitement et l'analyse automatique des flux d'images par des machines. Elle est considérée comme un domaine de l'intelligence artificielle (IA) qui permet aux ordinateurs de déduire des informations à partir d'images, de vidéos ou d'autres sources visuelles, car elle vise à doter les systèmes informatiques de la capacité de percevoir, d'interpréter et de comprendre visuellement leur environnement, à l'instar de la vision humaine. Elle utilise le machine Learning et le réseau de neurones pour apprendre la machine et le système de tirer les informations nécessaires dans les images, les vidéos ou tout autre donnée visuelle et faire la recommandation ou prendre une action [6].

1.1.2. La vidéosurveillance

La **vidéosurveillance** se définit comme un ensemble de techniques et de dispositifs permettant la capture, la transmission et l'analyse d'images dans le but de surveiller un espace défini. Elle peut être utilisée dans divers contextes : prévention du crime, contrôle d'accès, gestion du trafic routier, ou encore protection des infrastructures stratégiques.

Avec l'essor du numérique, la vidéosurveillance est passée de systèmes analogiques (caméras reliées à des magnétoscopes) à des systèmes **IP (Internet Protocol)** capables de transmettre les flux vidéo via des réseaux informatiques. Plus récemment, l'introduction de l'**intelligence artificielle (IA)** et de la **vision par ordinateur** permet de développer des systèmes dits de « surveillance intelligente », qui ne se contentent pas d'enregistrer mais sont capables de détecter automatiquement des anomalies, des comportements suspects ou des événements inhabituels [6].

1.3. Fondation du réseau des neurones convolutif

En 1959, les neurophysiologistes **David Hubel** et **Torsten Wiesel** ont mené une série d'expériences qui ont conduit à la publication de leur article intitulé « *Receptive fields of single neurons in cat's striate cortex* ». Dans ce travail, ils ont démontré que les neurones du cortex visuel des chats sont organisés en couches, et que ces couches apprennent à reconnaître des motifs visuels en procédant de manière hiérarchique : d'abord par l'extraction de caractéristiques locales, puis par la combinaison de ces informations afin d'obtenir une représentation de plus haut niveau. Cette découverte a posé les bases conceptuelles qui deviendront, plusieurs décennies plus tard, l'un des principes fondamentaux du **Deep Learning** et plus particulièrement des réseaux de neurones convolutifs [7].

Inspiré par cette découverte, **Kunihiko Fukushima** a proposé en 1980 le **Neocognitron**, un réseau de neurones autoorganisé à plusieurs couches, considéré comme le premier modèle théorique des **réseaux de neurones convolutifs (CNN)**. Comme le montre la Figure 1.

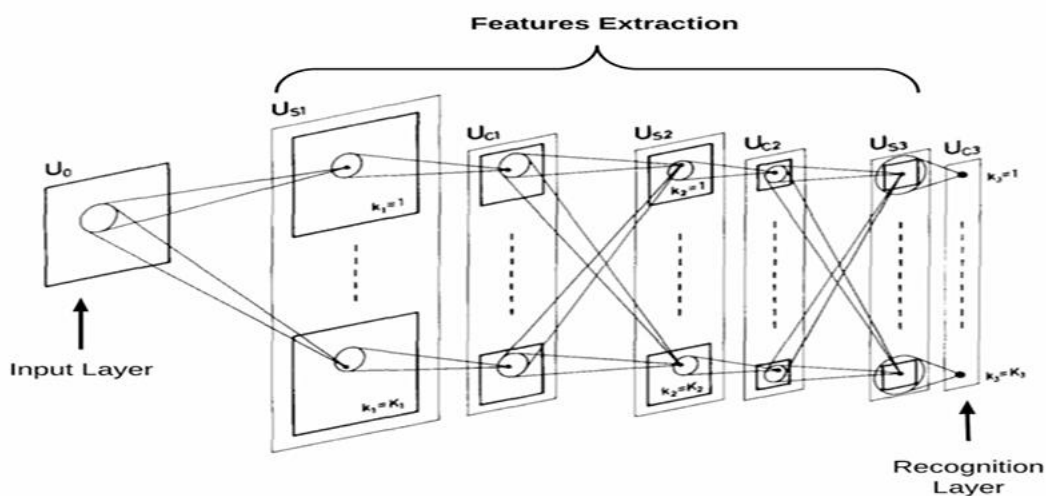


Figure 1 Diagramme schématisant les interconnexions entre les couches dans le neocognitron, Kunihiko Fukushima [7]

Une grande amélioration de l'architecture du Neocognitron a été faite par LeCun en 1989 en développant le framework modern du CNN, appelé LeNet-5, qui reconnaît avec succès les écrits à la main de la base des données MNIST. Après avoir découvert LeNet-5, à cause de plusieurs limitations comme le manque des données d'entraînement, manque d'innovation dans l'algorithme et l'inadéquation de la puissance de calcul, CNN n'a pas été performant face à divers problèmes complexes. Mais aujourd'hui, à l'ère du Big Data, nous disposons de vastes ensembles de données étiquetées. Des algorithmes plus innovants et des machines GPU particulièrement puissantes.

1.4 Fondements de l'intelligence artificielle appliquée à la vidéosurveillance

1.4.1 La convolution et son application en traitement d'images

La convolution est une opération mathématique qui consiste à appliquer une matrice appelée *noyau* (*kernel*) sur une matrice d'entrée, qui correspond dans notre cas à une image. Selon le choix du noyau, cette opération peut avoir différents objectifs tels que le floutage, l'amélioration de la netteté, la détection de contours ou le filtrage de certaines caractéristiques visuelles. Dans le cadre des réseaux de neurones convolutifs (CNN), la convolution est implémentée sous forme de couches convolutives dont le rôle est d'extraire automatiquement des caractéristiques locales pertinentes. Ces caractéristiques peuvent correspondre, par exemple, à la détection de la présence d'un objet, d'une silhouette ou d'un motif particulier. Progressivement, au fur et à mesure que les couches s'enchaînent, le réseau apprend à identifier des structures de plus en plus complexes, allant de simples contours à des formes entières.

Cette capacité d'extraction hiérarchique d'informations visuelles est particulièrement adaptée à un système de vidéosurveillance intelligente. En effet, elle permet d'analyser les images issues des caméras en mettant en évidence des éléments significatifs pour la sécurité, tels que la détection d'intrus ou d'objets suspects, tout en s'adaptant aux variations de luminosité, d'angles de vue et de contexte environnemental.

Formellement, la convolution discrète est définie par [8] :

$$Y[m,n] = \sum_{k=0}^K \sum_{l=0}^L H[k-l]X[m-k,n-l]$$

Ou :

- $Y[m, n]$: Valeur de la sortie (image filtrée) au pixel situé en position (m, n) .
- $X[m, n]$: Valeur du pixel de l'image d'entrée à la position (m, n) .
- $H[k, l]$: Valeur du noyau (ou kernel) à la position (k, l) . Le noyau est une petite matrice.
- k, l : Indices du noyau, qui parcourent la largeur et la hauteur du kernel.
- m, n : Indices du pixel dans l'image de sortie, qui correspondent aux positions où on applique le noyau.
- K, L : Taille du noyau, c'est-à-dire le nombre total de lignes (K) et de colonnes (L) sur lesquelles le noyau glisse.

1.4.2 Structure des réseaux des neurones

Les réseaux de neurones artificiels sont des modèles d'apprentissage inspirés du fonctionnement du cerveau humain, conçus pour apprendre des relations complexes entre des données d'entrée et des sorties. Ils sont constitués de neurones artificiels interconnectés, organisés en couches (*layers*). Chaque neurone applique une combinaison pondérée de ses entrées, suivie d'une fonction d'activation (*activation function*), permettant au modèle de capturer des relations non linéaires. Un réseau de neurones est généralement composé d'un input *layer*, de plusieurs *hidden layers* et d'un output *layer*, chacune ayant un rôle spécifique dans le processus d'apprentissage.

Dans le domaine de la vision par ordinateur, les *Convolutional Neural Networks (CNN)* sont largement utilisés en raison de leur capacité à extraire automatiquement des caractéristiques pertinentes à partir des images. Un CNN est principalement constitué de *convolutional layers*, qui permettent de détecter des motifs locaux tels que les contours, les textures et les formes, ainsi que de *pooling layers*, qui réduisent la dimension des données tout en conservant les informations essentielles. Les *fully connected layers*, généralement placées en fin d'architecture, assurent l'interprétation des caractéristiques extraites et produisent les sorties finales, notamment dans les tâches de classification.

L'apprentissage des réseaux de neurones profonds repose sur l'optimisation d'un grand nombre de paramètres, ce qui nécessite généralement des volumes importants de données annotées afin d'obtenir de bonnes performances et une capacité de généralisation satisfaisante. De plus, ce processus est coûteux en ressources computationnelles et requiert des temps d'entraînement élevés, en particulier lorsque le nombre de *hidden layers* est important. Ces contraintes

constituent un défi majeur dans de nombreuses applications pratiques, notamment lorsque les données disponibles sont limitées ou difficiles à collecter.

Face à ces limitations, le *transfer learning* apparaît comme une solution efficace permettant de tirer parti de modèles pré-entraînés sur de vastes ensembles de données. Cette approche consiste à réutiliser les couches initiales du réseau, qui ont déjà appris des représentations génériques, tout en adaptant les couches finales à la nouvelle tâche ciblée. Le *transfer learning* permet ainsi de réduire le besoin en données, le temps d'entraînement et les ressources nécessaires, tout en conservant des performances élevées. Toutefois, même avec l'utilisation du *transfer learning*, le risque de surapprentissage demeure présent. Dans ce contexte, les techniques de *regularization* jouent un rôle essentiel en contribuant à améliorer la capacité de généralisation du modèle et, par conséquent, ses performances sur des données non vues.

1.4.3 Transfert Learning

L'apprentissage par transfert (*Transfer Learning*) est une approche de l'apprentissage profond qui consiste à réutiliser les connaissances acquises par un modèle préalablement entraîné sur une large base de données, afin de les adapter à une nouvelle tâche spécifique. Plutôt que d'entraîner un réseau de neurones à partir de zéro ce qui nécessite un volume considérable de données et une puissance de calcul importante on exploite un modèle déjà entraîné (par exemple sur ImageNet) pour bénéficier de ses capacités d'extraction de caractéristiques générales. Concrètement, les premières couches d'un modèle pré-entraîné sont conservées car elles apprennent des représentations universelles, telles que les contours, les textures ou les formes de base, qui restent pertinentes quel que soit le contexte visuel. Les couches finales, en revanche, sont ajustées ou remplacées afin de spécialiser le modèle sur la tâche visée, comme la distinction entre un « intrus » et une situation « normale » dans le système de vidéosurveillance.

Cette stratégie présente plusieurs avantages : elle réduit considérablement le temps d'entraînement, limite les besoins en données d'apprentissage, et améliore les performances du modèle même lorsque le jeu de données disponible est restreint. Dans le cadre de ce projet, l'utilisation du Transfer Learning constitue donc un choix méthodologique pertinent, car elle permet de développer un système de surveillance intelligent performant et robuste, malgré des contraintes de ressources computationnelles et de données [9].

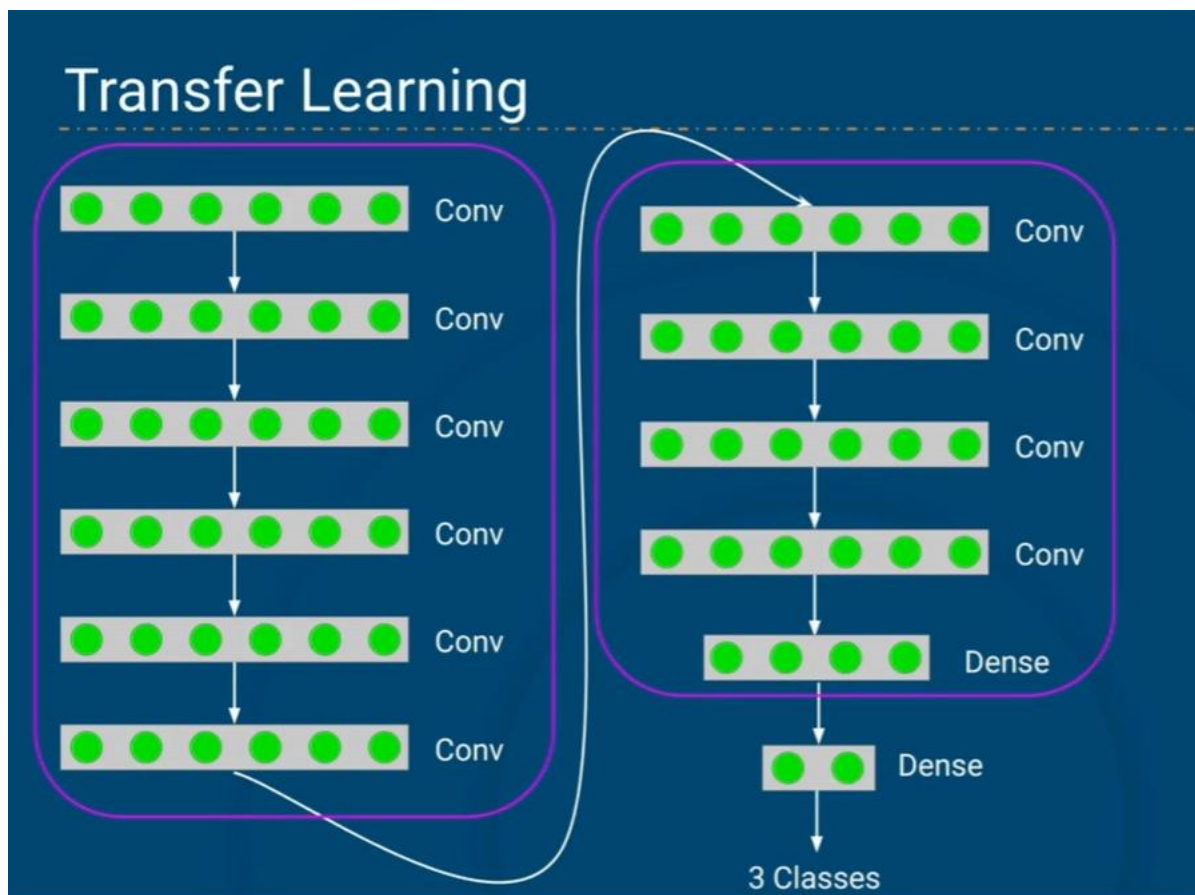


Figure 2 Architecture du modèle de transfert Learning basé sur un réseau de neurones convolutifs (CNN)

La Figure 2 présente l'architecture générale du modèle de transfert Learning utilisé dans ce travail. Le modèle repose sur un réseau de neurones convolutifs pré-entraîné, dont les couches convolutives sont conservées pour l'extraction de caractéristiques, tandis que les couches entièrement connectées sont adaptées à la nouvelle tâche de classification des classes.

1.4.4 Techniques de régularisation et callbacks

Dans l'apprentissage profond, un défi majeur est le risque de surapprentissage (*overfitting*), qui survient lorsque le modèle mémorise trop précisément les données d'entraînement et perd sa capacité à généraliser à de nouvelles données. Pour pallier ce problème, plusieurs techniques de régularisation sont utilisées afin d'améliorer la robustesse et la performance du réseau de neurones.

Parmi ces techniques, on peut citer [9]:

- **Dropout** : cette technique consiste à désactiver de manière aléatoire certains neurones du réseau lors de chaque itération d'entraînement. En empêchant le modèle de trop

dépendre de connexions spécifiques, le dropout favorise une meilleure généralisation et réduit le risque de surapprentissage

- **Early Stopping** : il s'agit d'un mécanisme qui interrompt automatiquement l'entraînement du modèle dès que la performance sur l'ensemble de validation commence à se dégrader, même si la performance sur l'ensemble d'entraînement continue à s'améliorer. Cela évite que le réseau ne s'ajuste excessivement aux données d'entraînement et assure une meilleure capacité de prédiction sur des données nouvelles.
- **Learning Rate Schedule** : cette méthode ajuste dynamiquement le taux d'apprentissage (*Learning rate*) au cours de l'entraînement. Un taux d'apprentissage trop élevé peut empêcher le modèle de converger, tandis qu'un taux trop faible ralentit considérablement l'apprentissage. L'utilisation d'un Schedule permet donc d'optimiser la vitesse de convergence tout en stabilisant la descente du gradient.
- **Checkpointing** : cette pratique consiste à sauvegarder périodiquement l'état du modèle pendant l'entraînement. En cas de problème (interruption du processus ou dégradation des performances), il est possible de revenir à une version antérieure offrant de meilleurs résultats. Cette approche permet de sécuriser le processus d'entraînement et de garantir la conservation des meilleures performances atteintes.

Dans le cadre de ce projet de vidéosurveillance intelligente, l'intégration de ces différentes techniques de régularisation vise à renforcer la fiabilité, la stabilité et la capacité de généralisation du modèle, afin qu'il reste performant face aux conditions variables de surveillance, telles que les changements de luminosité, d'angles de vue ou d'environnement [9].

1.5 Concept du réseau des neurones convolutif

Les **réseaux de neurones convolutifs (CNN ou ConvNet)** sont une extension des réseaux de neurones artificiels, conçus pour traiter efficacement les données visuelles. Leur architecture hiérarchique leur permet d'extraire automatiquement des caractéristiques à différents niveaux d'abstraction : les premières couches identifient des motifs simples (bords, couleurs, textures), tandis que les couches plus profondes reconnaissent des structures complexes (formes, objets, visages).

L'une des principales forces des CNN réside dans le **partage des poids**, qui réduit le nombre de paramètres à entraîner, limite le risque de surapprentissage et améliore la capacité de généralisation. De plus, les couches d'extraction et de classification sont apprises conjointement, ce qui rend les prédictions plus robustes et mieux adaptées aux caractéristiques extraites.

Aujourd'hui, les CNN se sont imposés comme une méthode incontournable dans la **vision par ordinateur** et trouvent des applications dans de nombreux domaines :

- Classification d'images (AlexNet, ResNet) : reconnaissance d'objets ou de scènes dans les flux vidéo. [10]
- Détection et suivi d'objets (YOLO, Faster R-CNN, SSD) : identification en temps réel de personnes, véhicules ou anomalies dans des environnements surveillés [11].
- Reconnaissance faciale et d'expressions (FaceNet, VGG-Face) : identification d'individus et analyse d'émotions dans le cadre de la sécurité [12].
- Sécurité et vidéosurveillance (YOLO, OpenPose) : analyse en continu des flux pour détecter des comportements suspects ou prévenir des incidents [13].
- Santé et secours d'urgence (U-Net, DeepLab) : détection d'anomalies ou de situations critiques (par exemple, chute de personnes, analyse biomédicale rapide en cas d'urgence) [13].
- Transport intelligent (YOLO, SSD) : reconnaissance de véhicules et de piétons pour la gestion de la circulation et la prévention d'accidents [11].

Ces multiples applications expliquent pourquoi les CNN représentent aujourd'hui l'une des approches les plus utilisées et les plus performantes dans les systèmes d'intelligence artificielle basés sur la vision

1.5.1 Le deep Learning (Apprentissage profondi)

« Le Deep Learning est un sous-domaine de Machine Learning qui utilise des réseaux neuronaux multicouches, appelés Deep neural network, pour imiter la capacité complexe de prise de décision du cerveau humain. Une forme du Deep Learning alimente aujourd'hui la plupart des applications d'intelligence artificielle (IA) présentes dans notre vie quotidienne. » [6]. La principale différence entre Le Deep Learning et le Machine Learning réside dans la structure de l'architecture du réseau neuronal sous-jacent. Les modèles d'apprentissage automatique "non profonds" et traditionnels utilisent des réseaux neuronaux simples avec une

ou deux couches de calcul. Les modèles d'apprentissage profond, eux, utilisent trois couches ou plus, mais généralement des centaines, voire des milliers de couches, pour entraîner les modèles.

Alors que les modèles de l'apprentissage supervisé nécessitent des données d'entrée structurées et étiquetées pour produire des résultats précis, les modèles de **Deep Learning** peuvent utiliser l'apprentissage non supervisé. Grâce à cette approche, les modèles de Deep Learning sont capables d'extraire par eux-mêmes les caractéristiques, les attributs et les relations nécessaires pour générer des résultats fiables à partir de données brutes et non structurées. De plus, ces modèles disposent de la capacité d'évaluer et de raffiner leurs propres prédictions afin d'améliorer progressivement leur précision.

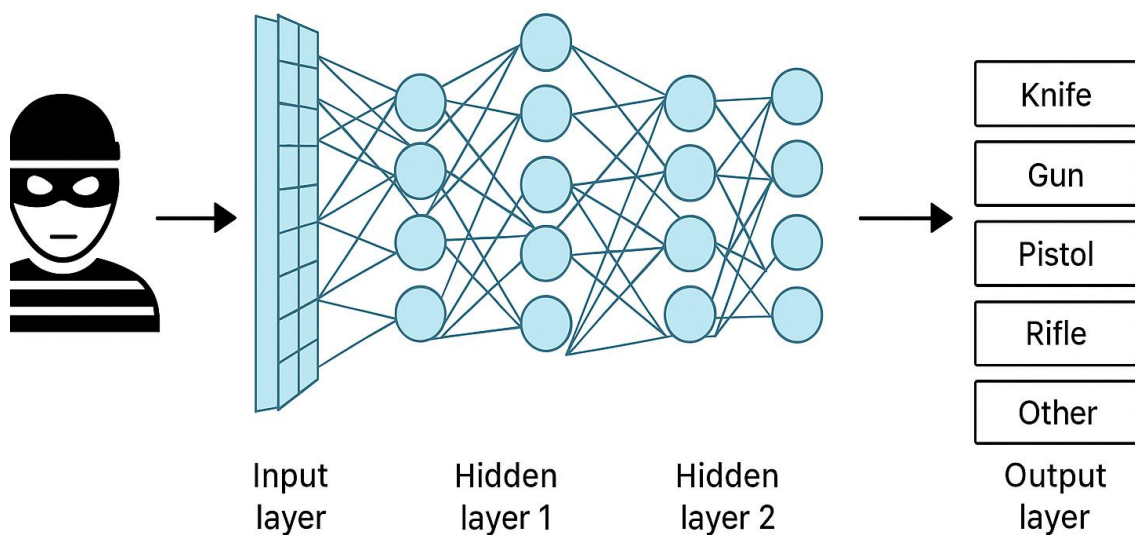


Figure 3 Exemple d'un réseau de neurones multicouche ayant 5 classes

L'image de la Figure 3 est un bon exemple pour comprendre la structure d'un réseau de neurones ainsi que le fonctionnement du *Deep Learning* dans le cadre du présent travail. À l'entrée, on retrouve une image que l'on souhaite classifier (ici une classification en 5 classes). Le réseau comporte 4 couches cachées, en plus de la couche d'entrée et de la couche de sortie. Cet exemple illustre clairement la technique du *Deep Learning*.

1.6 Domaine d'application

La vidéosurveillance, renforcée par l'intelligence artificielle et la vision par ordinateur, trouve aujourd'hui une vaste palette d'applications dans différents secteurs. Bien qu'elle soit principalement associée à la sécurité publique, son champ d'utilisation s'est considérablement élargi avec l'essor des technologies numériques et l'intégration de l'analytique intelligente.

Dans le cadre spécifique de ce travail, la vision par ordinateur est appliquée à la **sécurité dans la ville de Goma**. Cette région, marquée par des défis sécuritaires et sociaux, peut bénéficier des avancées de la surveillance intelligente afin de prévenir les menaces, de renforcer la confiance des populations et de soutenir le développement socio-économique. Toutefois, les domaines d'application de la vidéosurveillance sont beaucoup plus vastes et peuvent être regroupés comme suit :

- Sécurité publique et urbaine :

Ce domaine regroupe la surveillance des espaces publics (marchés, gares, aéroports, frontières), la prévention et dissuasion des actes criminels (vandalisme, vols, agressions), la gestion des manifestations et événements de masse, et l'identification et suivi de personnes recherchées grâce à la reconnaissance faciale.

- Protection des infrastructures critiques :

Ce domaine regroupe la surveillance des sites sensibles tels que centrales électriques, aéroports, ports, installations militaires ou bâtiments gouvernementaux, la détection d'intrusions ou d'activités suspectes dans les zones stratégiques.

- Gestion du trafic et des transports :

Ce domaine regroupe le contrôle en temps réel de la circulation routière, la détection automatique des infractions (excès de vitesse, non-respect des feux), le suivi des flux de véhicules et optimisation de la mobilité urbaine et la surveillance ferroviaire et aéroportuaire pour garantir la sécurité des passagers.

- Secteur privé et entreprises :

Ce domaine regroupe la sécurisation des locaux commerciaux, industriels et bancaires, la suivie des activités dans les usines pour prévenir les accidents et optimiser la productivité et la protection contre le vol interne ou externe

- Secteur résidentielle

Ce domaine regroupe la protection des domiciles grâce aux cameras connectées, la contrôle d'accès aux habitations et suivi à distance via smartphone

Au fait le domaine d'application de la vidéosurveillance est vaste mais on peut retenir ceux cités ci-haut.

Il existe d'autres raisons moins connues du grand public. Par exemple, la vidéosurveillance améliore la gestion des incidents et permet une intervention plus rapide et efficace. Elle peut être utile dans la prévention du suicide ou lors d'accidents sur la voie publique, contribuant indirectement à maintenir les primes d'assurance à un niveau raisonnable. De plus, la surveillance des axes routiers permet d'informer les automobilistes en temps réel sur les conditions de circulation. Les enregistrements fournis par les caméras de surveillance ont également permis de résoudre certaines affaires criminelles. Par exemple, après les attentats du métro de Londres en juillet 2005, les enregistrements ont été utilisés pour identifier les poseurs de bombes, bien qu'ils n'aient pas été considérés comme indispensables. Cependant, les études indépendantes menées en France et à l'étranger n'ont pas pu démontrer de manière concluante si la vidéosurveillance prévient ou réduit les crimes. Le gouvernement britannique a lui-même estimé qu'il était difficile d'évaluer les effets bénéfiques, bien que Scotland Yard ait affirmé en 2008 que les 500 000 caméras de vidéosurveillance à Londres n'avaient permis d'élucider que 3 % des vols dans la rue [14].

1.7 Mode de vidéosurveillance

On distingue principalement **deux grands modes de vidéosurveillance** : la vidéosurveillance **analogique**, plus traditionnelle, et la vidéosurveillance **IP (numérique)**, qui représente une évolution moderne des systèmes de sécurité.

1.7.1 La vidéosurveillance analogique

La vidéosurveillance analogique repose sur des caméras analogiques dont la fonction est de **capturer les images et de les transmettre à un enregistreur** (souvent un magnétoscope ou un DVR basique) via un câble coaxial. Le système est généralement composé de caméras, d'un moniteur, d'un enregistreur et du câblage nécessaire. Telle que le montre la Figure 4.

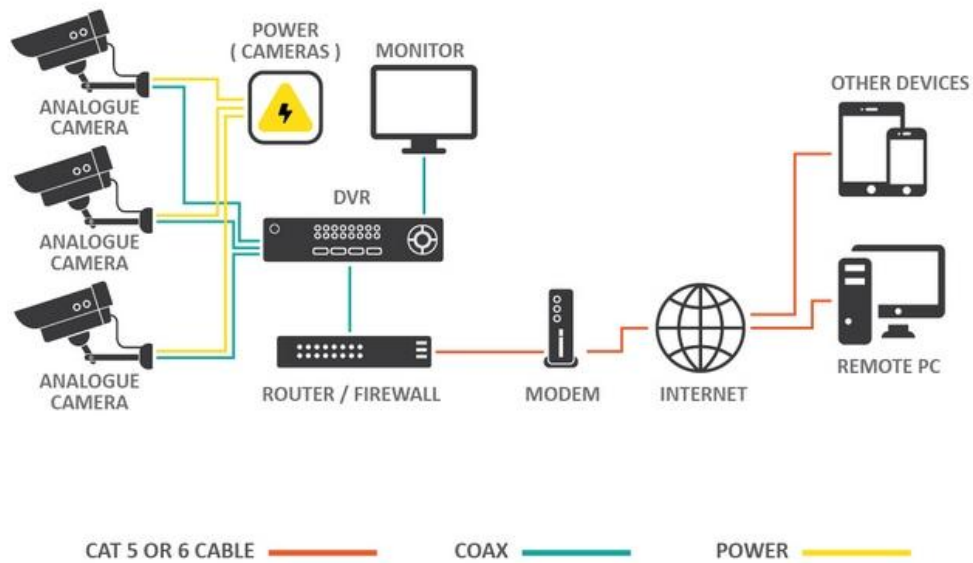


Figure 4 configuration classique d'un système de vidéosurveillance analogique

Les principales caractéristiques des caméras de sécurité analogiques incluent une résolution d'image plus faible et un processus d'installation plus simple. Elles fonctionnent avec des résolutions en définition standard, généralement comprises entre 480p et 720p. Les caméras analogiques capturent des signaux vidéo continus sous forme de fluctuations de tension transmises par des câbles coaxiaux, qui sont ensuite traités par le DVR. La simplicité de cette technologie la rend robuste et fiable pour les besoins de surveillance de base [15].

1.7.2 La vidéosurveillance IP (numérique)

La vidéosurveillance IP constitue une évolution de l'analogique. Elle repose sur des **caméras connectées à un réseau informatique** (LAN, Internet, Wi-Fi, VPN, etc.) et reliées à un serveur central. Les images sont capturées, transmises via le réseau et stockées sur des disques durs grâce à un logiciel de gestion vidéo.



Figure 5 Image d'une caméra digitale

Le passage d'un signal analogique à un signal numérique nécessite une étape d'échantillonnage. Pour cela, on utilise un procédé mathématique consistant à multiplier le signal analogique par un peigne de Dirac [8].

1.7.2.1 Avantage du camera digital

Les caméras IP numériques offrent une qualité d'image supérieure, essentielle pour capturer avec précision les détails dans les enregistrements de surveillance. Leur capacité à s'intégrer facilement à d'autres systèmes numériques via des réseaux sans fil réduit la complexité et les coûts liés aux infrastructures de câblage. De plus, la scalabilité des caméras numériques permet une extension aisée des systèmes de surveillance, puisqu'elles peuvent être ajoutées sans difficulté aux réseaux existants, sans provoquer de perturbations majeures [15].

1.7.2.2 Inconvénient du camera digital

Cependant, les caméras IP numériques ne sont pas exemptes de limites. Elles ont généralement un coût plus élevé, tant pour l'installation initiale que pour la maintenance, en raison de la technologie avancée qu'elles intègrent. De plus, comme elles dépendent fortement des connexions réseau, elles peuvent être vulnérables à des problèmes tels que la latence ou les menaces de cybersécurité. Cela nécessite la mise en place de mesures de cybersécurité robustes, ce qui peut engendrer des coûts et des complexités supplémentaires [16].

1.7.3 Choix de l'utilisation de la caméra

Le choix de l'utilisation de la caméra repose sur plusieurs critères telle que :

0. Le cout :

Les caméras analogiques sont moins chères à l'achat, tandis que les caméras IP nécessitent un investissement initial plus élevé mais offrent des économies à long terme.

1. La qualité d'image :

Les caméras IP offrent une résolution supérieure, adaptée aux besoins d'identification précise, contrairement aux analogiques limitées en définition.

2. La scalabilité :

Les systèmes numériques sont facilement extensibles, permettant d'ajouter de nouvelles caméras sans modifier lourdement l'infrastructure.

3. Les infrastructures existantes :

Les entreprises disposant déjà d'un système analogique peuvent opter pour une solution hybride afin de réduire les coûts de transition.

4. L'intégration avec d'autres systèmes :

Les caméras IP s'intègrent plus efficacement avec d'autres systèmes de sécurité modernes (alarmes, contrôle d'accès, etc.).

5. L'installation et la maintenance :

Les caméras IP sont généralement plus simples à installer et à entretenir que les caméras analogiques.

1.8 Technologies, langages et outils de modélisation

La mise en œuvre d'un système de vision par ordinateur pour la surveillance intelligente repose sur un ensemble cohérent de technologies, de langages de programmation et d'outils matériels et logiciels. Ces composants assurent la capture, le traitement et l'interprétation des données visuelles en temps réel. C'est la capacité de reproduire les capacités de vision humaine dans des machines. Cependant, alors que les humains utilisent des « rétines, des nerfs optiques et des parties dédiées de leur cerveau » selon la Figure 6, pour collecter et traiter les informations visuelles.

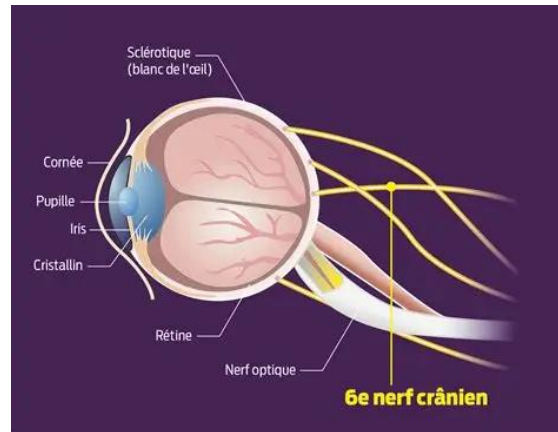


Figure 6 Infographie médicale du nerf optique

Ce processus est complètement différent chez les machines. Pour apprendre aux machines à voir, nous nous appuyons sur une série des composants technologiques, notamment :

- **Capteurs** : Peuvent être des caméras ou tout autre appareils spécifiques pouvant capturer les données visuelles ;
- **Données** : Ce sont les éléments de bases dans l'apprentissage de la machine car c'est sur base des données que la machine permet de prendre une décision. Ces données sont en général les images ou les vidéo (pour le cas de la vision par ordinateur) ;
- **L'algorithme** : Est l'étape de l'analyse des données, pour ce faire on nettoie et prépare les données d'image, notamment le filtrage, le redimensionnement ou la normalisation de l'image

Mais qu'à cela ne tienne la technologie de la vision par ordinateur utilisent plusieurs outils qui seront énumérer dans la section suivante.

1.8.1 Outils de développement pour la vision par ordinateur

Les outils de développement constituent la base du prototypage et de la mise en œuvre des solutions de vision par ordinateur. Parmi les plus utilisés, on retrouve :

Algorithme de Viola-Jones [17]

C'est l'un des premiers algorithmes performants pour la détection faciale. Il repose sur l'utilisation de *features de Haar* et d'un classificateur en cascade, garantissant rapidité et efficacité. Bien qu'aujourd'hui supplanté par des réseaux neuronaux conventionnels (CNN), il reste une base pédagogique et une référence dans l'histoire de la vision par ordinateur, en particulier pour l'initiation aux techniques de détection d'objets

Open CV (Open -Source Computer Vision Library) [18]

Bibliothèque open source très répandue, offrant des fonctions pour le traitement d'images, la détection d'objets, le suivi de mouvement et l'analyse vidéo.

MATLAB avec Computer Vision Toolbox [19]

Souvent utilisé dans la recherche et l'enseignement, MATLAB permet de prototyper rapidement des algorithmes de traitement d'images et d'apprentissage machine

MediaPipe (Google) [20]

Framework moderne pour la détection et le suivi en temps réel (visage, gestes, pose humaine), conçu pour les environnements mobiles et embarqués.

1.9 Langages de programmation et Framework d'IA

- Afin de réaliser ce travail, nous avons opté pour les technologies listées ci-dessous, dont la plupart reposent sur le langage Python. Ce choix se justifie par sa simplicité d'utilisation, notamment grâce à une syntaxe claire et proche du langage courant, facilitant ainsi la compréhension et le développement. Ainsi, nous avons retenu les éléments suivants : **Python** : langage dominant dans le domaine en vision par ordinateur grâce à sa simplicité et son riche écosystème. Les bibliothèques (OpenCV, **NumPy**, **Pandas**, **scikit-learn**). Il permet l'implémentation de modèles d'apprentissage automatique. [21]
- **PyTorch** : Framework très utilisé dans la recherche pour l'entraînement et l'évaluation de modèles de Deep Learning. Sa flexibilité en fait un outil privilégié pour le prototypage [22].
- **TensorFlow et Keras** : utilisés pour l'optimisation et le déploiement de modèles d'IA, notamment sur appareils mobiles via (TensorFlow Lite) [23].
- **C++** : recommandé pour les environnements embarqués nécessitant un traitement temps réel avec des contraintes de latence minimale. Il est souvent utilisé conjointement avec OpenCV. [24]

1.10 Plateformes et matériels embarqués (Arduino, Raspberry Pi, Jetson Nano)

Pour l'implémentation dans un contexte de surveillance intelligente, les composants matériels jouent un rôle central :

- **Arduino** : ce microcontrôleur sera utilisé pour contrôler le servomoteur, permettant d'orienter la caméra selon différents angles afin de capter les flux vidéo et images. Il jouera un rôle central dans la gestion des périphériques (moteurs, capteurs, alarmes) au sein de l'architecture de vidéosurveillance [25].
- **Raspberry Pi** : bien qu'il représente une solution économique pour exécuter localement des algorithmes de traitement vidéo, son coût constitue une contrainte. Ainsi, dans mon travail, je privilégierai l'utilisation d'un ordinateur portable pour le traitement et l'exécution des algorithmes de reconnaissance et d'analyse vidéo [26]
- **Caméras compatibles** :
 - **Caméras USB** : elles seront utilisées dans les phases de prototypage et de test, en raison de leur simplicité d'intégration [27].
 - **Caméras IP** : elles seront envisagées pour une extension future du projet, notamment dans des installations professionnelles, grâce à leur flexibilité et leurs fonctionnalités avancées

1.11 Outils de modélisation et simulation

Parmi les outils de modélisation on peut citer :

- **UML (Unified Modeling Language)** : pour modéliser l'architecture logicielle et les interactions entre modules.
- **Simulink (MATLAB)** : pour simuler des flux vidéo et tester des algorithmes de traitement.

Proteus et Fritzing : pour modéliser la partie matérielle (schémas électroniques et câblages).

1.12 Conclusion partielle

Ce premier chapitre a permis de présenter les bases théoriques et techniques de la vidéosurveillance, en retraçant son évolution des systèmes analogiques aux solutions numériques et IP, ainsi que ses domaines d'application, ses avantages et ses limites. Nous avons également introduit les fondements de l'intelligence artificielle, en particulier la convolution, les réseaux neuronaux convolutionnels, le transfert Learning et les mécanismes d'optimisation. Ces éléments constituent la passerelle vers le chapitre suivant.

Chapitre 2

Conception du système de surveillance basé sur la vision par ordinateur

2.1 Introduction

Dans ce chapitre, nous présentons la conception et la simulation du système de vidéosurveillance intelligente développé dans le cadre de ce mémoire. Ce travail s'inscrit dans la continuité du chapitre précédent, consacré aux fondements théoriques de la vision par ordinateur et aux systèmes de vidéosurveillance intelligents. L'objectif principal est de concevoir et d'implémenter un système capable de détecter automatiquement des situations anormales ainsi que des objets dangereux à partir de flux vidéo en temps réel. Pour atteindre cet objectif, nous exploitons des techniques avancées d'apprentissage profond (Deep Learning), notamment des modèles de détection d'objets et de classification d'anomalies.

La réalisation du système repose sur :

- L'entraînement de modèles de détection d'armes ;
- L'entraînement d'un modèle de détection d'anomalies comportementales ;
- L'intégration de ces modèles dans une application de surveillance interactive ;
- L'adaptation du système à des environnements locaux et embarqués.

Ce chapitre détaille l'architecture générale du système, les choix technologiques effectués, ainsi que les différentes étapes de simulation et de validation.

2.2 Architecture générale du système

Le système proposé repose sur une architecture modulaire composée de plusieurs modules complémentaires :

❖ Module d'acquisition vidéo :

Permet la capture des flux vidéo à partir d'une webcam.

❖ **Module de prétraitement :**

Assure le redimensionnement des images, la normalisation et la préparation des données avant leur traitement par les modèles d'intelligence artificielle.

❖ **Module d'analyse par intelligence artificielle :**

Ce module constitue le cœur du système. Il intègre :

- Un modèle de détection d'armes basé sur l'apprentissage profond ;
- Un modèle de détection d'anomalies comportementales ;
- Un mécanisme de prise de décision combinant les résultats des différents modèles.

❖ **Module d'affichage et d'alerte :**

Ce module est responsable de :

- L'affichage des résultats en temps réel (boîtes englobantes, scores de confiance) ;
- La classification de l'état du système (normal, anomalie, critique) ;
- L'émission d'alertes visuelles.

L'architecture logicielle a été conçue de manière modulaire afin de :

- Faciliter les tests et les améliorations futures ;
- Permettre l'intégration de nouveaux modèles ;
- Assurer la compatibilité avec différents environnements matériels.

Cette modularité permet d'adapter le système aussi bien :

- À un environnement local (webcam d'un ordinateur),
- Ou encore à une caméra IP dans une architecture cloud.

La Figure 7 permet de donner le schéma bloc de l'architecture du système

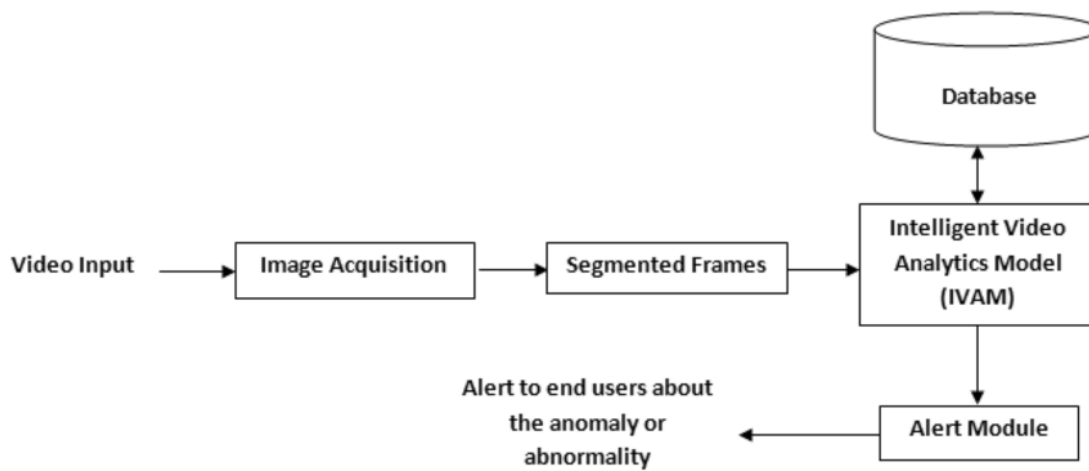


Figure 7 Schema bloc de l'architecture du système [28]

2.3 Conception du système

La conception du système repose sur l'architecture modulaire présentée dans la section précédente. Chaque module a été développé afin d'assurer une chaîne complète de traitement allant de l'acquisition du flux vidéo jusqu'à l'analyse intelligente et la génération d'alertes. Cette approche modulaire permet d'améliorer la lisibilité du système, de faciliter les tests et de permettre l'intégration future de nouveaux modèles.

Le premier élément du système est le **module d'acquisition vidéo**, chargé de capturer les images provenant de la caméra. Dans ce travail, l'acquisition des flux vidéo est réalisée à l'aide de la bibliothèque OpenCV, largement utilisée dans le domaine de la vision par ordinateur. Cette bibliothèque permet d'accéder facilement à une webcam ou à une caméra connectée et d'obtenir un flux d'images en temps réel. Les images capturées sont ensuite transmises aux modules suivants pour être analysées. Le choix d'OpenCV se justifie par sa compatibilité avec Python et par la richesse de ses fonctions dédiées au traitement d'images et de vidéos.

Une fois les images capturées, elles passent par un **module de prétraitement**. Ce module a pour objectif de préparer les images avant leur analyse par les modèles d'intelligence artificielle. Les opérations principales consistent à redimensionner les images, normaliser les données et adapter le format des images aux exigences des modèles de détection. Ces transformations permettent d'optimiser la performance des algorithmes et d'assurer une meilleure cohérence dans le traitement des données.

Le cœur du système est constitué par le **module d'analyse basé sur l'intelligence artificielle**. Ce module regroupe plusieurs modèles d'apprentissage profond permettant d'analyser le contenu des images capturées. L'environnement de développement utilisé pour cette étape est Python, avec la distribution Anaconda pour la gestion des bibliothèques et des environnements virtuels. Les phases de prototypage et de test ont été réalisées avec Jupyter Notebook, tandis que l'entraînement des modèles a été effectué sur Google Colab afin de bénéficier de ressources GPU adaptées à l'apprentissage des réseaux neuronaux.

Plusieurs modèles ont été intégrés dans ce module afin d'améliorer la capacité d'analyse du système. Un premier modèle a été développé pour la **détection d'armes à feu**, notamment les pistolets, les kalachnikovs et d'autres armes similaires. Ce modèle permet d'identifier automatiquement la présence d'armes dans le champ de la caméra.

Afin de mieux comprendre le contexte de la scène, un **modèle de détection des humains** a également été intégré. Ce modèle repose sur l'architecture YOLO pré-entraînée sur la base de données COCO, qui permet de reconnaître la classe *person*. La détection des personnes dans la scène est importante car elle permet de relier la présence d'objets dangereux à une activité humaine, ce qui améliore la pertinence de l'analyse.

En complément, un **modèle de détection des couteaux** basé sur YOLO pré-entraîné a été ajouté au système. Ce modèle permet d'identifier la présence d'objets tranchants susceptibles de représenter un danger. L'association entre la détection d'un couteau et la présence d'une personne permet d'évaluer plus précisément le niveau de risque. Le système intègre également un **modèle de détection d'anomalies comportementales**, dont l'objectif est d'identifier des situations inhabituelles ou suspectes dans l'environnement surveillé. Ce modèle analyse les caractéristiques de la scène afin de détecter des comportements qui s'écartent d'une situation normale. Les résultats produits par l'ensemble de ces modèles sont ensuite transmis à un **module de décision et d'alerte**. Ce module combine les informations issues des différentes détections afin de déterminer l'état de la scène observée. Trois niveaux d'état sont définis : normal, anomalie et critique. L'état critique est déclenché lorsqu'une arme ou un couteau est détecté, ou lorsqu'une anomalie comportementale est identifiée en présence d'une personne. Dans ce cas, le système génère une alerte visuelle permettant d'attirer l'attention de l'utilisateur.

Cette approche basée sur la combinaison de plusieurs modèles d'intelligence artificielle permet d'améliorer la robustesse et la fiabilité du système de surveillance, tout en offrant une meilleure capacité d'analyse des situations potentiellement dangereuses.

2.4 Entraînement des modèles

Pour notre cas nous avons utilisé deux dataset (mis à part des modèle fine-tuner), un de classification et l'autre de détection.

2.4.1 Modèl de classification

Pour le modèle de classification nous avons utilisé le dataset « Dataset UCSD Anomaly Détection » téléchargé depuis kaggle¹. Cette base des données est composée des séquences vidéo capturées par une caméra fixe surveillant une zone piétonne. Les anomalies présentes dans les scènes incluent notamment : des véhicules (vélos, voitures), des objets non autorisés ainsi que les comportements anormaux par rapport aux déplacements piétons.

Ces vidéos sont ensuite transformées en image pour l'entraînement du modèle dont nous avons 26600 images de 133 vidéos ou chaque vidéo est composer de 200 images illustrant la suite logique de la vidéo. Avant de passer à l'entraînement de ce modèle nous avons reparti ces données en *train*, *test* et *val*. Tel que :

- Pour le d'entraînement(train) on a pris 70% des images
- Pour les données de test on a pris 25%
- Pour les données de validation on a pris 5%

NB : Il est également important de noter que le dataset de ce modèle ne contient qu'une seule classe, celle des anomalies. C'est pourquoi l'utilisation d'un autoencodeur est particulièrement appropriée, car ce type de modèle permet d'apprendre les caractéristiques des scènes normales et de détecter les écarts significatifs lors de l'apparition d'une anomalie.

¹ [Lien vers base des données USCD](#)

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 128, 128, 1)	0
conv2d (Conv2D)	(None, 128, 128, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 64)	36,928
up_sampling2d (UpSampling2D)	(None, 64, 64, 64)	0
conv2d_3 (Conv2D)	(None, 64, 64, 32)	18,464
up_sampling2d_1 (UpSampling2D)	(None, 128, 128, 32)	0
conv2d_4 (Conv2D)	(None, 128, 128, 1)	289

Total params: 74,497 (291.00 KB)
Trainable params: 74,497 (291.00 KB)
Non-trainable params: 0 (0.00 B)

Figure 8 hyperparamètre du premier modèle

Comme on peut le voir sur la Figure 8, le modèle adopté est un autoencodeur convolutionnel composé de plusieurs couches Conv2D, MaxPooling2D et UpSampling2D, organisées en une architecture encodeur-décodeur. L'entrée du réseau est une image en niveaux de gris de dimension $128 \times 128 \times 1$. La partie encodeur comprend deux couches convolutionnelles utilisant respectivement 32 et 64 filtres (3×3) avec la fonction d'activation ReLU, suivies de couches de max pooling (2×2) permettant la réduction progressive des dimensions spatiales. Le décodeur reconstruit ensuite l'image à l'aide de couches convolutionnelles et d'opérations d'upsampling, avec une couche de sortie utilisant la fonction d'activation Sigmoid, adaptée à une normalisation des pixels entre 0 et 1. Comme indiqué dans le résumé du modèle (Figure 9), le réseau comporte au total 74 497 paramètres entraînaables, optimisés à l'aide de l'algorithme Adam, et la performance de reconstruction est évaluée par la fonction de perte Mean Squared Error (MSE). Cette configuration permet au modèle d'apprendre efficacement la représentation des images normales afin d'identifier les écarts significatifs lors de la phase de test.

2.4.2 Modél de détection

Pour le modèle de détection, nous avons utilisé la base de données « **Weapon Detection** » téléchargée depuis la plateforme Roboflow². Cette base de données est constituée d'images représentant des personnes en possession d'armes à feu dans différents contextes (intérieur, extérieur, vidéosurveillance, etc.).

Le dataset comprend au total **40 829 images**, réparties comme suit :

- Train : 28580 images
- Val : 8166 images
- Test : 4083 images

Avant la phase de prédiction, il est indispensable d'annoter les images afin de permettre l'apprentissage supervisé du modèle d'intelligence artificielle. Dans notre cas, les annotations (boîtes englobantes) ont été générées et vérifiées à l'aide de l'outil Roboflow, puis exploitées pour l'entraînement du modèle.

Pour pouvoir entraîner le modèle nous avons dû ajuster plusieurs hyperparamètres en utilisant ce code :

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
    data="/content/dataset/data.yaml",
    epochs=25,
    imgsz=640,
    batch=16,
    optimizer="AdamW",
    lr0=0.001,
    weight_decay=0.0005,
    patience=10,
    augment=True,
    device=0
)
```

Figure 9 Code d'entraînement du modèle

² Lien vers la base des données : <https://universe.roboflow.com/ir-uzlqh/weapon-detection-jqd3x>

Ainsi, comme le montre la Figure 9, l'entraînement du modèle de détection d'objets a été réalisé en utilisant la bibliothèque Ultralytics, basée sur le framework PyTorch. Le modèle pré-entraîné YOLOv8n (yolov8n.pt) a été chargé comme point de départ afin de bénéficier du transfert d'apprentissage (transfer learning), ce qui permet d'accélérer la convergence et d'améliorer les performances.

Le fichier data.yaml spécifie la structure du dataset, incluant les chemins vers les images d'entraînement, de validation et de test ainsi que les classes à détecter.

Les principaux hyperparamètres utilisés lors de l'entraînement sont les suivants :

- **epochs = 25** : le modèle parcourt l'ensemble du dataset 25 fois afin d'ajuster progressivement ses poids.
- **imgsz = 640** : les images sont redimensionnées en 640×640 pixels pour standardiser l'entrée du réseau.
- **batch = 16** : 16 images sont traitées simultanément à chaque itération.
- **optimizer = "AdamW"** : algorithme d'optimisation utilisé pour la mise à jour des poids du réseau. AdamW permet une convergence rapide et une meilleure régularisation.
- **lr0 = 0.001** : taux d'apprentissage initial contrôlant la vitesse d'ajustement des poids.
- **weight_decay = 0.0005** : terme de régularisation L2 visant à limiter le surapprentissage.
- **patience = 10** : mécanisme d'early stopping arrêtant l'entraînement si aucune amélioration n'est observée après 10 epochs.
- **augment = True** : activation de la data augmentation pour améliorer la capacité de généralisation du modèle.
- **device = 0** : utilisation du GPU pour accélérer l'entraînement.

Ce choix d'hyperparamètres représente un compromis entre performance, stabilité d'apprentissage et temps de calcul, adapté à un environnement d'exécution Google Colab avec GPU Tesla T4.

2.4.3 Quelques difficultés rencontrées en utilisant l'environnement local

Avant la phase d'entraînement initiale du modèle de détection, plusieurs ajustements ont été effectués afin d'améliorer les performances et la stabilité globale du système. Ces ajustements ont porté principalement sur le redimensionnement des images ainsi que la normalisation des données d'entrée.

Il a été constaté que la taille et la qualité des données d'entraînement jouent un rôle déterminant dans la précision des prédictions. Lors de la première phase d'entraînement, les données utilisées provenaient d'images locales. Ces images ont été labellisées manuellement à l'aide de l'outil **LabelImg**, une étape particulièrement chronophage. L'entraînement du modèle a ensuite été réalisé sur une machine locale en utilisant uniquement le processeur (CPU), ce qui a considérablement ralenti le processus d'apprentissage. Malgré la durée importante de l'entraînement, les résultats obtenus se sont révélés peu satisfaisants, avec un taux d'erreur élevé lors des phases de prédiction.

Afin d'améliorer la qualité de la labellisation et des performances du modèle, il a été décidé d'exploiter YOLOv8x pour la régénération automatique des fichiers texte (.txt) contenant les annotations des objets, permettant ainsi une meilleure définition des classes. Cette approche a conduit à une amélioration notable de la qualité des données annotées. Toutefois, l'entraînement sur une machine locale restait fortement contraint par la puissance de calcul disponible. À titre illustratif, un entraînement sur un jeu de données d'environ 10 000 images avec 50 époques nécessitait près de quatre jours de calcul, sans aboutir à une convergence complète du modèle.

Face à ces limitations, la solution optimale a consisté à recourir à une infrastructure cloud, en particulier Google Colab, combinée à Google Drive pour le stockage et la gestion des résultats. Cette approche a permis de bénéficier de ressources matérielles plus performantes, notamment l'utilisation de GPU, réduisant considérablement le temps d'entraînement et améliorant la fiabilité des prédictions. Par ailleurs, des tests empiriques ont été menés afin de déterminer une valeur de seuil d'anomalie offrant un bon compromis entre la sensibilité du système et la réduction du taux de fausses alertes.

Enfin, lors de l'intégration des modèles au sein de l'application de surveillance, plusieurs ajustements techniques ont été réalisés. Des mécanismes de gestion des ressources ont été mis en place afin d'éviter les conflits entre bibliothèques, notamment lors de l'utilisation simultanée de différents frameworks de deep learning. Ces optimisations, bien que mineures sur le plan technique, se sont révélées essentielles pour garantir le bon fonctionnement et la robustesse du système en conditions réelles.

2.5 Présentation des résultats

L'évaluation des performances du système repose sur plusieurs indicateurs statistiques et graphiques permettant d'analyser la qualité des prédictions du modèle de détection ainsi que du modèle d'anomalie. Les principaux outils utilisés sont :

- La matrice de confusion normalisée ;
- Les courbes Recall–Confidence
- Les courbes Precision–Recall
- Les courbes Precision–Confidence
- Les courbes F1–Confidence
- Les courbes d'apprentissage (training curve)

2.5.1 La matrice de confusion normalisée issue des données des tests

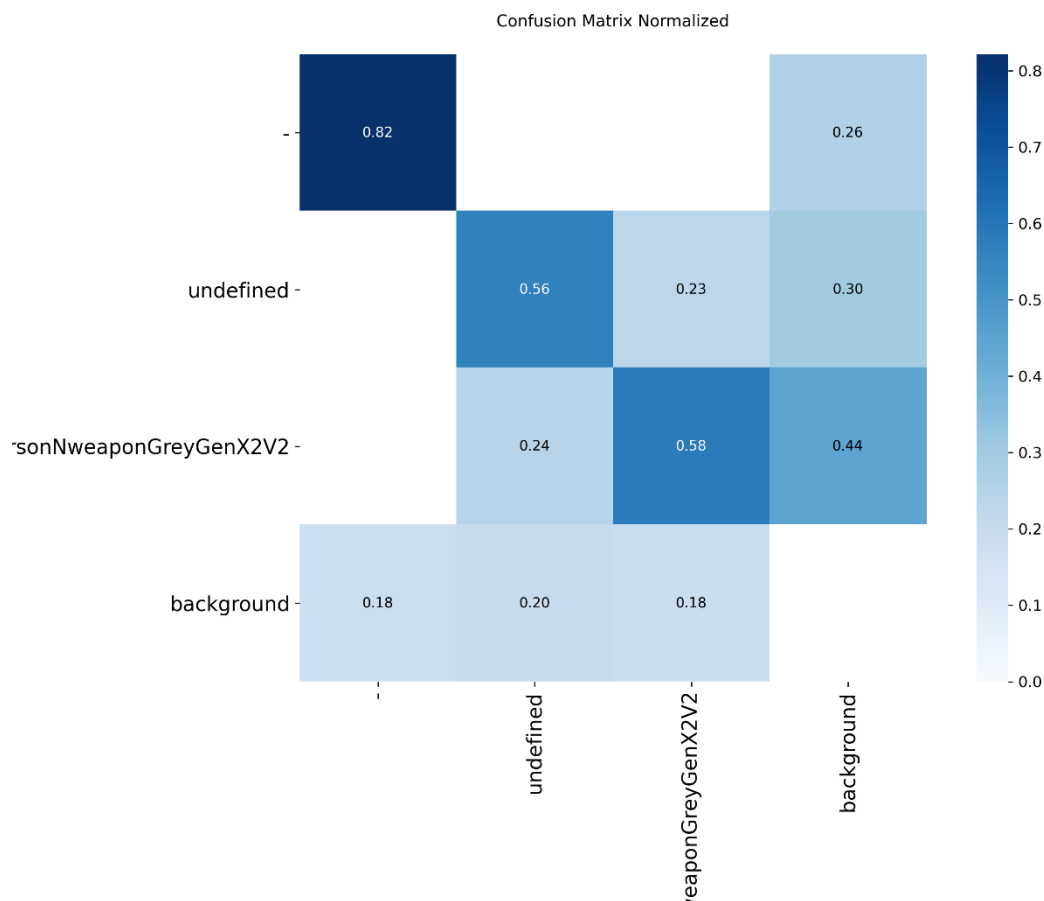


Figure 10 Matrice de confusion normaliser pour les données de test

Avant de procéder à l'interprétation de la Figure 10, il convient tout d'abord de préciser la dénomination des classes associées à ce modèle.

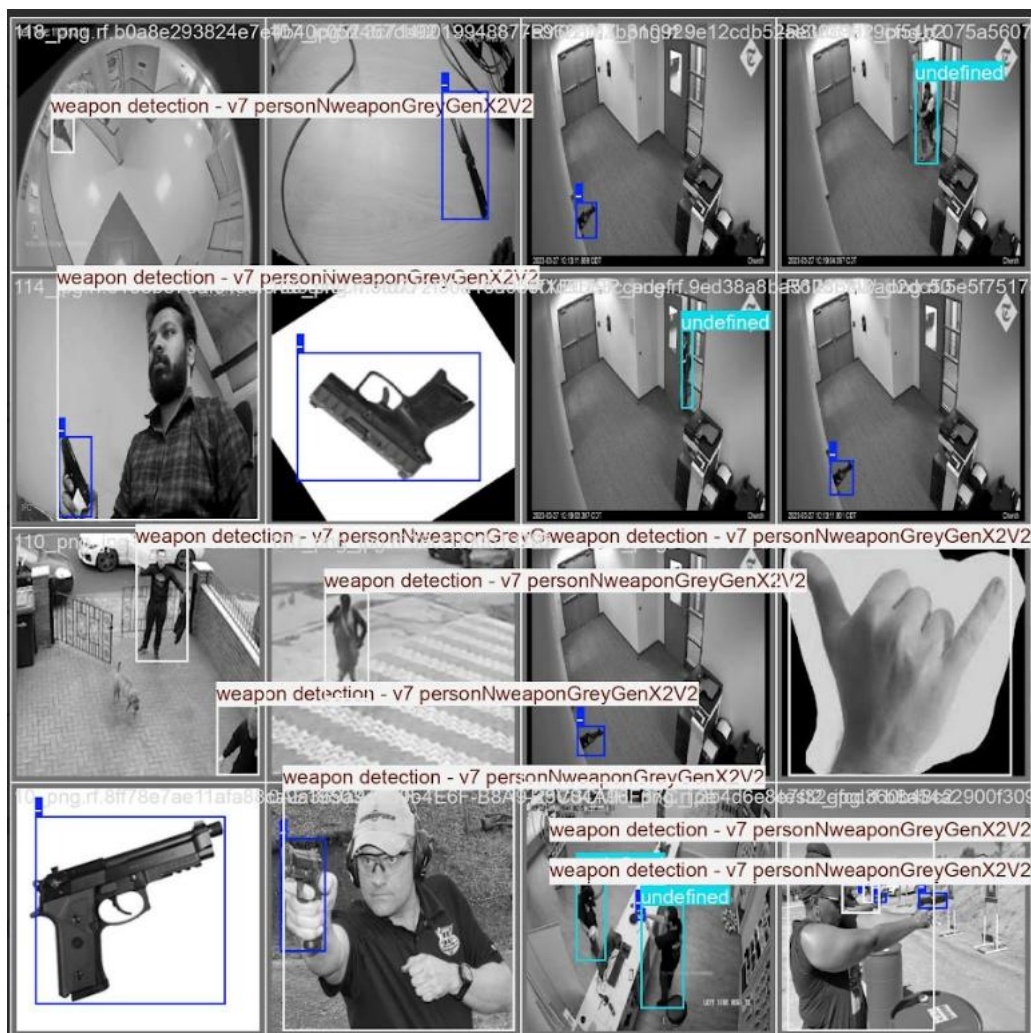


Figure 11 Figure illustrative des noms de classe du dataset

Le modèle de détection développé comporte trois classes distinctes. La première, notée « - », correspond à la classe regroupant les armes seules. La deuxième, intitulée « undefined », désigne les instances n'appartenant ni à la classe des personnes armées ni à celle des armes seules. Enfin, la classe « weapon detection -v7 personNweaponGreyX2V2 » regroupe les individus porteurs d'armes. Cette distinction entre les différentes classes est clairement illustrée à la Figure 11.).

Alors, l'interprétation de la matrice montrée à la Figure 10 repose sur le fait que les colonnes représentent les classes réelles (true labels) tandis que les lignes correspondent aux classes prédites par le modèle. Étant une matrice de confusion normalisée, chaque colonne est

normalisée par le nombre total d'échantillons appartenant à la classe réelle correspondante, ce qui permet d'obtenir des valeurs comprises entre 0 et 1.

Déterminés par :

$$C_{IJ_norm} = \frac{C_{IJ}}{\sum_{I=1}^K C_{ij}}$$

Où :

- C_{IJ_norm} : est la nouvelle valeur c'est-à-dire la valeur normalise dans la matrice
- C_{IJ} : est la valeur de départ c'est-à-dire de la matrice non normalisée
- K : est le nombre des classes du modèle

Les valeurs situées sur la diagonale principale représentent les prédictions correctes du modèle, tandis que les valeurs hors diagonale traduisent les erreurs de classification, c'est-à-dire les confusions entre différentes classes. L'analyse de cette matrice de confusion normalisée montre que le modèle YOLO pour les données issues du test, que la classe "-" est correctement prédite avec un taux de 82%. En revanche, les classes "undefined" et "PersonWeaponGreyGenX2V2" présentent une confusion mutuelle notable (23% et 24%), ce qui suggère une similarité visuelle entre ces catégories. De plus, une proportion non négligeable d'instances est classée comme background, traduisant la présence de faux négatifs. Ces résultats indiquent que le modèle offre des performances satisfaisantes mais perfectibles, notamment pour les classes visuellement proches. Cette confusion peut s'expliquer par plusieurs facteurs, notamment la petite taille des objets, les conditions d'éclairage variables ainsi que la complexité et l'encombrement des scènes vidéo analysées.

2.5.2 Courbe Recall–Confidence

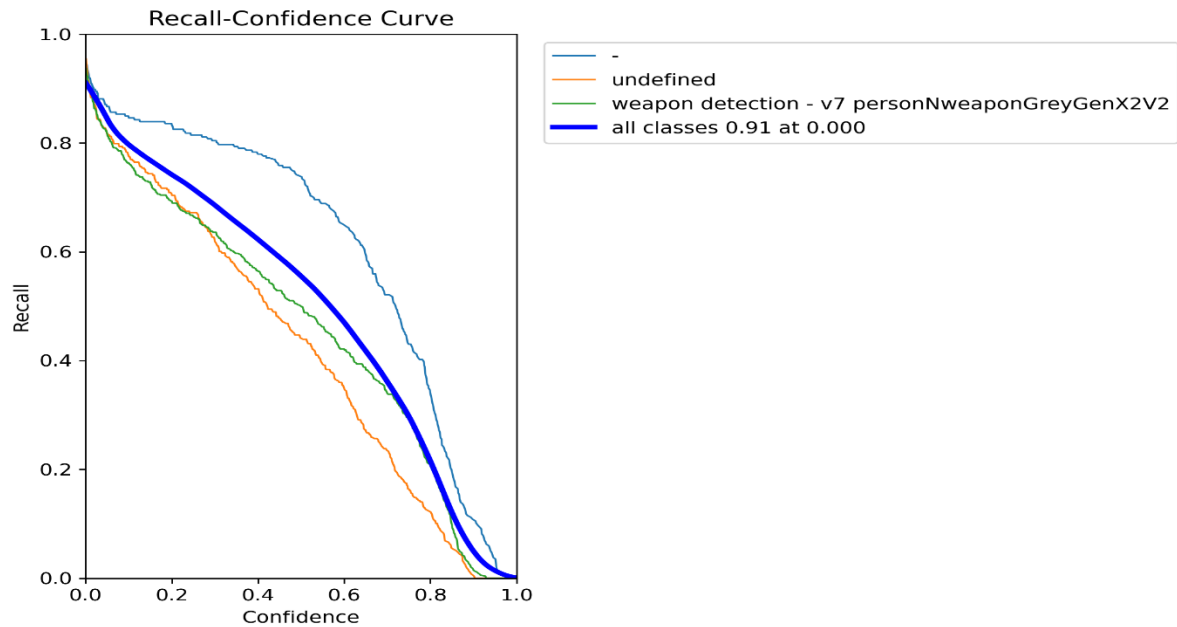


Figure 12 Graphique Recall confidence

La Figure 12 présente la courbe *Recall–Confidence*. On observe que le rappel (Recall) est maximal lorsque le seuil de confiance est faible, puis diminue progressivement à mesure que le seuil augmente. Cela signifie que lorsque le modèle accepte plus de prédictions (seuil bas), il détecte davantage d’objets, mais au risque d’augmenter les faux positifs. À l’inverse, un seuil élevé réduit le rappel, car certaines détections correctes sont rejetées. La courbe globale montre un bon rappel initial (≈ 0.9), ce qui indique que le modèle détecte une grande proportion d’objets au seuil faible.

2.5.3 Courbe Precision–Recall (PR)

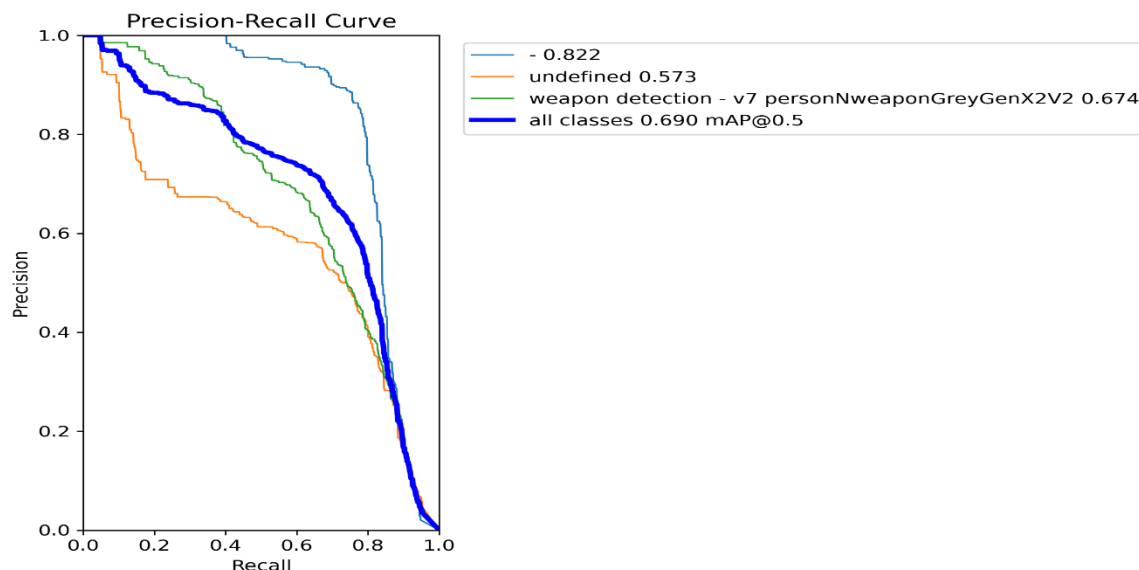


Figure 13 Graphique précision-rappel

La Figure 13 illustre la courbe *Precision–Recall* (PR). Cette courbe est essentielle pour évaluer les performances globales du modèle. Le [mAP@0.5](#) (correspond à la moyenne des précisions calculées pour un seuil d'Intersection over Union (IoU) fixé à 0.5) global est d'environ **0.69**, ce qui traduit une performance modérée. On remarque que la classe principale (*weapon detection*) obtient une performance acceptable (≈ 0.67), tandis que la classe *undefined* présente des performances plus faibles (≈ 0.57), indiquant une confusion plus importante pour cette catégorie.

2.5.4 Courbe Precision–Confidence

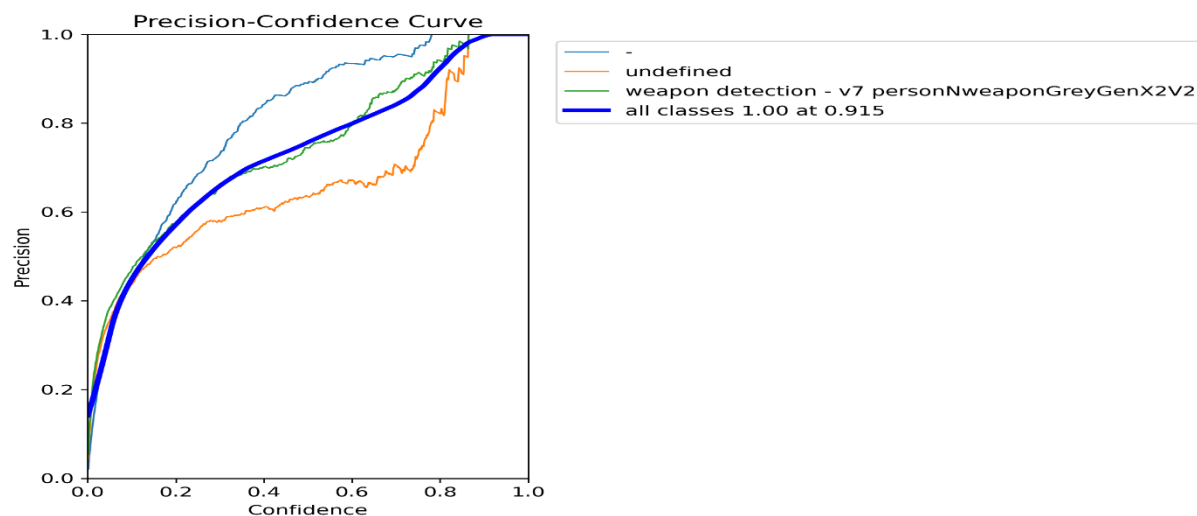


Figure 14 Graphique de la precision-confidence

La Figure 14 montre la courbe *Précision–Confidence*. On constate que la précision augmente avec le seuil de confiance. Lorsque le seuil dépasse 0.9, la précision approche 1.0, ce qui signifie que les prédictions conservées à ce niveau sont presque toutes correctes. Cependant, cela se fait au détriment du rappel, car moins d’objets sont détectés. Ce graphique met en évidence le compromis classique entre précision et rappel.

2.5.5 Courbe F1–Confidence

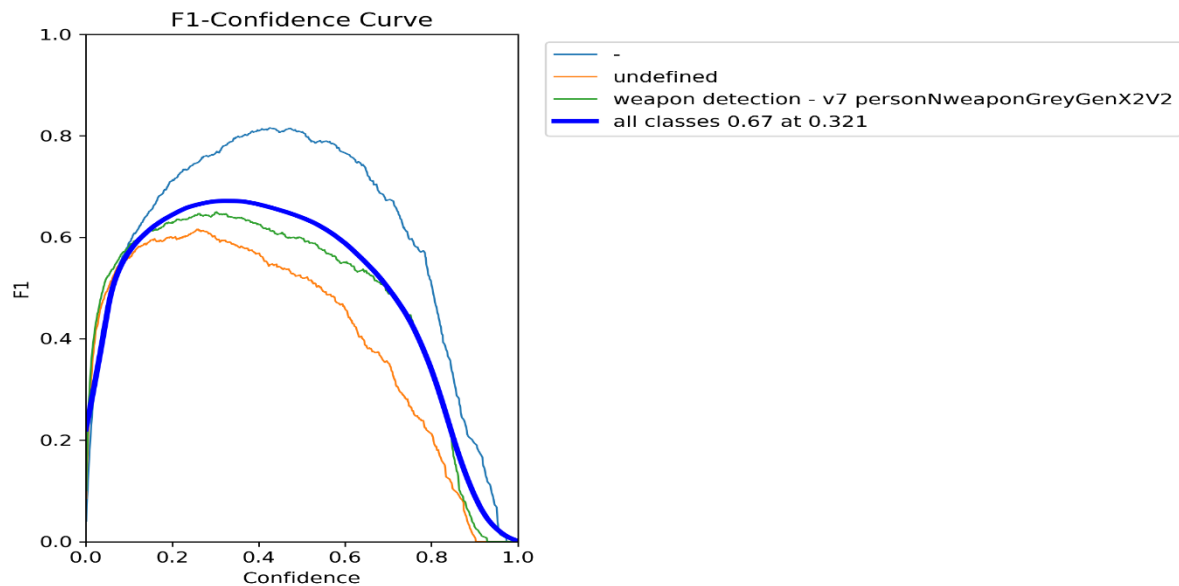


Figure 15 Graphique F1 Confidence

La Figure 15 présente la courbe *F1–Confidence*. Le score F1 représente l’équilibre entre précision et rappel,

Déterminer par :

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Le score maximal obtenu est d’environ 0.67, pour un seuil de confiance d’environ **0.32**. Cela indique que ce seuil constitue le meilleur compromis entre détection correcte et limitation des erreurs. Au-delà de cette valeur, le F1 diminue progressivement.

En gros, les résultats obtenus montrent que le modèle YOLOv8 présente des performances globalement satisfaisantes en détection d’armes. L’analyse des courbes d’évaluation met en évidence le compromis classique entre précision et rappel : lorsque le seuil de confiance augmente, la précision s’améliore tandis que le rappel diminue. La courbe Precision–Recall

permet d'estimer le **mAP@0.5**, qui atteint environ **0.69 (69 %)**. Le mAP@0.5 correspond à la moyenne des précisions calculées pour un seuil d'Intersection over Union (IoU) fixé à 0.5, ce qui signifie qu'une prédiction est considérée correcte si le chevauchement entre la boîte prédite et la boîte réelle dépasse 50 %. Cette valeur indique que le modèle obtient une performance modérée mais exploitable. Par ailleurs, le score F1 maximal d'environ 0.67, atteint pour un seuil de confiance intermédiaire, confirme un équilibre acceptable entre précision et rappel. Globalement, les résultats montrent que le modèle est fonctionnel, bien que perfectible.

2.5.6 Courbe d'apprentissage de l'autoencodeur

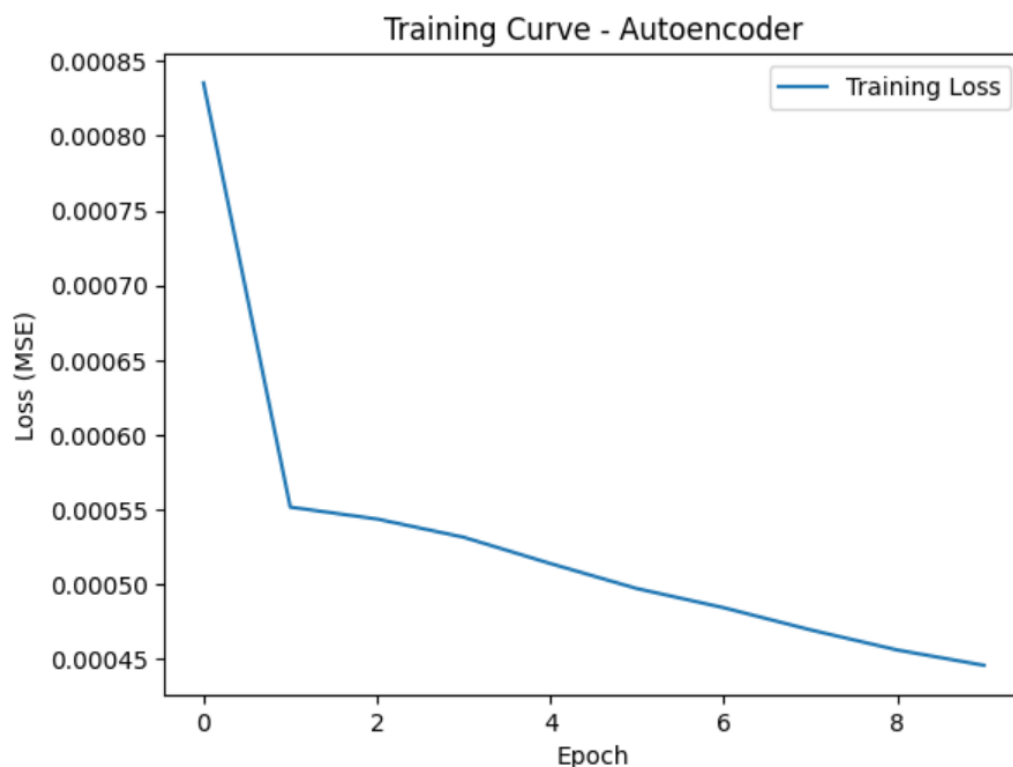


Figure 16 Graphique du training curve de l'autoencodeur

La Figure 16 montre l'évolution de la fonction de perte (Mean Squared Error) de l'autoencodeur au cours de l'entraînement. On observe une diminution rapide de l'erreur durant les premières époques, suivie d'une diminution progressive, indiquant que le modèle apprend efficacement à reconstruire les images normales. La convergence stable de la courbe confirme que le modèle a correctement capturé les caractéristiques des scènes normales, ce qui est essentiel pour la détection d'anomalies basée sur l'erreur de reconstruction

2.6 Conclusion partielle

Cette partie a permis de concevoir et de simuler un système de vidéosurveillance intelligent basé sur l'apprentissage profond. L'architecture modulaire développée intègre l'acquisition vidéo, le traitement des images et un module d'intelligence artificielle combinant un modèle de détection d'objets fondé sur YOLOv8 (pour les personnes, armes à feu et couteaux) ainsi qu'un autoencodeur convolutionnel dédié à la détection d'anomalies. L'exploitation d'un modèle pré-entraîné a permis d'obtenir des performances satisfaisantes en temps réel tout en réduisant la complexité d'entraînement. Les résultats expérimentaux, notamment à travers les métriques de précision, rappel et F1-score, confirment la faisabilité du système pour l'identification de situations potentiellement critiques. Malgré certaines limites liées aux conditions d'éclairage et aux confusions entre objets visuellement proches, l'approche proposée démontre la pertinence de l'intégration de techniques de vision par ordinateur dans un système de surveillance intelligent.

Chapitre 3

Implémentation et Validation du Système de Surveillance

3.1. Introduction

Dans ce chapitre, nous présentons la réalisation pratique de notre système de vidéosurveillance intelligent. Tout d'abord, nous expliquons l'environnement matériel et logiciel utilisé pour l'implémentation du système. Ensuite, nous présentons les différents modules intégrés, notamment l'acquisition du flux vidéo, l'analyse par intelligence artificielle basée sur le modèle **YOLOv8**, ainsi que le module de détection d'anomalies. Par la suite, nous décrivons le fonctionnement général du système et l'intégration avec une caméra IP. Enfin, nous présentons le prototype final réalisé ainsi que les tests expérimentaux effectués pour valider son bon fonctionnement en conditions réelles.

3.2 Schéma fonctionnel du système

La Figure 17 présente le schéma fonctionnel du système réalisé grâce à l'extension Lucid Add depuis Word. Elle illustre l'architecture fonctionnelle du système de vidéosurveillance intelligent implémenté en Python à l'aide de FastAPI.

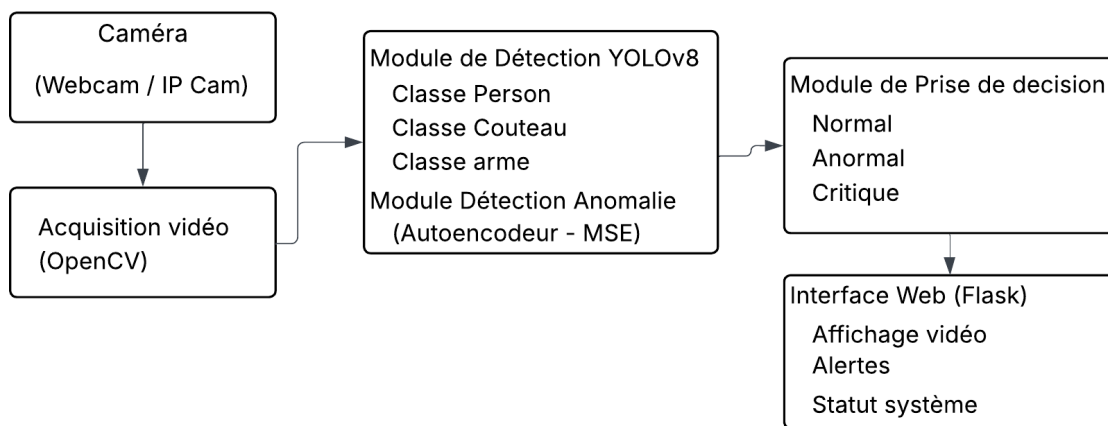


Figure 17 Fonctionnement du système

Le processus de fonctionnement du système débute par l'acquisition des images via une caméra locale ou un flux IP, les images étant transmises au serveur backend sous forme de requêtes

HTTP vers l'endpoint /analyze_frame. Après décodage avec OpenCV, chaque image est traitée par plusieurs modules d'intelligence artificielle. La première repose sur des modèles YOLOv8 dédiés à la détection d'objets, notamment les classes *person*, *knife* et autres objets dangereux, avec des seuils de confiance configurables. Le second module réalise une détection d'anomalies à l'aide d'un autoencodeur entraîné, qui calcule l'erreur de reconstruction basée sur la moyenne des erreurs quadratiques [29] :

$$MSE = \frac{1}{n} \sum (x_i - \hat{x}_i)^2$$

Avec :

- n : nombre total de pixels de l'image
- x_i : valeur réelle du pixel
- $\hat{x}_i$: Valeur reconstruite par l'autoencodeur

La valeur du seuil a été déterminée conformément à la règle statistique basée sur la loi normale centrée réduite de Gauss, en considérant un niveau de confiance de 99,7 % :

$$Seuil = \mu + 3\sigma \text{ [30]}$$

Où :

- μ : la moyenne
- σ : l'écart-type

Si le score dépasse le seuil, la scène est considérée comme anormale. Les résultats des détections d'objets et du score d'anomalie sont ensuite transmis au module de prise de décision, qui détermine dynamiquement un niveau de risque parmi : *normal*, *anomaly*, *dangerous_object*, *human_intrusion* ou *critical* (cas le plus grave correspondant à la détection simultanée d'un humain et d'une arme). En cas d'événement critique ou d'intrusion humaine, une image annotée est automatiquement sauvegardée et les statistiques globales sont mises à jour. Enfin, un tableau de bord web intégré (généré via FastAPI) permet d'afficher en temps réel les indicateurs du système (nombre total d'images traitées, armes détectées, anomalies, événements critiques et présences humaines) assurant ainsi la supervision continue et l'interaction avec l'utilisateur.

3.3. Configuration de l'environnement de l'expérience

Afin d'évaluer les performances du système de vidéosurveillance intelligente proposé, il est essentiel de définir l'environnement expérimental dans lequel les différentes simulations et validations ont été réalisées. Cette section présente les caractéristiques matérielles et logicielles utilisées, les scénarios de test considérés ainsi que les indicateurs d'évaluation permettant de mesurer les performances du système.

3.3.1. Configuration matérielle

Afin d'atteindre les objectifs fixés dans ce travail, nous avons utilisé un ordinateur personnel disposant d'une configuration adaptée au développement et à l'exécution d'applications de vision par ordinateur. Toutefois, cette configuration présente certaines limites, notamment l'absence d'une carte graphique dédiée (GPU), ce qui réduit la puissance de calcul nécessaire à l'entraînement efficace des modèles d'intelligence artificielle.

Cet environnement a néanmoins permis de développer l'interface graphique du système proposé ainsi que d'assurer l'interaction avec la caméra intégrée de l'ordinateur. Pour la détection d'images et la visualisation en temps réel, nous avons choisi d'utiliser la webcam de l'ordinateur. Ce choix s'explique principalement par sa simplicité d'utilisation, étant donné qu'elle est déjà intégrée dans la machine. Cependant, il convient de noter que d'autres dispositifs de capture vidéo peuvent également être utilisés, tels que le module **ESP32-CAM** ou d'autres caméras IP. Ces dispositifs peuvent être intégrés au système grâce à leur adresse IP, permettant ainsi leur connexion et leur utilisation dans l'application de surveillance.

L'ordinateur physique constitue également le moteur principal de ce travail, car il agit comme un serveur local pour le stockage et l'exécution du système. Bien que certaines étapes du projet aient fait recours au cloud pour garantir la pérennité et la sécurité des données, une partie importante du stockage des informations est conservée localement.

Par ailleurs, la plateforme **Google Colab** a été utilisée pour l'entraînement des modèles d'apprentissage profond. En effet, réaliser ces entraînements localement aurait nécessité plusieurs jours de calcul, en raison de l'absence d'un GPU performant. De plus, cela aurait augmenté le risque de perte de données, car il aurait fallu maintenir l'ordinateur allumé pendant une longue période.

Enfin, il convient de signaler que l'annotation manuelle des images en local s'avère particulièrement difficile, voire très contraignante, en raison du volume important des images

à traiter avec des outils tels que **Labelling**. Pour cette raison, nous avons également eu recours à des bases de données déjà préparées et partiellement annotées disponibles en ligne, notamment sur les plateformes **Kaggle** et **Roboflow**, ce qui a permis de faciliter et d'accélérer la préparation du dataset utilisé dans ce travail.

En gros, ce système de sécurité fonctionne sous ce pipeline de la Figure 18:

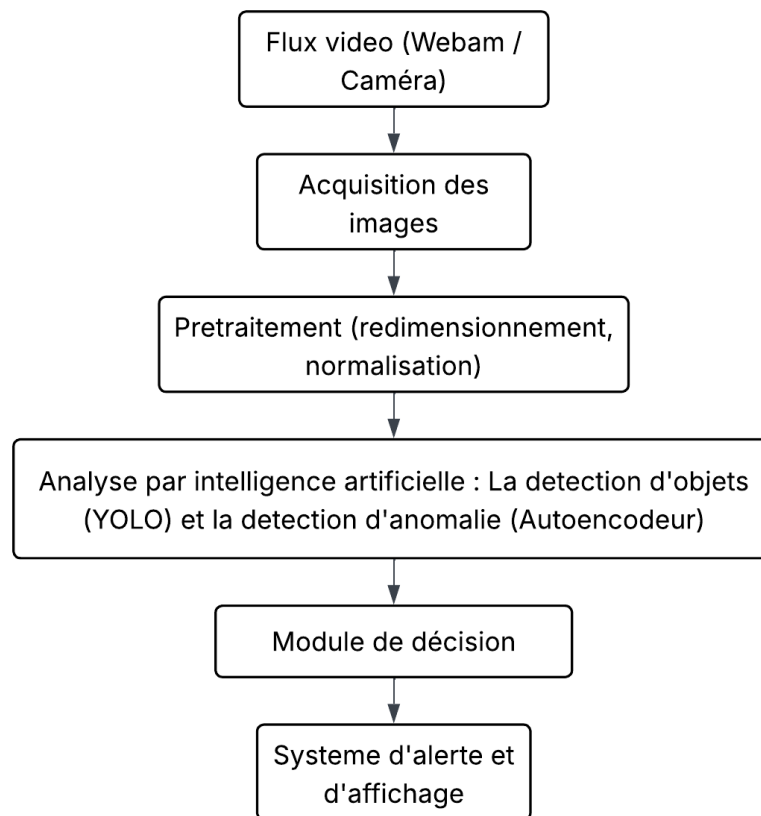


Figure 18 pipeline du fonctionnement du système

Ainsi, afin de se faire une idée des performances du modèle de détection d'objets, il convient de visualiser les résultats obtenus lors des tests :



Figure 19 Résultat de l'entrainement du modèle des données de test

Pour interpréter correctement la Figure 19, il convient de noter que le label [AWK37.1] de chaque classe est suivi de la valeur de sa prédiction normalisée sur une échelle de 0 à 1. À partir de ces valeurs, il apparaît que la classe « arme » est prédite de manière nettement plus fréquente, ce qui est cohérent avec la matrice de confusion normalisée présentée à la Figure 10.

3.4 Environnement de développement

Pour la réalisation de ce projet, la distribution **Anaconda** a été utilisée afin de configurer un environnement de travail adapté aux exigences du Deep Learning. Anaconda est une plateforme open-source permettant la gestion des environnements virtuels et des dépendances logicielles. Son interface en ligne de commande, *Anaconda Prompt*, facilite l'installation des bibliothèques, l'activation d'environnements isolés et l'exécution des scripts Python. Cette approche garantit la compatibilité des versions et évite les conflits entre packages.

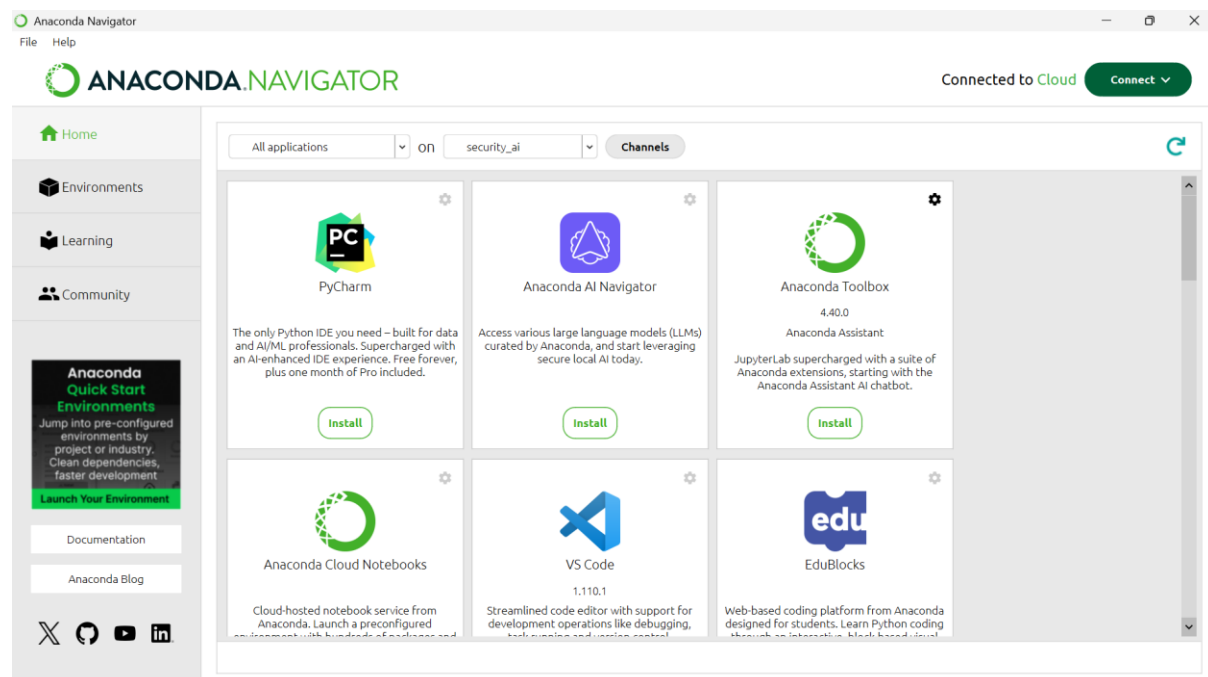


Figure 20 Présentation de l'environnement anaconda

Dans cet environnement, plusieurs bibliothèques essentielles ont été installées :

- TensorFlow : utilisé pour la conception et l'entraînement de l'autoencodeur ainsi que le calcul de l'erreur de reconstruction (MSE).
- Ultralytics (YOLO) : employé pour la détection d'objets en temps réel dans les images et flux vidéo.
- PyTorch : nécessaire au fonctionnement des modèles YOLO intégrés via Ultralytics.

- FastAPI : utilisé pour développer l'API backend exposant les modèles d'intelligence artificielle.
- Uvicorn : serveur ASGI permettant l'exécution et le déploiement de l'application.
- OpenCV et NumPy : dédiés respectivement au traitement d'images et aux calculs numériques.

3.4.1 Lancement de l'application

Notre application est composée de deux modules principaux :

- Un module de **surveillance**, chargé du traitement des flux vidéo et de la détection d'événements ;
- Un module de **visualisation des activités**, permettant l'affichage et le suivi des résultats générés par le système.

Le lancement de ces deux composantes est centralisé dans un fichier exécutable Python dédié. Son exécution s'effectue via la ligne de commande, à partir de l'environnement virtuel préalablement activé, en utilisant une instruction de type :

python run_all.py

Ce script orchestre le démarrage des différentes instances du serveur (notamment via Uvicorn), permettant ainsi l'accès aux interfaces de surveillance et de visualisation. Cette approche centralisée simplifie le déploiement et garantit une exécution cohérente des différents modules du système.

3.4.2 Interface du module de surveillance

Une fois l'application lancée, l'utilisateur a la possibilité d'accéder soit au module de surveillance, soit au module de visualisation des activités. Il est toutefois recommandé d'initier d'abord le module de surveillance, puisqu'il constitue le cœur fonctionnel du système.

L'interface du module de surveillance se présente sous la forme d'un tableau de bord interactif, structuré en plusieurs zones distinctes.

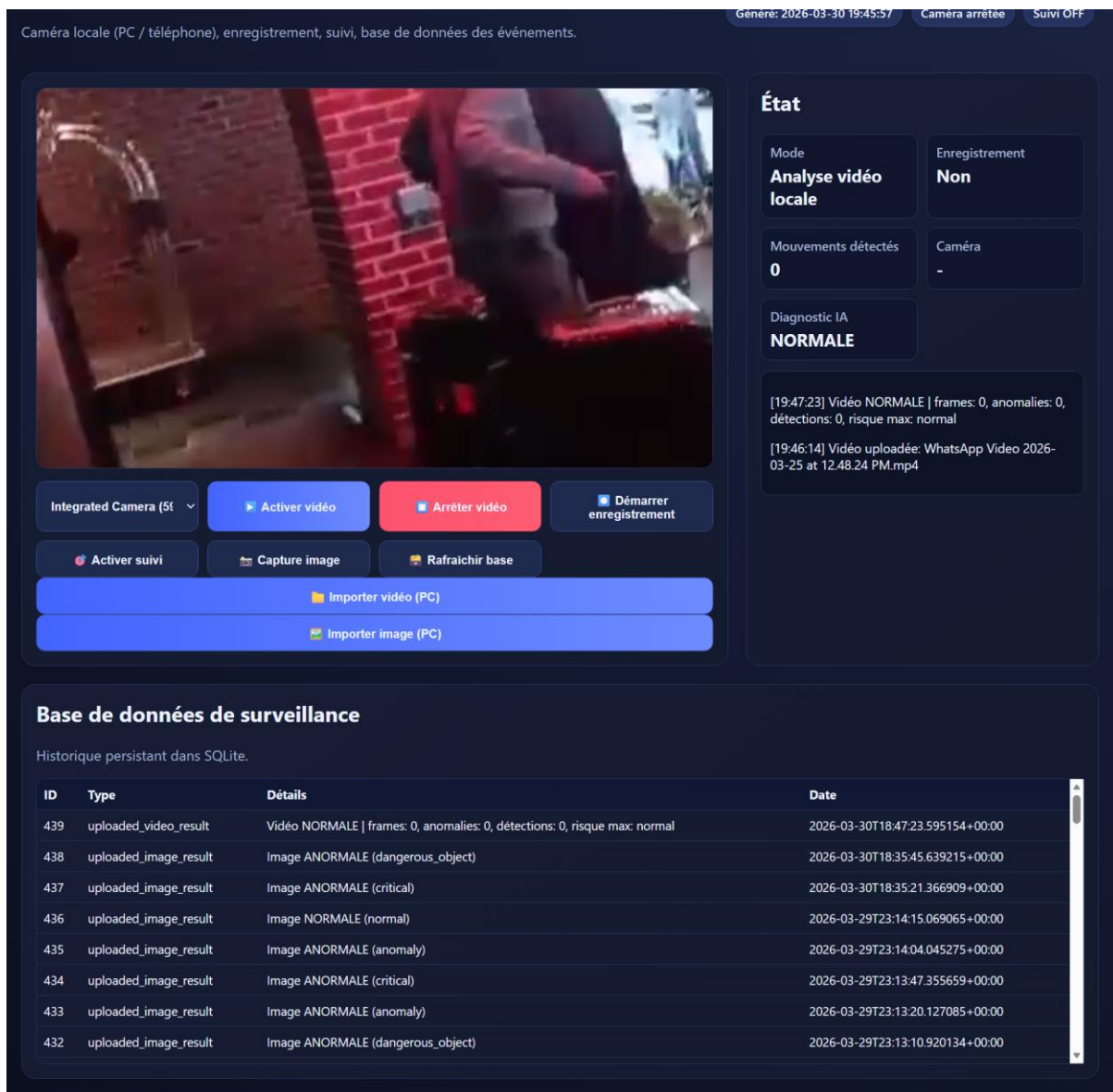


Figure 21 Interface du module de surveillance

La **Erreur ! Source du renvoi introuvable.** illustre le traitement des données réelles par notre système. Comme le montre la figure, la sortie indique que la vidéo analysée correspond à une situation normale. Une question se pose alors : comment le système parvient-il à analyser cette vidéo ? La réponse réside dans le fait que la vidéo est décomposée en une série d'images, puisque celle-ci n'est, en réalité, qu'une succession d'images assemblées.

Pour réaliser cette opération, l'utilisateur doit cliquer sur le bouton « Importer vidéo » afin de charger une vidéo depuis son poste, ou sur le bouton « Importer image » pour charger une image individuelle. Les détails relatifs aux autres éléments de l'interface sont présentés ci-après :

- La zone d'affichage vidéo

C'est la partie centrale de l'interface est dédiée au flux vidéo en temps réel provenant de la caméra locale (PC ou téléphone). Cette zone permet :

- La visualisation directe du flux capturé ;
- L'affichage des détections effectuées par le modèle d'intelligence artificielle ;
- L'affichage des détections effectuées par le modèle d'intelligence artificielle ;

Lorsque la caméra n'est pas activée, cette zone reste inactive, comme illustré dans la figure correspondante.

- **Zone d'état**

C'est sur la partie droite de l'interface qui regroupe les informations principales relatives au fonctionnement du système en autre :

- Mode : indique si le système est actif ou inactif ;
- Enregistrement : précise si la capture vidéo est en cours de sauvegarde ;
- Mouvements détectés : affiche le nombre d'événements identifiés ;
- Caméra : indique la source vidéo sélectionnée.

- **Zone des boutons de commande**

C'est la partie inférieure, plusieurs boutons permettent d'interagir avec le système, notamment pour :

- Démarrer ou arrêter la caméra ;
- Activer ou désactiver le suivi ;
- Activer l'enregistrement ;
- Sélectionner la source vidéo.
- Rafraichir la base

- **Zone de la base des données**

Cette zone permet de voir l'historique de visualisation.

Au-delà des vidéos, il est important de noter que notre système intègre également un modèle de détection. Afin d'évaluer les performances de ce modèle, nous avons utilisé quelques images à des fins d'illustration :



Figure 22 Résultat de notre modèle pour la validation

Ainsi, le Figure 22 nous renseigne sur la manière dont notre modèle perçoit les données lors des tests. On constate que le modèle parvient globalement à détecter les objets, mais qu'il présente également certaines lacunes telle que l'on voit dans la Figure 22. Bien que notre modèle parvienne à détecter les objets il y a également quelques images que le model n'a pas su reconnaître.

3.4.3 Module de visualisation des activités

Le module de visualisation des activités permet d'analyser et de consulter les résultats générés par le système de surveillance. Contrairement au module de surveillance qui fonctionne en temps réel, ce module est orienté vers l'analyse statistique et le suivi des événements détectés.

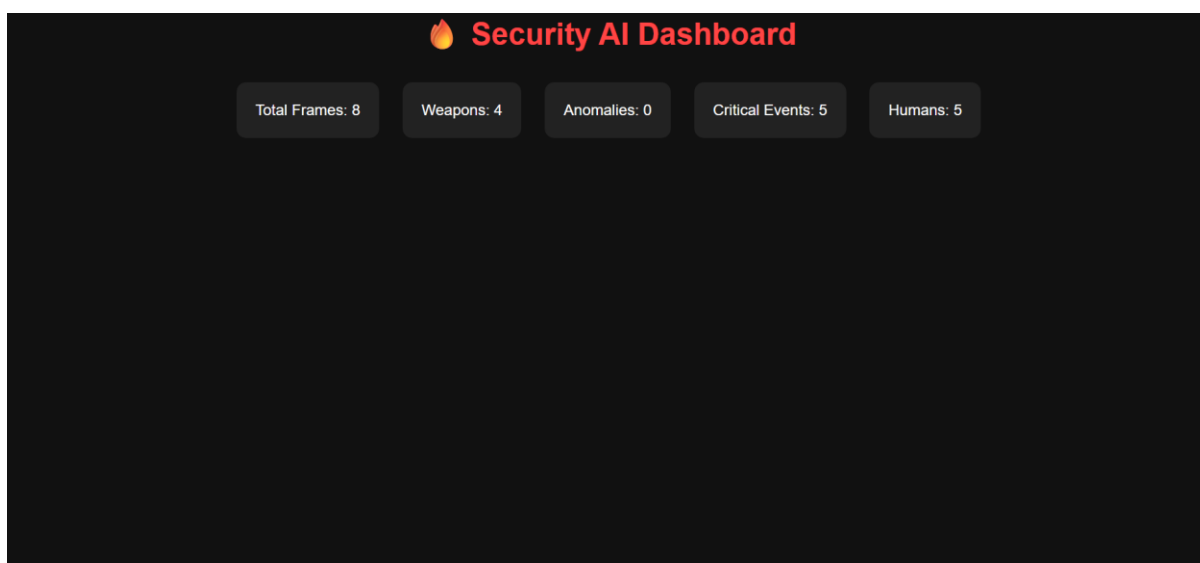


Figure 23 Interface du module de visualisation

L'interface se présente sous la forme d'un tableau de bord intitulé : **Security AI Dashboard**. Ce tableau de bord affiche plusieurs indicateurs numériques permettant d'évaluer l'activité du système :

- Total Frames : nombre total d'images (frames) traitées par le système.
- Weapons : nombre d'objets classés comme armes détectés.
- Anomalies : nombre d'événements considérés comme comportements anormaux.
- Critical Events : nombre d'événements critiques nécessitant une attention particulière.
- Humans : nombre de personnes détectées par le modèle d'intelligence artificielle.

3.4.3.1 Rôle du module de visualisation

Le module de visualisation joue un rôle essentiel dans ce système de surveillance, car il permet d'assurer le suivi global des performances en fournissant une vue synthétique des données traitées. Il facilite l'analyse statistique des différentes détections effectuées (personnes, armes, anomalies ou événements critiques), contribue à l'évaluation du niveau de risque en fonction des événements enregistrés et offre une consultation rapide et claire des principaux indicateurs de sécurité. Ainsi, il constitue un outil d'aide à la décision permettant une interprétation efficace des résultats générés par le module de surveillance.

Lorsque le système n'a pas encore traité de flux vidéo ou lorsque la surveillance est inactive, les valeurs affichées restent initialisées à zéro, comme suit :

- Total Frames : 8
- Weapons : 4
- Anomalies : 0
- Critical Events : 5
- Humans : 5

Ces valeurs évoluent dynamiquement dès que le module de surveillance est activé et que des événements sont détectés.

Ainsi, le module de visualisation constitue une interface d'aide à la décision, permettant une lecture claire et synthétique des informations issues du système intelligent de surveillance.

3.4.3.2 Comparaison entre ces deux modules

Le module de surveillance et le module de visualisation remplissent des fonctions complémentaires au sein de l'application. Le module de surveillance constitue la partie

opérationnelle du système : il assure la capture du flux vidéo en temps réel, la détection des objets ou comportements suspects à l'aide des modèles d'intelligence artificielle, le suivi des mouvements ainsi que l'enregistrement des événements. Il agit donc directement sur les données brutes issues de la caméra et représente le cœur fonctionnel de l'application.

En revanche, le module de visualisation est orienté vers l'analyse et l'interprétation des résultats produits par le module de surveillance. Il ne traite pas directement le flux vidéo, mais affiche des indicateurs statistiques tels que le nombre total d'images traitées, les personnes détectées, les armes identifiées, les anomalies ou les événements critiques. Ce module permet ainsi une lecture synthétique des performances du système et facilite la prise de décision.

Ainsi, tandis que le module de surveillance assure l'acquisition et le traitement en temps réel des données, le module de visualisation fournit une analyse structurée et statistique des informations générées.

3.5 Conclusion partielle

Dans ce chapitre, nous avons présenté l'implémentation et la validation du système de surveillance intelligent. L'architecture développée repose sur FastAPI pour la gestion du serveur et de l'interface de surveillance, combinée à l'utilisation de modèles d'intelligence artificielle. Le modèle YOLOv8 a été utilisé pour la détection d'objets tels que les personnes et les armes, tandis qu'un autoencodeur a permis d'identifier les anomalies à partir de l'erreur de reconstruction (MSE) et d'un seuil statistique.

Les expérimentations réalisées ont permis d'évaluer les performances du système dans différents scénarios. Les résultats obtenus montrent une précision élevée dans la détection des personnes et des objets dangereux, ainsi qu'une capacité efficace à identifier des comportements anormaux dans le flux vidéo. De plus, les tests de performance indiquent que le système est capable de fonctionner en quasi temps réel, ce qui le rend adapté à une utilisation dans des applications de surveillance intelligente. Ainsi, les résultats expérimentaux confirment la pertinence de l'architecture proposée et démontrent que le système constitue une solution efficace pour l'assistance à la sécurité.

Conclusion générale

Ce mémoire avait pour objectif d'explorer la contribution potentielle de la vision par ordinateur au renforcement de la sécurité dans la ville de Goma, marquée par des défis liés à l'insécurité, aux limites infrastructurelles et à la prédominance de méthodes traditionnelles de surveillance. Les travaux réalisés démontrent que la vision par ordinateur constitue une solution pertinente et applicable localement. En concevant un système de vidéosurveillance intelligent basé sur YOLOv8 pour la détection d'armes et de personnes, ainsi qu'un autoencodeur convolutionnel pour la détection d'anomalies, nous avons montré qu'il est possible d'automatiser l'analyse des flux vidéo en temps réel et de classifier les situations en niveaux de risque (Normal, Anomalie, Critique). Les résultats expérimentaux obtenus ($mAP@0.5 \approx 0.69$ et $F1\text{-score} \approx 0.67$) confirment que le système est capable d'identifier des situations potentiellement dangereuses avec un compromis acceptable entre précision et rappel.

Ainsi, à la question de savoir comment un système intelligent peut contribuer à prévenir ou réduire les actes de violence et de banditisme, ce travail apporte une réponse concrète : la détection précoce d'armes, la corrélation avec la présence humaine et l'identification d'anomalies permettent une intervention plus rapide et mieux orientée. De même, les données collectées et structurées par le module de visualisation offrent des indicateurs exploitables pour l'analyse comportementale et la prise de décision stratégique par les autorités. L'intégration de telles technologies dans les stratégies des forces de l'ordre apparaît donc envisageable, à condition d'accompagner leur déploiement par des formations adaptées et une infrastructure minimale stable. Enfin, cette recherche montre que la transition vers une sécurité numérisée est techniquement possible, même dans un contexte contraint, à travers l'utilisation d'outils accessibles comme Python, Anaconda et Google Colab.

Cependant, une analyse critique révèle certaines limites. Les performances du modèle, bien que satisfaisantes, restent perfectibles, notamment face aux variations d'éclairage, aux objets visuellement proches et aux environnements complexes. L'étude a été réalisée sur des jeux de données spécifiques qui ne couvrent pas l'ensemble des réalités locales, ce qui peut limiter la généralisation des résultats. Par ailleurs, les contraintes énergétiques, le manque d'infrastructures réseau robustes et l'insuffisance de compétences spécialisées constituent des freins majeurs à un déploiement à grande échelle. Enfin, les enjeux liés à la protection des données et à la cybersécurité méritent une attention plus approfondie.

En dépit de ces limites, ce travail apporte une contribution scientifique et technique significative en proposant une architecture modulaire, adaptable et relativement économique pour la surveillance intelligente. Il ouvre la voie à des recherches futures orientées vers l'amélioration des performances des modèles, l'optimisation pour systèmes embarqués, le renforcement de la cybersécurité et l'adaptation à des contextes opérationnels réels.

Bibliographie

- [1] B. Marr, 7 Amazing Examples Of Computer And Machine Vision In Practice, 2019.
- [2] [En ligne]. Available: <https://www.vumacam.co.za/>. [Accès le 29 Mars 2026].
- [3] Y. K. KAHILU, «CONFLICTUALITE EN AFRIQUE DE GRANDS LACS : CAUSES, CONSEQUENCES,» Revue Intelligence Stratégique, Kolwezi/Lualaba, 2022.
- [4] Nkiendiel, Alain Akwir, Cour des methode des elements finis, Goma, 2025.
- [5] G. Rayane, Conception et réalisation d'une caméra de surveillance intelligente : Détection et reconnaissance d'objet, Algerie, 2023.
- [6] IBM, «Computer vision,» IBM, 2025. [En ligne]. Available: https://www.ibm.com/think/topics/computer-vision?utm_source=chatgpt.com. [Accès le 16 August 2025].
- [7] D. H. H. a. T. N. Wiesel, «Receptive fields and functional architecture of monkey striate cortex.,» London, Journal of Physiology, 1968, pp. 215-243.
- [8] P. A. A. Nkiendiel, Cours de sigmaux et système, Goma, 2025.
- [9] WQU, «ai-lab,» WQU, [En ligne]. Available: <https://learn.wqu.edu/my-courses/courses/ai-lab02/modules/2-crop-disease-in-uganda/tasks/2-4-transfer-learning-and-cross-validation>. [Accès le 24 August 2025].
- [1] w. I. S. a. H. Krizhevsky, «ImageNet Classification with Deep Convolutional Neural
0] Networks.,» IEEE, Ukraine, 2012.
- [1] J. Readmon, «You Only Look Once: Unified, Real-Time Object Detection,» CVPR,
1] Washington, 2016.
- [1] F. S. e. al., «FaceNet: A Unified Embedding for Face Recognition and Clustering,» CVPR,
2] California, 2015.
- [1] Z. C. e. al., «Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,»
3] CVPR, California, 2017.
- [1] wired, «Step Up Surveillance,' U.S.A.,» Associated Press, 24 July 2005. [En ligne].
4] Available: <https://www.wired.com/2005/07/step-up-surveillance-u-s-a/>. [Accès le 19 August 2025].

- [1] pimloc, «secureredact,» secureredact, [En ligne]. Available:
5] <https://www.secureredact.ai/articles/analog-vs-digital-security-cameras>. [Accès le 21 August 2025].
- [1] O. B. A. E. a. R. P. Emmanuel Goldberg, «LEDthere be DoS: Exploiting variable bitrate
6] IP cameras for network DoS,» Cyber Labs@BGU,Ben-Gurion University of the Negev, Israel, Israel, 2025.
- [1] P. V. e. M. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features,»
7] IEEE CVPR, Etas Unis, 2001.
- [1] G. Bradski, «The OpenCV Library,» Dr. Dobb's Journal of Software Tools, Boston, 2000.
8]
- [1] MathWorks, «Computer Vision Toolbox User's Guide,» , documentation officielle,
9] régulièrement mise à jour., Etas Unis.
- [2] Google, «MediaPipe: A Framework for Building Perception Pipelines,» 2019.
0]
- [2] G. v. Rossum, «Python tutorial,» Python Software Foundation, documentation officielle..
1]
- [2] A. P. e. al., «PyTorch: An Imperative Style, High-Performance Deep Learning Library,»
2] NeurIPS, 2019..
- [2] M. A. e. al, « TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,»
3] 2016.
- [2] B. Stroustrup, «The C++ Programming Language,» 4th Edition, Addison-Wesley, 2013.
4]
- [2] Arduino, «Arduino Documentation,» Arduino, [En ligne]. Available:
5] <https://www.arduino.cc>. [Accès le 3 fevrier 2026].
- [2] R. P. Foundation, «Raspberry Pi Documentation,» [En ligne]. Available:
6] <https://www.raspberrypi.org>. [Accès le 31 mars 2026].
- [2] U. I. Forum, «USB Device Class Definition for Video Devices,» norme officielle USB
7] Video Class (UVC).
- [2] S. A. S. S. Marigoudar, «Block diagram of intelligent video analytics model (IVAM),»
8] Researchgate, septembre 2024. [En ligne]. Available:

<https://www.researchgate.net/publication/329541271/figure/fig1/AS:962245114753027@1606428521917/Block-diagram-of-intelligent-video-analytics-model-IVAM.png>.

[Accès le 09 February 2026].

[2] K. UDUJA, Video Surveillance Anomaly Detection: A Review on Deep Learning Benchmarks, IEEE Access, 2016.

[3] mygeekytutor, «regles-regles-empiriques-statistiques,» 2026. [En ligne]. Available: <https://mathcracker.com/fr/regles-regles-empiriques-statistiques>. [Accès le 25 Fevrier 2026].

[3] Verkada, «DVR vs. NVR,» Verkada. [En ligne].
1]

[3] Verkada, «DVR vs. NVR,» Verkada, 2025. [En ligne]. Available: <https://info.verkada.com/compare/dvr-vs-nvr/>. [Accès le 26 September 2025].

[3] A. DIASIVI, Video-surveillance pour la société Sepcongo, Kinshasa: Memoire Online, 3] 2015.

ANNEXES

Code pour le module de visualisation

```
import datetime
import os
from typing import Any, Dict, List, Optional

# Compat PyTorch 2.6+ (évite l'erreur weights_only=True par défaut)
os.environ.setdefault("TORCH_FORCE_NO_WEIGHTS_ONLY_LOAD", "1")

import cv2
import numpy as np
import requests
import torch
from fastapi import FastAPI, File, Header, HTTPException, UploadFile
from fastapi.responses import HTMLResponse, JSONResponse, Response, StreamingResponse
from tensorflow.keras.models import load_model
from ultralytics import YOLO
from fastapi.middleware.cors import CORSMiddleware

def _patch_torch_for_ultralytics() -> None:
    """Compatibilité PyTorch 2.6+ pour charger des checkpoints YOLO legacy."""
    try:
        from ultralytics.nn.tasks import (
            ClassificationModel,
            DetectionModel,
            OBBModel,
            PoseModel,
            SegmentationModel,
        )

        torch.serialization.add_safe_globals(
            [DetectionModel, SegmentationModel, ClassificationModel, PoseModel, OBBModel]
        )
    except Exception:
        pass
```

```

_patch_torch_for_ultralytics()

ANOMALY_THRESHOLD = float(os.getenv("ANOMALY_THRESHOLD", "0.0015787"))
WEAPON_CONF_THRESHOLD = float(os.getenv("WEAPON_CONF_THRESHOLD", "0.4"))
HUMAN_CONF_THRESHOLD = float(os.getenv("HUMAN_CONF_THRESHOLD", "0.3"))
KNIFE_CONF_THRESHOLD = float(os.getenv("KNIFE_CONF_THRESHOLD", "0.15"))
API_KEY = os.getenv("SECURITY_API_KEY", "")

DETECTION_MODEL_PATH = os.getenv("DETECTION_MODEL_PATH", "models/detection_model.pt")
if not os.path.exists(DETECTION_MODEL_PATH):
    DETECTION_MODEL_PATH = os.getenv("FALLBACK_DETECTION_MODEL_PATH",
    "models/yolov8_weapon.pt")

ANOMALY_MODEL_PATH = os.getenv("ANOMALY_MODEL_PATH", "models/autoencoder_ucsd.keras")
HUMAN_MODEL_PATH = os.getenv("HUMAN_MODEL_PATH", "models/yolov8n.pt")
KNIFE_MODEL_PATH = os.getenv("KNIFE_MODEL_PATH", "models/yolov8s.pt")
CAMERA_STREAM_URL = os.getenv("CAMERA_STREAM_URL", "http://127.0.0.1:8000/stream")
SAVE_DIR = os.getenv("SAVE_DIR", "saved_events")
os.makedirs(SAVE_DIR, exist_ok=True)

if not os.path.exists(DETECTION_MODEL_PATH):
    raise RuntimeError(f'Modèle de détection introuvable: {DETECTION_MODEL_PATH}')
if not os.path.exists(ANOMALY_MODEL_PATH):
    raise RuntimeError(f'Modèle d'anomalie introuvable: {ANOMALY_MODEL_PATH}')

app = FastAPI(title="Security AI API", version="2.1.0")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"], # en production mets ton domaine
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

def _load_yolo_model(path: str, model_name: str) -> Optional[YOLO]:
    if not os.path.exists(path):
        print(f'[WARN] {model_name} absent ({path}). Détection désactivée.')
        return None

```

```

try:
    return YOLO(path)
except Exception as exc:
    raise RuntimeError(
        f"Impossible de charger {model_name}: {path}. "
        "Si vous utilisez PyTorch 2.6+, installez torch==2.5.1 (ou <2.6)."
    ) from exc

weapon_model = _load_yolo_model(DETECTION_MODEL_PATH, "modèle de détection")
anomaly_model = load_model(ANOMALY_MODEL_PATH, compile=False)
human_model = _load_yolo_model(HUMAN_MODEL_PATH, "modèle humain")
knife_model = _load_yolo_model(KNIFE_MODEL_PATH, "modèle couteau")

stats = {
    "total_frames": 0,
    "total_weapons": 0,
    "total_anomalies": 0,
    "critical_events": 0,
    "total_humans": 0,
}

event_history: List[Dict[str, Any]] = []

def _check_api_key(x_api_key: Optional[str]) -> None:
    if API_KEY and x_api_key != API_KEY:
        raise HTTPException(status_code=401, detail="Invalid API key")

def _compute_anomaly_score(frame: np.ndarray) -> float:
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    resized = cv2.resize(gray, (128, 128)).astype("float32") / 255.0
    input_img = resized.reshape(1, 128, 128, 1)
    reconstructed = anomaly_model.predict(input_img, verbose=0)
    return float(np.mean((input_img - reconstructed) ** 2))

def _extract_detections(results: Any, target_class_name: Optional[str] = None) -> List[Dict[str, Any]]:
    detections: List[Dict[str, Any]] = []
    for result in results:
        for box in result.bboxes:
            cls_id = int(box.cls[0])

```

```

class_name = result.names.get(cls_id, str(cls_id))
if target_class_name and class_name.lower() != target_class_name.lower():
    continue
conf = float(box.conf[0])
x1, y1, x2, y2 = [int(v) for v in box.xyxy[0].tolist()]
detections.append(
    {
        "class_id": cls_id,
        "class_name": class_name,
        "confidence": conf,
        "bbox": [x1, y1, x2, y2],
    }
)
return detections

def _detect_weapons(frame: np.ndarray) -> List[Dict[str, Any]]:
    if weapon_model is None:
        return []
    return _extract_detections(weapon_model(frame, conf=WEAPON_CONF_THRESHOLD))

def _detect_humans(frame: np.ndarray) -> List[Dict[str, Any]]:
    if human_model is None:
        return []
    return _extract_detections(human_model(frame, conf=HUMAN_CONF_THRESHOLD),
target_class_name="person")

def _detect_knives(frame: np.ndarray) -> List[Dict[str, Any]]:
    if knife_model is None:
        return []
    detections = _extract_detections(knife_model(frame, conf=KNIFE_CONF_THRESHOLD),
target_class_name="knife")
    for detection in detections:
        detection["risk_weight"] = 5
    return detections

def _risk_level(weapon_detected: bool, anomaly_detected: bool, human_detected: bool) -> str:
    if human_detected and weapon_detected:
        return "critical"
    if human_detected:

```

```

        return "human_intrusion"
    if weapon_detected:
        return "dangerous_object"
    if anomaly_detected:
        return "anomaly"
    return "normal"

def _alert_actions(risk_level: str) -> Dict[str, Any]:
    if risk_level == "critical":
        return {"threat_detected": True, "buzzer_on": True, "display_message": "ALERTE ROUGE"}
    if risk_level == "human_intrusion":
        return {"threat_detected": True, "buzzer_on": True, "display_message": "ALERTE HUMAIN"}
    if risk_level in {"dangerous_object", "anomaly"}:
        return {"threat_detected": True, "buzzer_on": True, "display_message": "ALERTE"}
    return {"threat_detected": False, "buzzer_on": False, "display_message": "Affichage normal"}

@app.get("/health")
def health() -> Dict[str, Any]:
    return {
        "status": "ok",
        "detection_model": DETECTION_MODEL_PATH,
        "anomaly_model": ANOMALY_MODEL_PATH,
        "human_model_enabled": human_model is not None,
        "knife_model_enabled": knife_model is not None,
    }

@app.get("/", response_class=HTMLResponse)
def dashboard() -> str:
    return """
<html><head><title>Security AI Dashboard</title>
<style>body{font-family:Arial;background:#111;color:white;text-
align:center}.card{background:#222;padding:20px;margin:10px;border-radius:10px;display:inline-
block}h1{color:#ff4444}</style>
</head><body>
<h1> 🔥 Security AI Dashboard</h1>
<div class="card">Total Frames: <span id="frames">0</span></div>
<div class="card">Weapons: <span id="weapons">0</span></div>
<div class="card">Anomalies: <span id="anomalies">0</span></div>
<div class="card">Critical Events: <span id="critical">0</span></div>

```

```

<div class="card">Humans: <span id="humans">0</span></div>
<script>
  async function loadStats(){
    const res = await fetch('/stats');
    const data = await res.json();
    document.getElementById('frames').innerText = data.total_frames || 0;
    document.getElementById('weapons').innerText = data.total_weapons || 0;
    document.getElementById('anomalies').innerText = data.total_anomalies || 0;
    document.getElementById('critical').innerText = data.critical_events || 0;
    document.getElementById('humans').innerText = data.total_humans || 0;
  }
  setInterval(loadStats, 1000);
</script></body></html>
"""

```

```

@app.get("/stats")
def get_stats() -> JSONResponse:
    return JSONResponse(content={k: int(v) for k, v in stats.items()})

@app.post("/analyze_frame")
async def analyze_frame(
    frame: UploadFile = File(...),
    x_api_key: Optional[str] = Header(default=None),
) -> Dict[str, Any]:
    _check_api_key(x_api_key)

    raw = await frame.read()
    np_img = np.frombuffer(raw, np.uint8)
    decoded = cv2.imdecode(np_img, cv2.IMREAD_COLOR)
    if decoded is None:
        raise HTTPException(status_code=400, detail="Invalid image file")

    weapon_detections = _detect_weapons(decoded)
    knife_detections = _detect_knives(decoded)
    all_weapon_detections = weapon_detections + knife_detections
    human_detections = _detect_humans(decoded)
    anomaly_score = _compute_anomaly_score(decoded)

    weapon_detected = len(all_weapon_detections) > 0

```

```

anomaly_detected = anomaly_score > ANOMALY_THRESHOLD
human_detected = len(human_detections) > 0
risk_level = _risk_level(weapon_detected, anomaly_detected, human_detected)

stats["total_frames"] += 1
if weapon_detected:
    stats["total_weapons"] += 1
if anomaly_detected:
    stats["total_anomalies"] += 1
if human_detected:
    stats["total_humans"] += 1
if risk_level in {"critical", "human_intrusion"}:
    stats["critical_events"] += 1

timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
filename = f'{SAVE_DIR}/{risk_level}_{timestamp}.jpg'
for detection in all_weapon_detections + human_detections:
    x1, y1, x2, y2 = detection["bbox"]
    color = (0, 0, 255) if detection.get("class_name", "").lower() == "person" else (255, 0, 0)
    cv2.rectangle(decoded, (x1, y1), (x2, y2), color, 2)
    cv2.imwrite(filename, decoded)
event_history.append(
    {
        "time": datetime.datetime.now().isoformat(),
        "risk_level": risk_level,
        "weapon_count": len(all_weapon_detections),
        "humans_detected": len(human_detections),
        "anomaly_score": anomaly_score,
    }
)
return {
    "weapon_detected": weapon_detected,
    "weapon_count": len(all_weapon_detections),
    "weapon_detections": all_weapon_detections,
    "anomaly_detected": anomaly_detected,
    "anomaly_score": anomaly_score,
    "human_detected": human_detected,
    "human_count": len(human_detections),
    "human_detections": human_detections,

```

```

        "risk_level": risk_level,
        **_alert_actions(risk_level),
    }
@app.post("/predict")
async def predict_legacy(frame: UploadFile = File(...), x_api_key: Optional[str] = Header(default=None)) ->
Dict[str, Any]:
    return await analyze_frame(frame=frame, x_api_key=x_api_key)

@app.get("/camera_feed")
def camera_feed() -> Response:
    try:
        response = requests.get(CAMERA_STREAM_URL, stream=True, timeout=5)
        return StreamingResponse(response.raw, media_type="multipart/x-mixed-replace; boundary=frame")
    except Exception as exc:
        return JSONResponse(status_code=500, content={"error": str(exc)})

if __name__ == "__main__":
    import uvicorn
    uvicorn.run("main:app", host="0.0.0.0", port=8000, reload=False)

```