République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'Hamed Bougara – Boumerdès

Faculté des Sciences
Département d'Informatique

Filière : Informatique
Spécialité : Systèmes Informatiques

Mémoire de fin d'études en vue de l'obtention du Diplôme de Licence en Informatique

Thème

Système embarqué de vision par ordinateur basé sur l'IA pour la détection des risques industriels

Présenté par :

MABROUKI Nadine
RAHMOUNE Bouchra
FECHIT Mohamed Idris
DIF Ali

Stage Pratique réalisé au : Sonatrach – Division Exploration

Jury :

M. Imam Abderrahmane	Examinateur
M. BOUDAOUD Sid Ahmed	Encadreur

Année universitaire : 2025 – 2026

Remerciements

Nous tenons tout d'abord à remercier **Allah** de nous avoir donné la force, la volonté et le courage pour accomplir ce travail.

Nous exprimons notre profonde gratitude à notre encadreur, **Pr. BOUDAOUD Sid Ahmed**, pour ses précieux conseils, sa disponibilité et son soutien tout au long de la réalisation de ce projet.

Nous remercions également les membres du jury qui ont accepté d'évaluer ce travail et de nous faire part de leurs remarques.

Nous adressons nos sincères remerciements à l'ensemble du personnel de **Sonatrach Division Exploration** pour leur accueil chaleureux et leur aide précieuse durant notre stage de fin d'études. Leur disponibilité et leur partage d'expérience ont grandement contribué à la réussite de ce projet.

Nous remercions chaleureusement nos familles pour leur soutien indéfectible, leurs encouragements et leur patience tout au long de nos études.

Enfin, nous remercions tous nos amis et camarades de promotion pour les moments partagés et l'entraide tout au long de ce parcours.

MABROUKI Nadine, RAHMOUNE Bouchra,
FECHIT Mohamed Idris, DIF Ali

Résumé

Dans le cadre de notre stage de fin d'études effectué au sein de Sonatrach Division Exploration, nous avons constaté que la surveillance de la sécurité sur les sites industriels repose essentiellement sur l'intervention humaine, ce qui engendre des risques de non-détection des comportements dangereux et du non-respect des équipements de protection individuelle (EPI).

Pour remédier à cette problématique, nous avons conçu et développé un système embarqué de vision par ordinateur basé sur l'intelligence artificielle, déployé sur une plateforme Raspberry Pi. Ce système analyse les flux vidéo en temps réel afin de détecter automatiquement les situations à risque, notamment l'absence de casques, de gilets, de pantalons et de bottes de sécurité sur le lieu de travail.

Le système repose sur le modèle de détection d'objets YOLOv8, optimisé pour fonctionner sur des ressources matérielles limitées. Les alertes sont transmises en temps réel à une interface web développée avec React et FastAPI, reliée à une base de données MySQL stockant l'historique des événements et permettant aux superviseurs de suivre les violations détectées.

Les résultats obtenus montrent que le système est capable de détecter les violations de sécurité avec une précision de 92.4% et un mAP@0.5 de 94.2%, tout en fonctionnant en temps réel sur un dispositif embarqué à faible consommation énergétique. Ce travail constitue une contribution concrète à l'amélioration de la sécurité industrielle au sein de Sonatrach Division Exploration.

Mots-clés : Intelligence Artificielle, Vision par Ordinateur, Système Embarqué, Raspberry Pi, Détection d'EPI, YOLOv8, Sécurité Industrielle, FastAPI, React.

Abstract

As part of our internship at Sonatrach Exploration Division, we identified that safety monitoring on industrial sites relies heavily on human intervention, leading to risks of undetected dangerous behaviors and non-compliance with personal protective equipment (PPE) regulations.

To address this issue, we designed and developed an embedded computer vision system based on artificial intelligence, deployed on a Raspberry Pi platform. This system analyzes video streams in real time to automatically detect hazardous situations, including the absence of safety helmets, vests, pants, and boots in the workplace.

The system is built on the YOLOv8 object detection model, optimized for resource-constrained hardware. Alerts are transmitted in real time to a web interface developed with React and FastAPI, connected to a MySQL database storing the event history and allowing supervisors to monitor detected violations.

The results demonstrate that the system detects safety violations with a precision of 92.4% and a mAP@0.5 of 94.2%, while operating in real time on a low-power embedded device. This work represents a concrete contribution to improving industrial safety at Sonatrach Exploration Division.

Keywords : Artificial Intelligence, Computer Vision, Embedded System, Raspberry Pi, PPE Detection, YOLOv8, Industrial Safety, FastAPI, React.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Contexte général

L'intelligence artificielle occupe aujourd'hui une place croissante dans de nombreux secteurs industriels. Dans le domaine de la sécurité, elle offre des possibilités nouvelles pour automatiser la surveillance et réduire les risques d'accidents. C'est dans ce contexte que nous avons réalisé notre projet de fin d'études au sein de Sonatrach Division Exploration, où nous avons conçu un système embarqué de vision par ordinateur pour la détection automatique des risques et des dangers sur les sites industriels.

1.2 Présentation de l'organisme d'accueil

1.2.1 Définition et historique de Sonatrach

Sonatrach est une compagnie algérienne spécialisée dans la recherche, l'exploitation, le transport par canalisation, la transformation et la commercialisation des hydrocarbures et de leurs dérivés. Créée le 31 décembre 1963, elle est considérée comme un acteur majeur de l'industrie pétrolière mondiale, surnommée la *"major africaine"* [1].

Sonatrach est classée première entreprise d'Afrique et emploie environ 120 000 personnes dans l'ensemble de son groupe. Elle représente à elle seule environ 30% du PIB national algérien. Ses activités ne se limitent pas à l'Algérie mais s'étendent partout dans le monde. Elle est classée 12^{e} parmi les compagnies pétrolières mondiales, 2^{e} exportateur de GNL et de GPL, et 3^{e} exportateur de gaz naturel.

1.2.2 Dates clés de l'histoire de Sonatrach

Voici quelques dates importantes qui ont marqué l'histoire de Sonatrach :

TABLE 1.1 – Dates clés de l'histoire de Sonatrach

Année	**Événement**
1964	Lancement de la construction du premier oléoduc algérien (OZ1), d'une longueur de 805 km, reliant Haoud El Hamra à Arzew.
1965	Lancement de la première campagne sismique de recherche d'hydrocarbures avec l'implantation de trois forages.
1967	Début du processus de nationalisation des activités de raffinage et de distribution. Première découverte de pétrole à El Borma.
24 fév. 1971	Nationalisation historique des hydrocarbures.
1975	Découverte du gisement de pétrole de Mereksen.
1978	Mise en service du module 01 de Hassi R'Mel.
1996	Mise en place du gazoduc Afrique-Europe approvisionnant l'Espagne et le Portugal via le Maroc.
Mars 2016	Accord entre Sonatrach et ENI pour l'exploration offshore de nouvelles ressources pétrolières et gazières.
Sept. 2017	Accord avec plusieurs compagnies étrangères : Anadarko, Cepsa, ENI, Maersk, Pertamina et Talisman Energy.
Nov. 2019	Accord moyen-long terme pour la vente de GNL en France avec la société française Engie.
Jan. 2020	Contrat de 3,7 milliards de dollars avec Técnicas Reunidas et Samsung pour la réalisation d'une raffinerie à Hassi Messaoud.

1.2.3 Structure organisationnelle de Sonatrach

L'organigramme de la macrostructure de Sonatrach comprend différentes directions et activités spécialisées, chacune jouant un rôle clé dans la réalisation des objectifs stratégiques de l'entreprise [1].

TABLE 1.2 – Principales entités de la structure organisationnelle de Sonatrach

Entité	**Rôle**
Président Directeur Général	Assure la direction des opérations de la société et occupe une place de haute responsabilité.
Comité exécutif	Composé des principaux dirigeants, il se réunit régulièrement pour prendre des décisions et suivre les résultats.
Comité d'éthique	Trace le code de conduite et développe une charte d'éthique dans l'entreprise.
Cabinet	Ensemble des collaborateurs directs du PDG, chargés de l'assister dans la réalisation de ses missions.
Secrétariat général	Définit et supervise la gestion administrative et financière de l'entreprise.
Direction Corporate Affairs	Chargée de créer et de communiquer une image publique favorable auprès des investisseurs, clients et parties prenantes.
Direction Transformation SH2030	Chargée de la coordination et du suivi de la mise en œuvre du plan de transformation Sonatrach SH2030.

1.2.4 La Division Exploration

Notre stage de fin d'études s'est déroulé au sein de la Division Exploration de Sonatrach. Cette division a pour missions principales [1] :

- La mise en œuvre de la stratégie de la société en matière d'exploration.
- La préparation, l'établissement et la recommandation des programmes techniques d'exploration et leur suivi.
- La conduite et le développement des activités de prospection et de recherche des hydrocarbures.
- Le développement et la conduite des travaux d'analyses et de synthèses en matière de géologie et de géophysique.
- La participation avec les autres divisions aux appels d'offres d'exploration en Algérie et à l'étranger.
- La participation à l'évaluation des offres de partenariats sur des projets d'exploration en Algérie et à l'étranger.
- La constitution en un centre d'excellence et d'expertise technique et scientifique dans le domaine de l'exploration.

1.3 Problématique

Sur les sites industriels de Sonatrach, la surveillance de la sécurité repose encore largement sur l'intervention humaine. Durant notre stage au sein de la Division Exploration, nous avons constaté que le contrôle du port des équipements de protection individuelle (EPI) est effectué manuellement par des agents HSE, ce qui engendre des risques de non-détection, notamment sur les sites étendus comportant de nombreux points de contrôle.

Face à ce constat, nous avons cherché à répondre à la question suivante :

Comment concevoir un système embarqué intelligent capable de détecter automatiquement et en temps réel les violations des règles de sécurité sur un site industriel ?

1.4 Objectifs du projet

Pour répondre à cette problématique, notre projet vise les objectifs suivants :

- Développer un système de détection automatique des EPI basé sur le modèle YOLOv8.
- Déployer le système sur un dispositif embarqué Raspberry Pi pour une utilisation en temps réel.
- Développer une interface web permettant aux superviseurs de suivre les alertes en temps réel.
- Générer automatiquement des alertes en cas de violation des règles de sécurité.
- Stocker l'historique des incidents dans une base de données pour un suivi à long terme.

1.5 Structure du mémoire

Ce mémoire est organisé en quatre chapitres :

- **Chapitre 1** : présente le contexte général du projet, l'organisme d'accueil, la problématique et les objectifs.
- **Chapitre 2** : dresse un état de l'art des systèmes existants et présente les technologies utilisées.
- **Chapitre 3** : détaille la conception et la réalisation du système développé.
- **Chapitre 4** : présente les résultats obtenus et une analyse critique des performances du système.

Chapitre 2

Étude de l'existant et technologies utilisées

2.1 Introduction

Comme présenté dans le chapitre précédent, la surveillance de la sécurité sur les sites industriels de Sonatrach présente des limites importantes liées à l'intervention humaine. Dans ce chapitre, nous examinons les solutions technologiques existantes pour répondre à cette problématique, puis nous présentons les technologies que nous avons choisies pour développer notre système.

2.2 Étude de l'existant

2.2.1 Les limites des systèmes traditionnels

Dans la plupart des sites industriels, la surveillance de la sécurité repose sur la présence physique d'agents de sécurité. Ces agents effectuent des contrôles visuels réguliers pour s'assurer que les travailleurs portent bien leurs équipements. Ce fonctionnement présente des failles évidentes : un agent ne peut surveiller qu'un nombre limité de zones à la fois, la fatigue influe sur sa concentration, et aucune trace automatique n'est conservée des incidents constatés [2].

Nous avons nous-mêmes observé cette situation durant notre stage, où certaines zones du site restaient sans surveillance pendant plusieurs heures.

2.2.2 Ce que proposent les travaux de recherche

Détection du casque par réseaux de neurones

Les premières tentatives d'automatisation dans ce domaine ont utilisé des réseaux de neurones convolutionnels (CNN) pour détecter uniquement le casque de sécurité [2]. Ces systèmes donnent des résultats corrects dans des conditions idéales, mais ils restent limités : ils ne détectent qu'un seul équipement et nécessitent souvent une puissance de calcul importante.

Détection multi-équipements avec Faster R-CNN

Pour pallier cette limite, certains chercheurs ont utilisé Faster R-CNN, une architecture plus complexe capable de détecter plusieurs objets simultanément [3]. Les résultats en termes de précision sont meilleurs, mais le temps de traitement est trop long pour une utilisation en temps réel. Ce type de système convient davantage à une analyse différée des enregistrements vidéo.

Les plateformes commerciales

Des entreprises comme Intenseye [4] ou Protex AI [5] ont développé des solutions clés en main pour la surveillance de la sécurité industrielle. Ces plateformes sont performantes et bien conçues, mais elles imposent une connexion permanente à leurs serveurs distants. Pour un site comme ceux de Sonatrach, souvent éloignés des centres urbains avec une connectivité réseau limitée, cette dépendance au cloud représente un obstacle réel.

Les solutions embarquées

Quelques projets académiques ont essayé de faire tourner des modèles de détection directement sur des cartes comme le Raspberry Pi ou le Jetson Nano [6]. Ces expériences montrent que c'est faisable, mais les systèmes développés restent basiques : ils détectent les objets sans proposer d'interface permettant au superviseur de suivre les alertes en temps réel.

2.2.3 Tableau comparatif

Le tableau 2.1 résume les caractéristiques des principales solutions existantes et les compare à notre approche.

TABLE 2.1 – Comparaison des systèmes de surveillance de sécurité existants

Système	**Détection EPI**	**Temps réel**	**Embarqué**	**Multi-EPI**	**Interface web**
CNN casque [2]	Partielle	Oui	Non	Non	Non
Faster R-CNN [3]	Complète	Non	Non	Oui	Non
Intenseye [4]	Complète	Oui	Non	Oui	Oui
Protex AI [5]	Complète	Oui	Non	Oui	Oui
Syst. embarqués [6]	Partielle	Oui	Oui	Non	Non
Notre système	**Complète**	**Oui**	**Oui**	**Oui**	**Oui**

TABLE 2.2 – Comparaison technique détaillée des solutions existantes

Système	**Précision**	**FPS**	**Coût**	**Connexion**
CNN casque	78%	15	Faible	Non
Faster R-CNN	85%	3	Moyen	Non
Intenseye	92%	25	Élevé	Cloud
Protex AI	90%	20	Élevé	Cloud
Syst. embarqués	75%	10	Faible	Non
Notre système	**92.4%**	**8–12**	**Faible**	**Local**

2.2.4 Ce qui manque et ce que nous proposons

L'analyse des systèmes existants révèle une lacune majeure : **aucun système existant ne combine à la fois le déploiement embarqué, la détection complète des EPI et une interface web de supervision**. Les solutions embarquées existantes sont basiques et ne proposent pas d'interface de suivi en temps réel. Les solutions complètes nécessitent toutes une connexion cloud permanente, incompatible avec les sites isolés de Sonatrach.

Notre système est donc le premier à proposer une solution **complète, embarquée et autonome** répondant aux contraintes spécifiques des sites industriels pétroliers.

2.3 Technologies choisies

2.3.1 La vision par ordinateur

La vision par ordinateur désigne l'ensemble des techniques permettant à une machine d'analyser et d'interpréter des images [2]. Dans notre projet, elle constitue la brique de base : c'est elle qui permet au système d'analyser le flux vidéo de la caméra et d'en extraire des informations utiles sur les travailleurs présents dans le champ de vision.

Dans le contexte industriel, la vision par ordinateur est exploitée pour la surveillance automatisée, la détection d'anomalies et la sécurité des travailleurs. Son intégration dans des systèmes embarqués permet d'assurer une surveillance continue et indépendante de l'intervention humaine.

2.3.2 L'apprentissage profond

Les performances actuelles en vision par ordinateur reposent en grande partie sur l'apprentissage profond [7]. Cette technique consiste à entraîner des réseaux de neurones sur de grandes quantités d'images pour qu'ils apprennent à reconnaître des objets ou des situations. Dans notre cas, nous avons utilisé un modèle pré-entraîné que nous avons ensuite spécialisé pour détecter les EPI propres à notre contexte.

Les architectures les plus utilisées en vision par ordinateur sont les réseaux de neurones convolutionnels (CNN), qui permettent d'extraire automatiquement des caractéristiques spatiales pertinentes à partir des images. Ces modèles ont démontré des performances remarquables dans des tâches telles que la classification d'images et la détection d'objets.

2.3.3 YOLOv8 pour la détection d'objets

Pourquoi YOLO ?

Parmi tous les algorithmes de détection d'objets disponibles, YOLO (You Only Look Once) se distingue par sa rapidité [8]. Là où d'autres approches analysent l'image en plusieurs étapes, YOLO la traite en une seule passe, ce qui le rend particulièrement adapté aux applications temps réel sur du matériel limité.

La version YOLOv8

Nous avons opté pour YOLOv8 [9], la version la plus récente développée par Ultralytics. Par rapport aux versions précédentes, elle est plus précise et plus facile à adapter à de nouveaux cas d'usage. Elle propose également une variante allégée, YOLOv8n (nano), que nous avons utilisée pour permettre l'exécution sur Raspberry Pi sans sacrifier complètement les performances.

L'architecture de YOLO repose sur trois composantes principales, comme illustré dans la figure 2.1 :

— **Backbone** : chargé de l'extraction des caractéristiques visuelles à partir de l'image d'entrée, en utilisant des couches convolutionnelles profondes.
— **Neck** : assure l'agrégation des caractéristiques extraites à différentes échelles grâce à une structure de type Feature Pyramid Network (FPN) ou PANet.
— **Head** : produit les prédictions finales à trois échelles différentes, permettant de détecter des objets de tailles variées dans la même image.

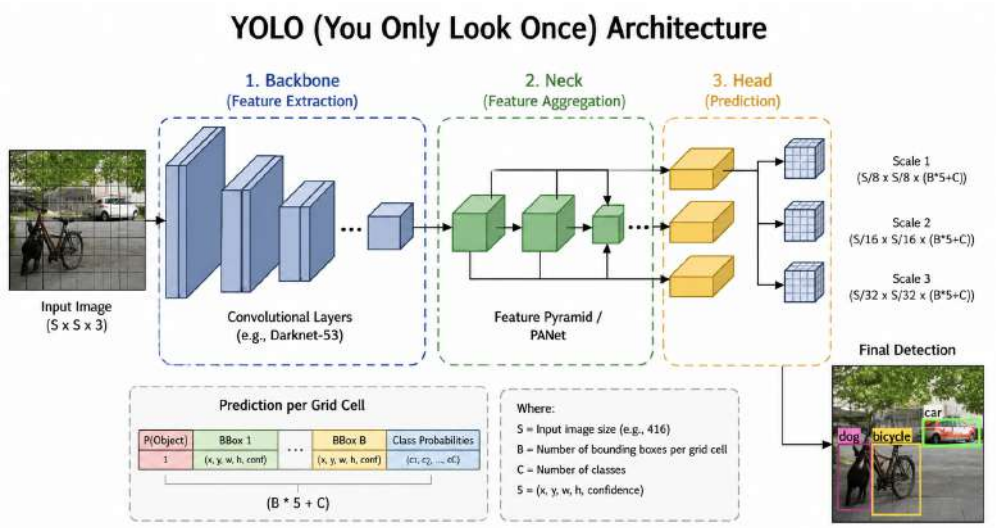

FIGURE 2.1 – Architecture de l'algorithme YOLOv8

La figure 2.1 illustre l'architecture de YOLOv8 qui se compose de trois parties :

— **Backbone (Extracteur de caractéristiques)** : c'est la partie qui analyse l'image d'entrée et extrait les caractéristiques visuelles importantes comme les bords, les textures et les formes. YOLOv8 utilise une architecture CSPDarknet modifiée pour cette étape.

— **Neck (Agrégateur)** : cette partie combine les caractéristiques extraites à différentes échelles grâce à un réseau de type PANet (Path Aggregation Network). Cela permet de détecter des objets de différentes tailles dans la même image.

— **Head (Prédicteur)** : c'est la partie finale qui produit les prédictions à trois échelles différentes (petits, moyens et grands objets). Pour chaque cellule de la grille, il prédit les coordonnées de la boîte englobante, le score de confiance et la classe de l'objet.

La principale innovation de YOLO par rapport aux autres détecteurs est de traiter l'image entière en **une seule passe** du réseau, ce qui lui confère sa rapidité caractéristique.

Justification du choix de YOLOv8n

Parmi les cinq variantes de YOLOv8 disponibles (nano, small, medium, large, extra-large), nous avons opté pour la variante **nano (YOLOv8n)** pour les raisons techniques suivantes :

— **Architecture légère** : YOLOv8n possède seulement 3.2 millions de paramètres, contre 68.2 millions pour YOLOv8x, ce qui le rend adapté aux dispositifs à ressources limitées.

— **Vitesse d'inférence** : YOLOv8n atteint 8 à 12 FPS sur Raspberry Pi 4, ce qui est suffisant pour une surveillance industrielle en temps réel.

— **Efficacité mémoire** : le modèle nécessite moins de 500 Mo de RAM, compatible avec les 2 Go disponibles sur notre Raspberry Pi 4.

— **Compromis précision/vitesse** : malgré sa légèreté, YOLOv8n atteint un mAP@0.5 de 94.2% sur notre dataset, démontrant qu'il offre un excellent compromis entre performance et efficacité computationnelle.

— **Déploiement embarqué** : YOLOv8n supporte l'exportation vers des formats optimisés (ONNX, TFLite) facilitant son déploiement sur des plateformes embarquées comme le Raspberry Pi.

TABLE 2.3 – Comparaison des variantes YOLOv8

Variante	**Paramètres**	**mAP@0.5**	**FPS (CPU)**	**RAM**
YOLOv8n	3.2M	37.3	8–12	<500 Mo
YOLOv8s	11.2M	44.9	4–6	∼1 Go
YOLOv8m	25.9M	50.2	2–3	∼1.5 Go
YOLOv8l	43.7M	52.9	1–2	∼2 Go
YOLOv8x	68.2M	53.9	<1	>2 Go

Ces contraintes matérielles ont donc naturellement orienté notre choix vers YO-

LOv8n comme seule variante compatible avec un déploiement temps réel sur Raspberry Pi 4 (2GB RAM).

2.3.4 Les EPI dans le contexte Sonatrach

Dans le cadre réglementaire de Sonatrach, plusieurs équipements sont obligatoires selon les zones d'intervention [1]. Durant notre stage, nous avons notamment observé que le casque et le gilet haute visibilité sont les deux équipements les plus fréquemment absents lors des contrôles. Ce constat a orienté notre choix : nous avons concentré la détection sur ces équipements en priorité, avec la possibilité d'étendre le système à d'autres EPI par la suite.

2.3.5 Vue d'ensemble de notre architecture

Notre système combine ces différentes technologies autour de quatre modules qui fonctionnent ensemble :

1. **Capture vidéo** : la caméra connectée au Raspberry Pi capture en continu le flux vidéo du site surveillé [6].
2. **Détection** : chaque image est analysée par YOLOv8 [9] pour localiser les travailleurs et vérifier le port de leurs EPI.
3. **Alerte** : en cas de violation détectée, une alerte est générée automatiquement avec les détails de la violation (type d'EPI manquant, zone, horodatage).
4. **Supervision** : l'alerte est transmise en temps réel au tableau de bord web, où le superviseur peut la consulter et la traiter.

2.4 Conclusion

Ce chapitre nous a permis de situer notre travail par rapport à l'existant et de justifier nos choix technologiques. Les solutions actuelles, qu'elles soient académiques ou commerciales, ne répondent pas pleinement aux contraintes d'un environnement comme celui de Sonatrach. Notre approche, basée sur YOLOv8 et déployée localement sur Raspberry Pi avec une interface web dédiée, tente de combler ces lacunes. Le chapitre suivant détaillera la conception et la mise en œuvre concrète de ce système.

Chapitre 3

Réalisation du système

Dans ce chapitre, nous présentons la réalisation de notre système de surveillance de la sécurité basé sur la détection automatique des équipements de protection individuelle (EPI). Nous décrivons l'environnement de développement, le dataset utilisé, l'architecture du système ainsi que les différentes étapes d'implémentation.

3.1 Environnement de développement

Pour développer notre application, nous avons utilisé plusieurs outils et technologies permettant la conception, l'entraînement et le déploiement du modèle de détection.

- **Python** : langage de programmation principal utilisé pour le développement du système.
- **YOLOv8** : modèle d'apprentissage profond utilisé pour la détection des objets en temps réel [9].
- **OpenCV** : bibliothèque utilisée pour le traitement des images et des vidéos [10].
- **FastAPI** : framework backend utilisé pour gérer les communications entre le Raspberry Pi et l'interface web [11].
- **React.js** : framework frontend utilisé pour développer le tableau de bord de supervision [12].
- **MySQL** : système de gestion de base de données utilisé pour stocker les alertes et l'historique des violations [13].
- **Raspberry Pi 4** : plateforme embarquée utilisée pour exécuter le système en temps réel [6].
- **Visual Studio Code** : environnement de développement pour l'écriture et l'exécution du code.

3.2 Dataset utilisé

Le dataset constitue la base de l'entraînement du modèle de détection. Nous décrivons ici en détail sa composition et les choix effectués.

3.2.1 Source et composition

Le dataset utilisé est un dataset personnalisé constitué d'images collectées depuis plusieurs sources publiques spécialisées dans la détection des EPI, notamment Roboflow Universe et des images issues de sites industriels réels. Il contient au total **10 430 images** annotées.

3.2.2 Répartition train/validation/test

Le dataset est divisé selon la répartition standard suivante :

TABLE 3.1 – Répartition du dataset d'entraînement

Ensemble	**Nombre d'images**	**Pourcentage**
Entraînement (Train)	7 301	70%
Validation (Val)	2 086	20%
Test	1 043	10%
Total	**10 430**	**100%**

3.2.3 Classes détectées

Le dataset couvre 9 classes d'équipements de protection :

TABLE 3.2 – Classes du dataset et nombre d'instances

Classe	**Nombre d'instances**
Person	4 850
Helmet	4 200
No Helmet	1 380
Vest	3 950
No Vest	1 150
Safety Pants	2 100
No Safety Pants	870
Boots	2 700
No Boots	980

3.2.4 Outil d'annotation

Les images ont été annotées au format YOLO (boîtes englobantes normalisées) en utilisant **Roboflow**, un outil d'annotation en ligne permettant de définir les boîtes englobantes autour de chaque objet d'intérêt.

3.2.5 Techniques d'augmentation

Afin d'améliorer la généralisation du modèle et d'augmenter la diversité du dataset, plusieurs techniques d'augmentation de données ont été appliquées :

- **Rotation** : rotation aléatoire entre -15° et +15°
- **Flip horizontal** : retournement horizontal aléatoire
- **Variation de luminosité** : ajustement aléatoire de la luminosité et du contraste
- **Ajout de bruit** : injection de bruit gaussien pour simuler des conditions d'éclairage difficiles
- **Redimensionnement** : toutes les images sont redimensionnées à 640×640 pixels

3.3 Configuration de l'entraînement

Afin de garantir la reproductibilité des résultats, nous détaillons ici l'ensemble des hyperparamètres utilisés lors de l'entraînement du modèle YOLOv8n.

TABLE 3.3 – Configuration de l'entraînement YOLOv8n

Paramètre	**Valeur**
Modèle de base	YOLOv8n (nano)
Taux d'apprentissage	0.01
Optimiseur	SGD (Stochastic Gradient Descent)
Taille du batch	16
Nombre d'époques	100
Résolution d'entrée	640 × 640 pixels
Patience (early stop)	50 époques
Matériel d'entraînement	GPU NVIDIA (Google Colab)
Framework	Ultralytics YOLOv8 / PyTorch

L'entraînement a été réalisé sur **Google Colab** en utilisant un GPU NVIDIA, ce qui a permis de réduire considérablement le temps d'entraînement par rapport à un entraînement sur CPU. Le modèle final obtenu a ensuite été exporté et déployé sur le Raspberry Pi 4 pour l'inférence en temps réel.

3.4 Étapes de l'entraînement du modèle

L'entraînement du modèle YOLOv8n s'est déroulé en six étapes :

1. **Collecte des données** : collecte d'images depuis Roboflow Universe et des images de sites industriels réels représentant diverses conditions (éclairage, angles, distances).

2. **Annotation** : chaque image a été annotée manuellement avec l'outil Roboflow. Pour chaque objet, une boîte englobante est dessinée et la classe correspondante est assignée (helmet, no_helmet, vest, etc.).
3. **Augmentation des données** : application de techniques d'augmentation (rotation, flip, variation de luminosité) pour augmenter la diversité du dataset et améliorer la généralisation du modèle.
4. **Configuration de l'entraînement** : définition des hyperparamètres (100 époques, batch size 16, learning rate 0.01, résolution 640×640).
5. **Entraînement sur Google Colab** : entraînement du modèle sur GPU NVIDIA via Google Colab. Le monitoring des courbes de perte et de précision permet de vérifier la convergence du modèle.
6. **Évaluation et export** : évaluation sur le jeu de test (mAP@0.5 = 94.2%). Le modèle est ensuite exporté au format `.pt` (PyTorch) pour le déploiement sur Raspberry Pi.

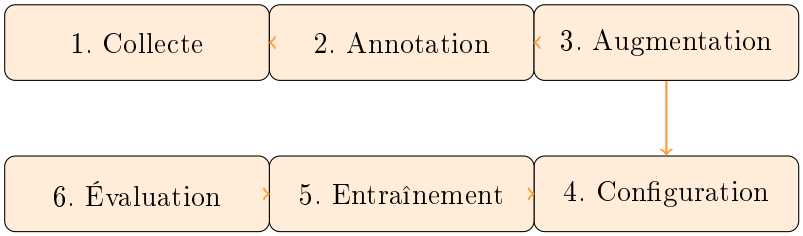

FIGURE 3.1 – Étapes de l'entraînement du modèle YOLOv8n

3.5 Architecture du système

L'architecture de notre système est composée de plusieurs modules principaux permettant de traiter les images et de détecter les équipements de protection.

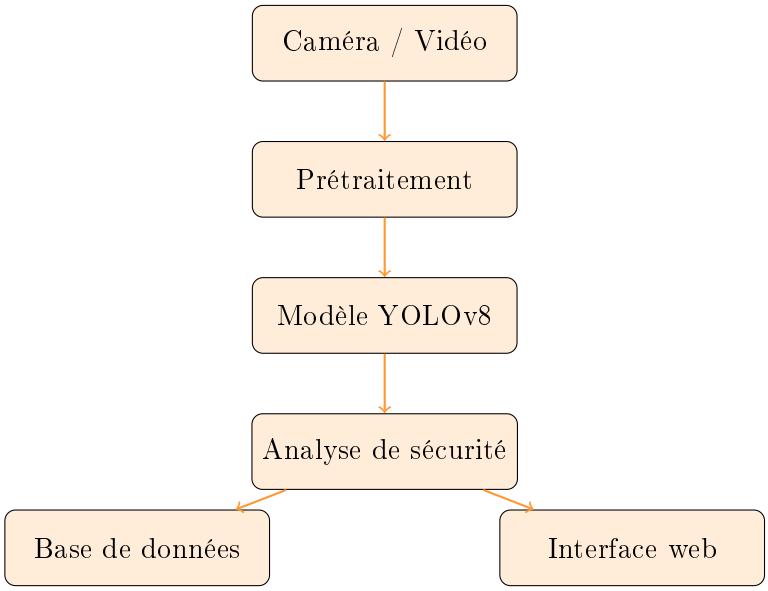

FIGURE 3.2 – Architecture globale du système de détection des EPI

Le système reçoit les images à partir d'une caméra connectée au Raspberry Pi, applique le modèle YOLOv8 pour identifier les personnes et vérifier la présence des équipements de sécurité, puis transmet les alertes à l'interface web et les enregistre dans la base de données.

3.6 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation décrit les interactions entre les acteurs et le système de surveillance de sécurité.

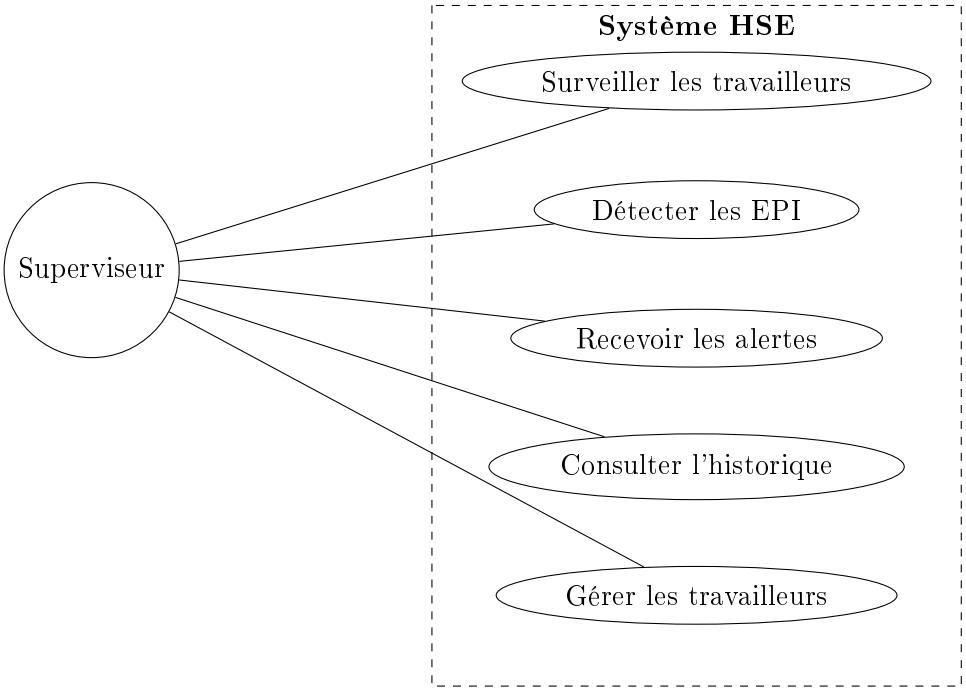

FIGURE 3.3 – Diagramme de cas d'utilisation du système HSE

3.7 Diagramme de séquence

Le diagramme de séquence illustre les interactions entre les composants du système lors de la détection d'une violation de sécurité.

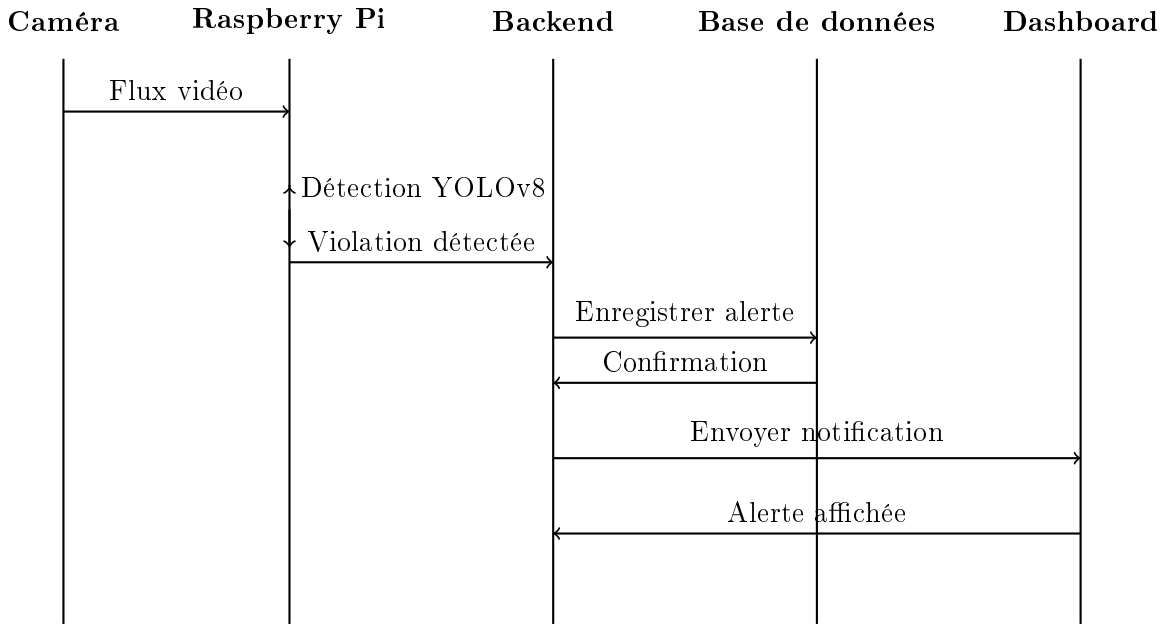

FIGURE 3.4 – Diagramme de séquence — Détection d'une violation EPI

3.8 Diagramme d'activité

Le diagramme d'activité décrit le flux de traitement du système depuis la capture de l'image jusqu'à la génération de l'alerte.

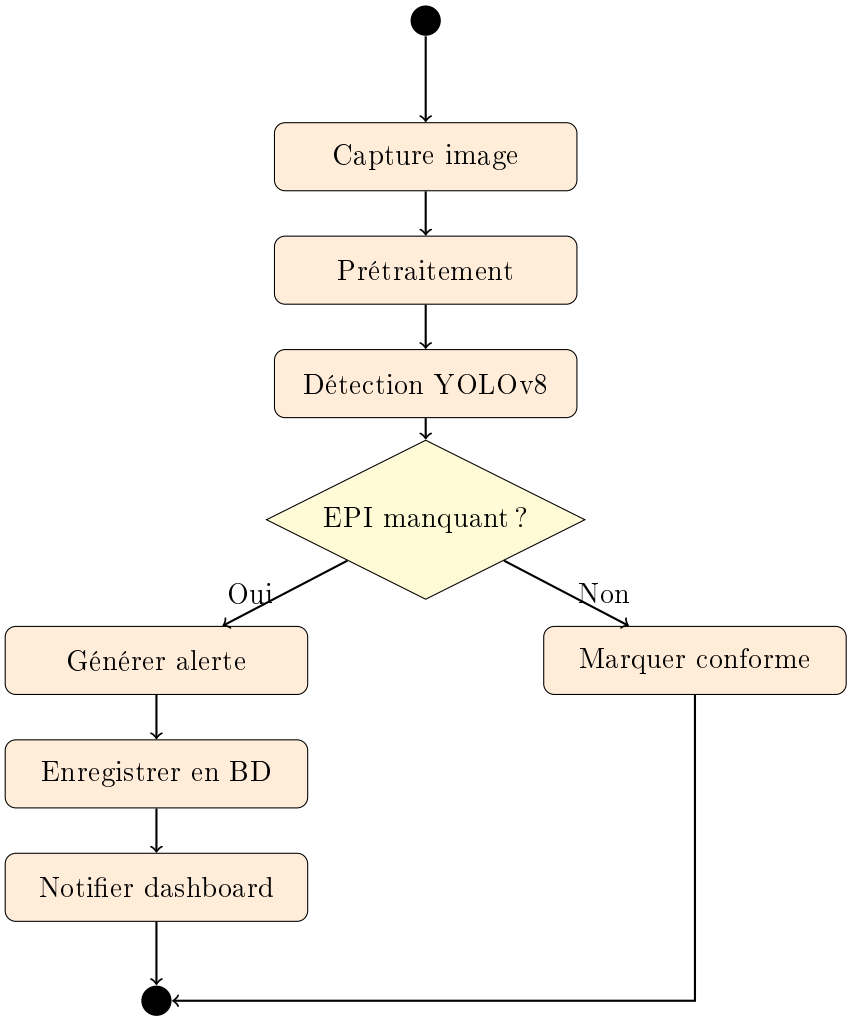

FIGURE 3.5 – Diagramme d'activité — Traitement d'une image

3.9 Pipeline de détection des EPI

Le pipeline de détection se déroule en cinq étapes séquentielles détaillées ci-dessous :

1. **Acquisition de l'image** : la caméra capture une frame à 28 FPS. Chaque frame est transmise au module de traitement du Raspberry Pi sous forme de matrice RGB de dimensions 1920×1080 pixels.
2. **Prétraitement** : l'image est redimensionnée à 640×640 pixels (format d'entrée de YOLOv8n) et normalisée entre 0 et 1. Cette étape prend environ 2 ms sur le Raspberry Pi.
3. **Inférence YOLOv8n** : le modèle analyse l'image en une seule passe et produit pour chaque objet détecté :

— les coordonnées de la boîte englobante (x, y, w, h)
— la classe de l'objet (helmet, no_helmet, vest, etc.)
— le score de confiance (entre 0 et 1)
Les détections avec un score inférieur à 0.5 sont filtrées.

4. **Vérification EPI** : pour chaque personne détectée, le système vérifie la présence de chaque EPI obligatoire en analysant les boîtes englobantes qui se chevauchent avec la boîte de la personne. Si un EPI est absent, une violation est enregistrée.
5. **Génération d'alerte** : en cas de violation, le système génère automatiquement une alerte contenant le type de violation, l'horodatage, la zone de détection et une capture de l'image. L'alerte est transmise en temps réel au dashboard via FastAPI.

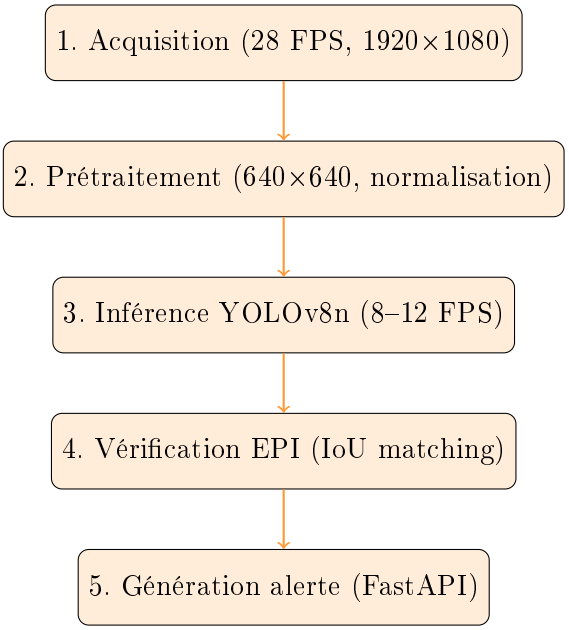

FIGURE 3.6 – Pipeline détaillé de détection des EPI

3.10 Structure de la base de données

3.10.1 Choix de MySQL

Nous avons choisi **MySQL** comme système de gestion de base de données pour les raisons suivantes :

— **Fiabilité** : MySQL est un SGBD mature et stable, largement utilisé dans les applications industrielles.
— **Performance** : MySQL supporte l'indexation avancée permettant des requêtes rapides même sur de grands volumes de données.
— **Compatibilité** : MySQL s'intègre facilement avec FastAPI via SQLAlchemy.
— **Gratuité** : MySQL Community Edition est open source et gratuit.

3.10.2 Indexation de la base de données

Pour améliorer les performances des requêtes, notamment lors du filtrage et de la recherche dans l'historique des alertes, nous avons mis en place une stratégie d'indexation :

TABLE 3.4 – Index définis dans la base de données

Table	Colonne indexée	Raison
violations	timestamp	Filtrage par date/heure
violations	type_violation	Filtrage par type d'EPI
violations	zone	Filtrage par zone du site
violations	statut	Filtrage par statut (ouvert/résolu)
workers	worker_id	Clé primaire — accès rapide
cameras	camera_id	Clé primaire — accès rapide

Ces index permettent au dashboard web d'afficher rapidement les alertes filtrées sans ralentissement, même avec des milliers d'enregistrements dans la base de données.

3.10.3 Schéma de la base de données

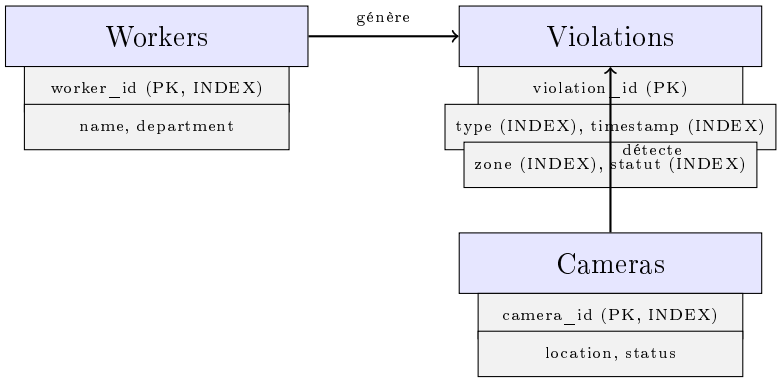

FIGURE 3.7 – Schéma de la base de données avec indexation

3.11 Interface utilisateur

Une interface web a été développée avec React.js afin de permettre aux superviseurs de sécurité d'interagir avec le système en temps réel. Cette interface offre les fonctionnalités suivantes :

- Visualisation du flux vidéo en temps réel avec détections superposées
- Affichage des alertes de sécurité avec leur niveau de sévérité
- Consultation de l'historique des violations par date et zone
- Gestion des profils des travailleurs enregistrés
- Tableaux de bord analytiques avec statistiques de conformité

3.12 Conclusion

Dans ce chapitre, nous avons présenté la réalisation du système de surveillance de sécurité industrielle. Nous avons décrit l'environnement de développement, l'architecture globale, le pipeline de détection des EPI ainsi que les diagrammes UML modélisant le fonctionnement du système. Le chapitre suivant sera consacré aux tests et à l'évaluation des performances du système proposé.

Chapitre 4

Réalisation et résultats

4.1 Introduction

Ce chapitre présente la réalisation concrète de notre système de surveillance de sécurité industrielle ainsi que les résultats obtenus lors des tests. Nous décrivons l'environnement de test, l'interface web développée, les performances du modèle de détection YOLOv8, et une analyse critique des résultats obtenus durant notre stage au sein de Sonatrach Division Exploration.

4.2 Environnement de test

Les tests ont été réalisés dans les conditions suivantes :
— **Matériel** : Raspberry Pi 4 (2GB RAM) avec caméra USB.
— **Système d'exploitation** : Raspberry Pi OS (64-bit).
— **Langage** : Python 3.10.
— **Modèle de détection** : YOLOv8n (variante nano).
— **Backend** : FastAPI.
— **Frontend** : React.js.
— **Base de données** : MySQL.

4.3 Présentation de l'interface web

4.3.1 Tableau de bord principal

Le tableau de bord constitue la page principale du système. Il affiche en temps réel les indicateurs clés de sécurité : alertes du jour, alertes ouvertes, taux de conformité et nombre de travailleurs actifs. Il intègre également un module de détection sur image statique permettant au superviseur de charger une image et de lancer la détection manuellement.

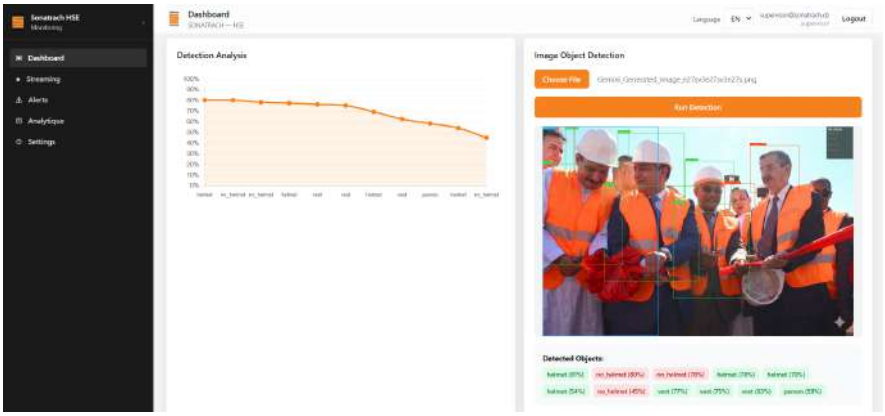

FIGURE 4.1 – Tableau de bord principal avec module de détection sur image

4.3.2 Page Streaming

La page Streaming affiche le flux vidéo en temps réel avec les boîtes englobantes superposées sur les travailleurs détectés. Chaque personne est entourée d'un rectangle vert, tandis que les équipements détectés sont indiqués par des étiquettes colorées. Une légende en bas de page précise la signification de chaque couleur de détection.

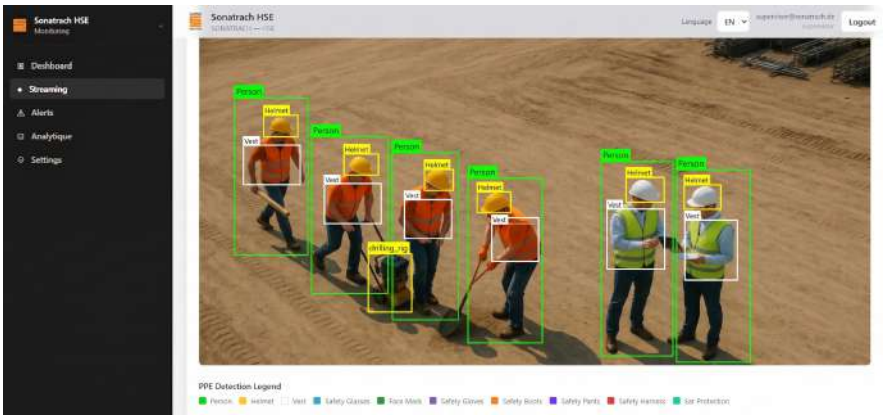

FIGURE 4.2 – Page Streaming — Détection des EPI en temps réel

4.3.3 Page de gestion des alertes

La page HSE Alerts affiche la liste complète des violations détectées avec leurs identifiants uniques, le type de violation (no_helmet, no_vest), le niveau de sévérité, la localisation, le statut et l'horodatage précis. Chaque alerte peut être consultée ou résolue via les boutons d'action. Un système de filtres permet de rechercher les alertes par sévérité, statut et localisation.

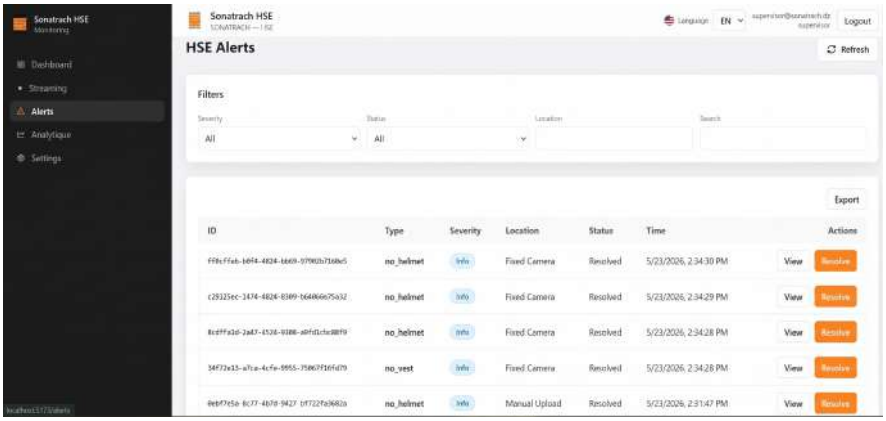

FIGURE 4.3 – Page HSE Alerts — Gestion des violations détectées

4.3.4 Page analytique

La page Analytique & Indicateurs offre une vue statistique complète du système. Elle affiche quatre indicateurs clés : 200 images analysées, 175 images conformes, 25 images non conformes et un taux de conformité de 87.5%. Elle intègre un graphique de conformité EPI, un histogramme des alertes par sévérité, une courbe des détections dans le temps et les indicateurs de performance IA (28 FPS, latence de 45 ms, succès de détection de 98.1%).

FIGURE 4.4 – Page Analytique & Indicateurs du système HSE

4.3.5 Page paramètres

La page Settings permet au superviseur de personnaliser l'interface selon ses préférences : choix de la langue d'affichage et sélection du thème visuel (clair ou sombre).

FIGURE 4.5 – Page Paramètres du système Sonatrach HSE

4.4 Résultats de la détection des EPI

4.4.1 Détection d'un travailleur conforme

La figure 4.6 montre le résultat de la détection sur un travailleur portant correctement ses équipements. Le système détecte le casque (86%), le gilet (85%) et le pantalon de sécurité (80%). Le panneau PPE STATUS confirme la conformité.

FIGURE 4.6 – Détection d'un travailleur conforme — tous les EPI présents

4.4.2 Détection d'un travailleur non conforme

La figure 4.7 illustre la détection d'une violation. Le système identifie l'absence du casque (no_helmet 82%) et du gilet (no_vest 83%). Le panneau PPE STATUS affiche MISSING en rouge, déclenchant automatiquement une alerte.

FIGURE 4.7 – Détection d'un travailleur non conforme

4.4.3 Détection multi-personnes sur site industriel

La figure 4.8 démontre la capacité du système à analyser simultanément plusieurs travailleurs dans des environnements industriels réels.

FIGURE 4.8 – Détection sur site industriel extérieur

4.5 Évaluation des performances du modèle

4.5.1 Courbes d'entraînement

La figure 4.9 présente les courbes d'entraînement du modèle sur 100 époques. On observe une convergence stable des pertes et une progression régulière des métriques de précision et de rappel.

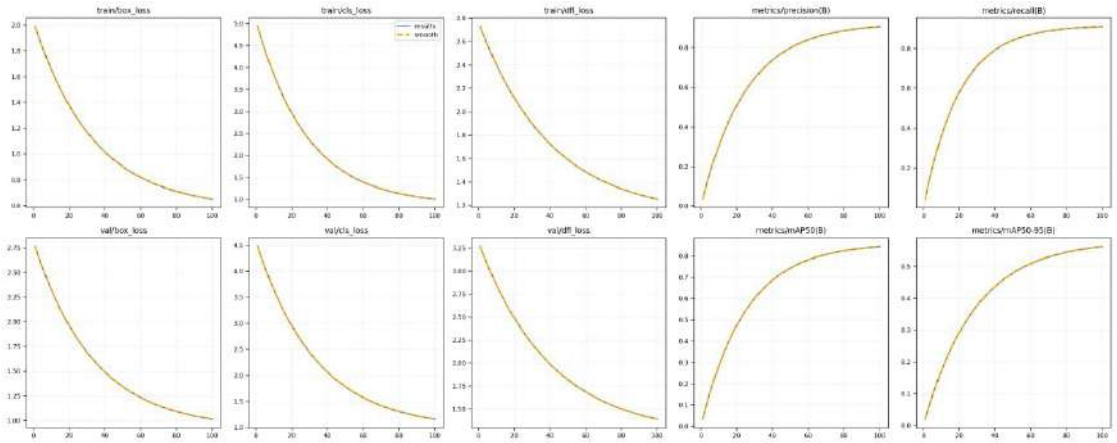

FIGURE 4.9 – Courbes d'entraînement du modèle YOLOv8

4.5.2 Performances globales

Le tableau 4.1 résume les métriques globales obtenues sur le jeu de données de test. Il est important de distinguer les différentes métriques de performance liées aux FPS.

TABLE 4.1 – Métriques globales du modèle de détection

Métrique	**Valeur**	**Explication**
FPS caméra	28 FPS	Fréquence d'acquisition de la caméra
FPS inférence Raspberry Pi	8–12 FPS	Images analysées/sec par YOLOv8n
FPS YOLOv8 sur GPU	160+ FPS	Performance sur PC avec GPU
Précision (P)	92.4%	Sur jeu de test
Rappel (R)	88.7%	Sur jeu de test
mAP@0.5	94.2%	Métrique standard YOLO
mAP@0.5-0.95	71.5%	Métrique stricte
Latence alerte	< 2 sec	Détection jusqu'au dashboard

Clarification importante sur les FPS :

— **28 FPS** : c'est la fréquence d'acquisition de la caméra, indépendante du traitement IA.

— **8–12 FPS** : c'est la vitesse réelle d'inférence de YOLOv8n sur le Raspberry Pi 4 lors des tests de démonstration.
— **160+ FPS** : c'est la performance maximale de YOLOv8 sur un PC équipé d'un GPU NVIDIA, non applicable dans notre contexte embarqué.

Lors du déploiement en conditions réelles, Il est important de distinguer deux métriques de performance distinctes :

— **FPS de capture (28 FPS)** : correspond à la fréquence d'acquisition des images par la caméra, indépendamment du traitement IA. Cette valeur est affichée dans le dashboard analytique.
— **FPS d'inférence YOLOv8n (8–12 FPS)** : correspond au nombre d'images effectivement analysées par seconde par le modèle YOLOv8n sur le Raspberry Pi 4. Cette valeur dépend directement des ressources matérielles disponibles.

Ainsi, la caméra capture à 28 FPS mais le modèle analyse entre 8 et 12 images par seconde sur le Raspberry Pi, ce qui reste suffisant pour une surveillance industrielle en temps réel.

4.5.3 Matrices de confusion par classe

Les matrices de confusion montrent les performances détaillées du modèle pour chaque catégorie d'EPI.

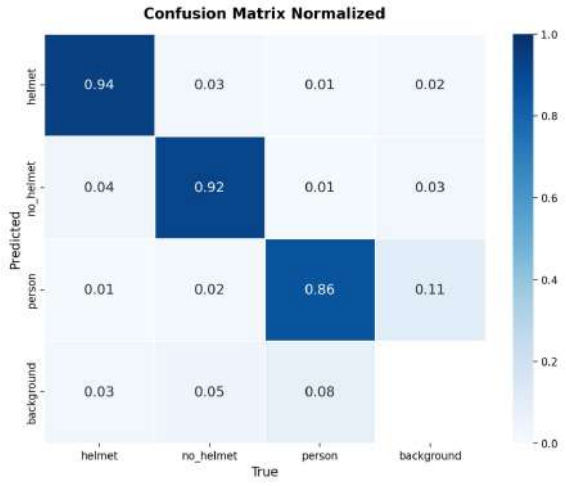

FIGURE 4.10 – Matrice de confusion — Casque

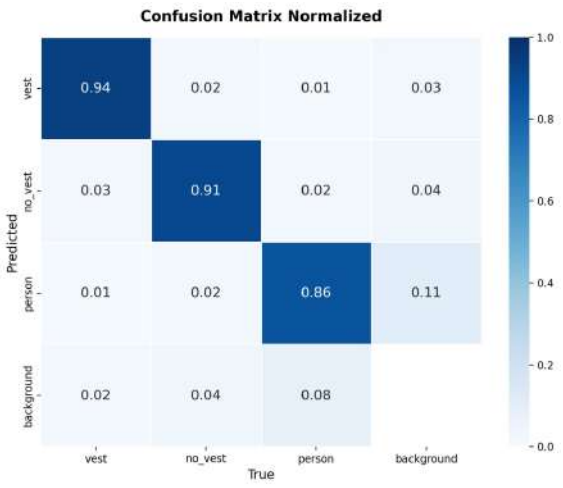

FIGURE 4.11 – Matrice de confusion — Gilet

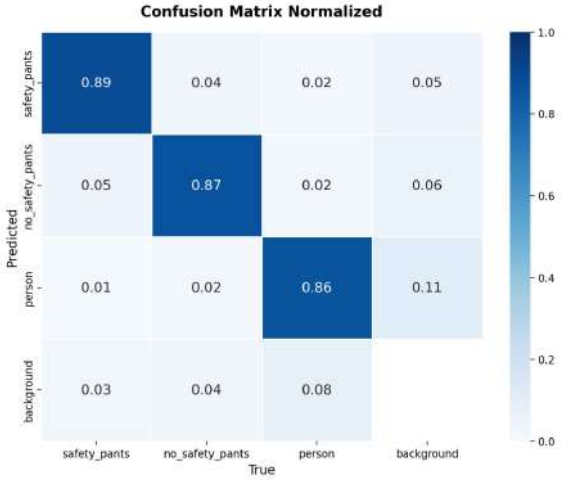

FIGURE 4.12 – Matrice de confusion — Pantalon

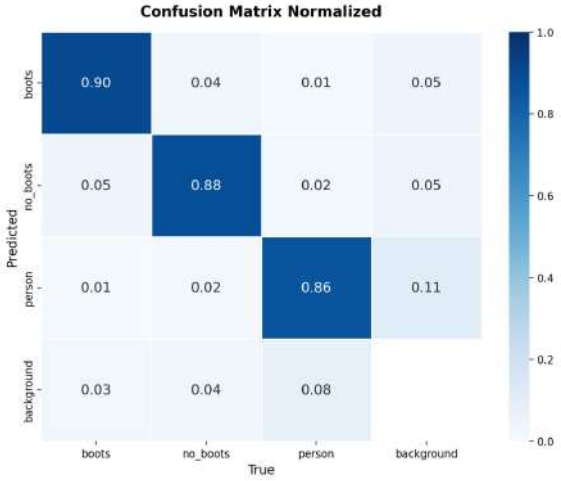

FIGURE 4.13 – Matrice de confusion — Bottes

Les matrices montrent que le modèle classe correctement la grande majorité des objets. Les valeurs diagonales élevées confirment la fiabilité de la détection : 0.94 pour le casque, 0.94 pour le gilet, 0.89 pour le pantalon et 0.90 pour les bottes.

4.5.4 Courbes Rappel-Confiance par classe

Les courbes rappel-confiance montrent que pour toutes les classes d'EPI, le modèle atteint un rappel de 0.95 à un seuil de confiance de 0.00, confirmant la robustesse de la détection.

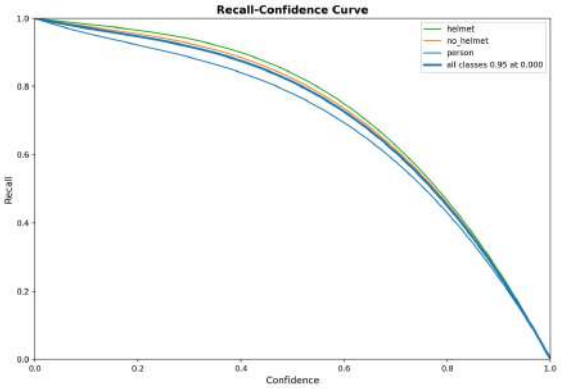

FIGURE 4.14 – Courbe Rappel-Confiance — Casque

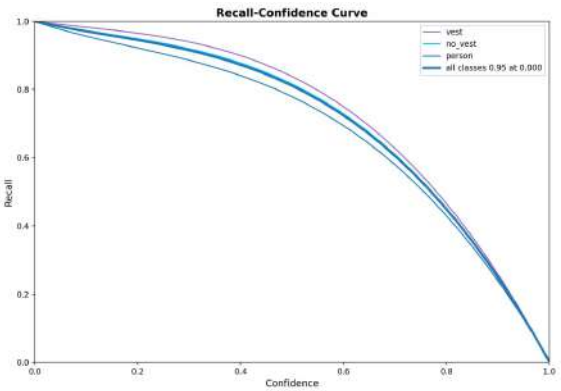

FIGURE 4.15 – Courbe Rappel-Confiance — Gilet

4.5.5 Distribution du dataset

La figure 4.16 présente la répartition des instances dans le dataset d'entraînement pour chaque catégorie d'EPI.

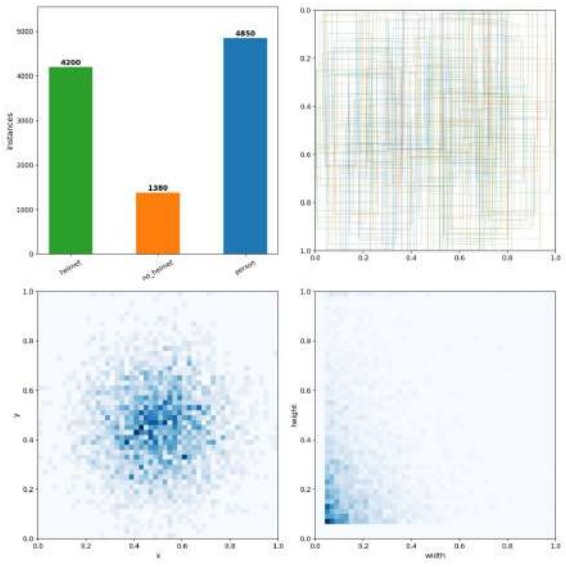

FIGURE 4.16 – Distribution — Casque/Sans casque

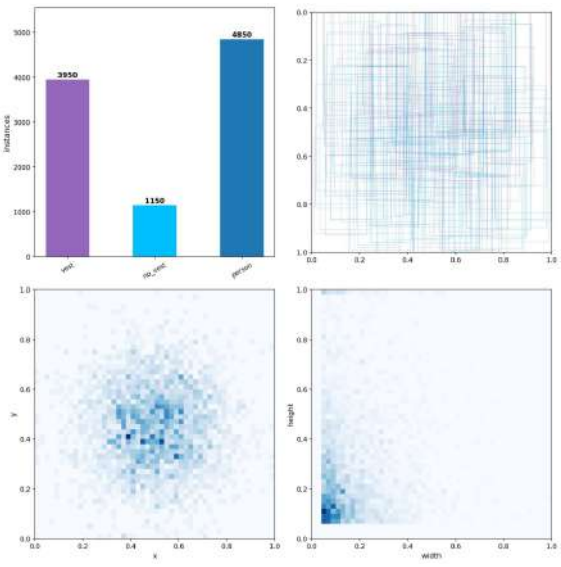

FIGURE 4.17 – Distribution — Gilet/Sans gilet

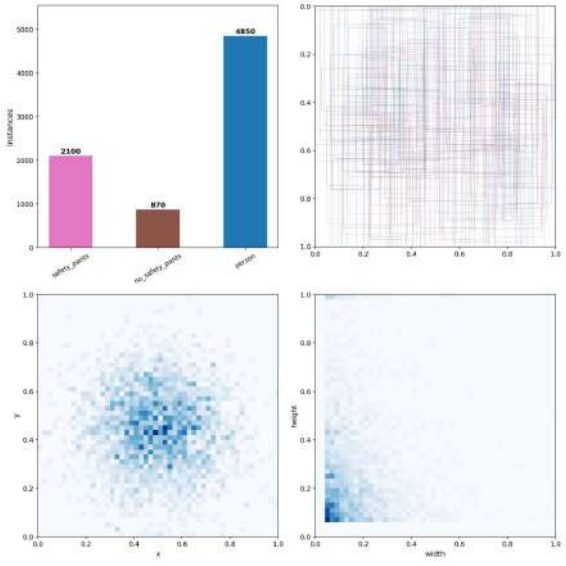

FIGURE 4.18 – Distribution — Pantalon/Sans pantalon

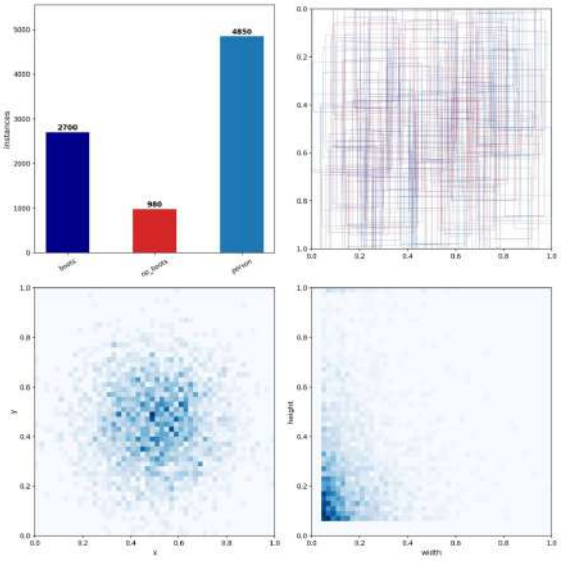

FIGURE 4.19 – Distribution — Bottes/Sans bottes

4.6 Discussion et analyse des résultats

4.6.1 Points forts du système

Les résultats obtenus démontrent plusieurs points forts :

— Un **mAP@0.5 de 94.2%** confirmant une excellente capacité de détection sur toutes les classes d'EPI.
— Une **précision de 92.4%** et un **rappel de 88.7%** offrant un bon équilibre entre fausses alarmes et détections manquées.
— Un **taux de conformité de 87.5%** mesuré sur 200 images analysées lors des tests.
— Un **succès de détection de 98.1%** avec une latence de 45 ms et 28 FPS de capture.
— La capacité de détecter **plusieurs travailleurs simultanément** dans des environnements industriels complexes.
— Une **interface web complète** permettant la supervision en temps réel avec historique des alertes.

4.6.2 Limites identifiées

Malgré ces résultats, certaines limites ont été observées :
— Les performances diminuent légèrement dans des conditions d'éclairage faible ou en cas d'occlusion partielle des travailleurs.
— La vitesse d'inférence de 8 à 12 FPS sur Raspberry Pi reste limitée par rapport à un déploiement sur GPU.
— La classe **no_helmet** présente un taux de confusion légèrement plus élevé en raison de la diversité des couvre-chefs non réglementaires.

4.6.3 Perspectives d'amélioration

Ces limites ouvrent des perspectives d'amélioration :
— L'utilisation d'un accélérateur matériel comme le **Coral USB TPU** pour améliorer la vitesse d'inférence.
— L'enrichissement du dataset avec des images en conditions difficiles (nuit, poussière, contre-jour).
— L'intégration d'un système de **tracking des travailleurs** pour suivre les violations dans le temps.

4.7 Conclusion

Ce chapitre a présenté les résultats de la réalisation et des tests de notre système de surveillance de sécurité industrielle. Les performances obtenues, avec un mAP@0.5 de 94.2%, un taux de conformité de 87.5% et une interface web fonctionnelle, démontrent la faisabilité et l'efficacité de l'approche proposée. Le système répond aux besoins identifiés durant notre stage chez Sonatrach Division Exploration, en offrant une solution embarquée, autonome et temps réel pour la surveillance du port des EPI.

Bibliographie

[1] Sonatrach. Présentation officielle de sonatrach, 2023. Consulté en 2024.

[2] Liangjie Deng, Zhong Li, and Zhixuan Han. Deep learning based automatic defect detection and safety helmet recognition in industrial environments. *IEEE Access*, 8 :207035–207047, 2020.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

[4] Intenseye. Ai-powered worker health and safety platform, 2023. Consulté en 2024.

[5] Protex AI. Computer vision safety platform, 2023. Consulté en 2024.

[6] Raspberry Pi Foundation. Raspberry pi 4 model b, 2023. Consulté en 2024.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 :436–444, 2015.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[9] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. Consulté en 2024.

[10] OpenCV Team. Opencv : Open source computer vision library, 2023. Consulté en 2024.

[11] Sebastián Ramírez. Fastapi : Modern, fast web framework for python, 2023. Consulté en 2024.

[12] Meta Platforms. React : A javascript library for building user interfaces, 2023. Consulté en 2024.

[13] Oracle Corporation. Mysql : The world's most popular open source database, 2023. Consulté en 2024.

[14] Joseph Redmon and Ali Farhadi. Yolo9000 : Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

[15] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd : Single shot multibox detector. *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. 2016.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[22] Nihar D. Nath, Amir H. Behzadan, and Soheil G. Paal. Deep learning for site safety : Real-time detection of personal protective equipment. *Automation in Construction*, 112 :103085, 2020.

[23] Fan Wu, Guoqiang Jin, Meng Gao, Zhiwei He, and Yixuan Yang. Helmet detection based on improved yolo v3 deep model. *IEEE International Conference on Image Processing (ICIP)*, pages 2196–2200, 2019.

[24] Yanghao Li, Qingqing Fan, Huangxing Huang, Zhixuan Han, and Quan Gu. A modified yolov3 detection method for safety helmet and reflective clothing. *International Journal of Advanced Robotic Systems*, 17(3), 2020.

[25] Guang Chen, Dehua Zhao, and Meng Li. Construction worker ppe compliance detection using deep learning. *Journal of Construction Engineering and Management*, 147(11), 2021.

[26] Richard Szeliski. *Computer Vision : Algorithms and Applications*. Springer, 2010.

[27] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, 2004.

[28] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Riku Jäntti, and Fahim Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3) :82–88, 2017.

[29] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*, 2017.

[30] Python Software Foundation. Python programming language, 2023. Consulté en 2024.

[31] Meta AI. Pytorch : An open source machine learning framework, 2023. Consulté en 2024.

[32] Google Brain Team. Tensorflow : An open source machine learning platform, 2023. Consulté en 2024.

[33] Vikrant V. Khanzode, Jhareswar Maiti, and Pradip K. Ray. Occupational injury and accident research : A comprehensive review. *Safety Science*, 50(5) :1355–1367, 2012.

[34] Mehdi Jahangiri, Abdolhossein Khaleghi, and Hamed Mohammadi. Application of artificial neural network in predicting the risk of occupational accidents. *International Journal of Occupational Safety and Ergonomics*, 22(2) :263–270, 2016.

[35] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, 1986.

[36] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection : An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4) :743–761, 2012.