REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IA02/IA/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Intelligence Artificielle**

UNE SOLUTION EDGE IA POUR LA DETECTION DU VOL A L'ETALAGE

Par :

GHAMRI MOHAMED ISLAM

Soutenu le 21/06/2023 devant le jury composé de :

Rachida SAOULI	Prof	Président
Salim BITAM	Prof	Rapporteur
Meryem BERGHIDA	MCB	Examinateur
Imed ALLAL	Dr	Invité

Année universitaire 2022-2023

Remerciements

Avant tout, je remercie ALLAH de m'avoir aidé à réaliser mon Projet et de m'avoir accordé la patience et le courage pendant ces longues années d'étude.

Je tiens à adresser mes chaleureux remerciements à mon encadrant, Monsieur Salim Bitam, ainsi qu'à Monsieur Imed Allal et à l'équipe de développement de NAMLA Networks (Abdlghani Kabot, Rania Bouguesri) qui m'ont aidé à élaborer ce projet grâce à leur patience, disponibilité, conseils et soutien tout au long de cette période.

Enfin, je tiens à remercier tous ceux qui m'ont aidé de près ou de loin à réaliser ce mémoire, en particulier mes parents pour leur soutien et leur patience. Merci à vous tous.

ABSTRACT

Shoplifting, also known as pickpocketing or retail theft, is a criminal act involving the fraudulent taking of goods from a store without paying for them. This typically occurs when the thief conceals the items on their person, in their clothing, a handbag, or a backpack, and leaves the store without paying for them.

After studying various approaches and methods used in related works to address this issue, we were able to propose a new solution for this problem. It involves a hybrid CNN-ANN optimized method capable of detecting shoplifting from a video or a camera deployed in Edge AI to obtain several advantages, such as reduced inference time, reduced system storage size, and solution mobility.

In our work, we utilized a UFC-Crime database that contains videos of anomalies (shoplifting, theft, fights, etc.). After analyzing and cleaning these videos, we obtained 20 shoplifting videos and 270 normal videos that we could use. Additionally, we collected and purchased some other shoplifting videos from the internet. Despite these efforts, we still needed more shoplifting videos, so we employed a data augmentation method (videos). Subsequently, we trained our system using these videos, employing the hybrid approach: a pre-trained C3D model for feature extraction and a second ANN model for video classification. We achieved a good precision rate of over 94%. However, in order to reduce the inference time and system storage size and to deploy this system on Edge AI, we had to optimize our models using the OpenVino platform. Unfortunately, we encountered a problem: the precision of our optimized models was reduced compared to the initial models. To address this, we retrained our optimized models using the post-training technique to improve precision. As a result, we obtained an acceptable precision of over 91% while reducing storage size and inference time.

Subsequently, we created two versions of our graphical user interface using ReactJS. We then dockerized our work into a container composed of three containers: one for the inference code, another for our graphical interface, and the last one for the MQTT protocol to establish a connection between our front-end and back-end. Finally, we deployed our solution on Edge AI.

Keywords: Edge AI, CNN, ANN, data augmentation, pre-trained model, C3D, OpenVino, OpenVino post-training, ReactJS, Docker, MQTT.

Résumé

Le vol à l'étalage, également connu sous le nom de vol à la tire ou vol à l'étalage, est un acte criminel consistant à prendre frauduleusement des biens dans un magasin sans en payer le prix. Cela se produit généralement lorsque le voleur dissimule les articles sur sa personne, dans ses vêtements, un sac à main ou un sac à dos, et quitte le magasin sans les avoir payés.

Apres avoir étudié différentes approches et méthodes utilisé dans des travaux connexes pour résoudre cette problématique, nous avons pu de proposer une nouvelle solution pour cette problématique qui s'agit d'une méthode hybride CNN-ANN optimisé capable de détecter de vol à l'étalage à partir d'une vidéo ou d'une caméra déployé dans l'Edge IA pour d'obtenir quelques avantages comme le temps d'inférence réduit , la taille de stockage de notre système réduit et de mobilité de notre solution.

Dans notre travail, nous avons utilisé une base de données UFC-Crime qui contient des vidéos d'anomalies (vol à l'étalage, vol, bagarre …), après l'analyse et le nettoyage de ces vidéos nous avons obtenu que de 20 vidéos de vol à l'étalage et 270 vidéos normal que nous pouvons utiliser et pour cela nous avons collecté et acheté quelques autres vidéos de vol sur l'internet et même avec cette méthode nous avons encore besoin d'autres vidéos de vol et pour cela nous avons utilisé la méthode d'augmentation de données (vidéos), ensuite on a entrainé notre système avec ces vidéos utilisant l'approche hybride, un modèle pré-entrainé C3D pour l'extraction de caractéristiques et un deuxième modèle ANN pour classification de nos vidéos et nous avons sorti avec une bonne précisons de plus de 94%, mais pour réduire le temps d'inférence et la taille de stockage de notre système et pour avoir déployer ce système sur l'Edge IA nous somme obliger d'optimisé nos modèle utilisant la plateforme OpenVino, cependant nous avons rencontré un problème : la précision de nos modèle optimisé était réduit par rapport aux premiers modèles et pour cela nous avons re-entrainée nos modèles optimisé avec la technique de post-entraînement pour améliorer la précision et comme ca nous avons obtenu une précision acceptable de plus de 91% et nous avons réduit le stockage et aussi le temps d'inférence.

Ensuite, nous avons créé deux version de notre interface graphique utilisant ReactJS, après nous avons dockerisé notre travail dans un conteneur compose de trois conteneurs (un conteneur de docker pour le code d'inférence et deuxième conteneur pour notre interface

graphique et le dernier conteneur pour protocole de MQTT pour faire une liaison entre notre Front-end et notre Back-end), Enfin, nous avons déployé notre solution dans l'Edge IA.

Mots clés : Edge IA, CNN, ANN, augmentation de données, modèle pré-entrainé, C3D, OpenVino, post-entraînement d'OpenVino, ReactJS, docker, MQTT.

ملخص:

السرقة من المتاجر، المعروفة أيضًا بالسرقة بالتلميح أو السرقة في المحلات، هي جريمة تتمثل في ارتكاب سرقة بالاحتيال للسلع من متجر دون دفع ثمنها. عادة ما يحدث ذلك عندما يخفي اللص الأشياء على جسده، في ملابسه، حقيبة يد أو حقيبة ظهر، ثم يغادر المتجر دون دفع ثمنها.

بعد دراسة مختلف النهج والأساليب المستخدمة في الأعمال ذات الصلة لحل هذه المشكلة، تمكنا من اقتراح حلاً جديدًا لهذه المشكلة. يتعلق الأمر بأسلوب متكامل بين شبكات العصب الاصطناعية والتعلم العميق المُحسَّن قادر على اكتشاف السرقة من المتاجر من خلال فيديو أو كاميرا مثبتة على منصة الذكاء الحوسبي المحافظة. ويتم تحقيق عدة مزايا من هذا الأسلوب مثل تقليل وقت التحليل وحجم التخزين للنظام ومرونة الحل.

في عملنا، استخدمنا قاعدة بيانات UFC-Crime التي تحتوي على مقاطع فيديو للأحداث غير الطبيعية (مثل السرقة من المتاجر، السرقة، الشجار، إلخ). بعد تحليل وتنظيف هذه المقاطع، حصلنا على 20 فيديو للسرقة من المتاجر و270 فيديو عادي يمكننا استخدامها. بالإضافة إلى ذلك، قمنا بجمع وشراء بعض مقاطع الفيديو الأخرى للسرقة من الإنترنت. ومع ذلك، كنا لا نزال بحاجة إلى مزيد من مقاطع الفيديو للسرقة من المتاجر، لذا استخدمنا تقنية زيادة البيانات (الفيديو). بعد ذلك، قمنا بتدريب نظامنا باستخدام هذه المقاطع، باستخدام النهج المتكامل، نموذج مسبق التدريب C3D لاستخراج الميزات ونموذج ANNثانوي لتصنيف مقاطع الفيديو. حققنا نتائج دقة جيدة تفوق 94%. ومع ذلك، لتقليل وقت التحليل وحجم التخزين للنظام ونشر هذا النظام على منصة الذكاء الحوسبي المحافظة، كان علينا تحسين نماذجنا باستخدام منصة .OpenVino للأسف، واجهنا مشكلة: تم تقليل دقة نماذجنا المُحسَّنة مقارنة بالنماذج الأولية. لحل هذه المشكلة، قمنا بإعادة تدريب نماذجنا المُحسَّنة باستخدام تقنية ما بعد التدريب لتحسين الدقة. وبالتالي، حققنا دقة مقبولة تفوق 91% مع تقليل حجم التخزين ووقت التحليل.

ثم، قمنا بإنشاء نسختين من واجهة المستخدم الرسومية باستخدام .ReactJS ثم استخدمنا Docker لتجميع عملنا في حاوية تتكون من ثلاث حاويات) حاوية للكود المتعلق بالتحليل، وحاوية ثانية لواجهة المستخدم الرسومية، والحاوية الأخيرة لبروتوكول MQTT لتأمين الاتصال بين الواجهة الأمامية والخلفية .(في النهاية، قمنا بنشر حلنا على منصة الذكاء الحوسبي المحافظة.

الكلمات الرئيسية: الذكاء الحوسبي المحافظ، شبكات العصب الاصطناعية، التعلم العميق، زيادة البيانات، نموذج مسبق التدريب، C3D، OpenVino، ما بعد التدريب فيOpenVino ، ReactJS، Docker، MQTT.

Table des matières

Table de Figures

Chapitre 01 :

Chapitre 02 :

Chapitre 03 :

Chapitre 04 :

Introduction Générale

Le vol à l'étalage est un crime courant qui pose un défi considérable pour les propriétaires des magasins, les forces de l'ordre et le système judiciaire. Il consiste à voler des articles dans un magasin de détail en les dissimulant dans ses vêtements ou son sac et en quittant le magasin sans payer. Selon l'association nationale de prévention du vol à l'étalage, environ 1 personne sur 11 visitant un magasin, est un voleur à l'étalage, et les voleurs de ce type ne sont arrêtés qu'une fois tous les 48 vols. Malgré le nombre croissant de caméras de surveillance dans les lieux publics tels que les centres commerciaux et les magasins de détail, le vol à l'étalage reste un problème majeur, avec environ 27 millions de personnes qui volent chaque année, entraînant des pertes d'au moins 13 milliards de dollars américains. Bien que les caméras CCTV soient principalement utilisées pour prévenir le vol à l'étalage, il est nécessaire de mettre en place une surveillance vidéo automatique capable de détecter les événements liés au vol à l'étalage, compte tenu des limites des ressources humaines de surveillance. Dans ce mémoire, nous conduisons une étude afin de résoudre la problématique suivante :

Est-il possible de développer un système intelligent basé sur une approche hybride pour la détection de vol à l'étalage et son implémentation sur l'Edge de l'intelligence artificielle (Edge IA) ?

Pour réaliser ce système, nous utilisons un ensemble de données ; le dataset UCF-Crime qui contient 14 classes, chaque classe représentant un type de crime [27]. Toutefois, pour notre problématique axée sur le vol à l'étalage, nous avons sélectionné uniquement les deux classes "Normalvideos" et "Shoplifting". Cependant, cette dernière classe souffrait d'un nombre réduit de vidéos, c'est pourquoi nous avons collecté plusieurs vidéos de vol et nous avons utilisé la technique de data augmentation pour équilibrer les deux classes. Par ailleurs, nous avons développé un système intelligent hybride qui utilise deux modèles : le premier est un modèle pré-entraîné qui extrait les caractéristiques de la vidéo, et le deuxième est un modèle de classification de la vidéo à deux classes : vol à l'étalage ou classe normale. Nous avons optimisé cette solution en utilisant la plateforme OpenVino pour améliorer le temps d'inférence et minimiser la taille de la solution pour pouvoir l'implémenter sur l'Edge IA.

Notre mémoire est organisée en quatre chapitres. Le premier chapitre présente les concepts clés dans le domaine de la sécurité dans un immeuble commercial, tels que le vol à

l'étalage, la vidéosurveillance, l'IoT, la vidéo, l'Edge Computing et la vision par ordinateur. Ces technologies ont un impact direct sur la sécurité des individus et des entreprises. Le deuxième chapitre présente les travaux antérieurs dans le domaine de l'application de l'intelligence artificielle à la surveillance vidéo pour la détection de vol à l'étalage à partir de vidéos ou de caméras. Nous passons en revue les méthodes existantes et les avancées dans ce domaine. Le troisième chapitre décrit notre système conçu en illustrant les différentes étapes de développement et de conception de ce système de détection de vol à l'étalage en temps réel, qui se basent sur des techniques de deep learning et de l'Edge computing. Nous présentons les techniques utilisées pour optimiser les modèles de détection, la collecte et l'augmentation des données, ainsi que l'implémentation du système sur des périphériques Edge. Le quatrième et dernier chapitre de notre mémoire présentera l'environnement logiciel et matériel sur lequel notre système de détection de vol à l'étalage sera réalisé. Nous y aborderons également les langages de programmation et les outils que nous avons exploités. En outre, nous évaluerons les résultats obtenus avec notre système et discuterons des perspectives futures dans ce domaine.

Chapitre 01 :

Le vol à l'étalage IA et l'Intelligence Artificielle en périphérie: Notions de Base

Introduction

Le vol à l'étalage est un problème fréquent dans les immeubles commerciaux, qui peut avoir un impact significatif sur la rentabilité des entreprises. Ce chapitre est consacré aux concepts clés dans le domaine de la sécurité sur un immeuble commercial, notamment le vol à l'étalage, la vidéosurveillance, l'IoT, la vidéo, l'Edge Computing et la vision par ordinateur et quelques concepts essentiels. Ces concepts sont très importants dans cette étude pour faire face à cette problématique qui touche notre société moderne où les menaces de sécurité sont de plus en plus fréquentes et diverses. Il est donc essentiel de comprendre ces notions afin de mieux appréhender les enjeux de sécurité liés à ces technologies.

1.1. Vol à l'étalage

Le vol à l'étalage se réfère au vol de produits dans un magasin de vente au détail durant les heures d'ouverture. Ce délit implique généralement qu'un individu cache un article sur lui-même, dans des poches, sous des vêtements ou dans un sac, avant de quitter le magasin sans le payer. Les termes "vol à l'étalage" et "voleur à l'étalage" ne sont pas habituellement définis par la loi et sont considérés comme une forme de vol. Les détaillants utilisent le terme "démarque" pour désigner les produits perdus en raison du vol à l'étalage, ainsi que d'autres pertes telles que les déchets, les dommages non assurés aux produits et le vol commis par des employés.

Le vol à l'étalage peut être pratiqué par des individus amateurs agissant par impulsion ou par des criminels de carrière qui en font leur source de revenu. Les criminels de carrière peuvent avoir recours à plusieurs personnes pour commettre des vols, certaines distrayant les employés du magasin tandis que d'autres dérobent les articles. Les articles les plus fréquemment volés sont ceux dont le prix est proportionnel à leur taille, comme les lames de rasoir jetables, les vitamines, les boissons alcoolisées et les cigarettes.

Les détaillants utilisent différentes stratégies pour limiter le vol à l'étalage, comme le stockage de produits coûteux dans des vitrines verrouillées, l'attache d'articles à des étagères ou des supports à vêtements, l'utilisation de capteurs magnétiques ou radioélectriques, ou encore l'embauche d'agents de sécurité en civil. Les employés peuvent également être formés pour détecter les voleurs à l'étalage potentiels.

Le vol à l'étalage a été documenté pour la première fois à Londres au XVIe siècle. Au début du XIXe siècle, ce délit était principalement commis par des femmes, comme le gang des

Forty Elephants. Dans les années 1960, le vol à l'étalage a été considéré comme un acte politique, et les chercheurs ont divisé les voleurs à l'étalage en deux catégories : les "boosters" (professionnels qui revendent ce qu'ils volent) et les "snitches" (amateurs qui volent pour leur usage personnel) [1].

Figure 1.1. Vol à l'étalage

1.2. Vidéosurveillance

La vidéosurveillance, également appelée videoprotection, est un système de surveillance à distance qui utilise des caméras pour capturer des images dans un espace public ou privé. Ces images peuvent être traitées automatiquement, visionnées, archivées ou supprimées. L'objectif principal de la vidéosurveillance est d'assurer la sécurité, la sûreté ou le respect d'une procédure spécifique. Lorsque le système est industrialisé et qu'un opérateur peut surveiller plusieurs lieux simultanément sur des écrans, cela s'appelle la télésurveillance stricte et est utilisé à des fins civiles. Les partisans de la vidéosurveillance affirment qu'elle peut aider à prévenir les actes de terrorisme et de criminalité, tels que les attaques à main armée, les cambriolages et les agressions dans les espaces publics, ainsi qu'à faciliter le contrôle social,

comme la gestion des mouvements de foule [2] [3].

Figure 1.2.Une caméra de vidéosurveillance

1.2.1 Composition d'un système de vidéosurveillance

Un système de vidéosurveillance est composé de trois types d'équipements : [4]

Figure 1.3. Un système de vidéosurveillance

1.2.1.1 Équipements de réception

Pour garantir une surveillance efficace dans un système de vidéosurveillance, le choix des caméras est essentiel. Selon les exigences de l'utilisateur, l'environnement et le budget alloué, il est important de sélectionner les caméras appropriées pour surveiller les zones cibles. Il existe différents types de caméras telles que les caméras couleur ou noir et blanc, fixes ou mobiles, anti-vandalisme ou discrètes, avec ou sans son, infrarouge, etc. [4].

1.2.1.2 Équipements de gestion

Pour assurer la gestion et l'exploitation des images filmées dans un système de vidéosurveillance, plusieurs équipements tels que des DVR, NVR, serveurs et logiciels sont disponibles. Toutefois, le choix des équipements de gestion doit être déterminé en fonction des besoins exprimés par le client dans le cahier des charges [4].

1.2.1.3 Équipements de visualisation

L'équipement essentiel pour visualiser les images captées en direct ou enregistrées par un système de vidéosurveillance est le moniteur, également appelé écran. Actuellement, il existe plusieurs options pour la visualisation, notamment une visualisation fixe sur un ordinateur ou une télévision au bureau, ainsi qu'une visualisation mobile sur un smartphone ou une tablette, et même à distance via Internet. En fonction des besoins de l'utilisateur, il est important de choisir le type d'écran, notamment sa taille et sa technologie [4].

1.3. Vidéo

Pour parler de la vidéo, on peut dire que c'est une technologie qui permet de capturer, stocker et diffuser des images en mouvement, accompagnées éventuellement de son. Elle est utilisée dans divers domaines tels que le divertissement, les médias, la surveillance, la recherche scientifique, la médecine et l'éducation. En général, la vidéo est associée à d'autres technologies comme l'enregistrement audio, l'édition vidéo, la compression vidéo et la diffusion en direct. Il existe plusieurs formats vidéo courants tels que MP4, AVI, MOV, WMV et MPEG [5].

1.4. Internet des objets

L'Internet des objets est une infrastructure qui permet de connecter des capteurs simples et parfois bruités à des applications de haut niveau, en comblant ainsi l'écart entre les données brutes collectées et les services sophistiqués offerts par la virtualisation.

1.4.1. Définition de l’IOT

L'Internet des objets (IoT) est un réseau de dispositifs électroniques normalisés et interconnectés, tels que des capteurs, des appareils mobiles et des objets physiques, qui permettent une communication transparente et fluide entre le monde physique et le monde virtuel. Grâce à cette interconnexion, il est possible de récupérer, stocker, transférer et traiter

des données de manière efficace et sans interruption. En conclusion, l'IoT permet une identification directe et sans ambiguïté des entités numériques et des objets physiques [6].

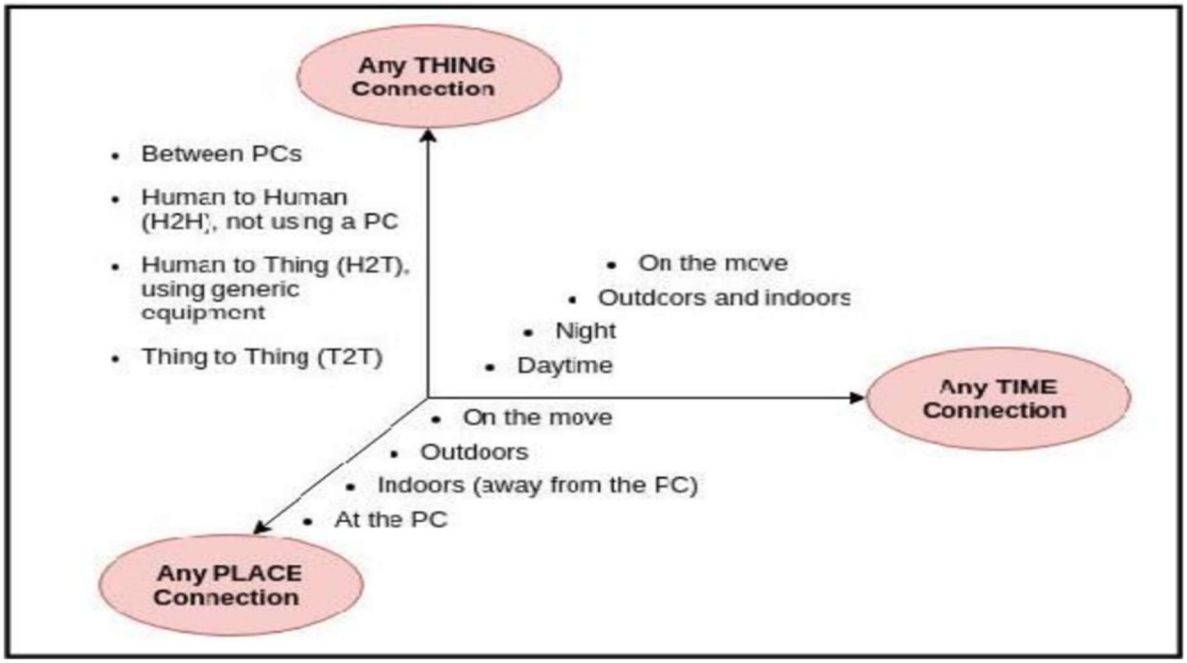

Figure 1.4. Nouvelle dimension pour internet

1.4.2 Concept d'IOT

L'IOT est un réseau interconnectant des machines, des objets vivants et non vivants intégrés à des capteurs, actionneurs, électronique, logiciels et connectivité réseau, permettant le transfert de données sans l'intervention humaine. Cette interconnexion permet de détecter ou de contrôler les objets à distance, créant ainsi des opportunités pour une intégration directe du monde physique dans les systèmes informatiques. Cela entraîne une amélioration de l'efficacité, de la précision et des avantages économiques, ainsi qu'une réduction de l'interaction humaine.

"L'Internet des Objets (IoT) est un réseau mondial de services interconnectés et d'objets intelligents de toutes natures destinés à soutenir les humains dans les activités de la vie quotidienne grâce à leurs capacités de détection, de calcul et de communication. Leurs aptitudes à observer le monde physique et à fournir des informations pour la prise de décision, seront partie intégrante de l'architecture de l'Internet du futur" [7].

1.5. Vision par ordinateur

La vision par ordinateur est un champ interdisciplinaire qui se concentre sur la manière dont les ordinateurs peuvent extraire des informations significatives à partir d'images numériques ou de vidéos. Elle vise à automatiser des tâches qui peuvent être effectuées par le système visuel humain. Selon [8], la vision par ordinateur concerne "l'extraction automatique, l'analyse et la compréhension d'informations utiles à partir d'une seule image ou d'une séquence d'images". Cette discipline scientifique s'intéresse à la théorie derrière les systèmes artificiels qui extraient des informations à partir d'images. Les données d'image peuvent prendre de nombreuses formes, telles que des séquences vidéo, des vues de plusieurs caméras ou des données multidimensionnelles provenant d'un scanner médical. La vision par ordinateur vise également à appliquer ses théories et modèles à la construction de systèmes de vision par ordinateur [9][10][11].

En relation avec la vidéo, la vision par ordinateur permet d'extraire des informations à partir des images en mouvement, telles que la détection d'objets en mouvement, la reconnaissance de visages, la reconnaissance de gestes et de mouvements, la mesure de distances, etc. Ces informations peuvent ensuite être utilisées pour diverses applications telles que la surveillance de la sécurité, l'analyse du comportement des consommateurs, la réalisation d'études de marché et la surveillance de la circulation [12].

Figure 1.5.Détection de piétons dans diverses situations sur la voirie.

1.6. Edge Computing

1.6.1 Définition d'Edge Computing

L'edge computing, également connu sous le nom d'informatique en périphérie ou d'informatique en périphérie de réseau, est une méthode de traitement des données qui vise à optimiser les performances du cloud computing en effectuant des traitements automatiques et des analyses directement à la périphérie du réseau, à proximité de la source des données. Cette approche réduit les besoins en bande passante entre les capteurs et les centres de traitement de données, en mobilisant des ressources qui peuvent être connectées de manière intermittente, telles que des ordinateurs portables, des smartphones, des tablettes ou des capteurs. L'objectif est d'éviter la transmission de données peu pertinentes vers les centres de données ou le cloud, améliorant ainsi la réactivité et la fluidité des réactions. Les techniques d'edge computing comprennent notamment les réseaux de capteurs sans fil, l'acquisition de données transférables et itinérantes, l'analyse des signatures sur les portables, le traitement coopératif en peer-to-peer, la réalité augmentée, le stockage de données réparties et les services de cloud à distance [13][14][15].

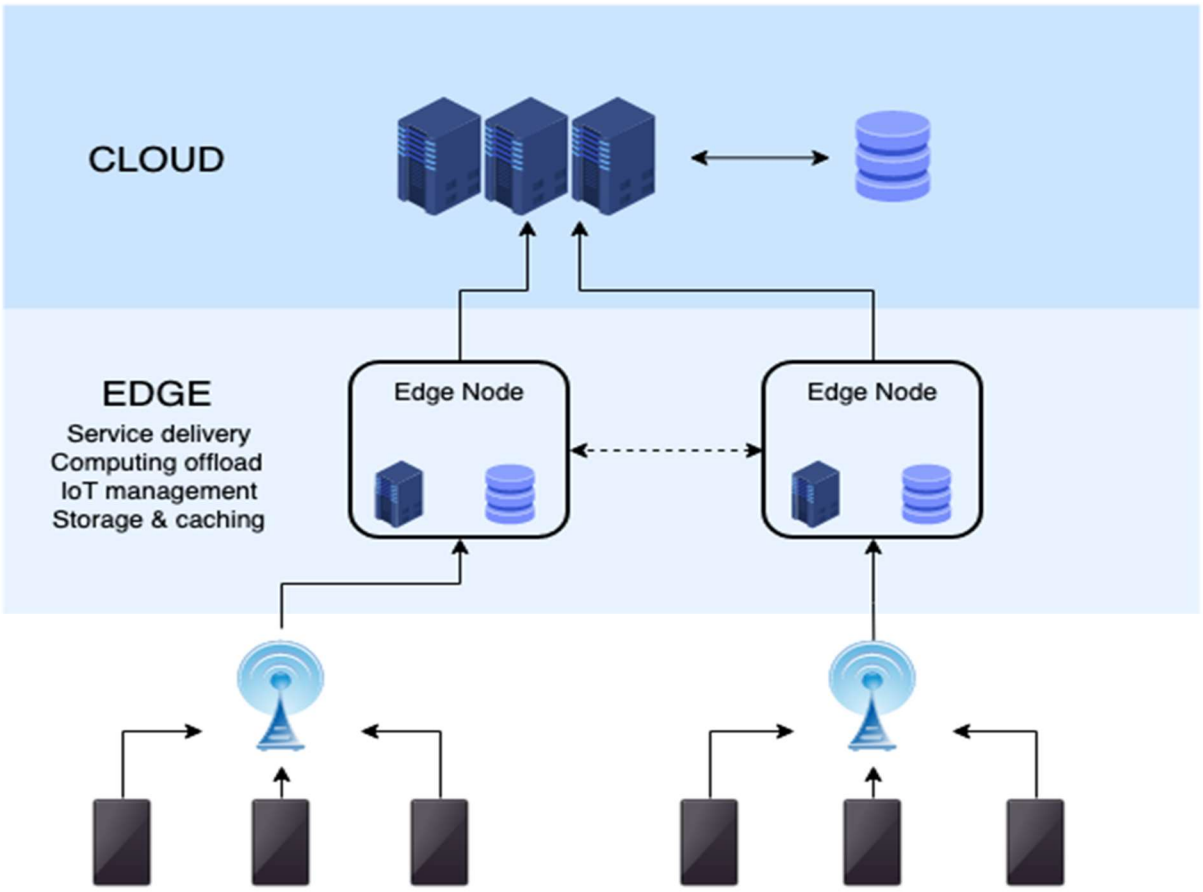

Figure 1.6.L'infrastructure de l'Edge Computing.

1.6.2 Edge IA

L'Edge IA, également appelée Intelligence Artificielle en périphérie, se réfère à l'utilisation de l'IA dans les appareils Edge afin de permettre une prise de décision intelligente locale, sans dépendre d'un traitement distant dans le cloud. Cette approche offre des avantages tels que des temps de réponse plus rapides, une réduction de la latence et une meilleure efficacité de la bande passante en réduisant les besoins de transfert de données vers des centres de traitement distants. Les applications de l'Edge IA incluent la reconnaissance vocale et d'image, la détection d'anomalies, la sécurité et la surveillance, ainsi que d'autres applications liées à l'IoT [16].

1.7. Réseau de neurones artificiels

Les réseaux de neurones artificiels sont des systèmes qui ont été conçus à partir d'une inspiration schématique du fonctionnement des neurones biologiques, puis se sont rapprochés des méthodes statistiques [17][18][19]. Ils sont généralement optimisés par des méthodes

d'apprentissage de type probabiliste, notamment bayésien, et sont utilisés à la fois dans les applications statistiques et les méthodes de l'intelligence artificielle. Les réseaux de Kohonen, par exemple, sont des paradigmes utilisés pour créer des classifications rapides. En modélisation des circuits biologiques, les réseaux de neurones artificiels permettent de tester des hypothèses fonctionnelles issues de la neurophysiologie et de comparer les conséquences de ces hypothèses avec la réalité. En somme, ces systèmes fournissent un mécanisme perceptif indépendant des idées propres de l'implémenteur et des informations d'entrée au raisonnement logique formel [20].

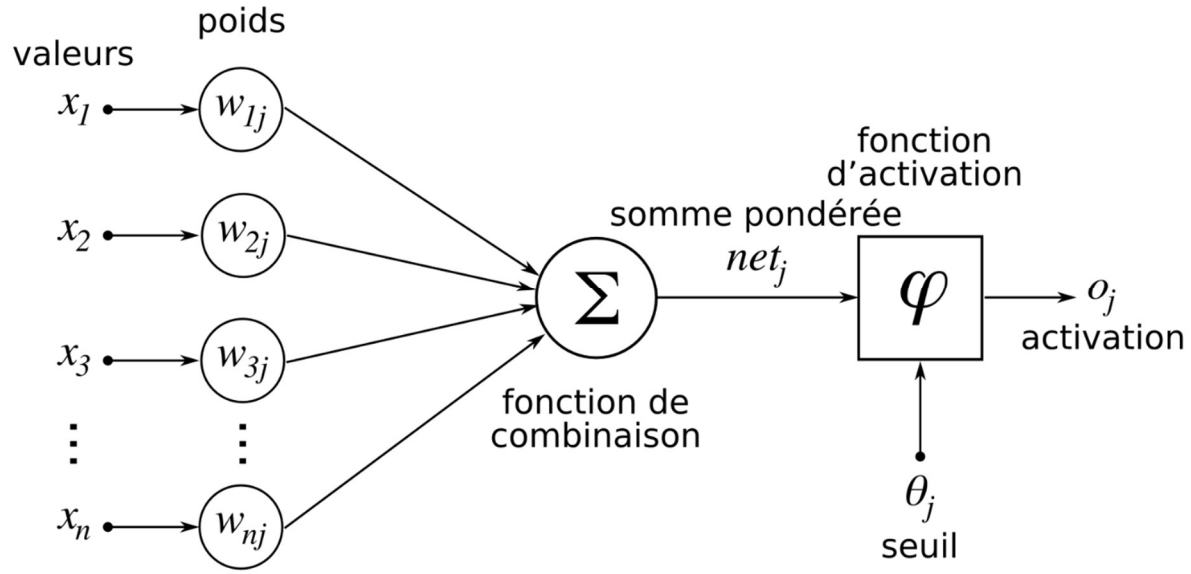

Figure 1.7. Un réseau de neurones artificiels.

1.7.1 Réseau de neurones à convolution (CNN)

Le réseau de neurones à convolution (CNN) est un type d'architecture de réseau de neurones communément utilisé dans l'analyse d'images et de vidéos. Le CNN utilise des couches de convolution pour extraire des caractéristiques à différents niveaux de granularité, puis des couches de pooling pour agréger les informations et réduire la dimension de l'image, augmentant ainsi la robustesse du modèle [21].

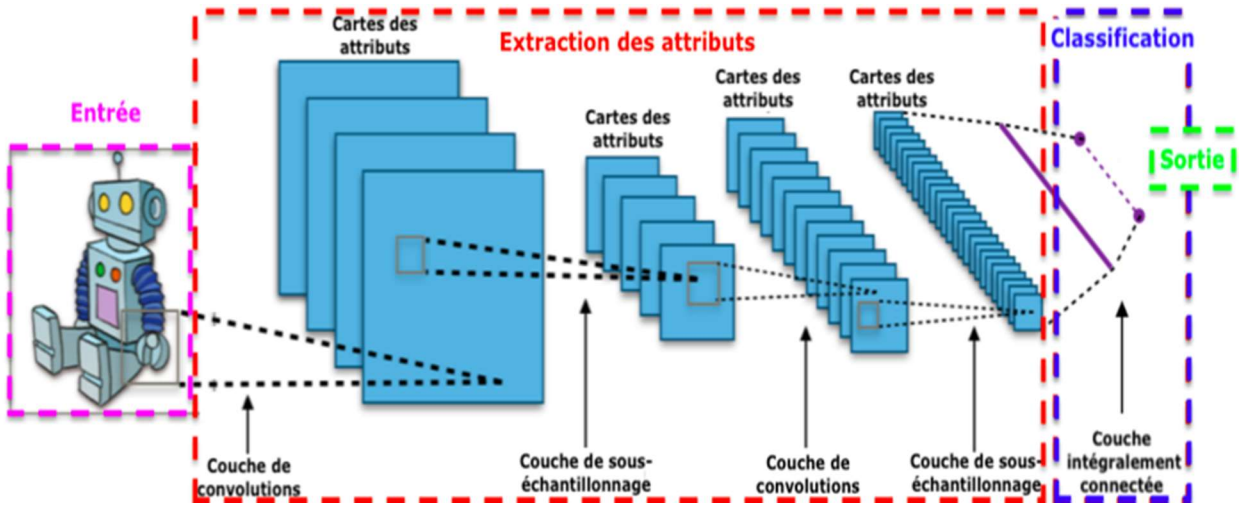

Figure 1.8. Un réseau de neurones à convolution (CNN).

1.7.2 Un réseau de neurones récurrent (RNN)

Un réseau de neurones récurrent (RNN) est une architecture de réseau de neurones adaptée au traitement de données séquentielles, telles que des séries temporelles ou des phrases dans un texte. Contrairement aux réseaux de neurones classiques, les RNN contiennent une boucle de rétroaction qui leur permet de conserver une mémoire interne de l'information précédemment traitée [22].

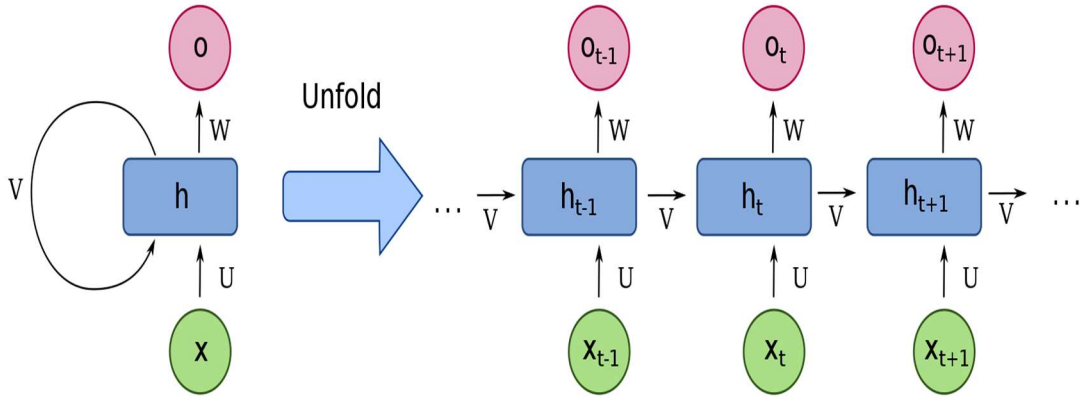

Figure 1.9. Un réseau de neurones récurrent (RNN).

1.8. Open Visual Inference and Neural Network Optimization

OpenVINO (Open Visual Inference and Neural Network Optimization) est une plateforme de développement de logiciels gratuite et open-source proposée par Intel. Elle comprend un ensemble d'outils, de bibliothèques et de modèles pré-entraînés qui permettent aux développeurs de créer et de déployer des applications de vision par ordinateur et d'apprentissage en profondeur sur une variété de plates-formes matérielles. OpenVINO permet également d'optimiser les modèles d'apprentissage profond pour une exécution efficace sur des processeurs Intel, y compris des processeurs basse consommation comme les processeurs Intel Movidius et Intel Atom. La plateforme prend en charge plusieurs cadres d'apprentissage profond tels que TensorFlow, PyTorch, Caffe, MXNet et ONNX [23].

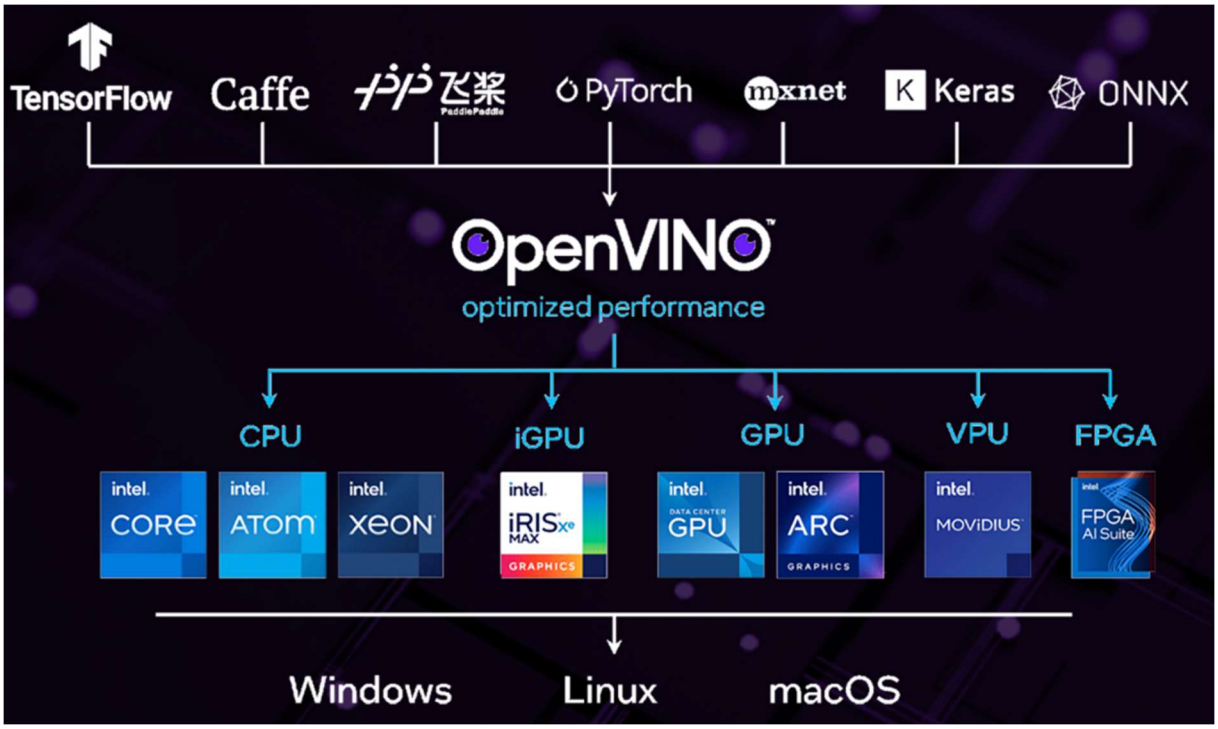

Figure 1.10. Plateforme OpenVINO.

1.8.1 L'optimisation d'un modèle

Pour une utilisation sur les appareils Edge, l'optimisation d'un modèle de machine learning consiste à le rendre efficace en termes d'utilisation de ressources limitées telles que la mémoire et la puissance de calcul. Les techniques couramment utilisées pour optimiser les modèles Edge comprennent la quantification de poids, la compression de modèle, l'optimisation d'architecture et l'utilisation de matériel spécialisé tel que les processeurs Edge TPU de Google [24] [25].

1.9. Augmentation de données

L'augmentation de données (Data augmentation en anglais) est une méthode très répandue en apprentissage automatique et en vision par ordinateur. Elle permet d'augmenter la quantité de données d'entraînement en appliquant des transformations aux données existantes. Ces transformations incluent des opérations telles que la rotation, le changement d'échelle, le recadrage, la translation, l'inversion, l'ajout de bruit ou de flou, et bien d'autres encore. Le but de cette technique est d'élargir la variabilité des données d'entraînement, évitant ainsi la suradaptation (overfitting) du modèle à un ensemble de données spécifique. En outre, l'augmentation de données peut également améliorer la résilience du modèle en permettant la reconnaissance de variations mineures dans les données, telles que des changements d'éclairage ou de perspective [26].

1.10. UFC-Crime Dataset

La base de données UCF-Crime est un ensemble de vidéos de 128 heures de durée totale, composée de 1900 vidéos longues et non coupées d'événements criminels réels, tels que des incidents violents, des arrestations, des incendies criminels, des agressions, des accidents de la route, des cambriolages, des explosions, des bagarres, des vols à main armée, des vols, des vols à l'étalage et du vandalisme. Cet ensemble de données est utilisé pour l'évaluation de méthodes de détection d'anomalies dans des vidéos de surveillance. Il contient 13 types d'anomalies sélectionnées pour leur impact significatif sur la sécurité publique [27].

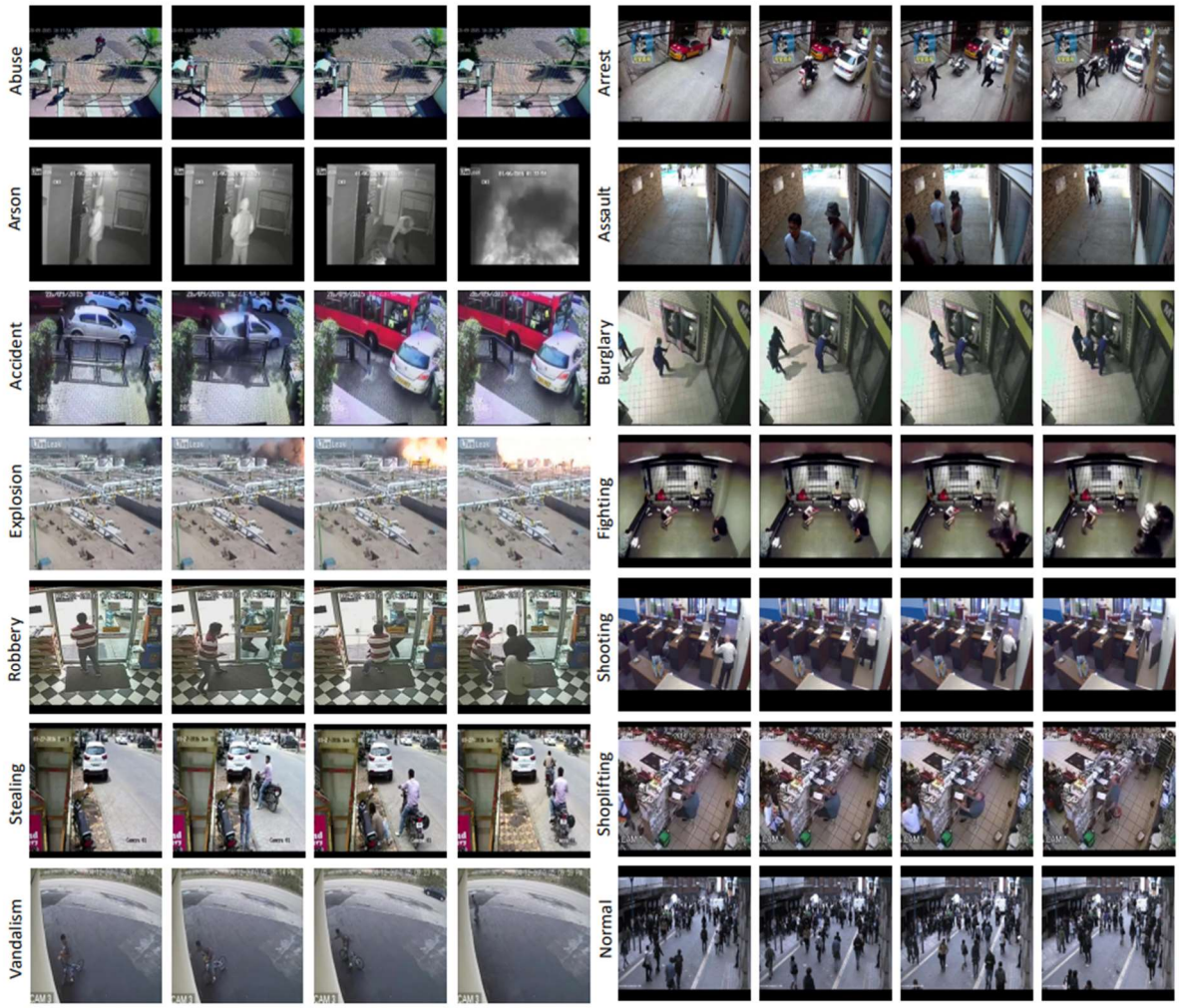

Figure 1.11. Contenu d'UFC-Crime Dataset.

Dans notre cas, nous utilisons seulement deux classes : "Shoplifting" et "NormalVideos". Dans la première classe, nous avons seulement 50 vidéos de vol, tandis que dans la deuxième, nous avons 720 vidéos.

La figure 1 .12 montre une image correspondant à une vidéo où des clients choisissent simplement des produits. L'image de la figure 1.13 correspond à une vidéo avec la présence de "vol à l'étalage".

Figure 1.12 vidéo normal.

Figure 1.13 Vol à l'étalage.

1.11. Le protocole MQTT

MQTT est un protocole de messagerie basé sur des normes, ou un ensemble de règles, utilisé pour la communication de machine à machine. Les capteurs intelligents, les objets portés sur soi (wearables) et les autres appareils de l'Internet des objets (IoT) doivent généralement transmettre et recevoir des données sur un réseau aux ressources limitées et à la

bande passante restreinte. Ces appareils IoT utilisent MQTT pour la transmission des données, car il est facile à implémenter et peut communiquer efficacement les données IoT. MQTT prend en charge la messagerie des appareils vers le cloud et du cloud vers l'appareil

Le protocole MQTT est devenu une norme pour la transmission de données IoT, car il offre les avantages suivants :

Léger et efficace : L'implémentation de MQTT sur l'appareil IoT nécessite des ressources minimales, de sorte qu'il peut même être utilisé sur de petits microcontrôleurs. Par exemple, un message de contrôle MQTT minimal peut ne contenir que deux octets de données. Les en-têtes des messages MQTT sont également de petite taille afin que vous puissiez optimiser la bande passante du réseau [28].

Évolutif : L'implémentation de MQTT nécessite une quantité minimale de code qui consomme très peu d'énergie en fonctionnement. Le protocole possède également des fonctionnalités intégrées pour prendre en charge la communication avec un grand nombre d'appareils IoT. Vous pouvez donc implémenter le protocole MQTT pour vous connecter à des millions de ces appareils [28].

Fiable : De nombreux appareils IoT se connectent sur des réseaux cellulaires peu fiables, avec une faible bande passante et une forte latence. MQTT possède des fonctionnalités intégrées qui réduisent le temps que le dispositif IoT prend pour se reconnecter au cloud. Il définit également trois niveaux de qualité de service différents pour garantir la fiabilité des cas d'utilisation IoT : au plus une fois (0), au moins une fois (1) et exactement une fois (2) [28].

Sécurisé : MQTT permet aux développeurs de chiffrer facilement les messages et d'authentifier les appareils et les utilisateurs à l'aide de protocoles d'authentification modernes, tels que OAuth, TLS1.3, Customer Managed Certificates, etc [28].

Bonne prise en charge : Plusieurs langages comme Python disposent d'un support étendu pour implémenter le protocole MQTT. Les développeurs peuvent donc l'implémenter rapidement avec un codage minimal dans tout type d'application [28].

Conclusion

En conclusion, les concepts présentés dans ce chapitre sont tous des éléments clés dans le domaine de la sécurité dans un immeuble commercial. Le vol à l'étalage, la

vidéosurveillance, l'IOT, la vidéo, l'Edge Computing et la vision par ordinateur sont des technologies qui ont un impact direct sur la sécurité des individus et des entreprises. Par conséquent, il est possible de mieux appréhender les risques et de mettre en place des mesures de sécurité adéquates pour assurer la protection de chacun. Il est donc important de continuer à s'informer sur ces technologies en constante évolution afin de rester à jour sur les enjeux de sécurité qui en découlent. Dans le chapitre suivant, nous allons présenter un résumé de trois importants articles qui sont étroitement liés à notre étude, en mettant en évidence leurs contributions principales et leurs limites éventuelles.

Chapitre 02 :

Détection du vol à l'étalage par une approche intelligente : Travaux connexes

Introduction

Dans ce chapitre, nous présentons un résumé de trois importants articles qui sont étroitement liés à notre étude, en mettant en évidence leurs contributions principales et leurs limites éventuelles.

Après avoir présenté ces travaux, nous discutons de la façon dont ils se rapportent à notre recherche, en soulignant les similitudes et les différences avec notre approche. Nous mettons également en évidence les lacunes actuelles dans les travaux antérieurs, qui justifient la nécessité de poursuivre les recherches dans ce domaine.

2.1. Travaux similaires

2.1.1 Detection of Shoplifting on Video Using a Hybrid Network [29]

Objectif de cette étude : Dans cette étude [29], Kirichenko et al. présentent une solution pour détecter le vol à l'étalage dans les enregistrements vidéo en utilisant un réseau neuronal hybride qui combine des réseaux de convolution et récurrents. L'article décrit l'utilisation de l'ensemble de données UCF-Crime pour former les ensembles d'entraînement et de test, ainsi que les résultats de classification qui ont montré une précision élevée de 93%. L'approche proposée combine les points forts des CNN et des RNN pour extraire des caractéristiques spatiales et temporelles des données vidéo. L'article discute également l'utilisation d'unités récurrentes à portes (Gate recurrent Unit -GRU-) dans une méthode de classification de vidéos pour la reconnaissance du vol à l'étalage. Le document [29] propose un classificateur de réseau neuronal hybride pour identifier le comportement de vol à l'étalage dans les images de caméras de surveillance, qui a le potentiel de fonctionner en temps réel dans les centres commerciaux.

Approches utilisées :

Plusieurs approches ont été utilisées dans cet article pour résoudre le problème de la détection du vol à l'étalage dans les enregistrements vidéo. Tout d'abord, une combinaison de réseaux de convolution et récurrents a été utilisée pour extraire des caractéristiques spatiales et temporelles des données vidéo. Ensuite, un réseau neuronal hybride a été proposé pour identifier le comportement de vol à l'étalage dans les images de caméras de surveillance. Ce réseau combine les forces des CNN et des RNN pour améliorer la précision de la

classification. Enfin, l'article discute également de l'utilisation d'unités récurrentes à portes (GRU) dans une méthode de classification vidéo pour la reconnaissance du vol à l'étalage.

Outils utilisés :

L'article mentionne l'utilisation de plusieurs outils pour mener à bien la recherche. Tout d'abord, le programme pour résoudre le problème de reconnaissance du vol à l'étalage a été écrit en Python. Ensuite, l'environnement interactif de calcul web Jupyter Notebook a été utilisé pour traiter l'ensemble de données, tandis que l'environnement Colaboratory de Google Research a été utilisé pour construire l'architecture du réseau neuronal et entraîner et tester le modèle. Le programme a été exécuté dans un environnement cloud avec des processeurs Tensor de Google, ce qui a permis d'obtenir une vitesse élevée lors de l'utilisation de bibliothèques d'apprentissage automatique telles que TensorFlow. Enfin, l'article mentionne également l'utilisation de réseaux de convolution tels que InceptionV3 et MobileNetV3Large, ainsi que des réseaux de neurones récurrents avec des nœuds à portes pour extraire des caractéristiques des données vidéo.

L’architecture de la solution proposée :

Cette architecture utilise une combinaison de réseaux de convolution et récurrents pour extraire des caractéristiques spatiales et temporelles des données vidéo. Plus précisément, l'article décrit l'utilisation d'un réseau de convolution MobileNetV3Large pour extraire des caractéristiques spatiales des images vidéo, qui sont ensuite fournies à un réseau neuronal récurrent avec des nœuds à portes (GRU) pour extraire des caractéristiques temporelles. Le réseau neuronal hybride ainsi créé est utilisé pour identifier le comportement de vol à l'étalage dans les images de caméras de surveillance. L'article décrit également l'architecture du modèle de séquence (classificateur récurrent) utilisé pour la classification vidéo, qui comprend une couche d'entrée GRU, une couche GRU suivante, une couche de suppression aléatoire pour réduire le surapprentissage, une couche entièrement connectée avec une fonction d'activation ReLU et une couche de sortie entièrement connectée avec une fonction d'activation sigmoïde.

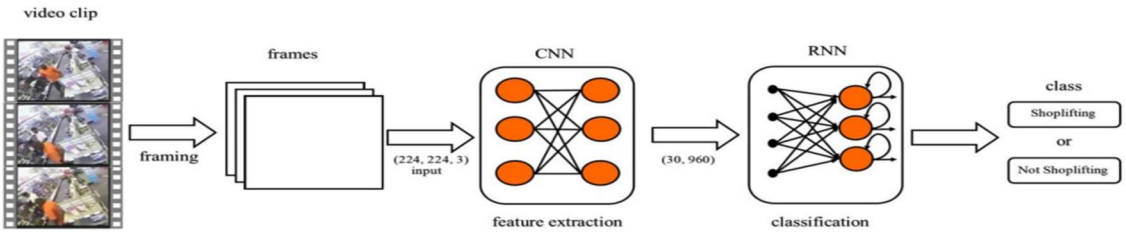

Figure 2.1. L'architecture générale du premier article [29].

Résultats atteints :

L'article présente des résultats prometteurs pour la détection du vol à l'étalage dans les enregistrements vidéo. Le modèle de réseau neuronal hybride proposé a atteint une précision de classification élevée de 93% sur l'ensemble de données UCF-Crime. Les auteurs ont également effectué des expériences pour augmenter artificiellement la taille de l'ensemble de données, ce qui a entraîné une augmentation de la précision de 5 à 7%. L'article présente également une matrice de confusion et des métriques de précision, de rappel et de score F1 pour évaluer les performances du modèle. Les résultats montrent une précision élevée pour les deux classes (0,92 pour le comportement normal et 0,93 pour le vol à l'étalage). Les auteurs ont également comparé leurs résultats avec d'autres études similaires et ont constaté que leur modèle avait une précision supérieure à celle des autres études.

Batch Size	Iterations	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy
4	228	99.34	0.026	90.28	0.407	90.14
8	114	98.02	0.066	92.84	0.247	91.27
16	57	99.34	0.027	90.79	0.326	92.11
32	29	98.57	0.051	90.79	0.302	92.83
64	15	99.45	0.015	93.09	0.338	93.19

Figure 2.2. Les résultats présentés dans [29].

Limitation de cette solution :

Il est possible d'extraire certaines limitations potentielles de cette étude. Tout d'abord, l'ensemble de données utilisé pour former le modèle (UCF-Crime) peut ne pas être représentatif de toutes les situations de vol à l'étalage dans les magasins. Par conséquent, le modèle peut ne pas être capable de détecter certains comportements de vol à l'étalage qui ne sont pas inclus dans l'ensemble de données. De plus, l'article ne mentionne pas la capacité du modèle à détecter les faux positifs, c'est-à-dire les situations où le modèle signale un vol à l'étalage qui n'a pas eu lieu. Enfin, l'article ne discute pas de la capacité du modèle à

fonctionner dans des conditions de faible éclairage ou de qualité vidéo médiocre, ce qui peut affecter la précision de la classification.

2.1.2 An Automatic Shoplifting Detection from Surveillance Videos [30]

Objectif de cette étude :

Dans [30], Gim et al. proposent une méthode de détection automatique des comportements de vol à l'étalage à partir de vidéos de surveillance en utilisant le réseau de fusion optique de la région d'intérêt (ROI) pour mettre en évidence les caractéristiques nécessaires de manière plus précise. La méthode proposée extrait un objet de personne en tant que ROI à l'aide de Mask-R-CNN, puis détermine le comportement de vol à l'étalage en utilisant la quantité de changement dans l'objet de personne ROI à l'aide du flux optique. L'objectif de l'étude est de détecter le comportement de vol à l'étalage sans utiliser les caractéristiques de l'ensemble des images vidéo.

Approches utilisées:

L'article décrit une approche pour la détection automatique des comportements de vol à l'étalage à partir de vidéos de surveillance en utilisant le réseau de fusion optique de la région d'intérêt (ROI) pour mettre en évidence les caractéristiques nécessaires de manière plus précise. La méthode proposée extrait un objet de personne en tant que ROI à l'aide de Mask-R-CNN, puis détermine le comportement de vol à l'étalage en utilisant la quantité de changement dans l'objet de personne ROI à l'aide du flux optique. Cette approche diffère des méthodes précédentes qui extraient des caractéristiques liées à la quantité de changement des actions dans l'ensemble de l'image. L'article mentionne également d'autres approches proposées pour la détection de comportements anormaux à partir de vidéos de surveillance, telles que AVS-NET, la détection d'anomalies en temps réel et la détection d'anomalies à l'aide de l'apprentissage en profondeur.

Outils utilisés:

L'article mentionne plusieurs outils utilisés pour la détection automatique des comportements de vol à l'étalage à partir de vidéos de surveillance. Tout d'abord, l'article utilise Mask-R-CNN pour extraire un objet de personne en tant que ROI à partir des vidéos de surveillance. Ensuite, l'article utilise le flux optique pour déterminer le comportement de vol à l'étalage en utilisant la quantité de changement dans l'objet de personne ROI. En outre,

l'article mentionne également l'utilisation de réseaux de neurones pour la prédiction des comportements de vol à l'étalage. Plus précisément, l'article utilise un réseau de neurones à trois couches entièrement connectées avec l'optique de ROI en entrée pour prédire les scores de comportement normal et de vol à l'étalage. Enfin, l'article mentionne également l'utilisation de plusieurs autres méthodes proposées pour la détection de comportements anormaux à partir de vidéos de surveillance, telles que AVS-NET, la détection d'anomalies en temps réel et la détection d'anomalies à l'aide de l'apprentissage en profondeur.

L’architecture de la solution :

L'article décrit l'architecture proposée pour la détection automatique des comportements de vol à l'étalage à partir de vidéos de surveillance. Cette architecture comprend deux modules : l'annotation segmentée d'image et le réseau de fusion optique de la région d'intérêt (ROI). Le module d'annotation segmentée d'image est utilisé pour extraire les comportements de vol à l'étalage et les comportements normaux en définissant une séquence d'une ROI. Pour ce faire, l'article utilise Mask-R-CNN pour extraire un objet de personne en tant que ROI à partir des vidéos de surveillance. Ensuite, l'article utilise le flux optique pour déterminer le comportement de vol à l'étalage en mesurant la quantité de changement dans l'objet de personne ROI. Le module de fusion optique de la région d'intérêt (ROI) utilise une séquence de trois images de ROI en entrée pour extraire le flux optique et prédire les comportements de vol à l'étalage et les comportements normaux.

L'article décrit également l'architecture du réseau de neurones utilisé pour la prédiction des comportements de vol à l'étalage, qui est un réseau de neurones à trois couches entièrement connectées avec l'optique de ROI en entrée. En somme, l'architecture de la solution proposée utilise Mask-R-CNN pour extraire une ROI à partir des vidéos de surveillance, puis utilise le flux optique pour détecter les comportements de vol à l'étalage et les comportements normaux. La solution utilise également un réseau de neurones pour la prédiction des comportements de vol à l'étalage.

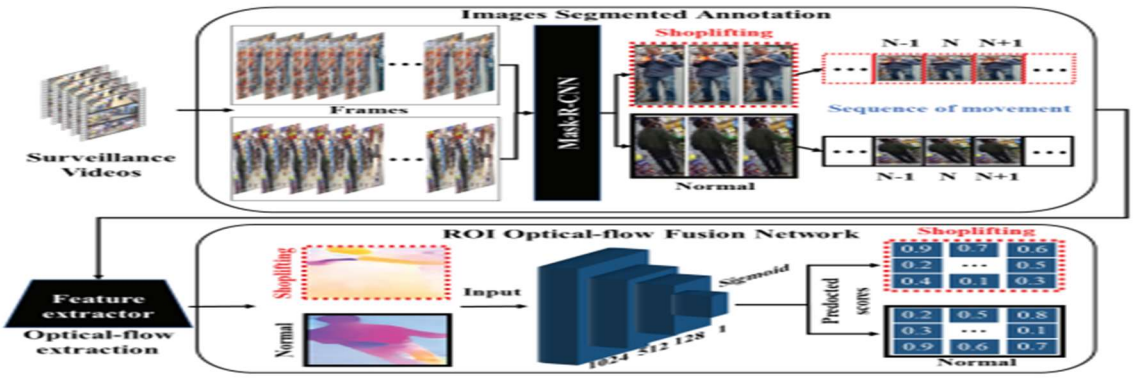

Figure 2.3. L'architecture générale du deuxième article [30].

Résultats obtenus :

L'article ne fournit pas de résultats expérimentaux détaillés, mais il mentionne que la méthode proposée a été testée sur des vidéos de surveillance pour détecter les comportements de vol à l'étalage. L'article indique également que la méthode proposée utilise le flux optique de la ROI pour extraire les caractéristiques nécessaires de manière plus précise, ce qui permet de détecter les comportements de vol à l'étalage sans utiliser les caractéristiques de l'ensemble des images vidéo. En outre, l'article mentionne que des expériences plus approfondies seront menées à l'avenir pour démontrer l'efficacité de la méthode proposée.

Limitations de cette solution :

Cette étude soufre de certaines limitations potentielles, on note que la méthode proposée utilise Mask-R-CNN pour extraire un objet de personne en tant que ROI à partir des vidéos de surveillance. Cela peut poser des problèmes si la qualité de la vidéo est faible ou si l'objet de personne n'est pas clairement visible dans la vidéo. De plus, la méthode proposée utilise le flux optique pour déterminer le comportement de vol à l'étalage en utilisant la quantité de changement dans l'objet de personne ROI. Cela peut ne pas être efficace si le comportement de vol à l'étalage est subtil ou s'il y a peu de changement dans l'objet de personne ROI. Enfin, l'article ne fournit pas de résultats expérimentaux détaillés pour démontrer l'efficacité de la méthode proposée, il est donc difficile de savoir si elle est efficace dans la détection des comportements de vol à l'étalage dans des conditions réelles.

2.1.3 Suspicious Behavior Detection on Shoplifting Cases for Crime Prevention by Using 3D Convolutional Neural Networks [31]

Objectif:

L'objectif de cette recherche est de proposer une méthode pour détecter les comportements suspects avant qu'un vol à l'étalage ne se produise en utilisant des réseaux de neurones convolutifs 3D. Martínez-Mascorro et al. se concentrent sur l'identification des comportements qui peuvent conduire à un crime plutôt que sur le crime lui-même, afin de fournir au personnel de surveillance plus d'opportunités pour prévenir les crimes. Ils proposent également une nouvelle méthode appelée "analyse des comportements pré-criminels" pour traiter les vidéos de surveillance et extraire les segments qui présentent des comportements suspects. Le document explore différentes configurations de paramètres système pour améliorer les performances du modèle et discute des études et des applications de l'apprentissage en profondeur et des réseaux de neurones convolutifs pour détecter les comportements anormaux et reconnaître les actions dans les scènes bondées à l'aide d'images et de vidéos de caméras de surveillance.

Les approches utilisées :

L'article mentionne plusieurs approches utilisées pour la surveillance, notamment la détection de mouvement, la reconnaissance faciale, le suivi, la détection de rôdeurs, la détection de bagages abandonnés, le comportement de foule et le comportement anormal. Les auteurs soulignent également que la plupart des approches se concentrent sur la détection de crimes plutôt que sur la prévention, et que la disponibilité limitée de données de surveillance peut poser des problèmes pour l'apprentissage en profondeur. Ils proposent donc une nouvelle méthode pour extraire les segments de vidéos de surveillance qui présentent des comportements suspects avant qu'un crime ne se produise, en utilisant des réseaux de neurones convolutifs 3D.

Outils utilisés :

Plusieurs outils pour la surveillance ont été envisagés, notamment la détection de mouvement, la reconnaissance faciale, le suivi, la détection de rôdeurs, la détection de bagages abandonnés, le comportement de foule et le comportement anormal. Pour la détection des comportements suspects avant qu'un vol à l'étalage ne se produise, les auteurs proposent l'utilisation de réseaux de neurones convolutifs 3D. Ils ont également utilisé Google Colaboratory, un outil de cloud computing gratuit pour l'écriture et l'exécution de code en utilisant une GPU virtuelle pour l'entraînement des modèles d'apprentissage en profondeur. Les auteurs ont également utilisé le jeu de données UCF-Crimes pour tester leur modèle.

L’architecture de la solution :

L'article décrit l'utilisation d'une architecture de réseau de neurones convolutifs 3D pour détecter les comportements suspects avant qu'un vol à l'étalage ne se produise. Les auteurs ont exploré différentes configurations de paramètres système pour améliorer les performances du modèle, notamment la profondeur, le nombre de couches, le nombre de filtres sur chaque couche, la taille du noyau, le padding et la stride. Ils ont également comparé leur architecture proposée à une architecture de référence de réseau de neurones convolutifs 3D. Les résultats ont montré que l'architecture proposée a dépassé 90% de précision dans les deux cas de données équilibrées et non équilibrées, et a réussi à classer avec succès 100% des échantillons suspects tout en obtenant un faible nombre de faux positifs à partir d'échantillons de comportement normal.

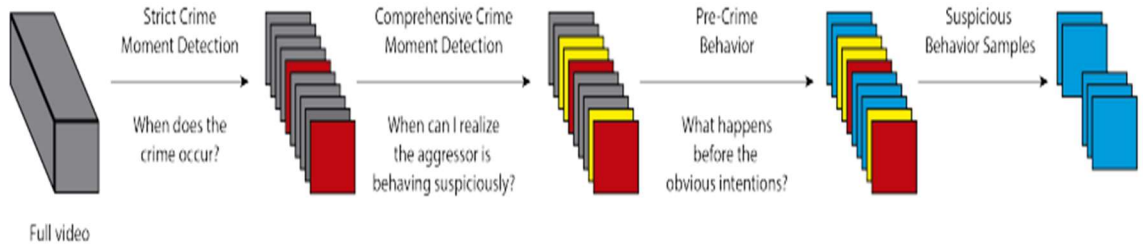

Figure 2.4. L'architecture générale du troisième article [31].

Résultats obtenus :

L'article présente les résultats de l'expérimentation menée pour détecter les comportements suspects avant qu'un vol à l'étalage ne se produise en utilisant des réseaux de neurones convolutifs 3D. Les auteurs ont exploré différentes configurations de paramètres système pour améliorer les performances du modèle et ont comparé leur architecture proposée à une architecture de référence de réseau de neurones convolutifs 3D. Les résultats ont montré que l'architecture proposée a dépassé 90% de précision dans les deux cas de données équilibrées et non équilibrées, et a réussi à classer avec succès 100% des échantillons suspects tout en obtenant un faible nombre de faux positifs à partir d'échantillons de comportement normal. Les auteurs ont également constaté que la résolution de 80x60 a donné les meilleurs résultats en termes de détection de comportements suspects, avec des taux de précision supérieurs à 85% pour les ensembles de données équilibrés et non équilibrés.

Dataset	Resolution			
	32x24	40x30	80x60	160x120
SBT_balanced_120_40t_10f	71.5%	71.5%	77.0%	77.0%
SBT_balanced_120_40t_10f_flip	73.6%	77.0%	83.3%	70.8%
SBT_balanced_120_30t_10f	75.9%	71.3%	71.3%	79.6%
SBT_balanced_120_30t_10f_flip	76.8%	72.2%	81.5%	78.6%

Figure 2.6. Les résultats obtenus par l'approche "analyse des comportements pré-criminels" [31].

Limitation de cette solution :

Plusieurs limitations de cette solution peuvent être citées, à savoir la disponibilité limitée de données de surveillance qui peut poser des problèmes pour l'apprentissage en profondeur. De plus, la surveillance matérielle est souvent la propriété privée d'une entreprise, ce qui limite la quantité de données disponibles pour entraîner et tester de nouveaux modèles de surveillance. De plus, les vidéos de surveillance ne contiennent souvent qu'une petite fraction des occurrences de crimes, ce qui peut rendre difficile la détection de comportements suspects. Enfin, bien que l'architecture proposée ait montré des résultats prometteurs, elle nécessite encore des améliorations pour être utilisée dans des environnements de surveillance réels.

D'autres travaux similaires

Les approches les plus courantes pour appliquer l'intelligence artificielle à la surveillance vidéo comprennent principalement la détection de mouvement, la reconnaissance faciale, la surveillance de l'inactivité et la détection de comportements anormaux [32].

Dans [33], les auteurs ont proposé un modèle d'algorithme génétique qui générait des réseaux neuronaux pour classifier les comportements humains dans les vidéos. Les auteurs ont développé des réseaux neuronaux peu profonds et profonds qui utilisaient les changements de posture des personnes dans les séquences vidéo en tant qu'entrée.

Pour détecter les vols violents dans les séquences vidéo, [34] a proposé l'utilisation d'un modèle de séquence d'apprentissage profond dans lequel un extracteur de caractéristiques était formé. Les caractéristiques ont ensuite été traitées avec deux couches de mémoire de convolution à long terme et passées à travers des couches entièrement connectées pour la classification.

Dans l'article [35], les auteurs ont présenté une approche de détection d'anomalies en utilisant un modèle C3D préentraîné pour l'extraction de caractéristiques et un réseau neuronal pleinement connecté pour effectuer la régression. En utilisant la base de données UCF-Crime, les auteurs ont entraîné leur modèle sur 11 classes et ont testé ses résultats sur des vidéos de vols, de bagarres et d'incidents de circulation.

Les auteurs de [36] ont présenté une approche pour détecter des anomalies en temps réel; l'algorithme a été entraîné pour classer 13 comportements anormaux tels que le vol et le cambriolage, la bagarre, la fusillade et le vandalisme. Ils ont utilisé un réseau de neurones 3D CNN pour extraire les caractéristiques et étiqueter les échantillons en deux catégories: normale et anormale. Leur modèle incluait une fonction de perte de notation et entraînait un réseau neuronal pleinement connecté pour prendre des décisions.

Dans [37], les auteurs ont analysé la présence de violence dans les vidéos de surveillance. À cette fin, les auteurs ont proposé l'utilisation d'un modèle d'apprentissage profond basé sur les réseaux neuronaux convolutifs 3D sans utiliser de fonctions manuelles ou une architecture RNN exclusivement pour l'encodage de l'information temporelle. Pour évaluer l'efficacité et l'efficience du modèle, les auteurs l'ont testé expérimentalement sur trois ensembles de données de référence.

Dans [38], les auteurs ont étudié les procédures de classification vidéo existantes afin de recommander le processus le plus efficace et productif. Les auteurs ont montré que l'utilisation combinée d'un CNN (réseau neuronal convolutif) et d'un RNN (réseau neuronal récurrent) était plus performante que les méthodes dépendantes du CNN.

Dans le manuscrit [39], l'auteur a été proposé de combiner un réseau 3D CNN et LSTM pour prédire les actions humaines. Dans cette approche, le 3D CNN a été utilisé pour l'extraction de caractéristiques et LSTM pour la classification. Le résultat du modèle dépendait de la pose, de l'éclairage et de l'environnement. La fiabilité de ces caractéristiques a permis la prédiction des actions humaines.

Discussion

L'application de l'intelligence artificielle à la surveillance vidéo est de plus en plus répandue. Les approches les plus courantes incluent la détection de mouvement, la reconnaissance faciale, la surveillance de l'inactivité et la détection de comportements anormaux. Les auteurs de différentes approches ont proposé des modèles d'algorithme

génétique, des réseaux neuronaux peu profonds et profonds, ainsi que des modèles de séquence d'apprentissage profond pour classifier les comportements humains dans les vidéos et détecter les vols violents. Certains ont également utilisé des modèles C3D préentraînés pour l'extraction de caractéristiques et des réseaux de neurones pleinement connectés pour effectuer la régression ou la classification.

D'autres auteurs ont proposé des approches pour détecter des anomalies en temps réel en classant différents comportements anormaux tels que le vol, le cambriolage, la bagarre, la fusillade et le vandalisme. Ils ont utilisé des réseaux de neurones 3D CNN pour extraire les caractéristiques et étiqueter les échantillons en deux catégories: normale et anormale. Certains ont analysé la présence de violence dans les vidéos de surveillance en utilisant un modèle d'apprentissage profond basé sur les réseaux neuronaux convolutifs 3D. D'autres ont étudié les procédures de classification vidéo existantes afin de recommander le processus le plus efficace et productif.

Enfin, certains ont proposé de combiner un réseau 3D CNN et LSTM pour prédire les actions humaines. Dans cette approche, le 3D CNN a été utilisé pour l'extraction de caractéristiques et LSTM pour la classification. Le résultat du modèle dépendait de la pose, de l'éclairage et de l'environnement, et la fiabilité de ces caractéristiques a permis la prédiction des actions humaines. Ces différentes approches permettent de mieux comprendre et de détecter les comportements anormaux dans les vidéos de surveillance, ce qui peut être utile pour prévenir les actes criminels et renforcer la sécurité.

Conclusion

En conclusion, les travaux antérieurs dans le domaine de l'application de l'intelligence artificielle à la surveillance vidéo ont connu une évolution remarquable, mais il reste encore des lacunes à combler. Les modèles proposés ont permis de mieux comprendre et de détecter le comportement anormal le vol en temps réel. Toutefois, des défis subsistent, tels que la qualité des données de surveillance et la complexité des scénarios réels.

En comparant les travaux antérieurs, nous avons identifié les lacunes qui justifient la nécessité de poursuivre la recherche dans ce domaine et comment notre recherche peut contribuer à combler certaines des lacunes actuelles dans les travaux antérieurs. Dans l'ensemble, ces différentes approches ont le potentiel de prévenir les actes criminels et de renforcer la sécurité dans les espaces publics, mais il reste encore un peu à faire pour

améliorer l'efficacité et la fiabilité de ces systèmes ; par conséquent, nous proposons dans le chapitre suivant notre approche dans l'objectif de surmonter les lacunes recensées.

Chapitre 03 :

Conception d’une solution d’Edge IA pour la détection de vol à l’étalage

Introduction

Dans ce chapitre, nous proposons un système de détection de vol à l'étalage à partir d'une vidéo ou d'une caméra de surveillance déployer dans l'Edge IA Basé sur une approche CNN-ANN. Cette conception explore les différentes étapes impliquées dans la création de cette solution, en commençant par une analyse approfondie des exigences et des contraintes du projet, suivie d'une exploration des différentes options de conception et de leur évaluation. Nous présentons également les différents diagrammes décrivant le fonctionnement de notre système, donnons un aperçu de son architecture générale et de sa méthodologie, et détaillons chaque élément de manière approfondie. La conception détaillée est ensuite présentée, avec une justification claire des choix effectués. Ce chapitre sert de base pour la mise en œuvre pratique de la solution, qui sera abordée dans le chapitre suivant.

3.1 Conception générale du système proposé

Notre système passe par trois étapes qui sont :

- Création de modèles pour la détection de vol à l'étalage à partir d'une vidéo.
- Optimisation de nos modèles en utilisant la plateforme OpenVino.
- Implémentation de ce système dans l'Edge IA.

Dans la première étape, "Création de modèles pour la détection de vol à l'étalage à partir d'une vidéo", nous avons cherché une base de données contenant des vidéos de vols à l'étalage ainsi que des vidéos normales de shopping dans des immeubles commerciaux. Nous avons finalement sélectionné la base de données UCF-Crime. Ensuite, nous avons suivi plusieurs étapes, dont le prétraitement des vidéos, l'apprentissage et finalement le test, pour aboutir à deux modèles : l'un pour l'extraction des caractéristiques et l'autre pour la classification des vidéos.

Ensuite, nous sommes passés à la deuxième étape, "Optimisation de nos modèles en utilisant la plateforme OpenVino". Nous avons cherché à optimiser nos modèles pour améliorer leur rapidité d'utilisation ou pour minimiser leur taille afin de pouvoir les implémenter sur des appareils Edge et bénéficier de ces avantages.

Après cette étape, nous avons implémenté notre système dans un réseau IoT en utilisant le protocole MQTT et avons créé une interface pour notre tableau de bord (Dashboard). Nous

avons ensuite déployé notre travail en dockerisant notre modèle via OVMS et le tableau de bord.

L'architecture générale de notre système est illustrée par la figure suivante :

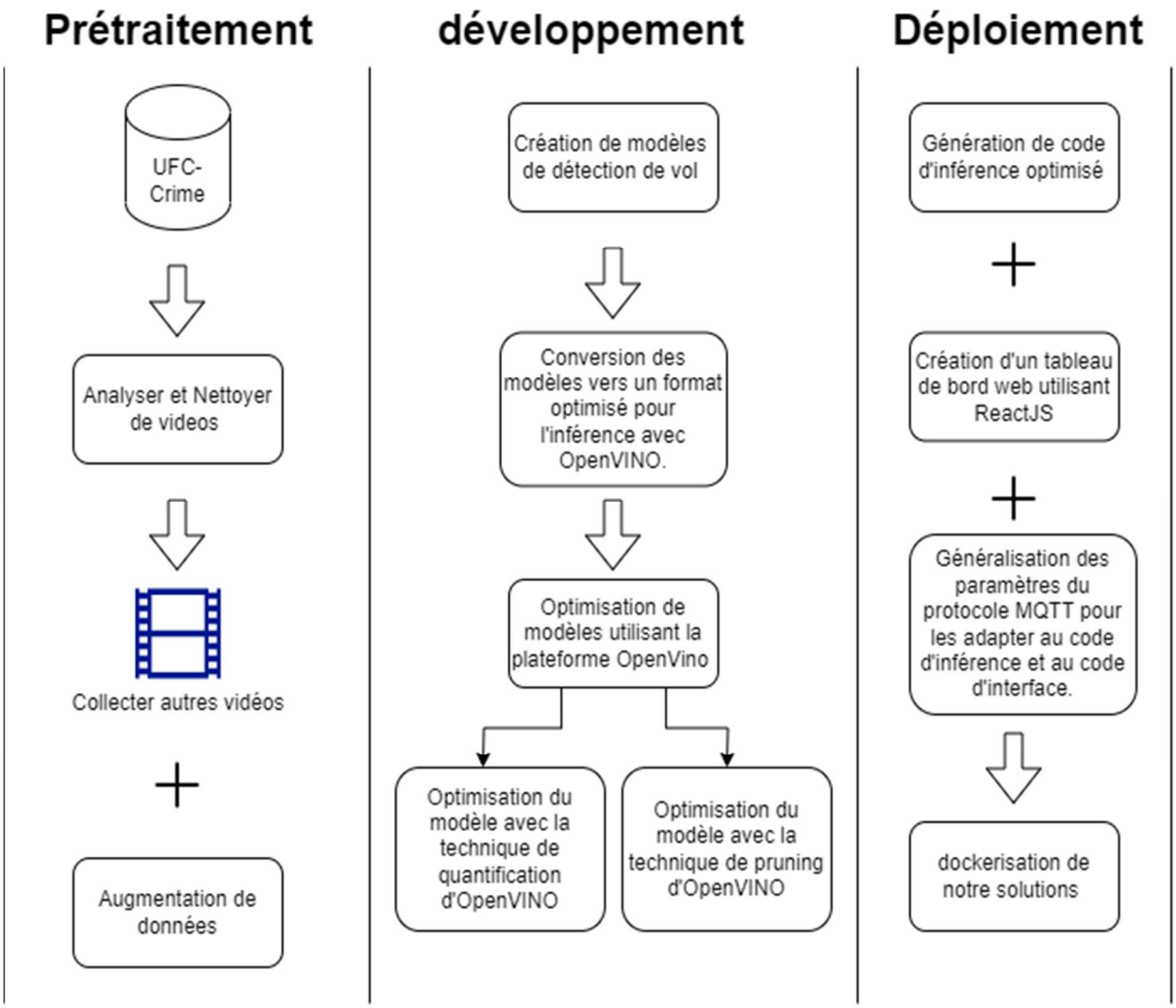

Figure 3.1. L'architecture générale de notre système.

3.2 Conception détaillée

3.2.1 Création de notre modèle pour la détection de vol à l'étalage à partir d'une vidéo

Comme mentionné précédemment, le processus de détection de vol à l'étalage en temps réel commence par la création d'un sous-système de détection à partir d'une vidéo, car cette phase est la base de notre projet et nous permet de déterminer si une vidéo correspond à un vol ou non. Nous avons utilisé l'architecture de la base de données UFC-Crime pour notre

modèle, en modifiant les entrées et les couches du modèle C3D pré-entraîné, afin de créer un modèle de classification simple qui peut être implémenté sur l'Edge et bénéficier de ses avantages, comme illustré dans la figure suivante.

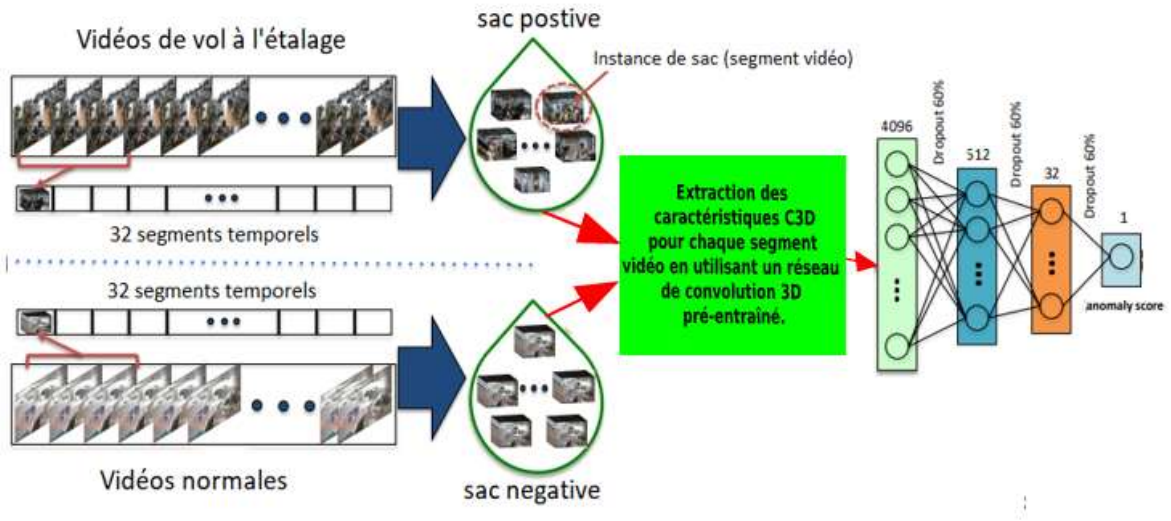

Figure 3.2. L'architecture générale de notre modèle.

Afin de collecter des vidéos de vols à l'étalage ainsi que des vidéos normales pour notre processus d'apprentissage et de test de modèle, nous avons été confrontés à un manque de ressources (vidéo). Dans notre base de données, nous avons trouvé seulement 280 vidéos de classe normale et 50 vidéos de vols à l'étalage. Pour résoudre ce problème, nous avons acheté quelques vidéos de vols sur internet. Cependant, même avec ces vidéos supplémentaires, nous avons dû utiliser des techniques d'augmentation de données pendant la phase de collection pour augmenter la quantité de données.

Au cours de cette étape, nous avons effectué cinq opérations :

A. **Collecte et nettoyage de vidéos :** Nous avons déjà identifié le problème de manque de vidéos et avons mentionné précédemment que nous avons résolu ce problème en collectant 15 vidéos supplémentaires de vols à l'étalage. Nous avons également nettoyé les vidéos de classe normale, ne gardant que celles qui présentent des symboles commerciaux, ce qui nous a permis d'obtenir que de 280 vidéos normale.

B. Analyse de vidéos : Nous avons analysé les vidéos et avons constaté que la plupart étaient trop longues (entre 10 et 35 minutes), avec seulement une petite partie de chaque vidéo présentant un vol (environ 20 à 40 secondes). Pour analyser les 50 vidéos longues et découper celles qui présentent un vol, nous aurions besoin de

beaucoup de temps et d'efforts. Nous avons donc décidé de ne travailler que sur les 35 vidéos les plus pertinentes. Cela nous a amenés à rencontrer à nouveau le problème de manque de données, ce qui nous a conduits à ajouter l'étape suivante.

C. **Augmentation de données :** Dans cette étape, nous avons utilisé des techniques d'augmentation de données pour augmenter la quantité de données dans notre ensemble de vidéos. Étant donné que chaque vidéo ne représente qu'un ou deux vols maximum, nous avons choisi de ne pas découper les vidéos en plusieurs segments. Nous avons plutôt utilisé des techniques telles que le changement de couleur, qui permet d'ajuster la teinte, la saturation et la luminosité de la vidéo, ainsi que la superposition, qui consiste à ajouter des éléments visuels tels que du texte ou des images pour fournir des informations supplémentaires. Nous avons ainsi obtenu un total de 70 vidéos de vols à l'étalage pour l'apprentissage, ce qui nous permet d'observer des comportements réels de vols à l'étalage.

D. **Prétraitement :** Le prétraitement de la vidéo pour l'extraction de caractéristiques implique plusieurs étapes :

   Tout d'abord, nous avons prétraité chaque image de la vidéo. Nous avons commencé par extraire 16 images (intervalles) de la vidéo en prenant des images à des moments équidistants. Ensuite, nous redimensionnons chaque image en 171x128 pixels et soustrayons la moyenne du fichier c3d_mean.npy (utilisé pour normaliser les images). Nous redimensionnons ensuite chaque image en 112x112, ajoutons une dimension supplémentaire pour les échantillons et retournons le résultat. Ensuite, nous avons utilisé deux fonctions (interpolate et extrapolate) pour interpoler et extrapoler les caractéristiques d'une vidéo. La fonction interpolate est utilisée pour interpoler les caractéristiques d'une vidéo. Elle prend en entrée des caractéristiques et un nombre de caractéristiques par sac. Elle interpole ensuite les caractéristiques en prenant une moyenne pondérée de caractéristiques voisines et normalise les vecteurs résultants. La fonction extrapolate est utilisée pour extrapoler les caractéristiques d'une vidéo. Elle prend en entrée des caractéristiques et le nombre de trames à extrapoler. Elle utilise ensuite une interpolation linéaire pour extrapoler les caractéristiques.

   En général, nous avons utilisé les fonctions de prétraitement pour extraire des caractéristiques de chaque clip vidéo en appliquant la fonction preprocess_input. Les caractéristiques sont ensuite interpolées et extrapolées à l'aide des fonctions interpolate et extrapolate, respectivement. Les caractéristiques résultantes sont utilisées comme entrée pour le modèle d'extraction de caractéristiques.

E. **Extraction des caractéristiques :** Nous extrayons des caractéristiques visuelles de la couche FC6 (fully connected) du modèle C3D. Avant de calculer les caractéristiques, nous redimensionnons chaque image vidéo en 240 x 320 pixels et fixons le taux de trame à 30 images par seconde. Nous calculons les caractéristiques C3D pour chaque clip vidéo de 16 images, suivi d'une normalisation. Pour obtenir les caractéristiques d'un segment vidéo, nous prenons la moyenne de toutes les caractéristiques des clips de 16 images de ce segment. Nous avons enregistré ces caractéristiques (4096 Dimension) de toutes nos vidéos collectées dans un fichier texte (chaque vidéo dans un fichier séparé) pour les utiliser ultérieurement afin d'entraîner notre modèle de classification.

F. Modèle de classification : Nous utilisons un réseau de neurones à trois couches entièrement connectées (FC). La première couche FC comprend 512 unités, suivies de couches FC de 32 et 1 unité. Une régularisation de dropout de 60% est utilisée entre les couches FC. Nous avons expérimenté avec des réseaux plus profonds mais n'avons pas observé une meilleure précision de détection. Nous utilisons ReLU comme fonction d'activation pour la première couche et la fonction d'activation Sigmoid pour la dernière couche FC, et nous employons l'optimiseur Adagrad avec un taux d'apprentissage initial de 0,001. Les paramètres des contraintes de parcimonie et de régularité dans la perte de classement MIL sont fixés à $\lambda 1 = \lambda 2 = 8 \times 10{-}5$ pour obtenir les meilleures performances. Nous divisons chaque vidéo en 32 segments non chevauchants et considérons chaque segment vidéo comme une instance du sac. Le nombre de segments (32) est fixé empiriquement. Nous avons également expérimenté avec des segments temporels superposés à plusieurs échelles, mais cela n'affecte pas la précision de la détection. Nous sélectionnons aléatoirement 30 sacs positifs et 30 sacs négatifs comme mini-lots. Nous calculons les gradients par différenciation automatique de mode inverse sur le graphe de calcul. Plus précisément, nous identifions l'ensemble de variables sur lesquelles la perte dépend, calculons le gradient pour chaque variable et obtenons le gradient final grâce à la règle de la chaîne sur le graphe de calcul. Chaque vidéo traverse le réseau et nous obtenons le score pour chacun de ses segments temporels. Ensuite, nous calculons la perte et effectuons une rétropropagation de la perte pour l'ensemble du lot.

G. Test : Pour évaluer notre modèle de classification, nous avons suivi les étapes suivantes :

Tout d'abord, nous avons choisi de tester 17 vidéos de vols à l'étalage, chacune de ces vidéos étant longue et contenant un vol à l'étalage à un moment donné, suivi du reste de la vidéo où l'on peut voir des gens faire du shopping dans un centre commercial. Nous avons également choisi 40 vidéos de classe normale, pour un total de 67 nouvelles vidéos à utiliser pour tester notre modèle.

Ensuite, nous avons extrait les caractéristiques de chaque vidéo, comme nous l'avons mentionné dans les étapes précédentes. Nous avons ensuite fait passer ces vidéos à notre modèle de classification. Nous avons créé une fonction pour calculer la précision de notre modèle. Dans cette fonction, nous avons déjà connu l'intervalle de frames de vol, donc nous avons créé une liste de taille 32 (comme notre sortie) qui prend 0 pour les séquences de frames normales et 1 pour la séquence de vol. Nous avons ensuite comparé cette liste avec celle qui est sortie de notre modèle pour chaque vidéo, et nous avons obtenu une précision.

Enfin, nous avons calculé la moyenne de toutes les vidéos.

3.2.2 Optimisation de nos modèles avec la plateforme OpenVino

Dans cette étape, nous avons utilisé plusieurs techniques et étapes pour atteindre notre objectif. Nous avons utilisé l'architecture OpenVino pour optimiser les modèles et les déployer sur des périphériques Edge, afin de les utiliser en temps réel. Les étapes et les techniques que nous avons utilisées, ainsi que l'architecture d'optimisation, sont les suivantes :

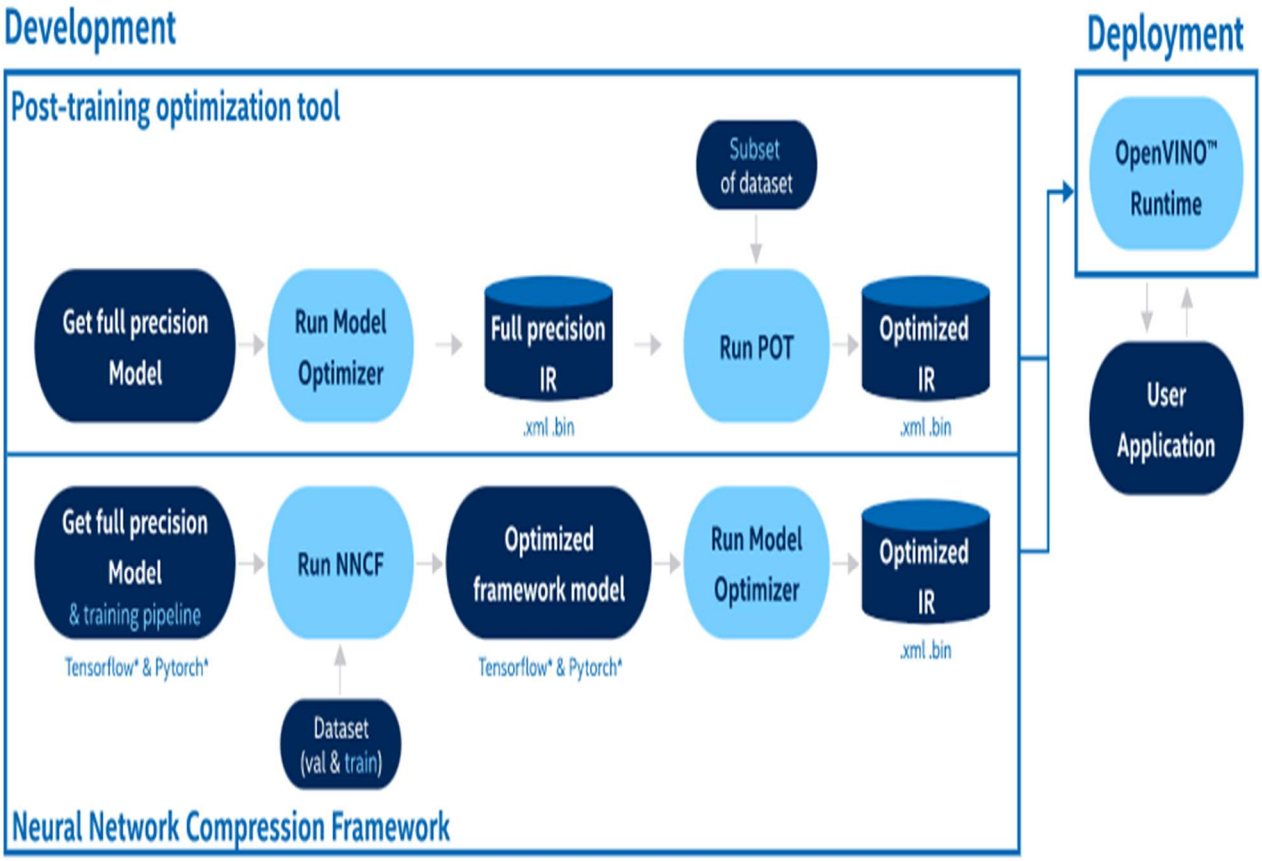

Figure 3.4. L'architecture de développement et de Déploiement de la plateforme OpenVino.

A. **Techniques de conversion et d'optimisation de modèles :** Dans cette partie, nous avons utilisé la plateforme OpenVino pour convertir deux modèles : un modèle pré-entraîné C3D et notre propre modèle de classification. La conversion a été réalisée en utilisant la fonctionnalité Model Optimizer fournie par OpenVino. Cette fonctionnalité permet de convertir des modèles dans divers formats (TensorFlow, Caffe, MXNet, etc.) en modèles compatibles avec l'Intel OpenVino Inference Engine.

Une fois les modèles convertis, nous avons appliqué des techniques d'optimisation pour améliorer leur précision et leur efficacité. La première technique que nous avons utilisée est la quantification de type d'entrée, qui consiste à convertir les données d'entrée de point flottant 32 bits (FP32) en données de point flottant 16 bits (FP16). Cela permet de réduire la quantité de données à traiter, ce qui peut améliorer les performances du modèle.

La deuxième technique que nous avons utilisée est l'élagage (pruning) de modèles. Cette technique consiste à supprimer les poids du modèle qui ont une faible

contribution à la sortie du modèle. Cela permet de réduire la taille du modèle et de réduire le temps d'inférence. Nous avons utilisé la fonctionnalité de pruning fournie par OpenVino pour élaguer les modèles convertis.

B. **Technique de post-entraînement pour améliorer la précision :** Après avoir appliqué les techniques d'optimisation mentionnées ci-dessus, nous avons constaté une légère diminution de la précision des modèles. Pour améliorer la précision, nous avons utilisé une technique de post-entraînement (fine-tuning).

Le post-entraînement est un processus de ré-entraînement d'un modèle sur un ensemble de données supplémentaire. Dans notre cas, nous avons utilisé un ensemble de données plus petit pour ré-entraîner les modèles convertis et élagués. Nous avons utilisé le même processus d'entraînement que celui utilisé pour l'entraînement initial de nos modèles, mais cette fois avec un taux d'apprentissage plus faible et un nombre d'époques plus petit.

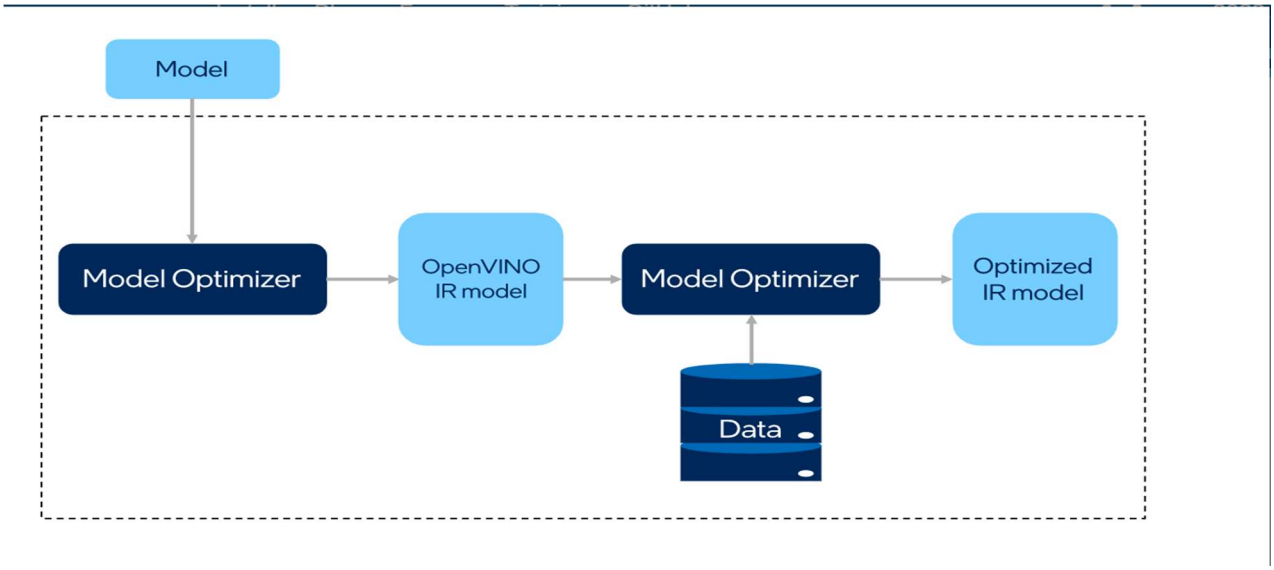

Figure 3.5. L'architecture de post-entraînement utilisé dans la plateforme OpenVino.

En utilisant les techniques d'optimisation mentionnées ci-dessus, nous avons réussi à réduire considérablement la taille des modèles et à améliorer les temps d'inférence tout en maintenant des niveaux de précision comparables à ceux des modèles d'origine. Cependant, après l'application de la quantification et de l'élagage, nous avons constaté une légère diminution de la précision des modèles.

En appliquant la technique de post-entraînement, nous avons réussi à améliorer la précision des modèles sans augmenter leur taille ou leur temps d'inférence. Les modèles post-entraînés

ont atteint des niveaux de précision supérieurs à ceux des modèles d'origine, ce qui confirme l'efficacité de cette technique pour améliorer les performances des modèles.

En conclusion, nous avons décrit dans cette section les techniques que nous avons utilisées pour convertir et optimiser nos modèles, ainsi que la technique de post-entraînement que nous avons utilisée pour améliorer la précision de ces modèles. Grâce à ces techniques, nous avons réussi à réduire la taille des modèles et à améliorer leur temps d'inférence, tout en maintenant des niveaux de précision comparables à ceux des modèles d'origine.

Ces techniques sont particulièrement utiles pour le déploiement de modèles sur des périphériques embarqués, où la taille du modèle et les exigences en matière de calcul peuvent être limitées. De plus, ces techniques peuvent être utilisées pour améliorer les performances des modèles existants sans avoir à ré-entraîner complètement ces modèles.

Dans l'ensemble, ces techniques ont été très efficaces pour améliorer les performances de nos modèles, et pourraient être utiles à d'autres chercheurs souhaitant améliorer les performances de leurs propres modèles.

3.2.3 Déploiement de ce système dans l'Edge

Pour implémenter notre système dans l'Edge, nous avons suivi plusieurs étapes. Tout d'abord, nous avons créé notre code d'inférence en utilisant la plateforme OpenVino. Nous avons commencé par importer les bibliothèques nécessaires, telles que OpenVINO, Matplotlib, OpenCV, NumPy et Paho MQTT. Ensuite, nous avons défini des fonctions pour prétraiter les entrées vidéo, interpoler les caractéristiques et extrapoler les sorties. Nous avons également créé des fonctions pour extraire des images à partir d'un fichier vidéo et créer des clips vidéo à partir de ces images. Ensuite, nous avons chargé les modèles IR (Representational Intermediary) pour le C3D et le modèle de classification. Enfin, nous avons capturé des images à partir de la caméra, écrit les images dans un fichier vidéo et utilisé les modèles pour classifier en temps réel un vidéo de 5 secondes. Les résultats de la classification ont été envoyés à notre MQTT Broker.

Dans la deuxième étape, nous avons créé un tableau de bord (Dashboard) pour notre système en utilisant React JS. Dans ce tableau de bord, nous avons connecté notre système à notre MQTT Broker et nous avons souscrit à une topique spéciale de détection de vol à l'étalage pour recevoir les résultats précédents et les afficher sur notre tableau de bord.

Après avoir testé localement notre système, nous avons finalement déployé notre travail en dockerisant nos modèles via OVMS et notre tableau de bord. La dockerisation de nos modèles nous a permis de les encapsuler dans des conteneurs, ce qui facilite leur déploiement et leur exécution sur différents environnements.

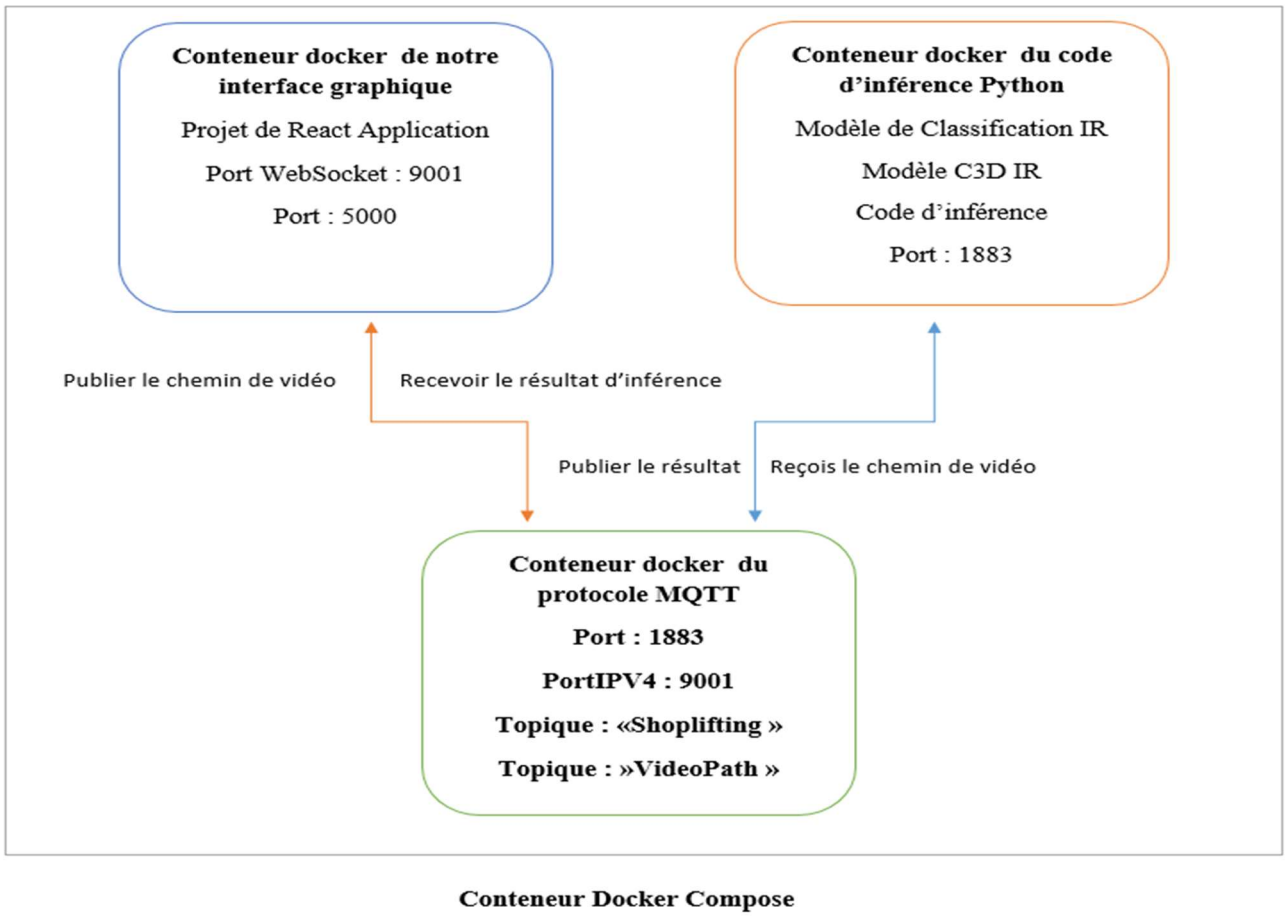

Figure 3.6. L'architecture de notre solution après dockerisation.

Conclusion

En conclusion, ce chapitre a présenté les différentes étapes de développement et d'implémentation d'un système de détection de vol à l'étalage en temps réel, en utilisant des techniques de deep learning et de l'Edge computing. Nous avons décrit les techniques utilisées pour optimiser les modèles de détection, la collecte et l'augmentation des données, ainsi que l'implémentation du système sur des périphériques Edge. Dans le chapitre quatre, nous allons définir comment nous avons implémenté notre solution, présenter et discuter les résultats obtenu.

Chapitre 04 :

Implémentation de notre solution Edge IA pour détecter le vol à l'étalage dans les immeubles commerciaux

Introduction

Dans ce chapitre, nous allons présenter l'implémentation de notre modèle proposé ainsi que les différents outils utilisés. Nous allons également présenter et analyser les résultats obtenus en termes de précision et le temps d'inférence de notre solution Edge IA pour détection de vol à l'étalage dans les immeubles commerciaux.

4.1. Logiciels et bibliothèques utilisés dans l'implémentation

a. Python

Python est un langage de programmation interprété, orienté objet, de haut niveau et dynamiquement sémantique. Il est particulièrement souhaitable pour le développement d'applications rapides ainsi que pour une utilisation en tant que langage de Scripting ou de colle pour lier les composants existants grâce à ses structures de données intégrées de haut niveau, à la saisie dynamique et à la liaison dynamiques. La syntaxe simple de Python priorise la lisibilité et le rend facile à apprendre, ce qui réduit le coût de la maintenance du programme. Le support de Python pour les modules et les paquets favorise la modularité et la réutilisation du code dans les programmes. Pour toutes les plateformes populaires, l'interprète Python et la bibliothèque standard complète sont librement distribuables et disponibles sous forme source ou binaire.

Python provoque fréquemment les programmeurs à tomber amoureux en raison de la productivité accrue qu'il offre. Le cycle d'édition-test-débogage est extrêmement rapide car il n'y a pas d'étape de compilation. Les programmes Python sont simples à déboguer car une défaillance de segmentation n'est jamais causée par un bug ou une entrée incorrecte. Au lieu de cela, l'interprète soulève une exception lorsqu'il trouve une erreur. L'interprète imprime une trace de pile si l'application n'attrape pas l'exception. Définir des points de rupture, évaluer des expressions arbitraires, inspecter des variables locales et globales, passer par le code une ligne à la fois et d'autres fonctionnalités sont toutes possibles avec un débogageur au niveau source. La capacité de Python à effectuer l'introspection est démontrée par le débogage, qui est développé en Python. D'autre part, l'ajout de quelques déclarations

d'impression au code source est souvent le moyen le plus facile de déboguer un programme en raison du court cycle d'édition-test-déboguer [40].

b. TensorFlow

Pour exécuter l'apprentissage automatique, l l'apprentissage Profond et d'autres charges de travail d'analyse statistique et prédictive, les chercheurs de Google ont créé le cadre open source TensorFlow. Pour les utilisateurs tels que les scientifiques des données, les statisticiens et les modélisateurs prédictifs, il est destiné à accélérer le processus de conception et de mise en œuvre d'applications d'analyse sophistiquées.

Les ensembles de données qui sont organisés sous forme de graphique en tant que nœuds de calcul sont gérés par le programme TensorFlow. Les tensors sont produits par des vecteurs multidimensionnels ou des matrices représentées par les bords reliant les nœuds dans un réseau. Parce que les programmes TensorFlow utilisent une architecture de flux de données qui fonctionne avec des résultats de calcul intermédiaires généralisés, ils sont particulièrement accessibles à des applications de traitement parallèle à très grande échelle, les réseaux neuronaux étant un exemple populaire.

Les ensembles d'API de haut niveau et de bas niveau sont tous deux inclus dans le cadre. Afin de simplifier le processus de développement de pipelines de données et de programmation d'applications, Google recommande d'adopter celles de haut niveau partout où cela est possible. Cependant, la société prétend que la compréhension de la façon d'utiliser les API de base de bas niveau TensorFlow peut être utile pour l'expérimentation et la résolution de problèmes d'application. Il fournit également aux utilisateurs un "modèle mental" du fonctionnement interne de la technologie d'apprentissage automatique, selon Google.

Les applications TensorFlow peuvent fonctionner sur les CPU standard et les unités de traitement graphique plus puissantes (GPU), ainsi que sur les unités de traitement de tensor (TPU) de Google, qui sont des matériels spécialisés construits dans le seul but d'accélérer les tâches de Tensor Flow. Les applications internes de la société et les services en ligne, y

compris son algorithme de recherche RankBrain et sa technologie de cartographie Street View, ont été alimentés par les premiers TPU de Google, qui ont été révélés publiquement en 2016 et utilisés en conjonction avec TensorFlow.

En mettant la deuxième génération de TPU à la disposition des clients de Google Cloud Platform pour la formation et l'exécution de leurs propres modèles d'apprentissage automatique au début de 2018, Google a renforcé ses initiatives externes TensorFlow. Les charges de travail basées sur TensorFlow sont facturées par seconde ; selon Google, le service Cloud TPU a d'abord été introduit comme un programme bêta avec des « quantités limitées » des appareils disponibles pour l'utilisation [41].

C. Keras

Basée sur Python, Keras est une collection open source de blocs de construction de réseaux neuronaux. TensorFlow, Theano, PlaidML et d'autres frameworks peuvent tous être utilisés avec Keras. La bibliothèque a d'abord été créée comme un projet de recherche pour le système d'exploitation intelligent neuro-électronique ou ONEIROS, mais elle a été développée plus tard pour être conviviale et flexible. François Chollet, un développeur de Google qui a précédemment créé le modèle de réseau neuronal profond XCeption, est l'auteur principal de Keras. Bien que Keras ait été officiellement introduit, la bibliothèque de base de TensorFlow de Google ne l'a pas incorporé jusqu'en 2017. L'intégration Microsoft Cognitive Toolkit de Keras a également reçu un soutien supplémentaire.

La plate-forme open-source Keras prend en charge les réseaux neuronaux récurrents et convolutifs en plus d'une bibliothèque d'éléments d'apprentissage automatique fréquemment utilisés, y compris les objectifs, les fonctions d'activation et les optimisateurs. Pour les clients souhaitant déployer des modèles d'apprentissage profond sur des smartphones fonctionnant sous iOS et Android, Keras fournit également le développement de plateformes mobiles. Les données de 2018 montrent que 22% des plus de 200.000 utilisateurs de la bibliothèque l'utilisent en fait [42].

D. PyCharm

La programmation Python est réalisée à l'aide de l'environnement de développement intégré (IDE) PyCharm. En plus de soutenir le développement web de Django, il offre une analyse de code, un débogage graphique, un testeur d'unité intégré, une intégration avec les systèmes de contrôle de version, et plus encore. La société JetBrains en République tchèque crée PyCharm [43].

Il est cross-platform et fonctionnel avec Linux, macOS et Microsoft Windows. Il existe une édition communautaire de PyCharm qui est distribuée sous la Licence Apache et une Edition Professionnelle qui est diffusée sous une licence commerciale. Moins de fonctionnalités sont incluses dans PyCharm Community Edition que dans Professional Edition [44].

Configuration utilisée dans l'implémentation

La configuration du matériel utilisé dans notre implémentation est :

- Un Desktop i5 8eme generation CPU
- Carte graphique Intel UHD 620
- RAM de taille 16 GO
- Disque dur SSD de taille 256 GB
- Système d'exploitation Windows 10

4.2. Système de détection de vol à l'étalage à partir d'une vidéo

Ce modèle est basé sur les étapes suivantes :

A. Prétraitement : Voici les étapes pour préparer les vidéos :

1. Importation des bibliothèques nécessaires :

- os : pour travailler avec les fichiers et les chemins d'accès
- c3d : module contenant la définition du modèle C3D
- matplotlib : utilisé pour configurer le backend graphique
- cv2 : bibliothèque OpenCV pour le traitement d'images
- numpy : pour travailler avec des tableaux multidimensionnels

- keras.utils.data_utils : pour télécharger le fichier c3d_mean.npy

Ensuite, nous définissons plusieurs fonctions :

1. Fonction "preprocess_input" :
   - Cette fonction prend une vidéo en entrée et effectue les opérations suivantes :
     - Divise la vidéo en 16 intervalles équidistants (frames) et sélectionne les frames correspondants.
     - Redimensionne chaque frame à la taille 128x171.
     - Soustrait la valeur moyenne (mean) des pixels de chaque frame en utilisant le fichier c3d_mean.npy.
     - Recadre chaque frame à la taille 112x112.
     - Ajoute une dimension supplémentaire pour les échantillons (samples).
   - La fonction renvoie les frames prétraités sous la forme d'un tableau numpy.
2. Fonction "sliding_window" :
   - Cette fonction prend un tableau (arr), une taille de fenêtre (size) et un pas (stride) en entrée.
   - La fonction découpe le tableau en fenêtres glissantes de la taille spécifiée avec le pas spécifié.
   - La fonction renvoie un tableau numpy contenant les fenêtres glissantes.
3. Fonction "get_video_clips" :
   - Cette fonction prend le chemin d'accès à une vidéo (video_path) en entrée.
   - La fonction obtient les frames de la vidéo en utilisant la fonction "get_video_frames".
   - La fonction découpe les frames en clips de 16 frames avec un pas de 16 frames en utilisant la fonction "sliding_window".
   - La fonction renvoie les clips de la vidéo et le nombre total de frames.
4. Fonction "get_video_frames" :
   - Cette fonction prend le chemin d'accès à une vidéo (video_path) en entrée.
   - La fonction ouvre la vidéo à l'aide de OpenCV (cv2.VideoCapture).
   - La fonction extrait les frames de la vidéo et les convertit de BGR à RGB.
   - La fonction renvoie les frames de la vidéo sous forme d'une liste.

Maintenant, nous utilisons ces fonctions pour préparer nos vidéos pour le modèle d'extraction de caractéristiques :

a. Nous lisons tous les chemins des vidéos à partir d'un fichier texte dans une liste.

b. Ensuite, pour chaque vidéo dans la liste, nous utilisons la fonction "get_video_clips" pour découper la vidéo en clips de 16 frames.
c. Chaque clip est converti en un tableau numpy, puis prétraité à l'aide de la fonction "preprocess_input".
d. Les caractéristiques prétraitées sont stockées dans une liste ou un tableau.

Ceci résume l'implémentation des étapes décrites pour préparer les vidéos et extraire les caractéristiques.

```
for jj in range(len(vdlines)):

    video_name = os.path.basename(vdlines[jj]).split('.')[0]
    # read video
    video_clips, num_frames = get_video_clips(vdlines[jj])
    # extract features
    print("nombre de clip est ", len(video_clips))
    rgb_features = []
    start = time.time()
    for i, clip in enumerate(video_clips):
        clip = np.array(clip)
        if len(clip) < frame_count:
            continue

        clip = preprocess_input(clip)
```

Figure 4.1. *Code de boucle de prétraitement de vidéos pour l'apprentissage.*

B. Extraction des caractéristiques : Pour l'extraction des caractéristiques, nous avons utilisé un modèle pré-entraîné appelé C3D. Pour utiliser ce modèle afin d'extraire les caractéristiques, nous avons défini les deux fonctions suivantes :

i. La fonction **C3D(weights='sports1M')** crée un modèle de réseau de neurones convolutionnels 3D (C3D). Le modèle est construit en utilisant la classe **Sequential** de Keras. Il est composé de plusieurs couches de convolution, de sous-échantillonnage et de couches entièrement connectées. Les convolutions 3D sont effectuées pour extraire les caractéristiques spatiales et temporelles des données d'entrée. Le modèle est configuré avec des activations ReLU, des couches de regroupement (pooling) et des

couches de mise à zéro (zero padding) pour le remplissage. En fin de compte, le modèle produit une sortie softmax de dimension 487, correspondant aux classes de l'ensemble de données.

ii. La fonction **c3d_feature_extractor()** crée un modèle d'extraction de caractéristiques basé sur le modèle C3D. Elle utilise la fonction **C3D()** pour obtenir un modèle complet C3D, puis extrait la couche 'fc6' qui contient les caractéristiques les plus profondes et les plus abstraites du modèle. Le modèle d'extraction de caractéristiques est configuré en spécifiant les entrées et les sorties du modèle d'origine. Cela permet d'obtenir une représentation intermédiaire des données d'entrée, qui peut être utilisée dans des tâches telles que la classification ou la recherche de similarité.

Notre fonctionnalité avec ce modèle consiste à prendre chaque clip de l'étape précédente et à les passer à notre modèle c3d_feature_extractor. Au début, nous collectons toutes les caractéristiques vidéo dans une liste. Ensuite, nous utilisons la fonction 'interpolate' qui prend en entrée un ensemble de caractéristiques et les interpole en calculant la moyenne sur des intervalles spécifiques. Les caractéristiques interpolées sont ensuite normalisées avant d'être renvoyées en tant que tableau numpy. Enfin, nous enregistrons les caractéristiques de toutes les vidéos sous forme de fichier texte sur notre PC, afin de passer à l'étape de classification.

Finalement, on a abouti à un modèle qui va résoudre à notre problématique, ce modèle est résumé dans la figure suivante :

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1_input (InputLayer)     [(None, 16, 112, 112, 3)  0
                             ]
conv1 (Conv3D)               (None, 16, 112, 112, 64)  5248
pool1 (MaxPooling3D)         (None, 16, 56, 56, 64)    0
conv2 (Conv3D)               (None, 16, 56, 56, 128)   221312
pool2 (MaxPooling3D)         (None, 8, 28, 28, 128)    0
conv3a (Conv3D)              (None, 8, 28, 28, 256)    884992
conv3b (Conv3D)              (None, 8, 28, 28, 256)    1769728
pool3 (MaxPooling3D)         (None, 4, 14, 14, 256)    0
conv4a (Conv3D)              (None, 4, 14, 14, 512)    3539456
conv4b (Conv3D)              (None, 4, 14, 14, 512)    7078400
pool4 (MaxPooling3D)         (None, 2, 7, 7, 512)      0
conv5a (Conv3D)              (None, 2, 7, 7, 512)      7078400
conv5b (Conv3D)              (None, 2, 7, 7, 512)      7078400
zero_padding3d (ZeroPadding   (None, 2, 9, 9, 512)     0
3D)
pool5 (MaxPooling3D)         (None, 1, 4, 4, 512)      0
flatten (Flatten)            (None, 8192)              0
fc6 (Dense)                  (None, 4096)              33558528
=================================================================
Total params: 61,214,464
Trainable params: 61,214,464
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4.2. Configuration du modèle C3D trouvée.

C. L'apprentissage : Après avoir collecté toutes les caractéristiques de nos vidéos, nous sommes passés à l'étape d'entraînement de notre modèle de classification. Pour réaliser cette dernière, nous avons suivi les étapes suivantes :

Au début, nous avons créé la fonction d'objectif personnalisée (custom_objective). Cette fonction définit l'objectif utilisé pour l'entraînement du modèle. Elle effectue des opérations sur les vraies étiquettes (y_true) et les prédictions du modèle (y_pred) afin de calculer la perte associée.

Ensuite, nous avons défini l'architecture du modèle. Le modèle a été créé en utilisant la classe Sequential de Keras. Il s'agit d'un modèle à couches entièrement connectées (fully connected) avec des couches de neurones denses. Chaque couche a été configurée avec des initialisations

de poids, des régularisations et des fonctions d'activation spécifiques. Ensuite, nous avons compilé le modèle en spécifiant la fonction d'objectif (custom_objective), l'optimiseur (Adagrad) et les métriques à utiliser (précision).

Finalement, on a abouti à un modèle qui va résoudre à notre problématique, ce modèle est résumé dans la figure suivante :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               2097664

dropout (Dropout)            (None, 512)               0

dense_1 (Dense)              (None, 32)                16416

dropout_1 (Dropout)          (None, 32)                0

dense_2 (Dense)              (None, 1)                 33

=================================================================
Total params: 2,114,113
Trainable params: 2,114,113
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4.3. Configuration du modèle de classification trouvée.

4.3. Visualisation et Utilisation

Afin de faciliter l'utilisation de notre système, nous avons conçu deux versions pour interface graphique (web) utilisant React JS. La différence entre nos deux interface est dans l'entrée : dans la première interface en utilisant une cameras pour capturer des vidéos après la sauvegarde de ces vidéos seront traiter en temps réels, et la deuxième version prend des vidéos déjà existant comme entrée puis elle les traitre. Dans nos interfaces nous avons plusieurs états (état initial, état finale, état intermédiaire), ses états sont illustrés par les figures suivantes :

A. **Version 1 :** dans cette version nous avons 3 états suivants :

i. Etat initial : Dans l'état initial nous avons 3 cartes, chaque carte présente une information par exemple la première carte présente le nom de vidéo sélectionné qui prend une valeur initial «No information », la deuxième présente statuts de notre vidéo (ici on a deux valeur « Safe » qui signifie que notre vidéo est normal ,la deuxième valeur «Alerte » qui signifie que dans notre vidéos nous avons une vol à l'étalage).la figure suivante présente l'état initial de cette version :

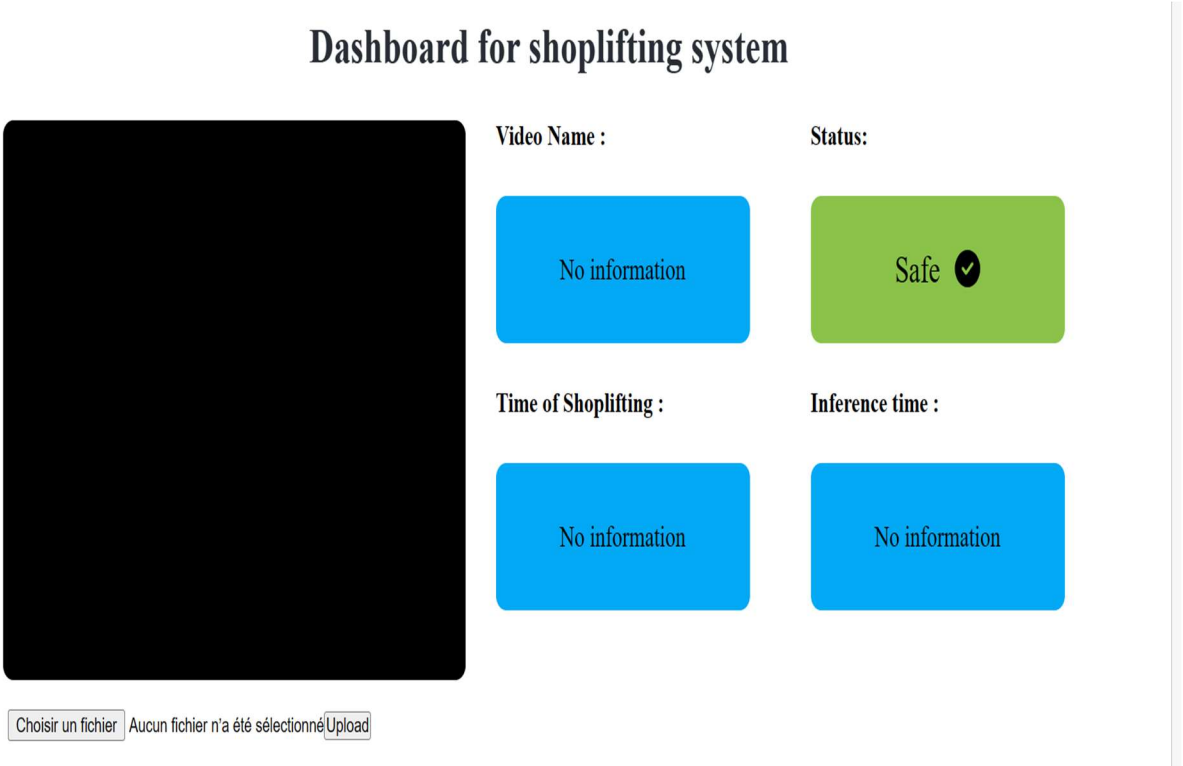

Figure 4.4. Vue générale sur l'interface graphique version 1 du système.

ii. Etat intermédiaire : le commencement de cet état est ensuite la sélection d'une vidéo, dans cet état notre interface publie la source de vidéo (path) dans une topique de notre broker, notre projet python reçois ce message ensuite il traite la vidéo sélectionné. Cet état est illustré dans la figure suivante :

Figure 4.5. Vue sur l'interface graphique version 1 du système après la sélection d'une vidéo.

iii. Etat final : dans l'état final notre interface affiche les informations et résultat de traitement par MQTT broker, ces infos sont le nom de vidéo et le statut de cette vidéo («safe» ou «alerte») et le temps déroulement de vol dans la vidéo comme elle est illustré dans les figures suivantes :

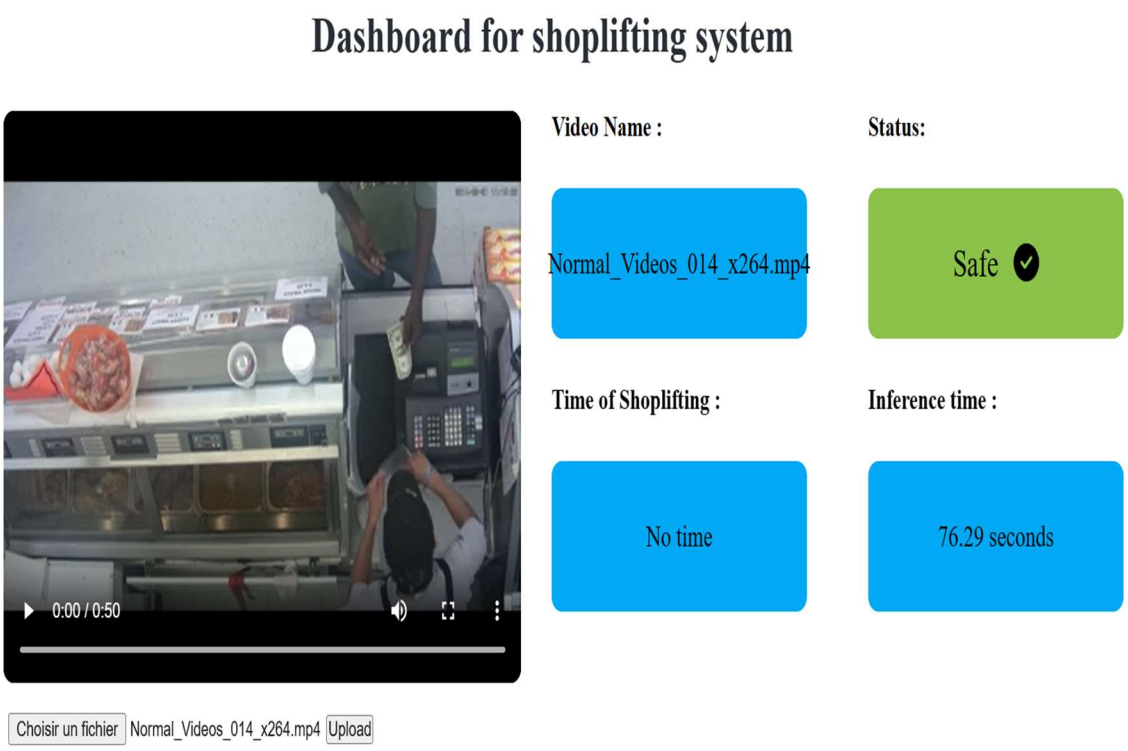

Figure 4.6. Vue sur l'interface graphique version 1 du système après le traitement de la vidéo normal.

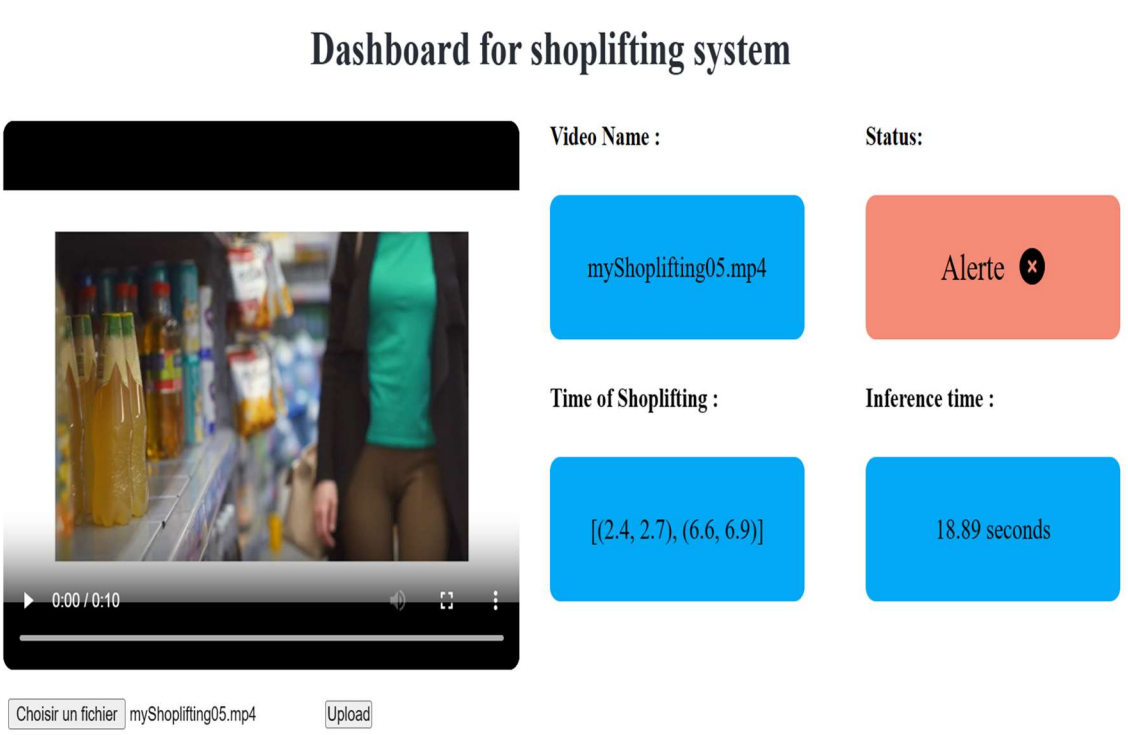

Figure 4.7. Vue sur l'interface graphique version 1 du système après le traitement de la vidéo de vol.

B. **Version 2 :** dans cette version aussi nous avons 3 états suivants :

i. Etat initial : Dans l'état initial nous avons 3 cartes, chaque carte présente une information par exemple la première carte présente le nom de la vidéo traitée qui prend une valeur initial «No information », la deuxième présente statut de notre vidéo (ici on a deux valeur « Safe » qui signifie que notre vidéo est normal ,la deuxième valeur «Alerte » qui signifie que dans notre vidéos nous avons un vol à l'étalage).la figure suivante présente l'état initial de cette version :

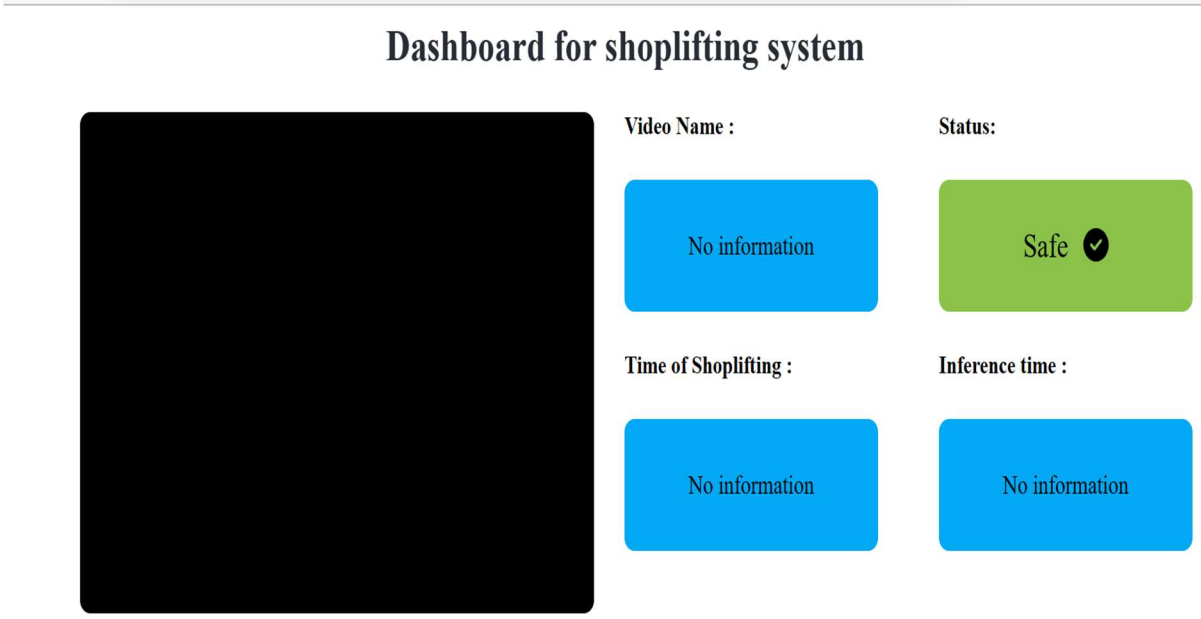

Figure 4.8. Vue générale sur l'interface graphique version 2 du système.

ii. Etat intermédiaire : le commencement de cet état est ensuite la capture et la sauvegarde de la vidéo (cette version capture et sauvegarde les vidéos automatique chaque 5 seconds), dans cet état notre interface publie la source de vidéo (path) dans une topique de notre broker, notre projet python reçoit ce messages et traite la vidéo sauvegardé. Cet état est illustré dans la figure suivante : (on ne peut pas obtenir une capture lors la camera fonctionne à cause de les lois de JavaScript pour sécuriser les utilisateurs)

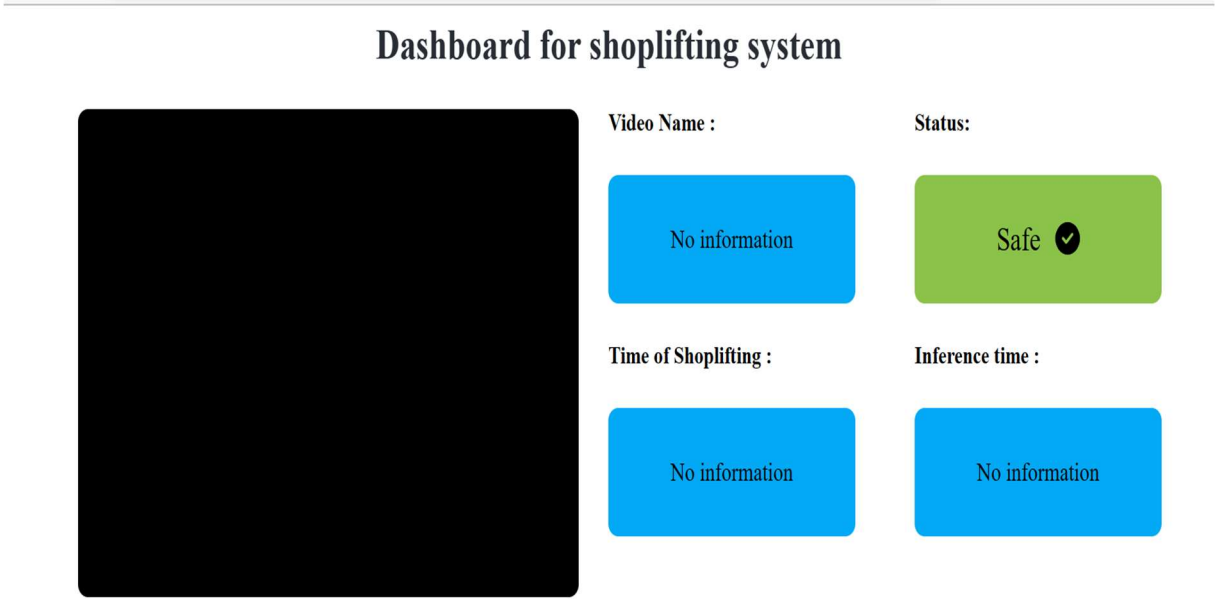

Figure 4.9. Vue générale sur l'interface graphique version 2 du système en cours de traitement.

iii. Etat final : dans l'état final notre interface affiche les informations et résultat de traitement par MQTT broker, ces infos sont le nom de vidéo et statuts de cette vidéo («safe» ou «alerte») et le temps de déroulement de vol dans la vidéo comme elle est illustré dans les figures suivantes : (on ne peut pas obtenir une capture lors la camera fonctionne à cause de les lois de JavaScript pour sécuriser les utilisateurs, de plus nous n'avons pas d'un cas de vol à partir de cette version)

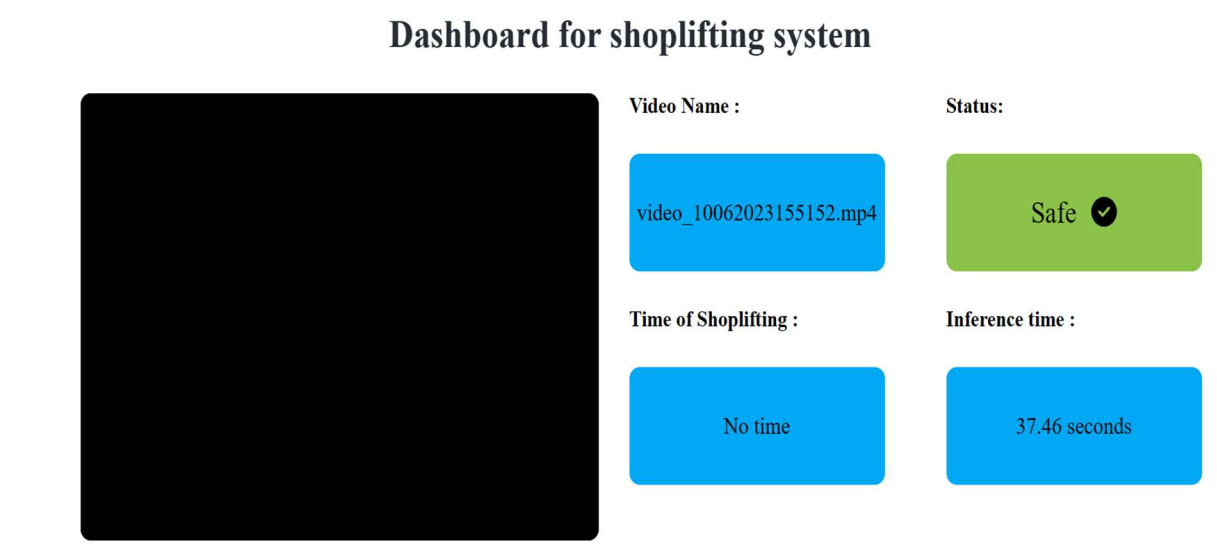

Figure 4.10. Vue générale sur l'interface graphique version 2 du système après recevez les résultats.

4.4. Résultats

Dans notre projet, on a créé deux versions de notre solution, un modèle initial qui a la possibilité de détecter les vols à l'étalage à partir d'une vidéos, et un autre modèle optimisé qui a généralement les mêmes possibilité de détecter les vols à l'étalage à partir d'une vidéos par rapport à notre modèle sans optimisation, par la suite, on a effectué une comparaison entre ce deux version et avec les autres modèles de travaux connexes . Les détails sur les résultats sont :

4.1 Modèle sans optimisation :

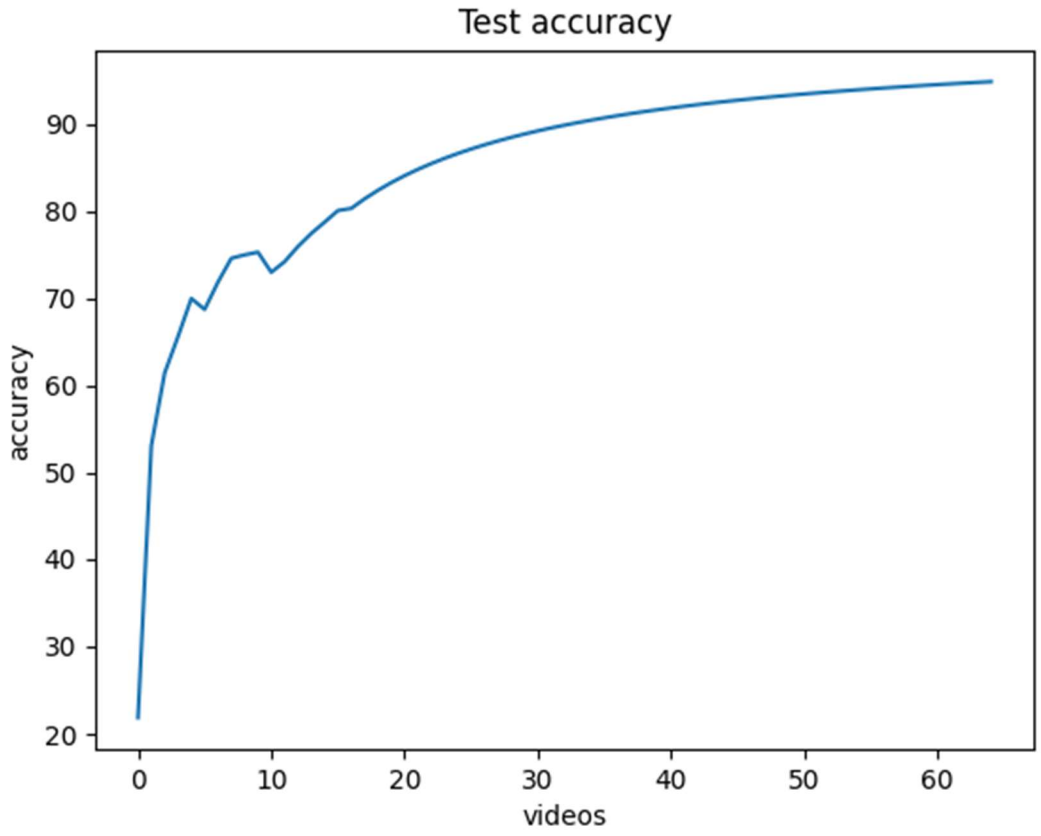

Figure 4.11. Graphe de l'Accuracy.

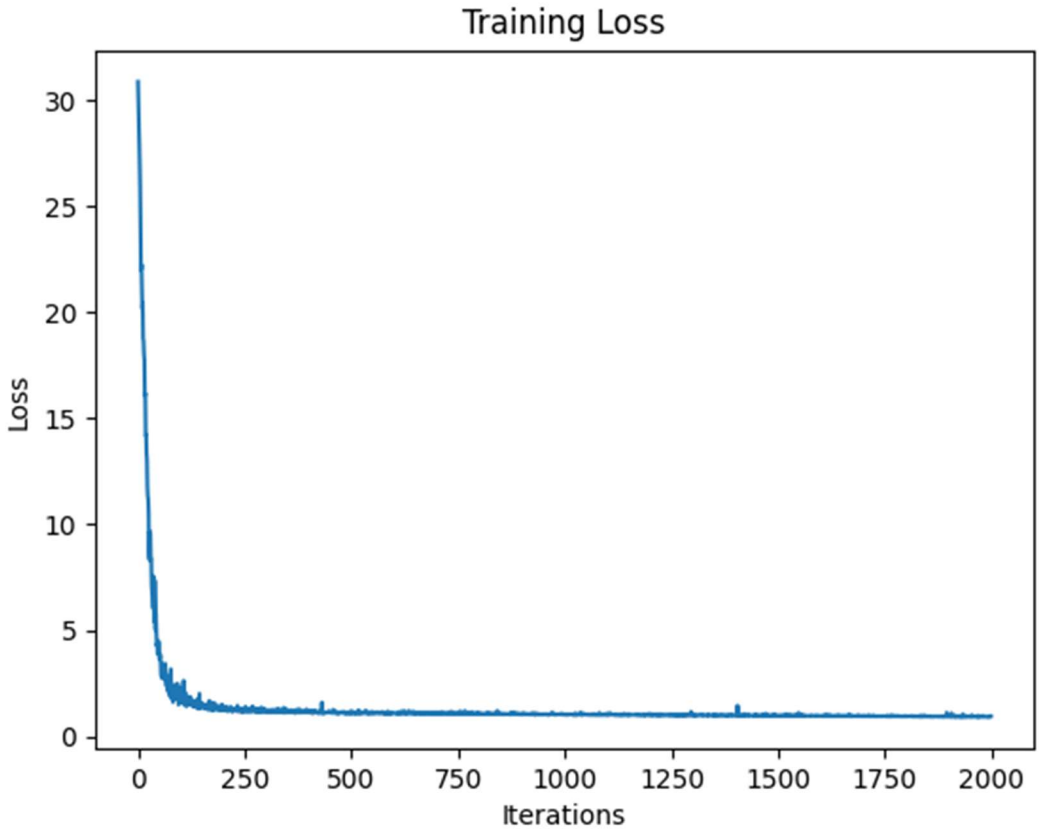

Figure 4.12. Graphe de la perte.

4.2 Modèle avec optimisation :

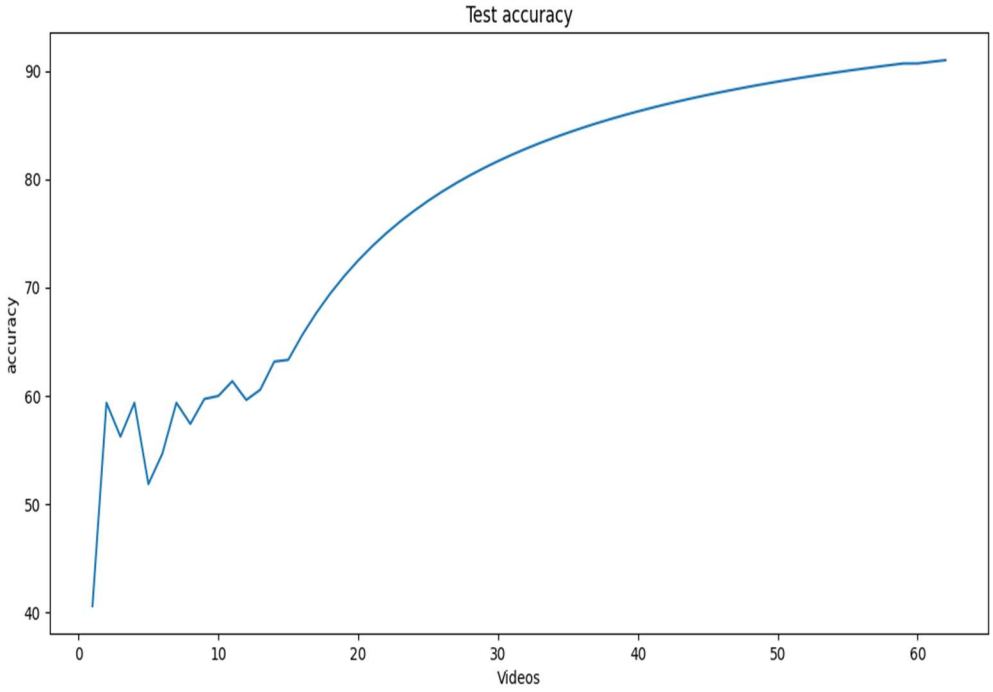

Figure 4.13. Graphe de l'Accuracy.

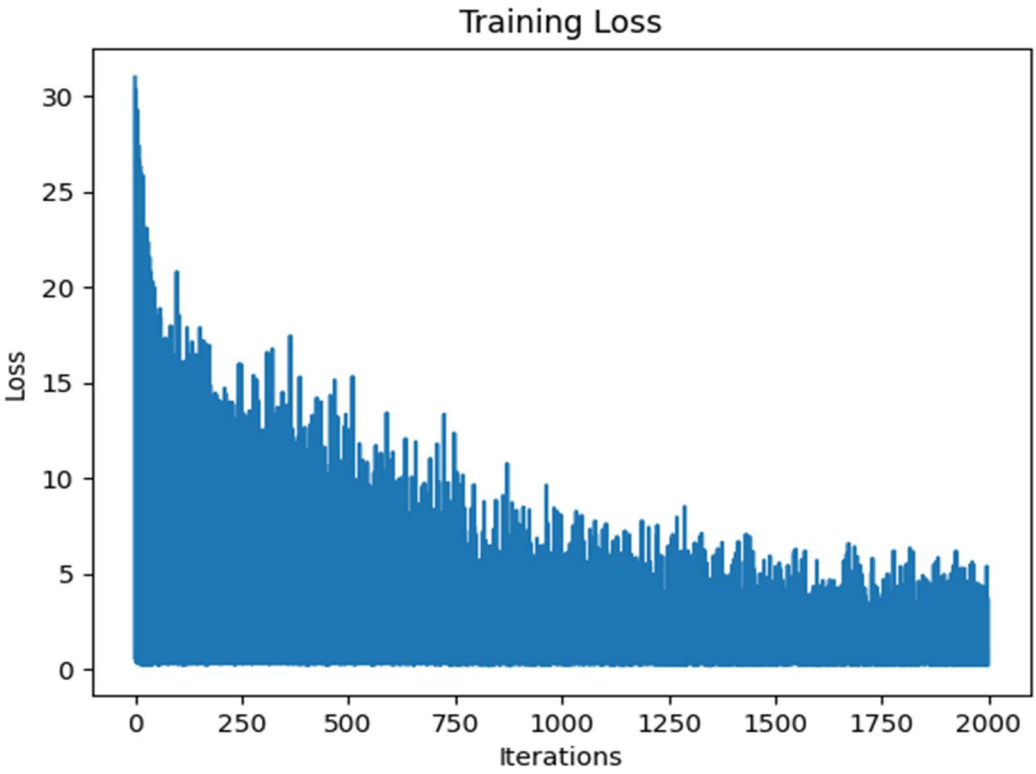

Figure 4.14. Graphe de la perte.

4.3 La différence entre le modèle normal et le modèle optimisé :

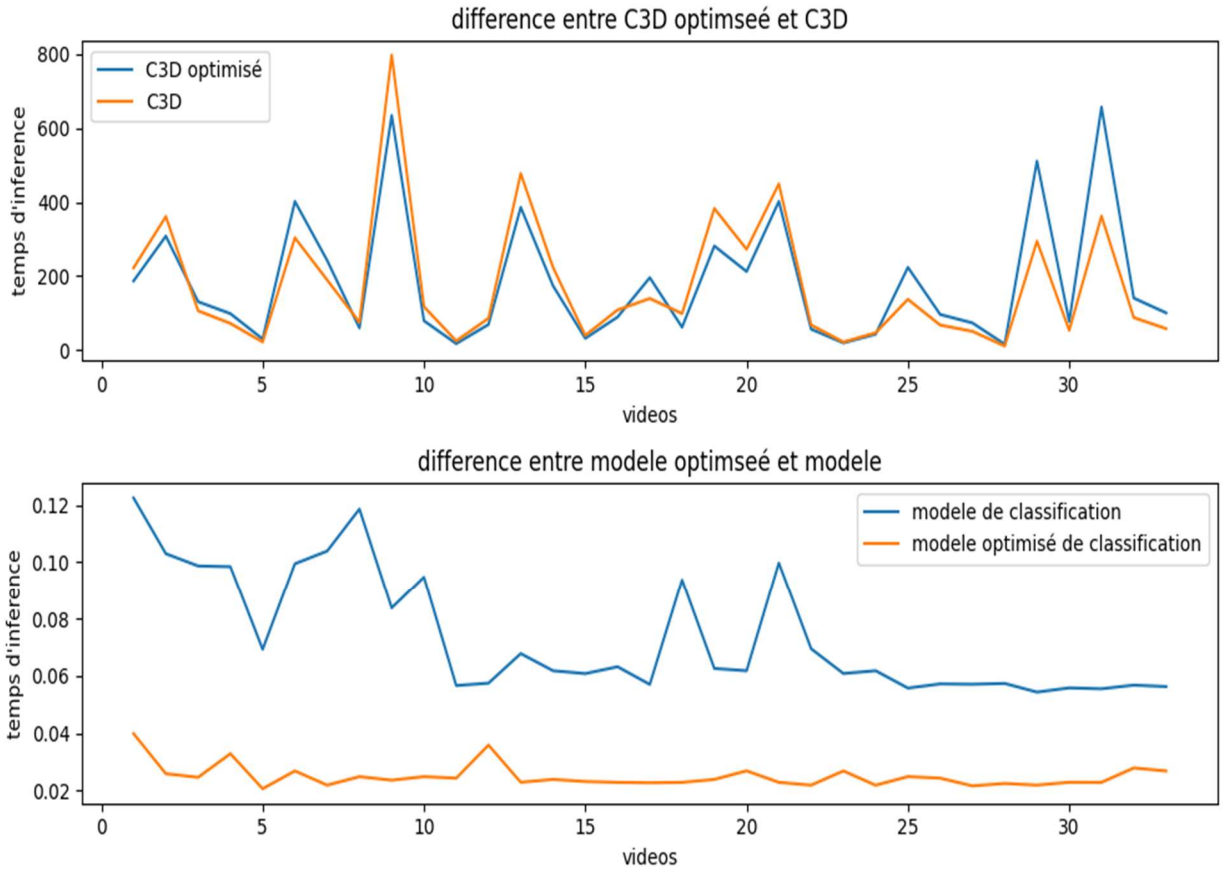

Figure 4.15. Les graphes de différence entre modèle et modèle optimisé en temps d'inférence.

4.5. Discussion et comparaisons

Après avoir entrainé et testé nos modelés, nous sommes sortis avec les résultats représentés dans la figure suivante :

Base de données	Modèle	Perte	Meilleure Précision	Temps d'inférence	Taille de fichier
UFC-Crime	C3D + Notre modèle	1.20	94.85%	0.074 SC (ANN) 1.9 SC pour traiter une seconde (C3D)	305 mo + 16 mo
UFC-Crime	**C3D optimisé + Notre modèle optimisé**	**4.02**	**91.47%**	**0.025 SC (ANN) 1.5 SC pour traiter une seconde (C3D)**	**116 mo + 4 mo**
UFC-Crime	C3D + GRU [29]	—	93%	—	—
SBT-balanced	3D CNN [31]	—	92.5%	—	—

Figure 4.16 : Tableau de la comparaison de nos résultats avec deux travaux similaires**.**

Dans notre étude, nous avons montré la supériorité du notre solution optimisé pour la détection de vol à l'étalage qui offre des performances acceptable avec une faible complexité et de temps d'inférence réduit par rapport les autres solutions, notre modèle est non seulement robuste dans les ensemble d'entrainement, mais aussi dans les petits ensembles de test que nous avons fait, Il nous a fourni une bonne précision comparable à celle des autres solutions.

Conclusion

Dans ce chapitre, nous avons présenté les étapes les plus importantes que nous avons faites pour mettre en œuvre notre solution proposée. Nous avons exposé la mise en œuvre de notre solution proposée où nous avons réussi à améliorer une architecture (C3D-ANN) donnant une précision de 91%. Au final, nous avons mis en place deux interfaces graphiques où nous avons déployé notre solution.

Conclusion générale

La détection automatique de vols à l'étalage peut offrir de nombreux avantages, notamment la réduction des pertes économiques pour les propriétaires des magasins, la dissuasion des voleurs potentiels, l'amélioration de la sécurité du personnel, l'augmentation de la confiance des clients. En mettant en place un tel système, les détaillants peuvent améliorer la sécurité globale de leur établissement, réduire les incidents de vol à l'étalage et créer un environnement plus sûr et plus confiant pour tous les acteurs impliqués.

Dans ce projet, nous avons étudié et développé, une solution Edge IA capable de détecter de vol à l'étalage d'à partir d'un camera ou des vidéos déployé d'un l'Edge et pour faciliter l'utilisation de notre solution, nous avons créé une interface graphique simple accessible via le web. Cette interface permet aux utilisateurs de visualiser les résultats de la détection en temps réel.

Nous avons concentré notre étude, en premier sur l'analyse, le nettoyage et l'augmentation des vidéos, si qui nous permet de performer la précision de notre modèle.

Le système réalisé, sur la base de détection de vol à l'étalage dans les immeubles commerciaux s'est illustrée par le développement de deux versions de nos modèles (C3D et ANN, C3D optimisé et ANN optimisé), à la recherche de meilleurs pour le cas étudié.

Les résultats obtenus, suivant l'utilisation de deuxième version de nos modèles (version optimisé) C3D-ANN on a eu une précision qui dépasse 91%, Nous pouvons conclure que notre système pourrait améliorer la sécurité de les immeubles commerciaux.

Ensuite, nous avons implémenté ce travail à l'Edge IA en passant par la dockerisation de notre solution. Après avoir testé notre solution, nous avons constaté qu'elle était efficace et offrait plusieurs avantages réels aux utilisateurs. Dans le futur, nous avons la capacité d'améliorer encore notre solution grâce à l'obtention de nouvelles vidéos de vols réels.

Ce travail nous a permis d'acquérir une immense quantité de connaissances dans le domaine de la vision par ordinateur, du machine learning, du deep learning, du cloud, de l'Edge IA, de l'augmentation de données et du déploiement des travaux à l'Edge. Il nous a également permis de mettre en pratique nos connaissances théoriques dans le monde réel. Le temps que nous avons passé à lire des articles et à étudier du code nous a aidés à nous initier à la recherche.

Bibliographie

[1] Mary Owen Cameron, The Booster and the Snitch : Department Store Shoplifting, Glencoe, Illinois, Free Press of Glencoe, 1964 (ASIN B002NGZUJU, lire en ligne [archive])

[2] Article 17 de la loi n° 2011-267 du 14 mars 2011 d'orientation et de programmation pour la performance de la sécurité intérieure [archive], sur Légifrance.

[3] ↑ « Quand la "vidéoprotection" remplace la "vidéosurveillance" » [archive], Le Monde, 16 février 2010 (consulté le 9 décembre 2020)

[4] Lakhdar Imlouli, GUIDE DE LA VIDÉOSURVEILLANCE : tout ce qu'il faut savoir sur les différentes technologies de la vidéosurveillance, Suresnes, Les Éditions du Net, 2014, 152 p. (ISBN 978-2-312-02810-1)

[5] Lopes, A. (2018). Video processing: from frame to pixel. Springer.

[6] https://www.postscapes.com/smart-greenhouses/ consulté : Mars 2021.

[7] Han, M., & Zhang, H. (2013). Business intelligence architecture based on internet of things. Journal of Theoretical & Applied Information Technology, 50(1),90-95.

[8] Dana H. Ballard; Christopher M. Brown (1982). Computer Vision. Prentice Hall. ISBN 978-0-13-165316-0.

[9] ^ Huang, T. (1996-11-19). Vandoni, Carlo, E (ed.). Computer Vision : Evolution And Promise (PDF). 19th CERN School of Computing. Geneva: CERN. pp. 21–25. doi:10.5170/CERN-1996-008.21. ISBN 978-9290830955. Archived (PDF) from the original on 2018-02-07.

[10] ^ Milan Sonka; Vaclav Hlavac; Roger Boyle (2008). Image Processing, Analysis, and Machine Vision. Thomson. ISBN 978-0-495-08252-1.

[11] ^ http://www.bmva.org/visionoverview Archived 2017-02-16 at the Wayback Machine The British Machine Vision Association and Society for Pattern Recognition Retrieved February 20, 2017

[12] Szeliski, R. (2011). Computer vision: algorithms and applications. Springer Science & Business Media.

[13] Office québécois de la langue française, « informatique en périphérie » [archive], sur Le grand dictionnaire terminologique (consulté le 31 janvier 2019)

[14] ↑ Mohamed Medhat Gaber, Frederic Stahl et Joao Bártolo Gomes, Pocket Data Mining : Big Data on Small Devices, Springer International Publishing, 2014, 1re éd., 108 p. (ISBN 978-3-319-02710-4).

[15] ↑ Ahmad Tohami, « Kubernetes à la mode Edge » [archive], sur https://www.nexworld.fr/ [archive]

[16] Jie, H., Lu, Y., Li, S., & Li, W. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8), 1738-1762.

[17] François Blayo et Michel Verleysen, Les réseaux de neurones artificiels, PUF, Que Sais-je No 3042 [archive], 1re éd., 1996

[18] Léon Personnaz et Isabelle Rivals, Réseaux de neurones formels pour la modélisation, la commande et la classification, CNRS Éditions, 2003.

[19] Richard P. Lippman, « An Introduction to Computing with Neural Nets », IEEE ASSP Magazine, avril 1987, p. 4-22

[20] Neural Networks : biological comput

[21] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[22] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[23] Site Web officiel d'Intel OpenVINO: https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

[24] Chen, L., Zhu, H., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 801-818).

[25] Zhang, S., Wen, W., Fardad, M., & Li, Y. (2018). Efficient and accurate approximations of nonlinear convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1984-1992).

[26] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 1-48.

[27] https://www.crcv.ucf.edu/projects/real-world/

[28] https://aws.amazon.com/fr/what-is/mqtt/#:~:text=MQTT%20is%20a%20standards%2Dbased,constrained%20network%20with%20limited%20bandwidth.

[29] Computation 2022, 10(11), 199; https://doi.org/10.3390/computation10110199

[30] Gim, U.-J., Lee, J.-J., Kim, J.-H., Park, Y.-H., & Nasridinov, A. (2020). An Automatic Shoplifting Detection from Surveillance Videos (Student Abstract). Proceedings of the AAAI Conference on Artificial Intelligence, 34(10), 13795-13796. https://doi.org/10.1609/aaai.v34i10.7169

[31] Computation 2021, 9(2), 24 ; arXiv:2005.02142 [cs.CV]

[32] Rehman, A.; Belhaouari, S.B. Deep Learning for Video Classification: A Review. TechRxiv 2021, preprint.

[33] Flores-Munguia, C.; Ortiz-Bayliss, J.C.; Terashima-Marin, H. Leveraging a Neuroevolutionary Approach for Classifying Violent Behavior in Video. Comput. Intell. Neurosci. 2022, 2022, 1279945. [CrossRef] [PubMed]

[34] Morales, G.; Salazar-Reque, I.; Telles, J.; Diaz, D. Detecting violent robberies in cctv videos using deep learning, IFIP advances in information and communication technology. In

Artificial Intelligence Applications and Innovations; Springer International Publishing: Cham, Switzerland, 2019; pp. 282–291.

[35] Nasaruddin, N.; Muchtar, K.; Afdhal, A.; Dwiyantoro, A.P.J. Deep anomaly detection through visual attention in surveillance videos. Big Data 2020, 7, 87. [CrossRef]

[36] Sultani, W.; Chen, C.; Shah, M. Real-world anomaly detection in surveillance videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6479–6488.

[37] Li, J.; Jiang, X.; Sun, T.; Xu, K. Efficient Violence Detection Using 3D Convolutional Neural Networks. In Proceedings of the 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 18–21 September 2019; pp. 1–8.

[38] Islam, M.S.; Sultana, S.; Kumar Roy, U.; Al Mahmud, J. A review on video classification with methods, findings, performance, challenges, limitations and future work. J. Ilm. Tek. Elektro Komput. Dan Inform. (JITEKI) 2020, 6, 47–57. [CrossRef]

[39] Alfaifi, R.; Artoli, A.M. Human action prediction with 3D-CNN. SN Comput. Sci. 2020, 1, 286. [CrossRef]

[40] https://www.python.org/doc/essays/blurb/

[41] https://www.techtarget.com/searchdatamanagement/definition/TensorFlow

[42] https://deepai.org/machine-learning-glossary-and-terms/keras

[43] "JetBrains Strikes Python Developers with PyCharm 1.0 IDE". eWeek. Archived from the original on January 22, 2013.

[44] PyCharm 3.0 community edition source code now available Jet Brains. October 2013.