

École doctorale : Cognition, Langage, Interaction

THÈSE

pour obtenir le grade de docteur délivré par

**Université Paris 8 Vincennes à Saint-Denis**

Spécialité doctorale *Informatique*

*présentée et soutenue publiquement par*

**Fabien POIRIER**

le 14/09/2023

**Détection d'anomalies en temps réel dans un flux vidéo**

Directeur de thèse : Professeur Gilles **BERNARD**, U. Paris 8, Paragraphe

Co-directeur de thèse : MCF Rakia **JAZIRI**, U. Paris 8, Paragraphe

**Jury**

M. Pierre <b>GANÇARSKI</b> ,	Professeur	U. Strasbourg	ICube	Président
M. Kurosh <b>MADANI</b> ,	Professeur	U. Paris Est Créteil	LISSI	Rapporteur
M. Mustapha <b>LEBBAH</b> ,	Professeur	U. Paris Saclay	David	Rapporteur
M. Haythem <b>ELGHAZEL</b> ,	MCF	U. Lyon 1	LIRIS	Examinateur
Mme Rakia <b>JAZIRI</b> ,	MCF	U. Paris 8	Paragraphe	Co-directrice
M. Gilles <b>BERNARD</b> ,	Professeur	U. Paris 8	Paragraphe	Directeur

## **Remerciements**

Je tiens à remercier toutes les personnes ayant contribué au succès de cette thèse en commençant par mes encadrants madame Rakia Jaziri et monsieur Gilles Bernard.

Mon entreprise Othello et plus particulièrement son dirigeant Camille Srour pour sa confiance et pour avoir fait en sorte que cette thèse se déroule dans les meilleures conditions possibles en me fournissant tous les outils nécessaires.

De plus, je souhaite exprimer ma sincère reconnaissance envers les relecteurs et les membres du jury pour leurs précieux conseils et leurs contributions qui ont grandement amélioré ce travail.

Pour finir, je souhaite aussi remercier ma famille et mes amis qui m'ont soutenu pendant ces 4 années.

## Résumé

**Mots-clés :** détection d'anomalies, vidéo, temps réel, reconnaissance d'actions, détection d'objets, deep learning, réseaux à convolution, réseaux récurrents, réseaux de neurones.

Cette thèse s'inscrit dans une convention CIFRE entre l'entreprise Othello et le laboratoire **LIASD**. L'objectif est un système d'intelligence artificielle détectant en temps réel des dangers dans un flux vidéo. Pour y parvenir, une nouvelle approche combinant analyse temporelle et spatiale a été proposée.

Plusieurs pistes ont été explorées pour améliorer la détection d'anomalie, en y intégrant la détection d'objets, de la pose humaine et du mouvement. Pour l'explicabilité des résultats, des techniques propres aux images comme les cartes d'activation et de saillance ont été étendues aux vidéos, et une méthode originale a été proposée.

L'architecture proposée réalise une classification binaire ou non suivant qu'on veut alerter ou donner la cause. De nombreux modèles de réseaux neuronaux ont été testés, et trois d'entre eux ont été retenus. **You Only Looks Once (YOLO)** a été employé pour l'analyse spatiale, un **Convolutional Recurrent Neuronal Network (CRNN)** composé de **VGG19** et d'un **Gated Recurrent Unit (GRU)** pour l'analyse temporelle, et un perceptron multi-couche pour la classification. Ces modèles traitent des données de types différents et peuvent être combinés en parallèle ou en série. Bien que le mode parallèle soit plus rapide, le mode série est généralement plus fiable.

Pour entraîner ces modèles, l'apprentissage supervisé a été choisi, et deux jeux de données propriétaires ont été créés. Le premier est consacré à des objets ayant un rôle potentiel dans les anomalies et le second est constitué de vidéos contenant des anomalies ou non. Cette approche permet de traiter à la fois des flux vidéo continus et des vidéos finies, d'où une meilleure flexibilité dans la détection.

## Abstract

**Keywords :** anomaly detection, video, real-time, action recognition, object detection, deep learning, convolutional networks, recurrent networks, neural networks.

This thesis is part of a CIFRE agreement between the company Othello and the **LIASD** laboratory. The objective is to develop an artificial intelligence system that can detect real-time dangers in a video stream. To achieve this, a novel approach combining temporal and spatial analysis has been proposed.

Several avenues have been explored to improve anomaly detection by integrating object detection, human pose detection, and motion analysis. For result interpretability, techniques commonly used for image analysis, such as activation and saliency maps, have been extended to videos, and an original method has been proposed.

The proposed architecture performs binary or multiclass classification depending on whether an alert or the cause needs to be identified. Numerous neural network models have been tested, and three of them have been selected. **You Only Looks Once (YOLO)** has been used for spatial analysis, a **Convolutional Recurrent Neuronal Network (CRNN)** composed of **VGG19** and a **Gated Recurrent Unit (GRU)** for temporal analysis, and a multi-layer perceptron for classification. These models handle different types of data and can be combined in parallel or in series. Although the parallel mode is faster, the serial mode is generally more reliable.

For training these models, supervised learning was chosen, and two proprietary datasets were created. The first dataset focuses on objects that may play a potential role in anomalies, while the second consists of videos containing anomalies or non-anomalies. This approach allows for the processing of both continuous video streams and finite videos, providing greater flexibility in detection.

# Sommaire

<b>Introduction</b>	<b>7</b>
<b>Acronymes</b>	<b>10</b>
<b>I Problématique et état de l'art</b>	<b>13</b>
<b>1 Problématique</b>	<b>17</b>
<b>2 État de l'art</b>	<b>35</b>
<b>II Système réalisé</b>	<b>65</b>
<b>3 Système</b>	<b>69</b>
<b>4 Expérimentations et résultats</b>	<b>85</b>
<b>Conclusion</b>	<b>141</b>
<b>Glossaire</b>	<b>146</b>
<b>Mes publications</b>	<b>149</b>
<b>Bibliographie</b>	<b>151</b>
<b>Liste des tableaux</b>	<b>161</b>
<b>Table des figures</b>	<b>165</b>

<b>Table des matières</b>	<b>169</b>
---------------------------	------------

# Introduction

Les appareils visant à assurer notre sécurité (caméra, micro, drone, etc...) sont de plus en plus répandus. Cependant, nous ne sommes toujours pas capables d'apporter une protection et une aide efficace aux citoyens. Ces appareils ayant pour but de garantir notre sécurité sont, dans la plupart des cas, utilisés simplement comme outil de dissuasion. Actuellement, la télésurveillance permet à un opérateur physique de contrôler à distance plusieurs lieux simultanément grâce à un système de caméras, disposé dans un espace public ou privé. Les images obtenues par ces caméras sont transmises sur une batterie d'écrans pour être visionnées et analysées, puis archivées ou détruites. Cette surveillance a pour but de contrôler les conditions de sécurité et de sûreté de ces lieux. Généralement, ces images sont analysées par des personnes physiques. Cette surveillance est donc une tâche longue et coûteuse; de plus, l'efficacité d'un tel système dépend de l'attention et de la réactivité du surveillant.

Dans les années à venir, de plus en plus de lieux seront équipés de ces outils. Ce qui aura pour conséquence d'alourdir la charge de travail des surveillants déjà en place ainsi qu'accroître la main d'œuvre nécessaire.

Avec l'avancée de l'intelligence artificielle dans de nombreux domaines comme celui de la reconnaissance faciale, la reconnaissance d'actions ou encore la détection et le suivi d'objet, il serait plus efficace d'automatiser au moins partiellement cette analyse, dans le but d'aider le surveillant dans sa tâche voire de le remplacer dans certains cas.

Ce travail est un projet de recherche effectué dans le cadre d'une convention CIFRE entre l'entreprise Othello<sup>1</sup> et le **Laboratoire d'Intelligence Artificielle et Science des Données (LIASD)**<sup>2</sup>. Il a pour but de développer des méthodes d'intelligence artificielle pour la détection d'anomalies en temps réel dans des flux de données vidéo et de réduire le temps d'intervention.

Les contributions de mon travail sont les suivantes :

- 
1. <http://www.othello.group>
  2. [Page du LIASD](#)

- Une problématique qui s'appuie sur l'idée de combiner l'analyse temporelle des séquences et l'analyse spatiale des images pour améliorer la détection d'anomalies, avec la contrainte de décider très rapidement des alertes à transmettre.
- Deux jeux de données, un d'images et un de vidéos ; ces vidéos représentent des anomalies ayant des répercussions par rapport à notre besoin industriel, et les images représentent des objets ayant une influence potentielle dans ces anomalies. Dans le cas de ce contrat CIFRE, ces jeux de données sont propriétaires.
- Un état de l'art incluant tous les aspects potentiellement utiles à cet objectif : analyse temporelle des séquences, analyse spatiale des images (détection d'objets, de pose, de mouvements), classification des anomalies, traçabilité des décisions.
- Un ensemble de tests sur une partie de cet état de l'art, concernant tout particulièrement les algorithmes de référence en analyse spatiale et temporelle de flux vidéo, incluant également différents vidéo-générateurs (Poirier et al., 2022).
- La proposition de méthodes originales, pour la traçabilité des décisions avec la méthode des contours (Poirier et al., 2023b), pour l'évaluation des résultats (méthode des badBox) ; ainsi que la participation à l'élaboration du programme de vidéo-génération que nous avons finalement choisi.
- Une architecture combinant l'analyse d'images dans le but de détecter des objets ou des poses humaines potentiellement suspect.e.s, du traitement des séries temporelles afin de surveiller les actions de ces objets, et de la classification pour déclencher des alertes, dans un temps raisonnable (Poirier et al., 2023a).
- La proposition de plusieurs modes, au niveau de l'articulation analyse spatiale et temporelle (mode série, mode parallèle), pour la traçabilité (mode traçage), et dans la classification (modes multi-classe et binaire à la fois par type d'anomalie et global).
- De nombreux tests de ces modes pour déterminer lesquels conviennent mieux à telle ou telle utilisation de notre programme.

## Guide de lecture

Ce mémoire se découpe en deux parties :

- La partie **I** a pour objet d'introduire le contexte de l'étude, de présenter la problématique, ainsi que de fournir une revue des travaux en matière de détection d'anomalies dans des données vidéos et de détection d'objets basés sur des techniques d'intelligence artificielle.
- La partie **II** se focalise sur le développement du système, en détaillant les contributions, les expérimentations réalisées et les résultats obtenus.

Ce mémoire se termine par une conclusion synthétisant mon travail, faisant un état de ses limitations et proposant des perspectives qui seraient intéressantes d'envisager à l'avenir.



# Acronymes

**C3D** Convolutional 3 Dimensions.

**CGRU** Convolutional Gated Recurrent Unit.

**CNIL** Commission Nationale de l'Informatique et des Libertés.

**CNN** Convolutional Neuronal Network.

**CRNN** Convolutional Recurrent Neuronal Network.

**FPS** Frame Per Second.

**GAN** Generative Adversarial Networks.

**GRU** Gated Recurrent Unit.

**KNN** K-Nearest Neighbours.

**LIASD** Laboratoire d'Intelligence Artificielle et Science des Données.

**LSTM** Long Short Term Memory.

**MLP** Multi Layer Perceptron.

**RCNN** Regional Convolutional Neuronal Network.

**RGPD** Règlement Général sur la Protection des Données.

**RNN** Recurrent Neronal Network.

**RoI** Region of Interest.

**SAM** Segment Anything Mode.

**SVM** Support Vector Machine.

**TCN** Temporal Convolutional Network.

**VGG** Visual Geometry Group.

**ViT** Vision Transformer.

**YOLO** You Only Looks Once.

# **Première partie**

## **Problématique et état de l'art**



# Table des matières

---

<b>1 Problématique</b>	<b>17</b>
1 Détection d'anomalie . . . . .	18
2 Objectif . . . . .	20
3 Analyse des vidéos . . . . .	21
4 Jeux de données . . . . .	24
5 Exigences . . . . .	31
6 Conclusion . . . . .	33
<b>2 État de l'art</b>	<b>35</b>
1 La détection d'anomalie dans les vidéos . . . . .	36
2 Détection d'objets . . . . .	49
3 Explicabilité . . . . .	62
4 Conclusion . . . . .	63

---



# Chapitre 1

## Problématique

### Sommaire

---

<b>1</b>	<b>Détection d'anomalie</b>	<b>18</b>
<b>2</b>	<b>Objectif</b>	<b>20</b>
<b>3</b>	<b>Analyse des vidéos</b>	<b>21</b>
<b>4</b>	<b>Jeux de données</b>	<b>24</b>
<b>5</b>	<b>Exigences</b>	<b>31</b>
<b>6</b>	<b>Conclusion</b>	<b>33</b>

---

Le problème posé dans l'introduction générale est la détection d'événements représentant une anomalie dans un flux vidéo. Nous allons ici commencer par décrire la détection d'anomalie d'un point de vue général, avant de préciser les conditions particulières de notre problème.

## 1 Détection d'anomalie

La détection d'anomalie est un domaine de recherche regroupant de nombreuses thématiques. En fouille de données, la détection d'anomalies est l'identification d'éléments, d'événements ou d'observations qui ne sont pas conformes à un modèle attendu. Les anomalies sont aussi appelées valeurs aberrantes (outliers), nouveautés, bruits, écarts et exceptions.

En partant des travaux de la Team Gabru<sup>1</sup>, selon les problématiques traitées, la détection d'anomalie peut appartenir à l'une des catégories suivantes :

- Intrusion : la détection d'intrusion fait référence à la détection d'activités malveillantes au sein d'un réseau ou système.
- Fraude : la détection de fraude fait référence à la détection d'activités criminelles visant une organisation commerciale telles que des banques, des agences d'assurance, etc.
- Anomalie médicale : la détection d'anomalies dans le domaine de la santé se fait généralement sur les dossiers des patients dans le but de trouver certaines pathologies ou états anormaux.
- Dommage industriel : cette détection a pour but d'identifier tout dommage matériel pour prévenir d'éventuels pannes, pertes, arrêts ou encore d'autres incidents.
- Anomalie textuelle : dans ce contexte, les anomalies sont représentées par des phrases aberrantes ou encore des sujets, articles ou documents anormaux.
- Anomalie visuelle : cette détection étant basée sur des images, elle peut aussi bien traiter des images fixes ou des images provenant de vidéo.
  - Pour les images fixes, nous allons observer diverses régions de celles-ci afin de s'assurer qu'aucune zone n'est anormale.

---

1. Dpt of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal, <https://github.com/cs60050/TeamGabru>.

- Pour des vidéos, nous allons plutôt observer des images successives pour repérer tout changement anormal d'une image à une autre (détection de mouvement).

Quel que soit le domaine abordé, il existe trois différents types d'anomalies :

- Anomalie ponctuelle

Une instance individuelle peut être considérée comme anormale par rapport au reste des données.

- Anomalie contextuelle

Une instance de données est anormale dans un contexte spécifique (mais pas autrement), nous parlons alors d'anomalie contextuelle (également appelée anomalie conditionnelle). Chaque instance est définie à l'aide des deux attributs suivants :

#### 1. Attributs contextuels

Les attributs contextuels sont utilisés pour déterminer le contexte de cette instance. Par exemple : dans les données de séries chronologiques, le temps est un attribut contextuel qui détermine la position d'une instance sur toute la séquence.

#### 2. Attributs comportementaux

Les attributs comportementaux définissent les caractéristiques non-contextuelles d'une instance. Exemple : dans un ensemble de données spatiales décrivant la pluviométrie moyenne du monde entier, la quantité de précipitations à n'importe quel endroit est un attribut comportemental.

- Anomalie collective

Une collection d'instances de données anormales liées entre elles par rapport à l'ensemble de données complet.

Une anomalie peut être vue comme un événement ayant une très faible probabilité de survenir. Pour traiter ce genre de problématique, il existe trois méthodes (Chakraborty et al., [s. d.](#)) :

1. Déterminer les valeurs aberrantes sans connaissance préalable des données.  
Il s'agit essentiellement de clustering, méthode non supervisée.

2. Modéliser à la fois la normalité et l'anormalité. Cette approche est une tâche de classification binaire avec supervision nécessitant des données étiquetées comme normales ou anormales.
3. Modéliser uniquement la normalité ou, dans de très rares cas, modéliser l'anormalité.

Pour un aperçu complet de la détection d'anomalies, il faut se référer aux travaux de Chandola, Banerjee et Kumar ([2009](#)), qui présentent la détection d'anomalie de manière générale, en détaillant pour chaque type de données les principaux algorithmes.

## 2 Objectif

Les anomalies qui nous intéressent dans cette thèse sont celles ayant un impact direct sur la sécurité et la sûreté des personnes présentes dans les vidéos. Il s'agit de la détection d'incident potentiellement dangereux dans des données visuelles, donc des anomalies contextuelles. Une telle anomalie peut être vue comme un pattern (action, comportement) irrégulier par rapport à une situation donnée.

L'objectif est donc de séparer les anomalies en fonction de leur gravité. Nous avons par exemple des anomalies mineures répertoriant toute action allant à l'encontre des règles de vie établies, comme une personne à vélo qui roulerait sur le trottoir, ou ne portant pas de casque, une personne commettant un excès de vitesse ou encore une personne traversant alors que le signal lumineux ne lui était pas favorable. Comme vous pouvez le voir parmi cette catégorie, nous retrouvons tout un tas d'actions dont la plupart sont communes et que vous avez peut-être vous-même déjà réalisées. Dans le même contexte, nous pourrions ajouter à ce groupe certaines anomalies dites nuisibles, mais courantes, comme par exemple un vol à l'étalage, du vandalisme ayant pour but de dégrader un bien ou un lieu ou encore un bagage abandonné qui arrêterait la circulation des transports.

D'un autre côté, nous retrouvons des anomalies que l'on pourrait qualifier de majeures, car elles ont un impact direct sur la sécurité d'une personne ou d'un groupe de personnes. Parmi celles-ci nous retrouvons des accidents, incendies, explosions, des actions terroristes ou encore des catastrophes naturelles comme des cyclones, tremblements de terre ou tsunami. Les événements pouvant être perçus comme anomalies sont donc divers et variés et n'ont pas tous les mêmes conséquences.

On peut résumer cela par la définition de deux types de comportement : l'un ne nécessitant pas d'intervention (cas normal) et l'autre nécessitant une intervention humaine (cas anormal).

Avec les progrès de la technologie, les outils pour détecter ce genre d'anomalie sont nombreux (caméras embarquées sur nos smartphones, caméras publiques ou encore moyens de partage via les réseaux sociaux). Notre problème est de créer un modèle capable de détecter rapidement ces incidents à partir d'une vidéo et de déterminer s'ils doivent faire l'objet d'une intervention humaine. De plus, étant donné que ces anomalies peuvent avoir de graves conséquences, l'idéal serait d'arriver à les détecter le plus tôt possible.

### 3 Analyse des vidéos

De par sa nature complexe, une vidéo peut être analysée de quatre manières différentes :

- Analyser uniquement le son
- Analyser séparément chaque image
- Analyser les séquences d'images
- Analyser à la fois le son et les images

Étant donné que la majorité des vidéos provenant de caméras de surveillance ne possèdent pas de son, nous avons choisi d'écarter le son de notre problématique, ce qui nous laisse deux possibilités (S. Zhu, C. Chen et Sultani, 2020) :

- Analyse temporelle : détection de l'anomalie dans le temps (sur une séquence), en indiquant les images de début et de fin de l'événement ;
- Analyse spatiale : détection de l'anomalie dans l'espace (sur une image), en identifiant les pixels de chaque image correspondant à l'événement.

Notre problème est fondamentalement un problème d'analyse temporelle, mais il est intéressant de remarquer que l'analyse spatiale n'est pas à écarter. En effet, lorsque nous nous intéressons aux actions pouvant être catégorisées comme anomalies, nous constatons que certaines d'entre elles sont liées à des objets particuliers. En tant qu'humains, nous pouvons attester qu'une scène ou une vidéo contient un incendie par la présence de fumée ou de flamme, une bagarre en observant les différentes personnes présentes, un accident de la route en observant les véhicules, ou encore un danger si nous voyons une arme blanche ou une arme à feu.

D'où l'idée que nous développerons dans ce travail de combiner l'analyse spatiale, plus spécifiquement la détection d'objet, avec l'analyse temporelle, car les

objets aident les humains à interpréter les situations (pas d'accident de voiture sans voiture). Cette approche a déjà été proposée par Doshi et Yilmaz (2020), avec la combinaison de **YOLO** V3 pour détecter les objets et de Flownet 2 afin d'extraire des caractéristiques de flux optique (optical flow). Ces informations sont ensuite combinées pour créer un vecteur de caractéristique traité par un **KNN**. En 2023, une autre approche a été proposée par Mustafa. (2022). Sa méthode, nommée AVAD (Autoencoder-based Video Anomaly Detection), utilise un autoencodeur convolutionnel pour détecter les trames anormales présentes dans une vidéo (déttection d'anomalies temporelles), combiné avec **YOLO** V5 afin d'identifier les objets responsables de l'anomalie détectée.

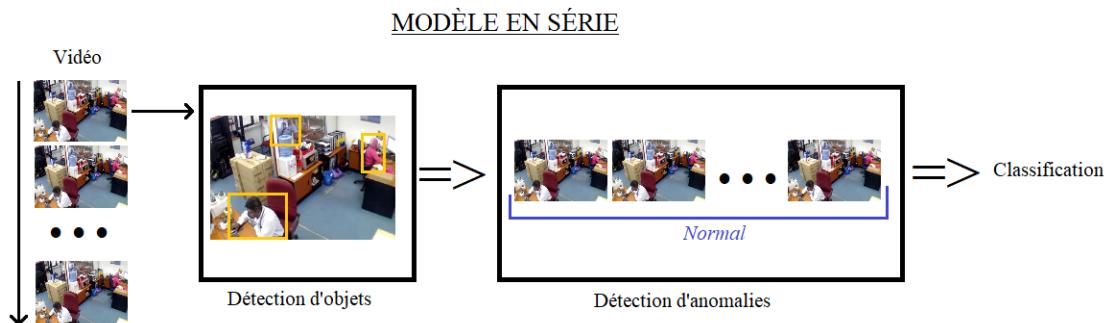
Bien sûr toutes les anomalies ne sont pas liées à un objet clé, ou bien ceux-ci ne sont pas toujours apparents. C'est généralement le cas dans des contextes de catastrophes naturelles : cyclone, tsunami, tremblement de terre. Dans ce genre de situation, il est parfois difficile de repérer des objets particuliers, comme le serait la vague en cas de tsunami ou encore une tornade.

Inversement, dans certaines circonstances, des objets clés peuvent être facilement identifiables mais ne pas représenter de danger. C'est par exemple le cas dans des aéroports où il est fréquent de croiser des militaires armés ayant pour but de garantir la sécurité de ces lieux.

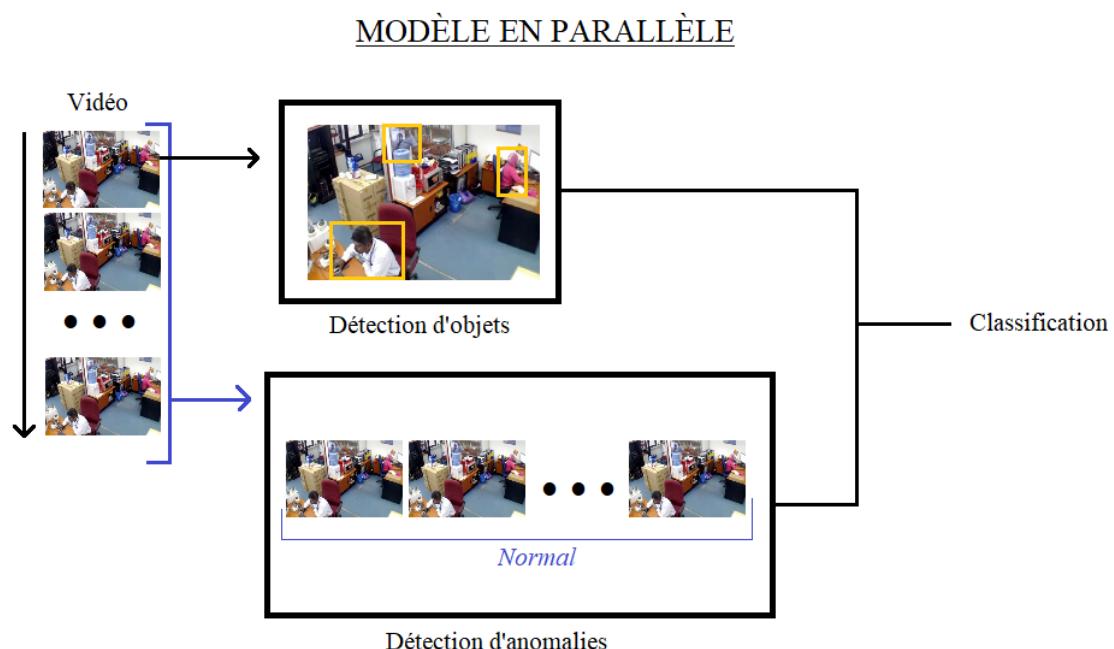
Lorsque nous sommes confrontés à ce genre de situations, nous sommes généralement capables de déterminer si elle comporte un risque en nous appuyant sur les objets environnants et plus particulièrement sur le comportement des personnes présentes dans la vidéo. En observant l'attitude des individus, nous pouvons déterminer le contexte et ainsi établir si la situation présente une anomalie. Par exemple, si une personne a une posture menaçante et une autre les mains en l'air, ou encore s'il y a un mouvement de panique, nous pouvons confirmer qu'une anomalie est en cours. Nous pouvons également utiliser les informations présentes dans les arrière-plans pour comprendre de quel type d'anomalie il est question. Si la caméra tremble, il pourrait s'agir de vents violents ou d'un tremblement de terre. Si des quantités anormales d'eau sont visibles, il pourrait y avoir eu un tsunami ou le débordement d'un fleuve ou d'une rivière.

Les deux analyses sont complémentaires et utilisent des types de données différents. L'analyse spatiale travaille sur des images statiques, tandis que l'analyse temporelle travaille sur des vidéos (séquences d'images). Pour combiner ces analyses, deux approches sont envisageables :

1. Déetecter les divers objets présents à l'écran puis vérifier qu'ils ne sont pas impliqués dans une anomalie. Ce qui revient à effectuer de la détection d'objet afin d'enrichir nos données avant une analyse de la séquence vidéo.



2. Analyser en parallèle la séquence ainsi que les objets présents.



Généralement, la détection d'anomalies peut se mesurer de deux manières :

1. Régression : retourner un score pour chaque donnée indiquant la probabilité que celle-ci soit une anomalie ;
2. Classification : attribuer une étiquette normale ou anormale (ou plus détaillée) à chacune de nos données.

Étant donné que nous souhaitons être capable d'identifier le type d'anomalie, nous partirons plutôt sur la seconde technique, à savoir étiqueter chaque élément

comme normal ou anormal ou même si possible attribuer une étiquette spécifique à chaque type d'anomalie. Concernant la méthode d'apprentissage, nous utiliserons des techniques d'apprentissage supervisée car, comme l'indiquent les travaux de la Team Gabru<sup>1</sup>, dans des contextes de données vidéo, elles surpassent à l'heure actuelle les techniques non supervisées.

## 4 Jeux de données

Pour être capable de faire la différence entre des actions normales et anormales, les algorithmes d'apprentissage neuronaux nécessitent d'être entraînés sur de grandes quantités de données. Ces données auront un impact direct sur les caractéristiques apprises par le modèle et sa prise de décision. Pour cela, il est donc nécessaire d'utiliser un jeu de données contenant un maximum de scènes représentant les cas que nous souhaitons traiter.

Dans cette section, nous allons passer en revue les différents jeux de données disponibles en lien avec notre problématique, pour voir dans quelle mesure ils répondent à notre problème. Nous partirons du récapitulatif fait par S. Zhu, C. Chen et Sultani (2020), présenté dans le tableau 1.1, que nous complèterons par quelques autres. Ce sont des jeux de données pour la détection d'anomalies dans des contextes de vidéosurveillance.

---

1. Dpt of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal, <https://github.com/cs60050/TeamGabru>.

Tab. 1.1 : Jeux de données cités par S. Zhu, C. Chen et Sultani (2020)

Dataset	# of Videos	Average Frames	Exemple Anomalies
UCSD Pred 1	70	201	Bikers, small carts
UCSD Pred 2	28	163	Bikers, small carts
Subway Entrance	1	121,749	Wrong direction, no payment
Subway Exit	1	64,901	Wrong direction, no payment
Avenue	37	839	Run, throw, new object
UMN	5	1,290	Run
DAD	1,730	100	Traffic accidents
CADP	1,416	366	Traffic accidents
A3D	1500	85	Traffic accidents
DADA	2000	324	Traffic accidents
DoTA	4677	156	Traffic anomalies, e.g. collision
Iowa DOT	200	27,000	Traffic accidents
ShanghaiTech	437	726	Bikers, cars
UCF Crime	1,900	7,247	Arson, accident, burglary, fighting
Street Scene	81	2509	Jaywalking, car illegally parked

Une partie de ces jeux de données traite uniquement de problèmes de circulation ; il s'agit de Dashcam Accident Dataset (DAD), Car Accident Dataset (CADP), A3D, Dota Detection of Traffic Anomaly (DADA), que nous avons écartés ; en effet, nous avons besoin d'un panel d'anomalies le plus diversifié possible. Parmi les jeux de données diversifiés, nous retrouvons des jeux de données tels que UCSD (voir figure 4.1) et ShanghaiTech, qui traitent d'anomalies que nous avons qualifiées de mineures comme des vélos roulant sur des voies réservées aux piétons, et que nous avons donc également écartées de notre champ d'étude. Plus intéressant, nous y retrouvons UCF Crime.

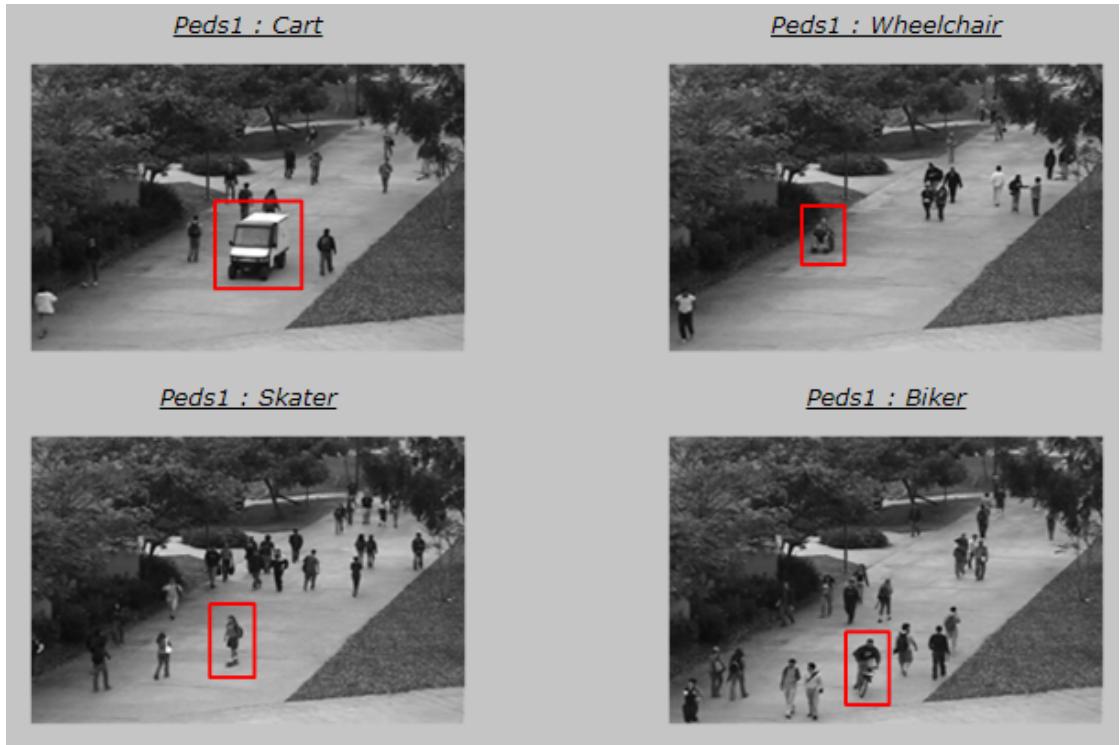


Fig. 4.1 : Extrait du jeu de données UCSD

UCF Crime est actuellement le jeu de données de référence pour la détection d'anomalies à partir de vidéos. Il a été conçu par Sultani, C. Chen et Shah (2018) à l'Université de Floride et contient 1900 vidéos non découpées représentant des anomalies réalistes couvrant 13 catégories : des abus, des arrestations, des incendies criminels, des agressions, des accidents de la route, des cambriolages, des explosions, des bagarres, des vols, des fusillades, des vols à l'étalage, des braquage et du vandalisme ainsi que 950 vidéos normales. L'ensemble d'apprentissage contient 800 vidéos normales et 810 vidéos anormales. Toutes les vidéos sont annotées temporellement avec des timestamps indiquant le début et la fin de chaque anomalie présente dans une vidéo. Ces annotations sont fournies sous forme d'un fichier CSV accompagnant le jeu de données. En outre, certaines vidéos comportent plusieurs anomalies, du même type ou de types différents. Par ailleurs, UCF Crime couvre aussi différentes conditions d'éclairage, résolutions d'image et angles de caméra.

Malheureusement, bien que son ensemble d'apprentissage contienne un nombre équilibré de vidéos (Normales/Anormales), cela n'est pas le cas des incidents qui y sont présents, ni même de leur durée. Le tableau 1.2 illustre la répartition des

vidéos et des anomalies pour chaque catégorie.

Tab. 1.2 : Répartition en catégories des vidéos de UCF Crime

Classe	Apprentissage		Test		Total	
	Vidéos	Séquences <sup>1</sup>	Vidéos	Séquences	Vidéos	Séquences
<b>Normal</b>	<b>800</b>	<b>800</b>	<b>150</b>	<b>150</b>	<b>950</b>	<b>950</b>
Abuse	48	56	2	2	50	58
Arrest	45	41	5	5	50	46
Arson	41	42	9	10	50	52
Assault	47	48	3	4	50	52
Burglary	87	87	13	15	100	102
Explosion	29	32	21	22	50	54
Fighting	45	67	5	5	50	72
Road Accident	127	134	23	23	150	157
Robbery	145	165	5	5	150	170
Shooting	27	33	23	25	50	170
Shoplifting	29	39	21	25	50	58
Stealing	95	137	5	7	100	144
Vandalism	45	59	5	8	50	67
<b>Total anomalies</b>	<b>810</b>	<b>940</b>	<b>140</b>	<b>156</b>	<b>950</b>	<b>1096</b>
<b>Total</b>	<b>1610</b>	<b>1740</b>	<b>290</b>	<b>306</b>	<b>1900</b>	<b>2046</b>

Selon la classe traitée, les séquences représentant un incident seront très variables, comme nous pouvons le voir sur les tableaux 1.3 (pour les données d'apprentissage) et 1.4 (pour les données de test), qui indiquent les durées des épisodes représentant une anomalie. Un coup de feu s'exprimera généralement en quelques secondes, ce qui n'est pas le cas d'une bagarre ou d'un vol mais le plus long reste la classe normale possédant des vidéos de plusieurs minutes, voire dépassant l'heure. Pour finir, certaines vidéos anormales ne contiennent aucune anomalie majeure, c'est par exemple le cas pour certaines vidéos de l'ensemble d'apprentissage de la classe "Arrest" ou encore "Burglary".

Tab. 1.3 : Durée des anomalies d'apprentissage de UCF Crime

Classes	Durée minimale	Durée moyenne	Durée maximale
<b>Normal</b>	6.7s	394.86s (6min 34)	32550s (9h20)
Abuse	2.88s	54.49s	110.2s (1min 50)
Arrest	6s	64.30s (1min 4)	255.88s (4min 15)
Arson	4.24s	32s	180s (3min)
Assault	1.92s	27.76s	278s (4min 38)
Burglary	1.68s	59.37s	321.2s (5min 20)
Explosion	5.2s	15.47s	30s
Fighting	3.6s	22.72s	62s
Road Accident	2.16s	8.35s	32s
Robbery	2.52s	36.15s	104.76s (1min 44)
Shooting	1.6s	4.93s	16.8s
Shoplifting	3.6s	15.5s	70.6s (1min 10)
Stealing	4.8s	31.78s	118.4s (1min 58)
Vandalism	3.6s	24.71s	81.6s (1min 21)

Tab. 1.4 : Durée des anomalies de test de UCF Crime

Classes	Durée minimale	Durée moyenne	Durée maximale
<b>Normal</b>	9.96s	144.20s (2min 24)	3599.9s (59min 59)
Abuse	3s	3.2s	3.4s
Arrest	12s	62.4s (1min 02)	124.8s (2min 04)
Arson	3.8s	32.632s	138s (2min 38)
Assault	14s	85.05s (1min 25)	276 (4 min 36)
Burglary	5.6s	44.542s	118.4s (1min 58)
Explosion	3s	13.856s	64.24s (1min 04)
Fighting	10.8s	35.464s	65.2s (1min 50)
Road Accident	1.2s	3.943s	7.6s
Robbery	4.8s	30.48s	72.6s (1min 12)
Shooting	3.2s	16.136s	69.6s (1min 09)
Shoplifting	2.4s	12.656s	108s (1min 48)
Stealing	2.4s	34.285s	123.6s (2min 30)
Vandalism	2.4s	11.35s	27.6s

S'ajoutent aux jeux de données précédents Movie Fight<sup>1</sup>, qui contient 200 extraits de films répartis en deux catégories "Fight" et "Non-fight", et "Hockey"

1. <https://www.kaggle.com/datasets/naveenk903/movies-fight-detection-dataset>

Fight”<sup>1</sup>, qui contient 1000 clips de bagarres survenues lors de matchs de hockey professionnels en Amérique du Nord entre les saisons 2009-2010 et 2018-2019 de la Ligue nationale de hockey (LNH).

En examinant ces jeux de données, nous avons constaté que ”Movie Fight” ne fournissait pas suffisamment de vidéos pour permettre l’apprentissage d’un modèle efficace, tandis que ”Hockey Fight” présentait un manque de diversité dans les scènes. Les extraits ont tous lieu dans les mêmes environnements, sont filmés avec les mêmes angles de vue et la même qualité d’image, ce qui peut limiter la capacité du modèle à s’adapter à des situations différentes.

En ce qui concerne UCF Crime, bien qu’il soit à l’heure actuelle le jeu de données de référence, de nombreux articles l’utilisant font état de mauvais résultats et recommandent de créer son propre jeu de données.

C’est par exemple le cas de Sernani et al. (2021) qui ont créé un jeu de données public appelé AIRTLab développé pour tester la robustesse des techniques de détection de la violence, plus particulièrement des faux positifs. Il est composé de 350 vidéos séparées en 2 catégories : violente et non-violente parmi lesquelles nous retrouvons des câlins, des applaudissements, des taquineries, etc...

Vrskova, Hudec, Kamencay et al. (2022a) font mention dans leur article que le jeu de données UCF souffre de données insuffisantes, mal nettoyées et d’un mauvais équilibrage. Contrairement aux vidéos normales qui peuvent faire plusieurs minutes, les vidéos anormales (anomalies), elles, ne représentent que quelques secondes. Ils ont donc décidé de créer le jeu de données Abnormal Activities contenant 1069 vidéos divisées en 11 classes contenant chacune environ 100 vidéos, parmi lesquelles nous retrouvons : menace au couteau, vol, bagarre, pollution (vidéos représentant des personnes qui jettent des ordures dans l’environnement), harcèlement, vandalisme, kidnapping, terrorisme, ivresse, mendicité (vidéos présentant des personnes qui agacent ou harcèlent d’autres personnes en mendiant ou en demandant de l’argent) ainsi que des vidéos normales. Ces scènes ont été réalisées dans différentes conditions d’éclairage. Lors de la création de l’ensemble de données, leur objectif était de simuler la disposition de caméras de sécurité dans des lieux publics tels que des bâtiments, des parkings et dans la nature.

Au vu de cette exploration, il était clair qu’aucun jeu de données ne présentait les qualités de diversité, de représentativité et de quantité que nous recherchions, même si certains de ces jeux peuvent être utilisés, en complément de notre jeu de données, pour le test.

Pour un jeu de données qui réponde à notre problématique, plusieurs solutions sont possibles :

---

1. <https://www.kaggle.com/datasets/yassershrief/hockey-fight-videos>

- Collecter de nouvelles données

Actuellement, il est nécessaire de réaliser manuellement les étapes de collecte, de nettoyage et de classification des données, car aucune technologie suffisamment avancée ne le permet. De plus, certaines anomalies ont une fréquence d'apparition moins élevée que d'autres. Par conséquent, elles sont plus difficiles à trouver. En raison de leur gravité, de la censure est parfois appliquée. Pour finir, dans une vidéo plusieurs actions peuvent avoir lieu, certaines étant considérées comme des anomalies tandis que d'autres sont normales. Ainsi, chaque vidéo contient soit une séquence normale, soit une ou plusieurs anomalies délimitées par des séquences normales. Par conséquent, un grand volume de vidéos ne signifie pas nécessairement une grande quantité de scènes exploitables.

- Utiliser de la data augmentation

Les vidéos étant composées de séquences d'images, une autre solution consiste à appliquer de la data augmentation sur chaque image afin d'augmenter de manière artificielle leur quantité. Parmi ces transformations, nous retrouvons des changements de luminosité, des effets miroir, des effets de flou, des zooms, etc. ou encore des techniques plus poussées nécessitant parfois l'utilisation de modèles de machine learning, comme la dégradation ou l'amélioration de la résolution de la vidéo ou encore de sa fluidité. Cependant, dans le contexte de la détection d'anomalies, l'utilisation de la data augmentation peut poser un problème. En effet, les données étant souvent peu nombreuses et brutes, les transformations artificielles peuvent altérer la qualité et la pertinence des données, ce qui peut compromettre les résultats de l'analyse.

- La génération de données synthétiques

Une dernière option est l'utilisation de réseau génératif afin de produire des données synthétiques. Cette technique a été utilisée par Jacob (2019) qui lui aussi dénonce le fait qu'aucun jeu de données n'a le volume ou la qualité suffisante pour entraîner correctement des modèles de deep learning et propose donc cette solution dans sa thèse.

En raison de contraintes budgétaires et de la disponibilité de ressources informatiques limitées, nous avons décidé de ne pas explorer l'option de la génération de données synthétiques pour résoudre ce problème de quantité de données. En effet, la création de données synthétiques nécessite une grande puissance de calcul, notamment pour l'apprentissage de réseaux génératifs, qui est très coûteux en termes de ressources. Par conséquent, nous avons opté pour d'autres approches,

telles que la collecte de nouvelles données et l'utilisation de techniques de data augmentation réalistes pour augmenter la quantité et la diversité de nos données.

Malgré cela la reconnaissance d'actions reste une tâche très difficile en raison des arrière-plans encombrés, des occlusions, des variations d'éclairage et de point de vue. Pour qu'un modèle soit capable de discerner avec précision une action, il faut donc posséder un ensemble de données contenant des vidéos aussi diverses que variées. L'énorme redondance d'informations complexifie la recherche d'informations discriminantes. D'après Arif et al. (2019), « Le succès des problèmes de reconnaissance d'action dépend d'un processus d'extraction de caractéristiques approprié. Cette extraction est très importante pour distinguer les variations présentes dans une scène. ». Chacune de nos vidéos devra donc être minutieusement nettoyée afin que notre modèle puisse extraire des caractéristiques pertinentes lui permettant par la suite d'être capable de généraliser son approche.

## 5 Exigences

Notre modèle ne devra pas se baser sur une classification binaire, mais plutôt être capable d'identifier le type précis d'anomalie, ce qui facilitera l'identification des personnes devant intervenir en cas de problème.

Cependant, ce choix implique qu'une liste représentative d'anomalies soit choisie pour l'apprentissage. Le modèle doit disposer d'une grande capacité d'adaptation pour détecter des anomalies absentes de cette liste. Il devra être capable de faire la différence entre des anomalies pouvant avoir de graves répercussions pour les personnes impliquées et celles ayant une incidence moins importante, afin de prioriser les interventions et ainsi garantir leur sécurité.

Il reste toutefois à déterminer s'il est préférable d'alerter souvent, même si cela peut entraîner des erreurs, ou de n'alerter que lorsqu'on est sûr, quitte à manquer certains cas.

De plus, il serait appréciable de réussir à localiser l'anomalie dans les données, ce qui rendrait le modèle plus transparent et permettrait à une personne physique de pouvoir vérifier la véracité de la détection effectuée. Mais comment faire comprendre au modèle ce qu'on attend de lui, lui indiquer où il doit porter son attention, sur le premier plan ou l'arrière-plan ? Les données traitées étant massives et complexes, une grande puissance de calcul est nécessaire. Par ailleurs, compte tenu du fait que les images se succèdent très rapidement, elles présenteront de ce fait une grande redondance d'informations qui devrait aider le modèle à déterminer les caractéristiques pertinentes.

Un point particulier concerne l'interprétabilité des résultats. Les algorithmes d'apprentissage automatique, en particulier neuronaux, sont couramment utilisés

pour nous aider à résoudre les différentes problématiques auxquelles nous nous heurtons. Malgré leur efficacité, ils sont généralement perçus comme des structures assez complexes et opaques (boîtes noires), capables de fournir une réponse au problème sans vraiment nous expliquer les décisions ayant conduit à celle-ci. D'une part, cela les rend difficiles à déboguer ou à améliorer lorsque les performances n'ont pas atteint l'objectif fixé, et d'autre part cela soulève de nombreux problèmes d'ordre éthique ou juridique. Comment avoir confiance en une IA si nous ne pouvons pas expliquer ses choix ? Comment transmettre efficacement ces informations à des non-experts si nous ne sommes pas capables de les comprendre ?

L'entrée en vigueur de la **Règlement Général sur la Protection des Données (RGPD)** en mai 2019 plus particulièrement l'article 22-1 stipulant qu'une décision ne peut être fondée exclusivement sur un traitement automatisé, ajouté au fait qu'un humain a besoin d'éléments explicites et non opaques pour prendre une décision, ont fortement accéléré les recherches dans ce domaine. C'est pourquoi l'interprétabilité et l'explicabilité sont devenues des enjeux majeur dans la recherche. De par cet intérêt, de nombreux travaux sont réalisés chaque année, particulièrement concernant les réseaux à convolution utilisant des images (Olah, Satyanarayan et al., 2018 ; Olah, Mordvintsev et Schubert, 2017 ; Bau et al., 2020 ; Zhang, Y. N. Wu et S.-C. Zhu, 2017).

D'après la **Commission Nationale de l'Informatique et des Libertés (CNIL)**, l'explicabilité « est la capacité de mettre en relation et de rendre compréhensibles les éléments pris en compte par le système d'IA pour la production d'un résultat. » Cet aspect du machine learning a donc pour but de nous aider à comprendre les mécanismes de prise de décision d'un modèle.

C'est pourquoi, en tant que contrainte supplémentaire, nous ajoutons la nécessité de visualiser la zone dans laquelle l'anomalie a lieu. Cette pratique est déjà mise en place pour la détection d'anomalies dans les images, et nous voulons donc l'intégrer pour la détection d'anomalies à partir de vidéos.

Pour finir, dans cette thèse, nous souhaitons effectuer de la détection d'anomalie et non de la prédiction. Notre analyse est effectuée *a posteriori* une fois l'action terminée, de ce fait, le temps de latence dépend du temps contenu dans la séquence à analyser. Étant donné les conséquences que peuvent avoir certaines anomalies, notre objectif sera donc de fournir une analyse dans le plus court délai possible, même si celle-ci n'est pas en temps réel à proprement parler.

Cette thèse étant une CIFRE, il est important de mentionner les contraintes imposées par l'entreprise. Les principales contraintes sont :

1. Il ne faut pas manquer des alertes potentielles, quitte à déclencher trop d'alertes ;

2. le modèle final doit pouvoir fonctionner aussi bien avec une carte CPU qu'avec des cartes GPU ;
3. Les types d'anomalies étudiés représentent des catégories reconnues par l'entreprise (bagarres, coup de feu, etc...).

## 6 Conclusion

Nos principaux défis sont donc les suivants :

- Constituer un jeu de données propriétaire et représentatif des différents types d'anomalies étudiées. Le problème central qui a initié ce défi réside dans le manque de données adéquates pour notre projet, comme souligné dans la section sur les jeux de données présentée précédemment. Les jeux de données existants, tels que "Movie Fight", "Hockey Fight" ou encore "UCF CRIME", ont mis en lumière les limitations en termes de volume et de diversité, entravant ainsi notre capacité à développer un modèle de détection d'anomalies performant. Face à cette lacune majeure, la nécessité d'obtenir un ensemble de données qui puisse couvrir la variété des anomalies que nous visons devient impérative.
- Labelliser les données pour constituer un ensemble de référence (*ground truth*) représentant clairement la frontière entre ce qui est normal et anormal. L'obligation de fournir un référentiel fiable et cohérent pour l'apprentissage de notre modèle a conduit à ce défi crucial. Étant donné que notre modèle est intrinsèquement lié à la capacité de distinguer les comportements normaux des anomalies, la précision de ce référentiel est de la plus haute importance, afin qu'il puisse apprendre de manière précise et cohérente à identifier les anomalies dans diverses situations.
- Découper les vidéos étiquetées pour ne conserver que les événements représentatifs. Ce défi découle de la nécessité de concentrer l'attention du modèle sur les événements les plus significatifs. L'énorme quantité de données vidéo crée un dilemme entre la conservation de la pertinence des informations et l'élimination du superflu. Les contraintes de capacité de stockage et de puissance de calcul jouent un rôle important dans la décision de découper les vidéos, visant à préserver les moments clés tout en évitant la surcharge d'informations non essentielles.
- Augmenter la quantité de données disponibles et équilibrer si besoin est. L'entraînement efficace d'un modèle de détection d'anomalies, surtout en

utilisant des techniques d'apprentissage profond, nécessite une quantité importante de données. Cependant, les données disponibles sont souvent insuffisantes et déséquilibrées en termes de fréquence des anomalies. Pour atteindre les niveaux de performance souhaités, la nécessité d'augmenter la quantité de données et de les équilibrer pour assurer une représentativité adéquate est nécessaire.

- Nettoyer les données pour filtrer le bruit présent dans celle-ci. La contrainte de qualité des données a conduit à ce défi. Les données bruitées ou incorrectement enregistrées peuvent altérer la capacité du modèle à discerner des motifs significatifs. La suppression du bruit, cependant, n'est pas sans défis. L'équilibre entre la préservation des informations pertinentes et la suppression des éléments indésirables est complexe, car la qualité des données nettoyées est essentielle pour garantir la performance et la fiabilité du modèle de détection.
- Concevoir un modèle permettant d'effectuer une analyse en quasi-temps réel, combinant analyse spatiale et temporelle. Motivée par la nécessité d'une intervention rapide dans des situations critiques, notre choix de développer un modèle d'analyse quasi-temps réel a pris tout son sens. Notre réflexion sur la conception de celui-ci a été inspirée par le désir de reproduire un comportement humain fusionnant l'analyse spatiale et temporelle, avec l'intention de capturer la manière dont les êtres humains appréhendent leur environnement en mettant l'accent sur les éléments saillants et en observant leur évolution dans le temps.
- Visualiser les caractéristiques utilisées dans la prise de décision. Les nouvelles réglementations, liées à la RGPD, ont suscité le besoin de rendre les décisions du modèle transparentes et compréhensibles. Le défi de visualiser les caractéristiques sélectionnées par le modèle a été provoqué par le souhait de répondre aux normes éthiques et légales, tout en garantissant la confiance dans les décisions automatisées.

Dans la suite de ce document, nous présenterons un état de l'art de la détection d'anomalie (analyse temporelle) et de la détection d'objet (analyse spatiale), puis nous expliquerons nos contributions et nos réponses à ces challenges.

# Chapitre 2

## État de l'art

### Sommaire

---

<b>1</b>	<b>La détection d'anomalie dans les vidéos . . . . .</b>	<b>36</b>
1.1	Convolutional RNN . . . . .	36
1.2	Convolution 3D . . . . .	39
1.3	Convolutional Auto encodeur . . . . .	40
1.4	Temporal Convolutional Network (TCN) . . . . .	42
1.5	Transformer . . . . .	44
1.6	Modèles génératifs . . . . .	48
<b>2</b>	<b>Détection d'objets . . . . .</b>	<b>49</b>
2.1	RCNN / Fast RCNN et Faster RCNN . . . . .	50
2.2	YOLO . . . . .	52
<b>3</b>	<b>Explicabilité . . . . .</b>	<b>62</b>
<b>4</b>	<b>Conclusion . . . . .</b>	<b>63</b>

---

# 1 La détection d'anomalie dans les vidéos

Le domaine de la vision par ordinateur regroupe de nombreuses techniques d'analyse de données visuelles pouvant être utilisées pour la détection d'anomalies, par exemple, le calcul de la trajectoire des objets présents à l'écran, l'analyse des histogrammes de couleur ou encore la réalisation de l'optical flow. Ces technologies doivent être calibrées à la main et ont généralement besoin d'ajustements selon la scène traitée. C'est ce problème que tentent de résoudre les technologies de deep learning en extrayant automatiquement de nos données des caractéristiques pertinentes afin d'être capables de généraliser leur analyse. De ce fait, nous nous sommes concentrés sur ce type de technologie. Pour plus d'informations concernant les techniques d'analyse standard, nous nous référerons à Jacob (2019).

## 1.1 Convolutional RNN

Les réseaux récurrents (**Recurrent Neronal Network (RNN)**) sont considérés comme les modèles neuronaux les plus adaptés pour le traitement de séries temporelles. Ils ont été mis au point par Elman (1990) et Jordan (1997) qui proposèrent tous deux leur propre version. Contrairement aux réseaux à propagation avant (Feed Forward), ils comportent des boucles de rétroaction entre les unités, qui leur permettent de mémoriser des séries temporelles dynamiques. Le plus connu d'entre eux, **Long Short Term Memory (LSTM)**, a été proposé par Hochreiter et Schmidhuber (1997), puis amélioré par Gers, Schmidhuber et Cummins (2000) avec l'introduction d'une porte d'oubli. L'avantage de ce type de réseau est que certaines informations liées aux données vues précédemment sont conservées et utilisées dans la prise de décision. Par ce mécanisme, les réseaux **LSTM** s'imposent comme des références dans l'analyse de séries temporelles.

**LSTM** n'est pas le seul **RNN** à avoir fait ses preuves dans ce cadre. Un autre modèle, **Gated Recurrent Unit (GRU)**, possède une architecture moins complexe, mais tout aussi performante. Là où un **LSTM** est composé d'une architecture à trois portes, avec une porte d'oubli (forget gate), une d'entrée (input gate) et une de sortie (output gate), le **GRU** lui n'en possède que deux : une porte de reset (reset gate) et une porte de mise à jour (update gate). La figure 1.1 détaille ces trois architectures.

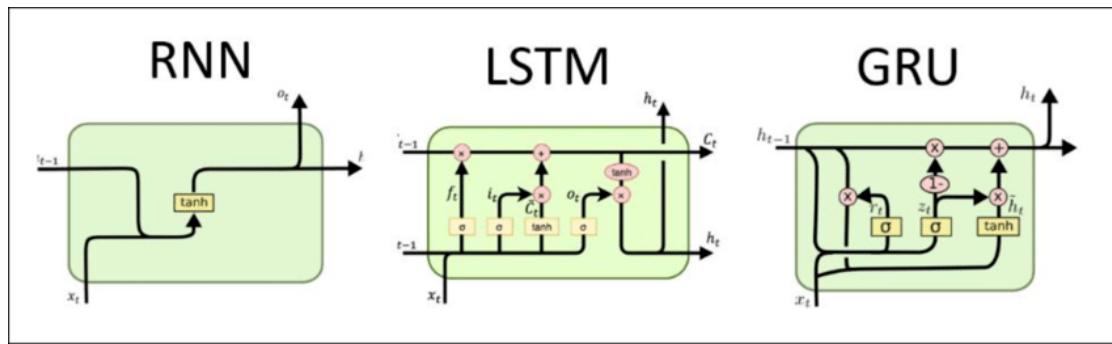


Fig. 1.1 : Architectures récurrentes (Toharudin et al., 2021)

Bien que les **RNN** soient capables de gérer correctement la temporalité liée aux données vidéo, ils ne sont pas adaptés pour traiter les images qui les composent. Cela est dû au fait que les images sont des données complexes avec des informations cruciales codées par la position des pixels. Malheureusement, les **RNN** ne sont pas conçus pour prendre en compte cette structure spatiale, ce qui les rend inappropriés pour traiter directement des images.

Pour effectuer du traitement d'image, l'approche courante est d'utiliser des réseaux à convolution appelés **Convolutional Neuronal Network (CNN)**. Ces réseaux, introduits par LeCun et al. (1989), sont capables d'extraire des caractéristiques spatiales à partir d'images en utilisant des opérations appelées convolutions. Parmi ces architectures, on peut citer **VGG19** un modèle proposé par Karen Simonyan et Andrew Zisserman (Simonyan et Zisserman, 2014) composé de 23 couches regroupées en 5 principaux blocs composés chacun de 2 à 4 couches de convolution suivies d'une couche de pooling (figure 1.2).

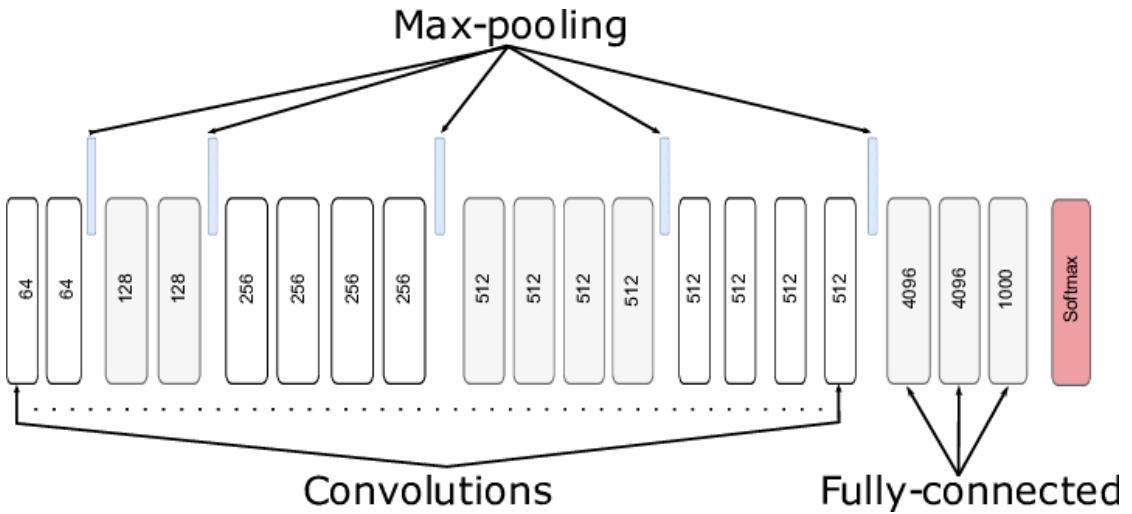


Fig. 1.2 : Architecture de VGG19 (Lagunas et Garces, 2018)

D'autres architectures sont également utilisées en traitement d'images. C'est par exemple le cas de ResNet (He et al., 2015), un réseau profond basé sur l'idée de "raccourcis". L'ajout de connexions résiduelles permet d'éviter la diminution de la performance due à l'ajout de couches supplémentaires. Inception (Szegedy, W. Liu et al., 2014) utilise des blocs d'inception pour extraire des caractéristiques à différentes échelles spatiales à partir d'une même image. InceptionResNet (Szegedy, Ioffe et al., 2016) combine les avantages de ResNet et Inception. Xception (Chollet, 2016) est une version améliorée de Inception qui utilise des convolutions séparables en profondeur en appliquant une convolution sur chaque canal de l'entrée, suivie d'une convolution spatiale. Cette méthode permet une réduction significative du coût de calcul tout en améliorant la précision du modèle. Ou encore, EfficientNet (Tan et Le, 2019) une famille d'architectures de réseaux à convolution qui utilisent une approche d'optimisation de l'équilibre entre la profondeur, la largeur et la résolution de l'image pour maximiser la précision tout en minimisant les coûts de calcul.

Afin de traiter des données vidéo, ces réseaux ont donc été combinés avec des réseaux récurrents tels que **LSTM** et ont donné lieu à deux types d'architectures, qui diffèrent en fonction du degré d'intégration des deux sortes de modèles : les Convolutional **LSTM** (Shi et al., 2015) (souvent abrégées en **CNN + LSTM**) et les ConvLSTM. Dans l'architecture Convolutional **LSTM** chaque image traverse les différentes couches convolutives pour produire un vecteur de caractéristiques, qui est ensuite traité par le **LSTM**. L'architecture ConvLSTM est un réseau de type **LSTM** dans lesquelles les multiplications matricielles internes ont été remplacées

par des convolutions.

Que ce soit pour de la reconnaissance d'action (Shi et al., 2015), de la détection d'anomalies (Majd et Safabakhsh, 2019) ou de la détection d'actions violentes (Vrskova, Hudec, Kamencay et al., 2022a ; Vrskova, Hudec, Sykora et al., 2020 ; Oliveira Lima et Figueiredo, 2021 ; Jahlan et Elrefaei, 2021), nombreux sont les travaux utilisant ce type de modèle pour effectuer de l'analyse de vidéo.

Bien qu'ils soient moins mentionnés dans la littérature, les modèles Convolutional **GRU** sont généralement plus performants que les Convolutional **LSTM**, et ils sont également moins coûteux en puissance de calcul (Ravi et Karray, 2021).

## 1.2 Convolution 3D

Ji et al. (2010) ont développé un modèle à base de convolutions 3D (**C3D**), pour la reconnaissance d'actions dans un flux vidéo. L'idée générale est d'observer chacune des images et prendre en compte les précédentes et suivantes afin de percevoir les différents mouvements réalisés par les objets présents à l'écran.

Contrairement aux approches de type Convolutional **LSTM** / **GRU**, qui réduisent chaque image en un vecteur de caractéristiques 1D, les convolutions 3D conservent les informations liées à la profondeur, et, dans le cas des vidéos, les changements inter images. Pour y parvenir, les noyaux de chaque couche se déplacent dans trois directions (x, y, z) afin de réaliser une carte d'activation 3D (figure 1.3).

Une autre différence est qu'au lieu de chercher des informations sur chaque image comme le ferait une convolution 2D, puis de confier à un réseau récurrent le rôle de les lier dans le temps, **C3D** se base sur des paquets d'images successives afin d'analyser les différences entre celles-ci et ainsi capturer les informations liées au mouvement. Cette technique a montré de très bons résultats dans de nombreux domaines tels que l'estimation de la pose humaine dans les images ou les vidéos, la reconnaissance d'actions (Ji et al., 2010 ; Vrskova, Hudec, Kamencay et al., 2022b ; Tran et al., 2014) et la segmentation d'images médicales.

En raison de leur efficacité à traiter des vidéos, les convolutions 3D ont aussi été utilisées dans des problématiques de détection d'anomalies (Mishra et al., 2020). Mais ces performances sont accompagnées d'une énorme charge de calcul.

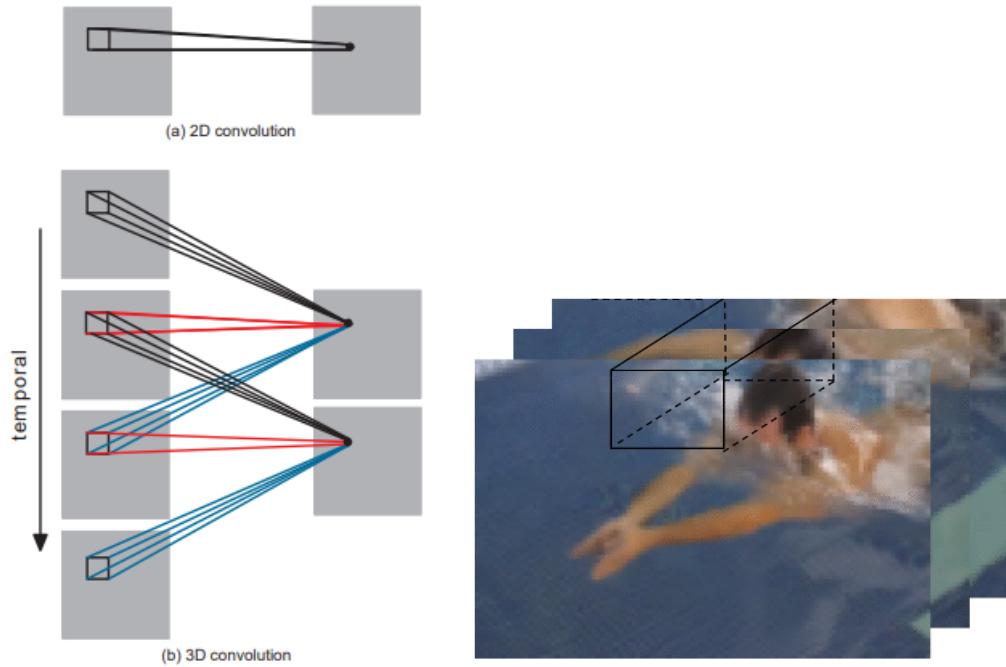


Fig. 1.3 : Fonctionnement des filtres de convolution, Ji et al. (2010) et Ghosh (2018)

### 1.3 Convolutional Auto encodeur

La détection d'anomalies dans la vidéo-surveillance est difficile en raison de la variété des types d'anomalies (voir chapitre 1), qui limitent l'utilisation de techniques supervisées. Dans ce cas-là, il est possible d'opter pour des approches dites non supervisées ne nécessitant aucune donnée étiquetée. Dans ce genre d'approche, les modèles de type auto-encodeur sortent du lot.

L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur (figure 1.4). Son fonctionnement est assez simple, les données d'entrée vont d'abord être encodées puis être passées au décodeur afin d'être reconstruites (décodées) (figure 1.5). La performance de ce modèle peut donc être mesurée en comparant les données initiales avec les données reproduites. Dans le cadre d'une détection d'anomalie, toute donnée aberrante proposée au modèle sera difficile à reconstruire et donc assez simple à identifier.

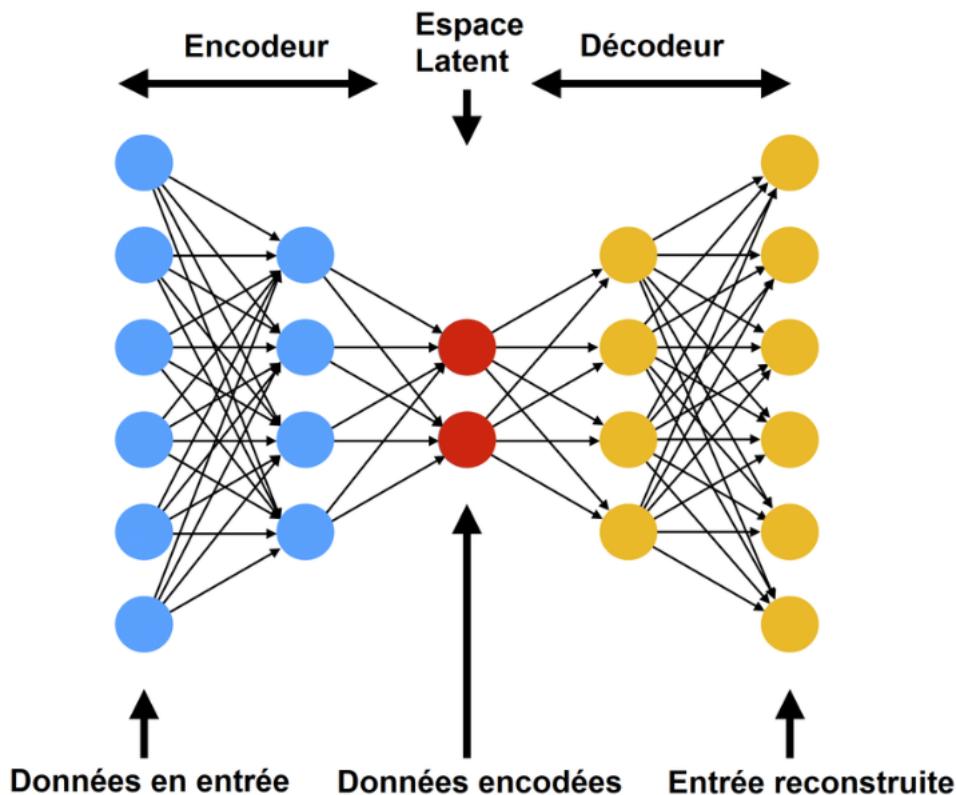


Fig. 1.4 : Architecture d'un auto-encodeur, Sublime (2022)

Pour le traitement d'images, l'encodeur et le décodeur sont normalement des **CNN**, très utilisés dans le cadre d'une détection d'anomalies visuelles (Jacob, 2019 ; M. Ribeiro, Lazzaretti et Lopes, 2018 ; An et Cho, 2015 ; Z. Chen et al., 2018 ; Hasan et al., 2016).

La représentation compressée (la sortie de l'encodeur et l'entrée du décodeur) ont généralement une structure 1D, ce qui peut entraîner une perte d'information potentiellement utile pour la reconstruction de l'action présente dans la vidéo.

Les auto-encodeurs possèdent de nombreuses variantes. Dans le contexte de traitement de données vidéo l'une d'elles consiste à remplacer les **CNN** par des convolutional **LSTM** dans le but de combiner les avantages liés aux **RNN** et aux auto-encodeurs, ce qui permet de reconstruire les images d'entrée à partir des images mémorisées précédemment (Rumelhart, Hinton et Williams, 1986 ; Medel et Savakis, 2016 ; Chong et Tay, 2017 ; Deepak, Chandrakala et Mohan, 2020 ; L. Wang et al., 2018 ; Luo, W. Liu et Gao, 2017).

Malgré leur efficacité lors de la phase de décodage, les auto-encodeurs reconstruisent chaque image de la vidéo, ce qui demande du temps et de la puissance de

calcul et limite donc leur utilisation dans des problématiques de temps réel où la réactivité est essentielle.

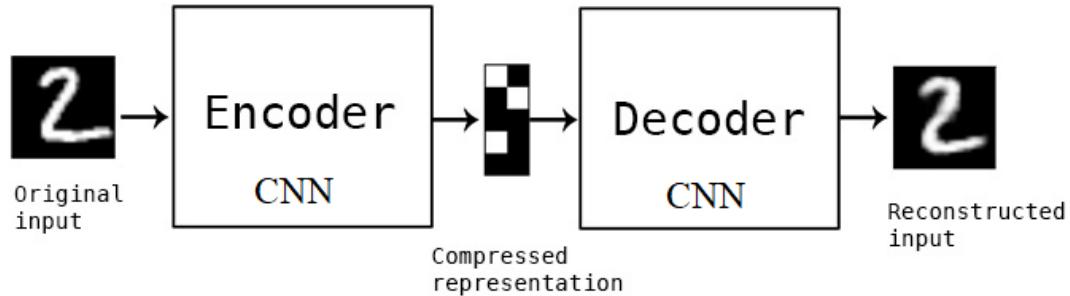


Fig. 1.5 : Auto-encodeur d'images, Chollet et al. (2015)

## 1.4 Temporal Convolutional Network (TCN)

Les Temporal Convolutional Networks **TCN** sont une famille de modèles de réseaux de neurones convolutifs conçus pour la modélisation de séquences temporelles (figure 1.6). Contrairement aux **RNN** traditionnels, qui utilisent des connexions récurrentes pour capturer la dépendance temporelle, les **TCN** utilisent des convolutions sur le temps pour capturer les motifs temporels dans les séquences.

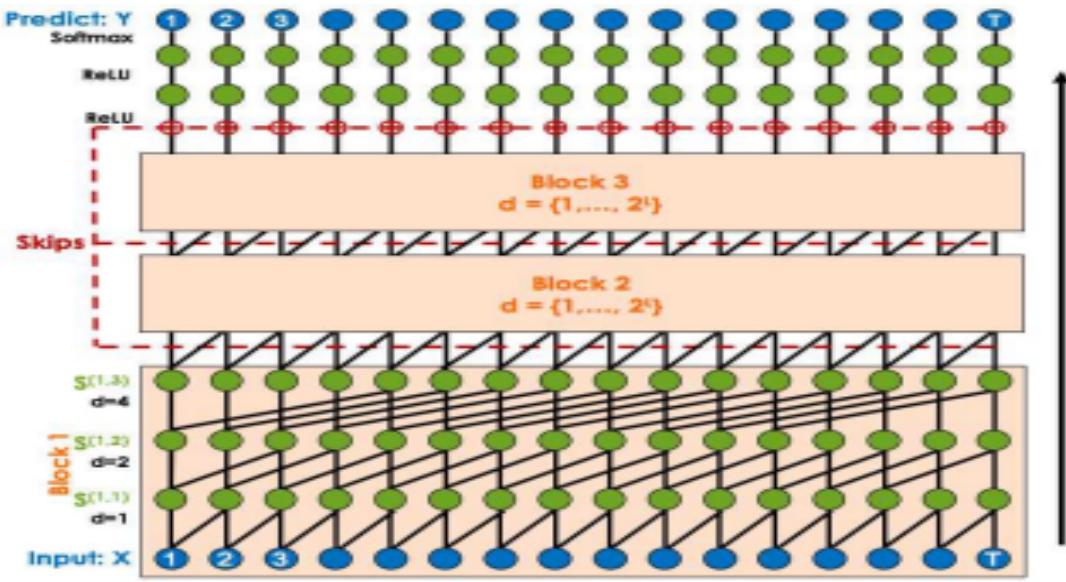


Fig. 1.6 : Architecture du TCN (Lea et al., 2016)

Les filtres dilatés jouent un rôle important dans la capacité des TCN à capturer des motifs temporels à différentes échelles. Les filtres dilatés sont des convolutions où les indices des valeurs à prendre en compte sont espacés d'un certain nombre de pas, appelé taux de dilatation. Cela permet aux TCN de capturer des motifs temporels à des échelles différentes, tout en ayant une complexité computationnelle raisonnable. De plus, les filtres dilatés peuvent être causaux ou non causaux. Les filtres dilatés causaux sont utiles pour les tâches de reconnaissance où il est important que le modèle ne puisse pas voir dans le futur. Les filtres dilatés non causaux, en revanche, peuvent être utilisés pour les tâches de classification où la séquence entière est disponible dès le début.

Lea et al. (2016) ont introduit une architecture de type encodeur-décodeur utilisant des TCN pour la segmentation et la détection d'actions dans les vidéos. La figure 1.7 montre comment les convolutions temporelles sont utilisées pour coder les séquences temporelles d'entrée ; puis les déconvolutions temporelles sont utilisées pour reconstruire les séquences de sortie. Les résultats expérimentaux semblent montrer que les TCN sont capables de capturer des motifs temporels complexes tels que les mouvements composant une action, tout en étant résistants aux variations temporelles. Cela signifie qu'ils peuvent reconnaître des actions même si elles ne se produisent pas exactement au même moment ou si elles ont des durées différentes à chaque occurrence.

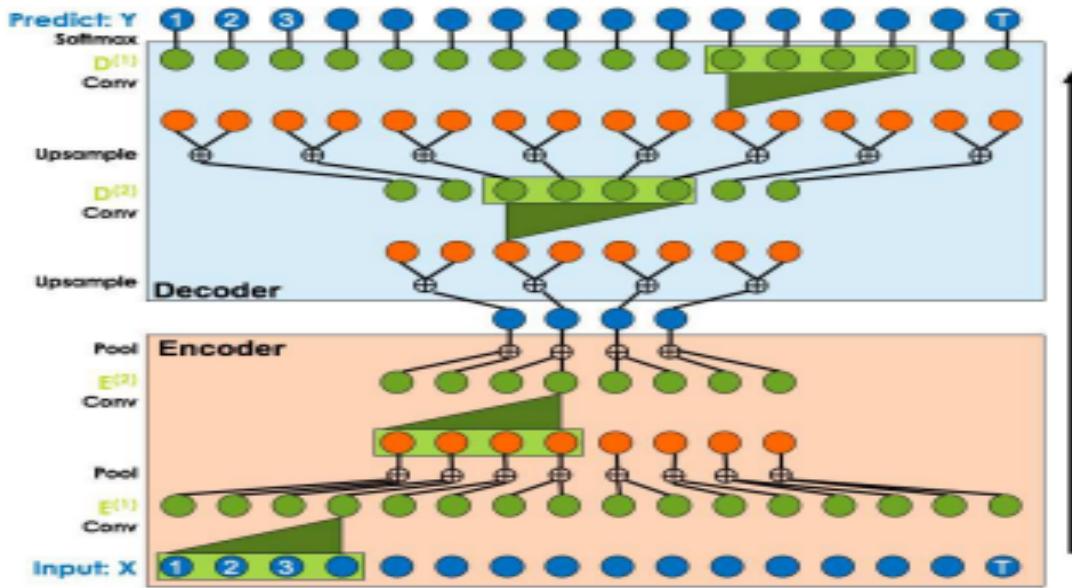


Fig. 1.7 : Architecture de l'auto encodeur à base de TCN (Lea et al., 2016)

## 1.5 Transformer

Les transformateurs sont des modèles introduits par Vaswani et al. (2017) dans le but de réaliser de l'analyse textuelle. Ces modèles peuvent être vus comme des modèles « sequences to sequences », c'est-à-dire qu'ils prennent en entrée une séquence de données et produisent en sortie une autre séquence de même type. Tout comme les RNN, ils sont capables de traiter des données séquentielles ; ils sont généralement utilisés pour la traduction.

Leur architecture (figure 1.8) est inspirée des auto-encodeurs. Elle consiste en un empilement d'encodeurs et de décodeurs (le même nombre de chaque côté). Chaque donnée va traverser les différents encodeurs. Une fois encodée, elle sera envoyée aux décodeurs et remontera les différentes couches. L'entrée de chaque décodeur contiendra à la fois l'information venant du décodeur précédent ainsi que celle provenant du dernier encodeur. Sa particularité est que chaque bloc du réseau à savoir chaque encodeur / décodeur possède un mécanisme d'attention, qui permet d'évaluer à quel point deux informations sont liées ou dissociées.

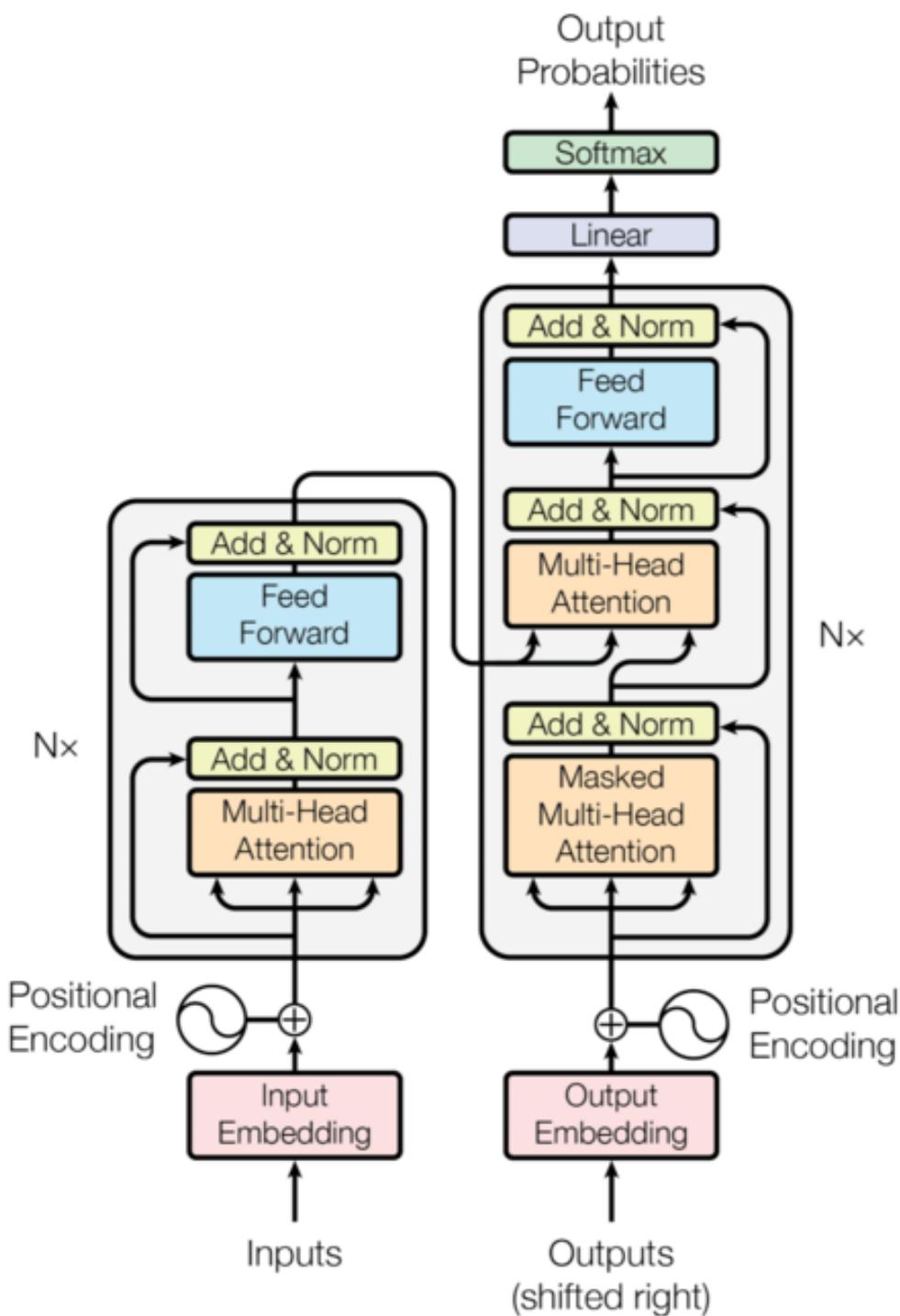


Fig. 1.8 : Architecture d'un transformer, Vaswani et al. (2017)

Plus récemment, en 2020-2021, les équipes de Google ont proposé une adaptation des réseaux de type transformer dans le but de traiter des images ou données vidéo (Dosovitskiy et al., 2020; Arnab et al., 2021) (figures 1.9, 1.10). Grâce au mécanisme d'attention, ces réseaux sont efficaces dans des tâches comme la détection d'objet ou la segmentation d'image. Ces modèles sont appelés des **Vision Transformer (ViT)**.

En 2021, des chercheurs de Facebook (Caron et al., 2021) ont proposé à leur tour un modèle nommé **DInstillation with NO label (DINO)** basé sur le même type de technologie. Ce modèle utilise donc des **ViT** couplés à de l'apprentissage auto-supervisé, ce qui lui permet de comprendre de façon autonome les zones importantes d'une image ou d'une vidéo, zone qui peut par la suite être visualisée à l'aide de cartes d'attention (figure 1.11).

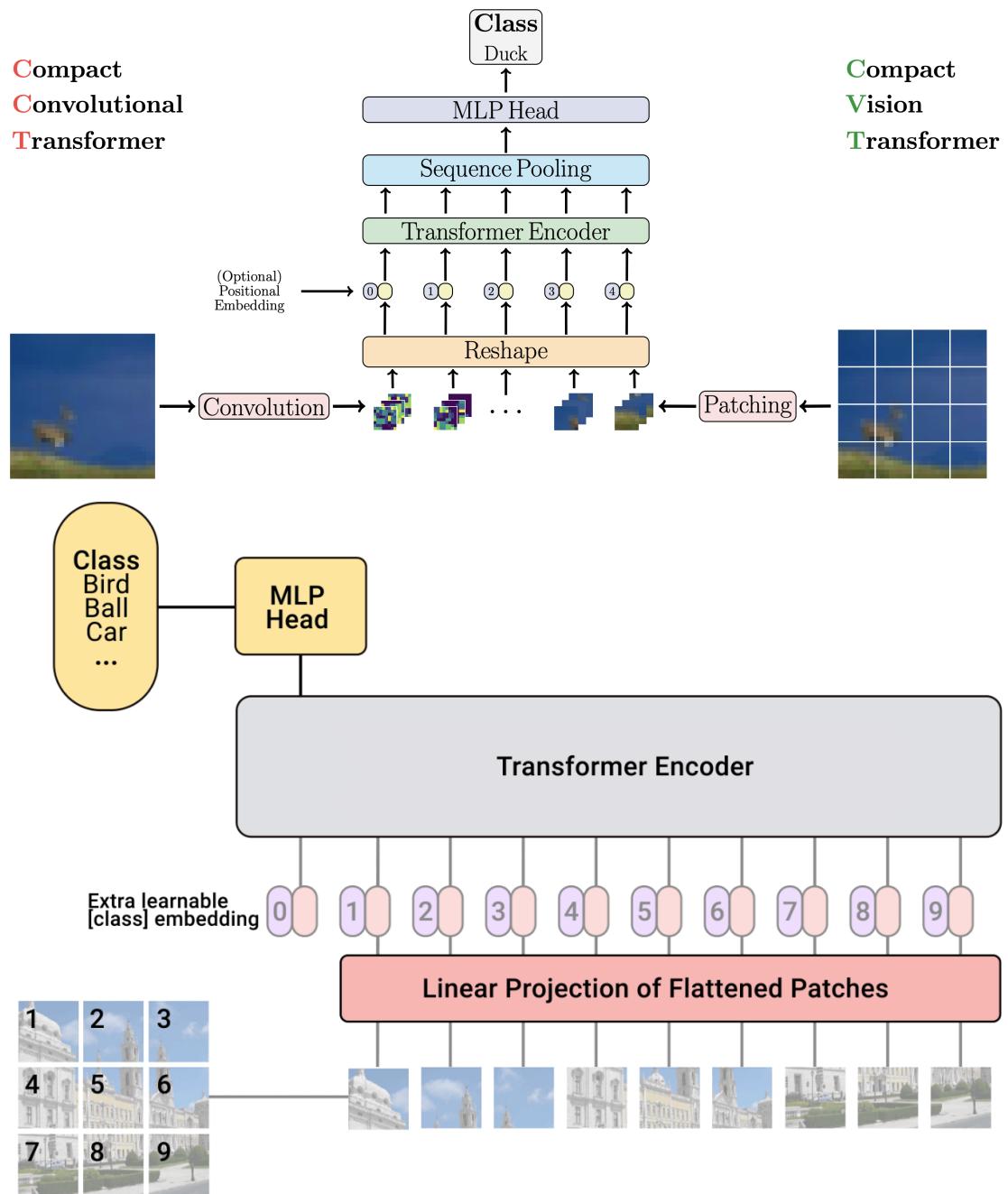


Fig. 1.9 : Vision transformer, P. Wang (2020) et Hassani et al. (2021)

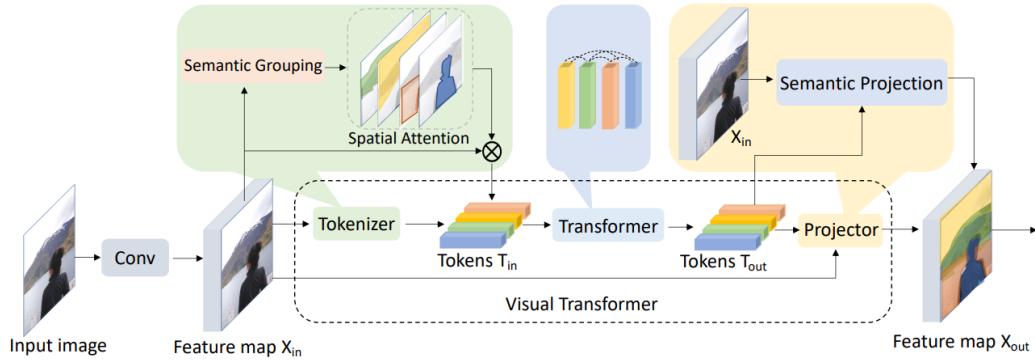


Fig. 1.10 : Vision Transformer, B. Wu et al. (2020)



Fig. 1.11 : Carte d'attention générée par DINO, Caron et al. (2021)

Pour plus d'informations concernant les différents types de vision transformers disponibles à ce jour, il faut se référer à l'étude de P. Wang (2020).

## 1.6 Modèles génératifs

Pour la détection d'anomalies dans des données vidéo, certaines recherches utilisent des modèles de type **Generative Adversarial Networks (GAN)** (Lin et al., 2021 ; Dimokranitou, 2017 ; Schlegl et al., 2017). De nos jours, ces modèles sont largement utilisés pour générer des données synthétiques (image, vidéo) ou encore pour améliorer la résolution de ce type de données.

**GAN** a été proposé par Goodfellow et al. (2014). Il est composé de deux sous modèles : un modèle génératif et un modèle discriminatif (figure 1.12). La tâche du générateur est de créer des données d'apparence naturelle similaires aux données originales. Le modèle discriminatif a pour tâche de déterminer si une donnée a une

apparence naturelle ou semble avoir été artificiellement créée. Au fur et à mesure de l'apprentissage, les deux modèles sont améliorés via ce qu'on appelle l'optimisation alternée jusqu'à ce que le discriminateur ne puisse plus faire la différence entre les données synthétiques (contrefaçons) et les données authentiques. L'idée est de former un modèle sur des données vidéo afin que celui-ci soit capable d'en produire synthétiquement puis d'utiliser le discriminateur en tant que classifieur afin de reconnaître les données anormales, c'est-à-dire toutes vidéos contenant une action représentant une anomalie.

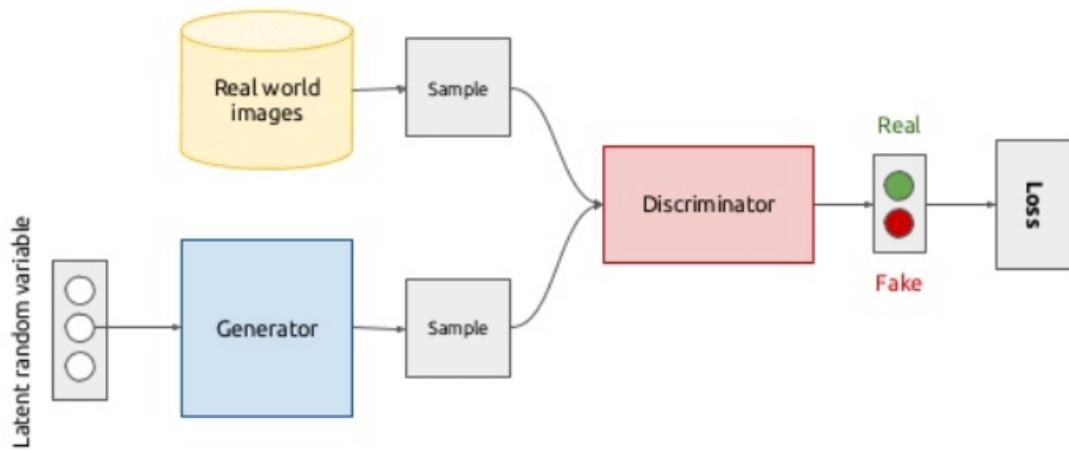


Fig. 1.12 : GAN vanilla Architecture, Cai et al. (2020)

S. Zhu, C. Chen et Sultani (2020) fournissent une discussion complète sur les méthodes d'apprentissage profond (deep learning) supervisées ou non, utilisées pour la détection d'anomalies dans les vidéos de surveillance. Par ailleurs, Ghosh (2018) étudie la reconnaissance d'actions dans les données vidéo et mentionne certaines techniques ou modèles qui n'ont pas été évoqués dans notre propre étude.

## 2 Détection d'objets

La détection d'objet a été introduite pour la première fois par Viola et Jones (2001). Leur idée de base était de réussir à détecter des visages ; pour cela, ils mirent au point un modèle de machine learning appelé « boost cascade » (figure 2.1).

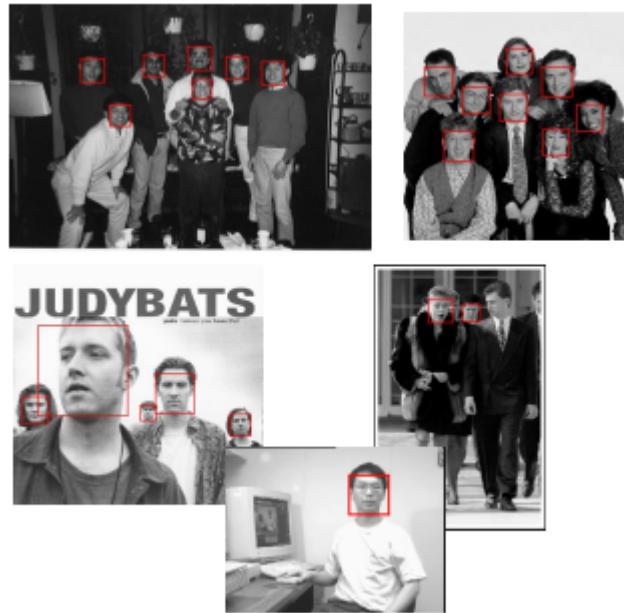


Fig. 2.1 : Boost Cascade (Viola et Jones, 2001)

Plus tard, d'autres technologies ont émergé comme les **Histogram of Oriented Gradients (HOG)** ou le **Deformable Parts Model (DPM)**, reposant toutes sur l'extraction de caractéristiques choisies à la main telles que les bords, les coins, les dégradés de l'image, couplée à des algorithmes d'apprentissage automatique non neuronaux.

Ce n'est qu'en septembre 2012 que le domaine franchit une nouvelle étape grâce à Krizhevsky, Sutskever et Hinton (2012), qui ont remporté la compétition ImageNet LSVRC-2012 avec leur réseau à convolution profond Alex-Net capable de reconnaître une grande quantité d'objet avec un taux d'erreur de 15,3%. En plus de gagner la compétition Imagenet, Alex et son équipe ont été les premiers à utiliser de l'apprentissage profond (deep learning) dans le but de réaliser de la classification d'image.

Cependant la détection d'objets est une problématique plus complexe que la classification d'images, car il ne suffit pas de reconnaître les objets présents, mais également de les localiser dans l'image.

## 2.1 RCNN / Fast RCNN et Faster RCNN

Pour résoudre ce problème de localisation, Girshick et al. (2013) ont proposé le modèle **Regional Convolutional Neuronal Network (RCNN)**, capable de reconnaître 80 types d'objets différents en utilisant une toute nouvelle technique fonctionnant

de la manière suivante : fragmenter l'image en des milliers de régions (sous-images), puis effectuer une classification pour chacune d'elles afin de détecter les objets présents (voir figure 2.2).

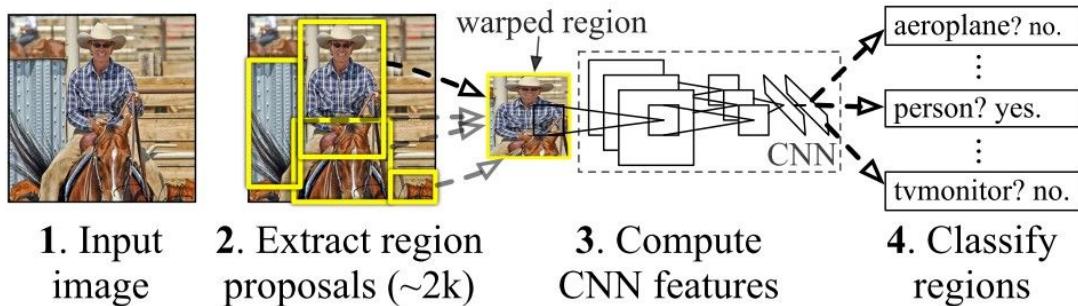


Fig. 2.2 : Fonctionnement de RCNN, Girshick et al. (2013)

Pour commencer, l'image d'entrée est découpée en 2000 sous-images appelées "régions d'intérêt", à l'aide de la méthode dite « Selective Search » (Uijlings et al., 2013). Cette méthode divise l'image d'entrée en plusieurs parties, et chaque partie voisine est comparée. Les parties qui ont des caractéristiques similaires au niveau de leur couleur, de leur texture ou de leur forme sont fusionnées afin de former ces dites régions. Chacune de ces régions est ensuite analysée par **RCNN**. Tout d'abord, elles passent dans un réseau de convolution pour extraire des caractéristiques clés. Puis, ces caractéristiques traversent un **SVM** ainsi qu'un modèle de régression pour identifier les objets présents dans ces différentes régions et concevoir les boîtes englobantes.

Bien que cette technique soit lente, car chaque image est découpée en 2000 régions avant d'être analysée par **RCNN**, elle n'en demeure pas moins efficace et sera adoptée par tous les modèles de détection d'objets à venir, dont **Fast-RCNN** ou encore **Faster-RCNN**, qui améliorent le modèle de base (Girshick, 2015 ; Ren et al., 2015).

Le principal apport de **Fast-RCNN** est que la recherche sélective ne s'effectue plus en amont sur l'image, mais plutôt sur les cartes de caractéristiques à l'aide d'une couche appelée « **RoI pooling layer** ». Chaque image traverse les couches de convolution afin de former des cartes de caractéristiques. Ces cartes sont ensuite envoyées à cette nouvelle couche pour y extraire des régions d'intérêt. Contrairement au modèle de base, le nombre de régions proposées n'est plus fixé, mais varie en fonction de l'image d'entrée. Ensuite, une couche de regroupement **RoI**

est utilisée pour redimensionner toutes les régions proposées, afin qu’elles puissent être introduites dans un réseau entièrement connecté pour être classées et détecter les boîtes englobantes de chaque objet.

Faster-**RCNN** quant à lui remplace cette méthode de selection par une couche spécifique appelée « region proposal network » (RPN), qui génère directement des régions d’intérêt. Cette couche utilise une approche de type fenêtre coulissante pour détecter les régions d’intérêt, ce qui permet de détecter les objets à différents échelles et tailles. Les régions d’intérêt sont ensuite classées par un réseau de couches entièrement connecté. Cette approche est plus rapide et plus efficace que les méthodes de sélection utilisées dans les versions antérieures.

## 2.2 YOLO

### 2.2.1 YOLOV1

En 2015-2016 Redmon, Divvala et al. (2015) publièrent l’algorithme **YOLO**, un nouveau modèle de détection d’objets qui marquera un tournant dans le domaine. Contrairement à **RCNN** et aux autres modèles traitant la détection comme un cas particulier de classification, **YOLO** a été pensé comme une tâche de régression.

De son nom, You Only Looks Once (en français « vous ne regardez qu’une seule fois »). **YOLO** commence par découper l’image en une grille de taille  $x * x$  puis, pour chacune de ces régions un nombre aléatoire de boîtes d’ancrage (anchors) vont être prédites (figure 2.3). Ces boîtes pourront donc avoir une taille et une position différentes selon le type d’objets recherchés. L’inconvénient est que via cette technique un même objet peut être détecté par plusieurs boîtes. Pour éviter cela, un calcul de **Non-Max Suppression (NMS)** sera ensuite réalisé pour conserver uniquement les boîtes englobantes maximisant la probabilité de chacun des objets trouvés (figure 2.4).

Pour chaque boîte englobante de la liste, nous récupérons celle ayant le meilleur score (taux de confiance) puis nous la comparons à toutes les autres boîtes pré-dites pour la même classe en utilisant le calcul **Intersection over Union (IoU)**. Ce calcul permet de connaître le degré de similitude / chevauchement entre deux boîtes. Chacune des boîtes ayant une similitude dépassant le seuil fixé par l’utilisateur (threshold) sera alors considérée comme représentant le même objet et par conséquent supprimée. De ce fait, **YOLO** est très rapide et s’est imposé comme un modèle de référence.

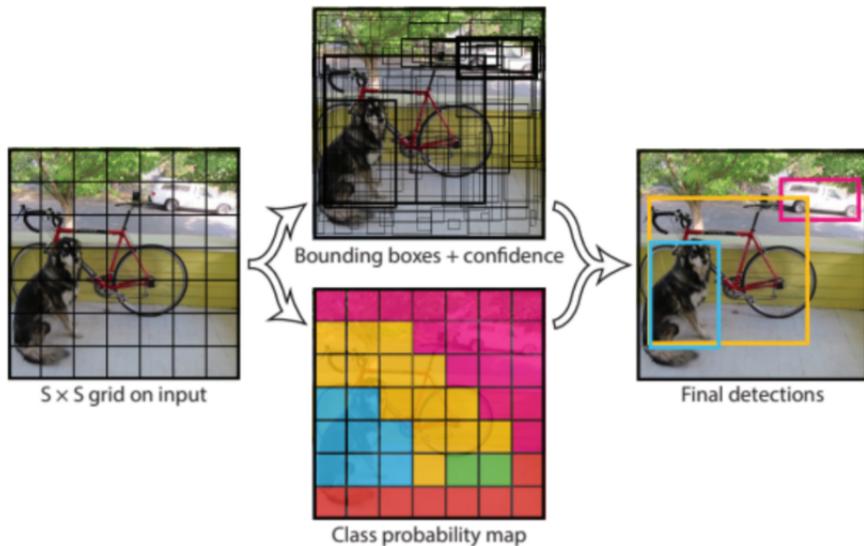


Fig. 2.3 : Fonctionnement de YOLO, Redmon, Divvala et al. (2015)

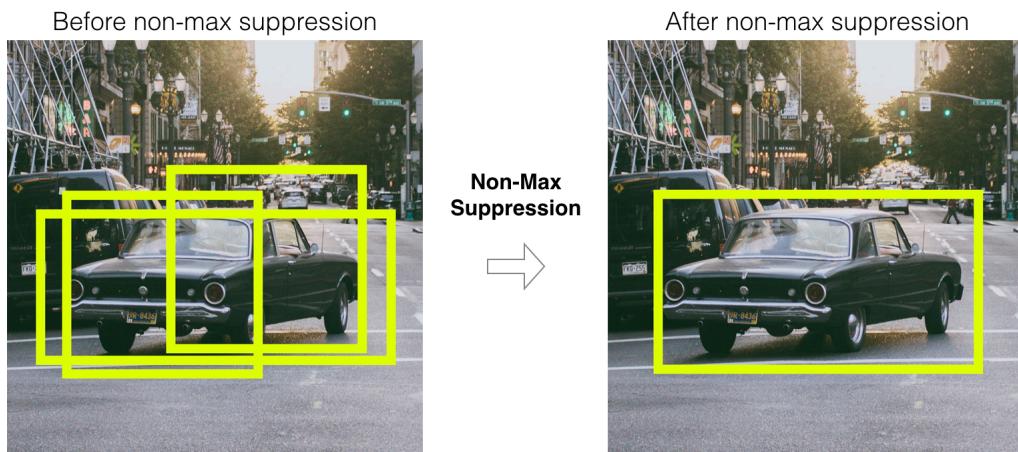


Fig. 2.4 : Non Maximal Suppression (NMS), Jain et Nandy (2019)

## 2.2.2 YOLOV2

Plus tard Redmon et Farhadi (2016) introduisent une nouvelle version de **YOLO**, appelée **YOLO9000** ou **YOLOV2**. Ce modèle est capable de reconnaître jusqu'à 9000 objets différents en temps réel grâce à une nouvelle architecture, Darknet-19.

Il s'agit d'un réseau à convolution composé de 19 couches convolutionnelles et 5 couches de max pooling, utilisant principalement des filtres de taille 3x3. Dans la première couche, il y a 32 filtres, qui sont ensuite doublés après chaque couche de max pooling. Cela signifie que le nombre de caractéristiques extraites augmente progressivement à mesure que les couches s'empilent, permettant au réseau de capturer des informations plus complexes au fur et à mesure de sa progression dans le modèle. Mis à part le changement d'architecture, la nouveauté ici est d'utiliser des boîtes d'ancrage définies selon le jeu de données de référence au lieu de les délimiter arbitrairement. Ceci permet au modèle d'avoir de meilleures performances, car généralement quelle que soit le contexte un même objet rentre plus ou moins dans le même moule.

### 2.2.3 YOLOV3

Redmon et Farhadi, 2018 introduisent leur dernière version, **YOLOV3**. Cette nouvelle version change une fois de plus l'architecture de base en intégrant une convolution composée de 53 couches entraînée sur IMAGENET et appelée Darknet-53. La fonction d'activation est aussi modifiée. Elle passe d'une activation softmax où l'on traite les classes de façon dépendante, avec un total de 100% pour l'ensemble des probabilités, à un classifieur logistique pour chaque classe ce qui permet un étiquetage multi-classe. En plus de cela, cette version effectue trois étapes de détection, ce qui permet au modèle d'être plus précis, meilleur pour reconnaître des petits objets et lui permet de mieux s'adapter à diverses tailles d'images. Malgré une petite perte de vitesse, cette nouvelle version gagne en efficacité et constitue une référence pour réaliser de la détection d'objets en temps réel. L'aspect multi-classe, qui est présent dans toutes les versions ultérieures, est intéressant en soi mais ne représente pas un intérêt particulier dans le cadre de cette thèse.

### 2.2.4 YOLOV4

Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao (2020), une nouvelle équipe, proposent **YOLOV4** en continuité avec la version précédente ; cette version est considérée par l'équipe originelle comme la suite logique de leurs travaux. L'architecture est désormais basée sur un modèle CSPDarknet53 inspiré des réseaux DenseNet. Dans les réseaux DenseNet introduits par Huang, Z. Liu et Weinberger (2016), chaque couche au sein d'un même bloc va transmettre sa sortie à toutes les couches suivantes du bloc ce qui a pour effet, de faciliter la rétro-propagation de l'erreur (figure 2.5).

Le réseau CSPDarknet53 se base sur un principe similaire mais utilise des couches CSPDenseNet qui n'envoient qu'une partie des caractéristiques en entrée

d'un bloc dense, et additionnent le reste directement à la sortie (figure 2.6). Un deuxième changement notable est celui de la fonction de perte maintenant appelé **Complete Intersection over Union (CIoU)** et prenant en considération l'IoU dans le calcul de l'erreur. Le dernier changement concerne le calcul de **NMS** permettant de filtrer les boîtes englobantes et ainsi éviter qu'un même objet soit détecté plusieurs fois. En effet, ce calcul a été remplacé par celui de **Distance Intersection over Union Non-Max Suppression (DIOU NMS)**, permettant de prendre en compte les centres des boîtes comparées, afin que deux objets similaires et proches ne soient pas considérés comme le même objet. Contrairement à la méthode précédente qui supprimait les boîtes englobantes prédites pour ne garder que la meilleure, cette nouvelle technique ajuste plutôt le taux de confiance de chaque boîte englobante. Du côté des ajouts, nous y retrouvons diverses techniques de data augmentation réalisées avant l'apprentissage du modèle (figure 2.7), ainsi qu'une méthode Drop-Block utilisée durant l'apprentissage, et supprimant la zone la plus représentative d'une image pour forcer le réseau à apprendre d'autres caractéristiques qu'il aurait sinon ignoré.

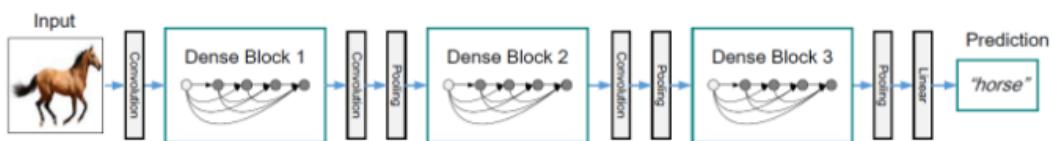


Fig. 2.5 : Architecture de DenseNet, Huang, Z. Liu et Weinberger (2016)

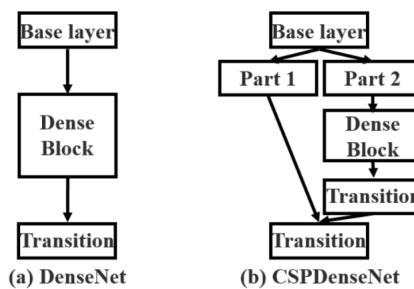


Fig. 2.6 : DenseNet vs CSPDenseNet, C.-Y. Wang, H.-Y. M. Liao et al. (2019)

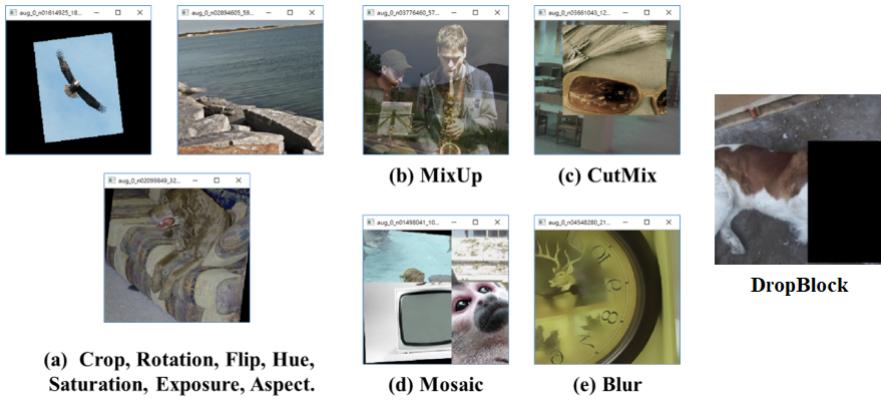


Fig. 2.7 : Data augmentation de YOLOV4, Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao (2020)

## 2.2.5 YOLOF, YOLOP, YOLOR, YOLOS, YOLOX

Plus tard en 2021, sont sorties de nombreuses variantes de **YOLO**, chacune traitant une problématique différente :

- YoloF (You Only Look One-level Feature) architecture basée sur le modèle Resnet couplé à des encodeurs [figure 2.8].

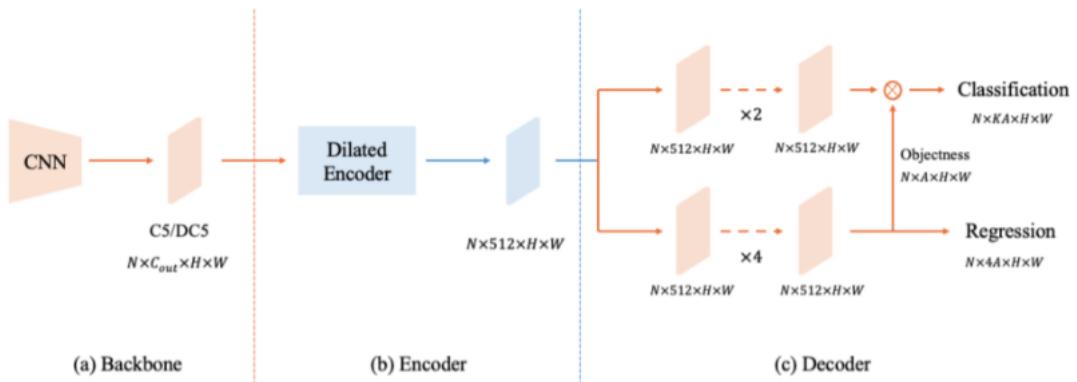


Image 17 – Architecture YOLOF

Fig. 2.8 : Architecture de YOLOF, Q. Chen et al. (2021)

- YOLOP (You Only Look Once for Panoptic Driving Perception) spécialisé pour la conduite autonome (figure 2.9).

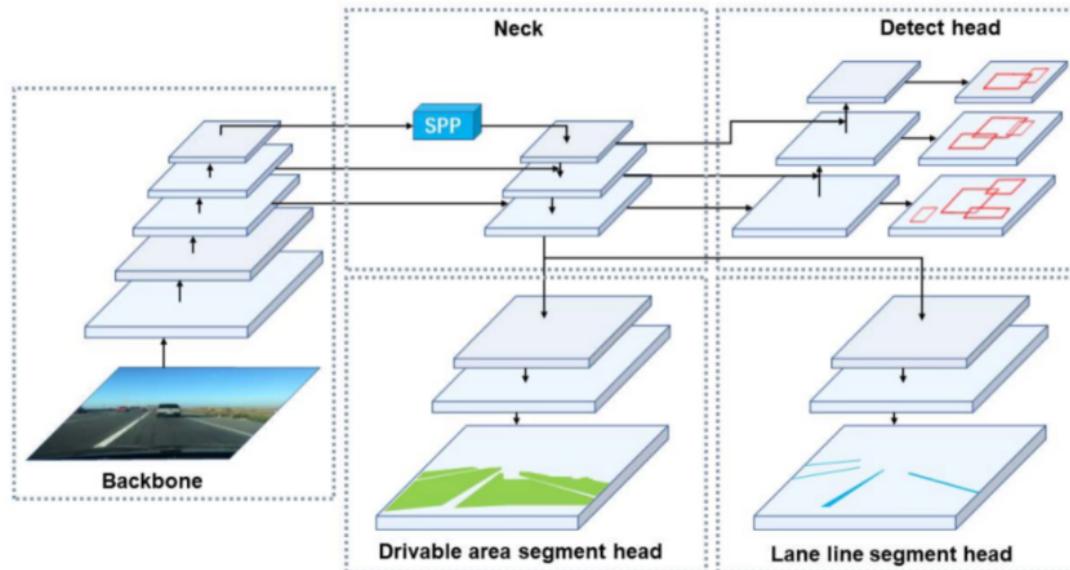
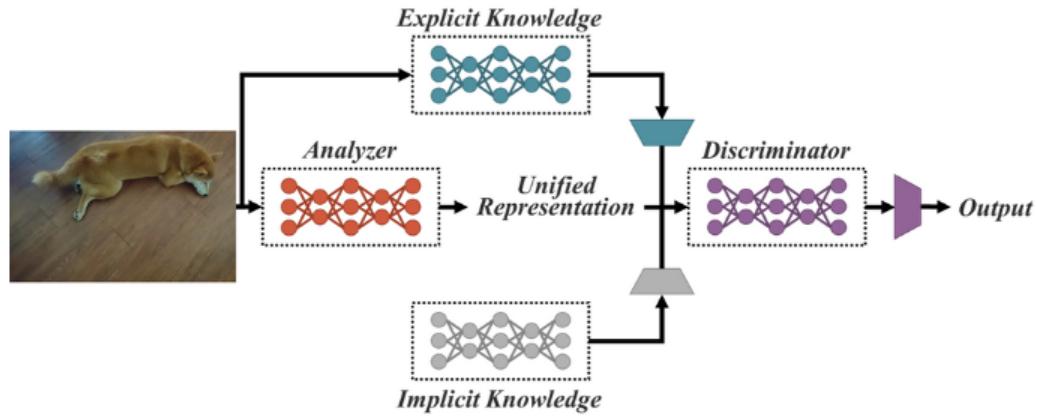


Image 13 – Architecture YOLOP

Fig. 2.9 : Architecture de YOLOP, D. Wu et al. (2021)

- YOLOR (You Only Learn One Representation) utilisant à la fois des connaissances explicites (résultant de son apprentissage) ainsi que des connaissances implicites. Grâce aux connaissances implicites acquises, ce modèle est très performant pour de l'apprentissage multitâche, il peut donc être utilisé pour d'autres tâches d'analyse d'images [figure 2.10].



Le réseau unifié de YOLOR : combiner les connaissances explicites et les connaissances implicites pour servir plusieurs tâches

Fig. 2.10 : Architecture de YOLOR, C.-Y. Wang, Yeh et H. Liao (2021)

- YOLOS (You Only Look at One Sequence) (Fang et al., 2021) est une variante de YOLO qui utilise des Transformers à la place des réseaux à convolution (voir la figure 2.11). Les transformers sont des réseaux ayant comme avantage de posséder un mécanisme d'attention, ce qui leur permet de détecter les zones importantes d'une image pouvant être représentées à l'aide de carte d'attention. Pour le moment, cette variante n'est pas optimisée. Les auteurs se sont simplement fixés pour objectif de montrer qu'il était possible d'utiliser des transformers pour effectuer de la détection d'objets.

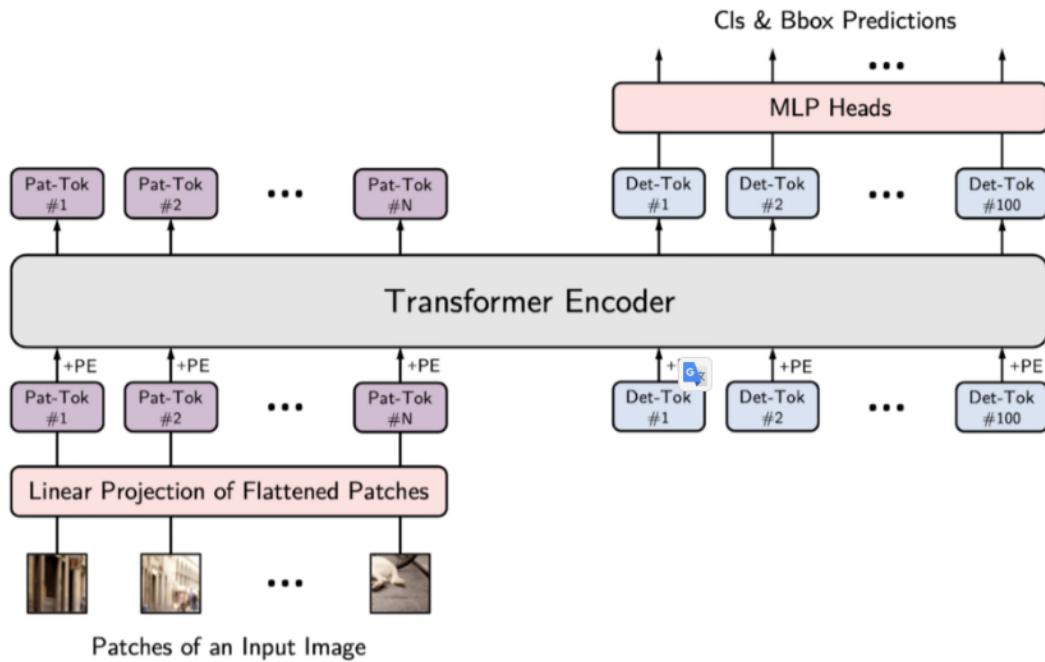


Image 18 – Architecture YOLOS

Fig. 2.11 : Architecture de YOLOS (Fang et al., 2021)

- (Ge et al., 2021) introduisent YOLOX, une variante de YOLOV3 n'utilisant aucune boîte d'ancrage, ce qui permet une plus grande rapidité et une plus grande précision.

## 2.2.6 YOLOV5

Quelques mois après la sortie de YOLOV4, une version 5 voit le jour<sup>1</sup> ; mais elle n'a fait l'objet d'aucune publication officielle. Cette version n'implémente ni n'invente de nouvelles techniques, elle est simplement l'extension PyTorch de YOLOV3 et non une continuité du code original. Elle a pour principal objectif de rendre YOLO compatible avec macOS et de proposer cinq versions différentes du modèle [figure 2.12].

1. <https://github.com/ultralytics/yolov5>

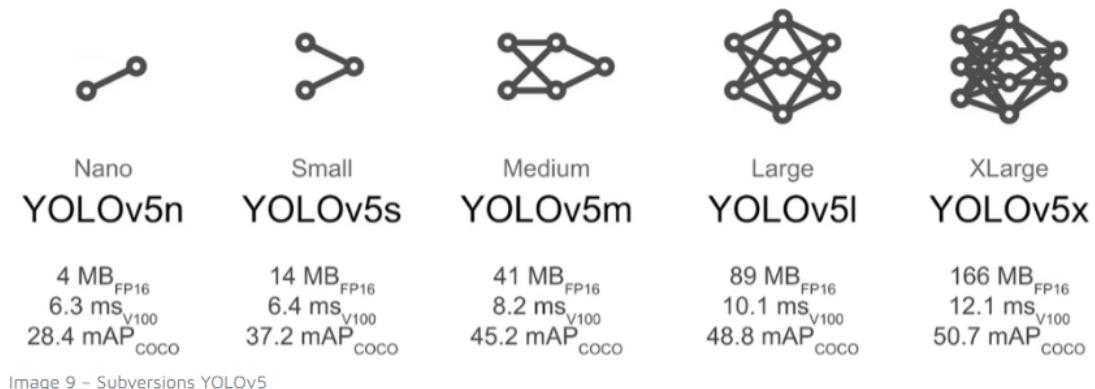


Fig. 2.12 : Architectures proposées par YOLOV5

### 2.2.7 YOLOV6

Récemment, la société chinoise Meituan a sorti MT-YOLOV6<sup>1</sup>, un algorithme ne faisant pas officiellement partie de la série YOLO mais fortement inspiré du YOLO originel. Elle n'a aucun rapport avec la version 5, mais elle la surpasse largement en termes de précision et vitesse.

### 2.2.8 YOLOV7

C.-Y. Wang, Bochkovskiy et H.-Y. M. Liao, 2022 proposent une version 7, développée par les auteurs de la version 4, dont l'architecture est basée sur YOLOR. Elle serait 120% plus rapide que YOLOV5 et plus performante que toutes les autres architectures disponibles y compris YOLOR version antérieurement la plus performante (figure 2.13). La figure 2.14 montre différentes applications qui ont été permises par cette version.

1. <https://github.com/meituan/YOLOv6>

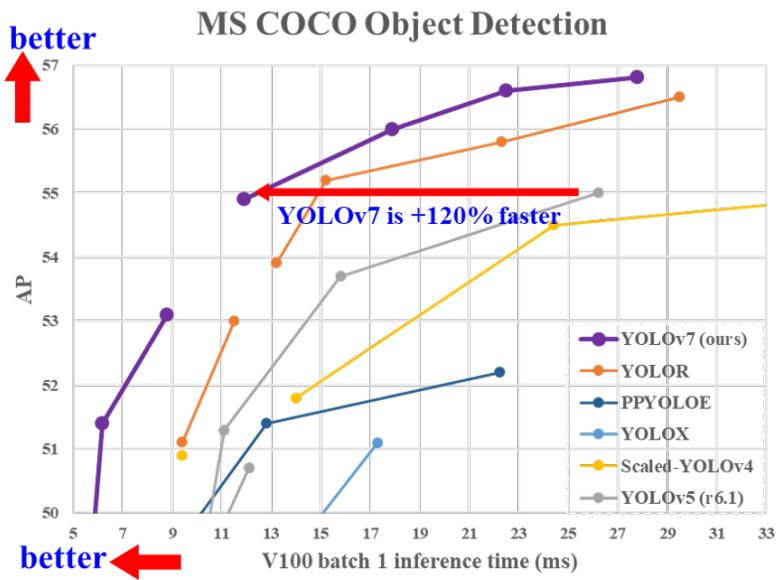


Fig. 2.13 : Performances de YOLOV7, C.-Y. Wang, Bochkovskiy et H.-Y. M. Liao (2022)

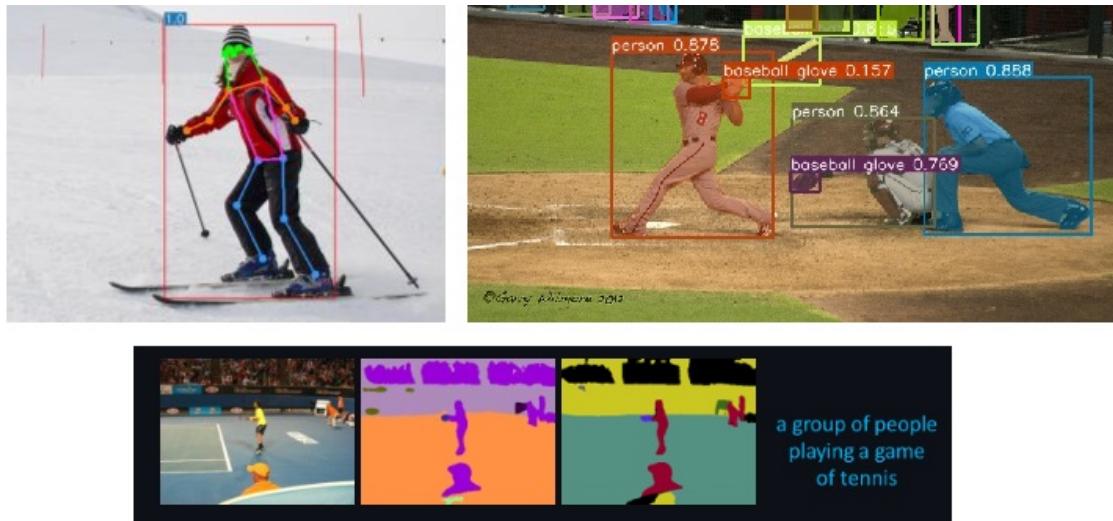


Fig. 2.14 : Applications de YOLOV7, C.-Y. Wang, Bochkovskiy et H.-Y. M. Liao (2022)

On retrouvera plus de détails sur le fonctionnement de ces modèles dans Sharma

(2018c), Sharma (2018b) et Sharma (2018a). Zou et al. (2019) retracent l'évolution du domaine durant ces vingt dernières années, résumé dans la figure 2.15.

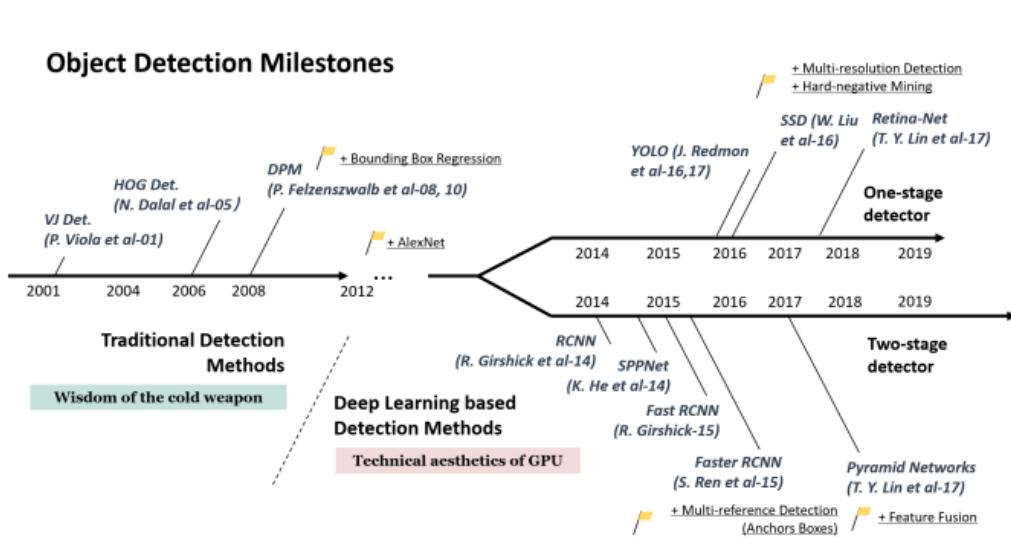


Fig. 2.15 : Évolution de la détection d'objet (Zou et al., 2019)

### 3 Explicabilité

Les techniques qui améliorent l'explicabilité dans le domaine visuel appartiennent à deux catégories :

D'un côté, nous retrouvons des technologies indépendantes du type de modèle utilisé tel que LIME (Local Interpretable Model-Agnostic Explanations) proposé par M. T. Ribeiro, Singh et Guestrin (2016), ou encore SHAP (SHapley Additive exPlanations) proposé par Lundberg et Lee (2017).

De l'autre, nous avons des technologies spécifiques à certains types de modèle comme les réseaux à convolution, pour lesquels nous retrouvons des techniques de visualisation de filtre de convolution (Jiang et al., 2021), de carte de saillance (Smilkov et al., 2017 ; Simonyan, Vedaldi et Zisserman, 2013), de cartes d'activation (Selvaraju et al., 2016 ; Chattpadhyay et al., 2017 ; H. Wang et al., 2019) etc..

Pour visualiser ces caractéristiques, il existe de nombreuses bibliothèques.

Kotikalapudi et al. (2017) proposent *Keras-vis*, une bibliothèque publique qui permet de visualiser les caractéristiques des réseaux de neurones convolutifs. Elle permet notamment de visualiser les filtres de convolution de chaque couche, c'est-

à-dire les matrices de poids qui permettent de détecter des motifs dans l'image. Ces filtres peuvent être visualisés sous forme de matrices de pixels, ce qui permet de mieux comprendre les motifs auxquels chaque filtre est sensible.

En outre, Keras-vis permet également de visualiser l'évolution de ces filtres au fil de l'apprentissage. Cela peut être utile pour comprendre comment le réseau apprend à extraire des caractéristiques de plus en plus complexes au fil du temps.

Enfin, la bibliothèque permet également de visualiser des cartes d'activation, c'est-à-dire les zones de l'image qui ont été activées par chaque filtre. Ces cartes peuvent aider à comprendre ce que le réseau « voit » dans l'image à chaque étape du traitement, ce qui peut être utile pour diagnostiquer d'éventuels problèmes de fonctionnement du réseau.

Plus tard Remy (2020) développe la bibliothèque Keract avec des fonctionnalités similaires. La même année Gotkowski et al. (2020) proposèrent une autre bibliothèque permettant de visualiser des cartes d'attention aussi bien 2D que 3D. La bibliothèque Keras, développée par Chollet et al. (2015), inclut aussi certaines de ces techniques de visualisation.

On se référera à Molnar (2019) pour une étude plus approfondie des techniques d'explicabilité.

## 4 Conclusion

Au cours de cette étude, différentes technologies ont été identifiées en vue de la détection d'anomalies, comprenant entre autres les Convolutional RNN, les ConvRNN, les autoencodeurs, les Transformers, les TCN et les GAN. Afin de mettre en lumière nos choix stratégiques et de fournir une base solide à notre démarche, un comparatif exposant les avantages et les inconvénients de chacune de ces options est présenté dans le tableau 2.1.

Tab. 2.1 : Comparaison de différentes techniques de détection d'anomalies

	CNN + RNN	ConvRNN	C3D	TCN	GAN	Auto-encodeur	Transformer
Documentation	Élevée	Élevée	Élevée	Faible	Élevée	Élevée	Élevée
Mise en place	Simple	Simple	Moyenne	Difficile	Difficile	Difficile	Difficile
Puissance de calcul	Basse	Basse	Modérée	Modérée	Énorme	Variable	Énorme
Mode	Supervisé	Supervisé	Supervisé	Supervisé	Non supervisé	Non supervisé	Supervisé
Vitesse	Rapide	Rapide	Rapide	Lente	Lente	Lente	Lente

En raison de nos contraintes en termes de puissance de calcul, causées par l'indisponibilité de serveurs GPU, nous avons été contraints de mettre de côté les GAN ainsi que les Transformers. De plus, dû à notre impératif de temps réel, l'utilisation des autoencodeurs a également été écartée. En effet, leur besoin de

reconstruire chaque image de nos vidéos aurait entraîné des délais considérables. A cause de leur manque de documentation et de leur efficacité jugée insuffisante pour répondre à nos besoins spécifiques en matière de détection d'anomalies dans les vidéos, les **TCN** ont également été écartés.

Après avoir analysé les ensembles de données disponibles pour aborder notre problématique, on constate que la très large majorité d'entre eux est orientée vers des approches supervisées, opposant classe normale et anomalies. Dans cette optique, les Convolutional **RNN**, les ConvRNN (pouvant être associés à des **GRU** ou des **LSTM** pour la partie **RNN**) ainsi que les convolutions 3D ont émergé comme les choix les plus pertinents et adéquats pour notre démarche.

En ce qui concerne la détection d'objets, nous pouvons envisager plusieurs modèles tels que **RCNN**, **Fast-RCNN**, **Faster-RCNN** et **YOLO**. Les spécificités de chaque modèle sont synthétisées dans le tableau 2.1, permettant ainsi une comparaison approfondie de leurs caractéristiques respectives.

Tab. 2.2 : Comparaison des diverses techniques de détection d'objets évoquées

	<b>RCNN</b>	<b>Fast-RCNN</b>	<b>Faster-RCNN</b>	<b>YOLO</b>
Documentation	élevée	élevée	élevée	élevée
Mise en place	simple	simple	simple	simple
Puissance de calcul	modérée	modérée	modérée	modérée
Vitesse	lente	lente	rapide	très rapide

Cependant, compte tenu de la contrainte de temps réel due aux répercussions que peuvent avoir les anomalies auxquelles nous nous intéressons, les plus rapides sont à privilégier. Il est vrai que **Faster-RCNN** est assez rapide et permet d'avoir de très bons résultats, mais pour des problématiques de temps réel, les modèles réalisant une analyse en une seule étape tels que **YOLO** sont de meilleurs candidats (Gandhi, 2018). Selon Doshi et Yilmaz (2020), **YOLO** serait le meilleur algorithme du groupe. Pour les citer : « Par rapport à d'autres modèles de pointe tels que SSD et ResNet, **YOLO** offre un traitement d'images par seconde (**FPS**) plus élevé tout en offrant de meilleures précisions. Pour la détection d'anomalies en ligne, la vitesse est un facteur critique, et donc nous préférons actuellement utiliser **YOLO V7** ». Ajoutons que **YOLO** a connu de nombreuses améliorations significatives ces dernières années, notamment en termes de précision et de vitesse de traitement.

Avec ses différentes versions, **YOLO** est le modèle le plus rapide disponible à ce jour. Au moment de la finalisation de cette thèse, une version 8 est apparue, que nous n'avons pas eu l'occasion de tester.

## **Deuxième partie**

### **Système réalisé**



# Table des matières

---

<b>3 Système</b>	<b>69</b>
1 Introduction . . . . .	70
2 Architecture globale . . . . .	70
3 Composant d'analyse spatiale . . . . .	71
4 Analyse temporelle . . . . .	73
5 Composant de correction . . . . .	76
6 Explicabilité . . . . .	77
7 Fonctionnement . . . . .	79
8 Conclusion . . . . .	83
<b>4 Expérimentations et résultats</b>	<b>85</b>
1 Introduction . . . . .	86
2 Prétraitement des données . . . . .	86
3 Expérimentations en mode série . . . . .	93
4 Fonctionnement en mode parallèle . . . . .	100
5 Mode traçage (tracabilité des résultats) . . . . .	106
6 Synthèse . . . . .	112
7 Analyse temporelle : la détection d'anomalie . . . . .	114
8 Analyse spatiale : la détection d'objet . . . . .	131
9 Conclusion . . . . .	136

---



# Chapitre 3

## Système

### Sommaire

---

<b>1</b>	<b>Introduction</b>	70
<b>2</b>	<b>Architecture globale</b>	70
<b>3</b>	<b>Composant d'analyse spatiale</b>	71
<b>3.1</b>	Images	72
<b>4</b>	<b>Analyse temporelle</b>	73
<b>4.1</b>	Extraction de caractéristiques spatiales	74
<b>4.2</b>	Détection	74
<b>4.3</b>	Vidéos	75
<b>5</b>	<b>Composant de correction</b>	76
<b>6</b>	<b>Explicabilité</b>	77
<b>7</b>	<b>Fonctionnement</b>	79
<b>8</b>	<b>Conclusion</b>	83

---

## 1 Introduction

Dans ce chapitre, nous allons présenter nos travaux et contributions. Nous commencerons par exposer l'architecture globale de notre système, ses composants, ainsi que les jeux de données créés pour la phase d'apprentissage. Nous poursuivrons en expliquant comment les expérimentations menées et les résultats obtenus ont influencé nos choix et nos approches. Enfin, nous conclurons ce chapitre en réalisant une synthèse de nos travaux.

## 2 Architecture globale

Dans la section 3, nous avons présenté notre objectif, qui consiste à développer un système de détection d'anomalies susceptibles de représenter un risque pour les personnes, en combinant l'analyse spatiale effectuée sur des images (incluant la détection d'objets) avec l'analyse temporelle d'un flux vidéo.

La figure 2.1 illustre le principe général de notre système, qui comporte deux parcours possibles. Le premier parcours, représenté en rouge, suit une disposition en série où le composant temporel prend en entrée la sortie du composant spatial. Le deuxième parcours, représenté en vert, suit une disposition en parallèle où chaque séquence d'images est traitée simultanément par les deux composants, avant d'être classée en combinant les détections de chacun. Enfin, notre système intègre un composant d'explicabilité nous permettant de mettre en valeur les zones d'attention ayant conduit à la détection du modèle. Ce composant peut être désactivé pour gagner en vitesse.

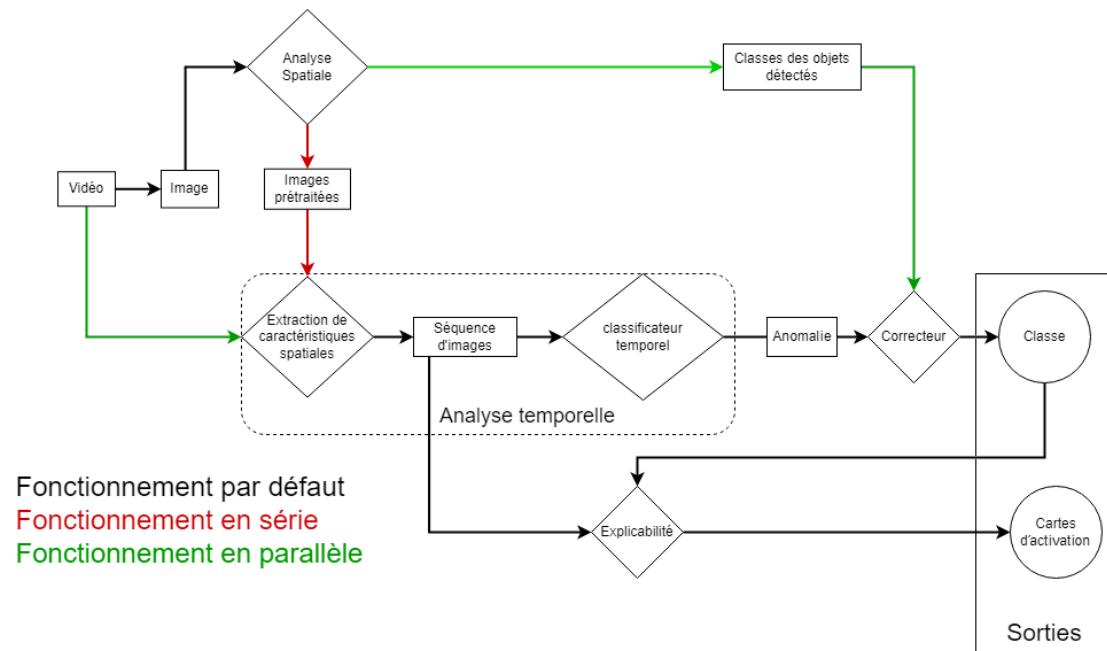


Fig. 2.1 : Architecture globale

Dans les sections suivantes nous expliquerons chacun des composants de la figure ci-dessus (les losanges), avant de récapituler le fonctionnement global.

### 3 Composant d'analyse spatiale

Le premier élément de notre système réalise une analyse spatiale de chaque image incluse dans la vidéo à traiter. En entrée il reçoit une image, en sortie il transmet à deux composants différents en fonction du mode de fonctionnement choisi : en mode parallèle, il transmet simplement la classe des objets détecté au composant Correction ; en mode série, il transmet l'image prétraitée au composant Extraction de caractéristiques spatiales. Selon le modèle choisi, ce prétraitement peut inclure de la détection d'objets, de la segmentation d'images ou de l'analyse de la pose humaine.

Nous avons choisi **YOLO**, pour les raisons évoquées dans la problématique (vitesse de traitement et performance, voir section 4), et aussi parce que cette famille de modèles fait l'objet de constantes recherches et améliorations par la communauté scientifique. Dans nos tests préalables nous avons comparé **YOLO** (à l'époque la version 3) avec Faster-**RCNN**, le plus rapide des autres modèles, pour constater qu'ils avaient un temps d'exécution similaire. Mais la version 7 de

**YOLO** les dépasse de très loin en rapidité. Dans cette étude nous avons comparé les performances des versions 3, 4 et 7. Les détails des tests et des résultats peuvent être trouvés dans la section [8](#).

Ce modèle est pré-entraîné sur les jeux de données Pascal-VOC<sup>1</sup> et COCO<sup>2</sup>, pour être capable de reconnaître 80 types d'objets parmi lesquels on retrouve : l'être humain, des véhicules (camion, vélo, moto, bus, voiture, avions, train, bateaux), des animaux (chien, chat, girafe...), de la nourriture (pizza, banane...) ainsi que des objets du quotidien (téléphone, télévision, chaise...). Parmi toutes ces classes, très peu sont utiles pour notre problématique. Seules les personnes et les véhicules peuvent être exploités dans des cas d'accidents ou de bagarres. De plus, des objets cruciaux pour notre problème, tels que des armes blanches, des pistolets ou encore des flammes, nous permettant de détecter de potentiels coups de feu ou incendies, n'y sont pas inclus.

Il a donc été essentiel de constituer notre propre jeu de données afin de ré-entraîner **YOLO**, c'est ce jeu que décrit la sous-section suivante.

### 3.1 Images

Notre jeu de données est propriétaire et combine des images extraites de vidéos utilisés dans le cadre de notre analyse temporelle ainsi que des images que nous avons nous-mêmes collectées et labellisées. Contrairement à ce que l'on pourrait penser, les images utilisées ne doivent pas nécessairement être réalistes, et il n'est pas obligatoire que ces images proviennent exclusivement de vidéos de surveillance.

Nous avons collecté plus de dix mille images représentant des armes à feu. Nous les avons d'abord divisées en deux classes : « gun » pour les petites armes et « weapons » pour les plus grandes. Cependant, toute arme représente un risque potentiel indépendamment de sa taille ou de son apparence. Par conséquent, nous avons finalement décidé de les regrouper en une seule classe, et avons constaté une amélioration des performances de notre système.

Nous avons constitué une classe d'images nous permettant d'identifier des flammes, ce qui est important dans le cas d'incendies. Nous avions également constitué une classe représentant des armes blanches, mais nous avons finalement décidé de ne pas l'inclure dans notre jeu de données final. Cette classe contenait des objets très différents et l'action en elle-même ressemblait davantage à une agression ou à une bagarre.

À l'heure actuelle, notre jeu de données contient donc trois classes :

- 
1. <https://pjreddie.com/projects/pascal-voc-dataset-mirror/>
  2. <https://cocodataset.org/#home>

1. armes à feu : environ 10 000 images,
2. flammes : plus de 2000 images,
3. personnes (représentant chacun des individus présents dans nos images).

Des images dans lesquelles aucune de ces classes n'est présente ont également été ajoutées à notre jeu de données, tant pour l'apprentissage que pour les tests.

Ce jeu de données est enrichi grâce aux techniques de data augmentation incluses dans **YOLO**, déjà mentionnées précédemment (figure 2.7). Nous avons utilisé plus de dix techniques différentes applicables à chacune de nos images, ce qui nous permet d'aboutir à près de 150 000 images en tout.

## 4 Analyse temporelle

Pour les raisons indiquées dans la section 4 de l'état de l'art, nous avons, pour l'analyse temporelle, choisi des approches bien documentées, simples à mettre en place et assez légères en ressources : les Convolutional RNN, ConvRNN et les convolutions 3D. Après avoir effectué des tests détaillés dans la section 7, nous avons finalement opté pour une architecture basée sur un Convolutional GRU (CGRU) détaillée figure 4.1.

Ce traitement est effectué par deux composants distincts : une convolution (extracteur de caractéristiques spatiales) et un **GRU** (classifieur temporel) que nous examinerons ci-dessous.

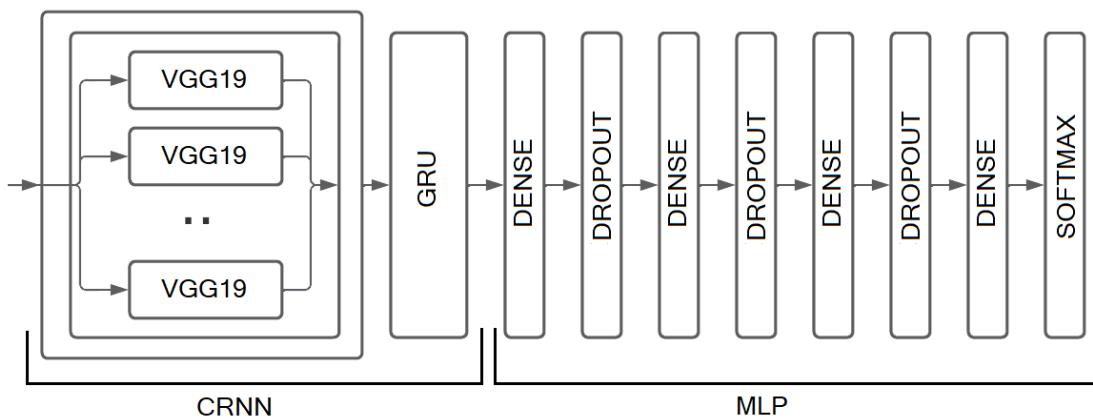


Fig. 4.1 : Architecture du module d'analyse temporelle (CGRU)

## 4.1 Extraction de caractéristiques spatiales

Le deuxième composant de notre système est consacré à l'extraction des caractéristiques spatiales ; il reçoit une séquence vidéo et renvoie cette séquence après mise en valeur de ses caractéristiques. En mode série, la séquence qu'il reçoit en entrée correspondent aux images d'origine modifiées par **YOLO**, en mode parallèle il reçoit simplement la vidéo d'entrée. La sortie est transmise au composant qui effectue l'extraction de caractéristiques temporelles et (facultativement) au module d'explicabilité.

Pour l'extraction des caractéristiques spatiales, nous avons choisi **VGG19**. Cependant, comme cette convolution est 2D, elle n'est pas adaptée au traitement des séquences d'images. Si nous l'appliquons directement sur nos données, nous perdrons de l'information car toutes les images de notre séquence devront être fusionnées pour tenir sur un format 2D. C'est pourquoi nous l'avons inclus dans une couche time distributed fournie par la bibliothèque Keras. Cette couche nous permet d'appliquer une ou plusieurs couches à chaque tranche temporelle de notre entrée.

Dans notre cas, appliquer **VGG** à chaque image composant notre séquence afin d'accumuler les caractéristiques provenant de différentes images et former notre séquence à analyser. Cette couche time distributed reçoit en entrée une séquence vidéo, extrait chaque image et la transmet au réseau **VGG** pour l'extraction des caractéristiques pertinentes. Une fois que ces caractéristiques sont extraites pour chaque image, nous obtenons en sortie une nouvelle série temporelle qui sera transmise au composant suivant.

## 4.2 Détection

Le composant suivant contient deux parties : un composant d'extraction de caractéristiques temporelles, appliqué sur la sortie de la couche time distributed, et un composant de classification, qui effectue la détection du type d'alerte à associer à la séquence.

L'extraction de caractéristiques temporelles est assurée par un **GRU** suivi de quelques couches totalement connectées ainsi que quelques couches d'oubli permettant d'éviter le sur-apprentissage.

Chaque séquence transmise par la couche time distributed va passer par la porte d'oubli de notre **GRU**, porte qui va permettre de contrôler le nombre d'informations qui doit être oublié. Les informations gardées, jugées pertinentes, seront données à la porte de mise à jour pour être apprises. Cette porte a pour rôle de concaténer les nouvelles données avec celles de l'état précédent puis, de les passer par une fonction sigmoïde pour détecter les caractéristiques importantes.

Après être passées par la porte de mise à jour, les informations jugées importantes sont ensuite traitées par les couches entièrement connectées de notre classificateur, un **MLP**, afin d'établir la détection finale.

### 4.3 Vidéos

Pour les raisons indiquées dans la problématique (voir page 29), nous avons décidé de créer notre propre jeu de données vidéo dans le but d'entrainer notre **CGRU**. Ce jeu est indépendant du jeu de données d'images que nous avons constitué pour l'apprentissage des objets.

Ce jeu de données est propriétaire et suit le modèle établi par UCF Crimes, c'est-à-dire opposer nos anomalies à la classe normale. Cependant, comme la collecte de données est une tâche longue, nous nous sommes limités à trois classes principales : la bagarre, les coups de feu et les incendies. De plus, certaines de ces catégories ont été plus difficiles à collecter que d'autres en raison de la rareté de ces événements ou de la censure. Contrairement à la classe normale, où toute vidéo quotidienne peut être considérée comme représentative, les anomalies ont des fréquences d'apparition différentes, ce qui peut affecter la quantité de données disponibles. Pour garantir un équilibre maximal, nous avons collecté autant de vidéos représentant une anomalie que de cas normaux.

Pour savoir si notre jeu de données peut être considéré comme fiable, il faut vérifier si notre système est capable d'obtenir de bonnes performances après l'apprentissage. Nous avons donc séparé notre jeu de données en sous-catégories binaires, opposant à chaque fois une anomalie à la classe normale, ce qui permet d'entraîner des modèles en utilisant uniquement certaines classes, ainsi que d'évaluer la fiabilité de chacune d'elles.

Notre jeu de données, décrit dans les tableaux 3.1 et 3.2, est constitué de trois classes représentant une vaste sélection de vidéos provenant de caméras de surveillance ou de smartphones. Ces vidéos ont des qualités et des résolutions différentes, couvrent divers angles de vue et sont enregistrées à différents moments de la journée. La figure 4.2 montre leur répartition par classe.

Tab. 3.1 : Répartition des vidéos de notre jeu de données

Classes	Train	Validation	Total
Bagarre	587	391	978
Coup de feu	247	64	311
Incendie	237	61	298

Tab. 3.2 : Durée des anomalies de notre jeu de données

Classes	Train				Validation			
	Min	Moyenne	Max	Total	Min	Moyenne	Max	Total
Bagarre	1.2s	5.1s	140s	50.05min	1s	4.8s	139s	31.50min
Coup de feu	1s	4.5s	16s	17.40min	1s	36.5s	36s	8min
Incendie	1.3s	51.3s	504s	3h40min	1.4s	49.8s	211s	46min

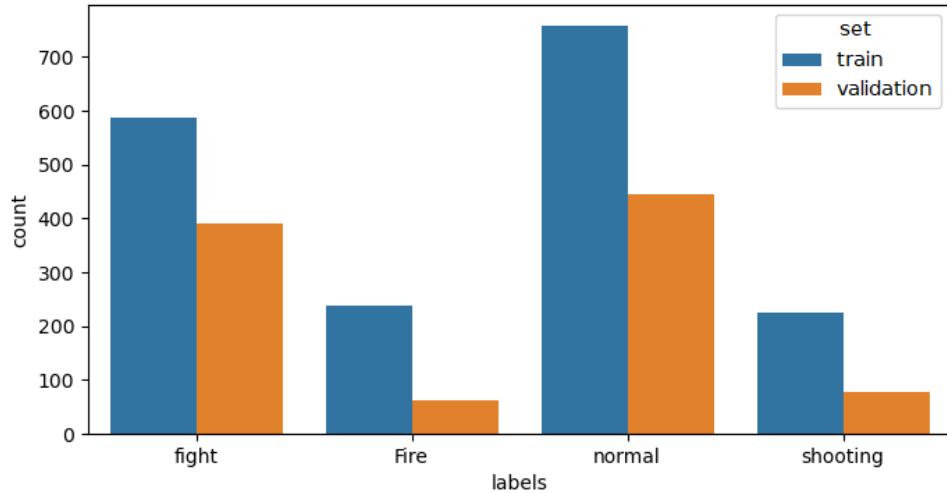


Fig. 4.2 : Répartition de nos données

## 5 Composant de correction

Le quatrième composant de notre système est utilisé pour combiner, quand on est en mode parallèle, les résultats de la détection d'objet avec ceux de l'analyse temporelle. Son rôle est de corriger la classe prédictive en sortie en se basant sur les résultats de l'analyse spatiale réalisée par **YOLO**.

Pour ce faire, nous avons constitué un dictionnaire où chaque type d'objet appris par **YOLO** est lié à une anomalie : les flammes avec l'anomalie "incendie" et les armes à feu avec l'anomalie "coup de feu".

Une fois que la classe a été prédictive par notre modèle **CGRU**, nous analysons les objets détectés par **YOLO** afin de vérifier si celui-ci n'a pas détecté une flamme ou une arme à feu dont la probabilité dépasse le taux de confiance fixé.

Si une flamme est détectée et que le modèle **CGRU** a prédit la classe "normal", nous changeons celle-ci en faveur de la classe "incendie" en fonction de la probabilité de la détection. Dans le cas où une arme à feu est détectée, nous avons conclu que celle-ci ne représentait un risque que dans les cas où elle pouvait être manipulée par une personne. Nous calculons donc l'**IoU** entre cette arme et toutes les personnes détectées dont le score dépasse le seuil de confiance fixé pour savoir si une personne est en contact avec cette arme. Si c'est le cas, nous corrigeons la classe prédite en la changeant pour la classe "coup de feu".

## 6 Explicabilité

L'Union Européenne définit l'explicabilité comme la capacité à comprendre et expliquer comment une décision a été prise par un système automatisé. Dans cette optique, notre module d'explicabilité vise à justifier les décisions prises par les composants de notre système de détection d'anomalies, en particulier pour faciliter la compréhension des décisions de notre modèle d'analyse temporelle (**CGRU**), et plus précisément, de notre convolution **VGG** travaillant avec des images.

En termes d'explicabilité, il est bien plus simple d'interpréter les caractéristiques apprises par nos convolutions étant donné que celles-ci seront visuelles contrairement à celles de notre **RNN**. À première vue, il semblerait qu'aucune des bibliothèques de visualisation propre au **CNN** ne soit adaptée à notre type de réseau, pour les raisons détaillées ci-dessous (voir figure 6.1).

Nous nous sommes donc inspirés des techniques utilisées pour expliquer la classification d'image. Le principe est de mettre en valeur visuellement la partie de l'image qui a provoqué la prise de décision, adapté à la vidéo.

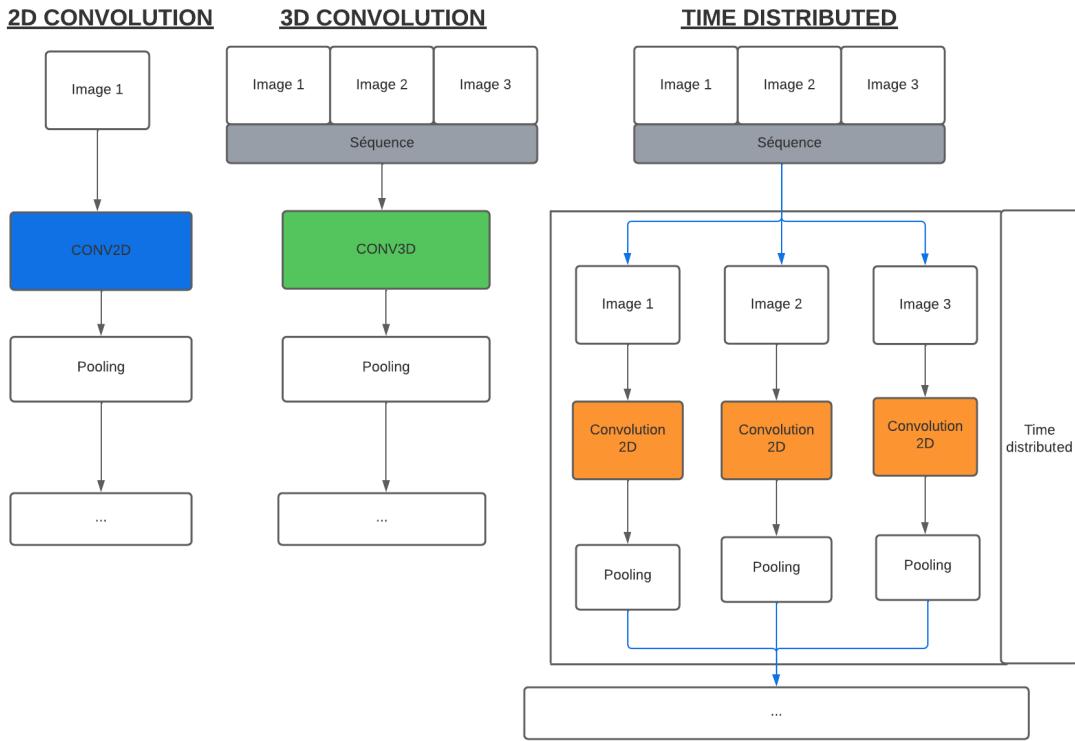


Fig. 6.1 : Convolution 2D / convolution 3D / Convolution incluse dans une couche time distributed

Dans le cas d'un modèle utilisant des couches de convolutions 2D ou 3D, toutes les couches du réseau sont directement connectées. Or, dans notre architecture, nous utilisons une convolution accessible à travers une couche time distributed, qui n'est donc pas directement connectée aux autres couches du réseau, ce qui bloque la propagation de l'information vers les autres couches.

Le deuxième problème concerne la troisième dimension ajoutée par le facteur temps. Contrairement aux modèles traitant des images ou des objets 3D pour lesquels nous aurions une seule donnée en entrée et donc une seule visualisation en sortie, ici nous traitons des vidéos. Nous avons donc plusieurs images en entrée mais une seule visualisation en sortie, représentant l'intégralité de notre séquence d'entrée.

Nous avons donc effectué une visualisation pour chacune de nos images afin de comprendre le traitement réalisé par notre sous-modèle. Il s'agira de visualiser les zones sur lesquelles le modèle s'appuie afin de détecter une anomalie. Pour cela, nous utiliserons principalement des cartes de saillance (saliency) ainsi que

des cartes d'activation.

Pour réaliser ce type de visualisation, nous devrons propager une information à travers notre réseau afin d'obtenir l'activation finale. Cela nous permettra de créer d'une part des cartes de saillance en calculant le gradient de cette activation par rapport à notre donnée d'entrée, d'autre part des cartes d'activation en calculant le gradient de cette activation par rapport à la sortie de la couche que nous souhaitons visualiser.

Pour réaliser les cartes de saillance, il nous faudra calculer notre gradient à partir d'une séquence et non d'une image. Heureusement, le résultat sera de même dimension que la donnée d'entrée, c'est-à-dire que nous obtiendrons une liste de gradients égale à la taille de notre séquence. Il nous suffira alors d'afficher ces différents gradients pour obtenir une carte de saillance pour chaque image.

En ce qui concerne les cartes d'activation, notre seule solution sera d'utiliser la sortie de la couche time distributed. Comme expliqué précédemment, il a pour but d'ajouter un facteur temps à nos données en nous permettant de réaliser le même traitement pour chaque image composant notre séquence, dans notre cas en appliquant **VGG19** à chacune d'elles. La sortie de cette couche peut alors être vue comme une liste de résultats, contenant une sortie pour chacune de nos images. Notre gradient sera de même dimension que la sortie de cette couche. Il nous sera alors possible d'appliquer chaque gradient à la sortie correspondante pour obtenir notre carte d'activation et ainsi la projeter sur l'image en question.

Nous avons enregistré les résultats de nos visualisations en parallèle avec la visualisation destiné à l'opérateur. Cette démarche facilite le débogage et permet de désactiver les visualisations générées en cas d'utilisation en production, ce qui accélère le processus. Cette approche permet également de maintenir une visualisation cohérente pour chaque image plutôt que d'avoir des zones d'attention changeantes, ce qui perturberait les personnes analysant les vidéos. La raison est que les réseaux de neurones convolutifs ne possèdent pas de zone d'attention fixe, contrairement aux vision transformers. Ils scannent plutôt les images en les balayant à la recherche de caractéristiques, d'où des zones d'attention qui se déplacent.

## 7 Fonctionnement

Le fonctionnement de notre modèle de détection d'anomalies est présenté en figure 7.1. Il repose sur une architecture sophistiquée qui intègre les deux types d'analyses mentionnées : l'analyse temporelle et l'analyse spatiale. Pour l'analyse temporelle, nous avons adopté une approche basée sur un Convolutional **GRU (CGRU)** qui se compose d'une combinaison de **VGG19**, **GRU** et d'un **MLP**. Ce module d'analyse temporelle est capable de traiter des séquences de 20 images consécutives, ce qui

permet de capturer les évolutions temporelles des anomalies dans des flux vidéos continus ou des vidéos finies. Le **CGRU** utilisé peut être multi-classe ou binaire selon la problématique traitée, mais il n'opère pas en multi-étiquettes, simplifiant ainsi le processus de classification.

En ce qui concerne l'analyse spatiale, notre modèle tire parti de **YOLOv7**. Dans le mode séquentiel, les pré-traitements appliqués aux images varient en fonction de l'architecture spécifique de **YOLO** choisie. Les opérations de pré-traitement peuvent englober des tâches telles que la segmentation d'images, la détection d'objets ou encore l'analyse de la pose humaine. Dans le mode parallèle, le système identifie les objets présents dans une séquence d'images, puis utilise ces informations pour effectuer la détection finale. Cette détection est réalisée en suivant un ensemble de règles définies comme suit :

- ⇒ Soit  $A$  l'ensemble des anomalies connues par le modèle.
  - ⇒ Soit  $O$  l'ensemble des objets clés connus par le modèle.
1.  $\forall \text{ anomalie} \in A : \text{classe} = \text{anomalie}$ .
  2.  $\neg(\exists \text{ anomalie} \in A) \wedge (\exists \text{ objet clé} \in O) : \text{classe} = \text{anomalie} \cap \text{objet}$ .
  - 3.

$$\text{classe} = \begin{cases} \text{coup de feu} & \text{si } (\text{objet clé} = \text{arme à feu}) \\ & \text{et } (\text{IOU entre cet objet et une personne} > 0) \\ \text{normal} & \text{sinon} \end{cases}$$

En résumé :

- Pour toute anomalie détectée, la classe est définie comme étant cette anomalie.
- Si aucune anomalie n'est détectée, mais qu'un objet clé connu par le modèle est présent dans la séquence, alors la classe est définie comme l'anomalie associée à cet objet clé.
- Si l'objet clé détecté est une arme à feu, le système vérifie si l'**IoU** entre cette arme à feu et une personne détectée est supérieur à 0. Si c'est le cas, cela signifie qu'une personne est à proximité d'une arme à feu, et la classe est alors définie comme "coup de feu". Dans le cas contraire, lorsque l'**IoU** est inférieur ou égal à 0, la classe est définie comme "normal".

Ainsi, le système prend en compte à la fois la présence d'anomalies, d'objets clés et les relations entre eux pour attribuer la classe appropriée à chaque situation détectée dans la séquence d'images.

Une caractéristique fondamentale de notre système est son module d'explicabilité qui peut être utilisé indépendamment du mode de fonctionnement choisi. Ce module permet d'expliquer les décisions prises par les différents composants de notre modèle, notamment le **CGRU**, en mettant en évidence les zones d'attention qui ont influencé les détections. Cette fonctionnalité est essentielle pour améliorer la transparence et la compréhension des résultats du modèle, contribuant ainsi à son utilité et à son adoption.

En somme, notre modèle de détection d'anomalies combine habilement l'analyse temporelle et spatiale pour identifier les comportements anormaux dans les vidéos finies ou en flux continu. En exploitant des modèles tels que **CGRU** et **YOLOv7**, tout en offrant des possibilités de personnalisation en fonction des besoins spécifiques de l'application. Dans la prochaine section, nous allons plonger dans les expérimentations et les résultats qui ont joué un rôle déterminant dans notre choix et notre configuration de ces modèles.

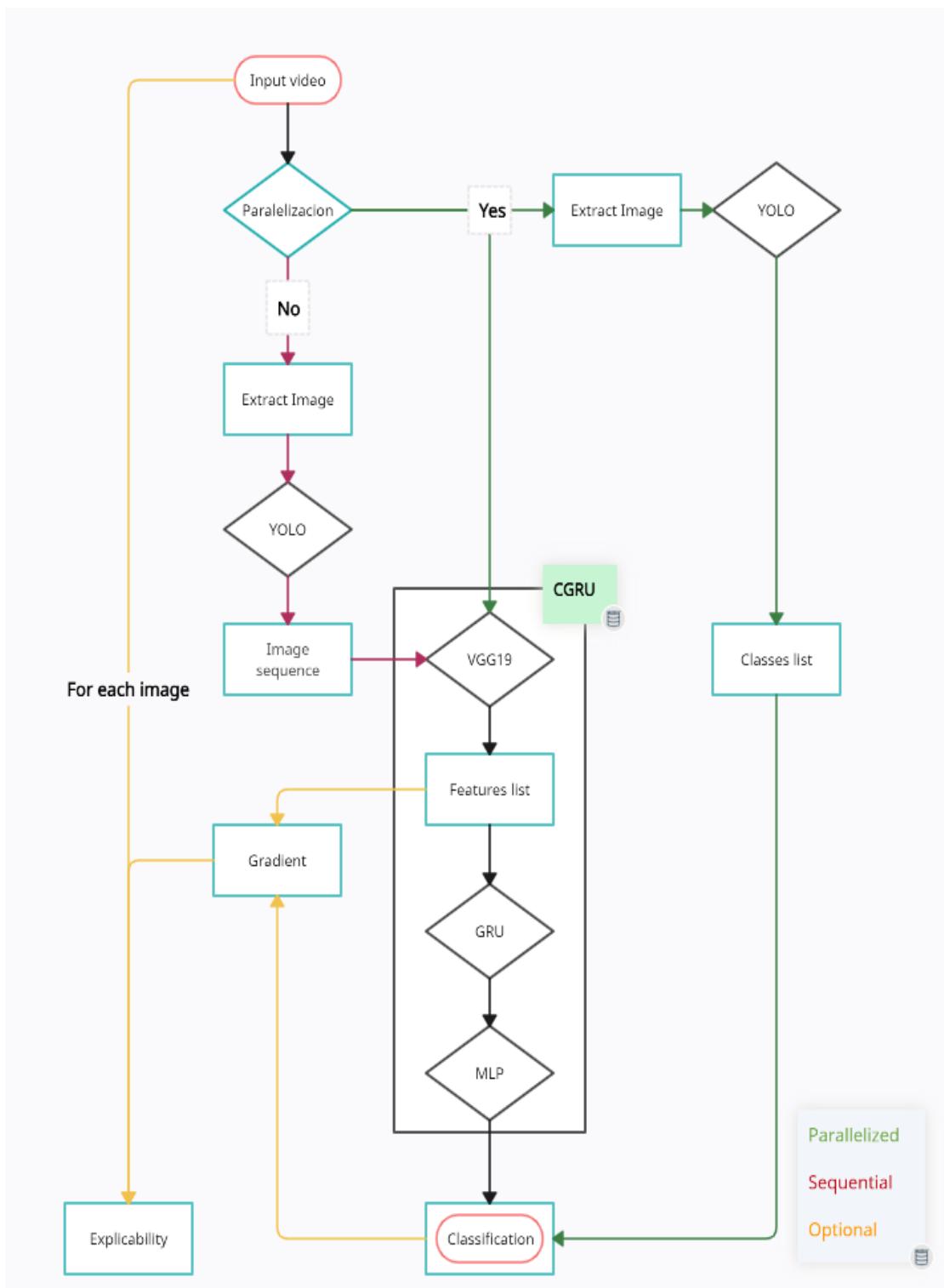


Fig. 7.1 : Fonctionnement global

## 8 Conclusion

Au fil de ce chapitre, nous avons présenté en détail chaque composant de notre architecture, qu'il s'agisse de l'analyse spatiale exploitant **YOLOv7** ou de l'analyse temporelle via le **CGRU**, tout en soulignant le rôle essentiel qu'ils jouent dans le processus global de détection d'anomalies. De plus, nous avons également exposé les jeux de données utilisés pour entraîner chacun de ces modèles, ainsi que les méthodes de prétraitement appliquées à ceux-ci. À présent, nous allons explorer les résultats des expérimentations que nous avons menées, qui ont guidé notre décision d'adopter ces modèles pour notre configuration.



# Chapitre 4

## Expérimentations et résultats

### Sommaire

---

<b>1</b>	<b>Introduction</b>	<b>86</b>
<b>2</b>	<b>Prétraitement des données</b>	<b>86</b>
<b>3</b>	<b>Expérimentations en mode série</b>	<b>93</b>
<b>4</b>	<b>Fonctionnement en mode parallèle</b>	<b>100</b>
<b>5</b>	<b>Mode traçage (tracabilité des résultats)</b>	<b>106</b>
<b>6</b>	<b>Synthèse</b>	<b>112</b>
<b>7</b>	<b>Analyse temporelle : la détection d'anomalie</b>	<b>114</b>
7.1	Choix du modèle	114
7.2	Résultats	118
<b>8</b>	<b>Analyse spatiale : la détection d'objet</b>	<b>131</b>
8.1	Choix du modèle	131
8.2	Résultats	135
<b>9</b>	<b>Conclusion</b>	<b>136</b>

---

## 1 Introduction

N'ayant pas pu avoir accès, durant mon doctorat, à de véritables caméras de surveillance, toutes les vidéos utilisées dans les expérimentations représenteront des flux d'informations finis (vidéos téléchargées en local). Les données vidéos demandent une grande puissance de calcul pour être traitées et, n'ayant pas accès à un serveur GPU au sein de mon entreprise ni de mon laboratoire de recherche, toutes les expérimentations présentées dans ce mémoire ont été réalisées sur un ordinateur équipé de 32 Go de RAM, d'un processeur Intel Core i9 contenant 16 cœurs cadencés à 2,3 GHz, ainsi que d'une carte graphique Nvidia GeForce RTX2080 possédant 8 Go de RAM dédiée.

Dans ce chapitre nous aborderons la phase d'expérimentation au cours de laquelle nous avons combiné nos deux modèles pour évaluer leurs performances en matière de détection d'anomalies. Nous étudierons comment les résultats obtenus varient selon le mode choisi, en parallèle ou en série. Après cela, nous présenterons les expériences menées en matière de détection d'objets et de détection d'anomalies, qui nous ont permis de déterminer quels modèles choisir pour chacune de nos approches. Enfin, nous conclurons cette partie par une synthèse globale de nos travaux.

## 2 Prétraitement des données

Dans cette section, nous détaillons les techniques de préparation appliquée à nos données.

Nous avons vu à la section 1 qu'une vidéo peut contenir une ou plusieurs anomalies délimitées par des séquences normales. En raison de cela, nous avons dû découper les vidéos contenant une anomalie pour isoler les anomalies et faciliter leur analyse. Malgré cela, la gestion des données reste un véritable défi pour ce projet, notamment en raison de la masse de données vidéos à traiter.

Charger l'intégralité du jeu de données d'un coup n'est tout simplement pas envisageable en raison des limitations de mémoire de notre système. Afin de résoudre ce problème, et charger nos données petit à petit en mémoire, nous avons opté pour l'utilisation d'un générateur<sup>1</sup>.

Notre première expérimentation a donc été de comparer l'effet de divers générateurs. Nous en avons principalement testé quatre (voir figure 2.1) :

1. Avec fenêtre coulissante pour former des séquences successives.

---

1. [Code source du générateur](#)

2. Avec fenêtre coulissante et chevauchement entre les séquences.
3. Avec pas dynamique, collectant  $x$  images par vidéo.
4. Avec fenêtre coulissante et pas dynamique.

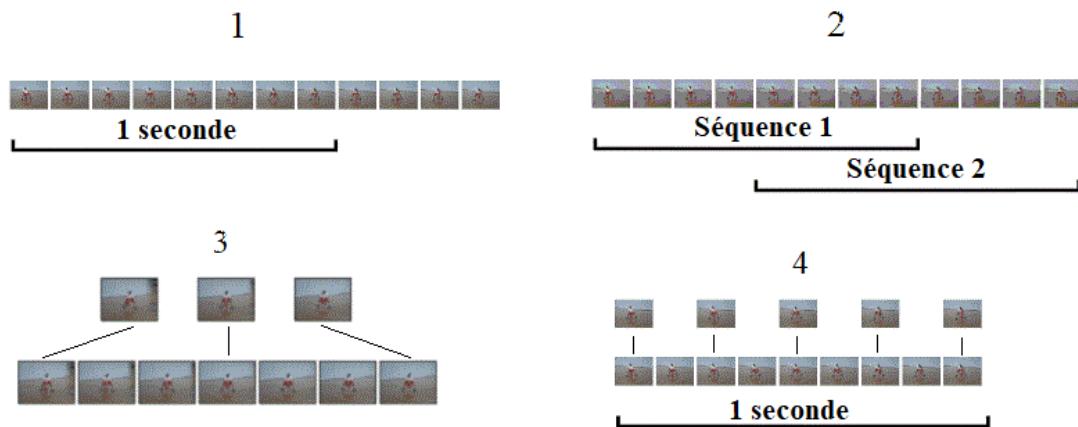


Fig. 2.1 : Fonctionnement des générateurs testés (Ferlet, 2019)

Le générateur avec lequel nous avons obtenu les meilleurs résultats est le troisième. Ceci s'explique par le fait qu'il résout les deux principaux désavantages qu'ont les générateurs utilisant une fenêtre coulissante (générateurs 1, 2, 4). En effet les modèles formés à l'aide d'un de ces générateurs ont un temps d'apprentissage qui dépend de la durée des vidéos : plus les vidéos à traiter sont longues, plus le temps d'apprentissage le sera. Leur deuxième inconvénient est que la taille de la fenêtre peut être assez complexe à définir étant donné que pour chaque vidéo celle-ci doit obligatoirement contenir l'intégralité de l'action que nous souhaitons apprendre au modèle sous peine d'influencer les performances de celui-ci.

Malgré ses avantages, le troisième générateur a quand même un assez gros défaut lui aussi lié au paramétrage de la séquence. Pour chacune des vidéos traitées il va calculer un pas dynamique en fonction du nombre de **Frame Per Second (FPS)** de la vidéo et du nombre d'images fixé pour constituer nos séquences. Le problème ici s'exprime surtout pour les jeux de données ayant des actions aux durées différentes, car dans ce cas-là, le temps séparant les images pourra être très variable. Pour des actions courtes seules quelques secondes sépareront les images constituant notre séquence, alors que dans les vidéos longues nous pourrons avoir plusieurs minutes entre chacune des images. Par conséquent plus, une vidéo sera

courte plus elle sera détaillée contrairement à une vidéo longue qui le sera beaucoup moins.

Par sa faculté à générer des séquences contenant des images non-successives le troisième générateur nous donne la possibilité, en changeant le pas, de paramétrier l'intervalle de détection voulu : pouvant aller d'une détection par séquence jusqu'à une détection par vidéo. Ce qui nous permet à terme, de pouvoir traiter aussi bien des vidéos finies que des flux vidéo continus.

Une fois notre générateur sélectionné, nous nous sommes intéressés à la taille de nos données. Nous avons fait varier la taille de nos images ; le tableau 4.1 montre que la taille optimale est de  $112 \times 112$ .

Tab. 4.1 : Variation de la taille de nos images

Taille des images	Précision	Rappel	F1-Score
80*80	20.2%	85.8%	32.8%
112*112	<b>22.9%</b>	<b>89.6%</b>	<b>36.5%</b>
140*140	22.1%	89.1%	35.4%

Pour la taille des séquences, nous avons testé des séquences de 15 à 30 images, car la durée minimale de nos vidéos est d'une seconde, soit, selon les standards actuels, 30 **FPS**. Le tableau 4.2 montre que la taille optimale est de 20 images.

Tab. 4.2 : Variation de la taille de nos séquences

Taille des séquences	Précision	Rappel	F1-Score
15	22.3%	<b>89.8%</b>	35.7%
20	<b>33%</b>	83.1%	<b>47.3%</b>
30	22.9%	89.6%	36.5%

Afin d'évaluer les performances de chacune de nos expérimentations, nous avons développé notre propre algorithme pour spécifier le pas voulu lors de l'extraction des images. Ceci permet d'évaluer nos modèles dans les mêmes conditions que lors de la phase d'apprentissage, c'est-à-dire effectuer une détection par vidéo ou une détection par séquence (afin de connaître les performances de celui-ci sur des flux continus). Notre objectif à terme étant d'aider des surveillants dans leurs tâches d'examen de flux continu, les performances liées à une détection par séquence seront privilégiées.

Après avoir paramétré nos séquences, nous avons enrichi nos données en appliquant sur chacune des images des techniques de data augmentation propres à ce type de données, telles que des effets miroir, des zooms et des changements de luminosité visant à enrichir nos données (voir Figure 2.2) et ainsi les multiplier par trois, tout en restant fidèles à des situations réalistes.

Ensuite, nous avons testé diverses techniques de prétraitement, en commençant par des méthodes couramment utilisées dans le domaine de la vision par ordinateur : le calcul de l'optical flow, la différence inter-images et l'application de masques (voir Figure 2.3).

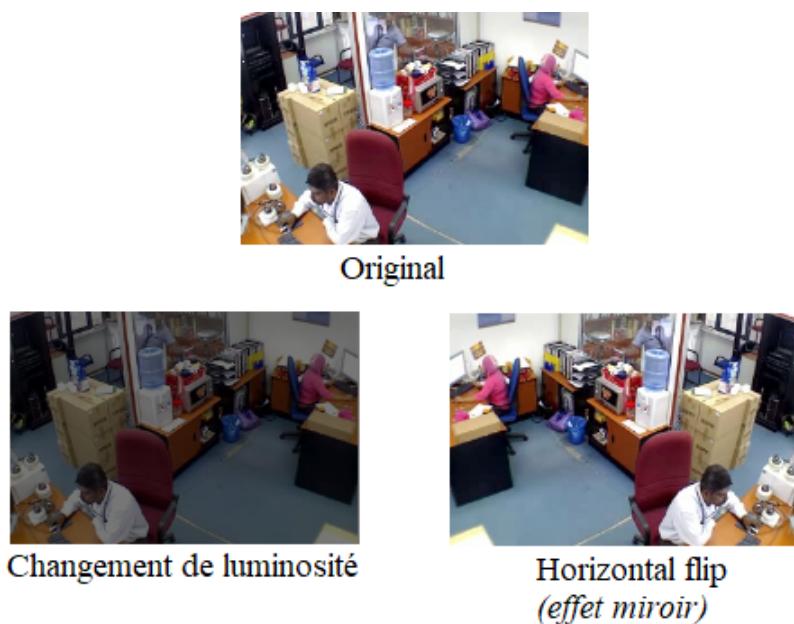


Fig. 2.2 : Data augmentation

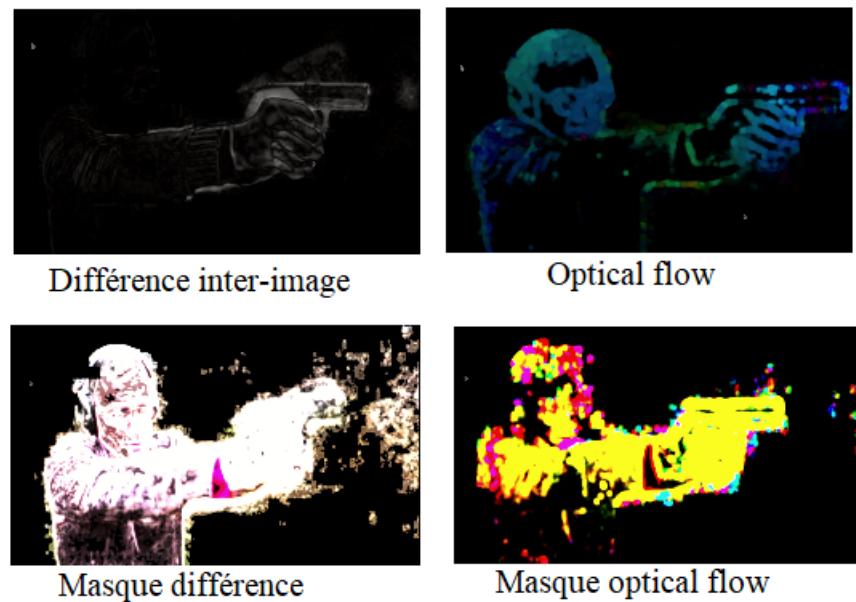


Fig. 2.3 : Pré-traitement appliqué à une image représentant un coup de feu

L’optical flow ne permet pas d’obtenir de résultats exploitables, l’accuracy reste stable (à 50%) tout au long de l’apprentissage. La différence inter-image donne des résultats un peu plus intéressants, qui sont améliorés par l’augmentation de données et dégradés par l’emploi d’un masque. Mais à notre grande surprise nous avons obtenu de meilleurs résultats lorsqu’on se limite à l’augmentation de données. Les performances obtenues sont résumées dans le tableau 4.3 (à part la première ligne, tous les résultats incluent l’augmentation de données). Avec aucun prétraitement, la précision est mauvaise (donc le F1-score), et le rappel excellent. L’augmentation de données sans autre prétraitement donne la meilleure précision et un rappel correct. La différence inter-images dégrade les résultats, avec ou sans masque.

Tab. 4.3 : Techniques appliquées au modèle Bagarre

Technique	Accuracy	Précision	Rappel	F1-Score
Aucune	<b>83.4%</b>	33%	<b>83.1%</b>	47.3%
Data augmentation	63.16%	<b>93.62%</b>	60.31%	<b>73.36%</b>
Différence inter-image	59.27%	92.19%	56.35%	69.99%
Optical Flow	X	X	X	X
Masque différence	23.10%	82.24%	10.80%	19.24%
Masque Optical Flow	X	X	X	X

Ces résultats peuvent s'expliquer par le fait que nous possédons de nombreuses vidéos filmées avec des smartphones issues principalement d'amateurs. Dans ce genre de vidéos la caméra est généralement mobile, par conséquent l'arrière-plan l'est aussi. Du coup la différence inter-image pousse le modèle à extraire des caractéristiques non-pertinentes, comme le montre la figure 2.4. Cette figure montre la différence entre deux images successives extraites d'une vidéo représentant une interview entre deux boxeurs avant leur combat, dans laquelle on peut remarquer que le texte présent sur les affiches en arrière-plan ressort davantage que les personnes présentes au premier plan.



Fig. 2.4 : Images représentant une bagarre

Nous avons ensuite employé des méthodes de préparation plus sophistiquées en soumettant nos données à un prétraitement par des modèles spécialisés. Les résultats générés ont ensuite été utilisés pour la détection d'anomalies.

Notre première idée a été d'utiliser **DINO** (voir figure 2.5), présenté dans la section 1.5 de l'état de l'art.



Fig. 2.5 : Carte d'attention pour une vidéo de la classe coup de feu

Après avoir testé DINO sur nos différents jeux de données, nous avons constaté que cette méthode demande un temps de calcul important, ce qui la rend inadaptée à notre problématique de quasi-temps réel.

Cependant, les résultats obtenus sur les vidéos de coups de feu ont été encourageants, comme en témoignent les tableaux 4.4 et 4.5. Le modèle est capable d'extraire des caractéristiques pertinentes pour cette classe d'événements, sans nécessiter de ré-apprentissage préalable.

Ce n'est malheureusement pas le cas pour des bagarres comme l'indique la courbe d'apprentissage présentée sur la figure 2.6. Ces résultats nous confortent dans notre décision d'écarter les Vision Transformer de notre étude pour nous concentrer sur des méthodes plus adaptées à notre problématique.

Tab. 4.4 : Performance sur le jeu de données coup de feu avec DINO en prétraitement

Classe	Accuracy	Précision	Rappel	F1-Score
Coup de feu	99.7%	99.2%	96%	97.6%

Tab. 4.5 : Matrice de confusion sur le jeu de données coup de feu en avec DINO en prétraitement

		Predicted	Coup de feu	
			Normal	Coup de feu
Truth	Coup de feu	99.96%	0.04%	
	Normal	4%	96%	

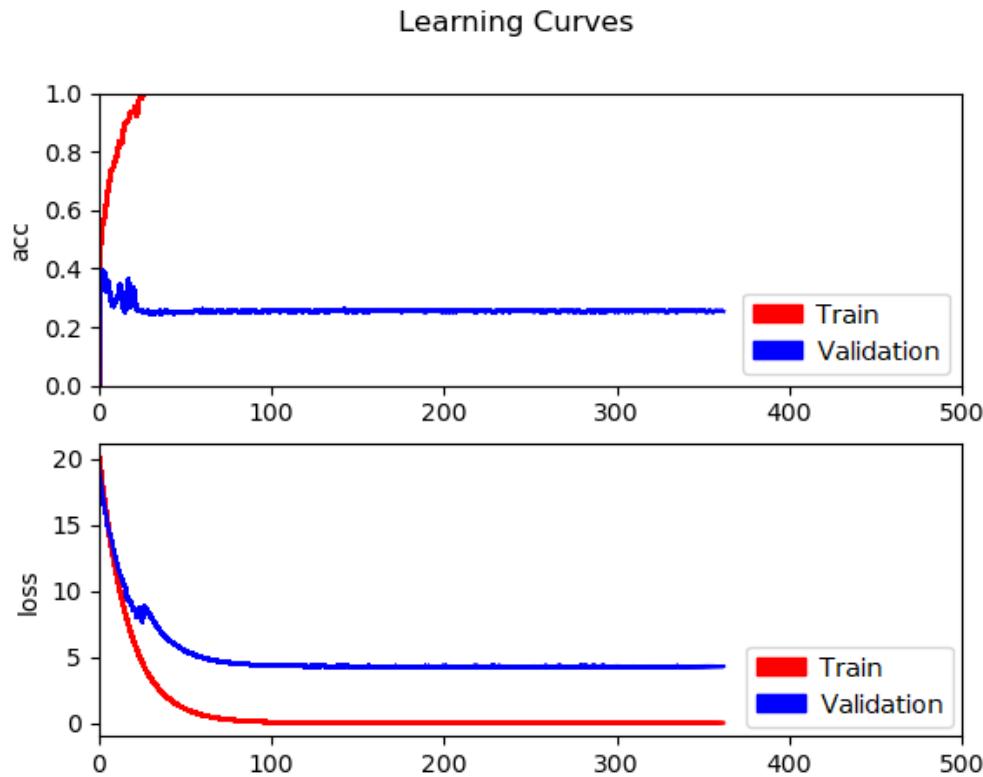


Fig. 2.6 : Surapprentissage pour la classe Bagarre avec DINO comme prétraitement

Nous avons également utilisé **YOLO** afin de détecter les objets et entités présentes dans nos vidéos ou encore estimer la pose des personnes, avec succès, c'est pourquoi nous l'avons intégré dans l'architecture proposée. Nous détaillons les résultats dans la section suivante.

### 3 Expérimentations en mode série

Dans cette section, nous allons évaluer les performances de nos modèles disposé en série, et montrer quels paramètres, quelles techniques et quelles stratégies permettent de les améliorer. Nous commençons par décomposer les vidéos en extraignant chacune de leurs images afin de les passer à **YOLO** pour détecter les différents objets présents dans les images et intégrer les résultats de cette détection (la boîte autour des objets). Ensuite, la séquence video est reformé afin d'être transmise

à **CGRU** pour effectuer nos détections. Nous avons comparé les performances de cette approche avec celles du modèle **CGRU** seul entraîné sur les mêmes vidéos. Les résultats peuvent être trouvés dans le tableau 4.6.

Tab. 4.6 : Performance de CGRU vs YOLO+CRU sur les classes coup de feu / normal

Modèle	Accuracy	Précision	Rappel	F1-Score
<b>CGRU</b>	91.89%	35.14%	84.86%	49.70%
<b>YOLO et CGRU</b>	91.89%	35.14%	84.86%	49.70%

On constatera qu'afficher les boîtes autour des objets n'a aucune incidence sur les performances de notre modèle. Ce qui signifie que celui-ci ne les prend pas en considération lors de son apprentissage. Nous avons poussé notre prétraitement en utilisant ces boîtes pour effectuer des masques afin de ne conserver que ces parties de l'image et ainsi retirer un maximum de l'arrière-plan (figure 3.1).

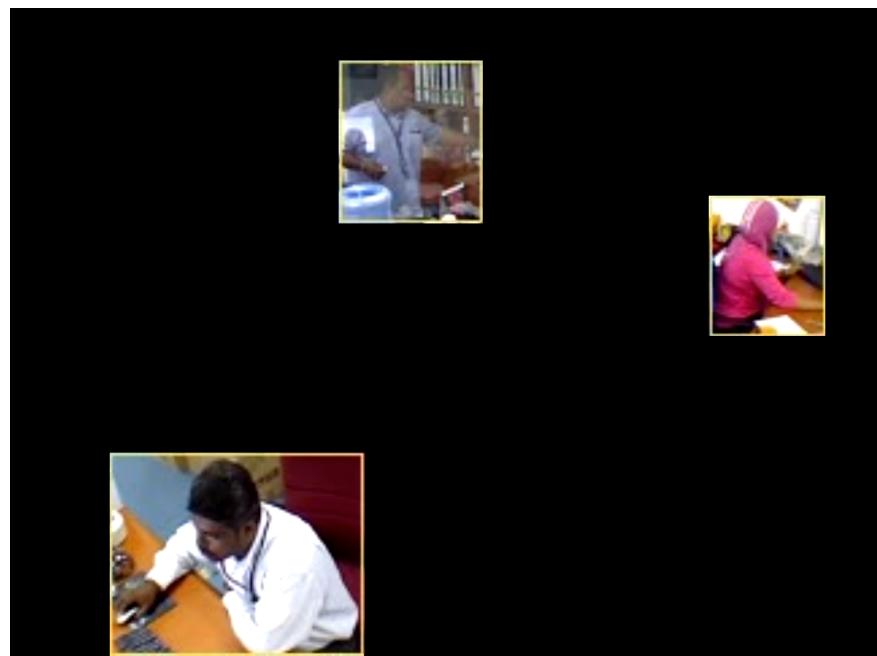


Fig. 3.1 : Masque effectué avec YOLO

Pour toute image où aucun objet n'est détecté, nous avions deux possibilités : soit laisser l'image d'origine, soit la remplacer par une image toute noire. Cette préparation étant importante pour la fiabilité de notre modèle de détection d'anomalie, nous avons décidé de tester les deux possibilités. D'après les résultats

obtenus (tableaux 4.7, 4.8, 4.9, 4.10), il semblerait que les performances soient assez proches ; malgré cela on peut observer de nombreuses différences. L'utilisation d'un fond noir semble améliorer la perception de la classe normale au détriment des autres classes (bagarre, incendie), ce qui peut être expliqué par l'hypothèse suivante : dans certains cas, des objets clefs sont présents comme des flammes ou personnes, mais **YOLO** ne les a pas détecté ou alors ces détections n'ont pas atteint la fiabilité et le taux de superposition fixé. Par conséquent, ces informations ont été perdues à cause du fond noir.

Les seuils de confiance et de superposition sont des paramètres importants dans la détection d'objets utilisant **YOLO**, car ils déterminent le niveau de confiance nécessaire pour considérer qu'un objet est détecté correctement. Ces paramètres sont donc fixés manuellement avant chaque analyse. Dans notre cas, le seuil de confiance a été fixé à 55%, tandis que le seuil de superposition a été fixé à 25%.

Tab. 4.7 : Performances pour un masque avec fond noir

Accuracy	Précision	Rappel	F1-Score
75.58%	79.34%	75.58%	76.50%

Tab. 4.8 : Matrice de confusion pour un masque avec fond noir

Truth \ Predicted	Bagarre	Coup de feu	Incendie	Normal
Bagarre	55.52%	4.43%	1.76%	38.29%
Coup de feu	16.20%	41.44%	1.53%	40.83%
Incendie	14.09%	12.49%	20.59%	52.83%
Normal	10.55%	2.98%	2.09%	84.38%

Tab. 4.9 : Performances pour un masque sans fond noir

Accuracy	Précision	Rappel	F1-Score
72.22%	82.37%	72.22%	75.68%

Tab. 4.10 : Matrice de confusion en pourcent pour un masque sans fond noir

Truth \ Predicted	Bagarre	Coup de feu	Incendie	Normal
Truth	Bagarre	Coup de feu	Incendie	Normal
Bagarre	63.06%	9.33%	1.96%	25.65%
Coup de feu	25.99%	41.45%	2.29%	30.27%
Incendie	19.23%	15.98%	32.44%	32.35%
Normal	15.33%	4.41%	2.07%	78.19%

Dans certains cas, nos anomalies représentent des actions spécifiques réalisées par des personnes physiques, actions que nous pouvons reconnaître par la pose que prennent ces personnes. Avec la version 7 de **YOLO**, on peut effectuer de la détection de pose en traçant le « squelette » de chaque personne présente à l'écran tel que pourraient le faire des technologies comme Openpose (voir le blog de Barla ([2022](#))).

Nous avons alors effectué une estimation de la pose pendant le prétraitement, afin de voir si celle-ci pouvait améliorer la détection d'anomalies liée aux comportements. Dans un premier temps, comme toutes nos anomalies ne sont pas liées à des personnes physiques, nous avons restreint notre jeu de données aux classes Bagarre et Coup de feu. De plus, nous avons aussi retiré l'arrière-plan des vidéos pour éviter l'extraction de caractéristiques non-significatives [figure [3.2](#)].

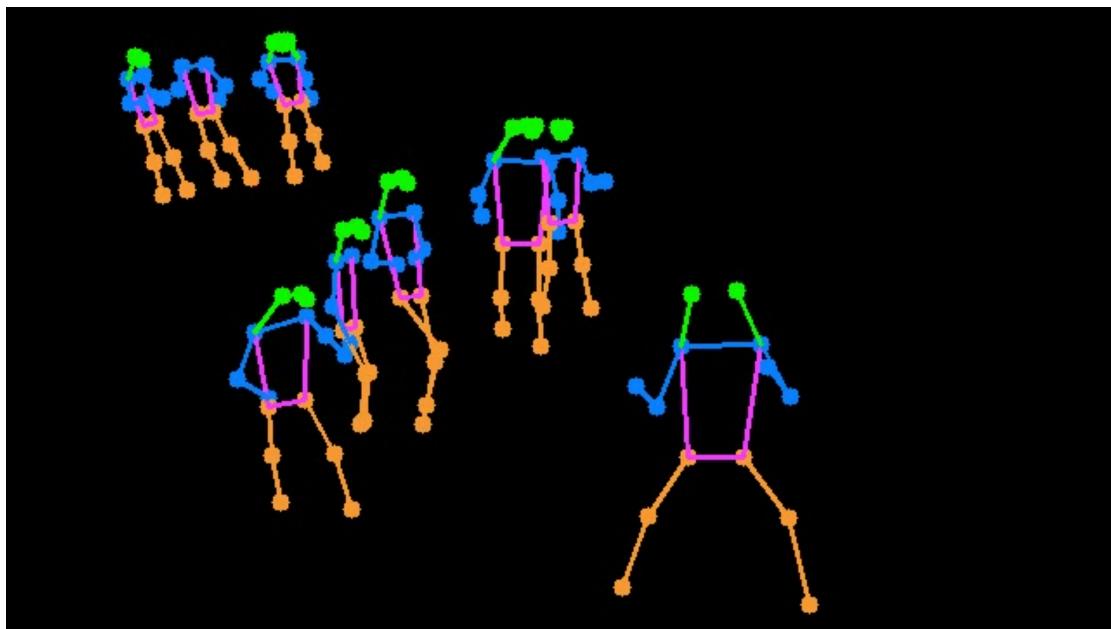


Fig. 3.2 : Estimation de la pose par YOLOV7 pour une bagarre, sans fond

Tab. 4.11 : Évaluation par séquence (3 classes), sans fond

Accuracy	Précision	Rappel	F1-Score
87.2%	90.9%	87.8%	89%

Tab. 4.12 : Matrice de confusion pour une évaluation par séquence (3 classes), sans fond

Truth \ Predicted			
	Bagarre	Coup de feu	Normal
Bagarre	64.5%	1%	34.5%
Coup de feu	11.7%	70.7%	17.6%
Normal	9.5%	0%	90.5%

Malgré de bonnes performances indiquées dans le tableau 4.11 4.12, sans le fond, il nous est impossible de détecter d'autres anomalies qui ne seraient pas rattachées à un comportement humain. Dans un second temps, nous avons donc

réintroduit le fond de nos images [figure 3.3, 4.14] ainsi que toutes les classes de notre jeu de données afin d'entraîner notre modèle et, voir si celui-ci est capable de détecter des anomalies comme des incendies en utilisant des données comportant ce type de pré-traitement [Voir tableau 4.13, 4.14].

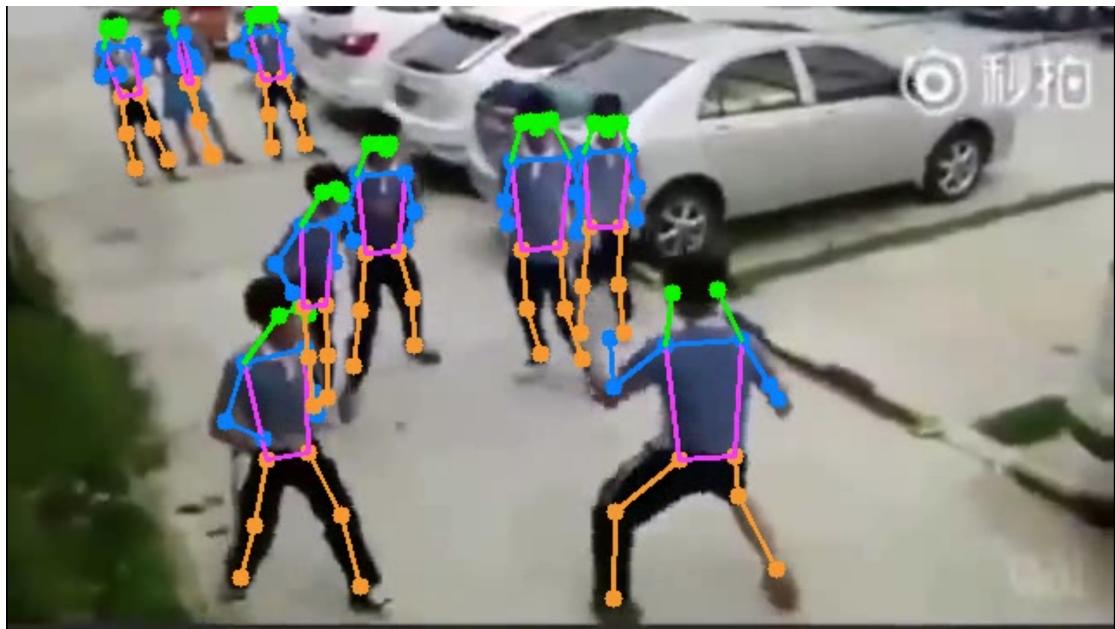


Fig. 3.3 : Estimation de pose par YOLOV7 pour une bagarre, avec fond

Tab. 4.13 : Évaluation par séquence (4 classes)

Accuracy	Précision	Rappel	F1-Score
87.3%	87.6%	87.3%	87.1%

Tab. 4.14 : Matrice de confusion en pourcent pour une évaluation par séquence (4 classes)

Truth \ Predicted				
	Bagarre	Coup de feu	Incendie	Normal
Bagarre	60.5%	2.4%	1.3%	35.8%
Coup de feu	10%	55.6%	14.8%	19.6%
Incendie	15.5%	10.6%	48%	25.9%
Normal	3.4%	0.6%	1%	95%

Au niveau des performances globales, il semblerait qu'ajouter la classe incendie n'ait pas de réel impact sur les performances, mais si on regarde les résultats de plus près, ce n'est pas le cas. Les matrices de confusion montrent une forte diminution de la fiabilité à détecter la classe Coup de feu, diminution due au fait que notre modèle étiquette certains coups de feu comme des incendies. Pour finir, nous avons décidé de remplacer notre modèle multi-classe par un modèle normal / anormal afin de vérifier l'impact de **YOLO**. Nous avons donc formé deux nouveaux modèles : l'un en utilisant les images sans fond, et l'autre en conservant celui-ci.

Tab. 4.15 : Détection normal / anormal sans fond (avec bagarre et coup de feu)

Accuracy	Précision	Rappel	F1-Score
89.18%	95.16%	92.25%	93.83%

Tab. 4.16 : Matrice de confusion de la détection normal / anormal sans fond basé sur 2 anomalies : bagarre et coup de feu

Truth \ Predicted		
	Anormal	Normal
Anormal	64.10%	35.90%
Normal	7.75%	92.25%

Tab. 4.17 : Détection normal / anormal avec fond (avec 3 anomalies : bagarre, coup de feu et incendie)

Accuracy	Précision	Rappel	F1-Score
84.46%	95.52%	84.84%	89.87%

Tab. 4.18 : Matrice de confusion de détection normal / anormal avec fond basé sur 3 anomalies : bagarre, coup de feu et incendie

Truth \ Predicted		
	Anormal	Normal
Anormal	82.78%	17.22%
Normal	15.15%	84.85%

Quelle que soit le prétraitement utilisé, rassembler les différents types d'anomalies améliore les performances de nos modèles, à la réserve près qu'inclure des anomalies n'ayant aucune relation avec la posture des personnes présentes, telles que des incendies, dégrade ces dites performances.

D'autre part, on peut observer que dans des cas de traitement multi-classe (tableaux 4.15, 4.16 et 4.17, 4.18), l'utilisation de **YOLO** pour prétraiter nos données améliore les performances de manière significative.

## 4 Fonctionnement en mode parallèle

Dans un second temps, nous avons testé nos modèles en mode parallèle. L'idée de cette démarche est d'utiliser une détection d'objets en parallèle de l'analyse temporelle puis de combiner les résultats pour sortir une détection en réduisant les taux de faux positifs et faux négatifs.

Notre première expérimentation a donc été de combiner notre **CGRU** formé sur la classe "coup de feu" et **YOLOV4** entraîné à reconnaître des armes à feu en utilisant une règle simple : si la classe normale est prédite mais qu'une arme à feu est détectée avec un taux de confiance dépassant un certain seuil, alors on remplacera la classe normale par la classe "coup de feu". Comme notre **CGRU** est entraîné à détecter des incendies et que **YOLOV4** est formé à reconnaître des flammes, nous les avons aussi combinés et comparés en suivant la même idée. Pour

chacune des combinaisons, nous nous contenterons d'évaluer leurs performances pour des détections réalisées sur chacune des séquences de 20 images présentes dans nos vidéos avec un seuil de confiance fixé à 55%.

Tab. 4.19 : Performance liée à la détection d'incendies

Modèle	Accuracy	Précision	Rappel	F1-Score
CGRU	86%	85.8%	<b>96.2%</b>	90.7%
<b>CGRU + YOLO</b>	<b>93.2%</b>	<b>94.6%</b>	95.9%	<b>95.2%</b>

		Matrice de confusion CGRU	
		Predicted	
Truth	Incendie	Incendie	Normal
	Normal	60.5%	39.5%
Truth	Incendie	3.7%	96.3%
	Normal		

		Matrice de confusion CGRU + YOLO	
		Predicted	
Truth	Incendie	Incendie	Normal
	Normal	86.4%	13.6%
Truth	Incendie	4%	96%
	Normal		

Tab. 4.20 : Performance liée à la détection de coup de feu (partie déjà présentée dans le tableau 4.6)

Modèle	Accuracy	Précision	Rappel	F1-Score
CGRU	91.89%	35.14%	84.86%	49.70%
<b>YOLO + CGRU</b>	<b>91.89%</b>	<b>35.14%</b>	<b>84.86%</b>	<b>49.70%</b>

Bien que les faux négatifs restent élevés, ce que montre le tableau 4.19, il y a une nette amélioration de la détection pour la classe incendie, alors que pour la classe coup de feu, il n'y a pas de changement. À première vue, il semblerait que notre **CGRU** et **YOLO** partagent les mêmes caractéristiques en matière de détection de coup de feu. Ces performances semblent dépendantes de la précision de **YOLO** ainsi que de la condition fixée dans le composant de correction (voir 5).

Nous avons donc décidé d'affiner notre condition pour la classe coup de feu afin de réduire les mauvaises détections. Vu que **YOLO** utilise un calcul d'IoU pour

filtrer les objets détectés, nous avons décidé de faire de même et de considérer que l'anomalie coup de feu ne pouvait survenir que si l'**IoU** entre une arme et une personne était supérieur à zéro, c'est-à-dire que les deux boîtes englobantes se touchent, se chevauchent ou que l'une soit incluse dans l'autre, comme dans la figure 4.1.



Fig. 4.1 : Résultat de YOLO V4 entraîné sur les classes "gun" + "person"

En partant de l'idée qu'une arme à feu ne représente un risque que lorsque celle-ci est utilisée par une personne, nous avons entraîné un nouveau modèle sur un jeu de données incluant des armes à feu et certaines images provenant du jeu de données COCO. Pour la détection de personne, le tableau 4.21 montre que ce nouveau modèle est plus performant que **YOLO** version 4, par contre pour la détection des armes à feu le modèle qui a effectué l'apprentissage uniquement sur celles-ci reste le meilleur.

Tab. 4.21 : Détection de la classe personne sur le jeu de données COCO + le nôtre

Modèle	Vrai Positif	Faux Positif	Faux Negatif
<b>YOLOv4</b>	37.57%	12.82%	62.43%
<b>YOLO</b> (gun / personne)	<b>45.27%</b>	<b>0.62%</b>	<b>54.73%</b>

Modèle	IoU	Précision	Rappel	F1-Score
<b>YOLOv4</b>	60.68%	74.56%	37.57%	49.69%
<b>YOLO</b> (gun / personne)	<b>80.76%</b>	<b>98.66%</b>	<b>46.32%</b>	<b>61.18%</b>

Tab. 4.22 : Comparaison de performance pour chacun de nos modèles

Modèle	Accuracy	Précision	Rappel	F1-Score
CGRU	91.89%	35.14%	84.86%	49.70%
CGRU + YOLO (gun)	91.89%	35.14%	84.86%	49.70%
CGRU + YOLO (gun + personne)	<b>92.03%</b>	<b>36%</b>	<b>88.73%</b>	<b>51.15%</b>

Matrice de confusion CGRU + YOLO (Personne + Gun)		
Truth \ Predicted		
	Coup de feu	Normal
Coup de feu	88.38%	11.62%
Normal	7.75%	92.25%

D'après les résultats du tableau 4.22, coupler le **CGRU** avec **YOLO** en réalisant un calcul d'**IoU** entre les personnes et armes à feu détectées permet de conserver de bonnes performances sur la classe normale et d'améliorer celle de la classe coup de feu d'environ 4%.

Dans la même optique, nous avons voulu voir s'il était possible de réduire les faux positifs une fois nos deux modèles combinés. Nous avons donc réalisé une nouvelle évaluation de ceux-ci en maintenant les conditions préalablement citées dans le but de réduire les faux négatifs, mais aussi les faux positifs. Pour chaque détection d'incendies, si aucune flamme n'est détectée par **YOLO** la séquence sera considérée comme normale, de même pour un coup de feu si aucun pistolet n'est perçu. Après évaluation, on peut constater une forte baisse du taux de faux positifs, mais elle s'accompagne d'une baisse du taux de vrai positifs, montrant un manque de précision de la part de **YOLO** (tableau 4.23).

Tab. 4.23 : Performance du modèle CGRU + YOLO

Modèle	Accuracy	Précision	Rappel	F1-Score
Coup de feu	96.91%	95.58%	36.39%	52.71%
Incendie	91.73%	90%	98.83%	94.24%

Matrice de confusion RCNN + YOLO (Réduction des faux positif)

Truth \ Predicted		
	Coup de feu	Normal
Coup de feu	36.40%	63.60%
Normal	0.9%	99.1%

Matrice de confusion RCNN + YOLO (Réduction des faux positif)

Truth \ Predicted		
	Incendie	Normal
Incendie	72.70%	27.30%
Normal	1.16%	98.84%

Comme les anomalies auxquelles nous nous intéressons peuvent avoir de graves conséquences sur les événements à venir, nous avons décidé de conserver le modèle minimisant le taux de faux négatifs, ce qui permet de rater le moins d'alertes possibles au risque d'en lancer des fausses. Ces fausses alertes pourront facilement être identifiées par un humain. De plus, comme il est plus simple de n'avoir qu'un seul modèle de détection d'objets lié à toutes nos anomalies, nous avons décidé là aussi de former YOLO sur nos 3 types d'objets à savoir les personnes, les armes à feu ainsi que les flammes. Pour terminer les expérimentations de nos modèles en parallèle, nous avons substitué notre modèle multi-classe par notre modèle normal / anormal dans le but de comparer les performances de ces deux architectures.

Tab. 4.24 : Performance générale : **YOLO** (multi-classes) + **CGRU** (multi-classes)

Accuracy	Précision	Rappel	F1-Score
78.42%	85.60%	78.42%	81.16%

CONFUSION MATRIX					
Truth	Predicted	Bagarre	Coup de feu	Incendie	Normal
Bagarre	Bagarre	63.66%	6.58%	1.93%	27.83%
Coup de feu	Coup de feu	9.94%	66.06%	9.33%	14.67%
Incendie	Incendie	13.66%	15.73%	57.71%	12.9%
Normal	Normal	7.43%	5.96%	3.98%	82.63%

Tab. 4.25 : Performance générale : **YOLO** (multi-classes) + **CGRU** (nomal / anomalous)

Accuracy	Précision	Rappel	F1-Score
84.15%	94.97%	85.01%	89.71%

CONFUSION MATRIX			
Truth	Predicted	Anormal	Normal
Anormal	Anormal	80.37%	19.63%
Normal	Normal	14.98%	85.02%

Comme nous pouvons le constater, rassembler toutes nos anomalies au sein d'une même classe permet d'augmenter les performances de notre modèle en évitant à celui-ci de confondre certains événements anormaux (tableau 4.25). En comparant nos modèles avec et sans l'ajout de **YOLO**, nous pouvons conclure que dans certains cas l'ajout de celui-ci permet d'obtenir des résultats légèrement moins performants voire équivalents (tableau 4.25 et 4.41). Cependant, dans la majorité d'entre eux, la présence de celui-ci permet d'améliorer les performances de nos divers modèles qu'ils soient spécialisés sur certaines classes ou multi-classe (tableau 4.24 et 4.37).

## 5 Mode traçage (tracabilité des résultats)

Le mode traçage, illustré par les figures 5.1 et 5.2 nous a permis de constater que bien que nos images soient successives, notre modèle **CGRU** ne se concentre pas sur les mêmes zones d'une image à l'autre. Afin de faciliter la visualisation de nos séquences, nous avons utilisé OpenCV pour extraire les contours de ces zones d'activation.



Fig. 5.1 : Cartes d'activation pour une vidéo provenant de la classe coup de feu extraite du jeu de test

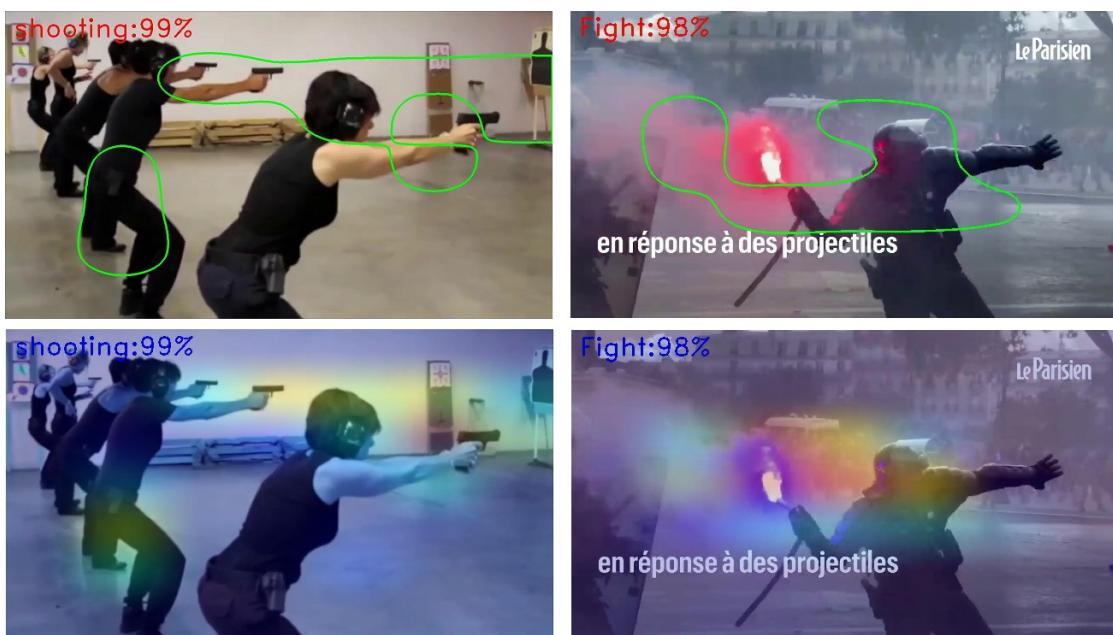


Fig. 5.2 : Exemple de visualisation pour les classes : coup de feu et bagarre

Cette visualisation des contours nous a aussi permis de remarquer des zones de faible activation difficilement perceptibles via la carte d'activation. En revanche la visualisation de contours possède quelques inconvénients : la détection des contours n'est pas très précise, il peut y avoir des contours inclus dans d'autres lorsqu'il existe une zone d'activation majeure entourée d'une zone mineure. De plus, à l'heure actuelle, ces contours ne nous indiquent pas l'intensité de ces activations.



Fig. 5.3 : Carte d'activation et visualisation des contours sur une image de la classe coup de feu



Fig. 5.4 : Carte d'activation et visualisation des contours sur une image de la classe bagarre

Les images présentées dans les figures 5.3 et 5.4 illustrent parfaitement les avantages et inconvénients de la visualisation de contours. Dans la figure 5.3, le pistolet est perçu comme une zone d'activation faible, difficile à voir sur la carte d'activation, mais très net dans les contours. On peut aussi remarquer une zone d'activation majeure entourée d'une zone mineure située à gauche de l'image située au niveau de la victime. La figure 5.4 représente une personne en train d'en frapper une autre. Avec la carte d'activation à gauche, il semble que le modèle aie bien perçu l'action mais ai predit une action normale au lieu d'une bagarre. La visualisation des contours, à droite, montre qu'en fait il n'a pas du tout détecter l'action.

Les cartes d'activation nous ont aussi permis d'observer l'influence qu'ont les autres couches (RNN, Dense, Dropout) sur les caractéristiques apprises par notre convolution, due à la rétro-propagation ainsi que de détecter des cas de sur-apprentissage visibles lorsque les zones d'activation (tâches de couleur) se concentrent sur des éléments du décor plutôt que sur les caractéristiques essentielles de l'image (figure 5.8). Cela nous a permis de mieux régler les paramètres de ces couches, en les modifiant jusqu'à ce qu'elles portent sur les parties intéressantes de nos images, comme on le voit dans les figures 5.5, 5.6, 5.7.



Fig. 5.5 : Cartes d'activation montrant la variation du dropout en utilisant 64 neurones pour GRU



Fig. 5.6 : Cartes d'activation pour une variation du nombre de neurones du GRU en utilisant un dropout de 50%



Fig. 5.7 : Cartes d'activation pour une variation du nombre de couches du classifieur



Fig. 5.8 : Cartes d'activation de notre modèle comprenant 1024 neurones pour le GRU et 0% de dropout

À l'aide de ces deux techniques, il est aussi possible de visualiser les caractéristiques de la classe normale. Dans cette classe, on retrouve un ensemble d'actions très varié comme travailler, se promener, faire du sport, etc... Pour cette classe, les mouvements réalisés sont généralement lents, contrairement à ceux des anomalies, en général brutaux et rapides. Pour un humain, la classe normale sera choisie si aucune des caractéristiques correspondant à une anomalie n'est trouvée. Or, ce n'est pas le comportement qu'adopte notre modèle. Pour détecter la classe normale, celui-ci doit trouver des caractéristiques la représentant. D'après ces diverses visualisations que sont les cartes d'activation, les filtres de convolution ainsi que les cartes de saillance [figure 5.9, 5.10, 5.11], pour notre vidéo d'exemple, il semblerait que notre modèle se base sur la posture et gestuelle des personnes présentes à l'écran afin de définir s'il s'agit ou non d'une anomalie.



Fig. 5.9 : Visualisation d'une vidéo normale provenant du jeu de test représentant une location.

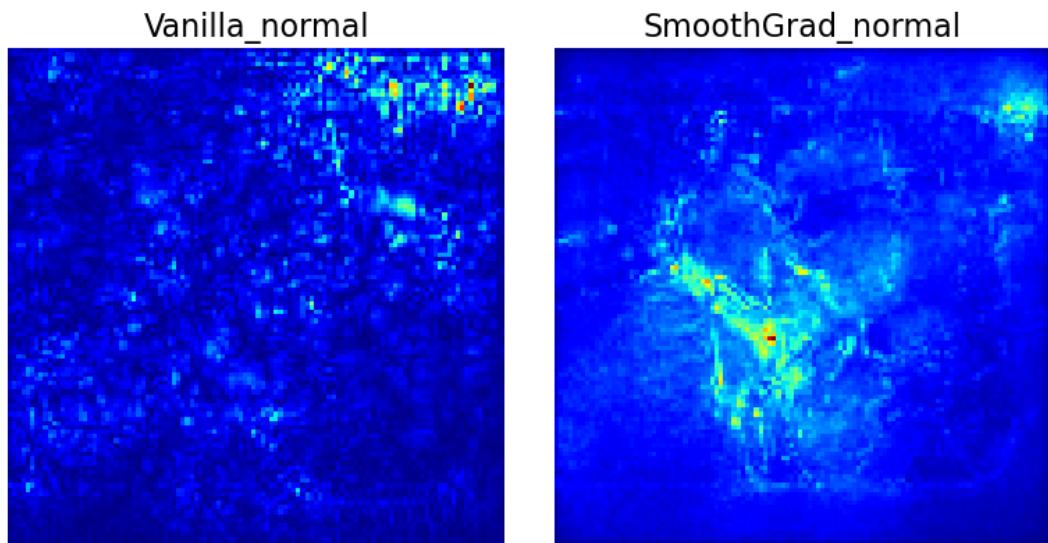


Fig. 5.10 : Carte de saillance pour une vidéo normal



Fig. 5.11 : 4 filtres de convolution pour les couches 1 et 2

## 6 Synthèse

Pour notre problématique de quasi-temps réel, nous terminerons en calculant le temps d'exécution de ces diverses modalités de fonctionnement. Afin de faciliter les comparaisons, nous utiliserons les mêmes vidéos. Le code que nous utilisons actuellement n'a pas été optimisé, faute de temps, mais cela ne gène pas la comparaison. Pour la partie où **YOLO** est en parallèle de notre **CGRU**, les deux modèles sont utilisés l'un après l'autre. Ils ne sont, pour le moment, pas gérés en multi-thread. Par ailleurs, **YOLOV7** étant plus rapide que la version 4, utilisée dans nos premiers tests pour la détection de nos objets, nous l'utilissons ici.

Tab. 4.26 : Temps d'exécution de YOLO et CGRU en parallèle

Durée de la vidéo	FPS de la vidéo	Moyenne de détections	Temps de traitements
16s	33	601ms	15s
44s	30	533ms	35s
9s	30	994ms	12s
35s	30	1.1s	57s
23s	30	1s06	35s
1min 43	30	758ms	116s (1min 56)
50s	30	826ms	61s
1min 05	30	886ms	83s (1min 23)
2s	30	847ms	847ms
9s	30	870ms	11s
2s	30	1s	1s

Tab. 4.27 : Temps d'exécution de YOLO et CGRU en série

Durée de la vidéo	FPS de la vidéo	Moyenne de détections	Temps de traitements
16s	33	1s	26s
44s	30	1s	71s (1min 11)
9s	30	1.5s	20s
35s	30	1.5s	81s (1min 21)
23s	30	1.5s	48s
1min 43	30	1.2s	193s (3min 13)
50s	30	1.3s	102s (1min 42)
1min 05	30	1.4s	134s (2min 14)
2s	30	1.3s	1.3s
9s	30	1.3s	17s
2s	30	1.5s	1.5s

Comme on pouvait s'y attendre, le mode série est plus lent que le mode parallèle (tableau 4.26, 4.27). Cela est dû au fait que YOLO doit préparer chacune des images avant que la série puisse être analysée. Par conséquent, plus la vidéo a de FPS, plus son traitement sera long. D'un autre côté, on peut aussi remarquer que nos deux modèles en parallèle ne sont pas encore capables de traiter nos vidéos en temps réel. Si on se base sur la moyenne de détection, ce n'est pas à proprement parler du temps réel, bien que les vitesses présentées dans le tableau soient relativement rapides.

Pour le mode parallèle, le temps d'exécution pourrait être réduit. En plus de gérer en multithread nos deux modèles, il est aussi possible de paramétrier à l'exécution le nombre d'images que doit analyser **YOLO** pour chaque séquence. Plus ce nombre sera petit et plus la vitesse de traitement sera élevée. Par exemple, en choisissant le nombre 1, chaque image sera analysée, tandis qu'en choisissant 2, une image sur deux le sera, et ainsi de suite.

## 7 Analyse temporelle : la détection d'anomalie

### 7.1 Choix du modèle

Pour choisir notre modèle d'analyse temporelle, nous avons testé plusieurs architectures provenant de modèles reconnus. Ce qui a pour avantage, en plus d'avoir accès au poids déjà entraîné de ceux-ci, d'être sûr de partir sur une architecture ayant déjà fait ses preuves dans un autre domaine plus ou moins proche du nôtre. Parmi cette liste, nous avons des modèles comme **C3D**, convLSTM ou encore des convolutional **RNN** pour lesquels nous avons comparé les effets de **LSTM** face à ceux de **GRU** et fait varier la convolution en sélectionnant à chaque fois des **CNN** ayant fait leurs preuves sur le jeu de données IMAGENET (**VGG19**, Resnet, Inception, etc..) voir section 6.

Nous avons donc commencé par tester 4 variantes :

1. ne pas charger les poids associés
2. charger les poids sans ré-entraîner le modèle
3. charger les poids et ré-entraîner chacune des couches
4. charger les poids et ré-entraîner juste quelques couches

D'après nos premiers résultats indiqués dans le tableau 4.28, l'apprentissage via transfer learning (variantes 3 et 4) est la technique qui donne les meilleures performances. Nous avons donc décidé de poursuivre nos expérimentations en utilisant celle-ci pour chacun de nos modèles.

Tab. 4.28 : Résultats de nos expérimentations

Modèle	Configuration	Accuracy	Précision	Rappel	F1-Score
C3D	Sans poids	82.6%	17.6%	73.3%	28.5%
C3D	Poids UCF Sport Entrainement de tous les blocs	82.9%	21.4%	<b>98.4%</b>	35.2%
C3D	Poids UCF Sport Entrainement du dernier bloc	83.2%	21.8%	98.3%	35.7%
C3D + SVM	Poids UCF Sport Entrainement du dernier bloc	67%	13.3%	98,00%	21.9%
C3D + ConvLSTM	Poids UCF Sport Entrainement du dernier bloc	83.1%	21.6%	98.1%	35.5%
CNN + RNN	VGG19 sans poids + GRU	83.3%	19.6%	84.2%	31.6%
CNN + RNN	VGG19 avec poids IMAGENET + GRU	83.3%	19.9%	85.6%	32.3%
CNN + RNN	VGG19 avec poids IMAGENET Entrainement du dernier bloc + GRU	<b>85.6%</b>	<b>22.9%</b>	89.6%	<b>36.5%</b>

Selon nos expérimentations, les réseaux de type **C3D** ont des performances assez proches de celles des réseaux **CNN + RNN**. Malgré cela les réseaux de type **CNN + RNN** ont montré un meilleur F1-score, c'est-à-dire un meilleur équilibre entre le taux d'alerte et de fausse alerte ; de ce fait nous avons décidé de choisir ce type d'architecture. En plus d'avoir un meilleur F1-score sur notre jeu de données, ces réseaux possèdent une architecture plus modulable, composés de deux réseaux séparés à savoir un **CNN** et un **RNN** pouvant facilement être remplacés par un autre modèle de la même catégorie. Par la suite, nous avons testé plusieurs architectures pour notre **CNN** en utilisant à chaque fois **GRU** pour la partie **RNN**.

Pour la convolution EfficientNet, nous avons testé plusieurs architectures telles que la B1, B2 ou encore la B7, mais aucune d'entre elles ne nous a permis d'obtenir de résultats exploitables. Les performances des différentes architectures de convolution testées, comme Inception, Inception-Resnet, Resnet, **VGG19**, **VGGPlace325** et Xception, sont répertoriées dans le tableau 4.29. Nous n'indiquons dans ce tableau que les résultats pour quelques couches réentraînées.

Tab. 4.29 : Performances des divers réseaux à convolution

Modèle	Couche réentraînée	Accuracy	Précision	Rappel	F1-Score
Inception	0	83.4%	19.8%	82.2%	31.9%
	12	83%	18.1%	73.3%	29%
Inception-Resnet	0	79.8%	16.8%	82.7%	28%
	3	82.7%	18.8%	79.9%	30.4%
	5	82.3%	18.1%	78.1%	29.4%
	12	81.9%	17.3%	76.1%	28.2%
Resnet	0	82.7%	19.9%	87.1%	32.5%
	10	89.4%	27%	73.2%	39.5%
	16	79.8%	15.7%	75.3%	26%
	19	91.2%	31%	79.6%	44.7%
	25	85.7%	21.4%	78.7%	33.7%
	31	87.1%	23.6%	77.5%	36.2%
VGG19	0	83.3%	19.9%	85.6%	32.3%
	3	85%	20.6%	80.4%	32.8%
	6	85.6%	22.9%	89.6%	36.5%
	9	91.6%	33%	83.1%	47.3%
	11	<b>92.2%</b>	<b>35.1%</b>	84.8%	<b>49.7%</b>
	13	84.6%	21.4%	86.8%	34.4%
	18	80.5%	17.1%	83.5%	28.4%
	0	86.9%	22.7%	79.5%	35.3%
VGG-Place325	4	86.6%	23.1%	81%	36%
	8	91%	30.9%	80.4%	44.6%
	10	86.5%	24.2%	86.8%	37.9%
	12	86.2%	22.5%	80.8%	35.2%
	0	79.4%	18.1%	<b>95.8%</b>	30.5%
Xception	6	81.4%	17.3%	77.9%	28.3%
	9	87.7%	23%	75.7%	35.3%
	16	85.1%	21.1%	80.2%	33.4%

Le tableau 4.30 récapitule les meilleures performances de chacun de ces modèles ; le numéro de couche réentraînée indiquée est celui qui a donné les meilleurs résultats. Ce tableau montre que VGG est le réseau à convolution avec lequel nous avons obtenu le meilleur F1-score.

Tab. 4.30 : Récapitulatif

Modèle	Couche réentraînée	Accuracy	Précision	Rappel	F1-Score
Inception	0	83.4%	19.8%	82.2%	31.9%
Inception-Resnet	3	82.7%	18.8%	79.9%	30.4%
Resnet	19	91.2%	31%	79.6%	44.7%
VGG19	11	<b>92.2%</b>	<b>35.1%</b>	<b>84.8%</b>	<b>49.7%</b>
VGG-PLACE325	8	91%	30.9%	80.4%	44.6%
Xception	9	87.7%	23%	75.7%	35.3%

Après avoir choisi **VGG19**, nous avons également exploré la possibilité de retirer une convolution pour voir comment cela affecterait ses performances. Les résultats de cette expérience sont présentés dans le tableau 4.31, et ils montrent clairement qu'on ne peut diminuer le nombre de convolutions.

Tab. 4.31 : Modification de l'architecture de **VGG19**

Modèle	Accuracy	Précision	Rappel	F1-Score
<b>VGG19</b>	<b>92.2%</b>	<b>35.1%</b>	<b>84.8%</b>	<b>49.7%</b>
<b>VGG19</b> (remove last convolution)	83.6%	20.3%	84.6%	32.8%

Nous avons ensuite utilisé cette architecture comme base pour comparer **LSTM** et **GRU**, dans le tableau 4.32, qui fait ressortir un net avantage pour **GRU**.

Tab. 4.32 : Comparaison des performances obtenues avec **LSTM** et avec **GRU**

Modèle	Accuracy	Précision	Rappel	F1-Score
<b>VGG19 + GRU</b>	<b>92.2%</b>	<b>35.1%</b>	<b>84.8%</b>	<b>49.7%</b>
<b>VGG19 + LSTM</b>	87.3%	24.8%	83%	38.2%

Le tableau 4.33 montre les conséquences du choix du nombre de couches totalement connectées nécessaires pour former notre classifieur. Dans la dernière ligne, X représente des résultats non exploitables en raison du sur-apprentissage. Le meilleur choix est donc avec trois couches cachées.

Tab. 4.33 : Recherche du nombre de couche entièrement connecté du classifieur

Couche full connected	Accuracy	Précision	Rappel	F1-Score
2 + output	89.8%	29.1%	84.2%	43.2%
3 + output	<b>92.2%</b>	<b>35.1%</b>	<b>84.8%</b>	<b>49.7%</b>
4 + output	91.2%	33%	83.1%	47.3%
5 + output	X%	X%	X%	X%

Enfin, nous avons utilisé les résultats obtenus ainsi que les techniques d'explicabilité mentionnées précédemment (section 6) pour déterminer le nombre de neurones nécessaires pour notre GRU et le pourcentage d'oubli de nos couches dropout.

## 7.2 Résultats

Dans cette section, nous présenterons nos résultats en matière de détection d'anomalies. Nous commencerons par aborder les performances obtenues pour chacun de nos modèles. À l'heure actuelle, nous disposons de trois modèles distincts pour la détection des bagarres, des coups de feu et des incendies. Nous évaluerons chacun d'eux selon deux aspects : leur capacité à classer des vidéos (une détection par vidéo) et leur capacité à détecter des anomalies dans un flux continu (une détection par séquence). Trois tableaux sont à chaque fois présentés : un tableau des métriques et deux matrices de confusion.

Tab. 4.34 : Performance de la détection des bagarres

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	85.6%	86%	84.9%	85.5%	773
Détection par séquence	63.1%	93.6%	60.3%	73.3%	9865

Matrice de confusion : une détection par vidéo

		Predicted	
		Bagarre	Normal
Truth	Bagarre	86.4%	13.6%
	Normal	15.0%	85.0%

Matrice de confusion : une détection par séquence

		Predicted	
		Bagarre	Normal
Truth	Bagarre	78.3%	21.7%
	Normal	39.8%	60.2%

Tab. 4.35 : Performance de la détection des coups de feu

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	86.5%	95.3%	80.5%	87.3%	134
Détection par séquence	91.8%	35.1%	84.8%	49.7%	13856

Matrice de confusion : une détection par vidéo

Truth	Predicted	
	Coup de feu	Normal
Coup de feu	80.5%	19.5%
Normal	5.3%	94.7%

Matrice de confusion : une détection par séquence

Truth	Predicted	
	Coup de feu	Normal
Coup de feu	84.8%	15.2%
Normal	7.7%	92.3%

Tab. 4.36 : Performance de la détection des incendies

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	83.8%	70.4%	93.9%	80.5%	93
Détection par séquence	86.0%	85.8%	96.2%	90.7%	9718

		Matrice de confusion : une détection par vidéo	
		Predicted	
Truth	Incendie	Incendie	Normal
	Normal	78.4%	21.6%
		6.0%	94.0%

		Matrice de confusion : une détection par séquence	
		Predicted	
Truth	Incendie	Incendie	Normal
	Normal	60.5%	39.5%
		3.7%	96.3%

En observant ces trois tableaux (4.35, 4.34, 4.36), nous pouvons voir que la détection de bagarres (tableau 4.34) est moins efficace dans les flux continus que la détection d'incendies (tableau 4.36). Si l'on se base uniquement sur les performances, la détection des coups de feu (tableau 4.35) est bien moins efficace lorsqu'il s'agit de traiter des flux continus. En revanche, si l'on observe les matrices de confusion, on peut constater une légère augmentation de la fiabilité d'environ 2%. En fait, la détection des coups de feu est celle ayant la meilleure proportion faux positif / faux négatif.

Dans un deuxième temps, nous avons combiné tous nos jeux de données afin de former un unique modèle multi-classe et ainsi comparer ces performances avec chacun des modèles spécialisés présentés précédemment. Notre nouveau jeu de données contient quatre classes (incendie, bagarre, coup de feu, normal) à la répartition déséquilibrée, comme on peut le voir dans la figure 4.2 du chapitre précédent.

Tab. 4.37 : Performance générale du modèle multi-classes

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Une détection par vidéo	81.6%	81.4%	81.6%	81.4%	973
Une détection par séquence	74.2%	85.8%	74.2%	78.6%	31713

Matrice de confusion : une détection par vidéo

Truth \ Predicted	Matrice de confusion : une détection par vidéo			
	Bagarre	Coup de feu	Incendie	Normal
Bagarre	82.9%	2.5%	0.7%	13.9%
Coup de feu	10.3%	58.4%	12.9%	18.4%
Incendie	8.4%	20%	61.6%	10%
Normal	10.3%	0.8%	1.5%	87.4%

Matrice de confusion : une détection par séquence

Truth \ Predicted	Matrice de confusion : une détection par séquence			
	Bagarre	Coup de feu	Incendie	Normal
Bagarre	64%	6%	2%	28%
Coup de feu	11.7%	58.6%	10%	19.7%
Incendie	13.7%	16.6%	55.7%	14%
Normal	7.6%	5.8%	3.9%	82.7%

Malgré le fait qu'il y ait beaucoup plus de données à traiter lors d'une analyse continue, notre modèle semble avoir des performances assez bonnes et assez proche de celui d'un traitement de vidéos finies, comme le montre le tableau 4.37. Afin de faciliter les comparaisons de nos divers modèles, nous avons évalué notre modèle multi-classe sur chaque portion de notre jeu de données dans les tableaux 4.38, 4.39, 4.40. À première vue, un modèle multi-classe est bien plus pratique à utiliser, mais est-il aussi performant que des modèles spécialisés ?

Pour la classe Bagarre nous observons des performances similaires pour la détection de vidéo ainsi qu'une amélioration pour la détection de séquence (flux continu), avec une meilleure détection de la classe normale, comme le montrent les tableaux 4.34, 4.38. Concernant les deux autres classes à savoir Incendie (tableaux 4.36, 4.40) et Coup de feu (tableaux 4.35, 4.39), on constate une baisse de l'accuracy d'environ 10 à 15% pour la classe Incendie allant jusqu'à 20% pour la classe Coup de feu accompagné d'une baisse du F1-Score allant de 5% à 10%.

Tab. 4.38 : Performance sur les bagarres

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	84.9%	87.1%	84.9%	86%	773
Détection par séquence	84.6%	87.8%	84.6%	86.1%	9865

Matrice de confusion : une détection par vidéo

Truth \ Predicted	Matrice de confusion : une détection par vidéo			
	Bagarre	Coup de feu	Incendie	Normal
Bagarre	82.8%	2.6%	0.8%	13.8%
Normal	11.3%	0.8%	0.8%	87.1%

Matrice de confusion : une détection par séquence

Truth \ Predicted	Matrice de confusion : une détection par séquence			
	Bagarre	Coup de feu	Incendie	Normal
Bagarre	64%	6%	2%	28%
Normal	9.9%	0.9%	0.4	88.8%

Tab. 4.39 : Performance sur les coups de feu

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	70.8%	89.4%	70.8%	77.2%	134
Détection par séquence	76%	95.1%	76%	83.8%	13856

Matrice de confusion : une détection par vidéo

		Bagarre	Coup de feu	Incendie	Normal
		Predicted	Truth		
	Coup de feu	10.4%		58.4%	12.9%
	Normal	3.6%		1.7%	7%
					87.7%

Matrice de confusion : une détection par séquence

		Bagarre	Coup de feu	Incendie	Normal
		Predicted	Truth		
	Coup de feu	11.6%		58.5%	10%
	Normal	5%		10.5%	7.6%
					76.9%

Tab. 4.40 : Performance sur les incendies

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score	Élément
Détection par vidéo	69.8%	87.4%	69.8%	76.9%	93
Détection par séquence	76.6%	86.9%	76.6%	81.4%	9718

Matrice de confusion : une détection par vidéo

Truth \ Predicted	Matrice de confusion : une détection par vidéo			
	Bagarre	Coup de feu	Incendie	Normal
Incendie	8.4%	20%	61.6%	10%
Normal	3%	0%	12.2%	84.8%

Matrice de confusion : une détection par séquence

Truth \ Predicted	Matrice de confusion : une détection par séquence			
	Bagarre	Coup de feu	Incendie	Normal
Incendie	13.7%	16.6%	55.7%	14%
Normal	4%	132%	9.7%	85%

Malgré de bonnes performances, notre modèle multi-classes reste inférieur aux modèles spécialisés, en raison du fait qu'il confond parfois les différents types d'incident. Afin de vérifier cela, nous avons donc décidé de former une dernière version de notre modèle en regroupant cette fois-ci toutes nos anomalies au sein d'une même classe dans le but de faciliter son analyse. Nous avons donc la classe normale ne représentant aucun problème ainsi que la classe anormale indiquant un incident nécessitant une intervention regroupant des bagarres, des coups de feu et des incendies. Comme à notre habitude, pour faciliter les comparaisons, nous avons évalué notre nouveau modèle sur chacune des sections de notre jeu de données.

Tab. 4.41 : Performance sur le jeu de données Normal / Anormal

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score
Détection par vidéo	85.71%	82.83%	86.74%	84.74%
Détection par séquence	84.27%	94.82%	85.31%	89.82%

Matrice de confusion : une détection par vidéo

		Predicted	
		Anormal	Normal
Truth	Anormal	84.84%	15.16%
	Normal	13.26%	86.74%

Matrice de confusion : une détection par séquence

		Predicted	
		Anormal	Normal
Truth	Anormal	79.72%	20.28%
	Normal	14.68%	85.32%

Tab. 4.42 : Performance sur le jeu de données Bagarre / Normal

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score
Détection par vidéo	84.59%	83.66%	85.82%	84.73%
Détection par séquence	88.53%	94.45%	91.65%	93.03%

Matrice de confusion : une détection par vidéo

Truth	Predicted	
	Bagarre	Normal
Bagarre	83.37%	16.63%
Normal	14.17%	85.83%

Matrice de confusion : une détection par séquence

Truth	Predicted	
	Bagarre	Normal
Bagarre	72.77%	27.23%
Normal	8.35%	91.65%

Tab. 4.43 : Performance sur le jeu de données Coup de feu / Normal

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score
Détection par vidéo	88.80%	82.81%	92.98%	87.60%
Détection par séquence	79.74%	99.34%	79.26%	88.17%

Matrice de confusion : une détection par vidéo

		Predicted	
		Coup de feu	Normal
Truth	Coup de feu	85.72%	14.28%
	Normal	7.02%	92.98%

Matrice de confusion : une détection par séquence

		Predicted	
		Coup de feu	Normal
Truth	Coup de feu	89.44%	10.56%
	Normal	20.73%	79.27%

Tab. 4.44 : Performance sur le jeu de données Incendie / Normal

Mode d'évaluation	Accuracy	Précision	Rappel	F1-Score
Détection par vidéo	91.39%	87.87%	87.87%	87.87%
Détection par séquence	88.21%	93.24%	90.03%	91.61%

Matrice de confusion : une détection par vidéo

Truth \ Predicted	Incendie	Normal
	Incendie	Normal
Incendie	93.33%	6.67%
Normal	12.12%	87.88%

Matrice de confusion : une détection par séquence

Truth \ Predicted	Incendie	Normal
	Incendie	Normal
Incendie	83.67%	16.33%
Normal	9.96%	90.04%

En comparant ces statistiques (4.37, 4.41) on constate que ce nouveau modèle basé sur deux classes (normal / anormal) est meilleur que notre modèle multi-classes. De plus il est, de manière générale, équivalent voire meilleur que nos modèles spécialisés (tableaux 4.38 vs 4.42, 4.39 vs 4.43, 4.40 vs 4.44). Cela incite à penser que le modèle multi-classe élimine correctement la classe normale, mais se trompe parfois de type d'anomalie.

Dans les cas où un seul type d'anomalie est important, un modèle spécialisé est donc à privilégier. En revanche, dès lors où nous souhaitons traiter plusieurs types d'anomalies, le modèle à utiliser dépendra de la problématique visée. Si nous voulons simplement détecter tout type d'incident et laisser l'humain déduire de quel type il s'agit, un modèle normal / anormal sera un meilleur candidat. Toutefois, si nous souhaitons connaître précisément le type d'anomalie ayant eu lieu, nous devons utiliser un modèle multi-classe. Même si celui-ci est moins efficace que le modèle binaire traitant lui aussi plusieurs types d'anomalies, il n'en demeure pas moins utilisable dans des conditions réelles.

Afin d'évaluer les performances de notre modèle multi-classe dans des situations réelles, nous avons utilisé un ensemble de dix vidéos non éditées, n'appartenant pas à notre jeu de données. Les images présentées ci-dessous sont extraites de ces

vidéos et illustrent les détections du modèle ainsi que leur score, affichés en rouge dans le coin supérieur gauche de chaque image.

Pour la classe "bagarre", nous avons choisi des images représentant des manifestations anti-pass sanitaire incluant des débordements (figure 7.1). Pour la classe "incendie", nous avons choisi l'incendie de la cathédrale de Notre-Dame ainsi qu'un incendie ayant eu lieu en juillet 2022 en Gironde (figure 7.2). N'ayant pas eu d'incident récent en matière de coups de feu, nous avons testé notre modèle sur des vidéos quelconques récupérées sur internet (figure 7.3).

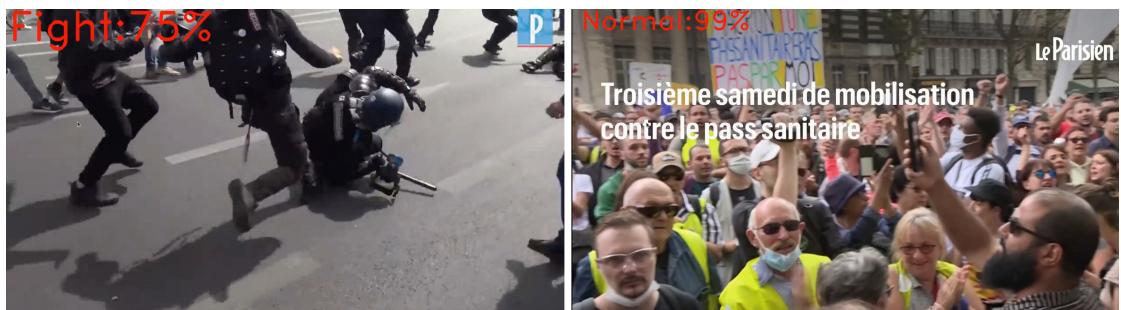


Fig. 7.1 : Détection de bagarre



Fig. 7.2 : Détection d'incendie



Fig. 7.3 : Détection de coup de feu

Le tableau 4.45 indique la vitesse d'exécution de notre modèle multi-classe sur l'ensemble des dix vidéos non découpées décrites ci-dessus. Bien que notre code ne soit pas optimisé, le temps de détection est dans l'ensemble plutôt honorable, étant compris entre 104 et 744 millisecondes.

Tab. 4.45 : Vitesse d'exécution de notre modèle

Durée de la vidéo	FPS de la vidéo	Moyenne de détection	Temps de traitement total
34s	30	468ms	23s
20s	30	500ms	13s
2min 20	25	263ms	46s
4min 56	30	104ms	46s
2min 06	25	128ms	20s
4min 36	30	122ms	50s
20s	30	285ms	8s
1min 30	30	248ms	33s
1min 12	30	140ms	15s
3min 08	30	744ms	1min 56

## 8 Analyse spatiale : la détection d'objet

### 8.1 Choix du modèle

Dans cette section, nous présentons nos expérimentations et résultats en matière de détection d'objet.

Notre première expérimentation a été d'entraîner les architectures **YOLOV3** et **YOLOV4** sur une portion de notre jeu de données contenant uniquement deux

classes à savoir « gun » et « weapons » afin de comparer leurs performances.

Pour cela, nous avons mis au point notre propre algorithme d'évaluation. Cet algorithme commence par charger les labels de référence liés à l'image prédite. Puis il va comparer les box de référence pour chacune des classes avec toutes les détections restantes pour celles-ci après les avoir filtrées via l'opération de **NMS**, dans le but de trouver celle ayant la meilleure **IoU**. Pour rappel, un même objet peut être détecté par plusieurs boites englobantes, le but de l'opération **NMS** est donc de filtrer toutes ces boites englobantes afin de conserver uniquement celles ayant le plus haut score de confiance. Une fois cette boite trouvée, nous calculons l'**IoU** afin de connaître le pourcentage de similitude entre la boite détectée et celle de référence.

Notre algorithme n'est donc pas efficace, ce qui se traduit par une lenteur dans son exécution. Toutefois, durant la phase d'évaluation, la vitesse de traitement n'est pas un enjeu majeur. Nous avons ajouté à notre système certains paramètres comme le taux de superposition et le taux de confiance utilisé lors de l'opération **NMS** ou encore, la possibilité de définir un taux minimal et maximal pour le **IoU** ainsi qu'une taille minimum requise pour les boites prédites. Toute détection ne respectant pas les paramètres fixés pour le **NMS** sera alors considérée comme non-valide et ignorée des statistiques. Concernant les paramètres liés au calcul de l'**IoU**, nous avons ajouté une nouvelle métrique intitulée **badBox** représentant une détection correcte (Vrai Positif) pour laquelle l'encadrement ne correspond pas aux exigences. Ces paramètres nous permettent de voir les performances qu'aurait notre modèle en situation réelle.

Tab. 4.46 : **YOLOV3** : confiance 55%, threshold 70%, IoU min 0% max 100% min box size 0px

Classe	Détecté	Total	Vrai Positif	Faux Positif	Faux Négatif
Gun	3285	3041	72.38%	4.93%	27.62%
Weapons	577	1308	31.19%	3.98%	68.81%

Classe	IoU	Précision	Rappel	F1-Score
Gun	79.04%	93.62%	72.38%	81.64%
Weapons	70.05%	88.7%	31.19%	46.15%

Tab. 4.47 : **YOLOV4** : confiance 55%, threshold 70%, IoU min 0% max 100%, min box size 0px

Classe	DéTECTé	Total	Vrai Positif	Faux Positif	Faux Négatif
Gun	4929	3041	79.61%	6.94%	20.39%
Weapons	1633	1308	52.14%	5.58%	47.68%
Classe	IoU	Précision	Rappel	F1-Score	
Gun	82.22%	91.98%	79.61%	85.35%	
Weapons	75.83%	90.33%	52.14%	66.12%	

La comparaison des tableaux 4.46 et 4.47 montre que la version 4 de **YOLO** est meilleure que la 3, même si elle est plus lente (les temps restent corrects malgré tout). Nous avons donc adopté la version 4 pour la suite de nos expérimentations.

En analysant nos fichiers mal labellisés, nous nous sommes aperçus que **YOLO4** confondait parfois les deux classes, ce qui avait pour conséquence de faire baisser ses performances lors de l'évaluation. Ceci s'explique par le fait que les images de nos deux classes sont assez similaires. Dans notre cas une arme à feu quel que soit son type représente un risque, nous avons donc décidé de fusionner ces deux classes afin d'éviter des baisses de performances.

Tab. 4.48 : Comparaison des performances de **YOLO4** (1 classe) avec **YOLO4** (2 classes)

Modèle	Vrai Positif	Faux Positif	Faux Négatif	
YoloV4 : 2 classes	<b>65.88%</b>	<b>6.26%</b>	<b>34.04%</b>	
YoloV4 : 1 classe	65.05%	18.44%	34.94%	
Modèle	IoU	Précision	Rappel	F1-Score
YoloV4 : 2 classes	<b>79.03%</b>	<b>91.16%</b>	<b>65.88%</b>	<b>75.74%</b>
YoloV4 : 1 classe	70.42%	77.91%	65.05%	70.90%

À première vue la section du tableau 4.48 représentant la fusion de nos classes présente de moins bonnes performances. Mais si on se base sur la courbe générée pendant l'apprentissage, le **Mean Average Precision (MAP)** est largement meilleur (figures 8.1, 8.2).

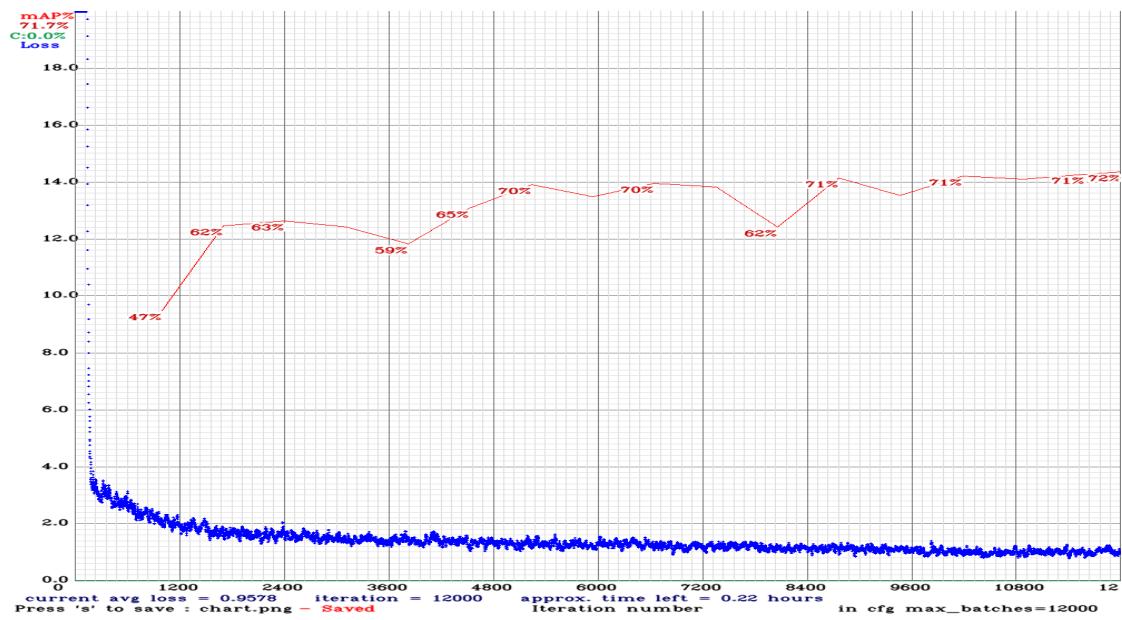


Fig. 8.1 : Courbe d'apprentissage YOLOV4 : 2 classes

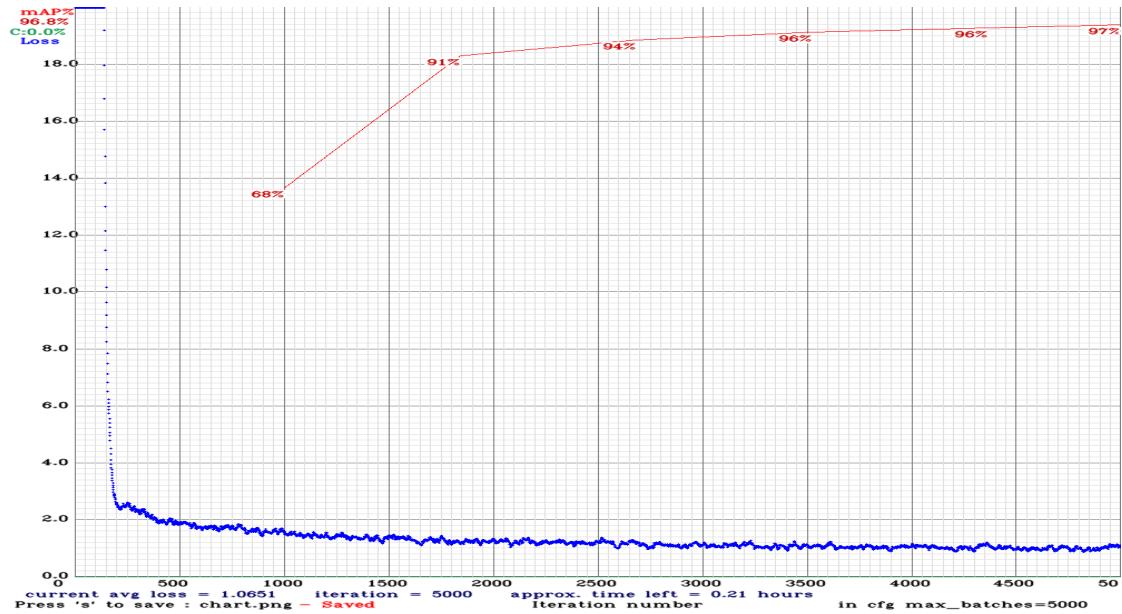


Fig. 8.2 : Courbe d'apprentissage YOLOV4 : 1 classe

## 8.2 Résultats

Après avoir obtenu des résultats satisfaisant avec le modèle **YOLOV4** entraîné sur une seule classe, nous avons évalué les performances de notre modèle dans des conditions de détection d'anomalies en partant de notre idée de base, à savoir si une arme à feu est détectée avec une assez bonne précision alors il s'agira de l'anomalie coup de feu, dans le cas contraire rien à signaler (figures 8.3, 4.49).

Bien évidemment, cette condition est assez subjective et la performance de notre modèle dépend donc de la situation et de sa fiabilité à détecter ces dits objets. Au vu de la matrice de confusion, il semblerait qu'utiliser **YOLO** formé à reconnaître des armes à feu pour réaliser de la détection de coup de feu donne d'assez bonnes performances (tableau 4.49, figure 8.3). Nous avons donc décidé de réaliser la même chose pour nos incendies en entraînant **YOLO** à reconnaître des flammes (tableau 4.50, figure 8.4).

Tab. 4.49 : **YOLO** : détection de coups de feu, confiance à 55%

		Accuracy	Précision	Rappel	F1-Score
		80%	16.07%	81.19%	27.79%
Truth	Predicted	Matrice de confusion			
		Coup de Feu	Normal		
Coup de feu		81.12%	18.88%		
Normal		20%	80%		

Tab. 4.50 : **YOLO** : détection d'incendies, confiance à 55%

		Accuracy	Précision	Rappel	F1-Score
		90.7%	89.1%	99.2%	93.8%
Truth	Predicted	Matrice de confusion			
		Incendie	Normal		
Incendie		69.5%	30.5%		
Normal		0.7%	99.3%		



Fig. 8.3 : Détection d'armes à feu via YOLOV4



Fig. 8.4 : Détection de flamme via YOLOV4

## 9 Conclusion

Nos tests nous ont permis de définir un ensemble de modèles dont les points communs sont : le choix de **YOLOV7** (même si une partie de nos tests ont été faits antérieurement à la parution de cette version, et utilisaient la version V4) ; le choix de **GRU** et de la convolution **VGG19** pour notre **CRNN** ; le choix de fusionner certaines de nos classes dans notre jeu de données ; et le choix de la tracabilité. Ceci laisse un certain nombre de possibilités de variation, en particulier le mode série ou parallèle et le choix du mode multi-classe ou binaire.

Concernant ces variations, notre étude nous permet de proposer les conseils suivants :

1. Si l'objectif est de détecter le type d'anomalie ayant eu lieu, les meilleurs résultats sont obtenus en utilisant **YOLOV7** ainsi qu'un **CRNN** multi-classe en mode série. Dans le cas où les anomalies seraient liées à des comportements

humains, le choix d'effectuer comme pré-traitement une estimation de la pose en retirant l'arrière-plan permettra d'obtenir de très bons résultats.

2. Pour des problématiques où le temps réel est important, le meilleur compromis fiabilité/vitesse est obtenu en utilisant **YOLOV7** et notre **CRNN** dans sa version **GRU**, en mode parallèle, avec les règles suivantes :
  - Si une anomalie est prédite, et qu'une flamme est détectée alors l'événement en cours est un incendie.
  - Si une anomalie est prédite, et qu'une personne est détectée avec une arme à proximité, il s'agit d'un coup de feu.
3. Si l'objectif consiste uniquement à avertir d'un danger quel qu'il soit, il faut privilégier l'utilisation d'un modèle binaire (normal / anormal) au lieu d'un modèle multi-classe. Bien qu'il soit moins précis sur le type d'anomalie en cours, ce type de modèle a obtenu les meilleures performances, tout modèle confondu, à condition de le combiner avec **YOLO** (en parallèle ou en série en fonction des besoins).







# Conclusion

Cette conclusion sera séparée en trois parties : la première sera une synthèse générale de mon travail, la deuxième évoquera ses limitations et la dernière détaillera diverses perspectives à envisager.

## 10 Synthèse générale

De nos jours, de nombreux lieux sont munis de caméras de surveillance visant à garantir notre sécurité. Toutes ces caméras ne peuvent être analysées en temps réel et sont plutôt utilisées comme moyen de dissuasion exploitées une fois les événements terminés. Pendant l'analyse en direct de ces images, le surveillant est en général responsable de plusieurs caméras. Cette tâche de surveillance représente donc une grosse charge de travail et dépend de l'attention du surveillant.

Nous avons proposé ici un ensemble de modèles qui forment un système de détection d'anomalies dans un temps relativement court à partir de vidéos ou de flux continus.

Pour répondre à cette problématique, j'ai exploré deux domaines liés à la vision par ordinateur : la reconnaissance d'action et la détection d'objets. Mon idée était de combiner ces deux techniques et ainsi se rapprocher d'un comportement humain nous permettant de détecter si une anomalie a lieu, de quel type il s'agit et de la localiser dans l'image et la séquence.

Au cours de nos recherches, nous avons pu observer deux principaux cas conduisant à des anomalies. L'un d'eux est lié aux actions réalisées par les divers objets ou entités présentes à l'écran, l'autre est rattaché à des objets particuliers qui, de par leur seule présence, amène à soupçonner une anomalie. Notre objectif consiste donc à repérer les divers objets présents à l'écran puis à analyser le comportement de ceux-ci afin de détecter d'éventuelles anomalies.

Dans ce projet doctoral, nous nous sommes concentrés sur des approches supervisées. Nous avons donc conçu deux jeux de données. Le premier est constitué d'images représentant des armes à feu, des personnes et des flammes, le second

regroupant des vidéos représentant des incendies, des bagarres et des coups de feu.

Le système proposé est composé de deux modèles pouvant être disposés en parallèle ou en série selon les besoins. Pour la détection d'objets, nous avons choisi **YOLOV7** pour ses performances en termes de vitesse d'exécution et sa capacité à traiter divers problèmes liés au traitement d'image, y compris l'analyse de la pose humaine. Du côté de la détection d'anomalies, nous avons opté pour un **Convolutional Recurrent Neuronal Network (CRNN)** composé de **VGG19** et d'un **GRU**, modèle que nous avons entraîné à l'aide d'un vidéo générateur sur des données vidéos finies, chacune résumée par une seule séquence d'images et testée par la suite pour effectuer de la classification de vidéos ou de séquences vidéos.

Il peut réaliser une détection par vidéo en mode batch ou une détection par séquence de 20 images dans un flux continu. Le système présenté est donc capable de fournir de très bons résultats, atteignant jusqu'à 90% de précision et de rappel dans certains cas. Malheureusement, celui-ci étant un système de détection, l'anomalie est détectée *a posteriori*. Il n'opère donc pas en temps réel, mais ses temps de réponse sont assez rapides quelle que soit l'architecture utilisée. En série, nous obtenons un taux de rafraîchissement moyen situé entre 1 et 1,5 seconde, et un taux inférieur à 1,2 seconde pour le mode parallèle. Cette vitesse pourrait encore être améliorée pour le mode parallèle en configurant **YOLO** de manière à ce qu'il ignore certaines images de la séquence. Par exemple, en analysant une image sur deux au lieu de toutes les images, ce qui est clairement une perte de temps pour des flux continus où la différence inter-image n'est que minimale. Malgré cela, quel que soit le mode utilisé (parallèle ou en série), les vitesses d'exécution restent largement inférieures au temps de réaction d'un surveillant.

## 11 Limitation de mon travail

Dans cette thèse, très peu de classes ont été utilisées pour l'entraînement du modèle, cependant, grâce à notre architecture (**CGRU**), il est également possible d'obtenir de bons résultats sur d'autres types d'anomalies, telles que les accidents de voiture comme le demonstre la figure 11.1, à condition d'utiliser un jeu de données adéquat.



Fig. 11.1 : Détection d'accident

Malgré de bonnes performances de la part de notre système et des temps de traitements acceptables pouvant encore être améliorés, nous n'avons pas été en mesure à l'heure actuelle de cibler la zone où ont lieu nos anomalies.

Nous avons également constaté que dans certains cas précis, les conditions établies visant à réduire le taux de faux négatifs n'étaient pas adéquates. Nous souhaitions, en mode parallèle, changer l'étiquette de toute séquence étiquetée comme normale si une flamme ou une personne avec une arme apparente était détectée. Malheureusement, sous certaines conditions, ces comportements peuvent être considérés comme normaux, c'est par exemple le cas dans des aéroports où il est fréquent de croiser des militaires armés, situation ne représentant pas obligatoirement un risque.

Pour finir, nous tenons à souligner que durant ce projet doctoral nous n'avons pas pu tester notre système sur des vidéos issues de caméras de surveillance. Nos performances ont donc toutes été mesurées en fonction de vidéos provenant de notre jeu de test. De plus, dû au fait que la collecte et le nettoyage de données représentent une tâche longue et coûteuse, nous avons formé notre modèle en utilisant uniquement 3 classes. Chaque nouvelle anomalie que nous souhaiterions ajouter nécessitera donc de collecter de nouvelles vidéos voir de nouvelles images, données pouvant être difficiles à trouver dans certains contextes. De plus, dans des cas d'utilisation de nos modèles en parallèle, il sera aussi parfois nécessaire d'élaborer de nouvelles règles de détection.

## 12 Perspectives

Une première amélioration envisageable consiste à remplacer le calcul d'**IoU** entre une personne et une arme à feu dans notre architecture en parallèle par un calcul de **Distance Intersection over Union (DIoU)**, qui prendrait en compte les centres de chacune de nos boîtes englobantes. Cette modification permettrait d'éviter de déclencher une alerte si une arme est détectée mais simplement portée.

Nous pourrions par ailleurs remplacer les règles définies manuellement dans le mode parallèle en optant pour un modèle d'apprentissage automatique tel qu'une Isolation Forest spécialisée dans la détection d'anomalies ou encore un **MLP** accompagné de suréchantillonnage, par exemple. Ces modèles pourraient prendre en entrée la sortie de notre **CGRU** ainsi que celle de **YOLO** et apporteraient une plus grande fiabilité tout en simplifiant l'ajout de nouvelles classes.

Une piste peu explorée dans notre travail consiste à utiliser l'analyse spatiale en parallèle pour modifier l'étiquette de certaines images lorsqu'un objet suspect est détecté, plutôt que de modifier l'étiquette de l'ensemble de la séquence (ce qui est le cas de l'actuel mode parallèle).

Il serait envisageable d'explorer d'autres pistes pour améliorer notre travail, telles que l'utilisation de modèles différents ou d'approches innovantes. Par exemple, nous pourrions évaluer les performances d'un modèle de type **ViT**, qui possède son propre mécanisme d'attention, en comparaison avec notre **CRNN**. Il serait également intéressant d'explorer l'utilisation de réseaux de neurones non supervisés, tels que des auto-encodeurs ou **DINO** (V1, V2), pour éviter la collecte et l'étiquetage de nouvelles données à chaque ajout de nouvelles classes.

Par ailleurs, dans l'architecture actuelle, une idée à tester serait de remplacer la convolution **VGG** par **YOLO** au sein de notre **CRNN**. **YOLO** est une architecture de réseau neuronal de détection d'objets basée sur des convolutions, qui pourrait fournir des informations supplémentaires sur les objets détectés et leurs positions dans chaque image de la vidéo. Cette approche pourrait améliorer la capacité de notre modèle à capturer les caractéristiques visuelles de chaque instant de la vidéo, et ainsi potentiellement améliorer ses performances.

Dans cette thèse, nous n'avons pas exploré l'utilisation de la segmentation d'image en raison des contraintes de quasi-temps réel. Cependant, cette idée pourrait être intéressante à tester dans des cas de détection d'anomalies où la rapidité de traitement n'est pas cruciale. Le but serait de vérifier si ce prétraitement permet d'améliorer les résultats obtenus. Pour évaluer cette approche, nous pourrions utiliser l'architecture **YOLOv7** spécialisée dans ce type de traitement ou d'autres modèles tels que Faster-**RCNN** déjà évoqués dans cette thèse ou encore **Segment Anything Mode (SAM)**, un modèle de segmentation open source proposé récem-

ment par Meta (Kirillov et al., 2023).

En ce qui concerne l'explicabilité, une approche intéressante consisterait à y intégrer de la détection d'objets. L'intégration de YOLO avec nos techniques de visualisation, telles que nos cartes d'activation, pourrait nous aider à mieux comprendre les décisions de notre modèle et nous permettre de mieux représenter la zone où l'incident détecté a eu lieu.

Pour finir, malgré le fait que toutes les vidéos ne contiennent pas de son, coupler notre approche avec une analyse de données audio permettrait, dans certains cas, d'identifier avec plus de fiabilité certaines anomalies telles que des coups de feu, des explosions ou même des accidents de voiture.



# Glossaire

**Complete Intersection over Union (CIoU)** : Il s'agit d'une amélioration de l'IoU qui prend en compte la distance entre les centres de la région prédite et de l'objet réel ainsi que leur orientation. CIoU offre une mesure plus précise de la qualité des détections d'objets.. [55](#)

**Deformable Parts Model (DPM)** : est un algorithme de détection d'objets utilisant des modèles de parties déformables pour représenter des objets dans une image, ce qui permet de gérer les variations de forme et de taille dans les données d'apprentissage.. [50](#)

**Distance Intersection over Union (DIoU)** : Il s'agit d'une variante de l'IoU qui mesure la similitude entre deux ensembles en utilisant la distance entre les centres des régions ainsi que leur intersection.. [144](#)

**Distance Intersection over Union Non-Max Suppression (DIoU NMS)** : Il s'agit d'une combinaison de DIoU et NMS pour éliminer les régions superposées tout en utilisant une mesure de similitude plus précise. [55](#)

**DIstillation with NO label (DINO)** : Technique d'apprentissage où un modèle appelé modèle maître est formé pour transférer sa connaissance à un modèle plus petit appelé modèle esclave. Le modèle esclave apprend à imiter les prédictions du modèle maître en utilisant les prédictions du modèle maître comme supervision et non les étiquettes de données d'apprentissage.. [46, 91, 144](#)

**Histogram of Oriented Gradients (HOG)** : est une technique de descripteur d'images utilisée en vision par ordinateur pour détecter les contours d'objets dans une image. Il utilise une histogramme des gradients orientés pour représenter les caractéristiques locales de l'image.. [50](#)

**Intersection over Union (IoU)** : Mesure de la similitude entre deux ensembles utilisé pour évaluer la qualité des détections d'objets ; IoU mesure la propor-

tion d'intersection entre la région prédite par le modèle et la région réelle de l'objet.. [52](#), [55](#), [77](#), [80](#), [101–103](#), [132](#), [133](#), [144](#)

**Mean Average Precision (MAP)** : Mesure globale de la qualité des détections d'objets qui prend en compte à la fois la précision (nombre de détections correctes) et le rappel (nombre de détections sur le nombre total d'objets). La moyenne des précisions pour chaque classe est utilisée pour évaluer le modèle de détection d'objet.). [133](#)

**Non-Max Suppression (NMS)** : Technique utilisée en détection d'objets pour éliminer les régions superposées (ou les régions dupliquées) en gardant seulement la région avec la plus grande probabilité d'être l'objet recherché.. [52](#), [55](#), [132](#)

# Mes publications

- [61] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Détection d'anomalies en temps réel dans le flux vidéo ». In : *Extraction et Gestion des Connaisances : EGC'2022* 38 (2022) (cf. p. 8).
- [62] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Enhancing Anomaly Detection in Videos using a Combined YOLO and a VGG GRU Approach ». International Conference on Computer Systems & Applications. Juin 2023 (cf. p. 8).
- [63] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Visualize the training of a time distributed convolution ». Article soumis au journal Engineering Applications of Artificial Intelligence. Jan. 2023 (cf. p. 8).



# Bibliographie

- [1] J. An et S. Cho. « Variational autoencoder based anomaly detection using reconstruction probability ». In : *Special Lecture on IE* 2.1 (2015), p. 1-18 (cf. p. 41).
- [2] S. Arif, J. Wang, T. U. Hassan et Z. Fei. « 3D-CNN-Based Fused Feature Maps with LSTM Applied to Action Recognition ». In : *Future Internet* 11 (2019), p. 42 (cf. p. 31).
- [3] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic et C. Schmid. « ViViT : A Video Vision Transformer ». In : *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), p. 6816-6826 (cf. p. 46).
- [4] N. Barla. *A Comprehensive Guide to Human Pose Estimation*. 2022. url : <https://www.v7labs.com/blog/human-pose-estimation-guide> (cf. p. 96).
- [5] D. Bau, J.-Y. Zhu, H. Strobelt, À. Lapedriza, B. Zhou et A. Torralba. « Understanding the role of individual units in a deep neural network ». In : *Proceedings of the National Academy of Sciences* 117 (2020), p. 30071-30078 (cf. p. 32).
- [6] A. Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao. « YOLOv4 : Optimal Speed and Accuracy of Object Detection ». In : *ArXiv* abs/2004.10934 (2020) (cf. p. 54, 56, 165).
- [7] L. Cai, Y. Chen, N. Cai, W. Cheng et H. Wang. « Utilizing Amari-Alpha Divergence to Stabilize the Training of Generative Adversarial Networks ». In : *Entropy* 22 (2020) (cf. p. 49, 165).
- [8] M. Caron, H. Touvron, I. Misra, H. J'egou, J. Mairal, P. Bojanowski et A. Joulin. « Emerging Properties in Self-Supervised Vision Transformers ». In : *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), p. 9630-9640 (cf. p. 46, 48, 165).
- [9] A. Chakraborty, A. Sharma, C. Pancholi, J. Arora, J. Grover et P. Agarwal. *ANOMALY DETECTION IN VIDEO FEEDS*. url : <https://ash-shar.github.io/reports/ML-Report-Anomaly-Detection.pdf> (cf. p. 19).

- [10] V. Chandola, A. Banerjee et V. Kumar. « Anomaly detection : A survey ». In : *ACM Comput. Surv.* 41 (2009), 15 :1-15 :58 (cf. p. 20).
- [11] A. Chattopadhyay, A. Sarkar, P. Howlader et V. N. Balasubramanian. « Grad-CAM++ : Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks ». In : *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), p. 839-847 (cf. p. 62).
- [12] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng et J. Sun. « You Only Look One-level Feature ». In : *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), p. 13034-13043 (cf. p. 56, 165).
- [13] Z. Chen, C. Yeo, B.-S. Lee et C. T. Lau. « Autoencoder-based network anomaly detection ». In : *2018 Wireless Telecommunications Symposium (WTS)* (2018), p. 1-5 (cf. p. 41).
- [14] F. Chollet. « Xception : Deep Learning with Depthwise Separable Convolutions ». In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 1800-1807 (cf. p. 38).
- [15] F. Chollet et al. *Keras*. 2015. url : <https://github.com/fchollet/keras> (cf. p. 42, 63, 165).
- [16] Y. S. Chong et Y. H. Tay. « Abnormal Event Detection in Videos using Spatiotemporal Autoencoder ». In : *International Symposium on Neural Networks*. 2017. doi : [10.1007/978-3-319-59081-3\\_23](https://doi.org/10.1007/978-3-319-59081-3_23) (cf. p. 41).
- [17] K. Deepak, S. Chandrakala et C. K. Mohan. « Residual spatiotemporal autoencoder for unsupervised video anomaly detection ». In : *Signal, Image and Video Processing* 15 (2020), p. 215-222 (cf. p. 41).
- [18] A. Dimokranitou. « Adversarial autoencoders for anomalous event detection in images ». Thèse de doct. Purdue University, 2017 (cf. p. 48).
- [19] K. Doshi et Y. Yilmaz. « Continual Learning for Anomaly Detection in Surveillance Videos ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2020), p. 1025-1034 (cf. p. 22, 64).
- [20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit et N. Houlsby. « An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale ». In : *ArXiv* abs/2010.11929 (2020) (cf. p. 46).
- [21] J. L. Elman. « Finding Structure in Time ». In : *Cogn. Sci.* 14 (1990), p. 179-211 (cf. p. 36).

- [22] Y. Fang, B. Liao, X. Wang, J. Fang, J. Qi, R. Wu, J. Niu et W. Liu. « You only look at one sequence : Rethinking transformer in vision through object detection ». In : *Advances in Neural Information Processing Systems* 34 (2021), p. 26183-26197 (cf. p. 58, 59, 166).
- [23] P. Ferlet. *How to work with Time Distributed data in a neural network*. 2019. url : <https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00> (cf. p. 87, 166).
- [24] P. Ferlet. *keras-video-generators*. 2020. url : <https://github.com/metal3d/keras-video-generators>.
- [25] R. Gandhi. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. 2018. url : <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e> (cf. p. 64).
- [26] Z. Ge, S. Liu, F. Wang, Z. Li et J. Sun. « YOLOX : Exceeding YOLO Series in 2021 ». In : *ArXiv* abs/2107.08430 (2021) (cf. p. 59).
- [27] F. A. Gers, J. Schmidhuber et F. Cummins. « Learning to Forget : Continual Prediction with LSTM ». In : *Neural Computation* 12 (2000), p. 2451-2471 (cf. p. 36).
- [28] R. Ghosh. <https://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review>. 2018. url : <https://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review> (cf. p. 40, 49, 165).
- [29] R. B. Girshick. « Fast R-CNN ». In : *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), p. 1440-1448 (cf. p. 51).
- [30] R. B. Girshick, J. Donahue, T. Darrell et J. Malik. « Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation ». In : *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2013), p. 580-587 (cf. p. 50, 51, 165).
- [31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville et Y. Bengio. *Generative Adversarial Networks*. 2014. arXiv : [1406.2661 \[stat.ML\]](https://arxiv.org/abs/1406.2661) (cf. p. 48).
- [32] K. Gotkowski, C. González, A. M. Bucher et A. Mukhopadhyay. « M3d-CAM : A PyTorch library to generate 3D data attention maps for medical deep learning ». In : *ArXiv* abs/2007.00453 (2020) (cf. p. 63).

- [33] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury et L. S. Davis. « Learning Temporal Regularity in Video Sequences ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 733-742 (cf. p. 41).
- [34] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li et H. Shi. « Escaping the Big Data Paradigm with Compact Transformers ». In : *ArXiv* abs/2104.05704 (2021) (cf. p. 47, 165).
- [35] K. He, X. Zhang, S. Ren et J. Sun. « Deep Residual Learning for Image Recognition ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), p. 770-778 (cf. p. 38).
- [36] S. Hochreiter et J. Schmidhuber. « Long Short-Term Memory ». In : *Neural Computation* 9 (1997), p. 1735-1780 (cf. p. 36).
- [37] G. Huang, Z. Liu et K. Q. Weinberger. « Densely Connected Convolutional Networks ». In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 2261-2269 (cf. p. 54, 55, 165).
- [38] S. Jacob. *Anomaly detection from videos : A deep learning approach*. McGill University (Canada), 2019 (cf. p. 30, 36, 41).
- [39] H. M. B. Jahlan et L. A. Elrefaei. « Mobile Neural Architecture Search Network and Convolutional Long Short-Term Memory-Based Deep Features Toward Detecting Violence from Video ». In : *Arabian Journal for Science and Engineering* 46 (2021), p. 8549-8563 (cf. p. 39).
- [40] H. Jain et S. Nandy. « Incremental Training for Image Classification of Unseen Objects ». In : *ResearchGate*, Aug (2019) (cf. p. 53, 165).
- [41] S. Ji, W. Xu, M. Yang et K. Yu. « 3D Convolutional Neural Networks for Human Action Recognition ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2010), p. 221-231 (cf. p. 39, 40, 165).
- [42] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng et Y. Wei. « LayerCAM : Exploring Hierarchical Class Activation Maps for Localization ». In : *IEEE Transactions on Image Processing* 30 (2021), p. 5875-5888 (cf. p. 62).
- [43] M. I. Jordan. « Serial Order : A Parallel Distributed Processing Approach ». In : *Advances in psychology* 121 (1997), p. 471-495 (cf. p. 36).
- [44] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár et R. B. Girshick. « Segment Anything ». In : *ArXiv* abs/2304.02643 (2023) (cf. p. 145).
- [45] R. Kotikalapudi et al.. *keras-vis*. <https://github.com/raghakot/keras-vis>. 2017 (cf. p. 62).

- [46] A. Krizhevsky, I. Sutskever et G. E. Hinton. « ImageNet classification with deep convolutional neural networks ». In : *Communications of the ACM* 60 (2012), p. 84-90 (cf. p. 50).
- [47] M. Lagunas et E. Garces. « Transfer Learning for Illustration Classification ». In : *ArXiv* abs/1806.02682 (2018) (cf. p. 38).
- [48] C. S. Lea, M. D. Flynn, R. Vidal, A. Reiter et G. Hager. « Temporal Convolutional Networks for Action Segmentation and Detection ». In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 1003-1012 (cf. p. 43, 44, 165).
- [49] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard et L. D. Jackel. « Backpropagation Applied to Handwritten Zip Code Recognition ». In : *Neural Computation* 1 (1989), p. 541-551 (cf. p. 37).
- [50] W. Lin, J. Gao, Q. Wang et X. Li. « Learning to detect anomaly events in crowd scenes from synthetic data ». In : *Neurocomputing* 436 (2021), p. 248-259 (cf. p. 48).
- [51] S. M. Lundberg et S.-I. Lee. « A Unified Approach to Interpreting Model Predictions ». In : *ArXiv* abs/1705.07874 (2017) (cf. p. 62).
- [52] W. Luo, W. Liu et S. Gao. « Remembering history with convolutional LSTM for anomaly detection ». In : *2017 IEEE International Conference on Multimedia and Expo (ICME)* (2017), p. 439-444 (cf. p. 41).
- [53] M. Majd et R. Safabakhsh. « A motion-aware ConvLSTM network for action recognition ». In : *Applied Intelligence* 49 (2019), p. 2515-2521 (cf. p. 39).
- [54] J. R. Medel et A. E. Savakis. « Anomaly Detection in Video Using Predictive Convolutional Long Short-Term Memory Networks ». In : *ArXiv* abs/1612.00390 (2016) (cf. p. 41).
- [55] B. Mishra, D. Garg, P. Narang et V. K. Mishra. « A hybrid approach for search and rescue using 3DCNN and PSO ». In : *Neural Computing and Applications* (2020), p. 1-15 (cf. p. 39).
- [56] C. Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable.* 2019 (cf. p. 63).
- [57] M. Mustafa.. « Real-Time Video Anomaly Detection for Smart Surveillance ». In : *SSRN Electronic Journal* (2022) (cf. p. 22).
- [58] C. Olah, A. Mordvintsev et L. Schubert. « Feature Visualization ». In : *Distill* (2017). <https://distill.pub/2017/feature-visualization>. doi : [10.23915/distill.00007](https://doi.org/10.23915/distill.00007) (cf. p. 32).

- [59] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye et A. Mordvintsev. « The Building Blocks of Interpretability ». In : *Distill* (2018). <https://distill.pub/2018/building-blocks>. doi : [10.23915/distill.00010](https://doi.org/10.23915/distill.00010) (cf. p. 32).
- [60] J. P. de Oliveira Lima et C. M. S. Figueiredo. « Temporal Fusion Approach for Video Classification with Convolutional and LSTM Neural Networks Applied to Violence Detection ». In : *Inteligencia Artif.* 24 (2021), p. 40-50 (cf. p. 39).
- [61] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Détection d'anomalies en temps réel dans le flux vidéo ». In : *Extraction et Gestion des Connaissances : EGC'2022* 38 (2022) (cf. p. 8).
- [62] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Enhancing Anomaly Detection in Videos using a Combined YOLO and a VGG GRU Approach ». International Conference on Computer Systems & Applications. Juin 2023 (cf. p. 8).
- [63] F. Poirier, R. Jaziri, C. Srour et G. Bernard. « Visualize the training of a time distributed convolution ». Article soumis au journal Engineering Applications of Artificial Intelligence. Jan. 2023 (cf. p. 8).
- [64] A. Ravi et F. Karray. « Exploring Convolutional Recurrent architectures for anomaly detection in videos : a comparative study ». In : *Discover Artificial Intelligence* 1 (2021) (cf. p. 39).
- [65] J. Redmon, S. K. Divvala, R. B. Girshick et A. Farhadi. « You Only Look Once : Unified, Real-Time Object Detection ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), p. 779-788 (cf. p. 52, 53, 165).
- [66] J. Redmon et A. Farhadi. « YOLO9000 : Better, Faster, Stronger ». In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 6517-6525 (cf. p. 53).
- [67] J. Redmon et A. Farhadi. « YOLOv3 : An Incremental Improvement ». In : *ArXiv* abs/1804.02767 (2018) (cf. p. 54).
- [68] P. Remy. *Keract : A library for visualizing activations and gradients*. <https://github.com/philipperemy/keract>. 2020 (cf. p. 63).
- [69] S. Ren, K. He, R. B. Girshick et J. Sun. « Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), p. 1137-1149 (cf. p. 51).

- [70] M. Ribeiro, A. E. Lazzaretti et H. S. Lopes. « A study of deep convolutional auto-encoders for anomaly detection in videos ». In : *Pattern Recognit. Lett.* 105 (2018), p. 13-22 (cf. p. 41).
- [71] M. T. Ribeiro, S. Singh et C. Guestrin. « “Why Should I Trust You ?” : Explaining the Predictions of Any Classifier ». In : *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016) (cf. p. 62).
- [72] D. E. Rumelhart, G. E. Hinton et R. J. Williams. « Learning representations by back-propagating errors ». In : *Nature* 323 (1986), p. 533-536 (cf. p. 41).
- [73] T. Schlegl, P. Seeböck, S. M. Waldstein, U. M. Schmidt-Erfurth et G. Langs. « Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery ». In : *ArXiv* abs/1703.05921 (2017) (cf. p. 48).
- [74] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh et D. Batra. « Grad-CAM : Visual Explanations from Deep Networks via Gradient-Based Localization ». In : *International Journal of Computer Vision* 128 (2016), p. 336-359 (cf. p. 62).
- [75] P. Sernani, N. Falcionelli, S. Tomassini, P. Contardo et A. F. Dragoni. « Deep Learning for Automatic Violence Detection : Tests on the AIRTLab Dataset ». In : *IEEE Access* 9 (2021), p. 160580-160595 (cf. p. 29).
- [76] P. Sharma. *A Practical Guide to Object Detection using the Popular YOLO Framework*. 2018. url : <https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framework-python/> (cf. p. 62).
- [77] P. Sharma. *A Practical Implementation of the Faster R-CNN Algorithm for Object Detection*. 2018. url : <https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/> (cf. p. 62).
- [78] P. Sharma. *A Step-by-Step Introduction to the Basic Object Detection Algorithms*. 2018. url : <https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/> (cf. p. 61).
- [79] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W.-K. Wong et W.-c. Woo. « Convolutional LSTM Network : A Machine Learning Approach for Precipitation Nowcasting ». In : *NIPS*. 2015 (cf. p. 38, 39).
- [80] K. Simonyan, A. Vedaldi et A. Zisserman. « Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps ». In : *CoRR* abs/1312.6034 (2013) (cf. p. 62).

- [81] K. Simonyan et A. Zisserman. « Very Deep Convolutional Networks for Large-Scale Image Recognition ». In : *CoRR* abs/1409.1556 (2014) (cf. p. 37).
- [82] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas et M. Wattenberg. « SmoothGrad : removing noise by adding noise ». In : *ArXiv* abs/1706.03825 (2017) (cf. p. 62).
- [83] J. Sublime. « L'apprentissage non-supervisé et ses contradictions ». In : *Bulletin 1024* (2022) (cf. p. 41, 165).
- [84] W. Sultani, C. Chen et M. Shah. « Real-world anomaly detection in surveillance videos ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 6479-6488 (cf. p. 26).
- [85] C. Szegedy, S. Ioffe, V. Vanhoucke et A. A. Alemi. « Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning ». In : *ArXiv* abs/1602.07261 (2016) (cf. p. 38).
- [86] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke et A. Rabinovich. « Going deeper with convolutions ». In : *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), p. 1-9 (cf. p. 38).
- [87] M. Tan et Q. V. Le. « EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks ». In : *ArXiv* abs/1905.11946 (2019) (cf. p. 38).
- [88] T. Toharudin, R. S. Pontoh, R. E. Caraka, S. Zahroh, Y. Lee et R. C. Chen. « Employing long short-term memory and Facebook prophet model in air temperature forecasting ». In : *Communications in Statistics - Simulation and Computation* 52 (2021), p. 279-290 (cf. p. 37, 165).
- [89] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani et M. Paluri. « Learning Spatiotemporal Features with 3D Convolutional Networks ». In : *2015 IEEE International Conference on Computer Vision (ICCV)* (2014), p. 4489-4497 (cf. p. 39).
- [90] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers et A. W. M. Smeulders. « Selective Search for Object Recognition ». In : *International Journal of Computer Vision* 104 (2013), p. 154-171 (cf. p. 51).
- [91] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser et I. Polosukhin. « Attention is all you need ». In : *Advances in neural information processing systems* 30 (2017) (cf. p. 44, 45, 165).

- [92] P. A. Viola et M. J. Jones. « Rapid object detection using a boosted cascade of simple features ». In : *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* 1 (2001), p. I-I (cf. p. 49, 50, 165).
- [93] R. Vrskova, R. Hudec, P. Kamencay et P. Sykora. « A New Approach for Abnormal Human Activities Recognition Based on ConvLSTM Architecture ». In : *Sensors (Basel, Switzerland)* 22 (2022) (cf. p. 29, 39).
- [94] R. Vrskova, R. Hudec, P. Kamencay et P. Sykora. « Human Activity Classification Using the 3DCNN Architecture ». In : *Applied Sciences* (2022) (cf. p. 39).
- [95] R. Vrskova, R. Hudec, P. Sykora, P. Kamencay et M. Benco. « Violent Behavioral Activity Classification Using Artificial Neural Network ». In : *2020 New Trends in Signal Processing (NTSP)* (2020), p. 1-5 (cf. p. 39).
- [96] C.-Y. Wang, A. Bochkovskiy et H.-Y. M. Liao. « YOLOv7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors ». In : *ArXiv abs/2207.02696* (2022) (cf. p. 60, 61, 166).
- [97] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen et J.-W. Hsieh. « CSPNet : A New Backbone that can Enhance Learning Capability of CNN ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), p. 1571-1580 (cf. p. 55, 165).
- [98] C.-Y. Wang, I.-H. Yeh et H. Liao. « You Only Learn One Representation : Unified Network for Multiple Tasks ». In : *J. Inf. Sci. Eng.* 39 (2021), p. 691-709 (cf. p. 58, 165).
- [99] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. ( Mardziel et X. Hu. « Score-CAM : Score-Weighted Visual Explanations for Convolutional Neural Networks ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), p. 111-119 (cf. p. 62).
- [100] L. Wang, F. Zhou, Z. Li, W. Zuo et H. Tan. « Abnormal Event Detection in Videos Using Hybrid Spatio-Temporal Autoencoder ». In : *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), p. 2276-2280 (cf. p. 41).
- [101] P. Wang. *Vision Transformer - Pytorch Overview*. 2020. url : <https://github.com/lucidrains/vit-pytorch> (cf. p. 47, 48, 165).
- [102] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, M. Tomizuka, K. Keutzer et P. Vajda. « Visual Transformers : Token-based Image Representation and Processing for Computer Vision ». In : *ArXiv abs/2006.03677* (2020) (cf. p. 48, 165).

- [103] D. Wu, M. Liao, W.-T. Zhang, X. Wang, X. Bai, W.-Q. Cheng et W.-Y. Liu. « YOLOP : You Only Look Once for Panoptic Driving Perception ». In : *Machine Intelligence Research* 19 (2021), p. 550-562 (cf. p. 57, 165).
- [104] Q. Zhang, Y. N. Wu et S.-C. Zhu. « Interpretable Convolutional Neural Networks ». In : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), p. 8827-8836 (cf. p. 32).
- [105] S. Zhu, C. Chen et W. Sultani. « Video Anomaly Detection for Smart Surveillance ». In : *ArXiv* abs/2004.00222 (2020) (cf. p. 21, 24, 25, 49, 161).
- [106] Z. Zou, Z. Shi, Y. Guo et J. Ye. « Object Detection in 20 Years : A Survey ». In : *Proceedings of the IEEE* 111 (2019), p. 257-276 (cf. p. 62).

# Liste des tableaux

1.1	Jeux de données cités par S. Zhu, C. Chen et Sultani (2020) . . . . .	25
1.2	Répartition en catégories des vidéos de UCF Crime . . . . .	27
1.3	Durée des anomalies d'apprentissage de UCF Crime . . . . .	28
1.4	Durée des anomalies de test de UCF Crime . . . . .	28
2.1	Comparaison de différentes techniques de détection d'anomalies . .	63
2.2	Comparaison des diverses techniques de détection d'objets évoquées	64
3.1	Répartition des vidéos de notre jeu de données . . . . .	75
3.2	Durée des anomalies de notre jeu de données . . . . .	76
4.1	Variation de la taille de nos images . . . . .	88
4.2	Variation de la taille de nos séquences . . . . .	88
4.3	Techniques appliquées au modèle Bagarre . . . . .	91
4.4	Performance sur le jeu de données coup de feu avec DINO en pré-traitement . . . . .	92
4.5	Matrice de confusion sur le jeu de données coup de feu en avec DINO en prétraitement . . . . .	92
4.6	Performance de CGRU vs YOLO+CRU sur les classes coup de feu / normal . . . . .	94
4.7	Performances pour un masque avec fond noir . . . . .	95
4.8	Matrice de confusion pour un masque avec fond noir . . . . .	95
4.9	Performances pour un masque sans fond noir . . . . .	95
4.10	Matrice de confusion en pourcent pour un masque sans fond noir .	96
4.11	Évaluation par séquence (3 classes), sans fond . . . . .	97
4.12	Matrice de confusion pour une évaluation par séquence (3 classes), sans fond . . . . .	97

4.13 Évaluation par séquence (4 classes) . . . . .	98
4.14 Matrice de confusion en pourcent pour une évaluation par séquence (4 classes) . . . . .	99
4.15 Détection normal / anormal sans fond (avec bagarre et coup de feu) . . . . .	99
4.16 Matrice de confusion de la détection normal / anormal sans fond basé sur 2 anomalies : bagarre et coup de feu . . . . .	99
4.17 Détection normal / anormal avec fond (avec 3 anomalies : bagarre, coup de feu et incendie) . . . . .	100
4.18 Matrice de confusion de détection normal / anormal avec fond basé sur 3 anomalies : bagarre, coup de feu et incendie . . . . .	100
4.19 Performance liée à la détection d'incendies . . . . .	101
4.20 Performance liée à la détection de coup de feu (partie déjà présentée dans le tableau 4.6) . . . . .	101
4.21 Détection de la classe personne sur le jeu de données COCO + le nôtre . . . . .	102
4.22 Comparaison de performance pour chacun de nos modèles . . . . .	103
4.23 Performance du modèle CGRU + YOLO . . . . .	104
4.24 Performance générale : YOLO (multi-classes) + CGRU (multi-classes)	105
4.25 Performance générale : YOLO (multi-classes) + CGRU (nomal / anormal) . . . . .	105
4.26 Temps d'exécution de YOLO et CGRU en parallèle . . . . .	113
4.27 Temps d'exécution de YOLO et CGRU en série . . . . .	113
4.28 Résultats de nos expérimentations . . . . .	115
4.29 Performances des divers réseaux à convolution . . . . .	116
4.30 Récapitulatif . . . . .	117
4.31 Modification de l'architecture deVGG19 . . . . .	117
4.32 Comparaison des performances obtenues avec LSTM et avec GRU . . . . .	117
4.33 Recherche du nombre de couche entièrement connecté du classifieur . . . . .	118
<b>4.34 Performance de la détection des bagarres</b> . . . . .	119
<b>4.35 Performance de la détection des coups de feu</b> . . . . .	120
<b>4.36 Performance de la détection des incendies</b> . . . . .	121
<b>4.37 Performance générale du modèle multi-classes</b> . . . . .	122
<b>4.38 Performance sur les bagarres</b> . . . . .	123
<b>4.39 Performance sur les coups de feu</b> . . . . .	124
<b>4.40 Performance sur les incendies</b> . . . . .	125

4.41 Performance sur le jeu de données Normal / Anormal . . . . .	126
4.42 Performance sur le jeu de données Bagarre / Normal . . . . .	127
4.43 Performance sur le jeu de données Coup de feu / Normal . .	128
4.44 Performance sur le jeu de données Incendie / Normal . . . . .	129
4.45 Vitesse d'exécution de notre modèle . . . . .	131
4.46 YOLOV3: confiance 55%, threshold 70%, IoU min 0% max 100%, min box size 0px . . . . .	132
4.47 YOLOV4: confiance 55%, threshold 70%, IoU min 0% max 100%, min box size 0px . . . . .	133
4.48 Comparaison des performances de YOLO4 (1 classe) avec YOLO4 (2 classes) . . . . .	133
4.49 YOLO: détection de coups de feu, confiance à 55% . . . . .	135
4.50 YOLO: détection d'incendies, confiance à 55% . . . . .	135



# Table des figures

4.1	Extrait du jeu de données UCSD	26
1.1	Architectures récurrentes (Toharudin et al., 2021)	37
1.2	Architecture de VGG19 (Lagunas et Garces, 2018)	38
1.3	Fonctionnement des filtres de convolution, Ji et al. (2010) et Ghosh (2018)	40
1.4	Architecture d'un auto-encodeur, Sublime (2022)	41
1.5	Auto-encodeur d'images, Chollet et al. (2015)	42
1.6	Architecture du TCN (Lea et al., 2016)	43
1.7	Architecture de l'auto encodeur à base de TCN (Lea et al., 2016)	44
1.8	Architecture d'un transformer, Vaswani et al. (2017)	45
1.9	Vision transformer, P. Wang (2020) et Hassani et al. (2021)	47
1.10	Vision Transformer, B. Wu et al. (2020)	48
1.11	Carte d'attention générée par DINO, Caron et al. (2021)	48
1.12	GAN vanilla Architecture, Cai et al. (2020)	49
2.1	Boost Cascade (Viola et Jones, 2001)	50
2.2	Fonctionnement de RCNN, Girshick et al. (2013)	51
2.3	Fonctionnement de YOLO, Redmon, Divvala et al. (2015)	53
2.4	Non Maximal Suppression (NMS), Jain et Nandy (2019)	53
2.5	Architecture de DenseNet, Huang, Z. Liu et Weinberger (2016)	55
2.6	DenseNet vs CSPDenseNet, C.-Y. Wang, H.-Y. M. Liao et al. (2019)	55
2.7	Data augmentation de YOLOV4, Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao (2020)	56
2.8	Architecture de YOLOF, Q. Chen et al. (2021)	56
2.9	Architecture de YOLOP, D. Wu et al. (2021)	57
2.10	Architecture de YOLOR, C.-Y. Wang, Yeh et H. Liao (2021)	58

2.11	Architecture de YOLOS (Fang et al., 2021) . . . . .	59
2.12	Architectures proposées par YOLOV5 . . . . .	60
2.13	Performances de YOLOV7, C.-Y. Wang, Bochkovskiy et H.-Y. M. Liao (2022) . . . . .	61
2.14	Applications de YOLOV7, C.-Y. Wang, Bochkovskiy et H.-Y. M. Liao (2022) . . . . .	61
2.15	Évolution de la détection d'objet (Zou et al., 2019) . . . . .	62
2.1	Architecture globale . . . . .	71
4.1	Architecture du module d'analyse temporelle (CGRU) . . . . .	73
4.2	Répartition de nos données . . . . .	76
6.1	Convolution 2D / convolution 3D / Convolution incluse dans une couche time distributed . . . . .	78
7.1	Fonctionnement global . . . . .	82
2.1	Fonctionnement des générateurs testés (Ferlet, 2019) . . . . .	87
2.2	Data augmentation . . . . .	89
2.3	Pré-traitement appliqué à une image représentant un coup de feu . . . . .	90
2.4	Images représentant une bagarre . . . . .	91
2.5	Carte d'attention pour une vidéo de la classe coup de feu . . . . .	92
2.6	Surapprentissage pour la classe Bagarre avec DINO comme prétraitement . . . . .	93
3.1	Masque effectué avec YOLO . . . . .	94
3.2	Estimation de la pose par YOLOV7 pour une bagarre, sans fond . . . . .	97
3.3	Estimation de pose par YOLOV7 pour une bagarre, avec fond . . . . .	98
4.1	Résultat de YOLO V4 entraîné sur les classes "gun" + "person" . . . . .	102
5.1	Cartes d'activation pour une vidéo provenant de la classe coup de feu extraite du jeu de test . . . . .	106
5.2	Exemple de visualisation pour les classes : coup de feu et bagarre . . . . .	107
5.3	Carte d'activation et visualisation des contours sur une image de la classe coup de feu . . . . .	107
5.4	Carte d'activation et visualisation des contours sur une image de la classe bagarre . . . . .	108
5.5	Cartes d'activation montrant la variation du dropout en utilisant 64 neurones pour GRU . . . . .	108

5.6	Cartes d'activation pour une variation du nombre de neurones du GRU en utilisant un dropout de 50% . . . . .	109
5.7	Cartes d'activation pour une variation du nombre de couches du classifieur . . . . .	109
5.8	Cartes d'activation de notre modèle comprenant 1024 neurones pour le GRU et 0% de dropout . . . . .	110
5.9	Visualisation d'une vidéo normale provenant du jeu de test représentant une location. . . . .	111
5.10	Carte de saillance pour une vidéo normal . . . . .	111
5.11	4 filtres de convolution pour les couches 1 et 2 . . . . .	112
7.1	Détection de bagarre . . . . .	130
7.2	Détection d'incendie . . . . .	130
7.3	Détection de coup de feu . . . . .	131
8.1	Courbe d'apprentissage YOLOV4 : 2 classes . . . . .	134
8.2	Courbe d'apprentissage YOLOV4 : 1 classe . . . . .	134
8.3	Détection d'armes à feu via YOLOV4 . . . . .	136
8.4	Détection de flamme via YOLOV4 . . . . .	136
11.1	Détection d'accident . . . . .	143



# Table des matières

Remerciements . . . . .	2
Résumé . . . . .	3
Abstract . . . . .	4
<b>Introduction</b>	<b>7</b>
<b>Acronymes</b>	<b>10</b>
<b>I Problématique et état de l'art</b>	<b>13</b>
<b>1 Problématique</b>	<b>17</b>
1 Détection d'anomalie . . . . .	18
2 Objectif . . . . .	20
3 Analyse des vidéos . . . . .	21
4 Jeux de données . . . . .	24
5 Exigences . . . . .	31
6 Conclusion . . . . .	33
<b>2 État de l'art</b>	<b>35</b>
1 La détection d'anomalie dans les vidéos . . . . .	36
1.1 Convolutional RNN . . . . .	36
1.2 Convolution 3D . . . . .	39
1.3 Convolutional Auto encodeur . . . . .	40
1.4 Temporal Convolutional Network (TCN) . . . . .	42
1.5 Transformer . . . . .	44
1.6 Modèles génératifs . . . . .	48

2	Détection d'objets . . . . .	49
2.1	RCNN / Fast RCNN et Faster RCNN . . . . .	50
2.2	YOLO . . . . .	52
2.2.1	YOLOV1 . . . . .	52
2.2.2	YOLOV2 . . . . .	53
2.2.3	YOLOV3 . . . . .	54
2.2.4	YOLOV4 . . . . .	54
2.2.5	YOLOF, YOLOP, YOLOR, YOLOS, YOLOX . .	56
2.2.6	YOLOV5 . . . . .	59
2.2.7	YOLOV6 . . . . .	60
2.2.8	YOLOV7 . . . . .	60
3	Explicabilité . . . . .	62
4	Conclusion . . . . .	63
<b>II</b>	<b>Système réalisé</b>	<b>65</b>
<b>3</b>	<b>Système</b>	<b>69</b>
1	Introduction . . . . .	70
2	Architecture globale . . . . .	70
3	Composant d'analyse spatiale . . . . .	71
3.1	Images . . . . .	72
4	Analyse temporelle . . . . .	73
4.1	Extraction de caractéristiques spatiales . . . . .	74
4.2	Détection . . . . .	74
4.3	Vidéos . . . . .	75
5	Composant de correction . . . . .	76
6	Explicabilité . . . . .	77
7	Fonctionnement . . . . .	79
8	Conclusion . . . . .	83
<b>4</b>	<b>Expérimentations et résultats</b>	<b>85</b>
1	Introduction . . . . .	86
2	Prétraitement des données . . . . .	86
3	Expérimentations en mode série . . . . .	93

4	Fonctionnement en mode parallèle . . . . .	100
5	Mode traçage (tracabilité des résultats) . . . . .	106
6	Synthèse . . . . .	112
7	Analyse temporelle : la détection d'anomalie . . . . .	114
7.1	Choix du modèle . . . . .	114
7.2	Résultats . . . . .	118
8	Analyse spatiale : la détection d'objet . . . . .	131
8.1	Choix du modèle . . . . .	131
8.2	Résultats . . . . .	135
9	Conclusion . . . . .	136
	<b>Conclusion</b>	<b>141</b>
10	Synthèse générale . . . . .	141
11	Limitation de mon travail . . . . .	142
12	Perspectives . . . . .	144
	<b>Glossaire</b>	<b>146</b>
	<b>Mes publications</b>	<b>149</b>
	<b>Bibliographie</b>	<b>151</b>
	<b>Liste des tableaux</b>	<b>161</b>
	<b>Table des figures</b>	<b>165</b>
	<b>Table des matières</b>	<b>169</b>