

# Projet Chef d'Œuvre - Compteur de Passants Uniques par Ré-Identification

Le Pennec Guilian

December 2021

Ce rapport est un bilan du projet chef d'œuvre entrepris pour la formation Data IA par Simplon et Microsoft. Il couvrira le déroulé du projet de son imagination à sa conception, en passant par les problèmes rencontrés et les améliorations possibles.

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Veille Exploratoire de la Documentation d'Intérêt</b>	<b>2</b>
2.1	Le/Les Support/Supports Physique/Physiques . . . . .	3
2.2	La Détection des Passants dans l'Image . . . . .	3
2.3	La Méthode de Ré-Identification . . . . .	4
2.4	Les Méthodes Déjà Disponibles . . . . .	7
<b>3</b>	<b>Les Données</b>	<b>8</b>
3.1	Le Besoin . . . . .	8
3.2	La Récupération des Données . . . . .	8
3.3	L'Espace de Stockage des Données . . . . .	12
3.4	Analyse et Nettoyage des Données . . . . .	13
<b>4</b>	<b>Les Modèles</b>	<b>14</b>
4.1	Les Modèles de Suivi d'objets . . . . .	14
4.1.1	Récupération des Boîtes Limites . . . . .	15
4.1.2	Traqueur par plus Courte Distance des Centroïdes . . . . .	16
4.2	Le Modèle de Ré-Identification . . . . .	17
4.2.1	Architecture . . . . .	19
4.2.2	Entraînement . . . . .	21
4.2.3	Résultats . . . . .	23
<b>5</b>	<b>L'Application</b>	<b>25</b>
5.1	La Base de Données Relationnelle . . . . .	26
5.2	Monitorage et Tests Unitaires . . . . .	27
<b>6</b>	<b>Résultats du Comptage Assisté de Ré-Identification</b>	<b>28</b>
<b>7</b>	<b>Communication et Méthodologie de Gestion durant Le Déroulement du Projet</b>	<b>29</b>
<b>8</b>	<b>Pistes d'Améliorations</b>	<b>30</b>
<b>9</b>	<b>Conclusion</b>	<b>31</b>

# 1 Introduction

Travaillant au sein de Groupama Loire Bretagne, l'objectif est, en suivant une démarche de "Recherche et Développement", de programmer un algorithme capable de compter les passages de personnes dans chaque agence. Le matériel utilisé pour la tâche doit également être défini, en favorisant tout de même les caméras de types caméras de surveillance.

Ce comptage pourrait permettre une meilleure gestion de la disponibilité des employés en agence en optimisant leur temps de travail en fonction des données récupérées. Des décisions comme l'ouverture, la fermeture ou l agrandissement d'agences peuvent également être facilitées par la récupération de ces informations. Enfin, dépendant du RGPD, le projet pourrait être agrandi pour inclure la reconnaissance d'âge et de genre afin de permettre une meilleure compréhension de la clientèle afin de conduire les démarches décisionnelles.

Une entreprise externe à Groupama avait déjà tenté de répondre à ce besoin de comptage distinct des passages en agences, mais les résultats, qui ne sont pas disponibles, étaient bien trop mauvais et le projet fut abandonné. Cependant, de cet échec, des informations ont pu être tirées. Premièrement et le point le plus important, il semblerait que l'une des raisons causant la sur-évaluation du nombre de personnes distincte ayant traversé l'agence, soit les employées, qui pour plusieurs raisons telles que raccompagner un client à la sortie ou partir en pause, passent dans le champ de la caméra et sont

recomptés plusieurs fois, faisant grandir les chiffres. Secondement, en moyenne 32 personnes différentes passe en agence chaque jour.

Bien que les déplacements des employées dans l'agence aient été identifiés comme facteur important du manque de précision dans le comptage, il est tout de même important pour la robustesse de la solution d'être apte à détecter une personne, cliente, qui re-passera en agence pour un oubli ou une autre raison quelconque. De ce fait, il n'est pas à considérer de garder des images de chaque employé à des fins de récupération d'objet (eng : Object Retrieval) dans une base de données déjà existante. De plus, garder en mémoire de telles données brutes, effectivement des photographies des employés et/ou des clients, n'est pas compatible avec le RGPD. Dans le contexte de données vectorisées par un modèle d'encodage, les spécificités légales ne nous sont pas connues, il est donc également préférable d'éviter de les conserver. D'une entente commune avec les parties prenantes de ce projet, aucune donnée ne devra être stockée plus d'une journée, et il est préférable de ne pas concentrer la ré-identification sur des caractères faciaux, à cause en partie du caractère plus intrusif que cela implique, mais également dû à la complexité d'obtenir des images précise du visage.

L'objectif est donc : créer un modèle capable de compter distinctement le nombre de personnes étant passé dans une agence, sans conserver à long terme de données sur ces dites personnes.

## 2 Veille Exploratoire de la Documentation d'Intérêt

La séparation du problème en étapes est simple :

1. Les données du théâtre des déplacements, dans notre cas les agences Groupama, doivent être récupéré.
2. Dans ces données, les passants doivent être détectés.
3. L'information relative à ces passants doit être récupérée.
4. Ces informations doivent alors être utilisées pour décider de la nature du passant entre "déjà vue" ou "nouveau".

Une liste de questions évidentes se posent alors pour répondre à chaque objectif :

- Quel matériel utiliser pour récupérer les données qui seront traitées par l'algorithme ?
- Quelle méthode utiliser pour détecter les passants ?
- Quelle catégorie d'algorithme choisir pour ré-identifier les passants ?
- Qu'est-ce qui se fait déjà pour répondre à cette problématique ?
- Quelles connaissances techniques sont nécessaires pour comprendre et résoudre le problème ?

## 2.1 Le/Les Support/Supports Physique/Physiques

Bien que ces outils soient utilisés à des fins de comptabilisation des passages, les lasers et détecteurs de mouvements ne suffisent pas pour la tâche de ré-identification. Nous avons besoin d'être capable d'extraire des informations de l'entité que l'on souhaite identifier, pour cela peu de choix subsistent autre que l'utilisation de caméras. Cependant, reste la question du type de caméras à utiliser, la couverture du spectre lumineux n'étant pas la même en fonction de l'instrument.

Il existe principalement deux types de caméras documentées par la recherche en ré-identification : infrarouge (IR) et spectre visible (RGB), cette dernière possédant une variante RGB-D (pour "Depth") qui n'a pas été prise en compte pour diverses raisons qui ne seront pas mentionnées. Une différence à noter est celle entre caméra infrarouge et caméra thermique, bien que les deux captent les rayonnements infrarouges émis par les corps et les objets, leurs sensibilités sont différentes. Ainsi, les caméras infrarouges captent les ondes d'une longueur d'un micron, là où les caméras thermiques captent celles d'une longueur d'une dizaine de microns. De ce fait, les caméras IR sont capables de vision nocturne uniquement, tandis que les caméras thermiques ont la capacité de détecter avec précision, les sources de chaleur et leur variations [32]. Dans ce rapport, les deux seront labellisées sous la bannière "IR". L'abréviation "IRT" pour "infrarouge thermique" sera utilisée pour préciser la nature exacte de la caméra si besoin.

L'un des attraits majeur qui stimule les recherches sur la ré-identification par caméras IR, est leur invariance aux changements de luminosité. Cependant certaines publications profitent de la particularité des caméras IRT pour récupérer des indicateurs discriminant disponibles seulement grâce à l'observation des cartes de températures, par exemple l'empreinte thermique (thermogramme) d'une personne peut être utilisé comme tel ou pour l'inférence des vaisseaux sanguins du visage. L'influence des émotions ; du statut post-exercice ou post-repas ; ou encore des températures extérieures sur le thermogramme d'une personne, cumulé au peu de recherches sur le sujet, font que les résultats de ré-identification par spectre IRT n'atteignent pas encore les capacités de l'état de l'art en spectre visible [17]. De ce fait, une grande partie des recherches, généralement sur les simples caméras IR, sont tournées vers leur utilisation en concert avec

une ou plusieurs caméras RGB [35, 19, 23, 3, 24, 18], exploitant ainsi leur capacité à voir dans le noir pour assister une ré-identification dans le spectre visible déjà présente.

D'autres variables favorisant l'utilisation du spectre visible sont : le peu d'ensembles de données disponible en images IR ; l'âge de la recherche utilisant le spectre visible, qui dépasse de loin celle utilisant le spectre IR et qui, par extension, signifie une plus grande documentation avec une plus grande visibilité et accessibilité ; ainsi que l'environnement contrôlé et éclairé que sont les agences Groupama, loin des situations variables qui rendent les caméras IR utiles et intéressantes. Le choix du matériel de capture est donc la caméra RGB.

## 2.2 La Détection des Passants dans l'Image

Il est difficile d'aborder la détection d'objets sans aborder les détecteurs à une passe comme YOLO (You Only Look Once), SSD (Single Shot Detector) ou RetinaNet, descendants de la famille des réseaux de neurones à convolutions basés sur les régions (RCNN). Effectivement, ces techniques représentent sans contestations, l'état de l'art du domaine que ce soit en terme de vitesse ou de performances. Tous ces modèles ont pour but de récupérer les "boîtes limites" ou "BBox" (eng : Bounding Box) sur une image, et de prédire la classe de leur contenu. Ainsi, nous pouvons isoler, par exemple, les humains présents dans la scène pour se concentrer sur uniquement la partie exacte de l'image en contenant.

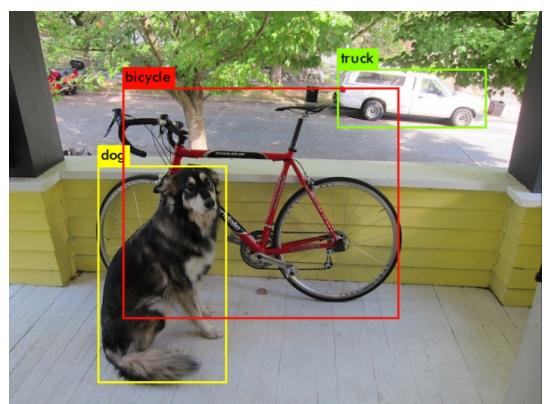


FIGURE 1 – Ici, le détecteur d'objets a remarqué la présence d'un chien, d'un vélo et d'un fourgon. Il est donc possible d'effectuer d'autres calculs sur, par exemple, le chien, afin de ne passer que sa partie de l'image dans un modèle qui prédirait sa race.

YOLO est probablement le modèle le plus connu, contrairement au RCNN qui font traiter toute l'image par un réseau de neurones à convolutions, puis utilisent un réseau de propositions de régions pour générer des possibles BBox, avant de passer ces propositions dans un classifieur, YOLO lui divise une image en une grille de  $S \times S$ , et créer pour chaque cellule  $B$  "boîtes d'ancrage", le modèle s'occupera ensuite de prédire par régression la classe contenue dans la boîte d'ancrage, ainsi que les modifications à apporter afin d'obtenir la boîte limite.

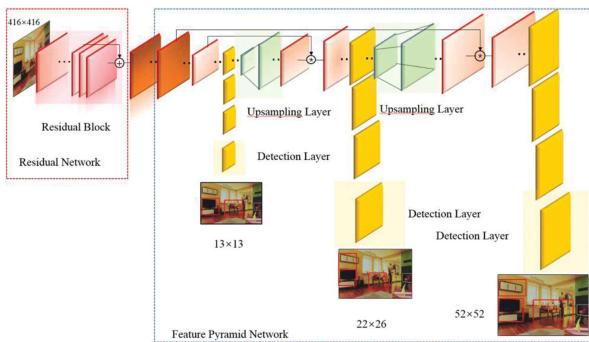


FIGURE 2 – L'architecture de YOLOV3. Les différentes tailles d'objets sont détectées par différents niveaux de mise à l'échelle.

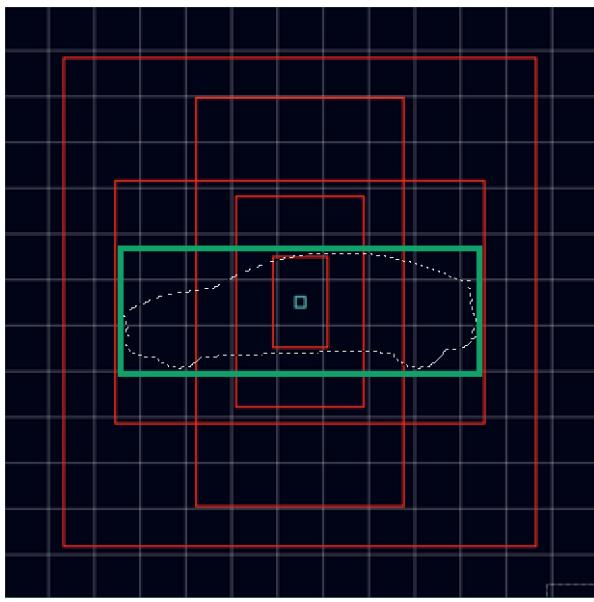


FIGURE 3 – Boîtes d'ancrages de multiple tailles générée par cellule. Dans YOLOV3, trois boîtes sont générée pour chacune des trois échelles, ce qui fait neuf boîtes par cellule au total.

Les modèles de détection d'objets sont entraînés sur de gigantesques ensembles de données comme COCO pour YOLO, cet entraînement peut durer

de nombreuses heures malgré l'utilisation de superordinateurs ; la génération de boîtes limites n'étant pas le point central du projet, nous nous contenterons d'importer l'architecture et les poids. La bibliothèque Gluon fut d'abord utilisée, mais cette dernière utilise des tenseurs MXNet pour ses calculs, et leur conversion en matrice numpy est très longue, prenant jusqu'à une seconde pour un vecteur de quatre valeurs, la raison semble être que la fonction de conversion attend l'arrêt de tout autre calcul en cours avant de se lancer. À la place, c'est la bibliothèque onnxruntime, qui permet la lecture et l'utilisation du format de modèle open source "Open Neural Network Exchange" (onnx), qui est utilisée. Cela me permet notamment de profiter du zoo de modèles disponible sur [le github officiel de Open Neural Network Exchange](#). Le choix de YOLOV3 se base en grande partie sur sa vitesse supposée, même si le matériel ne nous permet pas d'obtenir les 37.3 images par secondes ; des tests sur Tiny-YOLOV3, plus rapide que YOLOV3, ont montré une trop grosse perte en performance, justifiant de ne pas utiliser cette version réduite du modèle.

## 2.3 La Méthode de Ré-Identification

La documentation sur la ré-identification est abondante, mais une précision s'impose sur la nature de ces recherches en contraste avec les besoins du projet. Notre but est d'être capable de détecter une personne comme n'ayant jamais été vu auparavant, pour ensuite garder temporairement des informations descriptives relatives à elle, et être capable lors de la venue d'une nouvelle personne de comparer les nouvelles informations à celles déjà enregistrées afin de décider si oui ou non, la nouvelle personne est belle et bien nouvelle. Hors une majeure partie de la documentation, si ce n'est la totalité, applique la ré-identification dans un contexte de base de données pré-existante et de nature permanente, c'est ce qui est entendu par "réécriture d'objets", signifiant par là qu'une identité n'a pas comme vocation à être désigné comme nouvelle. Cette spécificité s'applique surtout du côté applicatif et de l'objectif principal : "trouver un moyen de vectoriser l'image d'une identité afin de la regrouper avec d'autres vecteurs représentant la même identité", reste le même.

Avant même de se tourner vers l'apprentissage machine, de populaires méthodes statistiques existent qui, malgré leurs simplicités, possèdent d'excellente capacité à extraire les caractéristiques d'une image. Les méthodes HOG (Histogramme de gradient orienté) [4], LBP ("Motif binaire local")

de respectivement 2005 et 1993, ou encore ORB (Orienté RAPIDE et tourné BREF) et SIFT (transformation de caractéristiques visuelles invariante à l'échelle) dont nous ne parlerons pas, se basent sur une décomposition des caractéristiques d'une image par comparaison des valeurs voisines à chaque pixel.

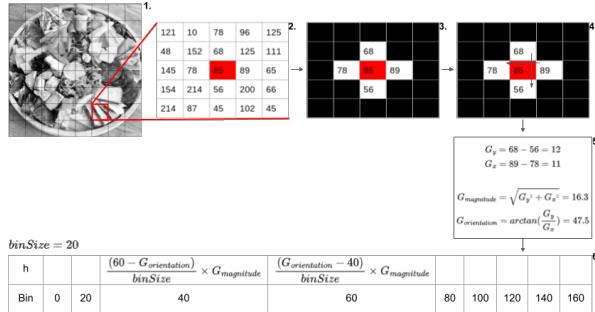


FIGURE 4 – 1. Une image est divisée en plusieurs sous-sections, typiquement de  $8 \times 8$ ,  $16 \times 16$  ou  $32 \times 32$  valeurs, un histogramme est généré comme indiqué dans la figure 5 pour chaque sous-section ;

2. On itère sur la liste de sous-sections ; 3. La matrice des pixels dans cette sous-section est parcourue, on ne prend en compte que ceux qui possèdent 4 voisins, diagonales non-comprises ; 4. On récupère le gradient des axes 2D en prenant, pour X, les valeurs au-dessus moins celles en dessous du pixel actuellement ciblé, et pour y, celles de droite moins celles de gauche ; 5. La magnitude du gradient total ainsi que sa direction sont calculé selon les instructions de la figure 6 ; 6. À partir de ces valeurs, on remplit l'histogramme de la sous-section actuelle, en partageant la magnitude entre les barres les plus proches.

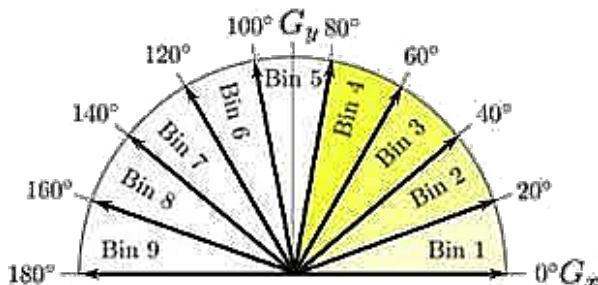


FIGURE 5 – HOG génère traditionnellement des histogrammes composé de 9 barres, dépendant de l'orientation en degrées de 0 à 180 et de 20 en 20, il est bien entendu possible de changer la taille de l'intervalle.

Dans le cas de HOG, une fois les histogrammes de chaque sous-section remplies, les valeurs sont normalisées. Pour ce faire, les sous-sections sont regroupées par 4, donc par carrées de  $16 \times 16$  valeurs

si  $8 \times 8$  était le choix initial, leur matrice d'histogramme respective de  $9 \times 1$  en devient donc une matrice de  $36 \times 1$ . Chaque valeur est ensuite divisée par la racine carrée de la somme de chaque valeur au carré, voir équation 1. Le vecteur résultant de la concaténation de tous les histogrammes de taille  $36 \times 1$ , représente l'encodage final de l'image par HOG. Dans le cas d'une image multi-canal, il est tout à fait possible de simplement créer un HOG pour chacun des canaux.

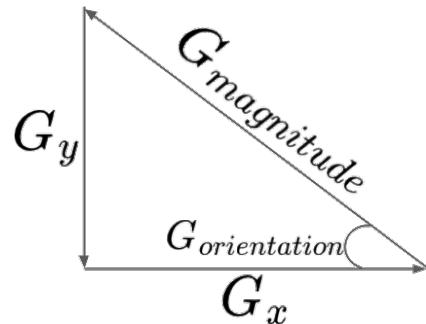


FIGURE 6 –  $G_x$  et  $G_y$  représentent les gradients dans chaque direction, récupérer la magnitude cumulée des gradients n'est alors qu'une question d'appliquer le théorème de Pythagore, l'orientation est le degré de l'angle dont la tangente est récupérable grâce à  $\tan(G_{\text{orientation}}) = \frac{G_y}{G_x}$ , donc l'angle en lui-même peut être calculé par  $G_{\text{orientation}} = \frac{G_y}{G_x}$

$$n = 36$$

$$V = [h_1, h_2, \dots, h_n]$$

$$c = \sqrt{\sum_{i=1}^n [v_i]} \quad (1)$$

$$\text{Norm}V = \left[ \frac{h_1}{c}, \frac{h_2}{c}, \dots, \frac{h_n}{c} \right]$$

La méthode LBP fonctionne de façon plus ou moins similaire, à la différence le "voisinage" pris en considération inclue les diagonales, et que le gradient n'est pas ce qui définit les valeurs de l'histogramme. À la place, la différence est prise entre le pixel central et ceux l'avoisinant, si le voisin est plus petit, on attribue un 0 à l'endroit correspondant dans une grille, si il est égal ou plus grand, un 1. Un noyau de poids est alors appliqué à la grille afin de convertir l'encodage binaire en décimal, par multiplication puis sommation (figure 7), la valeur résultante de cette opération est alors l'emplacement dans l'histogramme de taille 256 de la sous-section, où un 1 est ajouté. Comme pour HOG, chaque histogramme est normalisé avant d'être concaténé aux autres pour obtenir l'encodage final.

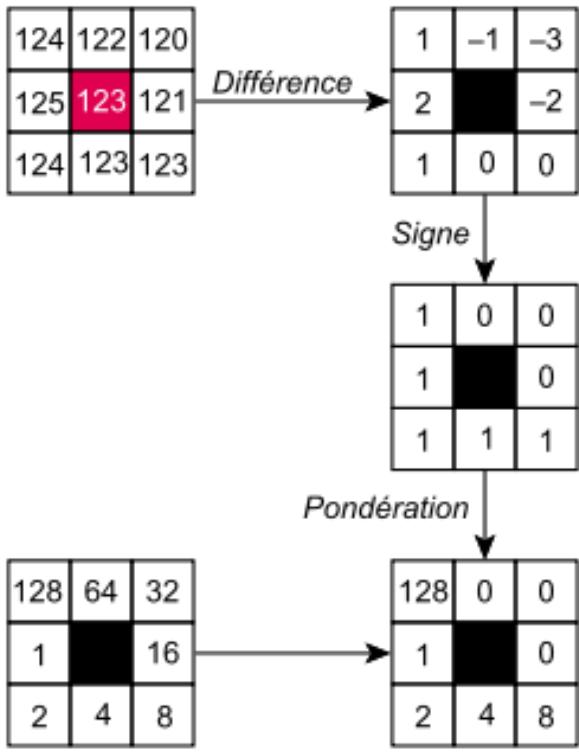


FIGURE 7 – Découpage étape par étape de la récupération du LBP d'une matrice de taille  $3 \times 3$ .

Ces méthodes d'extraction de caractéristiques ont déjà fait leur preuves [11, 36, 4] et peuvent être utilisées en tandem [6], HOG est très apte à encoder les bords et les coins, et peut être utilisé avec le filtre de Canny pour de meilleurs résultats [1]; tandis que LBP se spécialise dans la reconnaissance de texture et a déjà été utilisé pour des tâches de ré-identification avec succès.

La transformation de Fourier permet de décomposer un signal en de multiples fonctions sinusoïdales, cette technique peut également être utilisée pour des sons, mais également pour des images. Il est donc possible d'encoder une image par décomposition que ce soit par transformation de Fourier rapide ou, de préférence [30], transformation en ondelettes. Cette piste n'a pas été explorée en ré-identification et ne le sera donc pas dans ce rapport, mais il s'agit tout de même d'une idée qui, nous pensons, mérite son paragraphe.

Il ne fait aucun doute qu'utiliser des méthodes statistiques comme extracteurs de caractéristiques est viable, cependant la limite de temps disponible ne permet pas vraiment leur exploration, et leur faible documentation les voyant appliquées à de la

ré-identification ne garantit pas de résultats suffisants. Nous avons tout de même préféré les aborder dans un but d'exhaustivité.

Avant de parler des modèles, un point sur les discussions entre les parties prenantes quant à quelles caractéristiques utiliser pour la ré-identification. L'objectif reste comme discuté dans l'introduction, de respecter la vie privée des identités en se concentrant le moins possible sur le visage. Cette décision nous ferme les portes de beaucoup de méthodes pour la simple raison que la ré-identification par traits faciaux a gouverné le domaine pendant une grande partie de son existence, cependant cette tendance est sensiblement délaissée et une documentation de plus en plus importante se concentre sur des méthodes plus diverses, présentant tout de même d'excellentes performances. De ce fait, nous avons pu prendre en considération les stratégies suivantes : la reconnaissance de marche (Gait Recognition) [13], état de l'art moderne, est une méthode consistante à capturer la démarche d'une identité par inférence de son squelette, de multiple caméras ou une caméra RGB-D sont généralement utilisées, ce qui, rajouté à la complexité de la tâche, explique le choix d'une autre méthode ; au fil de nos discussions, une idée revenait régulièrement, utiliser les vêtements pour décrire des identités, bien entendu cette méthode rencontrerait le problème du changement d'habit si l'objectif était une ré-identification à long terme, mais il est peut probable que ce défaut nécessite d'être pris en compte lorsque la fenêtre des données n'est que d'une journée. Cette idée a bien entendu déjà été explorée allant même jusqu'à atteindre l'état de l'art en 2018, elle est souvent liée avec la description d'une image en langage naturel.

Li et al. [20] propose un modèle capable de reconnaître de lui-même les vêtements présents, entraîné en concert avec un modèle de traitement automatique des langues (NLP) et acceptant une image au format de boîte limite comme entrée. Niu et al. [15] améliore le modèle de Li et al. en divisant la boîte limite "globale" en 6 parties "locales", censées s'occuper de vêtements différents, et en effectuant des comparaisons locales-locales, locales-globales et globales-globales, tout en gardant l'aspect NLP. Kalayeh et al. [25] entraînent un modèle de segmentation sémantique pour récupérer les zones contenant chaque vêtement plus précisément que par de sous-boîtes limites. Bien que les papiers soient excellents, ces méthodes semblaient encore trop complexes pour le temps disponible et mon niveau de compétence actuel.

Une nouvelle et dernière idée était d'utiliser un auto-encodeur variationnelle (VAE). Ayant récemment appris leur fonctionnement, encoder une classe dans un espace latent sous forme de distribution semble coller parfaitement à la tâche de ré-identification, bien évidemment l'idée n'était pas nouvelle et quelques papier existent déjà utilisant des modèles génératifs [38]. La découverte de Beta-VAE [9] ne fut que plus encourageante pour justifier l'utilisation d'un auto-encodeur variationnelle, Beta-VAE encourage le désenchevêtrement des caractéristiques, signifiant que chaque scalaire dans le vecteur latent représente un attribut tangible de la structure de l'image encodé, l'orientation, la luminosité, la position, etc... Pour ce faire un paramètre " $\beta$ " est rajouté au terme de régulation dans la fonction de coût.

Pressée par le temps, la décision fut prise de reproduire l'auto-encodeur à variations normalisées (VNAE) qui compose une partie du modèle présenté par Ren et al. [27], un mois avant le début du projet. Ce modèle ne sera cependant pas celui présenté dans le chapitre 4.2, en effet malgré un dépassement du temps alloué au développement du VNAE, les efforts pour le faire marcher et obtenir les résultats du papier furent vains. Bien que le rythme de publication des auteurs puisse soulever des questions, et que l'absence de citations après un an n'aide pas, il n'est pas impossible que retenter l'implémentations, avec les nouvelles connaissances découlant de la réalisation de ce projet, ne finisse en succès. Recyclant autant que possible le travail déjà réalisé, le nouveau modèle, se base sur l'utilisation d'un extracteur de caractéristiques d'apprentissage profond pré-entraîné sur ImageNet, comme beaucoup dans la documentation vu jusque-là. Généralement le choix ce fait entre ResNet ([7, 8]) et InceptionV3 ([31]), le choix final de ResNet est entièrement basée sur son utilisation initial dans le VNAE, le reste de l'architecture suit les travaux de Luo et al. [21], qui utilise un mélange de fonctions de coût par triplet et par classification. De l'augmentation des données en ligne et utilisée avec notamment de la suppression aléatoire [39].

Avec ou sans utilisation des méthodes statistiques ou d'apprentissage profond pour réduire la dimensionnalité des images en servant d'extracteur de caractéristiques, il faut tout de même être capable de classifier les vecteurs représentant les passants comme appartenant à telle ou telle identité. Pour ce faire, les similarités cosinus ou euclidienne sont populaire, les classificateurs d'apprentissage profond avec l'entropie croisée comme fonction de coût

ayant tendance à séparer les encodages de façon cosine, avec fonction de coût par triplet, de façon euclidienne. Il est également possible d'appliquer des méthodes d'apprentissage machine, souvent, les algorithmes de k-moyennes, k plus proches voisins ou de machine à vecteurs de support sont utilisées. Dans notre cas, la similarité euclidienne a été empiriquement prouvé comme meilleur comparé à la similarité cosine, les autres méthodes n'ont pas été testé.

## 2.4 Les Méthodes Déjà Disponibles

Que ce soit pour s'en inspirer ou comme alternative financièrement plus intéressante qu'un développement interne, des recherches sur les alternatives possibles ont bien entendu déjà été réalisées. En recherchant des "applications/modèles de compteur de personnes avec ré-identification", les résultats alternes entre plateformes-en-tant-que-services (PaaS), plateformes de ventes de matériel physique (e.g., caméras), et modèles d'apprentissage machine.

La PaaS viso.ai, supporté par notamment samsung, intel et nvidia, propose des modèles pré-entraînés, ainsi qu'un outil de création d'applications par blocs et des outils de monitorage, les processus tournent sur le cloud ce qui permet la mise à l'échelle de l'application selon les besoins. L'avantage est la disponibilité d'une application de comptage, il suffirait donc de modifier son architecture pour insérer un modèle de ré-identification développé localement ce qui nous enlèverais la charge du développement d'un traqueur. Les désavantages viennent du besoin de développer le modèle de ré-identification, et du prix de minimum 795€ par mois..

Hikvision est un fournisseur de matériel informatique, avec une emphase sur la couverture vidéo, ils proposent des caméras à intelligences artificielles embarquées, capable de détection d'objets, comptage de foule et ré-identification de véhicules, liste non-exhaustive. L'avantage vient de l'aspect matériel, non seulement l'application est fournis, mais le système physique aussi. Le double tranchant de la nature embarqué de l'application est que sa modification ne fait pas partie de la liste des possibilités de bases. Nonobstant l'absence de ré-identification d'humain, le manque de contrôle sur les fonctionnalités rend la mise à l'échelle en fonction des besoins plus compliquée.

La majorité des recherches avait comme objectif la découverte d'une application prête à l'emploi. Surprenamment, le besoin exact auquel nous souhaitons répondre, n'a pu être qu'approximé par la problématique résolue par l'algorithme DeepSORT [figure 8](#). Ce modèle, amélioration de "SORT" par l'utilisation d'apprentissage profond, est un traqueur utilisant le filtre de Kalman (discuté partie 7) comme base en addition d'une ré-identification par-image dans l'objectif de confirmer l'association continue durant le suivi vidéo, et minimiser les chances d'échanges d'identités. Pour ce faire un classifieur, proposé pour DeepSORT par Nicolai Wojke et Alex Bewley [\[33\]](#), est entraîné avec comme objectif une meilleure séparation cosine entre les classes, dans cette optique une simple reformulation de la fonction de coût est développé. Bien que de la ré-identification entre en jeu, son utilisation n'est qu'à des fins d'assignation d'une image  $n$  à une image  $n + 1$ . Le problème est double, premièrement les ressources de calcul requises pour non seulement opérer de la détection d'objet, mais également de la ré-identification à chaque image du flux vidéo sont conséquentes ; deuxièmement bien

qu'il ne puisse suffire que de modifier le code afin de changer la fréquence des ré-identification, conserver les données et les utiliser pour un comptage distinct, cette tache devient rapidement compliquée sur du code sans commentaires, simplement modifier ce code déjà existant n'est également pas suffisant pour cocher les cases du référentiel. Globalement, DeepSORT donne probablement toutes les clés pour répondre à notre problème, eussent les circonstances été différentes, sa modification et son implantations dans un cadre applicatif auraient sans doute été au sommet de la liste des choix viables.

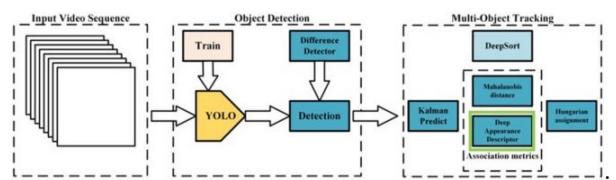


FIGURE 8 – Schéma représentant l'architecture de l'algorithme DeepSORT (Source : <https://medium.com/@riteshkanjee/deepsort-deep-learning-applied-to-object-tracking-924f59f99104>)

### 3 Les Données

#### 3.1 Le Besoin

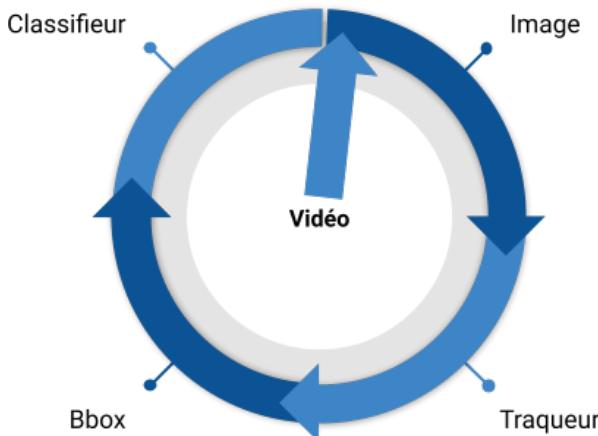


FIGURE 9 – Schéma du type de données reçus en entrée à chaque partie de l'application.

Le besoin en données ce présente suivant le schéma présenté dans la [figure 9](#) : une vidéo est découpée image par image, depuis ces images le Traqueur retourne les boîtes limites qui serviront d'entrée au Classifieur pour opérer la ré-identification.

Il faut donc :

- Des données sous format vidéo pour tester la

complétude de l'application, prouver qu'elle est opérationnelle et couvre toutes les fonctionnalités attendues.

- Des ensembles de boîtes limites organisées par identités pour entraîner le Classifieur.

#### 3.2 La Récupération des Données

Groupama ne possède pas de données exploitable pour répondre à notre besoin, l'objectif étant de présenter une preuve de concept avant d'investir dans l'installation de caméras. De ce fait, chaque ensemble de données doit être récupéré depuis internet. Bien que non-exhaustive, le site [CVonline](#) propose une liste d'ensembles de données répondant aux besoins de multiples sous-domaines de la vision par ordinateur, bien que son utilité dans l'obtention du contenu nécessaire fut limité, cette ressource reste intéressante à cataloguer.

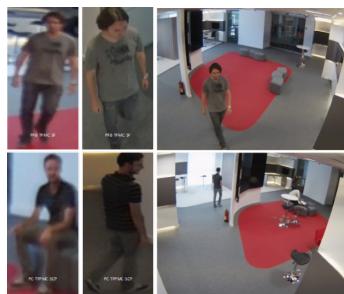
La première partie étant le Traqueur, les ensembles de données vidéos font l'objet des recherches initiales, l'idée est d'approximer du mieux que possible les conditions en agences, en angle de caméra comme en débit de passants. Pour cela les caméras extérieures ou dans des centres commerciaux bondés sont à éviter. Nous avons tout d'abord

exploré des répertoires de flux vidéos en ligne : EarthCam, quelques plans disposant de passants réguliers, mais une majorité de vues panoramiques de décors urbains ou naturels ; Insecam propose divers plans d'intérieurs de magasins avec des nombres variés de personnes, cependant, il s'agit souvent de petites échoppes où les vendeurs restent constamment à l'écran et où les passants sont rares, l'idéal serait d'avoir des allers-retours répétées de quelques individus, cumulé à plusieurs passages uniques d'identités diverses ; Shodan balaye l'internet à la recherche

de ports accessibles, quelques caméras sont donc disponible, mais rien d'intérêt de fut trouvé. Il était très vite évident qu'un flux vidéo en direct n'était pas le plus pratique, les bonnes circonstances sont difficile à trouver et la faible concentration des événements rendrait les tests beaucoup trop longs. La recherche d'ensembles de données spécifiquement pour le suivi vidéo se conclut par la récupération des deux seul candidats semblablement adéquat : ChokePoint Dataset ([figure 10a](#)) et PIROPO Database ([figure 10b](#)).



(a) ChokePoint Dataset, probablement parmi le plus simple des ensembles de données pour suivi vidéo. Il s'agit simplement d'une collection de scènes où diverses personnes traversent le champ. Sa simplicité est pratique pour le débogage du Traqueur, mais n'est pas indicatif des conditions réels. Certain individus traversent le champ dans plusieurs scènes, mais dans tous les cas rencontrées, les habits étaient différents. De ce fait trois choix subsistes : compter une seule identité sans se soucier des habits, considérer le changement d'habit comme un changement d'identité ou ne conserver que les images d'une identité habillée d'une certaine façon. La troisième option fut choisie, ne gardant que le passage, et donc les vêtements, avec le plus d'images disponibles.



(b) PIROPO Database, une bibliothèque de vidéos comprenant différents points de vue sur des scènes où une dizaine d'identités se meuvent, s'assoient et déplaces des objets. La grande salle et le plan large permettent de capturer une grande variété de situations, que ce soit en terme de poses et de distances, dévoilant les limites de YOLOV3, ou des croisements et autres patterns de circulations qui demandent à pousser les performances du Traqueur. PIROPO propose quelques images venant d'objectifs fish-eye, les tests montrent que YOLOV3 n'arrive pas à détecter les formes humaines dues aux déformations des bords de l'image. Cet ensemble de données permet une bonne balance de complexité grâce à la grande quantité de points de vue proposées, seul point négatif est la durée de chaque enregistrement qui a tendance à s'allonger par des périodes d'inactivités, ce qui rend les tests plus longs.



(c) Afin d'obtenir les données vidéos dont la complexité, la longueur, la densité des passages, etc... correspondes aux besoins, un ensemble de données fut développé. GBRALTAR (pour Groupama BRetAgn Loire TrAcking et Reidentification) contient 7 identités ; 3 plans de caméra : sur le côté, plongeant, et un vertical de bas en haut, qui ne sera pas utilisée dû au manque de recul de la caméra ; 5 minutes de vidéos ; entre 26 et 28 passages dans lesquels les identités traversent entre 48 et 52 fois, dépendant du plan de caméra. La grille descriptive de chaque passage sera disponible comme documentation supplémentaire optionnelle. La conception locale de ces vidéos nous permet de couvrir toutes les situations souhaitées, bien que la difficulté de mise en place des téléphones servant de caméras nous retourne des plans sans grand recule, complexifiant la tâche de YOLOV3 ainsi que du Classifieur. Cela veut cependant dire que de bons résultats sur ces données promettent, en théorie, de bon résultats en général.

FIGURE 10 – Échantillon récupéré des ensembles de données vidéos utilisés pour l'entraînement ou les tests de l'application.



(a) Market1501, probablement l'ensemble de boîtes limites le plus populaire. Bien que quelques identités furent gardées, les qualités d'images et les variations à l'intérieur de l'ensemble ne sont pas suffisantes pour un entraînement uniquement à partir de ce dernier.

(b) PRID2011, bien que vu plusieurs fois dans la documentation, PRID2011 possède des variations intra-identités, l'exemple ci-dessus étant le résultat d'une sélection manuelle des deux images les plus opposées pour cette identité.

(c) CUHK Person Re-identification Dataset, les images, en soit, ne sont pas de mauvaise qualité et dans l'absolu cet ensemble de données pourrait être utilisée, cependant, les images étant prise dans une université, les différences, notamment vestimentaires, entre les identités est réduite, le fond est également plus bruyant ce qui peut rendre l'entraînement plus imprévisible.

(d) ETHZ People Re-identification Dataset, possède, comment on peut l'observer à droite, quelques images de très bonne qualités, malheureusement l'absence de formatage et semblablement de nettoyage fait qu'une grande partie de l'ensemble est composée d'images inutilisables à l'instant de l'image de gauche.

FIGURE 11 – Échantillons pris d'une partie des ensembles de données pris en considération pour l'entraînement du Classifieur.

Dataset	Release time	# identities	# cameras	# images	Label method	Crop size	Mult shot
ViPeR	2007	632	2	1264	Hand	128X48	
ETH1,2,3	2007	85, 25, 28	1	8580	Hand	Vary	✓
QMUL-iLIDS	2009	119	2	476	Hand	Vary	✓
GRID	2009	1025	8	1275	Hand	Vary	
CAVIAR4ReID	2011	72	2	1220	Hand	Vary	✓
3DPeS	2011	192	8	1011	Hand	Vary	✓
PRID2011	2011	934	2	24541	Hand	128X64	✓
V4T	2011	47	2	752	Hand	Vary	✓
WARD	2012	70	3	4786	Hand	128X48	✓
SAINT-Softbio	2012	152	6	64472	Hand	Vary	✓
CUHK01	2012	971	2	3884	Hand	160X60	✓
CUHK02	2013	1816	10(5 pairs)	7264	Hand	160X60	✓
CUHK03	2014	1467	10(5 pairs)	13164	Hand/DPM	Vary	✓
RAID	2014	43	4	6920	Hand	128X64	✓
iLIDS-VID	2014	300	2	42495	Hand	Vary	✓
MPR Drone	2014	84	1		Pyramid Features(ACF)	Vary	✓
HDA Person Dataset	2014	53	13	2976	Hand/Pyramid Features(ACF)	Vary	✓
Shinpuhan Dataset	2014	24	16		Hand	128X48	✓
CASIA Gait Database B	2015(*see below)	124	11		Background subtraction	Vary	✓
Market1501	2015	1501	6	32217	Hand/DPM	128X64	✓
PKU-Reid	2016	114	2	1824	Hand	128X64	

FIGURE 12 – Liste non-exhaustive d'ensembles de données de ré-identification proposé par NEU-Gou sur GitHub

La deuxième partie est la récupération des "Bboxes" par le classificateurs afin de les encoder dans un format permettant la ré-identification par similarité euclidienne. De nombreux ensembles de données existent pour entraîner ce genre de modèles, la figure 12 proposent certains des plus populaires. Une difficulté imprévue est l'impossibilité d'accéder à certains ensembles, les raisons incluent les sites de téléchargements qui n'existent plus ou leur disponibilité réservée aux écoles ou aux instituts de recherches. Différents tests et analyses exploratoires furent effectués afin de réduire le nombre de candidats, quatre d'entre eux sont présentés dans la figure 11.

Après plusieurs tests non-concluant, s'entraînant uniquement sur Market1501, et après la découverte d'un papier brièvement mentionnant avoir entraîné avec succès un classifieur capable de distinguer plusieurs ensembles de données de ré-identification les uns des autres, que la conclusion fut tirée qu'il était nécessaire pour la robustesse et l'aptitude à généraliser du modèle que de l'en-

traîner sur un ensemble de données provenant de l'agglomération de plusieurs autres ensembles.

La contemplation de la qualité des images, de leur nombre par identité et de leurs variations dans la pose, la luminosité et/ou l'angle de la caméra ayant capturé la scène, nous pousse à trouver une autre solution que d'utiliser des ensembles de boîtes limites pré-découpées. Nous décidâmes d'appliquer le Traqueur avec YOLOV3 sur les données vidéos présenté dans la figure 10, afin d'en extraire les boîtes limites de façon légèrement pré-ordonnée. Cette stratégie nous permet de construire un ensemble de données avec une forte variation interne, il est composé de 27 identités provenant de ChokePoint Dataset [34], 19 de PIROPO Database [5], et 36 de Market1501 [37]. Cet ensemble sera, à l'avenir, référencé par le nom "ReIDMultiSet" et la distribution des images pour chaque identité le composant est illustré dans la figure 13. GBRALTAR est gardé pour servir de test, autant pour les scores de performances de la ré-identification que pour tester la globalité des fonctionnalités de l'application déployée ; PRID2011 [10] étant considéré comme simples, est utilisé comme comparaison des scores de performances de la ré-identification avec GBRALTAR.

Quelques clarifications quant à la figure 13 : chaque "identité" est une classe, chaque classe contient plusieurs images de cette identité. Les tests nous ont montré empiriquement qu'avoir le même nombre d'images dans chaque classe n'améliore pas les performances du modèle, les identités de PIROPO ont de loin plus d'images, mais cela ne fait que refléter l'extensivité de l'ensemble de données dont elles sont issues, la diversité intra-identité est comparable voire supérieur au reste de ReIDMultiSet et donc, l'entraînement n'en souffre pas. I.e. le modèle n'apprend pas à classifier des images quasiment identique à cause d'un grand nombre d'images de la même classe, il apprend à classifier correctement des images avec des caractéristiques différentes. À l'inverse, ChokePoint possède peu d'images par classe, cependant la qualité d'image et le caractère hétéroclite des identités contrebalance ce fait. Dans les vidéos de ChokePoint Dataset, les chemins empruntés à chaque passage sont relativement similaires, cette similarité pousse le modèle à utiliser d'autres caractéristiques que la position durant l'entraînement. ChokePoint est donc un bon ajout à PIROPO qui lui, ne possède pas forcément d'images suffisamment hautes qualité, et où les vêtements sont facilement confondable, mais qui possède en revanche une grande variété de poses. Les

classes issues de Market1501 ont été sélectionnées sur un critère de quantité, un tiers des classes étant dans le dernier quartile du nombre d'images par identité (table 1) ont été gardées.

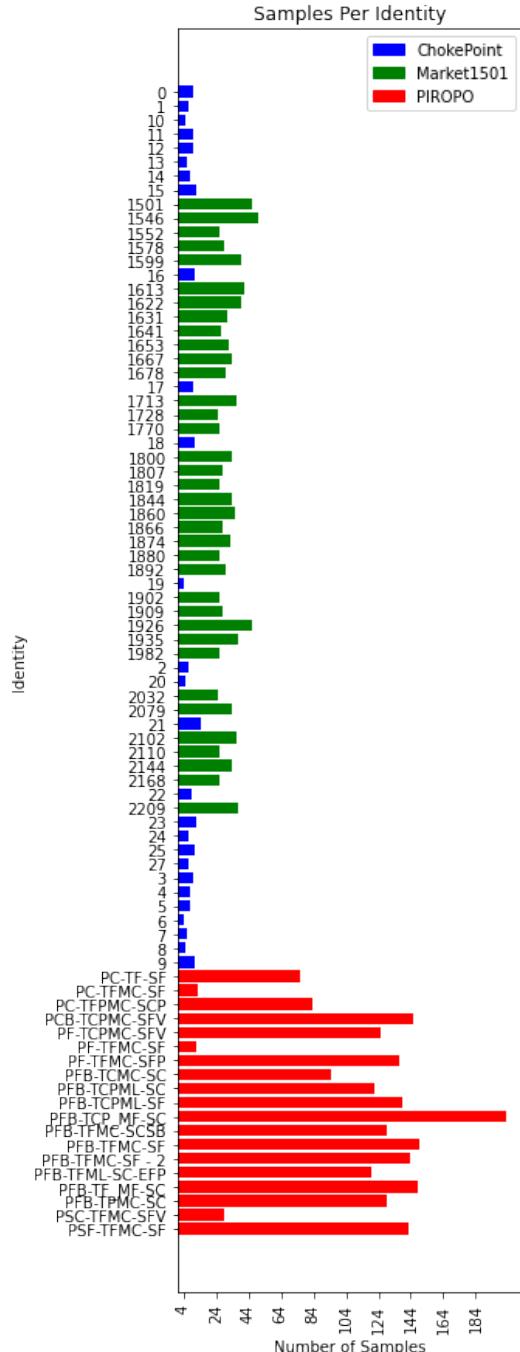


FIGURE 13 – Graphique en barres du nombre d'images par classes dans ReIDMultiSet, ainsi que l'appartenance initial de chacune de ces classes (de quel ensemble de données proviennent-elles).

Moyenne	17
Écart type	11
Min	2
25%	9
50%	15
75%	22
Max	72

TABLE 1 – Table descriptive des images par identités de Market1501. Les valeurs sont arrondies à l'entier le plus proche. 751 identités différentes sont présentes dans le Train de Market1501.

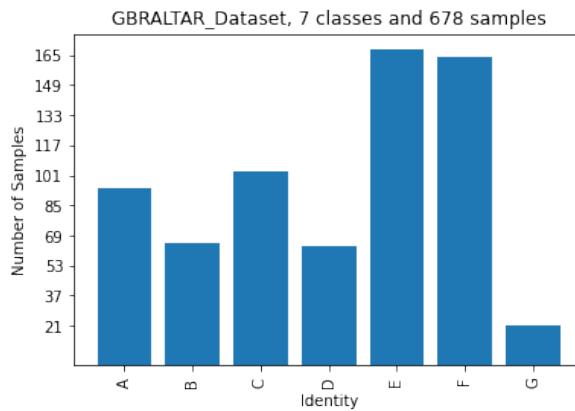


FIGURE 14 – Histogramme de la répartition des images pour chaque identité de GBRLTAR. L'identité labellisée "G" ne passe qu'une fois devant l'écran, ce qui explique le peu d'images. Les images contenant deux identités ont été supprimées.

### 3.3 L'Espace de Stockage des Données

Hormis GBRLTAR, tous les ensembles de données ont été importés depuis internet. Ils sont ensuite, GBRLTAR inclut, stockées dans les fichiers locaux sous forme d'arborescence de dossiers ([figure 15](#)). La génération des ensembles de boîtes limites à partir des données vidéos se déroule comme indiqué dans la [figure 16](#), les ensembles pré-coupés comme Market1501, eux, ont soit séparées d'emblée leurs identités dans plusieurs dossiers, soit mis toutes les images aux mêmes endroits, se contentant d'afficher le numéro de classe dans leur nom. Dans ce dernier cas, la création de l'arborescence de dossier séparant les images par identité peut être simplement automatisée ([figure 17](#)). ReIDMultiSet est également un dossier tel que ceux représentés dans la [figure 15](#).

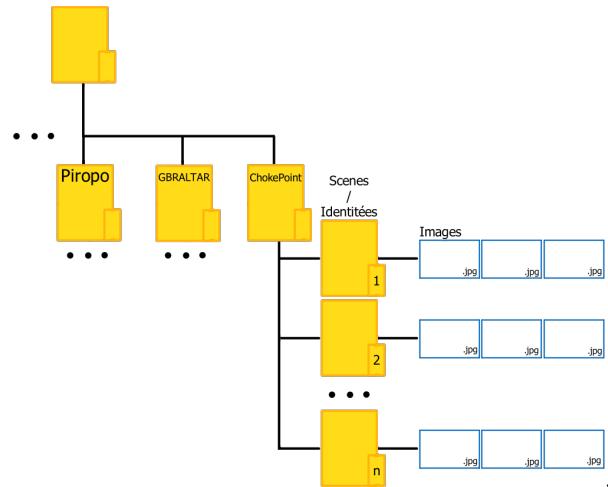


FIGURE 15 – Arborescence de dossiers servant d'espace de stockage aux données d'entraînement et de test. Les ensembles vidéos comme ChokePoint Dataset possèdent deux dossiers, un stockant les boîtes limites séparées par identités, et un autre contenant les scènes ou plans de caméras, dont le format vidéo a été décomposé en images.

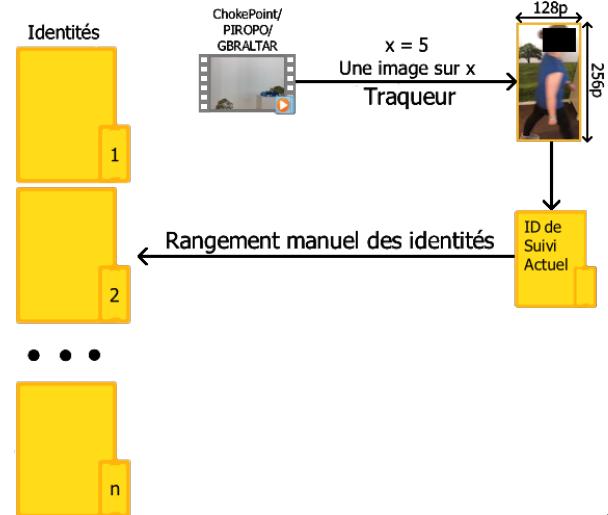


FIGURE 16 – Schéma de la génération des ensembles de données depuis les vidéos de ChokePoint Dataset, PIROPO Database et GBRLTAR. Le Traqueur est déjà capable de ranger les boîtes limites dans des dossiers spécifiques à un passage (i.e. spécifique à un comptage), la sortie du Traqueur est donc une série de dossiers ayant déjà, en majorité, qu'une seule identité de représenté. Le travail manuel restant est celui de fusionner ces dossiers afin de n'en avoir plus qu'un seul pour chaque identité distincte.

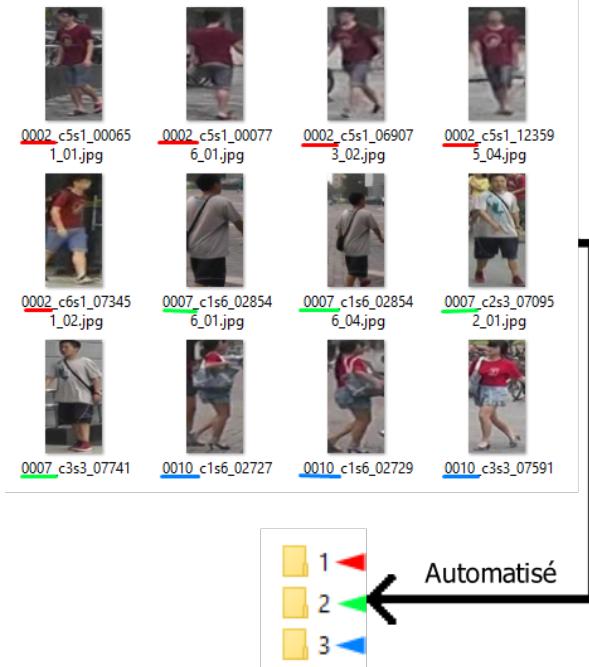


FIGURE 17 – Les numéros de classes pour chaque image sont récupérés et utilisés pour ranger les fichiers dans les dossiers correspondants. Dans le cas de Market1501, les numéros de classes ne sont pas linéaires de 1 à n, ils sont donc modifiés pour le devenir.

### 3.4 Analyse et Nettoyage des Données

Les données utilisées ont été créées par des organismes professionnels afin de permettre leur récupération et utilisation instantanées, de ce fait le nettoyage nécessaire n'est pas conséquent. La nature d'image de ces données signifie aussi qu'une majorité des éléments analysables sont subjectifs. Plusieurs pistes quant aux caractéristiques à analyser : le format de l'image (e.g. taille, résolution), le contenu de l'image (e.g. diversité d'ethnies, de genres, de poses, d'habits, qualité de la découpe des boîtes limites) et les caractéristiques de l'ensemble de données d'où les images proviennent (e.g. quantité d'images par identités, nombre d'identités). Certaines analyses ont déjà été effectuées, couvrant une ou plusieurs des catégories mentionnées, dans l'objectif de trier les ensembles de données par élimination et de ne garder que la sélection actuelle, les conclusions de ces analyses sont discuté dans la sous-section 3.2.

Les variables telles que la taille des images, est une standardisation dont la décision est notre, 256 pixels de haut pour 128 de large soit deux fois les di-

mensions d'une image de Market1501 est le format choisi par délibérations internes, la résolution quant à elle, ne reflète finalement pas de la qualité de l'image, plusieurs images floues possèdes un nombre de Pixels Par Pouces comparable à une image nette. La diversité des ethnies et des genres est une information permettant de favoriser l'absence de biais dont souffre certains algorithmes d'IA, cette information n'est cependant pas toujours disponible, Bien que Market1501 et ChokePoint Dataset nous informe de la répartition hommes-femmes, ce n'est pas le cas de Piropo. Aucun catalogue, peut être à raison, la distribution ethnique. N'utilisant pas la totalité des données de Market1501 et ChokePoint, un comptage manuel est nécessaire, cela nous permet d'obtenir la [table 2](#). ChokePoint indique une distribution homme-femme de 19-6 sur sa "porte 1" et de 23-6 sur la "porte 2", cependant les identités de chaque porte ne sont pas distinctes et il est donc très probable que le compte de la [table 2](#) soit représentatif de l'entièreté du ChokePoint Dataset.

	Femmes	Hommes
ChokePoint	6	21
PIROPO	5	14
Market1501	14	22
GBRALTAR	4	3
Total Entrainement	25	57
Total	29	60

TABLE 2 – Distribution des genres pour chaque partie utilisée des ensembles de données choisies, aucune donnée n'est disponible pour PRID et un comptage manuel n'était pas une option. Sans trop de surprises, le nombre d'entités féminines est largement inférieur.

La diversité de poses et d'habits ainsi que la qualité de la découpe des boîtes limites sont de bons exemples de variables subjectives, aucun indicateur numérique n'étant disponible pour les décrire. Un nettoyage manuel est cependant effectué les concernant, prenant aussi en compte la qualité d'image, dissocié de la valeur de résolution. Le nettoyage ne prend en considération que les ensembles de données provenant originellement de vidéos ([figure 10](#)), effectivement la méthode utilisée pour les transformer en ensembles de boîtes limites, nous procure un nombre assez imposant d'images similaire. Le nombre d'images par secondes étant généralement très haut, une boîte limite prise à l'image n, sera probablement hautement semblable à celle de l'image n+1, pour contrer ce fait, les boîtes limites sont récupérées toutes les 5 images, cela n'empêche cependant pas entièrement le problème étant donné qu'une identité peut (le cas le plus visible étant

dans GBALTAR) traverser l'écran de l'exacte même façon qu'une fois précédente.

La qualité des boîtes limites est un problème causé par deux facteurs. Dans le cas des ensembles pré-construits et sur certaines vidéos, la personne détectée est loin, et donc la qualité de l'image diminue. Autrement, le problème peut être causé par les quelques images où une personne sort ou rentre dans le cadre, elle n'est donc pas entièrement à l'écran, mais est tout de même détecté par YOLOV3, le résultat est la capture d'une boîte limites réduite et de mauvaise qualité. Les solutions furent d'imposer une taille minimale aux boîtes limites, ce qui réduit les distorsions lors de leur normalisation à la taille (256, 128), et de réduire le champ de capture des boîtes limites, de façon à ce qu'elle ne soit enregistrées que dans le cas où une personne est en entier dans le champ de la caméra.

Toutes les images de ChokePoint furent conservées à l'exception des changements de vêtements intra-identités comme expliqué dans la [figure 10a](#). En revanche, GBALTAR a subit une sélection arbitraire permettant de réduire le nombre d'images trop proches les unes des autres, environ deux tiers

des boîtes limites de chaque identité furent supprimées, à l'exception de la classe G. PIROPO ayant une profondeur de champ importante, le problème de qualité d'image y est pleinement visible, la boîte limite appartenant à une identité dans le fond de la pièce étant généralement difficile à ré-identifier même pour un humain. Sans possibilité d'automatisation, un nettoyage manuel fut appliqué, tout comme GBALTAR, les images trop similaires ont également été supprimées. Certaines identités de PIROPO possédant plusieurs milliers d'images, d'autre quelques centaines, et d'autres encore quelques dizaines, la décision fut de limiter ce nombre à un maximum de 150, ainsi certaines classes virent leur nombre d'images diminuer de plus de deux mille, là où d'autres sont resté intactes.

Les caractéristiques descriptives des jeux de données utilisées ont déjà été expliquées par les figures [13](#) et [14](#). PIROPO et GBALTAR y sont utilisés dans leurs intégralités, et ChokePoint manque quelques images par identités. En revanche n'est utilisé qu'une partie des classes de Market1501, un graphique est donc proposé en la [figure 18](#).

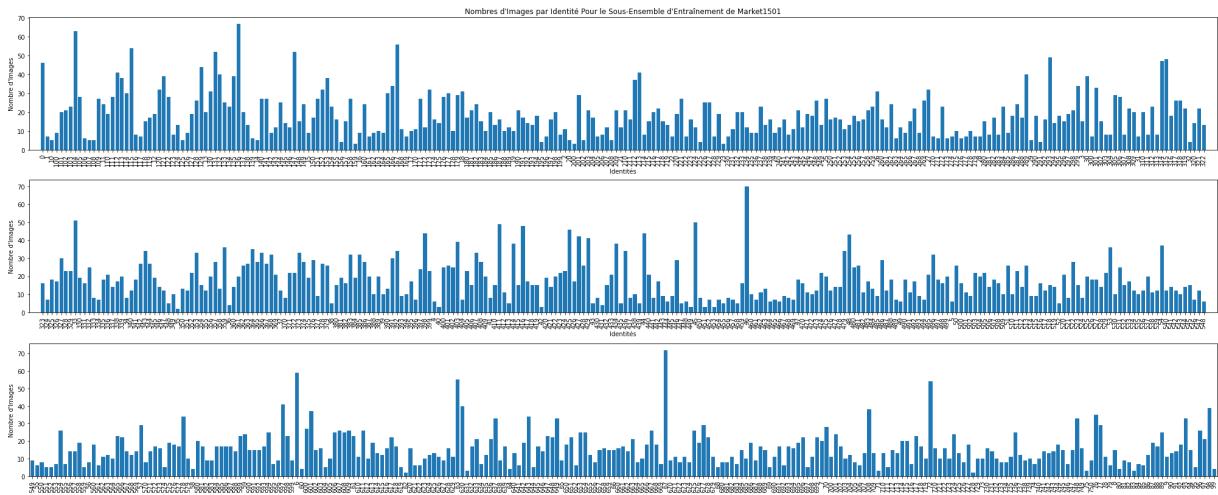


FIGURE 18 – Graphique représentant l'intégralité des classes du sous-ensemble d'entraînement de Market1501, et le nombre d'images pour chacune d'entre elles.

## 4 Les Modèles

L'enchaînement des composants qui ensemble, représentent le modèle de comptage assisté de ré-identification, devrait être bien connue à ce stade, néanmoins voici un récapitulatif en la [figure 19](#).

### 4.1 Les Modèles de Suivi d'objets

L'image est fournie au premier composant qu'est le détecteur d'objets, dans notre cas YOLOV3. Le modèle choisi peut être remplacé facilement par autre algorithme, les seules actions à effectuer se-

raient de modifier l'image d'entrée en conformité avec les paramètres acceptés par le nouveau modèle, et d'adapter la sortie pour fonctionner avec le Traqueur.



FIGURE 19 – Le modèle complet est composé de plusieurs sous-modèles modulables : le détecteur d’objets, le Traqueur et le Classifieur. Le détecteur d’objets et le Traqueur sont utilisés conjointement pour le suivi d’objets, tandis que le Classifieur, composé d’une dorsale (eng : backbone) et d’une sur-couche de réseau de neurones profond, se charge de l’encodage des boîtes limites dans un format rendant apte à la ré-identification. La ré-identification en elle-même est le résultat d’opérations sur une matrice de similarité euclidienne, ces opérations sont calculées dans l’objet Python qu’est le Traqueur.

#### 4.1.1 Récupération des Boîtes Limites

Le fonctionnement de YOLOV3 a déjà été discuté dans la [sous-section 2.2](#), cette partie ira plus en profondeur dans son implémentassions et insertion dans le pipeline, ainsi que sur le pré-traitement des images qui lui sont fournis en entrées.

Une classe "frameProcessor" est créée acceptant les images extraites du flux vidéo, elle est responsable du pré-traitement de l'image, de la gestion du détecteur et de la mise à jour du Traqueur. Lors de l'initialisation de l'instance, le fichier ONNX contenant l'architecture et les poids de YOLOV3, précédemment téléchargé sur [la page des modèles](#) sur le github de ONNX, est utilisé pour créer un objet de la classe "InferenceSession", de la bibliothèque onnxruntime. Chaque fois qu'une image est reçue par l'instance de "frameProcessor", la fonction "preprocessingYolo" est utilisée. Cette fonction est décrite dans l'[algorithme 1](#). Une fois l'image modifiée obtenue, la fonction "run" de "InferenceSession" est utilisée, avec comme entré l'image à la bonne taille et le vecteur contenant les dimensions originales de l'image, cette fonction nous permet de récupérer en

utilisant le modèle choisi, les boîtes limites et les scores de certitude pour chaque classe.

**Algorithme 1** Calculs effectués par la fonction pre-processingYolo, afin de préparer les données d'entrée à YOLOV3.

**Arguments :** img : image au format Pillow ; taille : tuple de la taille d'image accepté par le détecteur, au format (largeur, hauteur)

- 1:  $iw, ih \leftarrow img.size$  (Récupère la largeur et la hauteur de l'image d'entrée.)
- 2:  $w, h \leftarrow taille$  (Récupère la largeur et la hauteur voulue.)
- 3:  $echelle \leftarrow min(w/iw, h/ih)$  (Récupère la plus petite échelle pour agrandir l'image dans une des directions.)
- 4:  $nouveauW \leftarrow int(iw * echelle)$  (Récupère la nouvelle largeur grâce à l'ancienne et à l'échelle.)
- 5:  $nouveauH \leftarrow int(ih * echelle)$  (Récupère la nouvelle hauteur grâce à l'ancienne et à l'échelle.)
- 6:  $imgR \leftarrow img.resize((nw, nh), Image.BICUBIC)$  (Change la taille de l'image selon la nouvelle largeur et hauteur.)
- 7:  $padBG \leftarrow Image.new('RGB', taille, (128, 128, 128))$  (Crée une image de rembourrage RGB grise de la taille voulue.)
- 8:  $padOffset \leftarrow ((w - nouveauW)//2, (h - nouveauH)//2)$  (Récupère les coordonnées pour centrer l'image sur l'image de rembourrage.)
- 9:  $padBG.paste(imgR, padOffset)$  (Fusionne l'image à l'image de rembourrage afin d'obtenir une image rembourré de la bonne taille.)
- 10:  $imageData \leftarrow np.array(paddingBG, dtype = 'float32')$  (Convertie l'image en matrice numpy.)
- 11:  $imageData \leftarrow imageData/255$  (Normalise l'image entre 0 et 1.)
- 12:  $imageData \leftarrow np.transpose(imageData, [2, 0, 1])$  (Change l'ordre des dimensions pour obtenir [canaux, hauteur, largeur].)
- 13:  $imageData \leftarrow [imageData]$  (Englobe la matrice de l'image dans une dimensions supplémentaire comme demandé par YOLOV3.)

**Sortie :**  $imageData, [[w, h]]$  (Retourne l'image à la bonne taille ainsi qu'une liste contenant les dimensions d'origine.)

Seuls les scores attribués à la classe "Humain" sont conservées, la page github du YOLOV3 utilisé n'indique pas le format des informations sur les boîtes limites, mais il fut découvert qu'elles représentaient dans l'ordre : les coordonnée  $y$  basses, les coordonnées  $x$  de gauche, les coordonnées  $y$  du haut et les coordonnées  $x$  de droite.

#### 4.1.2 Traqueur par plus Courte Distance des Centroïdes

Les scores et coordonnées sont ensuite passés à une instance de la classe "Traqueur", c'est dans cette classe que se déroule le suivi et la ré-identification, seul le suivi sera pour l'instant discuté. La fonction principale, celle qui est appelée par la classe "frameProcessor", est la fonction "update", elle fait appel à plusieurs autres fonctions afin de réaliser le processus suivant : une image est reçue, avec elle un lot de boîtes limites avec leurs scores de similarités respectif; la fonction "getB-Box" ([algorithme 2](#)) est appelé afin de ne conserver que les boîtes limites dont le score de certitude se trouve supérieur à une valeur donnée entre 0 et 1 ; une fonction "NMS" appliquant une suppression non-max (la suppression non-max est une technique populaire pour laquelle des explications et méthodes d'intégration sont facilement trouvables en ligne) est utilisée afin de réduire le nombre de boîtes limites qui pourraient décrire la même identité; à partir des boîtes limites restantes, les centroïdes sont récupérés ([algorithme 3](#)) ; si une nouvelle personne est détecté, la fonction "register" ([algorithme 4](#)) s'occupe d'en faire une instance de la classe "Person" ; une matrice des distances euclidiennes entre les nouveaux et anciens centroïdes est récupéré via la fonction Scipy "cdist" ; utilisant cette matrice, les assignations sont faites entre les centroïdes de l'image précédente et ceux de l'image actuel, cette assignation n'est pas effectué si la distance est supérieur à une valeur donnée ; les centroïdes n'ayant pas été assigné sont comptées comme de nouvelles identités. C'est également dans la fonction "update" que l'image est modifié pour afficher les centroïdes et boîtes limites, et que la fonction "updateP" des instances de la classe "Person" est appelé afin de mettre à jour les informations relatives à cette identité (historique des scores et des positions). Dépendant des paramètres, la fonction "update" retourne une liste contenant : le dictionnaire dont les clés sont les identités et les valeurs sont des listes servant à stocker la version encodée du contenu de la boîte limite pour cette identité et l'horodatage de la capture ; et/ou l'image actuelle, qu'elle soit modifiée ou non.

---

#### Algorithme 2 Pseudo-code de la fonction getB-Box.

---

**Arguments :** bbox : liste de toutes les boîtes limites générée par le détecteur ; scores : liste des scores de certitude de la présence d'un être humain pour chaque boîte limite

```

1: idGardes  $\leftarrow \emptyset$  (Crée une liste vide pour contenir les indexées des boîtes limites conservées.)
2: scoresGardes  $\leftarrow \emptyset$  (Crée une liste vide pour contenir les valeurs des scores de boîtes limites conservées.)
3: pour  $i, v$  dans enumerate(scores) faire:
4:   si  $v >$  Constante paramétrable entre 0 et 1 alors:
5:     AJOUT(idGardes,  $i$ )
6:     AJOUT(scoresGardes,  $v$ )
7:   fin si
8: fin pour

9: bboxRetour  $\leftarrow \emptyset$  (Crée une liste vide pour contenir les boîtes limites conservées.)
10: pour  $idNum$  dans idGardes faire:
11:   AJOUT(bboxRetour,
12:           bbox[ $idNum$ ].astype('int'))
12: fin pour

13: si Constante booléenne définissant si oui ou non la suppression non-max doit être appliquée est défini comme Vrai alors:
14:   Sortie : NMS(bboxRetour, scoresGardes, idGardes)
14: sinon:
15:   Sortie : bboxRetour, scoresGardes
15: fin si
```

---

#### Algorithme 3 Pseudo-code de la fonction getCentroid.

---

**Arguments :** yBas, xGauche, yHaut, xDroite : coordonnées récupérés des boîtes limites restantes après les divers tris

```

1: CentroidX  $\leftarrow round((xDroite + xGauche)/2)$ 
   (Calculs de la coordonnée x du centroïde d'une boîte limite.)
2: CentroidY  $\leftarrow round((yHaut + yBas)/2)$ 
   (Calculs de la coordonnée y du centroïde d'une boîte limite.)
```

**Sortie** : [*CentroidX*, *CentroidY*]

---

---

**Algorithme 4** Pseudo-code de la fonction register.

**Arguments :** CentroidCoord, Bbox, Score : les coordonnées du centroïde, les coordonnées de la boîte limite et le score de certitude.

- 1: avec `self.nextID` une variable liée à la classe Traqueur qui s'auto-incrémente à chaque enregistrement d'une identité
  - 2: avec `self.objectsIn` un dictionnaire lié à la classe Traqueur qui store les objets "Person" durant leur suivi
  - 3: avec `self.objectsOut` une variable liée à la classe Traqueur qui store le nombre d'images durant lesquelles l'identité n'était pas visible durant leur suivi
  - 4: `self.objectsIn[self.nextID] ← personObject`
  - 5: `self.objectsOut[self.nextID] ← 0`
- 
- 6: `self.nextID ← self.nextID + 1`

Durant le suivi, une identité peut disparaître, que ce soit par une erreur temporaire du détecteur ou tout simplement dû à sa sortie du cadre de la caméra. De ce fait, deux fonctions supplémentaires sont utilisées, la première ([algorithme 5](#)) est appelée par la fonction "update" lorsqu'une identité précédemment visible ne l'est plus et incrémenté un compteur, la deuxième ([algorithme 6](#)) est appelé par la première lorsque le compteur dépasse une constante paramétrable afin de retirer l'identité de la liste des objets suivis. Une autre constante paramétrable est utilisée dans la deuxième fonction, conjointement avec un autre compteur qui lui, compte le nombre d'images durant lesquelles l'identité était visible, et si ce nombre est inférieur à la constante, cette identité est considéré comme étant une erreur et supprimé du total d'objets comptés.

---

**Algorithme 5** Pseudo-code de la fonction outOfFrame.

**Arguments :** ID : l'identifiant de l'identité absente de l'image actuellement en traitement

- 1: `self.objectsOut[ID] ← self.objectsOut[ID]+1`
- 2: **si** `self.objectsOut[ID] > Constante paramétrable alors:`
- 3:   `dropTracking(ID)`
- 4: **fin si**

1. `ratioDésiré` est défini selon `self.desiredOutputShape[0]/self.desiredOutputShape[1]`; `self.desiredOutputShape` étant un tuple de constantes représentant la hauteur et la largeur désiré des boîtes limites

---

**Algorithme 6** Pseudo-code de la fonction dropTracking.

**Arguments :** ID : l'identifiant de l'identité ayant été absente suffisamment longtemps pour arrêter son suivi

- 1: `self.TrueIdDict ← un dictionnaire lié à la classe "Traqueur" qui store la totalité des objets "Person"`
  - 2: **si** temps passé par `self.objectsIn[ID]` devant la caméra > Constante paramétrable **alors:**  
    `DELETE(self.TrueIdDict, ID)`
  - 3: **fin si**
  - 4: `DELETE(self.objectsIn, ID)`
  - 5: `DELETE(self.objectsOut, ID)`
- 

## 4.2 Le Modèle de Ré-Identification

La ré-identification se situe au sein de la fonction "update", lorsque une identité est comptée comme rentrant dans le cadre pour la première fois, elle est enregistrée dans le dictionnaire `self.objectsIn`, une série de questions sont posées :

1. Est-elle présente suffisamment longtemps pour ne pas être considéré comme une erreur du détecteur ?
2. Est-ce que le ratio longueur-largeur de sa boîte limite est compris dans un intervalle de  $ratioDsir \pm 0.5^1$  ?
3. Est-ce que le centroïde se situe dans une zone centrée de l'image comprenant 60% de cette dernière ?

Si toutes les réponses sont positives, alors le contenu de la boîte limite est extrait, sa taille est modifiés pour convenir au format souhaité et est encodé par le Classifieur. Afin de réduire les temps de calcul et les occasions pour qu'une erreur d'attribution survienne, ou simplement lorsqu'elle n'est pas utile ou demandée, la ré-identification ne s'opère pas si l'un des cas suivant est vrai :

- Aucune identité n'est déjà enregistrée.
- Toutes les identités pour lesquelles une personne entrante pourrait être ré-identifié comme étant, sont déjà en cours de suivi.
- L'identité en elle-même a déjà un suivi actif, auquel cas l'encodage est simplement ajouté à la liste d'encodages pour cette personne.

- La constante booléenne `self.ReIdentification`, lié à la classe "Traqueur", est défini comme fausse.

À défaut, la ré-identification prend place suivant les instructions de l'[algorithme 9](#). Une métrique fut développée pour l'occasion, originellement, la décision d'attribution était basé sur la classe la plus présente dans le top x des images les plus similaires. Cette méthode part du postulat que la ré-identification est forcément possible, i.e. que des images de l'identité sont forcément déjà disponibles, et ne peut donc pas être utilisé. Inspirés du métrique de la moyenne des précisions moyennes (eng : Mean Average Precision ou mAP), nous développons mTXS [ $\varepsilon_{mTXS}$ ] pour Moyenne des scores de Similarité du Top X (mean TopX Similarity) qui permet d'obtenir un score de certitude entre 0 et 1 et qui donnent une importance décroissante aux images du top X.

Comprendre le fonctionnement de la mAP peut aider à la compréhension de mTXS, il s'agit d'une métrique de performances appliquée à des problèmes de classification spécifiques comme le contrôle des capacités de modèles de détections ou de récupération de documents, dans notre cas, la précision mentionnée dans le nom n'est pas la même que celle utilisée habituellement en classification ([équation 2](#)), mais une version adaptée à la récupération d'objets ([équation 3](#)). La précision moyenne (AP) pour X objets récupérés est ensuite calculé itérativement en utilisant la formule de précision sûr chacun d'entre eux, balancé par une fonction de pertinence ([algorithme 7](#)) retournant 1 ou 0 dépendant de si l'objet est de la bonne ou mauvaise classe, le tout est divisé par le nombre total d'objets corrects à retourner ([équation 4](#)). La mAP est ensuite tout simplement la moyenne des AP de plusieurs requêtes différentes ([équation 5](#)).

$$\text{précision} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Positifs}} \quad (2)$$

$$\text{précision} = \frac{|\text{docs pertinents} \cap \text{docs récupérés}|}{|\text{docs pertinents}|} \quad (3)$$

$$AP@X = \frac{\sum_k^X \text{précision}@k * Pfunc(\text{classe}@k)}{|\text{vrais positifs attendus}|} \quad (4)$$

Avec N = nombre de requêtes de test

$$mAP = \frac{1}{N} \sum_{i=0}^N AP_i \quad (5)$$

---

**Algorithme 7** Pseudo-code de la fonction de pertinence "Pfunc".

---

**Arguments :** classe@k : classe de l'objet récupéré

```

1: si classe@k = Vrai Positif alors:
2:   Sortie : 1
3: sinon:
4:   Sortie : 0
5: fin si
```

---

mTXS s'apparente à la mAP dans le sens où à son instar, le procédé est itératif, appliquant une fonction à chaque élément récupéré dans le domaine d'un top X et calculant une moyenne, les différences sont sur la fonction utilisée et le fait que mTXS fonctionne par classe. Une valeur entre 0 et 1 est générée pour chaque identité et celle avec le plus haut score, si supérieur à une constante paramétrable entre 0 et 1, est attribué à la personne cible de la ré-identification.

---

**Algorithme 8** Pseudo-code de mTXS.

---

**Arguments :** topX : table des scores et classes

```

1: mTXS ← []
2: pour chaque classe "C" dans le topX faire:
3:   topX@C ← table contenant les indexes et
      scores de similarités des images dans le topX
      appartenant à la classe C
4:   mTXS@C ← []
5:   pour score, index dans topX@C faire:
6:     AJOUT(mTXS@C, score / ( $\log_e(index + 1 + 10^{-6}) + 1$ ))
7:   fin pour
8:   AJOUT(mTXS,
      [MOYENNE(mTXS@C, C)])
9: fin pour
10: prédiction ← MAX(mTXS, clé = 0)
Sortie : prédiction
```

---

La formule ligne 6 de l'[algorithme 8](#) possède différentes additions servants de mécanismes de sécurité, la première addition est utile car les indexes commences à 0, où le logarithme n'est pas défini, la seconde sert à éviter une division par 0, et la troisième empêche les indexes inférieurs à 3 de diviser par une valeur en dessous de 1, gardant le mTXS entre 0 et 1. La division par le logarithme fut rajouté suite à l'obtention de meilleur résultats en donnant plus d'importance à la première image retournée, gagnant 11% d'exactitude sur GBRLTAR.

---

**Algorithme 9** Pseudo-code de la ré-identification.

**Arguments :** encode : encodage vecteur de la boîte limite de l'identité sur laquelle la ré-identification est effectué

```
1: self.mTXSThreshold ← flottant paramétrable entre 0 et 1 lié à la classe "Traqueur"
2: self.classGlob ← liste liée à la classe "Traqueur" contenant chaque identifiant
3: self.codeGlob ← liste liée à la classe "Traqueur" contenant chaque vecteur d'encodage

4: classGlobTMP ← copie de self.classGlob
5: codeGlobTMP ← copie de self.codeGlob

6: INSERT(classGlobTMP, 0, ?) (Insert au début des listes : le nouvel encodage, ainsi qu'une valeur
   représentant une classe non-définie.)
7: INSERT(codeGlobTMP, 0, encode)

8: SimScores ←  $1 - \text{distanceEuclidienne}(\text{self.codeGlob}, \text{self.codeGlob}) / 127.5$  (La
   matrice des distances de chaque encodage à un autre est divisée par la "distance max", afin d'être
   normalisé entre 0 et 1. Au lieu d'utiliser la valeur maximum dans la matrice, la valeur de la marge
   de la fonction de coût par triplet utilisé lors de l'entraînement du modèle fut choisie. L'assumption
   est que définir une constante permet de ne pas forcer la ré-identification d'une nouvelle identité en
   stabilisant les scores de certitudes.)

9: SimScores.diagonal ← -100 (Supprime la diagonal pour ne pas prendre en compte l'image
   d'entrée comme un choix viable.)
10: vecteurReID ← table contenant les scores de similarités et les classes associées du premier vecteur
    de "SimScores", qui correspond au score de similarité entre le nouvel encodage et les anciens
11: topX ← X premières entrées de "vecteurReID" trié par ordre décroissant en fonction du score de
    similarité

12: prédiction ← mTXS(topX) (Récupère la prédiction et le score associé via la fonction mTXS.)
13: si prédiction[0] ≥ self.mTXSThreshold alors:
14:   identifiant de la personne ← prédiction[1]
15:   AJOUT(self.TrueIdDict[identifiant de la personne], [encode, date et heure de l'opération])
16: sinon:
17:   UPDATE(self.TrueIdDict, identifiant de la personne : [encode, date et heure de l'opération])
18: fin si
19: AJOUT(self.classGlob, identifiant de la personne)
20: AJOUT(self.codeGlob, encode)
```

---

#### 4.2.1 Architecture

Comme déjà mentionné dans la [sous-section 2.3](#), le modèle de ré-identification est composé d'une dorsale en l'extracteur de caractéristiques ResNetV2 combiné à un classifieur profond, le schéma de leur architecture est disponible [figure 20](#). Les paragraphes suivants expliqueront plus en détail : les blocs résiduels de ResNetV2, l'architecture du Classifieur couche par couche et les fonctions de coûts utilisés.

La famille de modèles ResNet sont des réseaux de neurones à convolutions très profonds. Pour atteindre une telle profondeur sans subir les problèmes qui y sont liés comme la disparition du gra-

dient ([figure 21](#)), les blocs résiduelles furent développé. L'idée est qu'un réseau de neurones ne devrait pas donner de moins bons résultats que son équivalent moins profond, pour ce faire les blocs résiduels sont composées de "raccourcis", ces raccourcis additionne la matrice d'identité aux canaux de sortie de chaque bloc, autrement dit l'entrée du modèle. L'addition permet au gradient d'être rétro-propagé à part égal aux deux entrées (le raccourci et les couches). Les couches sont actualisées ce qui modifie le gradient sur leur branche. Les deux gradients sont ensuite additionnés ([\[22\]](#)) pour être de nouveau distribués au prochain bloc [\[28, 26\]](#). Des informations contenue dans les premières sorties de couches peuvent aussi être conservées ce qui dans certains cas, permet d'améliorer les résultats.

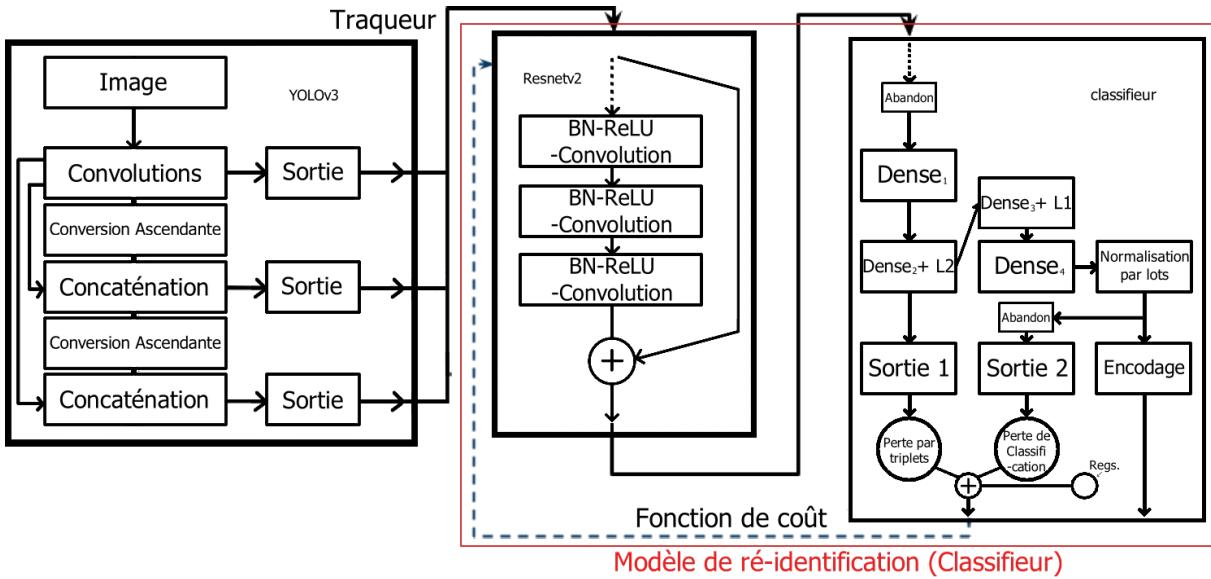


FIGURE 20 – Schéma du type de données reçu en entrée à chaque partie de l’application.

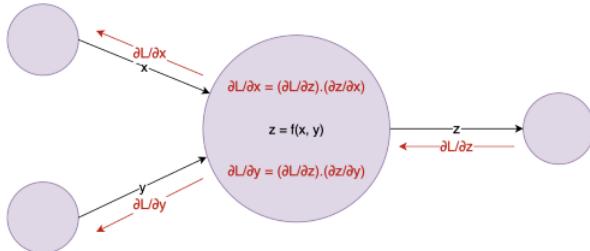
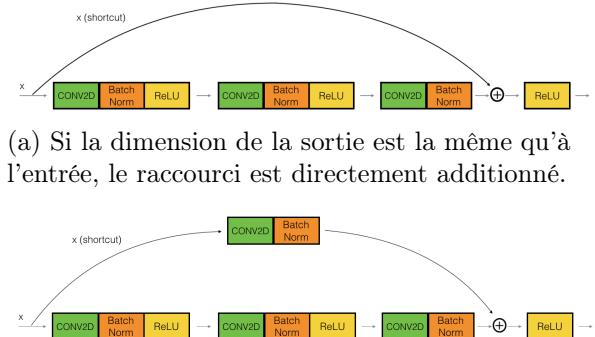


FIGURE 21 – Raisonnement derrière la disparition du gradient. Les dérivées rétropropagées de la couche n-1 comprenant les neurones x et y ( $\frac{\partial L}{\partial x}$  et  $\frac{\partial L}{\partial y}$ ), sont obtenues par la multiplication entre la dérivé arrivante :  $\frac{\partial L}{\partial z}$  et les dérivées respectives du gradient local des poids de la neurone couche n :  $[\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}]$ . L étant la perte. La multiplication successive de valeurs en dessous de 1 réduit le gradient jusqu'à sa disparition.

L’illustration de l’addition des gradients figure 23, occurrent durant la rétropropagation du gradient dans les blocs résiduel, est expliqué par l’équation 6.

$$\begin{aligned}
 y &= x + F(x) \\
 \frac{\partial L}{\partial x} &= \frac{\partial L}{\partial y} * \frac{\partial y}{\partial x} \\
 &= \frac{\partial L}{\partial y} * (x' + F'(x)) \\
 &= \frac{\partial L}{\partial y} * (1 + F'(x)) \\
 &= \frac{\partial L}{\partial y} + \frac{\partial L}{\partial y} * F'(x)
 \end{aligned} \tag{6}$$



(a) Si la dimension de la sortie est la même qu’à l’entrée, le raccourci est directement additionné.

(b) si la dimension de la sortie n’est pas la même qu’à l’entrée, le raccourci est modifié par une convolution.

FIGURE 22 – Types de raccourcis des blocs résiduels, les composants des ResNets. En l’occurrence l’architecture est celle de ResNetV2.

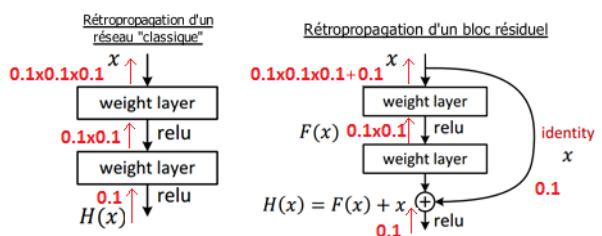


FIGURE 23 – À gauche, exemple de rétropropagation d’un réseau de neurones “classique”, à droite, exemple de rétropropagation du gradient à travers un bloc résiduel avec une connexion de raccourci sans mise à l’échelle. La valeur entrante 0.1 est un exemple du gradient que peut retourner une perte ou des neurones d’une couche n+1.

Les sorties des couches de convolutions de ResNetV2 sont réduites à un vecteur de taille 2048 par une couche de Sous-échantillonnage par Valeur Maximale (eng : Max Pooling). Le Classifieur qui accepte ce vecteur en entrée est composé de couches d'Abandons (eng : Dropout), Denses et de Normalisation par Lots (eng : Batch Normalization), il utilise la fonction de coût par triplet afin de favoriser la séparation euclidienne, cumulée à une perte par entropie croisée afin d'obtenir une meilleure séparation cosinus. Les couches d'Abandons sont paramétrées avec un taux de 20%, les couches Denses 2, 3 et 4 (figure 20) possède chacune 256 neurones, là où Dense 1 en possède 2048, afin de coller avec la sortie de ResNet. La couche de Normalisation par Lots vient du papier déjà mentionné "BNNeck", de Luo et al. [21], elle est utilisée avant la couche de sortie de classification, une couche Dense avec autant de neurones que le nombre de classes d'entraînement, afin d'obtenir des valeurs d'activations entre 0 et 1 pour la fonction de coût de classification de nature cosinus.

Nous avons donc un vecteur de caractéristiques extraites par ResNet, qui est transmis à plusieurs couches Dense résultant en un encodage de taille 256. Cet encodage est utilisé pour la perte par triplet et est également transmis à deux autres couches Denses, le résultat est normalisé entre 0 et 1 en utilisant la plus grande valeur d'activation présente dans le lot à ce stade. C'est cet encodage normalisé qui sera utilisé pour la ré-identification, il est également utilisé pour calculer la perte par entropie croisée résultant de la prédiction des classes.

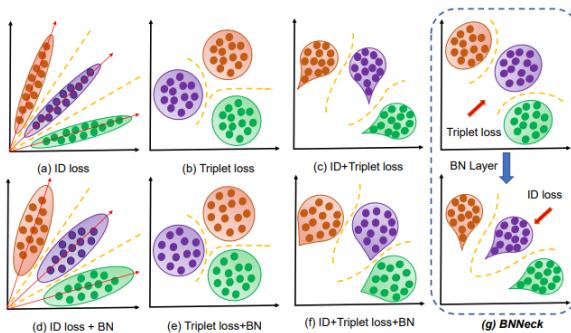


FIGURE 24 – Représentations latentes 2D d'encodage selon différentes architectures et fonctions de pertes. Image issue de "A Strong Baseline and Batch Normalization Neck for Deep Person Re-identification" par de Luo et al. [21]

#### 4.2.2 Entraînement

L'ensemble de données d'apprentissage "ReID-MultiSet" discuté dans la sous-section 3.2 est

d'une complexité satisfaisante, l'architecture du modèle est également capable de réduire le sur-apprentissage et de ce fait, l'augmentation en ligne des données n'est pas conséquente. ReIDMultiSet est séparé de la façon suivante : 85% pour l'entraînement (sois 3053 images appartenant à 82 classes.) et pour la validation (sois 488 images appartenant à 82 classes.). Les images du sous-ensemble d'entraînement sont occasionnellement retournées horizontalement, les retourner verticalement n'ayant pas de sens pour des images de passants. D'autres modifications étaient originellement appliquées durant l'augmentation, mais elles se sont prouvées néfastes pour le bon fonctionnement du modèle. Une seul autre modification subsiste en la suppression aléatoire, cette méthode sélectionne un rectangle de taille et position aléatoire dans une image, et y remplace les pixels d'origines par du gris (l'algorithme 10).

Le taux d'apprentissage est défini comme  $lr = 3.5 * 10^{-4}$  et l'entraînement ce fait sur 100 époques avec une patience de 25, signifiant l'arrêt de l'apprentissage si les résultats ne s'améliorent pas après 25 époques. Chaque matrice de poids est enregistrée en fichier ".h5", l'infrastructure Azure-Databricks en place impose cependant leur suppression à l'arrêt du noyau de calcul. Un phénomène dont la découverte est le fruit de la sérendipité est la soudaine baisse des erreurs et le gain en exactitude résultant de l'absence de normalisation durant le pré-traitement des données, peut-être est-ce dû à la présence d'une couche de normalisation par lots, mais cette étape standard rend notre modèle, en l'occurrence, inutilisable. De ce fait, la marge de la fonction de coût par triplet (figure 25) est augmentée pour atteindre 127.5, valeurs qui après plusieurs essaies semble être la plus stable. "TripletSemiHardLoss" de keras-tensorflow est utilisé, il s'agit d'un type de méthode par triplet où des triplets appartenant à des classes que le modèle a du mal à séparer jusqu'à une perte de 0 sont utilisés. La fonction d'entropie croisée utilisé est également l'implémentassions par keras. Tous les entraînements sont réalisés avec la graine 420 pour les générations de sous-ensembles et de lots.

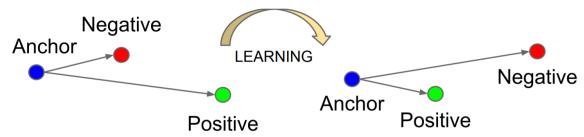


FIGURE 25 – Schéma explicatif de la fonction de coût par triplet.

---

**Algorithme 10** Pseudo-code de la fonction de suppression aléatoire, utilisé durant l'augmentation en ligne des données d'entraînement.

---

**Arguments :** img : image de l'ensemble d'entraînement ; proba : valeur décimal entre 0 et 1 décrivant la probabilité que la suppression aléatoire soit appliquée à cette image

```

1:  $p1 \leftarrow RANDOM\_UNIFORM(0, 1) > proba$  (Génère un booléen dépendant de si une valeur entre 0 et 1 obtenue d'une distribution uniforme est supérieur à  $proba$ .)

2: essais  $\leftarrow 0$  (C'est un procédé itératif, les dimensions du rectangle ne sont correctes que si elles rentrent dans l'image, et un nombre limité d'essais est autorisé, la boucle ne tourne que si  $p1$  est vrai, et ce jusqu'à 300 fois.)
3: while  $p1$   $\text{essais} \leq 300$  faire:
4:    $\text{essais} \leftarrow \text{essais} + 1$ 

5:    $\text{dimH} \leftarrow \text{essais} + 1$  (Hauteur de l'image.)
6:    $\text{dimL} \leftarrow \text{essais} + 1$  (Largeur de l'image.)
7:    $\text{surface} \leftarrow \text{dimH} * \text{dimL}$  (Surface de l'image.)

8:    $\text{surfaceRectangle} \leftarrow RANDOM\_UNIFORM(0.15, 0.4) * \text{surface}$  (La surface du rectangle à supprimer est un pourcentage de la surface total, le choix d'entre 15% et 40% est subjectif.)
9:    $\text{ratio} \leftarrow RANDOM\_UNIFORM(0.2, 1)$  (Ratio de la hauteur par rapport à la largeur du rectangle.)
10:   $H \leftarrow ENTIER(RACINE\_CARRE(\text{surfaceRectangle} * \text{ratio}))$  (Hauteur du rectangle.)
11:   $L \leftarrow ENTIER(RACINE\_CARRE(\text{surfaceRectangle} * (1 - \text{ratio})))$  (Largeur du rectangle.)

12:  si  $L < \text{dimLH} < \text{dimH}$  alors:
13:     $x1 \leftarrow RANDOM\_UNIFORM\_ENTIER(0, \text{dimL} - L)$  (Coordonnées x du rectangle.)
14:     $y1 \leftarrow RANDOM\_UNIFORM\_ENTIER(0, \text{dimH} - H)$  (Coordonnées y du rectangle.)

15:     $img[x1 : x1 + L, y1 : y1 + H, :] \leftarrow 127$  (Change les valeurs des pixels pour du gris.)
16:    Sortie : img (Renvoie l'image modifiée)
17:  fin si
18: fin while
19: Sortie : img (Renvoie l'image non modifiée)

```

---

Plus de poids est donné à la perte par triplet, la fonction de coût total est donc  $2 * tripletP + entropieP + k1 * L1 + k2 * L2$ ,  $k1$  et  $k2$  étant des hyper-paramètres de keras, et  $L2$  et  $L1$  étant les régularisations des couches Denses 2 et 3 respectivement. Bien que les pixels d'une image ne soient rarement des données présentant une co-linéarité, l'ajout de la régularisation  $L2$  présente dans la couche Dense 2 améliore la qualité des résultats, il est donc possible d'imaginer que les caractéristiques encodées par ResNet puissent posséder un certain lien, mais rien n'est avéré et une erreur de compréhension n'est pas impossible. Quant à la régularisation  $L1$  de Dense 3, c'est également suite à quelques tests que l'amélioration des résultats par son ajout fut découverts, cette régularisation étant responsable de réduire à 0 les caractéristiques les moins importantes, il est justifiable de penser que sa présence serait préférable dans Dense 1. Dû aux li-

mitations de ressources, peu de tests furent réalisés sur le placement optimal des régularisations et leur position actuelle n'est pas entièrement justifiable. L'amélioration principale suite à ces ajouts est la baisse de sur-apprentissage. Le paramètre Lambda par défaut de keras est gardé, avec  $\lambda = 0.01$  et donc, avec  $W$  étant le vecteur des poids de la couche, et  $|W|$  sa longueur,  $l1 = \lambda * \sum_{i=0}^{|W|} (|W_i|)$  et  $l2 = \lambda * \sum_{i=0}^{|W|} (W_i^2)$ .

Nous avons essayé d'utiliser la mAP comme fonction objectif, mais le coût computationnel bien trop élevé causa des erreurs de type "mémoire insuffisante". À défaut, un rappel (eng : callback) ne calculant la mAP qu'à la fin de chaque époque fut développé, mais ne choisir que les résultats avec la plus haute mAP sur le sous-ensemble de validation ne s'est pas présenté comme gage de qualité sur celui de test.

### 4.2.3 Résultats

N'utilisant pas le nombre brut de bonne prédictions dans le top X pour réaliser la classification d'un échantillon de test, la mAP, bien que toujours utile comme indicateur de performance, est en grande partie remplacé par l'exactitude ([équation 7](#)) pour l'analyse des résultats de tests.

$$f(yPred, yVrais) = \begin{cases} 1 & \text{Si } yPred == yVrais \\ 0 & \text{Sinon} \end{cases} \quad (7)$$

$$\frac{1}{|\text{échantillons}|} * \sum_{i=0}^{|\text{échantillons}|} f(pred_i, vrais_i)$$

Premièrement, observons les résultats sur l'ensemble de données d'entraînement et ses 82 classes. Les figures suivantes composent un exemple de requête, l'image procurée au modèle est affiché [figure 26](#), les scores de similarités et les images associées dans le top X retourné par le modèle [figure 27](#). La situation étant une évaluation sur des images utilisées pour l'entraînement du modèle, il est normal de s'attendre à des résultats proches de la perfection, deux identités sont néanmoins présente dans

le top X, avec en l'occurrence  $X = 10$ . L'identité retournée au positions 4, 5 et 6 est différente de l'identité attendu, qui est présente sur les 7 images restantes, nous pensons que la similarité visuelle des deux identités, physique autant que vestimentaire, est suffisamment conséquente pour justifier en partie l'erreur du modèle. Étant donné que le trajet effectué devant la caméra par chaque identité est le même, il est satisfaisant de voir qu'aucune image de la mauvaise classe, capturée au même moment de la traversé que l'image de requête (à l'instar de la neuvième image du top 10), fut retourné.



FIGURE 26 – Exemple d'image de requête.



FIGURE 27 – Exemple de top X, avec  $X = 10$ , retourné par une requête de ré-identification.

La [figure 28](#) met en images les résultats pour chaque classe de l'ensemble de données, ainsi que les résultats totaux. Sur le graphe de gauche, les différences nettes entre les mAP individuelles nous informe que certaines classes sont plus souvent confondue que d'autres, une étude plus approfondie sur ces classes serait intéressante afin d'apprendre pourquoi. Les hypothèses incluses la présence de plusieurs classes se ressemblant énormément, le manque d'images de bonne qualités et une variation intra-classe trop importante. Il est intéressant de constater que le graphe de gauche est une version plus lissée de celui de droite, la mAP étant basé sur l'ordre dans lequel les vrais positifs sont retournés. Il faut noter que le graphe de droite est arrondi à l'entier supérieur et que là où les résul-

tats sont affichés par top X, la mAP est calculé sur l'entièreté du vecteur de similarité retourné (dans l'ordre décroissant des scores).

Deux ensembles de tests sont utilisés : GBRAL-TAR d'une part et PRID2011 de l'autre. GBRAL-TAR est le plus difficile des deux dus à ses variations intra-classe et à la proximité de la caméra, ne permettant pas toujours la capture de l'entité dans son entier. Durant l'explication du fonctionnement de la mTXS, il fut mentionné que de meilleurs résultats étaient obtenus en donnant plus d'importance au premier échantillon retourné, cette observation nous a conduit à utiliser un top X avec  $X = 1$  durant les tests suivants.

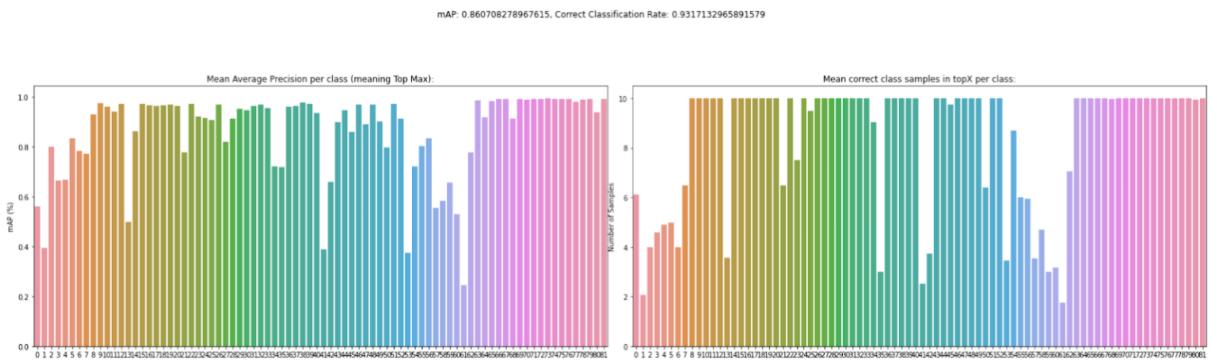


FIGURE 28 – Résultats des tests sur l’ensemble d’entraînement. La mAP est de 86% et l’exactitude de 93%. À gauche, est affiché la mAP par classe, à droite, le nombre moyen, par classe, de vrais positifs dans le top X.

La [figure 29](#) est un exemple des requêtes générées sur GBRLTAR pour le test, les blocs noirs ne sont pas de base sur l’image et sont présentes dans un but de censure dû à la nature confidentielle des images. Il est intéressant de constater la corrélation directe entre la [figure 30](#) et la [figure 14](#), signifiant que le plus de données sur une personne sont disponibles, le plus de chance nous avons pour une ré-identification correcte, ce qui peut être expliquée par les chances croissantes d’avoir un échantillon de la classe  $A$ ,  $A_1$ , ressemblant à un échantillon  $A_n$ .

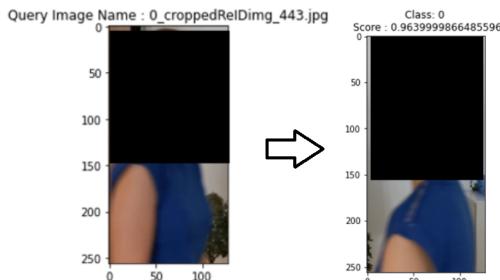


FIGURE 29 – Exemple de requêtage sur l’ensemble de test GBRLTAR.

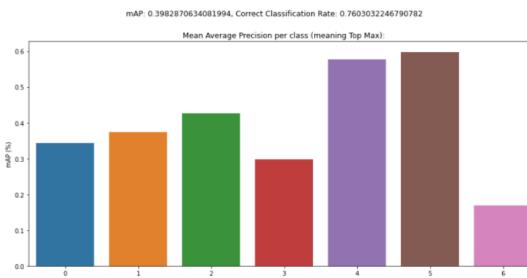


FIGURE 30 – mAP pour chaque classe de GBRLTAR, calculé sur le vecteur complet retourné, et non pas sur le top X. Si calculé sur le top X avec X = 1, alors la mAP et l’exactitude seraient identiques.

Ayant déjà expliqué l’aspect de "simplicité" de PRID2011, l’obtention d’une exactitude de 89%, contre 76% pour GBRLTAR, n’est pas surprenant, les deux ensemble étant d’un niveau de complexité opposé, nous nous attendons à obtenir des résultats à mi-chemins sur une situation réelle avec un bon placement des caméras.

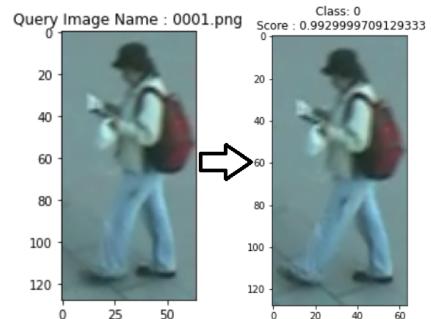


FIGURE 31 – Exemple de requêtage sur l’ensemble de test PRID2011.

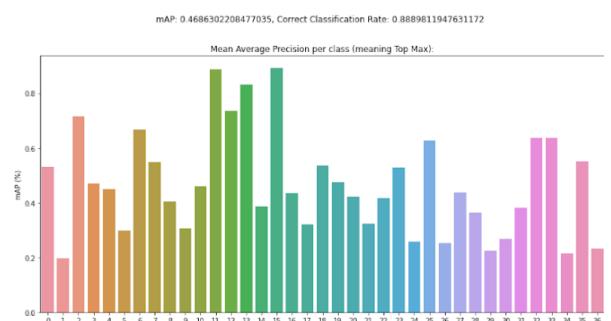


FIGURE 32 – mAP pour chaque classe de PRID2011. Calculé de la même manière que pour GBRLTAR.

L’objet de requête et l’image retournée, [figure 31](#), illustrent bien le discours précédent sur

l'abondance d'échantillons accroissant les chances d'images similaires. Dans le cas de PRID2011, cette proximité n'est cependant pas due au nombre

## 5 L'Application

Une application doit être construite pour servir de plate-forme permettant l'utilisation du modèle IA. Nous utilisons pour cela Dash, une infrastructure logicielle à la prise en main rapide grâce à son haut niveau d'abstraction. Dash est construit par dessus Plotly et React, deux infrastructures JavaScript, et est à code source ouvert (eng : Open Source).

L'application est munie d'un système d'enregistrement de compte, compte sur lequel une connexion est alors effectuée si le bon nom d'utilisateur et mot de passe sont fournis [figure 33](#). Si sur l'écran d'enregistrement un identifiant déjà connu est entrée, alors un message s'affiche expliquant que cette identifiant est déjà pris, autrement un message confirmant le succès de l'enregistrement de compte. Deux cas de figure sont pris en compte si les informations de connexions procurées par l'utilisateur ne sont pas correctes, soit la paire identifiant-mot de passe ne va pas ensemble, auquel cas un message précisant que l'un des deux est incorrect est affiché ; soit l'identifiant n'est pas enregistré, auquel cas un message informant l'utilisateur de l'absence de cet identifiant dans notre base de données est retourné. Si la connexion, c'est bien passé, alors l'utilisateur est redirigé vers la page principale de l'application.

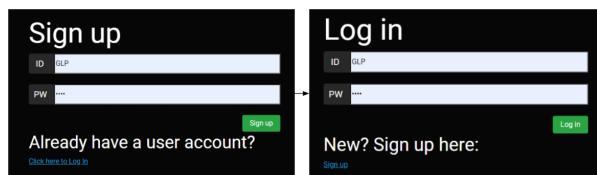


FIGURE 33 – Écran d'enregistrement de compte suivi de l'écran de connexion.

Le fonctionnement de Dash ne sera pas expliqué en détail, mais le principal est de savoir que les processus arrières (eng : Back-end) sont encapsulés dans des fonctions décorées. Les décorateurs contiennent des listes de trois types d'objets, des Entrées, des Sorties et des États, ces objets sont lié à des éléments HTML de l'application. Lorsque l'état d'un élément paramétré comme Entrée change, la fonction décorée est exécutée avec comme argu-

d'échantillons, mais est simplement dans la nature de l'ensemble de données.

ments les états des éléments passés comme Entrées et comme États. La fonction doit retourner une liste de sorties contenant autant d'éléments qu'il n'y a d'objets de type Sortie, les états des éléments lié à ces objets seront mis à jour en fonction du contenu de la liste retournée.

La [figure 34](#) illustre la génération de l'application point par point :

- 1) Le dossier contenant l'application, seul la base de données SQL ne s'y situe pas.
- 2) Le dossier contenant les codes Python de pré-traitement ou tout autre fonctions/classes lié aux fonctionnalités IA de l'application.
- 3) Le fichier principal de l'application, toutes les fonctions utilisant les décorateurs Dash s'y situe.
- 4) Tous les documents non-codes (cela n'inclut pas le fichier CSS) devant être accédé par le frontal doivent être placées dans le dossier "Assets", s'y situe donc les données vidéos à traquer.
- 5) Ce point inclut le dossier "HTML", qui ne contient qu'un seul fichier en AppLayout.py. Dash met à disposition des fonctions et classes Python permettant de simuler des balises HTML, la traduction est ensuite faite automatiquement par Dash afin de générer la page d'application dans le navigateur.
- 6) Le fichier CSS est automatiquement lu de par sa présence dans le dossier "Assets" afin de styliser la façade de l'application.
- 7) L'application est régie par un routeur, les blocs HTML définissant l'architecture de chaque page sont importés dans App.py, les pages sont alors affichées dépendant de l'URL actuel.
- 8) Le décorateur Dash permettant le déclenchement automatique de l'exécution d'une fonction.
- 9) La page principale de l'application ainsi que les liens entre les éléments HTML et les objets du décorateur.

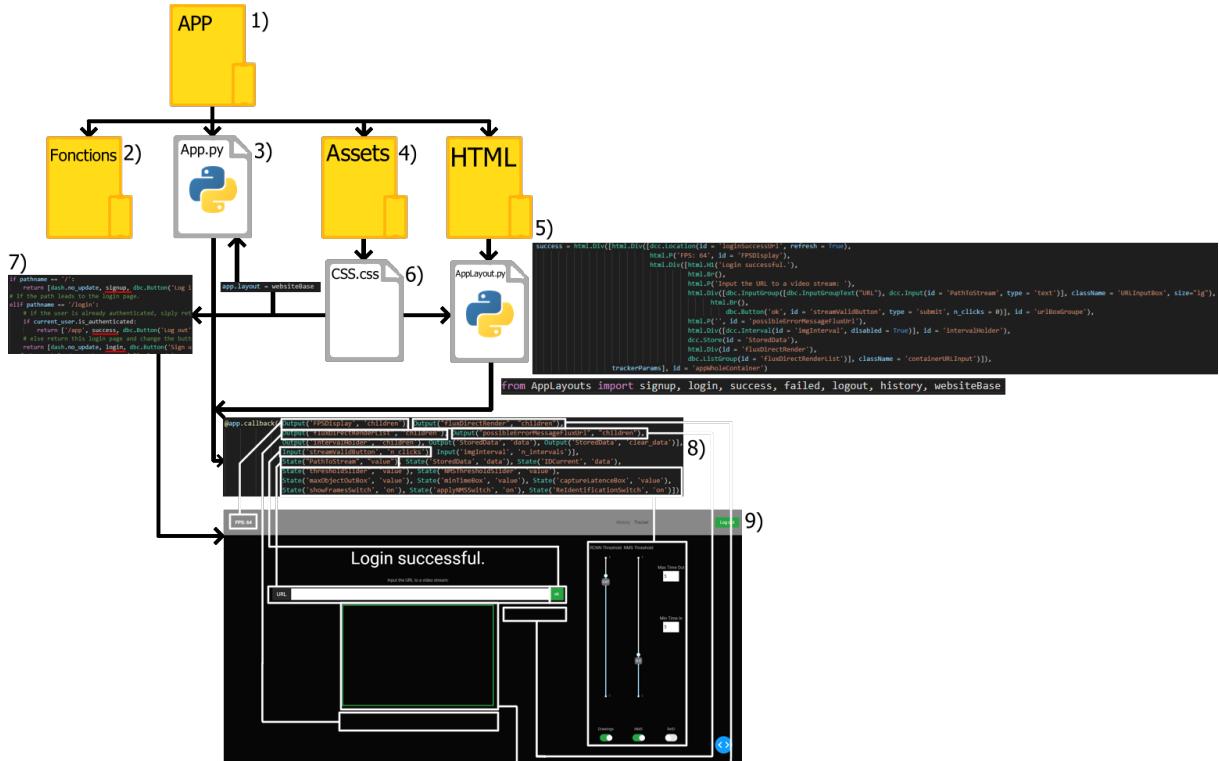


FIGURE 34 – Schéma du frontal (eng : front-end) de l’application, i.e. les interactions avec l’interface et la récupération de l’affichage en HTML-CSS, ainsi que l’arborescence des fichiers de l’application.

## 5.1 La Base de Données Relationnelle

Les besoins de la base de données sont : la gestion du système d’authentification, et l’enregistrement des résultats de suivis et des paramètres utilisées, dépendant de l’utilisateur connecté. Une base de données SQL est construite en utilisant SQLite3, les interactions entre l’application et la base de données (BDD) est gérée avec SQLAlchemy, une bibliothèque python permettant la traduction de code python en commandes SQL. La création de tables SQL via SQLAlchemy est très instinctive, les tables sont des classes et les variables de ces classes sont définis comme colonnes en instanciant des objets de la classe SQLAlchemy "Column". Une fois la classe de chaque table créée, il suffit d’utiliser la fonction SQLAlchemy "create\_all" et une commande SQL sera générée afin de construire les tables et leurs liens dans la BDD. Les lignes dans ces tables ne sont ensuite que des instances de la classe, dans Python, les valeurs de chaque colonne étant passées comme argument.

La base de données est relationnelle, comme illustrée par la figure 35, ce qui signifie que les tables sont liées entre elles via plusieurs types

de connexions. Notre BDD est composée de trois tables, "Users", "Tracks" et "Parameters" :

- 1) La table "Users" contient les informations relatives à l’utilisateur. C’est à cette table qu’est lié le système d’authentification, elle contient donc le nom d’utilisateur et le mot de passe.
- 2) La table "Tracks" contient les informations liées aux suivis. Chaque fois que le modèle est utilisé, le lien vers la vidéo ainsi l’heure et date du début du suivi sont enregistrés, lorsque le suivi s’arrête, un horodatage de cette arrêt et les résultats du suivi (le nombre des gens détecté) sont récupéré, le tout est inséré dans une ligne de la table.
- 3) La table "Parameters" contient la liste de paramètres passés au modèle, le Traqueur, pour un suivi.

Les connecteurs logiques de la figure 35 sont définis de sorte à ce que : un utilisateur puisse avoir *plusieurs* suivis, et un suivi soit lié à une liste de paramètres. De ce fait, une table "historique" a pu être intégrée, permettant à un utilisateur de voir les suivis qu’y lui sont liés, et les paramètres utilisés pour ces suivis.

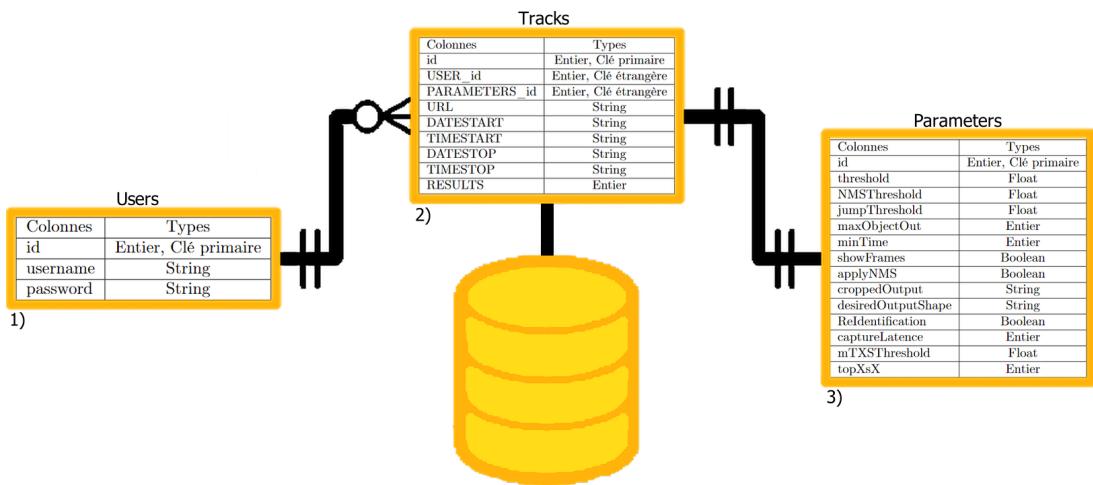


FIGURE 35 – Schéma de la BDD SQL.

Le système d'authentification est géré par la bibliothèque flask\_login, le raccordement de la BDD et de flask\_login à l'application sont expliqués [figure 36](#) :

- 1) LoginManager, une classe de flask\_login, est instancié, le lien avec l'application est simplement fait en passant la variable contenant l'élément "server" de l'instance d'application Dash.
- 2) L'objet "server" possède une fonction permettant sa configuration, c'est en l'utilisant que la BDD, dont le fichier est d'extension "sqlite", peut être lié à l'application.
- 3) Lors de la création de compte, le mot de passe est hashé, c'est-à-dire qu'il est encodé d'une manière indécryptable, c'est une mesure de sécurité bloquant la visibilité du mot de passe à un quelconque intrus dans la base de données. Un exemple d'insertion de ligne dans la table "Users" est également présenté.
- 4) Afin de pouvoir effectuer ces insertions, nous devons nous connecter au fichier .sqlite et créer un moteur en utilisant la bibliothèque sqlite3.
- 5) La table "User", qui reçoit la ligne insérée au point n°3, est modifiée afin d'inclure la classe "UserMixin" dans ses relations de parenté. "UserMixin" permet d'accéder aux fonctions de flask\_login lié à l'utilisateur connecté.
- 6) Lors de l'identification, la base de données est requêté afin de savoir si l'identifiant existe, puis si le mot de passe entré, hashé, est identique à celui attendu, si tout est correct, la fonction "login\_user" de flask\_login est utilisé pour connecter l'utilisateur.
- 7) Si un utilisateur est connecté lorsque l'URL

pointe vers la page de déconnexion, il est déconnecté et redirigé vers la page de connexion.

- 8) Il est possible de modifier l'URL afin d'atteindre certaines parties de l'application, le routeur vérifie donc systématiquement la présence d'un utilisateur connecté pour permettre l'accès aux pages autre que d'inscription et de connexion. Si l'utilisateur n'est pas connecté, un écran informant d'une "erreur d'authentification" est retourné.

## 5.2 Monitorage et Tests Unitaires

Le monitorage de l'application est fait par plusieurs biais : Dash, qui permet de contrôler le nombre d'erreurs, l'état du serveur (allumé, éteint), et l'architecture de l'application, avec les temps de réponse et les liens entre chaque "callback" Dash (les objets passés aux décorateurs des fonctions), et notifie par un pop-up lors d'un problème ; un compteur d'images par secondes, affiché en haut à gauche de l'écran ; et la journalisation des erreurs dans un fichier spécifique, une fonction est fournie afin d'analyser le contenu de ce fichier. Les tests unitaires, contrôlant le bon fonctionnement de l'application, sont gérés par plusieurs déclarations "try : except :" et "assert x", abstractions Python permettant de spécifier les réactions à une erreur et de vérifier l'état d'un objet. Ainsi, les erreurs sont cataloguées et ne coupent pas l'exécution en cours. Plusieurs messages guidant l'utilisateur à travers l'application sont également affichés selon la situation : mauvais paramètres d'identification, statut des processus, indicateurs d'un mauvais remplissage de champ, etc...

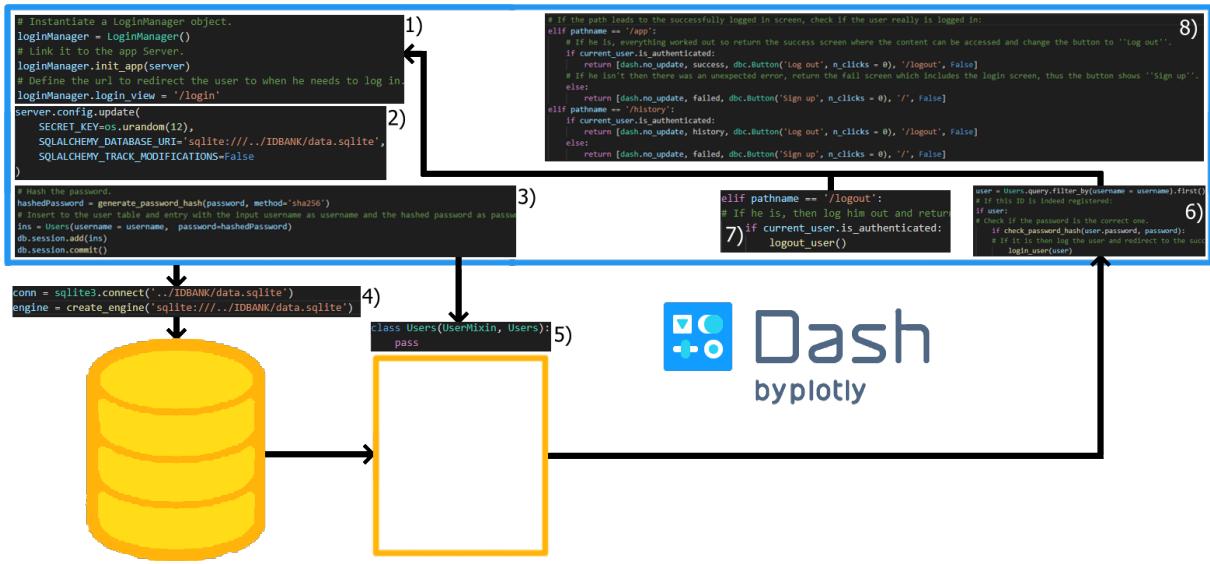


FIGURE 36 – Schéma du système d'authentification et son lien avec la base de données SQL.

## 6 Résultats du Comptage Assisté de Ré-Identification

Plus que les tests de ré-identifications effectuées sous-sous-section 4.2.3, les tests de comptage en utilisant l'application maintenant terminé, composée du Traqueur équipé du Classifieur, doivent être effectuées. L'ensemble vidéo de test sera bien entendu GBRALTAR, les comparaisons se feront comme suit : comptage manuel sans ré-identification, c'est-à-dire en comptant chaque passage devant la caméra comme une personne unique; comptage automatique sans ré-identification, où les résultats du Traqueur seul seront confronté au comptage manuel ; et enfin comptage automatique avec ré-identification, où nous comparerons le nombre d'identités que possède GBRALTAR (7) au nombre retourné par le Traqueur-Ré-identifieur.

Les valeurs par défaut des paramètres du Traqueur sont : valeur de seuil du RCNN pour la conservation des boîtes limites "threshold" = 0.85, valeur de seuil de la NMS "NMSThreshold" = 0.3, distance maximum qu'un centroïde peut parcourir en une image en pixel "jumpThreshold" = 200, nombre d'images qu'une entité peut passer hors du cadre avant de ne plus être suivi "maxObjectOut" = 5, nombre d'images minimum de présence pour qu'une identité soit compté "minTime" = 2, booléen de l'application ou non de la NMS "applyNMS" = True, booléen de la modification des images pour indiquer les boîtes limites et centroïdes "showFrames" = True, chemin vers un répertoire où le contenu des boîtes limites peuvent être sauve-

gardées "croppedOutput" = None, tuple hauteur, largeur de la taille désiré pour les boîtes limites "desiredOutputShape" = (128, 64), booléen de l'application ou non de la ré-identification "ReIdentification" = True, rythme de ré-identification et conservation du contenu des boîtes limites "captureLatence" = 5, valeur de X pour le vecteur top X "topXsX" = 1, valeur de seuil pour l'assignation à une identité par ré-identification "mTXSThreshold" = 0.88.

Sans ré-identification et avec ces paramètres, les résultats du Traqueur sont : sur GBRALTAR caméra "highAngle" de 50 personnes comptées contre 52 manuellement, sur GBRALTAR caméra "Side" de 48 personnes comptées contre 48 manuellement, sur GBRALTAR caméra "highAngle" de 49 personnes comptées contre 49 manuellement. Quelques erreurs cataloguées dans un fichier Excel, mais ces erreurs semblent se balancer entre elles.

En laissant les paramètres de suivi majoritairement par défaut : avec "threshold" à 0.87 et "NMSThreshold" à 0.3, avec "MaxTimeOut" et "MinTimeIn" à 5, "captureLatence" à 1 (donc chaque image). Les paramètres de ré-identification comme suit : avec "mTXSThreshold" à 0.973 (Obtenue en récupérant une valeur entre la médiane des mTXS résultant en une classification correcte durant les tests de ré-identification, 0.98, et celles des classifi-

cations incorrectes, 0.971. Les écarts-types sont de 0.009 dans les deux cas. Sur différents ensembles, ce nombre reste plus ou moins stable, changeant selon le niveau de "difficulté" des données. PRID est à 0.991 et 0.987 pour correct et incorrect, avec des écarts-types de 0.0042, dans ce cas 0.99 aurais pu être choisie.), "mTXSThreshold" qui s'auto-incrémente selon l'[équation 8](#); et avec un score de similitude généré par  $1 - \text{distance\_euclidienne}/127.5$ , la valeur de 127.5 étant obtenue par l'utilisation de la marge de la perte par triplet utilisé lors de l'entraînement du modèle.

Avec ré-identification et avec les paramètres ci-dessus, les résultats du Traqueur sont sur GBRALTAR caméra "highAngle", de 12 personnes compté au lieu de 7. Des contraintes de temps et de ressources ne nous ont permis de ne contrôler que cette caméra. Là où certaines identités ont été compté plusieurs fois, une grande partie des erreurs sont causées par des échanges d'identifiants, ce phénomène est causé par le croisement de plusieurs identités ou le non-suivi temporaire d'une identité alors qu'une autre se situait proche, résultant en un échange des identifiants et donc des suivis. La [table 3](#) décrit précisément les résultats du suivi avec ré-identification.

N°d'ID	contenu en identités réel * nombre de ré-identification de cette identité réel
1	A*4
2	B*3 + F*1 (Possible échange d'identifiants.)
3	C*1
5	D-B*1 (Une seule boîte limite à capturer deux identités.)
6	D*5 + D(accroupie)*1 + A*1 + G*2
14	B*2
16	D*1
25	D*1 + A ? (Possible échange d'identifiants.)
27	E*3 + A ? (Possible échange d'identifiants.) + F*5 + B-D*1
46	F*1
48	D(accroupie)*2
63	C-F*1 (Une seule boîte limite à capturer deux identités.)

TABLE 3 – Identités réelles de GBRALTAR, de A à G, représentées dans les vecteurs d'encodages de chaque identité comptée.

## 7 Communication et Méthodologie de Gestion durant Le Déroulement du Projet

Le projet s'est étendu sur une durée d'un an, suivant une méthodologie agile (motivé par l'environnement Groupama qui, comme beaucoup d'entreprises, favorise la sélection de certains éléments des méthodes Scrum et du système Kanban afin de mieux convenir aux besoins.) cette période fut découpé en quatre sprints de trois mois, chaque sprint est visible dans le tableau Trello de la [figure 37](#).

Quelques dates en rouges sont visibles sur le Trello, elles représentent des tâches entreprises, mais sans aboutissement avant la date limite, en voici l'explication : le modèle initial, celui devant originellement être développé jusqu'à la fin du second sprint, est le VNAE, notre inaptitude à l'implémenter et à reproduire les résultats attendus est brièvement discuté [sous-section 2.3](#). L'objectif suite à l'échec était de persévérer dans l'idée d'utiliser le VNAE durant le premier tiers du prochain sprint,

s'il venait à être fonctionnel, alors la fin du sprint sera dédié à son ajout au sein du Traqueur et à la programmation d'une partie, voir de la totalité des processus arrières de l'application. Autrement, 7 semaines seront allouées au développement d'un nouveau modèle, le Classifieur, qui est le sujet de la [sous-section 4.2](#), laissant une semaine pour son insertion dans le Traqueur. Le VNAE n'ayant jamais abouti, le deuxième chemin fut emprunté, rajoutant une carte "pivot de modèle" et causant le développement de l'application d'être retardé au prochain sprint.

Les cérémonies de notre méthodologie incluent des réunions d'une heure chaque lundi et jeudi. Un rapport hebdomadaire est également rédigé sur la plate-forme Jira, résumant les actions réalisées cette semaine, les possibles problématiques et les étapes à suivre. Finalement, un fichier document est créé

et maintenu à jour par un rapport de chaque action entreprise ainsi que les résultats obtenus, allant des résumés des recherches aux bilans de tests.

Les canaux d'échanges avec les parties prenantes

sont ceux de l'environnement de travail Groupama, ainsi Teams est préféré pour des échanges en visioconférences, Outlook est utilisé pour planifier les rendez-vous, et les journaux de projets sur Jira sont accessibles à tous.



FIGURE 37 – Tableau Trello. En bleue : recherches et mises à niveau, cela inclut des veilles techniques sur les méthodes disponibles, ainsi qu'une mise à niveau nécessaire en mathématiques. En vert : traitement des données. En violet : développement des modèles, Traqueur et IA. En jaune : développement de l'application et de la base de données SQL. Sans couleurs/noir : rédaction des livrables.

## 8 Pistes d'Améliorations

Les performances atteintes par notre modèle sont un grand pas en avant par rapport à un comptage sans ré-identification, cependant, nous sommes loin d'atteindre les performances requises pour un déploiement en agence. Bien entendu l'objectif réel était plus de réaliser une preuve de concept qu'un modèle atteignant l'état de l'art, le tout avec une approche de recherche et développement. Plusieurs points d'améliorations ont tout de même été pensés, et parfois testés. Ce chapitre a pour objectif de les présenter comme de potentielles pistes à creuser dans le but d'obtenir un meilleur modèle.

Que ce soit pour récupérer les boîtes limites, ou appliquée à ces dernières lors de la ré-identification, la détection du premier plan ou sous-traction d'arrière-plan (eng : background subtraction) consiste en la séparation du fond, généralement immobile, et du premier plan (e.g : une personne traversant le champ de la caméra). De nombreuses méthodes existent pour cela, statistiques [2, 14, 12] comme par apprentissage machine, les deux furent testé. Les méthodes statistique, dont les implémentations openCV des travaux de Stauffer

et Grimson [29], ont été appliquée sur l'image dans sa totalité, malheureusement ces méthodes et le post-traitement qu'elle requiert prennent beaucoup trop de temps à s'opérer, et ce, sans améliorations tangibles dans les performances du Traqueur. Une méthode d'intelligence artificielle par Zhanghan et al. [16] fut testé sur le contenu de boîtes limites, le rendu était la suppression constante de l'intégralité de l'image. La piste fut abandonnée, mais nous sommes persuadé que la suppression de l'arrière-plan faciliterait la ré-identification, en ne conservant que les pixels et attributs de l'identité.

Dans notre cas, la ré-identification commence sans aucun encodage d'enregistrés et les tests montre des fluctuations plus importantes plus le nombre d'encodage est bas. Pour remédier à ce problème, deux méthodes ont été pensées : pré-fournir des encodages venant des ensembles de données de test ou d'entraînement, et/ou modifier dynamiquement la valeur de "self.mTXSThreshold". La dernière méthode a été testée selon l'équation 8, et est fonctionnelle pour minimiser le problème, il ne le règle cependant pas à la source.

Avec "encodePourNorm" contenant un entier paramétrable définissant à quel nombre d'encodage le seuil == mTXSThreshold ; "debutAug" contenant un entier paramétrable définissant à quel nombre d'encodage le seuil commence à augmenter ; ".seuilMin" étant la valeur minimum pour le seuil ; et la variable "S" remplaçant mTXSThreshold à la ligne 11 de l'[algorithme 9](#) :

$$\begin{aligned} pas &= mTXSThreshold - seuilMin \\ x &= \min(\text{len}(codeGlob), encodePourNorm) \\ S &= seuilMin + pas * (\max(x - debutAug, 0) / 100) \end{aligned} \quad (8)$$

Il est évident que l'application est lente, et certaines "têtes de bouteilles" sont identifiables. Par exemple, le simple fait d'utiliser Dash nous limite à un traitement de 4 images par secondes (ips). Cette limitation est causée par l'affichage de la vidéo image par image, Dash doit donc récupérer l'image et l'envoyer à un serveur distant, cette simple action est exécuté en 250ms ce qui est très lent. Le plus évident est sans doute la limite en puissance de calcul, l'application tournant de la seul manière alors disponibles, c'est-à-dire sur un PC portable équipé d'un processeur Intel i5 de huitième génération, de ce fait, même un algorithme tel que YOLOV3, pourtant prévu pour du traitement en temps réel à plus de 40ips, devient un fort élément de ralentissement. Une vidéo de 5 minutes, lorsque traité par YOLOV3 sur notre machine, en devient une de 20 minutes, de 4ips nous passons donc à 1ips. Le reste des processus ne sont pas indépendamment problématiques, et même si leur nombre est conséquent leur influence n'est pas comparables à celle des autres points mentionnés, cependant l'optimisa-

tion est un métier en soit et il est plus que possible que la fluidité soit à portée de main en modifiant le code existant.

La plus grande source de difficultés reste le phénomène d'échange d'identifiants. Trois méthodes sont souvent utilisées pour améliorer les aptitudes des modèles de suivi d'objets, et leur implémentassent combiné a de fortes chances de régler notre problème. L'algorithme hongrois, bien qu'une charge de calcul supplémentaire, permettrait d'optimiser l'assignation des centroïdes, il est souvent appliqué en concert avec le filtre de Kalman, que nous avons échoué à implémenter, et qui permet de prédire la position n+1 d'un objet à partir de sa position n, de sa vitesse et de sa direction. Enfin une lame à double tranchant est la Ré-identification continue, qui consiste en la ré-identification systématique de chaque individu, chaque image. Bien que l'avantage soit la suppression totale du problème classique d'échange d'identifiants, le contre coup est double : non seulement est-ce que le prix computationnel est élevé, mais le plus de ré-identification, le plus de chance qu'une erreur d'attribution occurrent, ne faisant donc que modifier la source du problème. Nonobstant les craintes, ces pistes restent une base de recherche solide et pourrait permettre l'obtention de résultats s'approchant de la perfection.

Enfin, le nombre d'ensembles de données d'une qualité équivalente à celle de PIROPO Database est limité, la production et la mise à libre disposition de vidéos de haute résolution, possédant des scènes capturées sous plusieurs angles et distances, permettrais un meilleur entraînement des modèles et de meilleur résultats, procurés par une approche centrée sur les données.

## 9 Conclusion

Dans ce rapport, nous présentons un algorithme capable d'effectuer un comptage de passants uniques, et dont les résultats sont, suite à l'implémentassions d'un modèle d'intelligence artificielle de ré-identification, strictement meilleurs qu'un traqueur seul. Le modèle est construit de manière à ne pas favoriser des parties du corps qui peuvent être perçus comme des données privée comme le visage. Une application possédant un système d'authentification et une base de donnée relationnelle fut produite, elle est fonctionnelle et peut être utilisée sur n'importe quel ensemble vidéo mis sous forme de dossier d'images. Notre modèle offre également de

meilleurs résultats que ceux obtenus précédemment par un organisme externe. Un ensemble de données composé de trois vidéos, chacune capturant une scène d'un point de vue différent fut crée pour l'occasion, il n'est pas récupérable et son utilisation dans le présent rapport doit être censuré afin de respecter la vie privée des participants. Finalement, les limitations de notre modèle furent définies, et des pistes d'améliorations ont été présentées dans l'espoir que de nouveaux travaux puissent être commencés avec comme objectif l'obtention d'un produit apte au déploiement.

## Références

- [1] P. ANDAL et K. M. VINOTHIGA. « Review of Canny Edge Detection Algorithm and HOG Feature Extraction in Facial Expression Recognition ». In : 2018.
- [2] Thanarat CHALIDABHONGSE, David HARWOOD et Larry DAVIS. « A robust background subtraction and shadow detection ». In : *Proceedings on Asian Conference on Computer Vision* (jan. 2000).
- [3] Yehansen CHEN et al. « Neural Feature Search for RGB-Infrared Person Re-Identification ». In : *CoRR* abs/2104.02366 (2021). arXiv : [2104.02366](https://arxiv.org/abs/2104.02366). URL : <https://arxiv.org/abs/2104.02366>.
- [4] N. DALAL et B. TRIGGS. « Histograms of oriented gradients for human detection ». In : *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. T. 1. 2005, 886-893 vol. 1. DOI : [10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177).
- [5] Carlos R. DEL-BLANCO et al. « Robust people indoor localization with omnidirectional cameras using a Grid of Spatial-Aware Classifiers ». In : *Signal Process. Image Commun.* 93 (2021), p. 116135.
- [6] Songchenchen GONG et El-Bay BOURENNANE. « A method based on texture feature and edge detection for people counting in a crowded area ». In : *Digital Image and Signal Processing (DISP'19)*. Oxford, United Kingdom, avr. 2019. URL : <https://hal.archives-ouvertes.fr/hal-02275646>.
- [7] Kaiming HE et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv : [1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385).
- [8] Kaiming HE et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv : [1603.05027 \[cs.CV\]](https://arxiv.org/abs/1603.05027).
- [9] Irina HIGGINS et al. « beta-VAE : Learning Basic Visual Concepts with a Constrained Variational Framework ». In : *ICLR*. 2017.
- [10] Martin HIRZER et al. « Person Re-Identification by Descriptive and Discriminative Classification ». In : *Proc. Scandinavian Conference on Image Analysis (SCIA)*. 2011.
- [11] Di HUANG et al. « Local Binary Patterns and Its Application to Facial Image Analysis : A Survey ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41.6 (2011), p. 765-781. DOI : [10.1109/TSMCC.2011.2118750](https://doi.org/10.1109/TSMCC.2011.2118750).
- [12] Anand JALAL et Vrijendra SINGH. « A Robust Background Subtraction Approach Based on Daubechies Complex Wavelet Transform ». In : t. 191. Juill. 2011, p. 516-524. DOI : [10.1007/978-3-642-22714-1\\_53](https://doi.org/10.1007/978-3-642-22714-1_53).
- [13] Sun JIANDE, Wang YUFEI et Li JING. « Gait Recognition ». In : (juin 2017). DOI : [10.5772/68119](https://doi.org/10.5772/68119).
- [14] Claudio JUNG. « Efficient Background Subtraction and Shadow Removal for Monochromatic Video Sequences ». In : *Multimedia, IEEE Transactions on* 11 (mai 2009), p. 571-577. DOI : [10.1109/TMM.2009.2012924](https://doi.org/10.1109/TMM.2009.2012924).
- [15] Mahdi KALAYEH et al. « Human Semantic Parsing for Person Re-identification ». In : juin 2018, p. 1062-1071. DOI : [10.1109/CVPR.2018.00117](https://doi.org/10.1109/CVPR.2018.00117).
- [16] Zhanghan KE et al. « MODNet : Real-Time Trimap-Free Portrait Matting via Objective Decomposition ». In : *AAAI*. 2022.
- [17] Mate KRIŠTO et Marina IVAŠIĆ-KOS. « An overview of thermal face recognition methods ». In : mai 2018, p. 1098-1103. DOI : [10.23919/MIPRO.2018.8400200](https://doi.org/10.23919/MIPRO.2018.8400200).
- [18] Ji LEE et al. « Robust Pedestrian Detection by Combining Visible and Thermal Infrared Cameras ». In : *Sensors (Basel, Switzerland)* 15 (mai 2015), p. 10580-615. DOI : [10.3390/s150510580](https://doi.org/10.3390/s150510580).
- [19] Diangang LI et al. « Infrared-Visible Cross-Modal Person Re-Identification with an X Modality ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (avr. 2020), p. 4610-4617. DOI : [10.1609/aaai.v34i04.5891](https://ojs.aaai.org/index.php/AAAI/article/view/5891). URL : <https://ojs.aaai.org/index.php/AAAI/article/view/5891>.
- [20] Shuang LI et al. « Person Search with Natural Language Description ». In : (fév. 2017).
- [21] Hao LUO et al. « A Strong Baseline and Batch Normalization Neck for Deep Person Re-identification ». In : *CoRR* abs/1906.08332 (2019). arXiv : [1906.08332](https://arxiv.org/abs/1906.08332). URL : [http://arxiv.org/abs/1906.08332](https://arxiv.org/abs/1906.08332).

- [22] vijay M. *How to calculate gradients in ResNet architecture ?* URL : <https://stackoverflow.com/questions/44512126/how-to-calculate-gradients-in-resnet-architecture/448613210>.
- [23] Dat NGUYEN et Kang PARK. « Body-Based Gender Recognition Using Images from Visible and Thermal Cameras ». In : *Sensors* 16 (jan. 2016), p. 156. DOI : [10.3390/s16020156](https://doi.org/10.3390/s16020156).
- [24] Tien Dat NGUYEN et al. « Person Recognition System Based on a Combination of Body Images from Visible Light and Thermal Cameras ». In : *Sensors (Basel, Switzerland)* 17 (2017).
- [25] Kai NIU et al. *Improving Description-based Person Re-identification by Multi-granularity Image-text Alignments*. Juin 2019.
- [26] Hugh PERKINS. *Gradient backpropagation through ResNet skip connections*. URL : <https://stats.stackexchange.com/questions/268820/gradient-backpropagation-through-resnet-skip-connections>.
- [27] Jiawei REN et al. « HAVANA : Hierarchical and Variation-Normalized Autoencoder for Person Re-identification ». In : *CoRR* abs/2101.02568 (2021). arXiv : [2101.02568](https://arxiv.org/abs/2101.02568). URL : <https://arxiv.org/abs/2101.02568>.
- [28] Murray R. SPIEGEL. *Gradient Operator Distributes over Addition*. URL : [https://proofwiki.org/wiki/Gradient\\_Operator\\_Distributes\\_over\\_Addition](https://proofwiki.org/wiki/Gradient_Operator_Distributes_over_Addition).
- [29] Chris STAUFFER et W. GRIMSON. « Adaptive background mixture models for real-time tracking ». In : *Proceedings of IEEE Conf. Computer Vision Patt. Recog*, vol. 2 2 (jan. 2007).
- [30] Gilbert STRANG. « Wavelet transforms versus Fourier transforms ». In : *Bulletin of the American Mathematical Society* 28 (1993), p. 288-305.
- [31] Christian SZEGEDY et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv : [1512.00567 \[cs.CV\]](https://arxiv.org/abs/1512.00567).
- [32] THERMOGRAPHIE. *La différence caméra infrarouge et thermique*. URL : <https://www.thermographies.com/thermographie-difference-camera-infrarouge.html>.
- [33] Nicolai WOJKE et Alex BEWLEY. « Deep Cosine Metric Learning for Person Re-Identification ». In : *CoRR* abs/1812.00442 (2018). arXiv : [1812.00442](https://arxiv.org/abs/1812.00442). URL : [http://arxiv.org/abs/1812.00442](https://arxiv.org/abs/1812.00442).
- [34] Yongkang WONG et al. « Patch-based Probabilistic Image Quality Assessment for Face Selection and Improved Video-based Face Recognition ». In : *IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, juin 2011, p. 81-88.
- [35] Ancong WU et al. « RGB-Infrared Cross-Modality Person Re-Identification ». In : *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.
- [36] Ying ZHANG et Shutao LI. « Gabor-LBP Based Region Covariance Descriptor for Person Re-identification ». In : *Image and Graphics (ICIG), 2011 Sixth International Conference on* (août 2011). DOI : [10.1109/ICIG.2011.40](https://doi.org/10.1109/ICIG.2011.40).
- [37] Liang ZHENG et al. « Scalable Person Re-identification : A Benchmark ». In : *Computer Vision, IEEE International Conference on*. 2015.
- [38] Zhedong ZHENG et al. « Joint Discriminative and Generative Learning for Person Re-identification ». In : *CoRR* abs/1904.07223 (2019). arXiv : [1904.07223](https://arxiv.org/abs/1904.07223). URL : [http://arxiv.org/abs/1904.07223](https://arxiv.org/abs/1904.07223).
- [39] Zhun ZHONG et al. « Random Erasing Data Augmentation ». In : *CoRR* abs/1708.04896 (2017). arXiv : [1708.04896](https://arxiv.org/abs/1708.04896). URL : [http://arxiv.org/abs/1708.04896](https://arxiv.org/abs/1708.04896).