Mémoire de Licence APSIO

IOT et la reconnaissance d'image : mini box pour compter les vélos

Tandol Noémie et Véronési Kévin

A l'attention de M. Yahn Formanczak

Sommaire

Remerciements

Nous exprimons notre profonde gratitude à Monsieur Yahn Formanczak, qui nous a offert l'opportunité de travailler sur ce sujet passionnant et stimulant. Il a été un guide précieux et un soutien constant tout au long de ce projet, nous apportant ses conseils avisés, ses retours constructifs et son encouragement. Nous avons beaucoup appris de son expertise et de son expérience, et nous lui sommes reconnaissants pour sa confiance et sa bienveillance.

Résumé

Dans ce mémoire de licence APSIO, nous avons travaillé sur l'utilisation de l'IOT et de la reconnaissance d'image pour créer une mini box pour compter les vélos. Nous avons développé un programme pour une borne de compteur cycliste miniature et autonome, afin d'être déployé sur un grand nombre de ces appareils à moindre coût. L'objectif étant de recueillir différentes informations pour chaque cycliste, notamment l'heure de passage, le sens de circulation, la charge du vélo ou encore le port d'équipements de protection.

Notre approche est basée sur des cycles de développement itératifs suivi de présentations régulières à notre tuteur. Ce programme utilise plusieurs composants et modules pour implémenter un système de détection d'objets et de visualisation efficace. Nous avons choisi des outils spécifiques tels que PyTorch et Python en raison de leur adaptabilité à nos besoins et de leur facilité d'utilisation. Notre méthodologie d'évaluation rigoureuse a permis de mesurer les performances du système de détection d'objets sur des jeux de données spécifiques. Les résultats de ces tests ont montré que ce système est capable de détecter avec précision et rapidité les vélos dans différentes sources.

Abstract

This diploma dissertation, entitled "IOT and image recognition: Mini Box for Counting Bicycles", was written by Noémie Tandol and Kévin Véronési as part of the APSIO bachelor program. The aim of the project is to develop a miniature, autonomous bicycle counting box that allows a large number of these devices to be deployed at low cost.

The bicycle counters we propose are an open source project designed to meet the needs of local authorities who want to assess the impact of their infrastructure developments on bicycle traffic. They are also intended for cycling associations such as "2 pieds 2 roues" and "AF3V". These counters could also have a commercial application on long-distance routes such as EuroVelo, by encouraging shopkeepers to promote cycle tourism.

Our aim is to collect a range of information for each cyclist, including the time of passage, the direction of travel, the load on the bike and whether protective equipment is worn. Several steps will be necessary to achieve this objective.

Firstly, a comparative study of open source video recognition software will be carried out to find the solution best suited to our problem. Next, the feasibility of the project needs to be verified by building a prototype on Raspberry Pi to validate its operation. Finally, an in-depth analysis of the legality and ethics of the data collected will be carried out, taking into account the different uses (research, voluntary, community, commercial, etc.).

This research will enable us to better understand the impact of cycling, optimize existing infrastructure and further promote cycling. In addition, by adopting an open source approach, we hope to encourage collaboration and contributions from the community to continually improve our solution and encourage innovation in the field of cycle mobility.

Introduction

Contexte et motivation

Le vélo est plus qu'un simple moyen de transport, c'est un mode de vie. Il offre de nombreux avantages, tant pour la santé que pour l'environnement. Permettant aussi de découvrir des paysages et des cultures, de partager des expériences et des valeurs. Mais comment mesurer l'ampleur et l'évolution du trafic cyclable ? De quelle manière adapter les infrastructures et services aux besoins des cyclistes ? Pouvons-nous revaloriser et encourager la pratique du vélo ?

Notre projet vise à répondre à ces questions en développant une solution innovante : une borne de compteur cycliste miniature et autonome, qui peut être déployée facilement et à moindre coût sur un large territoire. Étant un projet open source il s'inscrit dans une démarche collaborative et participative. Il s'adresse aux collectivités locales, qui souhaitent évaluer l'impact de leurs politiques cyclables. Il est également destiné aux associations comme "2 pieds 2 roues" ou "AF3V", qui militent pour la promotion du vélo. Enfin, il s'adresse aux acteurs du tourisme, qui veulent profiter du potentiel économique du vélo.

Objectifs du projet

Notre objectif est de recueillir différentes informations pour chaque cycliste, notamment l'heure de passage, le sens de circulation, la charge du vélo, et le port d'équipements de protection. Pour atteindre cet objectif, plusieurs étapes seront nécessaires.

D'abord, une étude comparative des logiciels open source de reconnaissance vidéo sera réalisée afin de trouver la solution la plus adaptée à notre problématique. Ensuite, nous devrons vérifier la faisabilité du projet en créant un prototype sur Raspberry Pi, permettant ainsi de valider son fonctionnement. Enfin, une analyse approfondie de la légalité et de l'éthique des données collectées sera effectuée, en prenant en compte les différents cadres d'exploitation (recherche, associatif, municipalité, commercial, etc.).

Cette recherche nous permettra de mieux comprendre l'impact du trafic cycliste, d'optimiser les infrastructures existantes et de promouvoir davantage l'utilisation du vélo. De plus, en adoptant une approche open source, nous espérons favoriser la collaboration et la contribution de la communauté, afin d'améliorer continuellement notre solution et d'encourager l'innovation dans le domaine de la mobilité cycliste.

Méthodologie de développement

Pour mener à bien notre projet, nous avons adopté une approche basée sur des cycles de développement itératifs, avec des présentations régulières à notre tuteur. Cette méthode nous a permis d'obtenir des retours fréquents et d'ajuster notre travail en conséquence, en nous assurant que nous progressions dans la bonne direction et que nous répondions aux attentes du projet.

Notre processus de développement comprend plusieurs étapes clés :

1. Analyse des besoins
2. Conception initiale
3. Développement itératif
4. Présentations et retours
5. Tests et validation
6. Ajustements et améliorations

Grâce à l'adoption de cette méthodologie de développement itératif et à la tenue régulière de réunions avec notre tuteur, nous avons pu avancer efficacement et garantir que notre solution répondait parfaitement aux objectifs du projet. Avec cette approche, nous avons pu surmonter efficacement les difficultés rencontrées, tout en assurant une communication claire et un travail en équipe soutenu durant le développement du projet.

1. Etat de l'art

1.1. Vue d'ensemble des systèmes de détections d'objets

La détection d'objets est une tâche cruciale dans le domaine de la vision par ordinateur. Plusieurs approches et techniques ont été développées pour relever ce défi, notamment celles basées sur l'apprentissage profond (deep learning) et les algorithmes de vision par ordinateur classiques.

Dans le domaine de l'apprentissage profond, les réseaux de neurones convolutifs (CNN) sont largement utilisés pour la détection d'objets. Des architectures populaires telles que YOLO (You Only Look Once) et SSD (Single Shot MultiBox Detector) ont été proposées. Ces méthodes utilisent des réseaux convolutifs pour extraire des caractéristiques à partir des images, puis utilisent ces informations pour prédire les boîtes englobantes des objets détectés et leurs classes. Une classe dans le domaine de la détection d'objet correspond au type d'objet détecté, elle est souvent représentée par un numéro, qui devra ensuite être traduit par son label assigné.

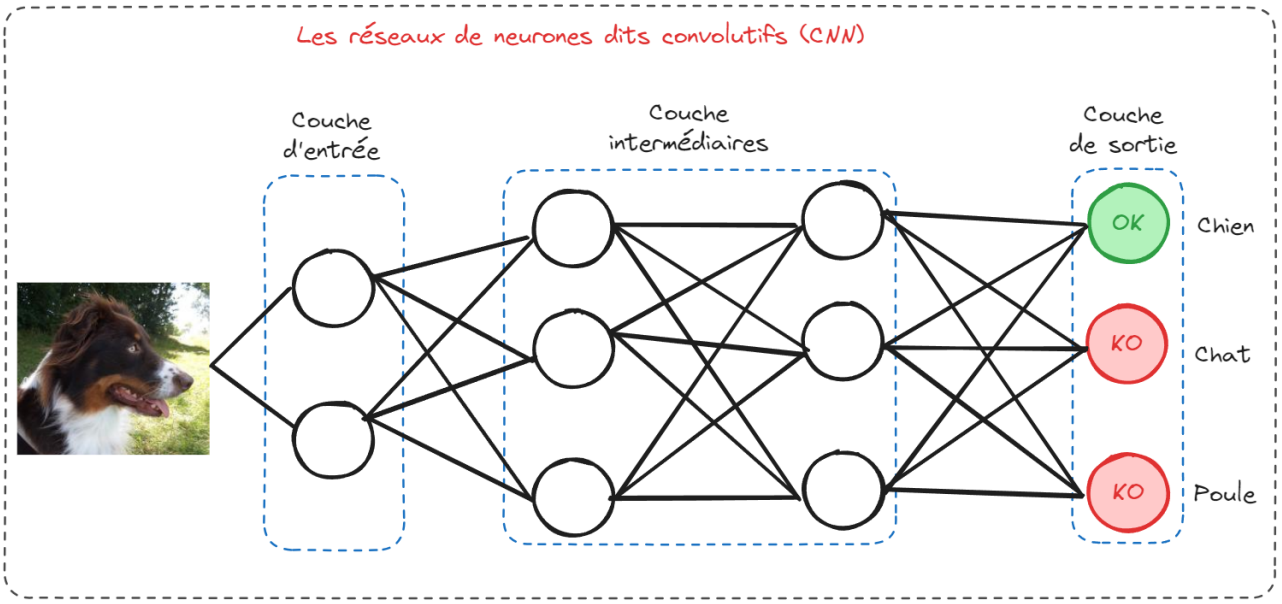

Figure 1 : Les réseaux de neurones convulsifs

D'autres approches basées sur l'apprentissage profond incluent l'utilisation de réseaux neuronaux récurrents (RNN) pour la détection séquentielle d'objets, ainsi que des méthodes de transfert de connaissances pour tirer parti des modèles pré-entraînés sur de grands ensembles de données.

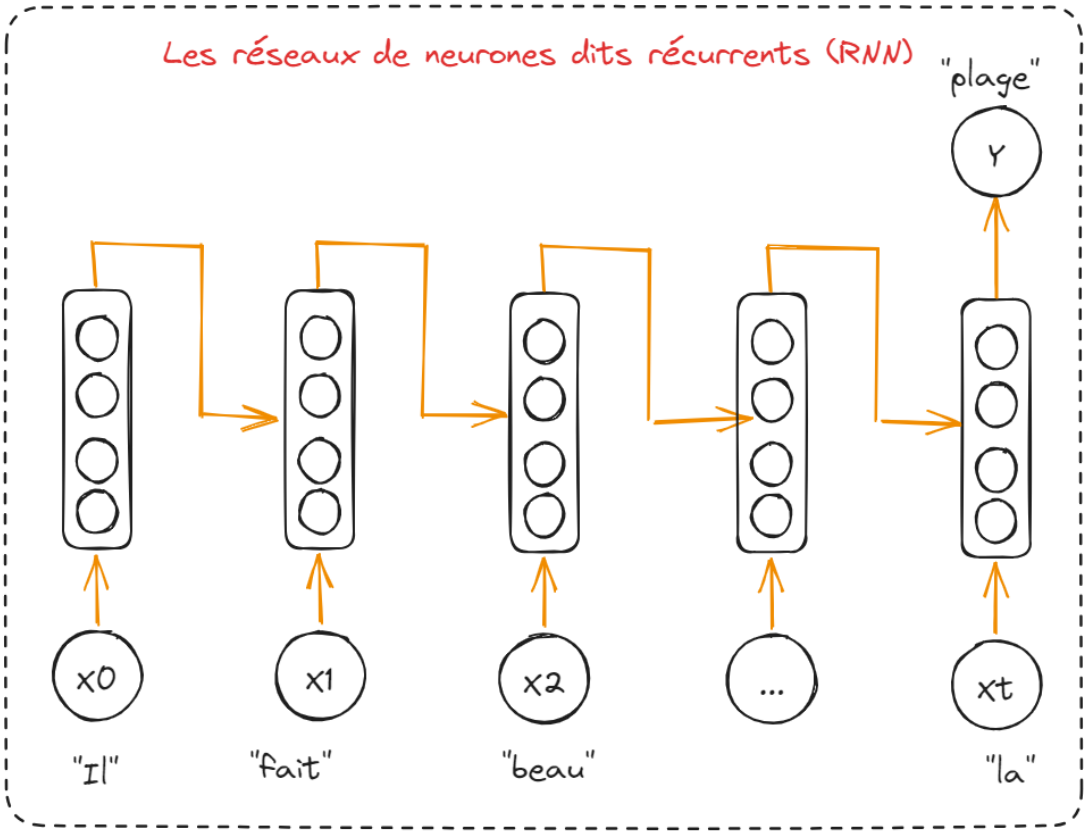

Figure 2 : Les réseaux de neurones récurrents

En ce qui concerne les algorithmes de vision par ordinateur classiques, des techniques telles que la détection de contours, la recherche de modèles et la correspondance d'objets sont souvent utilisées. Ces approches nécessitent souvent une ingénierie manuelle des caractéristiques et des classifieurs spécifiques pour détecter les images.

Une méthode couramment utilisée pour détecter des objets consiste à utiliser des caractéristiques locales telles que SIFT, SURF et ORB. Ces caractéristiques, appelées descripteurs, sont extraites autour des points d'intérêt identifiés dans l'image. Ils sont ensuite utilisés pour détecter et trouver des correspondances entre les objets.

La détection d'objets utilise une variété d'approches et de techniques, allant des méthodes basées sur l'apprentissage profond qui exploitent les capacités des réseaux de neurones convolutifs, aux algorithmes de vision par ordinateur classiques qui impliquent souvent une ingénierie manuelle des caractéristiques et des classifieurs spécifiques. Selon le contexte d'application et les critères de performance, certaines approches sont plus adaptées que d'autres, en raison de leurs forces et de leurs faiblesses respectives.

1.2. Les modèles existants

Dans le domaine de la détection d'objets, plusieurs travaux de recherche et modèles sont considérés comme pertinents. Parmi eux :

- **YOLO (You Only Look Once) :** YOLO est connu pour sa rapidité et sa précision. Il effectue la détection et la classification des objets en une seule passe sur l'image. Cela permet une détection en temps réel. Cependant, il peut avoir des difficultés avec de petits objets et des scènes complexes.
- **SSD (Single Shot MultiBox Detector) :** SSD est un modèle prisé offrant différentes tailles de boîtes englobantes à des résolutions variées. Il fusionne des caractéristiques à diverses échelles pour une détection plus précise. Bien que SSD soit rapide, il peut éprouver des difficultés avec les objets de petite taille.
- **Faster R-CNN (Region-based Convolutional Neural Networks) :** Faster R-CNN introduit une approche de région de proposition pour la détection d'objets. Il génère d'abord des régions potentielles, puis les classe et les affine pour obtenir les boîtes englobantes finales. Cela donne de bons résultats, précis, mais avec une vitesse de traitement plus lente.
- **RetinaNet :** RetinaNet est l'un des meilleurs modèles de détection d'objets qui s'est avéré efficace pour les objets denses et de petite taille. C'est la raison pour laquelle il est devenu un modèle de détection d'objets très répandu dans l'imagerie aérienne et satellitaire.
- **Mask R-CNN :** Mask R-CNN étend Faster R-CNN en ajoutant une étape de segmentation sémantique. Il est capable de détecter les objets et de segmenter

les régions correspondantes en masques précis. Mask R-CNN est utilisé pour des tâches telles que la détection d'objets avec une précision de segmentation fine, la détection des contours, etc.

Ces modèles ont des avantages spécifiques, comme la rapidité, la précision ou la segmentation précise des objets. Cependant, ils ont aussi des limites, comme des difficultés avec les petits objets ou les scènes complexes. Le choix du modèle dépend donc des exigences spécifiques de chaque application, qu'il s'agisse de la détection en temps réel, de la précision de segmentation, de la facilité d’utilisation ou d'autres critères de performance.

2. Analyse des besoins

2.1. Identification des exigences fonctionnelles et non fonctionnelles

Afin de répondre aux exigences spécifiques de ce projet, le système de détection des compteurs cyclistes doit impérativement inclure les fonctionnalités et caractéristiques suivantes :

- **Détection précise des vélos :** le programme de détection doit être capable de détecter de manière précise la présence de vélos. Cela signifie qu'il doit pouvoir distinguer les vélos des autres objets présents dans l'environnement, même dans des conditions de faible luminosité.
- **Identification des caractéristiques du vélo :** il est suggéré que le système puisse également fournir des informations supplémentaires sur le vélo détecté, comme s'il est chargé de bagages ou si la personne qui le conduit porte des équipements de protection. Cela permettra d'obtenir des données plus détaillées sur les comportements des cyclistes et d'évaluer l'utilisation des infrastructures et l'adoption des mesures de sécurité.
- **Utilisation sur un Raspberry Pi :** le programme de détection doit pouvoir être exécuté sur un Raspberry Pi, une plateforme populaire et peu coûteuse. Cela permettra un déploiement facile et économique du système.
- **Légèreté et optimisation du programme :** afin de permettre une utilisation efficace sur un Raspberry Pi, le programme de détection doit être léger et optimisé. Il doit être conçu pour offrir un taux de traitement par seconde significatif, permettant de traiter rapidement les données et de minimiser la perte d'informations de passage.
- **Programme open source :** celui-ci doit pouvoir être open source afin de pouvoir le proposer gratuitement à diverses structures comme des collectivités ou encore des associations et permettre à toutes personnes souhaitant le modifier, l’améliorer et le compléter de pouvoir le faire librement.

En résumé, le système de détection des compteurs cyclistes doit être capable de détecter avec précision la présence de vélos, d'identifier des caractéristiques spécifiques tels que le chargement du vélo ou les équipements de protection, d'être utilisé sur un Raspberry Pi et d'être léger et optimisé pour assurer un traitement rapide des données.

2.2. Définition des cas d'utilisation

Les cas d'utilisation spécifiques de ce projet de mémoire sont les suivants :

- **Détection du trafic cycliste :** le système est conçu pour détecter et compter les cyclistes sur les routes. Cela permettra aux collectivités d'étudier l'impact des évolutions de leurs infrastructures sur le trafic cycliste, en recueillant des données précises sur le nombre de cyclistes empruntant certaines voies.
- **Mesure de l'augmentation du trafic cycliste :** la pandémie, le développement des vélos à assistance électrique (VAE) et les progrès des infrastructures ont fait augmenter le nombre de cyclistes sur les routes. Il est donc essentiel de disposer d'un système performant pour mesurer cette tendance. Ce système permettra de collecter des données fiables, nécessaires pour apprécier l'importance de ce phénomène et le comprendre en profondeur.
- **Promotion du vélo et du tourisme cycliste :** le système peut être utilisé par des associations telles que "2 pieds 2 roues" ou "AF3V" qui promeuvent activement le vélo. Les données collectées par ce système seront d'une valeur inestimable pour renforcer leur plaidoyer en fournissant des informations précises sur l'utilisation des pistes cyclables et l'impact des politiques favorables au vélo. De plus, sur les itinéraires longue distance tels que les voies EuroVelo, le système peut jouer un rôle crucial en encourageant les commerçants à investir dans le tourisme cycliste. En démontrant le potentiel économique de ces itinéraires, il peut les convaincre de saisir cette opportunité passionnante.
- **Déploiement à moindre coût :** les compteurs cyclistes sont conçus pour être miniatures et autonomes, ce qui permet un déploiement à grande échelle à

moindre coût. Leur caractère open source favorise leur accessibilité et permet à un grand nombre d'acteurs de bénéficier de ces dispositifs de comptage.

Ce projet de mémoire vise à développer des compteurs cyclistes miniatures et autonomes, permettant la détection du trafic cycliste, la mesure de son augmentation, la promotion du vélo et du tourisme cycliste, ainsi que le déploiement à moindre coût pour les collectivités, les associations et les itinéraires longues distances.

2.3. Spécification des critères d'évaluation et de performance

Pour évaluer les performances de notre système de détection d'objets, nous avons défini plusieurs critères selon lesquels nous allons mesurer sa précision et son efficacité. Les critères d'évaluation comprennent :

- **Précision :** évaluation de la capacité du système à détecter tous les objets présents dans une scène donnée. Nous calculerons le taux de précision en comparant le nombre total d'objets détectés avec le nombre total d'objets réels.
- **Vitesse de détection :** mesure du temps nécessaire au système pour détecter les objets dans une image ou une vidéo. Nous enregistrerons le temps de traitement moyen par image ou par seconde pour évaluer l'efficacité du système.
- **Faux positifs et faux négatifs :** identification des erreurs de détection, notamment les objets mal classifiés ou manqués. Nous comptabiliserons les faux positifs (objets incorrectement détectés) et les faux négatifs (objets non détectés) pour évaluer la fiabilité du système.

En utilisant ces critères d'évaluation, nous pourrons analyser les performances de notre système de détection d'objets et identifier les domaines où des améliorations peuvent être apportées.

3. Informations sur la légalité de la reconnaissance d'image

3.1. Qui peut filmer la rue

Seules les autorités publiques (les mairies notamment) peuvent filmer la voie publique, les entreprises et les établissements accueillant du public n'ont pas l'autorisation. Ils peuvent seulement filmer les abords immédiats de leurs bâtiments et installations (la façade extérieure par exemple mais pas la rue en tant que telle) dans les lieux susceptibles d'être exposés à des actes de terrorisme.

Les particuliers ne peuvent filmer que l'intérieur de leur propriété. Ils ne peuvent pas filmer la voie publique, y compris pour assurer la sécurité de leur véhicule garé devant leur domicile.

La CNIL a publié sa position sur la question. Celle-ci considère légitime l'usage des caméras sur voie publique pour un dispositif comptabilisant les piétons, les voitures et les vélos dans le but d'aménager les zones. Cependant, si les personnes ne peuvent pas bénéficier de leur droit d'opposition, ces dispositifs doivent être autorisés par les services publics et ainsi obtenir une dérogation.

3.2. Traitements de données à caractère personnelle par dispositifs vidéo

Le traitement des données personnelles par le biais de dispositifs vidéo est réglementé par les directives du Comité européen de protection des données, en accord avec le RGPD. Selon le RGPD, ces règles ne s'appliquent pas si la vidéo ne permet pas d'identifier directement ou indirectement une personne, ou si elle est utilisée exclusivement à des fins personnelles ou domestiques. Toutefois, il convient de noter que cette exception est strictement évaluée dans le contexte de la vidéosurveillance.

Pour effectuer un traitement légal des données à caractère personnel par vidéo, plusieurs bases légales sont pertinentes. Il est possible de se baser sur les intérêts légitimes du responsable du traitement ou d'un tiers, à condition que ces intérêts soient réels, actuels et évalués par rapport à l'impact sur les droits et les

libertés des personnes filmées. Une autre base légale est la nécessité d'exécuter une mission d'intérêt public ou relevant de l'exercice de l'autorité publique. Le consentement libre, spécifique, informé et sans ambiguïté de la personne peut également être utilisé, bien que cela soit moins fréquent.

En ce qui concerne les droits des personnes filmées, le droit d'accès aux données peut être refusé dans certaines circonstances pour protéger les droits d'autres personnes présentes sur la vidéo ou si la recherche de l'image de la personne est rendue difficile en raison du volume important de données. Le droit à l'effacement et le droit d'opposition doivent être mis en œuvre en tenant compte de la spécificité des enregistrements vidéo. Ainsi, le floutage irréversible de l'image d'une personne enregistrée est considéré comme un effacement conforme au RGPD.

Le responsable du traitement doit informer les personnes concernées. Il peut afficher un panonceau avec les informations essentielles, renvoyant à un document plus détaillé conforme à l’article 13 du RGPD. Ces informations doivent être disponibles sous différents formats, numériques ou non, comme le téléphone, le bureau d’information ou l’affichage complet. Il faut que l’information soit accessible sans que la personne n’entre dans la zone filmée.

4. Conception et implémentation du système

4.1. Architecture globale du système

4.1.1. Description des différents composants et modules

Notre projet utilise plusieurs composants logiciels et modules pour implémenter un système de détection d'objets. L'un des principaux composants est le modèle de détection d'objets YOLOv5, qui est utilisé pour détecter les objets dans les images. Ce modèle est implémenté en utilisant la bibliothèque PyTorch pour le deep learning.

En plus du modèle de détection d'objets, notre système utilise également plusieurs modules pour prétraiter les données et gérer les résultats de la détection. Par exemple, nous utilisons un module pour convertir les résultats de la détection YOLOv5 en un format compatible avec la bibliothèque Norfair, qui est utilisée pour suivre les objets détectés dans les images.

Nous avons développé un site internet pour visualiser les données générées par notre programme. Pour cela, nous avons utilisé le framework Django et la librairie ChartJS pour créer un graphique clair et facilement filtrable. Nous avons choisi Django plutôt qu'une page web statique pour faciliter les futures améliorations, telles que le stockage des données en base de données ou la gestion des membres. Cela permettra d'ajouter de nouvelles fonctionnalités à notre projet à l'avenir.

Notre système est développé en Python, un langage de programmation populaire pour le développement de projets liés à l'apprentissage automatique et au traitement des données. Nous avons choisi ce langage en raison de sa facilité d'utilisation et de la disponibilité de nombreuses bibliothèques utiles pour notre projet.

En sommes, notre projet combine des composants et des modules pour créer un système efficace de détection et de visualisation d'objets. Nous avons opté pour des outils adaptés à nos besoins et faciles à utiliser, comme PyTorch et Python.

4.1.2. Présentation de la structure du code

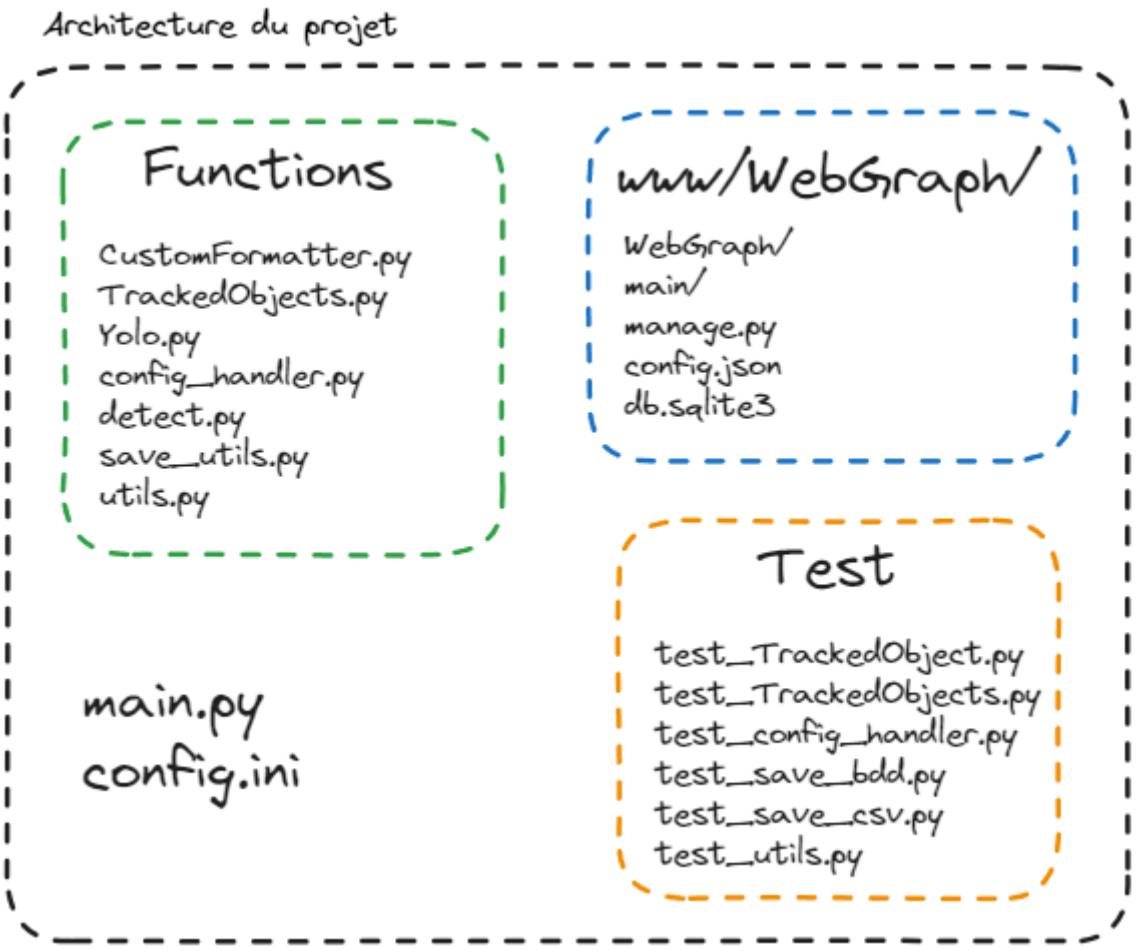

Figure 3 : Schéma de l'architecture du projet

Notre projet est organisé en plusieurs fichiers et répertoires comme nous pouvons le voir dans la Figure 3 pour faciliter la gestion du code source. Le répertoire principal contient les fichiers principaux du projet, tels que *main.py* qui lance le programme, et *config_handler.py* qui gère la configuration du système à partir du fichier config.ini ou d'un autre fichier s'il est passé en paramètre.

Le répertoire *Functions* contient plusieurs modules importants pour le fonctionnement de notre système. Par exemple, le module *Yolo.py* implémente l'interface avec le modèle de détection d'objets YOLOv5, tandis que le module *detect.py* gère la détection d'objets dans les images. D'autres modules tels que *TrackedObjects.py* et *utils.py* fournissent des fonctionnalités supplémentaires pour suivre les objets trouvés et traiter les résultats de la détection.

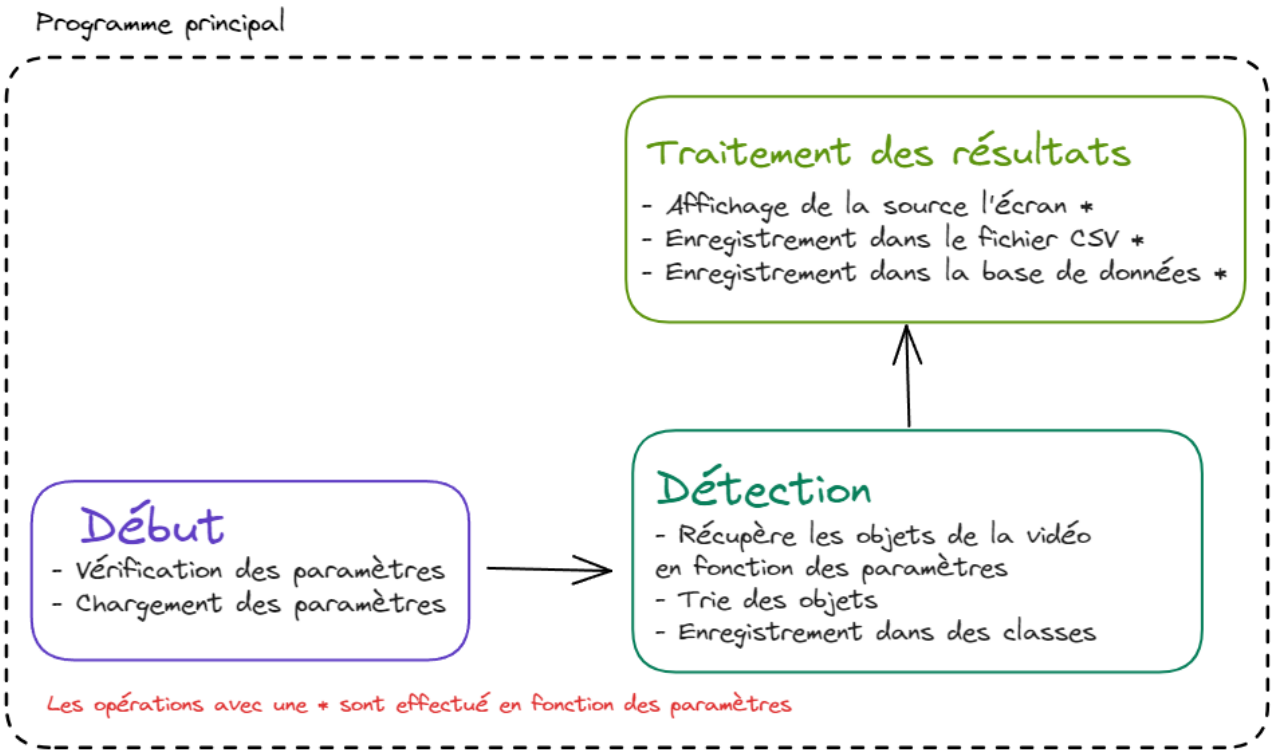

Figure 4 : Schéma de l'enchaînement des opérations

La Figure 4 montre le schéma de l'enchaînement des opérations qui sont effectuées par notre programme pour détecter les objets dans les images. Le programme commence par récupérer les objets de la vidéo en fonction des paramètres spécifiés, puis les trie pour les rendre uniques et les enregistre dans des classes qui permettront par la suite de calculer la direction. Selon les options de lancement, les résultats de la détection sont traités de différentes manières : affichage à l'écran de la source avec des boîtes autour des objets détectés et des informations sur leur direction, enregistrement dans un fichier CSV ou dans une base de données. Ce schéma montre comment les différents composants et modules de notre système interagissent entre eux pour traiter les données et détecter les objets dans les images.

Notre système utilise des algorithmes et des techniques avancées pour détecter les objets dans les images. Nous avons implémenté le modèle YOLOv5 pour la détection d'objets, en utilisant la bibliothèque PyTorch pour entraîner et exécuter le modèle. Nous avons également adapté notre système pour utiliser la bibliothèque Norfair pour suivre les objets détectés dans les images.

Concernant la visualisation des données sous forme graphique, nous avons créé un site avec le framework Django qui se trouve dans le dossier *www*. Ce site permet de recevoir un fichier CSV et génère un graphique qui peut être filtré pour afficher une sélection de points entre deux dates ou un certain type de classe d'objets.

Nous avons par ailleurs, réalisé différents tests unitaires pour tester notre code et éviter de casser des fonctionnalités lors des modifications, ces tests se trouvent dans le dossier Test et sont exécutés à chaque commit et pull-request sur la branche *develop* et *master* de github.

Pour réaliser notre projet, nous avons utilisé uniquement des librairies et des projets open source, qui nous ont permis de bénéficier de leur code et de leur expertise. Ils sont tous sous licence GPL-3.0 ou compatible, ce qui signifie qu'ils garantissent la liberté d'utiliser, de modifier et de redistribuer le code source, à condition de respecter les mêmes conditions pour les versions dérivées. Nous avons donc choisi de placer notre projet sous licence GPL-3.0 également, afin de respecter l'esprit de l'open source et de contribuer à la communauté.

Notre projet est organisé en plusieurs fichiers et répertoires pour faciliter la gestion du code source. Nous avons réalisé des algorithmes et des techniques avancées pour détecter les objets dans les images, en utilisant des outils tels que PyTorch et Norfair pour améliorer les performances de notre système.

4.1.3. Les différentes versions du système

Dans cette partie, nous aborderons les différentes versions de notre projet, avec leurs nouveautés et les problèmes qu'elles comportent. Les différentes versions sont également disponibles et téléchargeables dans la section "Release" du projet sur Github, à l'adresse suivante : Releases · Drosscend/MiniBox (github.com).

La première version de notre programme avait pour objectif de prendre des photos à intervalles réguliers à l'aide d'une caméra, puis de les enregistrer dans un dossier. Ensuite, un autre programme utilisait ces photos pour détecter les objets tels que les vélos, les voitures et les personnes, et enregistrait ces données dans un fichier

CSV. Pour développer cette version, nous avons décidé d'utiliser la bibliothèque YOLOV5 téléchargée depuis Github. Cependant, cette version avait plusieurs problèmes, notamment en termes de temps de détection, de réglementation liée à l'enregistrement des photos et de manque d'informations pertinentes dans le fichier CSV.

Dans la deuxième version de notre programme, nous nous sommes concentrés sur l'amélioration du système de détection et de sa rapidité. Nous avons découvert des technologies existantes permettant d'utiliser le modèle créé par YOLOV5 avec PyTorch, une bibliothèque open source développée par Facebook. En utilisant PyTorch, nous avons pu éviter de cloner le projet YOLOV5 localement sur notre ordinateur et transmettre directement les données de l'image au modèle pour obtenir les informations nécessaires. Cette avancée nous a permis de gagner du temps lors de la détection et d'éviter tous les problèmes liés au RGPD. De plus, cette nouvelle version incluait la possibilité de suivre les personnes en utilisant SORT, une bibliothèque Open Source spécialisée dans le suivi d'objets. Malgré ces améliorations, cette version présentait encore des problèmes, notamment en ce qui concerne le suivi peu fiable des objets et la vitesse insuffisante pour un traitement en temps réel.

La troisième version du programme comprend trois sous versions. La version 3.0 estimait la direction de chaque personne à partir de ses positions passées et offrait la possibilité d’enregistrer ces données dans une base de données. Elle permettait aussi à l’utilisateur de configurer le programme avec précision grâce à un fichier de configuration. La version 3.1 améliorait le suivi des objets avec Norfair, une bibliothèque Python légère et personnalisable pour le suivi en temps réel de plusieurs objets. Elle augmentait aussi le nombre de traitements par seconde en utilisant une carte graphique Nvidia avec PyTorch. La version 3.2 ajoutait la visualisation du fichier CSV sur un site internet développé avec Django.

4.1.4. Analyse de code clé dans notre système de détection d'objets

Dans cette partie nous allons vous présenter quelques parties du code qui sont importantes et qui réalisent des actions importantes.

Détection de la direction

```
CALCUL_DIRECTION_NB_POSITIONS = 4  # Nombre de positions à prendre en compte pour
calculer la direction
SPEED_THRESHOLD = 10  # Vitesse minimale pour que la direction soit prise en compte
def calculate_direction(positions: list) -> Optional[str]:
    total_distance = 0
    total_dx = 0
    total_dy = 0
    for i in range(1, CALCUL_DIRECTION_NB_POSITIONS):
        x1, y1, _, _ = positions[-i - 1]
        x2, y2, _, _ = positions[-i]
        dx = x2 - x1
        dy = y2 - y1
        distance = math.sqrt(dx ** 2 + dy ** 2)
        total_distance += distance
        total_dx += dx
        total_dy += dy

    mean_dx = total_dx / CALCUL_DIRECTION_NB_POSITIONS
    mean_dy = total_dy / CALCUL_DIRECTION_NB_POSITIONS
    mean_distance = total_distance / CALCUL_DIRECTION_NB_POSITIONS

    if mean_dx > 0 and mean_dy > 0:
        if mean_distance > SPEED_THRESHOLD:
            return "bottom-right"
    elif mean_dx > 0 > mean_dy:
        if mean_distance > SPEED_THRESHOLD:
            return "top-right"
    elif mean_dx < 0 < mean_dy:
        if mean_distance > SPEED_THRESHOLD:
            return "bottom-left"
    elif mean_dx < 0 and mean_dy < 0:
        if mean_distance > SPEED_THRESHOLD:
            return "top-left"
    else:
        return None
```

Ce code calcule la direction d'un objet en mouvement en utilisant ses positions précédentes. Il utilise ces 4 dernières positions pour calculer la direction moyenne et vérifie si la vitesse de l'objet est supérieure à un seuil prédéfini (10). Si la vitesse est supérieure au seuil, la fonction renvoie la direction de l'objet sous forme de chaîne de

caractères (“bottom-right”, “top-right”, “bottom-left” ou “top-left”). Sinon, elle renvoie None.

Sauvegarde dans le fichier CSV

```
def save_csv(list_of_directions: dict[Any, dict[str, int]], yolov5_paramms) -> None:
    csv_folder_name = yolov5_paramms["output_folder"]
    csv_file_name = yolov5_paramms["csv_name"]
    # verifie que le dosser csv_folder_name existe et le crée si ce n'est pas le cas
    if not os.path.exists(csv_folder_name):
        log.info("Création du dossier {}".format(csv_folder_name))
        os.makedirs(csv_folder_name)

    # récupération de la date au format international et UTC
    datestring = datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S")

    # enregistrement des données dans un fichier csv
    try:
        path = os.path.join(csv_folder_name, csv_file_name)
        with open(path, 'a', newline='') as f:
            writer = csv.writer(f)
            # si le fichier est vide, on écrit l'entête
            if os.path.getsize(path) == 0:
                writer.writerow(["date", "occurrence", "top-left", "top-right",
"bottom-left", "bottom-right", "classe"])
            for classe, infos in list_of_directions.items():
                writer.writerow([
                    datestring,
                    infos["total"],
                    infos["top-left"],
                    infos["top-right"],
                    infos["bottom-left"],
                    infos["bottom-right"],
                    classe
                ])
    except IOError as e:
        log.warning("Erreur lors de l'écriture dans le fichier CSV: " + str(e))
```

Ce code génère un fichier CSV qui récapitule le nombre total d'occurrences et la répartition par direction pour chaque classe d'objets détectés. Pour cela, la fonction requiert deux éléments en entrée : un dictionnaire contenant les directions des objets détectés classés par classe, ainsi qu'un dictionnaire de paramètres appelé yolov5_paramms qui comprend le nom du dossier et du fichier CSV.

La fonction effectue plusieurs vérifications et opérations. Tout d'abord, elle s'assure de l'existence du dossier spécifié. Si le dossier n'existe pas, elle le crée, ensuite, elle récupère la date actuelle au format international et UTC.

Une fois ces étapes préliminaires réalisées, la fonction enregistre les données dans le fichier CSV. Si ce dernier est vide, elle écrit l'en-tête. Dans le cas contraire, elle ajoute les données correspondantes à chaque classe d'objets détectés.

Classe permettant la détection d'objet

```
import logging
from typing import List, Optional, Union

import numpy as np
import torch

log = logging.getLogger("main")


class YOLO:
    def __init__(self, model_name: str, device: str, verbose: bool):
        # Vérification de la disponibilité de CUDA si nécessaire
        if device != "cpu" and not torch.cuda.is_available():
            log.error("Vous avez demandé un device CUDA, mais il n'est pas
disponible")
            exit(1)

        # Chargement du modèle
        self.model = torch.hub.load("ultralytics/yolov5", model_name, device=device,
verbose=verbose)

    def __call__(self,
                 img: Union[str, np.ndarray],
                 conf_threshold: float = 0.25,
                 iou_threshold: float = 0.45,
                 agnostic: bool = False,
                 multi_label: bool = True,
                 max_det: int = 50,
                 amp: bool = True,
                 classes: Optional[List[int]] = None
                 ) -> torch.tensor:
        self.model.conf = conf_threshold
        self.model.iou = 0.45
        self.model.agnostic = agnostic
        self.model.multi_label = multi_label
        self.model.max_det = max_det
        self.model.amp = amp

        # si aucune classe n'est spécifiée, on utilise toutes les classes
        if classes is not None:
            self.model.classes = classes

        detections = self.model(img, size=640)
        return detections
```

Ce code définit une classe *YOLO* qui utilise le modèle YOLOv5 pour la détection d'objets dans des images. La classe prend en entrée le nom du modèle à charger, le device sur lequel le modèle doit être chargé et un booléen pour afficher les informations de chargement du modèle. La méthode *__call__* de la classe prend en entrée une image et plusieurs paramètres optionnels pour la détection d'objets, tels que les seuils de confiance et d'intersection, les classes à détecter, etc. La méthode renvoie les résultats de la détection sous forme de tenseur PyTorch. Un tenseur PyTorch est une matrice multidimensionnelle contenant des éléments d'un seul type de données.

Détection et suivi des objets

```
        try:
            results = model(
                frame,
                conf_threshold=yolov5_paramms["conf_thres"],
                iou_threshold=yolov5_paramms["iou_thres"],
                classes=base_params["classes"],
                agnostic=yolov5_paramms["agnostic_nms"],
                multi_label=yolov5_paramms["multi_label_nms"],
                max_det=yolov5_paramms["max_det"],
                amp=yolov5_paramms["amp"],
            )
        except Exception as e:
            log.error("Erreur lors du traitement de l'image avec le modèle YOLOv5:
{}".format(e))
            continue

        detections = utils.yolo_detections_to_norfair_detections(results)

        try:
            # Utilisation de la librairie Sort pour suivre les personnes détectées
            tracked_objects = tracker.update(detections=detections)
        except Exception as e:
            log.error("Erreur lors du suivie des objets: {}".format(e))
            continue
```

Ce code utilise la classe précédemment décrite *YOLO* pour détecter des objets dans une image. Les paramètres du modèle, tels que les seuils de confiance et d'intersection, les classes à détecter, etc., sont définis à partir d'un dictionnaire yolov5_paramms et sont passés au modèle. Si une erreur se produit lors du traitement de l'image avec le modèle YOLOv5, un message d'erreur est affiché à l'utilisateur et le code continue. Les résultats de la détection sont ensuite convertis en

détections Norfair à l'aide de la fonction *yolo_detections_to_norfair_detections*. Ensuite, la librairie Norfair est utilisée pour suivre les objets détectés. Si une erreur se produit lors du suivi des objets, un message d'erreur est enregistré et le code continue.

Transformation des résultats de Pytorch en résultant utilisable par Norfair

```
def yolo_detections_to_norfair_detections(yolo_detections: torch.tensor) ->
List[Detection]:
    norfair_detections: List[Detection] = []

    detections_as_xyxy = yolo_detections.xyxy[0]
    for detection_as_xyxy in detections_as_xyxy:
        bbox = np.array(
            [
                [detection_as_xyxy[0].item(), detection_as_xyxy[1].item()],
                [detection_as_xyxy[2].item(), detection_as_xyxy[3].item()],
            ]
        )
        scores = np.array(
            [detection_as_xyxy[4].item(), detection_as_xyxy[4].item()]
        )
        norfair_detections.append(
            Detection(
                points=bbox, scores=scores, label=int(detection_as_xyxy[-1].item())
            )
        )

    return norfair_detections
```

La fonction *yolo_detections_to_norfair_detections* convertit les résultats de la détection YOLOv5, sous forme de tenseur PyTorch, en format Norfair, sous forme de liste d'objets *Detection*. Chaque objet *Detection* contient les coordonnées de la boîte englobante, le score de confiance et l'étiquette de classe de l'objet détecté par YOLOv5."

4.2. Expérimentation et résultat

4.2.1. Méthodologie d'évaluation

Pour évaluer les performances de notre système de détection d'objets, nous avons utilisé une méthodologie rigoureuse comprenant des jeux de données spécifiques et des métriques de performance standard. Nous avons évalué notre système sur des vidéos filmées dans des parcs et des rues passantes, où nous pouvons voir des vélos circuler. Nous avons visionné ces vidéos au préalable pour comparer les résultats attendus avec les données acquises par notre système.

Nous avons utilisé plusieurs métriques de performance pour évaluer notre système, notamment la précision et la vitesse de traitement. La précision mesure la capacité du système à détecter tous les objets présents dans les images. La vitesse de traitement mesure la rapidité avec laquelle le système peut traiter les images.

Nous avons effectué différents tests sur une vidéo de 3 minutes et 54 secondes montrant un fort passage de vélos, coureurs et voitures. Les informations de la vidéo sont les suivantes:

- Nombre de frames: 7096
- Nombre de vélos circulant: 20
- Nombre de vélo allant en haut à droite: 1
- Nombre de vélos allant en haut à gauche: 3
- Nombre de vélos allant en bas à droite: 13
- Nombre de vélos allant en bas à gauche: 3

Notre programme a généré le fichier CSV suivant:

```
date,occurrence,top-left,top-right,bottom-left,bottom-right,classe
2023-06-13 19:11:55,4,0,1,0,2,1
2023-06-13 19:12:55,10,0,1,0,8,1
2023-06-13 19:13:55,5,1,2,0,2,1
2023-06-13 19:14:30,2,0,1,1,0,1
```

D’après ce fichier CSV, nous pouvons observer que notre programme a détecté un total de 21 vélos dont:

- 1 se dirigeant en haut à gauche
- 4 se dirigeant en haut à droite
- 1 se dirigeant en bas à gauche
- 12 se dirigeant en bas à droite

Les résultats sont plutôt corrects malgré quelques imprécisions dans les directions. Le traitement de cette vidéo a été réalisé en 3 minutes et 34 secondes avec environ 33 itérations par seconde en utilisant une carte graphique.

Les résultats de notre évaluation ont montré que notre système est capable de détecter les vélos dans les vidéos, avec une précision élevée. Notre système a

également montré une vitesse de traitement rapide, ce qui le rend adapté à une utilisation en temps réel. En comparaison avec d'autres approches existantes, notre système a montré des performances compétitives en termes de précision.

En résumé, notre méthodologie d'évaluation rigoureuse a permis de mesurer les performances de notre système de détection d'objets sur des jeux de données spécifiques. Les résultats ont montré que notre système est capable de détecter avec précision et rapidité les vélos dans les vidéos.

4.2.2. Guide d'utilisation du programme

Ce guide vous expliquera comment installer, configurer et utiliser notre système de détection d'objets.

Étape 1 : Installation

1. Installation de Python

Python est le langage que nous avons utilisé pour développer notre projet, assurez-vous d'avoir un environnement Python supérieur à 3.11 installé sur votre système. Pour cela vous pouvez exécuter la commande suivante :

```
python --version
```

Figure 5 : Vérification de la version de Python

Si cette commande vous retourne une erreur cela indique que vous n'avez pas Python sur votre ordinateur, si cette commande vous retourne un numéro de version antérieure à la version 3.11 vous devrez mettre à jour Python.

Pour installer ou mettre à jour Python allez sur le site suivant Python Release Python 3.11.4 | Python.org et choisissez le lien correspondant à votre système d'exploitation et à votre architecture en bas de page. Pendant l'installation de Python, n'oubliez pas de cocher la case "add python to environment variables" pour simplifier les étapes suivantes.

Pour vérifier que vous possédez la bonne version de Python, vous pouvez relancer la commande précédente dans un nouveau terminal.

2. Téléchargement des sources

Après avoir téléchargé Python sur votre ordinateur, vous devez obtenir les sources du projet. Allez sur le lien [Releases · Drosscend/MiniBox (github.com)] et téléchargez la dernière version disponible, qui est actuellement la version **3.3.2**. Enregistrez-la sur votre ordinateur.

3. Création de l'espace d'exécution

Pour exécuter l'application et éviter d'avoir des conflits avec des applications déjà présentes sur l'ordinateur et différentes versions des dépendances que nous utilisons, nous vous conseillons d'utiliser un environnement virtuel. Pour cela ouvrez un terminal dans le dossier que vous souhaitez et entrer la commande suivante :

```
py -m venv .minibox # pour les utilisateurs sous Windows
python3 -m venv .minibox # pour les utilisateurs sous Linux
```

Cette commande crée l'environnement virtuel, une fois créé nous avons besoin d'activer cet environnement pour cela entrez la commande suivante :

```
.minibox\Scripts\activate # pour les utilisateurs sous Windows
source .minibox/bin/activate # pour les utilisateurs sous Linux
```

Figure 6 : Capture-d'écran de la création de l'environnement virtuel - commandes

Une fois ces deux commandes exécutées vous devriez avoir un dossier nommé ".minibox" créé dans le dossier, celui-ci contiendra toutes les librairies pour faire fonctionner le projet correctement.

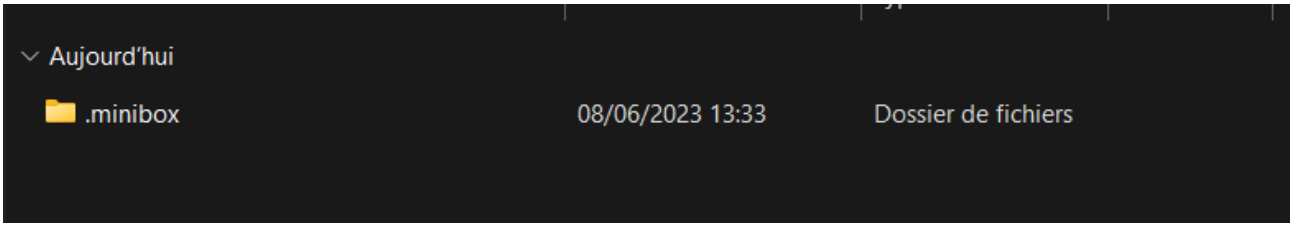

Figure 7 : Capture-d'écran de la création de l'environnement - dossier créé

Vous pouvez maintenant déplacer le contenu du fichier zip téléchargé précédemment dans ce dossier comme dans la capture ci-dessous :

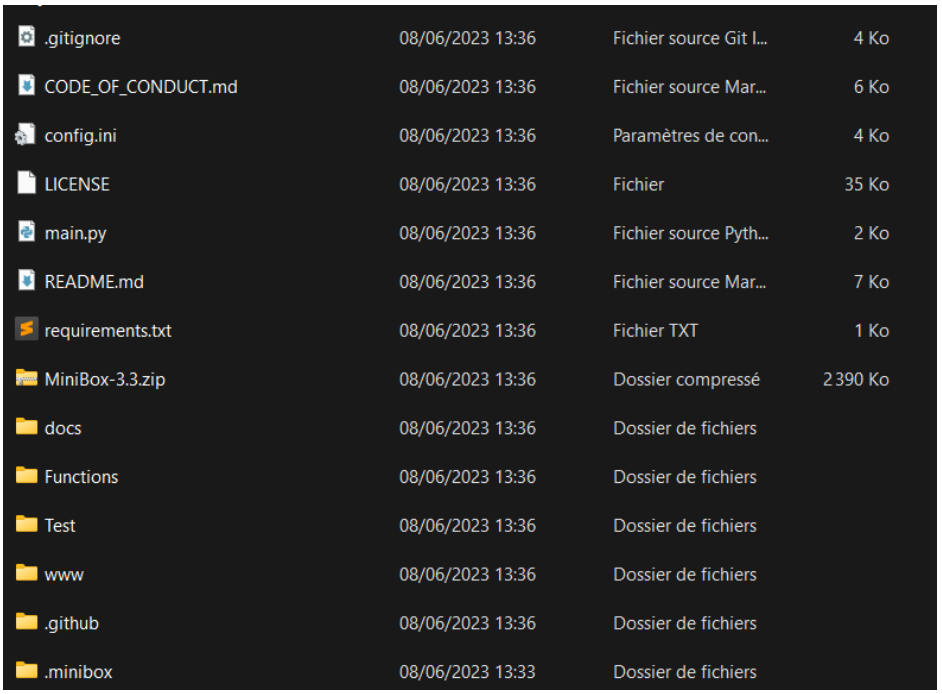

Figure 8 : Capture-d'écran des sources du projet

4. Installation des dépendances utiles au bon fonctionnement du projet

Pour télécharger les dépendances nécessaires au projet, ouvrez le terminal où vous avez activé l'environnement et tapez la commande suivante :

```
pip install -r requirements.txt # pour les utilisateurs sous Windows
pip3 install -r requirements.txt # pour les utilisateurs sous Linux
```

Cette commande télécharge toutes les dépendances nécessaires à partir du fichier "requirements.txt".

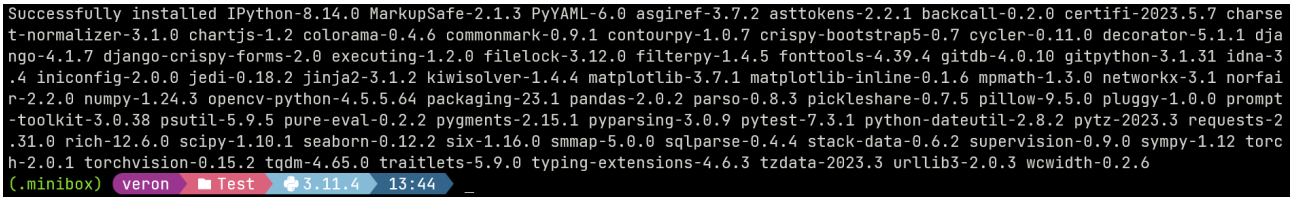

Figure 9 : Capture-d'écran de l'installation des dépendances nécessaires au programme

Si vous avez une carte graphique Nvidia et que vous souhaitez l'utiliser, vous pouvez taper la commande suivante, mais soyez patient car cette opération est longue et ralentit considérablement votre ordinateur.

```
pip install -U torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cu117 # pour les utilisateur de Windows
pip3 install -U torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cu117 # pour les utilisateur de Linux
```

Figure 10 : Capture-d'écran de l'installation des dépendances utiles pour utiliser les pilotes graphiques

5. Configuration du site web pour visualiser le fichier CSV

Pour configurer le serveur web, allez dans le dossier "www/WebGraph" et créez un fichier "config.json" avec le contenu ci-dessous (remplacer "votre secret key" par ce que vous voulez).

```
{
    "SECRET_KEY": "votre secret key"
}
```

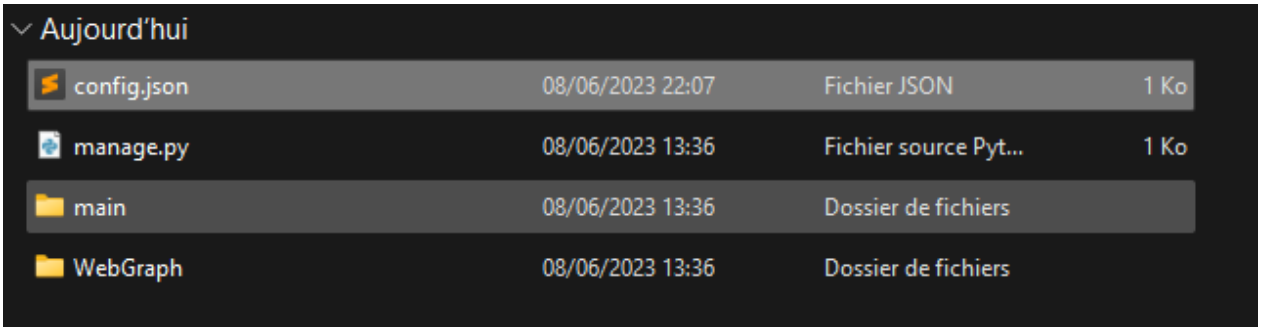

Figure 11 : Capture-d'écran de la configuration du serveur web

Pour terminer la partie installation, dans le même terminal que précédemment, tapez la commande suivante :

```
python www/WebGraph/manage.py migrate # pour les utilisateurs sous Windows
python3 www\WebGraph\manage.py migrate # pour les utilisateurs sous Linux
```

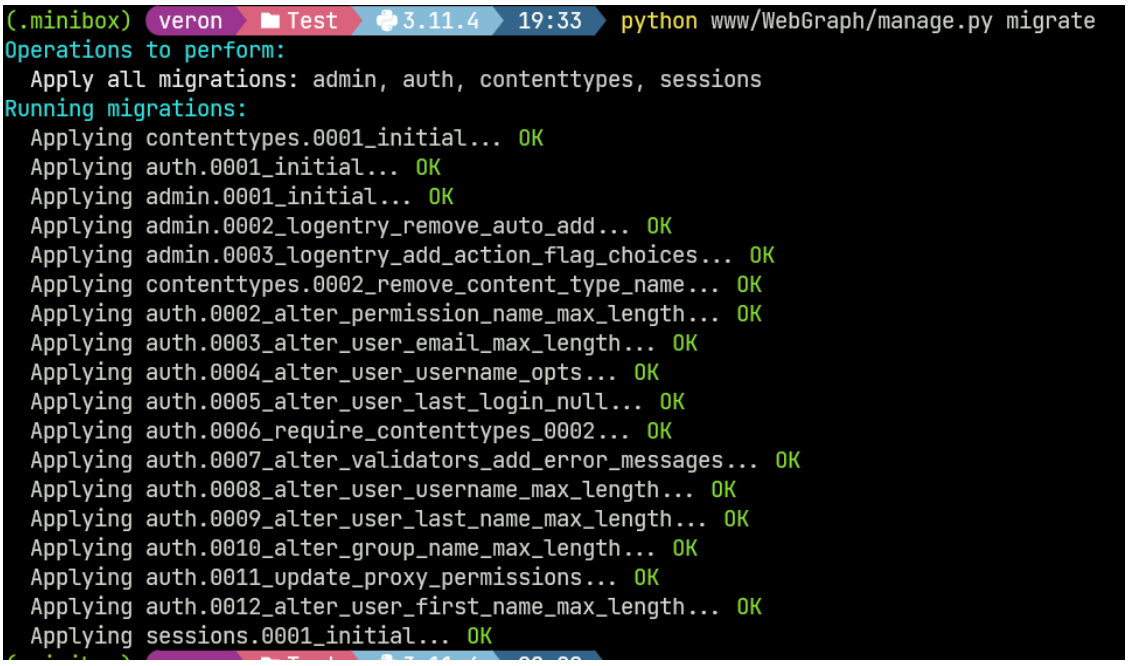

Figure 12 : Capture-d'écran de la génération de la base de données

Cette commande construit la base de données du serveur.

Étape 2 : Configuration

Pour personnaliser le projet, ouvrez le fichier "config.ini" et changez les options. Chaque ligne a un commentaire pour vous expliquer chaque option.

Si vous voulez plusieurs fichiers de configuration pour différents cas (dev, prod), copiez ce fichier et donnez-lui le nom que vous souhaitez. Mais gardez la

même disposition des paramètres et n'en supprimez pas. En annexe, il y a un fichier de configuration personnalisé qui utilise une vidéo "video3.mp4", qui affiche le résultat à l'écran et qui fonctionne avec la carte graphique (Fichier de configuration personnalisé du programme). Pour utiliser ce fichier de configuration, tapez la commande suivante :

```
python main.py -c customconfig.ini # pour les utilisateurs sous Windows
python3 main.py -c customconfig.ini # pour les utilisateurs sous Linux
```

Étape 3 : Exécution

1. Ouvrez le terminal que vous avez utilisé avant, ou si vous l'avez fermé, exécutez la commande ci-dessous comme à la première étape. Soyez dans le dossier des sources du projet.

```
.minibox\Scripts\activate # pour les utilisateurs sous Windows
source .minibox/bin/activate # pour les utilisateurs sous Linux
```

2. Exécutez la commande suivante pour lancer le programme :

```
python .\main.py # pour les utilisateurs sous Windows
python3 .\main.py # pour les utilisateurs sous Linux
```

Si vous avez un fichier de configuration personnalisé exécuter la commande suivante :

```
python .\main.py -c .\custom_config.ini # pour les utilisateurs sous Windows
python3 .\main.py -c .\custom_config.ini # pour les utilisateurs sous Linux
```

Si vous avez une carte graphique Nvidia et que vous avez installé les dépendances pour l'utiliser, vérifiez que l'option "device" est à 0 et non à cpu dans le fichier de configuration.

```
device = 0
```

3. Le système de détection d'objets sera lancé et commencera son traitement

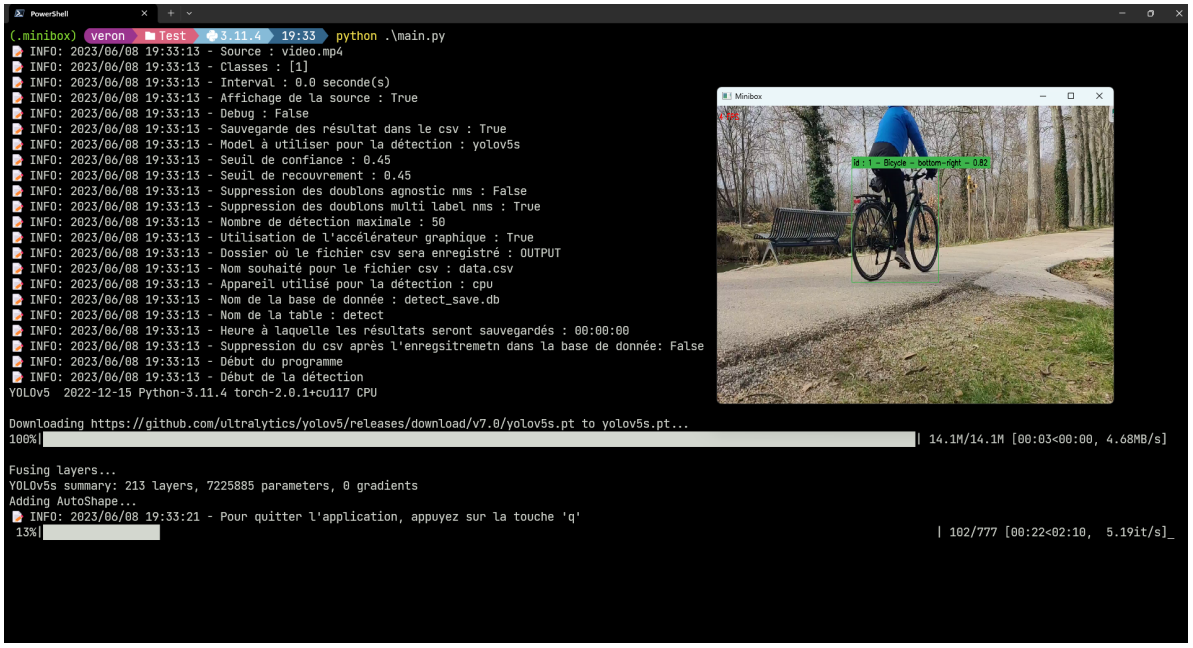

Figure 13 : Capture-d'écran de l'exécution du programme

Pour arrêter le traitement, vous pouvez :

- Si l'option "display_detection" est à "false", faire "CTRL+C" dans le terminal.
- Si l'option est à "true", appuyez sur "q" sur la fenêtre avec la source vidéo.

Étape 4 : Affichage des résultats

D'abord, vous pouvez voir les données du fichier CSV dans le dossier "OUTPUT" du projet. Il y a un fichier nommé "data.csv" avec des données comme celles-ci :

```
date,occurrence,top-left,top-right,bottom-left,bottom-right,classe
2023-05-30 11:09:00,2,0,0,0,2,1
2023-05-30 11:10:00,1,0,0,0,0,1
2023-05-30 11:11:00,4,1,1,0,2,1
2023-05-30 11:12:00,5,0,1,1,3,1
2023-05-30 11:13:00,3,0,2,0,1,1
2023-05-30 11:14:00,4,0,1,0,3,1
```

Pour voir les résultats en graphiques, vous devez démarrer le serveur avec le site web. Dans le terminal, tapez la commande suivante :

```
python www/WebGraph/manage.py runserver # pour les utilisateurs sous Windows
python3 www\WebGraph\manage.py runserver # pour les utilisateurs sous Linux
```

```
(.minibox) veron Test 3.11.4 22:08 python www/WebGraph/manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 08, 2023 - 22:15:37
Django version 4.1.7, using settings 'WebGraph.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 14 : Capture-d'écran du lancement du serveur web

Quand le serveur est lancé, allez sur l'url http://127.0.0.1:8000/, vous devriez voir ceci :

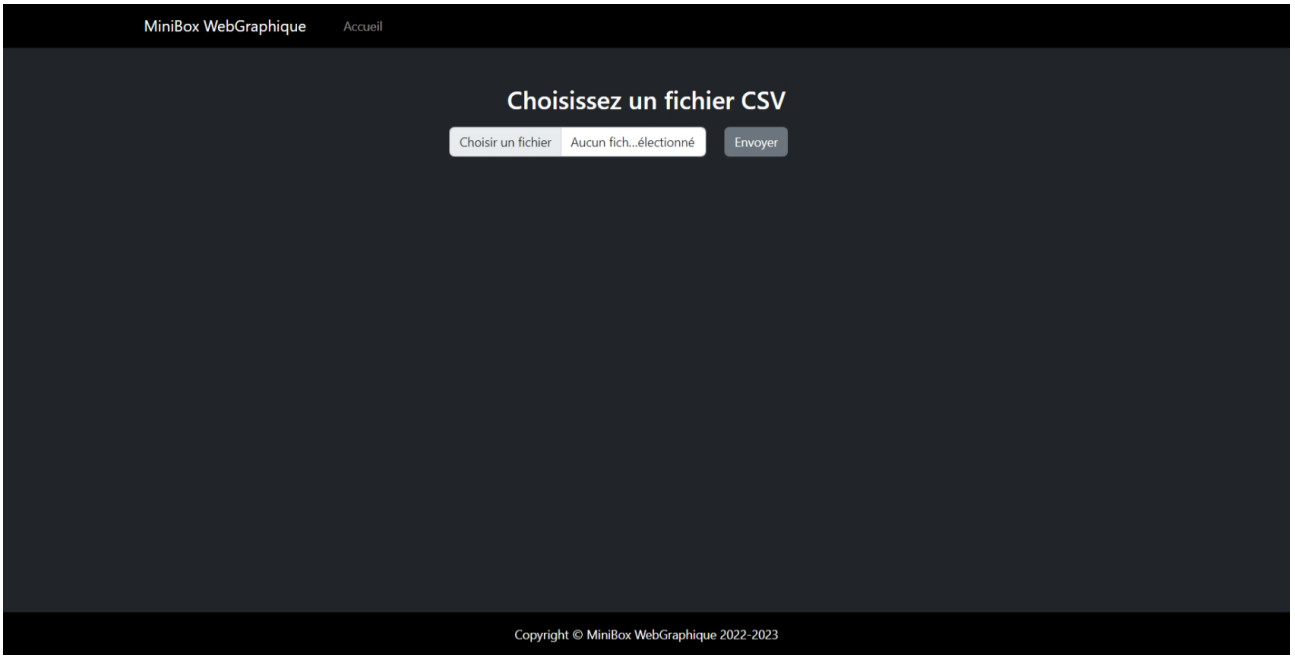

Figure 15 : Capture-d'écran du site web

Ajouter le fichier CSV disponible dans le dossier "Output" puis cliquer sur "Envoyer"

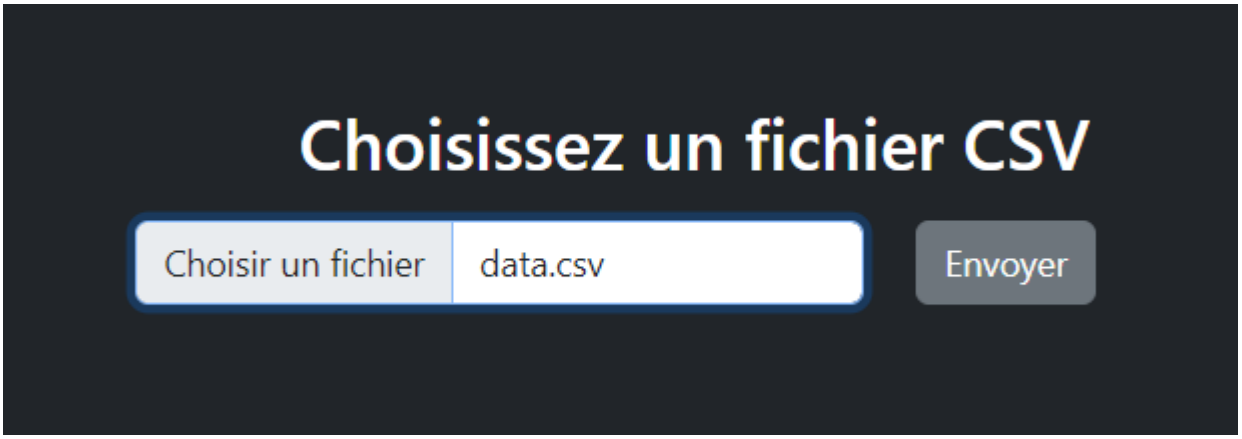

Figure 16 : Capture-d'écran de l'ajout du fichier csv

Vous pouvez maintenant visualiser les données et sélectionner la plage de date à afficher et les classes s'il y en à plusieurs :

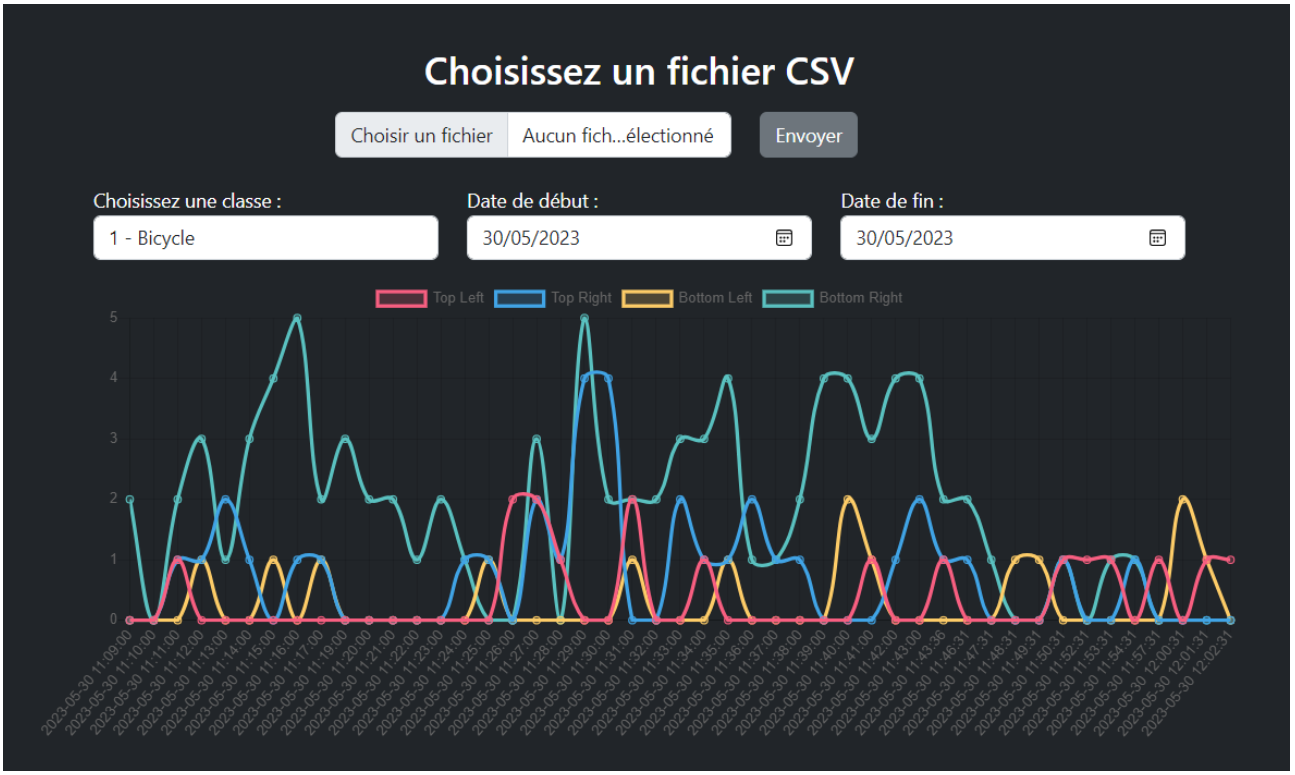

Figure 17 : Capture-d'écran du graphique comportant les données du csv

Étape 5 : Terminer correctement le programme

Pour finir le programme proprement, vous pouvez faire "CTRL+C" dans le terminal pour arrêter le serveur web et taper la commande suivante pour quitter l'environnement virtuel.

```
deactivate
```

Figure 18 : Capture-d'écran de la commande pour quitter l'environnement virtuel

Si vous voulez le rouvrir, tapez simplement la commande suivante :

```
.minibox\Scripts\activate # pour les utilisateurs sous Windows
source .minibox/bin/activate # pour les utilisateurs sous Linux
```

5. Limitations et perspectives

5.1. Limite actuelles du systèmes et des méthodes utilisées

Les limites actuelles de notre système de détection d'objets sont multiples. Tout d'abord, bien que nous soyons capables de détecter plusieurs classes d'objets simultanément, nous rencontrons des problèmes d'assignation lors du suivi, ce qui entraîne un traitement erroné par la suite. Par exemple, il arrive que notre système attribue l'identifiant d'une voiture à un vélo, et vice versa, ce qui compromet la fiabilité des résultats.

En outre, nous avons rencontré des difficultés lors de l'installation de notre système sur Raspberry. Cette contrainte technique rend l'utilisation de notre système moins accessible pour certains utilisateurs potentiels qui souhaiteraient l'utiliser sur cette plateforme spécifique.

Une autre limitation importante est notre incapacité à identifier des informations relatives spécifiques, telles que la charge d'un vélo. Notre système actuel n'est pas capable de faire la distinction entre un vélo chargé et un vélo vide, ce qui peut être une donnée pertinente dans certains contextes d'utilisation.

Ces limitations sont essentielles à prendre en compte pour améliorer notre système de détection d'objets. Nous devons travailler sur les algorithmes d'assignation lors du suivi pour éviter les confusions entre les classes d'objets. De plus, nous devons simplifier le processus d'installation pour rendre notre système plus accessible aux utilisateurs de Raspberry.

Enfin, il est crucial d'améliorer notre capacité à extraire des informations relatives spécifiques, comme la charge d'un vélo, afin d'enrichir les fonctionnalités de notre système. En repoussant les limites actuelles, nous pourrons offrir une expérience plus précise et plus complète à nos utilisateurs.

5.2. Propositions d'amélioration et de développement futur

Pour surmonter les limitations actuelles de notre système, nous avons identifié plusieurs pistes d'amélioration et de développement futur. Tout d'abord, une amélioration majeure consisterait à intégrer YOLOV8 avec son suivi intégré. Cette version développée par Ultralitycs est dans la même lignée que YOLOV5, mais elle offre la particularité d'utiliser directement un système de suivi d'objets. Malheureusement, nous n'avons pas pu utiliser cette version en raison de difficultés rencontrées avec une librairie appelée "lap" qui n'est pas compatible avec notre version de Python sous Windows. Cependant, en surmontant ce problème, l'intégration de YOLOV8 améliorerait la qualité de la détection et du suivi des objets, réduisant ainsi les faux positifs. De plus, cela permettrait d'augmenter le nombre de traitements par seconde en exploitant les nouveaux modèles disponibles.

Nous avons entrepris d'examiner comment résoudre ce problème en créant une demande sur le projet YOLOV8 à l'adresse suivante : Error when using track task · Issue #1328 · ultralytics/ultralytics (github.com). Nous avons également appliqué les solutions qui ont été proposées par des personnes dans les commentaires de cette demande et d'autres demandes similaires. Nous avons également réalisé une alternative au programme en utilisant la librairie numpy, mais après avoir été testé par les membres de l'équipe Ultralytics elle n'est pas encore suffisamment convaincante pour être pleinement utilisée à l'heure actuelle.

Une autre possibilité d'amélioration consisterait à créer notre propre modèle de détection. Nous pourrions ainsi détecter des éléments spécifiques tels que des personnes portant un sac ou un équipement de protection, des vélos électriques, ou tout autre élément pertinent pour effectuer des statistiques plus approfondies. Cette approche personnalisée nous donnerait davantage de contrôle sur les types d'objets détectés et nous permettrait d'adapter le modèle à nos besoins spécifiques.

Par ailleurs, nous envisageons d'implémenter une fonctionnalité permettant de définir des zones de détection spécifiques. Par exemple, il serait possible de définir des zones de détection au niveau des noms de rues, ce qui nous fournirait des données encore plus précises sur les lieux où les utilisateurs de vélos se déplacent le

plus. Cette amélioration nous permettrait de nous concentrer sur des éléments plus spécifiques que de simples indications géographiques, nous offrant ainsi une vision plus détaillée des habitudes de déplacement.

Pour faciliter l'installation sur un Raspberry Pi, il serait idéal de créer une image Docker qui installerait automatiquement toutes les dépendances nécessaires et utiliserait la version appropriée de Python. Cela permettrait également d'utiliser plusieurs bornes simultanément en utilisant le programme sur plusieurs Raspberry Pi.

Enfin, nous cherchons à exploiter les avancées en matière d'intelligence artificielle, notamment l'utilisation de Grounding Dino ([2303.05499] Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection (arxiv.org)). Avec cette avancée, nous pourrons explorer la possibilité de détecter certains éléments à partir de phrases. Par exemple, nous pourrions détecter toutes les personnes portant un gilet de sécurité jaune. Cette approche ouvrirait un champ de détection illimité, nous permettant d'effectuer des statistiques plus approfondies et d'obtenir des informations plus précises sur les comportements des utilisateurs de vélos.

En conclusion, en intégrant YOLOV8, en développant notre propre modèle de détection, en permettant la détection spécifique dans des zones prédéfinies, en réalisant une image Docker et en exploitant les avancées récentes en matière d'intelligence artificielle, nous pourrions considérablement améliorer notre système en termes de qualité de détection, de précision des données et d'extension des fonctionnalités. Ces développements futurs contribueraient à renforcer l'efficacité et la pertinence de notre projet, en offrant de nouvelles perspectives pour des analyses approfondies et des prises de décision éclairées dans le domaine de la mobilité urbaine.

Conclusion

Bilan du projet

Le projet visait à développer une borne de compteur cycliste miniature et autonome pour évaluer l'impact des évolutions des infrastructures sur le trafic cycliste. L'objectif était de recueillir différentes informations pour chaque cycliste, notamment l'heure de passage, le sens de circulation, la charge du vélo et le port d'équipements de protection. Pour atteindre cet objectif, plusieurs étapes ont été nécessaires, notamment une étude comparative des logiciels open source de reconnaissance vidéo, la création d'un prototype sur Raspberry Pi et une analyse approfondie de la légalité et de l'éthique des données collectées. Le projet a adopté une approche basée sur des cycles de développement itératifs avec des présentations régulières à notre tuteur. Les réalisations et les résultats obtenus ne sont pas explicitement mentionnés dans le contexte actuel de la page web.

Récapitulation des objectifs atteints

Notre système de détection d'objets répond à plusieurs exigences exprimées dans l'analyse des besoins. Il détecte avec précision les vélos dans les images ou vidéos et indique leur direction. Toutefois, il présente aussi des limites. Par exemple, il ne reconnaît pas certaines informations spécifiques, comme la charge d'un vélo ou le port d'un casque. De plus, il n'est pas facile de le lancer sur un Raspberry Pi.

En conclusion, notre système satisfait plusieurs critères identifiés dans l'analyse des besoins, mais il a aussi des faiblesses. Ses atouts sont sa capacité à détecter les vélos et à donner leur direction, tandis que ses défauts sont son incapacité à repérer des éléments sur les vélos et sa difficulté d'installation sur des appareils portables.

Contribution et impact du projet

Le projet a pour but de contribuer à la communauté scientifique et aux domaines d'application spécifiques en développant une borne de compteur cycliste miniature et autonome pour évaluer l'impact des évolutions des infrastructures sur le

trafic cycliste. L'impact potentiel du projet pourrait être une meilleure compréhension de l'impact du trafic cycliste, l'optimisation des infrastructures existantes et la promotion de l'utilisation du vélo. De plus, en adoptant une approche open source, le projet espère favoriser la collaboration et la contribution de la communauté, afin d'améliorer continuellement la solution et d'encourager l'innovation dans le domaine de la mobilité cycliste.

Webographie

- Datakeen, article expliquant en langage humain le deep Learning : www.datakeen.co
- CNIL, déploiement de caméras augmentées dans les espaces publics : CNIL
- SORT, algorithme simple de suivi en ligne et en temps réel pour le suivi d'objets multiples en 2D dans des séquences vidéo utilisé dans une première version de notre programme : abewley/sort
- YOLOV5, projet open source utilisé pour détecter les objets dans les vidéos : ultralytics/yolov5
- Norfair, Bibliothèque permettant le suivi multi-objets en temps réel utilisé dans notre programme : tryolabs/norfair
- GroundingDINO, projet Github sur l'utilisation de l'intelligence artificielle pour améliorer notre projet : IDEA-Research/GroundingDINO

Table des illustrations

Lexique

- **ChartJS :** Bibliothèque JavaScript pour la visualisation de données
- **CNN :** Un réseau de neurones convolutifs est un type de réseau de neurones artificiels utilisé dans l'apprentissage en profondeur pour traiter des données structurées en grille, telles que des images.
- **Django :** Framework web en Python
- **Image Docker :** Une image Docker est un fichier léger, autonome et exécutable qui contient tout ce qui est nécessaire pour exécuter une application, y compris le code, les bibliothèques, les variables d'environnement et les fichiers de configuration. Elle peut être utilisée pour créer des conteneurs Docker.
- **IOT :** L'Internet des objets désigne l'interconnexion d'objets physiques, de véhicules, de bâtiments et d'autres éléments intégrant des capteurs, des logiciels et des technologies de réseau pour collecter et échanger des données.
- **ORB :** Oriented FAST and Rotated BRIEF est un algorithme de vision par ordinateur pour détecter et décrire des caractéristiques locales dans les images. Il combine les détecteurs de points d'intérêt FAST et les descripteurs binaires BRIEF pour créer un algorithme rapide et robuste.
- **PyTorch :** PyTorch est une bibliothèque d'apprentissage automatique en Python développée principalement par Facebook's AI Research lab (FAIR) pour faciliter la création de modèles d'apprentissage en profondeur.
- **Raspberry Pi :** Le Raspberry Pi est un ordinateur monocarte à processeur ARM conçu pour être petit, peu coûteux et facile à utiliser pour l'apprentissage de la programmation et l'électronique.
- **RNN :** Réseau neuronal récurrent est un type de réseau de neurones artificiels conçu pour traiter des séquences de données en utilisant des boucles de rétroaction pour transmettre des informations d'une étape à l'autre.
- **SIFT :** Scale-Invariant Feature Transform est un algorithme de vision par ordinateur pour détecter et décrire des caractéristiques locales dans les images.

- **SSD :** Single Shot MultiBox Detector est un algorithme de détection d'objets en temps réel qui utilise un réseau de neurones convolutifs pour détecter et classifier les objets dans une image en une seule passe.
- **SURF :** Speeded Up Robust Features est un algorithme de vision par ordinateur pour détecter et décrire des caractéristiques locales dans les images, similaire à SIFT mais plus rapide à calculer.
- **VAE :** Vélo à assistance électrique
- **YOLO :** You Only Look Once est un algorithme de détection d'objets en temps réel qui utilise un réseau de neurones convolutifs pour détecter et classifier les objets dans une image en une seule passe.

Annexes

Exemple de fichier CSV

Fichier CSV généré à partir de 5 vidéos d'une durée totale de 1h21:

```
date,occurrence,top-left,top-right,bottom-left,bottom-right,classe
2023-05-30 11:09:00,2,0,0,0,2,1
2023-05-30 11:10:00,1,0,0,0,0,1
2023-05-30 11:11:00,4,1,1,0,2,1
2023-05-30 11:12:00,5,0,1,1,3,1
2023-05-30 11:13:00,3,0,2,0,1,1
2023-05-30 11:14:00,4,0,1,0,3,1
2023-05-30 11:15:00,5,0,0,1,4,1
2023-05-30 11:16:00,7,0,1,0,5,1
2023-05-30 11:17:00,4,0,1,1,2,1
2023-05-30 11:19:00,4,0,0,0,3,1
2023-05-30 11:20:00,2,0,0,0,2,1
2023-05-30 11:21:00,2,0,0,0,2,1
2023-05-30 11:22:00,2,0,0,0,1,1
2023-05-30 11:23:00,4,0,0,0,2,1
2023-05-30 11:24:00,4,0,1,0,1,1
2023-05-30 11:25:00,4,0,1,1,0,1
2023-05-30 11:26:00,2,2,0,0,0,1
2023-05-30 11:27:00,7,2,2,0,3,1
2023-05-30 11:28:00,2,1,1,0,0,1
2023-05-30 11:29:00,9,0,4,0,5,1
2023-05-30 11:30:00,6,0,4,0,2,1
2023-05-30 11:31:00,6,2,0,1,2,1
2023-05-30 11:32:00,3,0,0,0,2,1
2023-05-30 11:33:00,7,0,2,0,3,1
2023-05-30 11:34:00,6,1,1,0,3,1
2023-05-30 11:35:00,8,0,1,1,4,1
2023-05-30 11:36:00,5,0,2,0,1,1
2023-05-30 11:37:00,4,0,1,0,1,1
2023-05-30 11:38:00,3,0,1,0,2,1
2023-05-30 11:39:00,4,0,0,0,4,1
2023-05-30 11:40:00,6,0,0,2,4,1
2023-05-30 11:41:00,5,1,0,1,3,1
2023-05-30 11:42:00,5,0,1,0,4,1
2023-05-30 11:43:00,7,0,2,0,4,1
2023-05-30 11:43:36,5,1,1,0,2,1
2023-05-30 11:46:31,3,0,1,0,2,1
2023-05-30 11:47:31,1,0,0,0,1,1
2023-05-30 11:48:31,1,0,0,1,0,1
2023-05-30 11:49:31,1,0,0,1,0,1
2023-05-30 11:50:31,3,1,1,0,1,1
2023-05-30 11:52:31,1,1,0,0,0,1
2023-05-30 11:53:31,3,1,0,0,1,1
2023-05-30 11:54:31,2,0,1,0,1,1
```

```
2023-05-30 11:57:31,1,1,0,0,0,1
2023-05-30 12:00:31,2,0,0,2,0,1
2023-05-30 12:01:31,2,1,0,1,0,1
2023-05-30 12:02:31,1,1,0,0,0,1
```

Fichier de configuration personnalisé du programme

Voici un exemple de fichier de configuration personnalisé nommé “customconfig.ini”

```
[PARAMS]
source = video3.mp4
classes = 0,1
interval = 0
display_detection = True
display_fps = True
debug = False
save_in_csv = True

[YOLOV5_PARAMS]
weights = yolov5s
conf_thres = 0.45
iou_thres = 0.45
agnostic_nms = False
multi_label_nms = True
max_det = 50
amp = True
output_folder = OUTPUT
csv_name = data.csv
device = 0

[BDD_PARAMS]
save_in_bdd = False
bdd_name = detect_save.db
table_name = detect
time_to_save = 00:00:00
keep_csv = False
```