Object tracking and detection with YOLOv8 and StrongSORT algorithms captured by drone

Farkaš, Luka

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, Faculty of Science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:238937

University of Split
Faculty of Science

GRADUATE THESIS

OBJECT TRACKING AND DETECTION WITH YOLOV8 AND STRONGSORT ALGORITHMS CAPTURED BY DRONE

Luka Farkaš

Split, August 2023.

Basic documentation card

Graduate thesis

University of Split

Faculty of Science

Department of Computer Science

Ruđera Boškovića 33, 21000 Split, Croatia

Object tracking and detection with YOLOv8 and StrongSORT algorithms captured by drone

Luka Farkaš

Abstract

The popularity of computer vision tasks inspires the community to develop diverse object detection and multiple object tracking algorithms. This work is mainly based on the newest technologies in object detection (YOLOv8) and multiple object tracking (StrongSORT). The evolution of YOLOv8 and StrongSORT is reviewed, and algorithms are implemented into a unified model that detects and tracks objects in real-time. To enhance this work, we proposed EL-YOLOv8 to improve object detection on the VisDrone dataset which is a demanding object detection dataset for crowded and small object scenarios. The thesis also contains detailed dataset methodology, training approach, results, and model developed in Python.

Keywords: Object detection, object tracking, YOLOv8, StrongSORT, VisDrone, small objects

Graduate thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 39 pages, 27 figures, 5 tables and 70 references

Original language: English

Mentor: **Saša Mladenović Ph.D.** *Full Professor, Faculty of Science, University of Split*

Reviewers: **Saša Mladenović Ph.D.** *Full Professor, Faculty of Science, University of Split*

Goran Zaharija, Ph.D. *Assistant Professor, Faculty of Science, University of Split*

Dino Nejašmić, *Lecturer, Faculty of Science, University of Split*

Thesis accepted: **August 2023**

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Croatia

Praćenje i detekcija objekata pomoću YOLOv8 i StrongSORT algoritama snimanih dronom

Luka Farkaš

Sažetak

Popularnost zadataka računalnog vida nadahnjuje zajednicu da razvijaju raznovrsne algoritme za detekciju i praćenje objekata. Ovaj rad je temeljen na najnovijim tehnologijama u području detekcije objekata (YOLOv8) i praćenju objekata (StrongSORT). Evolucija YOLOv8 i StrongSORT-a je istražena i algoritmi su implementirani u jedinstven model za koji detektira i prati objekte u realnom vremenu. Kako bi unaprijedili ovaj rad, predložili smo EL-YOLOv8 za poboljšanje detekcije na VisDrone skupu podataka koji je zahtjevan skup podataka za detekciju objekata u scenarijima gužve i detekcije malih objekata. Ovaj rad sadrži i detaljnu metodologiju na skupu podataka, pristup uvježbavanju, rezultate i model razvijen u Pythonu.

Ključne riječi: detekcija objekata, praćenje objekata, YOLOv8, StrongSORT, VisDrone, objekti malih dimenzija

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 39 stranica, 27 grafičkih prikaza, 5 tablica i 70 literaturnih navoda

Izvornik je na engleskom jeziku.

Mentor: **Dr. sc. Saša Mladenović**, *redoviti profesor, Prirodoslovno-matematički fakultet, Sveučilište u Splitu*

Ocjenjivači: **Dr. sc. Saša Mladenović**, *redoviti profesor, Prirodoslovno-matematički fakultet, Sveučilište u Splitu*

Dr. sc. Goran Zaharija, *docent, Prirodoslovno-matematički fakultet, Sveučilište u Splitu*

Dino Nejašmić, *predavač, Prirodoslovno-matematički fakultet, Sveučilište u Splitu*

Rad je prihvaćen: **kolovoz 2023**

Contents

Declaration

of the independent preparation of the graduate thesis

I declare under full material and moral responsibility that I have independently created this work and that it does not contain copied or duplicated parts of text from the works of others, unless appropriately marked as quotations with the specified source from which they were taken.

In Split, 25.08.2023. Luka Farkaš

(student)

Introduction

Computer vision tasks, such as object detection and multiple object tracking rapidly grown in recent years. Driven by advancements in deep learning, increased computational power, and the growing field of application domains, these tasks have evolved into dynamic areas of study and innovations. Thanks to the open-source community, we have an opportunity to test algorithms and enhance their performance, both in general and for specific purposes. This thesis relies on multiple object detection and tracking with an emphasis on real-time processing. Drone imagery is also a fast-growing field in recent years, and it opens opportunities to automate and speed up tasks that are demanding for people. By combining computer vision algorithms and the possibilities of drone cameras, we take a step further toward this goal. According to this, we reviewed the relevant literature and chose algorithms to perform real-time multiple object detection and tracking. We selected the VisDrone dataset, known for its complexity and demanding characteristics, to thoroughly evaluate the effectiveness of our chosen algorithms in tackling the problems of small object detection and tracking. Chosen algorithms and dataset are additionally reviewed in the section Technical overview of the selected algorithms and dataset. To improve small object detection, we proposed a new model based on YOLOv8. Additionally, we describe our dataset methodology, training approach, and results. As a result of this thesis, we build the real-time multiple object tracking model that is publicly available in [66]. The main idea of the proposed model is to utilize it as a backbone for real-time multiple object tracking applications.

1. Literature review

A search strategy of the literature was conducted to identify all previously published literature review studies reporting on object detection and multiple object tracking, following the recommendations of preferred reporting items for systematic reviews and meta-analyses (PRISMA) [1]. Since the research resulted in studies published in 2022. and 2021. with a total of 238 citations and 183 citations respectively, citations were analyzed following the inclusion and exclusion criteria written in Table 1.1:

Exclusion criteria	**Inclusion criteria**
Traditional methods and algorithms	Deep learning-based approach
Not publicly available papers	State-of-the-art algorithms
Publications before 2014.	Algorithms and techniques
Algorithms for less common purposes	Real-time processing
/	Performance evaluation and comparison

Table 1.1 Exclusion and inclusion criteria for literature review based on relevant articles.

The following section reviews the literature for object detection and multiple object tracking (MOT) according to state-of-the-art approaches and techniques. This thesis is based on deep learning detection and tracking, so the literature review includes deep learning-based detection and tracking. Single object tracking is not considered as a separate problem but MOT includes single object tracking when only one object is on scene. This modular literature review approach provides a better insight into the algorithms and leads to simple reproductivity with the possibility of expanding the review for a specific purpose. We select relevant articles for the mentioned tasks and follow their literature as a backbone to review state-of-the-art object detection and MOT.

1.1. Object detection

Object detection is the most crucial and challenging computer vision task which has gained a rapidly increasing amount of attention in recent years. It is the basis for many other computer vision tasks such as image segmentation, pose estimation, object tracking, etc.

Zhengxia Zou et al. proposed “Object Detection in 20 Years: A Survey” [2], which reviews the light technical evolution, key technologies, and recent state-of-the-arts of object detection from the 1990s to 2022. A comprehensive analysis of detection speed-up techniques is also obtained in the survey. This section reviews state-of-the-art deep object detectors. Traditional object detection methods are not included. Convolutional neural networks (CNNs) showed the best performance for computer vision tasks, especially object detection. Processing spatial relationships in images and extracting feature maps leads to accurate detection and classification. Accordingly, CNN is the main architecture in which we are interested.

The year 2014 is a milestone in CNN based on two-stage detectors at the occurrence of region-based CNN (R-CNN). Ross Girshick et al. presented “Rich feature hierarchies for accurate object detection and semantic segmentation” [3] that outperform competitors on VOC 2012 dataset [4]. R-CNN is divided into three modules:

1. Region proposals: extracting category-independent regions that contain a set of candidate detections.
2. Feature extraction: extracts 4096-dimensional feature vector from proposed regions. This is accomplished by CNN with five convolutional and two fully connected layers. The input is a 227x227 RGB image.
3. Classification: class-specific linear support vector machine algorithms are used to classify each region. Finally, greedy non-maximum suppression fixes overlap with undesirable Intersection over union (IoU) scoring.

Figure 1.1.1 describes the above-mentioned R-CNN modules.

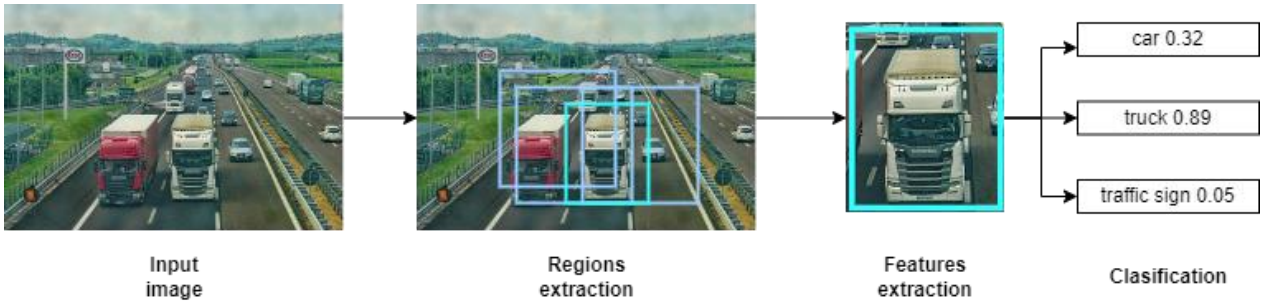

Figure 1.1.1 Region-based object detection (two-stage object detection).

Since then, object detection has started to evolve rapidly. A high number of overlapped proposals (over 2000 boxes) causes extremely slow detection. In a spatial pyramid pooling network (SPPNet), He et al. [5] approach that problem by eliminating the fixed-size image

input. Feature maps are only once computed from the entire image. Furthermore, features are pooled and prepared for training the detectors. This procedure is 20x faster than the R-CNN approach with the same mean average precision (mAP) score. Fast R-CNN [6] comes with recent innovations which speed up training and detection with significantly increasing detection accuracy. Fast R-CNN utilizes very deep convolutional neural network (VGG16) [7]. The faster R-CNN [8] finally presents a unified network that proposes regions and performs detections. The proposed network is called the region proposal network (RPN). RPN takes an image of any size as an input and outputs a set of proposals by utilizing a sliding window approach. Each sliding window predicts region proposals mapped into a fully connected layer. The fully connected layer is divided into regression and classification layers. Moreover, anchor boxes are situated in region proposals in different scales and aspect ratios. The described two-stage paradigm fixes the speed bottleneck of Fast R-CNN and achieves state-of-the-art detection accuracy and speed using 300 proposals per image. The mentioned CNN-based models utilize only the network's top-layer features without considering the deeper layers. Therefore, Lin et al. (2017) proposed "Feature Pyramid Networks for Object Detection" (FPN) [9]. FPN is a top-down architecture with lateral connections at all scales. It outputs feature maps at multiple levels using a fully convolutional approach. Figure 1.1.2 illustrates the comparison between vanilla CNN and the idea of the FPN approach for object detection. Faster R-CNN achieves state-of-the-art by utilizing FPN as a building block.

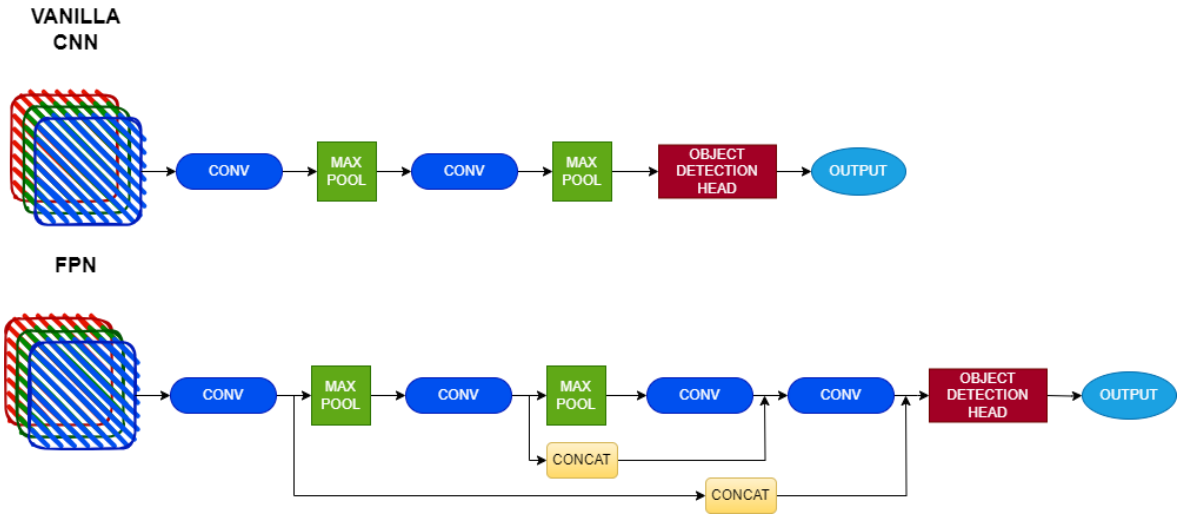

Figure 1.1.2 Vanilla CNN and FPN for object detection

Feature pyramid representations were previously avoided because of time consumption and memory costs. Meager speed performance and computational complexity of previously

mentioned two-stage object detectors are not desirable among engineering problems. A new milestone occurred in 2015 with the advent of one-stage CNN-based detectors. You Only Look Once (YOLO) is the first one-stage object detector proposed by Joseph Redmon et al. (2015) [10]. The main advancement in YOLO is the detection speed, which is in real-time. The fastest version of the YOLO algorithm reaches up to 155 frames per second (FPS) and 52.7% mAP on VOC07. The one-stage paradigm processes the entire image at once in a single CNN, as shown in Figure 1.1.3.

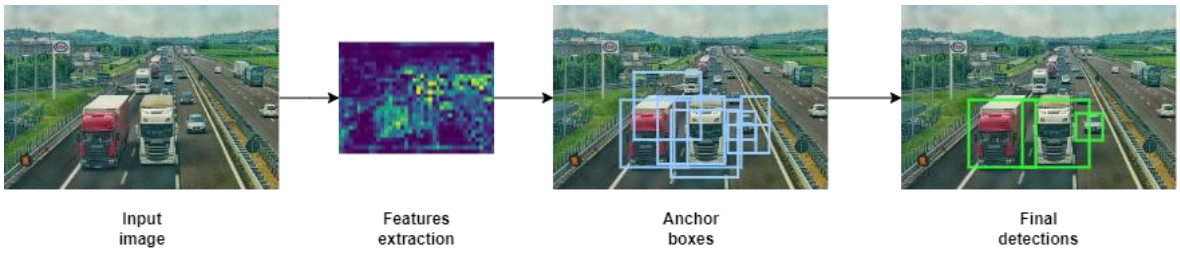

Figure 1.1.3 One-stage object detection based on the YOLO network.

Due to real-time inference, prediction accuracy drops by a few percent in contrary to two-stage object detectors. Additionally, YOLO became a popular framework for object detection and Redmon et al. propose three versions of the YOLO altogether [10, 11, 12]. The network is composed of 24 convolutional layers with max-pooling layers followed by 2 fully connected layers. The input image is resized to 448x448 and processed in one step. After Redmon dismissed the research on YOLO, the popularity of the YOLO algorithm continued. Bochkovskiy et al. (2020) [13] proposed YOLOv4 as an optimal speed and accurate object detector where recent improvements and augmentations enhance detection performance. Test results showed that the cross-stage partial network (CSPNet) [14] is an optimal backbone for the proposed object detector. YOLOv5 [15] appeared in the same year and brought new improvements with faster training and inference. G. Jocher et al. built YOLOv5 in the PyTorch framework [67] and proposed a pre-trained model on the MS COCO [16] dataset. The YOLOv5 framework is designed as an easy-to-use framework with lightweight models optimized for real-time inference with high accuracy. The popularity of the YOLO framework continued with YOLOv6 [17], YOLOv7[18], and YOLOv8 [19]. YOLOv6 is intended as an industrial single-stage object detector proposed by Chuyi Li et al. (2022). Wang, C.Y. et al. put more focus on training tips and tricks to improve detector performance in YOLOv7 [18]. To sum up the YOLO story, G. Jochner et al. proposed YOLOv8 [19], which presents an upgraded YOLOv5 with state-of-the-art object detection improvements.

A detailed comparison of YOLOv5 and YOLOv8 can be found in the section Technical overview of the chosen algorithms.

Besides the YOLO algorithms, a Single-shot multi-box detector (SSD) [20] was also presented in 2015 by Liu et al. VGG16 [7], an SSD backbone, outperforms YOLO in inference speed and accuracy on the VOC07 dataset [4]. SSD network is based on multi-scale feature maps, leading to better accuracy in small object detection scenarios. Due to the trade-off between speed and accuracy, Lin et al. (2020) suggested a one-stage object detector with an improved loss function. The proposed detector called RetinaNet [21] utilizes FPN [9] as a backbone for the extraction of rich multiscale features. Focal loss is a novel loss function that is an extension of cross-entropy loss. The proposed loss function handles large class imbalances by focusing on misclassified objects. The so far mentioned detectors mostly use anchor boxes to generate candidates for bounding boxes. Generating a huge amount of anchor boxes is time-consuming due to the different anchor box scales. YOLOv5 implements an algorithm to calculate appropriate anchor box dimensions that additionally extend training time. Hence, CornerNet [22] and CenterNet [23] advent to replace the anchor box paradigm with the points paradigm. Law G. et Dend J. proposed CornerNet, which predicts bounding boxes by top-left and bottom-right corners. Furthermore, a new pooling layer, corner pooling, is designed for corner point localization. Zhou X. et al. (2019) have gone a step further and proposed the detection of objects by single-point and then regress other objects' properties like dimension, 3D location, orientation, and pose. CenterNet achieves the best speed and accuracy tradeoff among one-stage object detectors on MS COCO [16] dataset. Unlike CornerNet, CenterNet does not require a separate stage after points estimation that combines groups of key points. The center point is predicted directly, without any post-processing. Figure 1.1.4 shows the difference between anchor-based predictions, corner-based predictions, and center-based predictions.

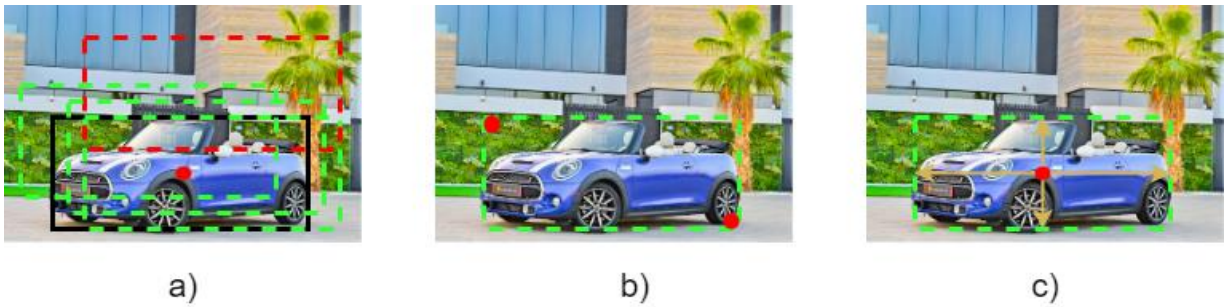

Figure 1.1.4 a) Anchor-based predictions – anchors with a sufficiently high IoU (green dashed box) are candidates according to ground truth (black box). b) Corner points bounding box prediction. c) Center-based bounding box prediction.

The key point object detection paradigm simplifies the repurpose of the object detection task to pose estimation, 3D location, etc. Detection transformer (DETR) [24] presents a new object detection approach with transformer neural network models. Transformers [25] are mainly presented as a new architecture for machine translation tasks. Carion N. et al. (2020) proposed an end-to-end object detection with transformers that combines CNN-based backbone and transformer encoder-decoder architecture to predict objects. Figure 1.1.5 illustrates previously described architecture.

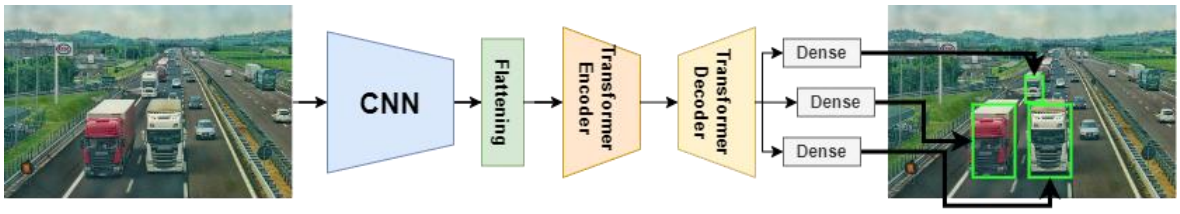

Figure 1.1.5 Transformer-based object detection.

This approach presents a new era in object detection without anchor boxes or the prediction of anchor points. Deformable DETR was proposed by X. Zhu et al. in 2020 with deformable attention modules. Test results have shown state-of-the-art performance on the MS COCO dataset [16], especially in the detection of small objects.

Having explored the realm of state-of-the-art object detection algorithms, we now transition our focus to deep learning MOT. MOT naturally follows the investigation of object detection, as the detections are the crucial input for effective object tracking.

1.2. Multiple object tracking

Object detection provides information about the location and class of an image. According to that, more complex tasks like crowd counting, distance and speed estimation, etc. are not possible to accomplish. Upgrading the computer vision task to object tracking enables object identification through frames, leading to considerably more possibilities for object measuring and manipulation. Moreover, multiple object tracking (MOT) handles missing detection through frames that complement object detection tasks. This section reviews state-of-the-art deep-based MOT based on S. Pal et al. [26] (2021) "Deep learning in multi-object detection and tracking: state of the art". This thesis is primarily focused on detection-based tracking so detection-free tracking will be excluded.

The detection-based tracking (DBT) (or tracking-by-detection) paradigm demands detections to link them into trajectories. Hence, DBT relies on object detection quality. We distinguishes between two modes of object tracking:

1. Online mode: frames are processed by only using current and previous frame data.
2. Offline mode: frames are processed by using arbitrary batches of frames.

Offline object tracking will not be considered. In recent years, deep architecture-based MOT has become a popular approach. S. Pal et al. [26] classify them into three categories:

1. Deep network features-based MOT enhancement.
2. Deep network embedding.
3. Deep network (end-to-end) learning.

Multiple hypothesis tracking (MHT) (2015) is an example of the **first category**. MHT utilizes features extracted from deep CNN and structures them into a hypothesis tree where a scoring function determines the best hypothesis for the detected object [27]. A re-identification approach is proposed in DeepSORT [28] where the Mahalonobis distance and the minimum cosine distance determine which track corresponds to detection. DeepSORT is explained in the section Technical overview of chosen algorithms. Furthermore, Siamese CNN [29] was proposed in 2016 to determine finer similarities between the tracks and detections. Two-stage feature learning with a combination of gradient boosting achieves state-of-the-art performance in pedestrian tracking scenarios. Siamese CNN-based architectures can also learn optical flow features, which are effective for improving the performance of object tracking. Figure 1.2.1 shows the MOT approach by using Siamese CNN.

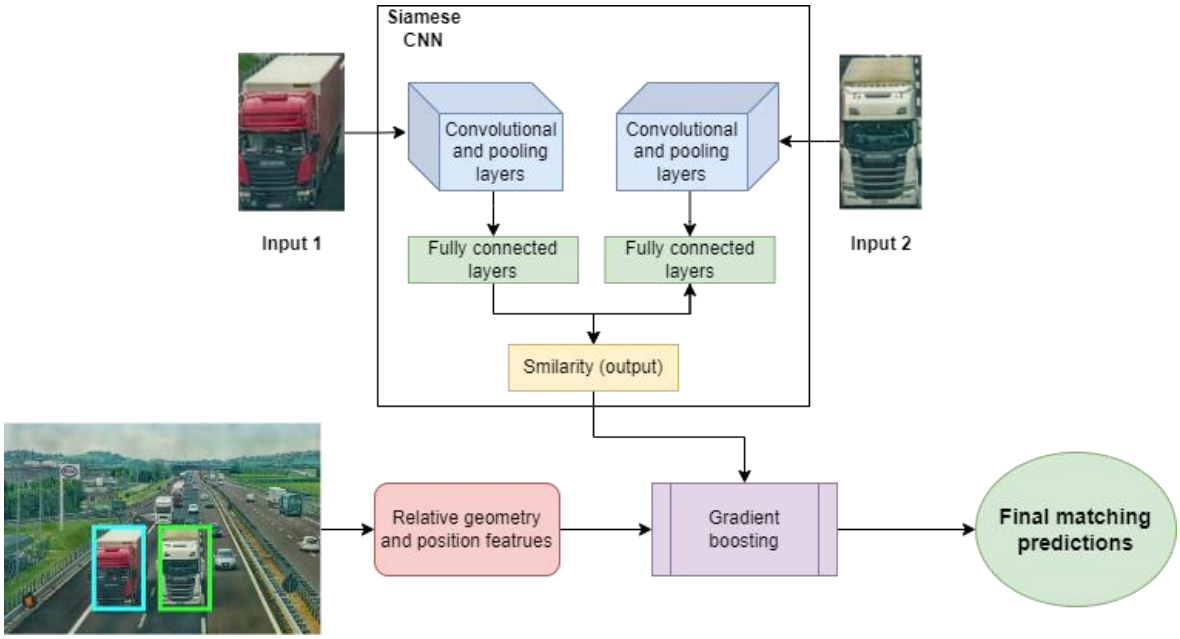

Figure 1.2.1 MOT approach with Siamese CNN.

Tang S. et al. [30] have implemented a lifted multicut framework for graph-based formulations that links person hypotheses with a re-identification to achieve state-of-the-art performance in people tracking scenarios. Lifted multicut framework introduced lifted edges whose task is to distinguish persons of similar appearance and enable long-range information. By combining lifted and regular edges, two people far apart in time who have similar appearances will be assigned as one if there exists a path along the regular edges. VGG 16-based architecture is utilized to extract different body parts and then symmetrical body parts scores are combined.

The second category contains deep CNNs as a core part of the entire tracking framework. The main task of deep CNNs is to determine whether the detections belong to the same object or not. L. Chen et al. [31] proposed a novel online MOT framework that utilizes deep CNN to differentiate categories. Different layers extract features from different levels in CNN. Top layer features provide rich information about categories, while lower layers usually distinguish instances from the same category more accurately. The proposed framework achieves state-of-the-art online tracking performance by combining category and instance classifiers through a particle filter. Firstly, the location of each object is estimated by a particle filter and appearance model for each frame. Secondly, a detection that does not overlap with the tracked object is used to initialize the new object and retrieve the missing object. Similarly to [31], Chu Q. et al. (2017) [32] proposed an MOT online tracker with a spatial-temporal attention mechanism. The mentioned tracker handles occlusion with

learned spatial features using convolution and fully connected layers. Regions of interest are shared with the classifier instead of the entire image to reduce computational complexity. Furthermore, Pal, S. K. et al. [26] focus on multi-class deep SORT [33] as a deep metric-based tracker that enhances the possibilities of deep SORT [28]. The main modification is that the association of an object with tracks is limited within the same class. Objects of the same class are obtained from the region-of-interest maps generated from G-RCNN [33]. Generally, associating detections with the track of the same class reduces computational complexity and enhances tracking accuracy. MOT with quadruplet CNNs [34] utilize Siamese and triplet networks to present quadruplet loss function. The proposed loss enforces additional constraints within temporary close-located detections unlike the ones with temporary gaps. The inputs in the proposed quadruplet CNN are four image patches and the next step is distance measuring between these four patches. CNN for appearance feature learning is based on AlexNet [35] with custom layers. A novel trajectory processing approach was proposed by Ma C. et al. [36] in 2018. Features are extracted by RNN and CNN, resulting in high-confidence tracklet candidates. Furthermore, tracklets from different objects are split by the bi-directional gated recurrent unit (GRU) into several sub-tracklets. The splitting (or cleaveining) step is illustrated in Figure 1.2.2.

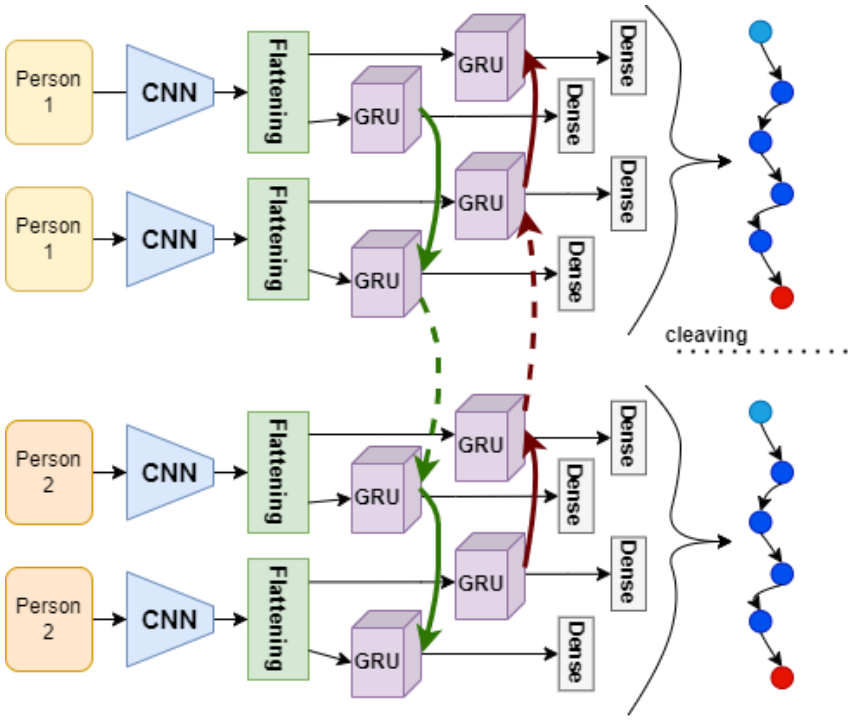

Figure 1.2.2 Cleavening step with bi-directional GRU.

Then, the Siamese GRU-based reconnection method is utilized to link previously cut sub-tracklets that belong to the same object. The described approach achieves state-of-the-art

performance by generating tracklet candidates and reconnecting them unlike previously mentioned MOT approaches, where the main problem is to link detection to tracklet. Generative learning-based MOT is utilized for parameter estimation to increase tracking performance. Fang K. et al. (2018) proposed the MOT approach by using recurrent autoregressive networks (RAN) [37]. The primary goal of RAN is to memorize the characteristics of the tracked object. Multiple RANs are utilized to compute association metrics between tracked objects and generated candidate detections with respect to previously calculated motion and appearance features. GRU is embedded into RAN to estimate recurrent parameters. Describing each object with a RAN that combines internal and external memory, Fang K. et al. achieve state-of-the-art MOT performance. A novel generative-based deep MOT is introduced in [70]. Fernando T. et al. proposed a lightweight sequential generative adversarial network architecture for person localization. The first step is to generate probability maps that classify the likelihood of each pixel from the input frame. Then, the watershed segmentation technique [38] is utilized to segment out each connected component. The next step is to predict the short-term and long-term trajectories. Short-term trajectories are used for data association while long-term predictions update objects to handle occlusions during the rendering of trajectories. The described long short-term memory (LSTM) approach replaces the computationally expensive re-identification step and outperforms deep learning-based approaches.

The **third category** includes end-to-end deep learning approaches where all stages are included in a single framework. "In 2016, Milan A. et al. presented a fully deep learning-based framework for Multiple Object Tracking (MOT) using Recurrent Neural Networks (RNN) [39]." The main problem is divided into two blocks: a state prediction and update block and a data association block. The first block is RNN based, capable of learning a dynamic model of targets, correcting state distribution, and learning to identify or terminate tracks based on state. The second block is LSTM based, and it calculates a matching matrix between tracklets and detections. The limitations of the mentioned end-to-end deep-base MOT are the lack of training images for the proposed model, only motion information being taken into consideration, and tracklet initialization and termination which do not consider context information. Therefore, Sadeghian A. et al. (2017) proposed “Tracking The Untrackable” [40]. The proposed approach utilizes recurrent neural networks (RNNs) to combine multiple clues for predicting long-term features. The RNN relies on appearance, motion, and interaction feature extractors whose outputs are combined through another

RNN. The combined features are fed into a fully connected layer where the softmax classifier determines whether the detections correspond to the tracklets. The mentioned MOT tracking approach is presented in Figure 1.2.3.

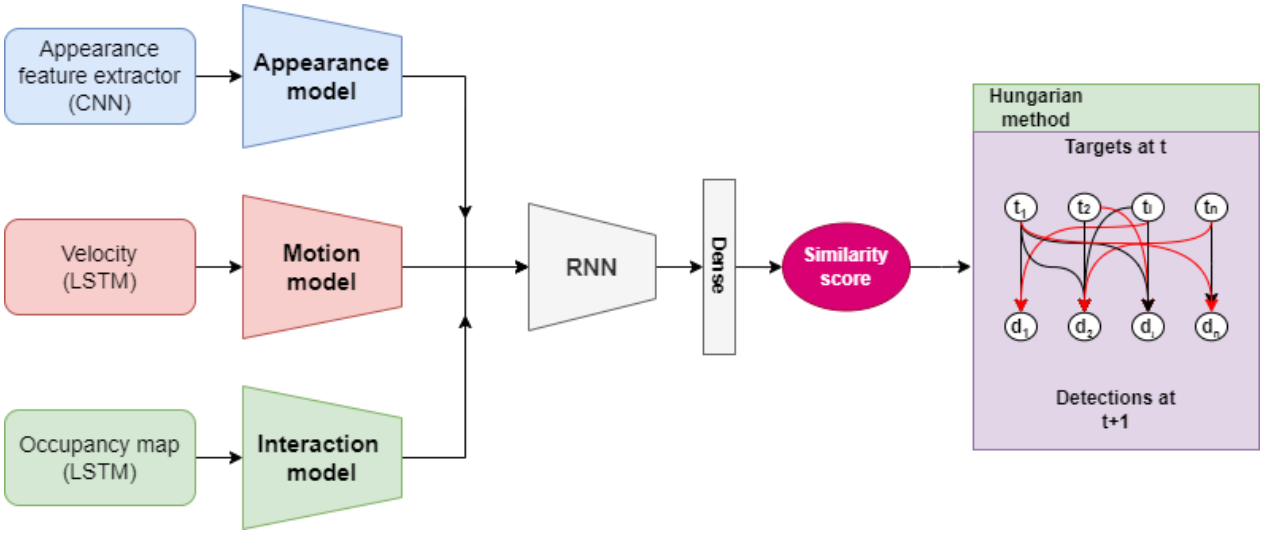

Figure 1.2.3 MOT with three feature extractors.

In [41], the authors offer insights into the performance of the appearance feature descriptor concerning the re-identification task. Kim C. et al. (2018) proposed an improved version of [40] by utilizing bilinear LSTM. Unlike in [40], gating networks are introduced as input feature extractors [41]. Gating is a type of selection where existing track proposals are not updated with all of the newly arrived detections. Motion gating utilizes vanilla LSTM for storing sequence information in a hidden state in which output is further processed into a fully connected layer. Appearance gating utilizes LSTM, whose hidden state becomes a weight factor for the appearance model of the specific object. The third model is a combination of motion and appearance gating that additionally enhances the process. Schulter S. et al. (2019) presented a solution to improve the association problem with the network flows for MOT [42]. The solution is presented by a directed graph where each detection is represented by two nodes and the edge between them. Additionally, in order to initialize and terminate tracklets, a source node and a second node are introduced for each tracklet. Each variable in the network is associated with a cost - the most crucial part which interacts between initialization, existence, termination, and the association of detections.

As we move forward, our study introduces the VisDrone [43] dataset to measure the performance of the selected algorithms with emphasis on small objects.

1.3. The dataset

Small object detection is the most challenging task in object detection. Due to the dimensions of small objects, which are additionally resized for real-time processing, CNNs have problems extracting these small-sized features. VisDrone is a dataset just like that, recorded in crowded areas with a high number of instances per image. The dataset is taken from the drone, which additionally aggravates the situation due to its moving camera. Drone imagery could become helpful for various problems like traffic supervision, monitoring of hard-to-reach areas, finding lost people in mountain areas, etc. We consider the VisDrone dataset as a convenient one for traffic supervision and hope that real-time object tracking with drones can open a lot of opportunities to improve, at least, traffic quality. When we are talking about object detection datasets, Microsoft Common Objects in Context (MS COCO) [16] must be mentioned as a standard to evaluate the performance of the object detection algorithm. YOLOv8 models are publicly available and pre-trained on this dataset. Due to similarities between classes in VisDrone and MS COCO, we will take into consideration pre-trained models. The following section reviews the MS COCO and pre-trained models.

1.3.1. Pre-trained checkpoints – MS COCO dataset

The proposed Ultralytics YOLOv8 models are pre-trained on the MS COCO [16]. Pre-trained models enable faster training on custom datasets with already learned features. The MS COCO dataset contains 80 classes, including vehicles, people, animals, everyday objects, etc. With more than 330,000 images and 1.5 million object instances, MS COCO is the standard for performance evaluation on object detection algorithms. Tasks such as object segmentation and key point detection are also included in the dataset [16]. The SUN [44] and ImageNet [45] datasets have more categories, while MS COCO contains more instances per category, which authors hypothesize as being more useful for learning complex models with precise location determination. The main emphasis of the MS COCO dataset is on objects in real contexts. The PASCAL VOC [46] and ImageNet [45] have less than 2 categories and 3 instances per image; MS COCO has on average 3.5 categories and 7.7 instances per image. Such distribution of instances demands more robust models for rich feature extraction and localization [16]. The following pictures in Figure 1.3.1.1 present ground truth (green bounding boxes) and predictions on the YOLOv8x pre-trained model (colored bounding boxes with class and confidence scores).

Figure 1.3.1.1 Comparison between ground truth and YOLOv8x predictions (pre-trained on MS COCO) on arbitrarily selected pictures from the VisDroneDET-2019 validation set.

Predictions from the extra-large model provide acceptable results in vehicle prediction. However, in situations where objects are smaller and further away from the camera, the results are disappointing. MS COCO pre-trained weights provide satisfactory detections on traffic cameras for detecting vehicles. Hence, images and videos captured by drones demand additional training data due to camera position and angle.

In this literature review, we have journeyed through the evolution of the state-of-the-art object detector and multiple object trackers. Among the diverse algorithms of choice, we have chosen to explore the capabilities of YOLOv8 and StrongSORT. Both selected algorithms have evolved and been refined over time, representing the recent advancements in the computer vision world. In the technical overview of the chosen algorithms, we delve into the technical aspects of these two chosen components. Additionally, the section reviews the VisDrone dataset that is selected to evaluate the efficiency of the selected algorithms and YOLOv8 pre-trained weights on the MS COCO dataset.

2. Technical overview of chosen algorithms and dataset

The previous section reviews the literature of deep learning-based object detectors and trackers with a brief review of the technical aspects of each detector or tracker. Accordingly, we additionally review the main methods that are utilized in this work with emphasis on technical aspects. The following sections review YOLOv8 [19] by comparing it with YOLOv5 and the evolution of SORT algorithms through DeepSORT and StrongSORT. Additionally, to improve small object detection, we proposed a modified YOLOv8 model to achieve better performance in small object detection scenarios. This section also includes a technical overview of the VisDrone dataset and pre-trained YOLOv8 checkpoints.

2.1. Object detection

YOLO framework is popular and widespread since its presence in 2015. Fast and accurate detections in one-stage bring opportunities to fine-tune models and utilize them in various aspects of object detection problems. YOLOv8 as a currently last version of the YOLO framework achieves state-of-the-art performance on the MS COCO dataset in terms of speed and accuracy. The biggest advantage of the YOLOv8 is a rich framework that enables faster workflow. Augmentations, hyperparameter tuning, automated dataset annotation, fast training, simple installation and reproductivity, exports to other network formats, etc. are features that stand out YOLOv8 from the other object detection frameworks. YOLOv5 also contains the most of mentioned features but architectural improvements on YOLOv8 lead to more accurate detections. Hence, we chose YOLOv8 for this thesis although it is still in development.

2.1.1. YOLOv8

YOLOv8 (You Only Look Once version eight) is the eighth version of the YOLO algorithm presented by Ultralytics. It comes in five different models, from nano to extra-large, as well as YOLOv5. However, YOLOv8 supports object detection, image classification, instance segmentation, and pose estimation, which is not supported in YOLOv5. The advantage of the YOLOv8 is that it comes under the Ultralytics package where YOLOv3, v5, and v8 are

supported, while v8 is the only version that supports all four computer vision tasks. YOLOv8 consists of a backbone and head. The backbone is built from cross-stage partial (CSP) blocks, similar to blocks in v5. The idea of the CSP blocks is to reduce computation by combining features positioned at the beginning of the CSP block with features at the end of the block. V5 and v8 models are popular for being accurate and real-time object detection models, which CSP backbone is responsible for. The CSP blocks:

1. Strengthen the learning capability of the lightweight YOLOv8 models.
2. Remove computational bottlenecks which result in faster inference. Tests on YOLOv3-based models show that CSP networks reduce 80% of computational bottlenecks.
3. CSP architecture also reduces the utilization of RAM, which enables the utilization of small CNN models on edge devices like Nvidia Jetson and similar [14].

The above mentioned advantages of the CSP backbone in v5 and v8 models make them popular and widespread in computer vision tasks. The following comparison is mainly related to the official YOLOv5 (7.0 version) [15] and Ultralytics (YOLOv8 – 8.0.106 version) [19] GitHub repositories.

2.1.2. CSP backbone

YOLOv5 utilizes a C3 bottleneck with 3 convolutional layers similar to YOLOv8 C2f bottleneck with 2 convolutional layers. Convolutional layers are identical with sigmoid linear units (SiLU), also called the swish activation. Despite YOLOv8 utilizing CSP blocks with one less convolutional layer, it also has fewer CSP blocks in the stage. There is also a split function that divides the tensor after convolution in the C2f block into 2 chunks. One chunk is sent through the bottleneck, while the other just waits for concatenation with the first one after the bottleneck. Comparison of the CSP blocks is in Figure 2.1.2.1.

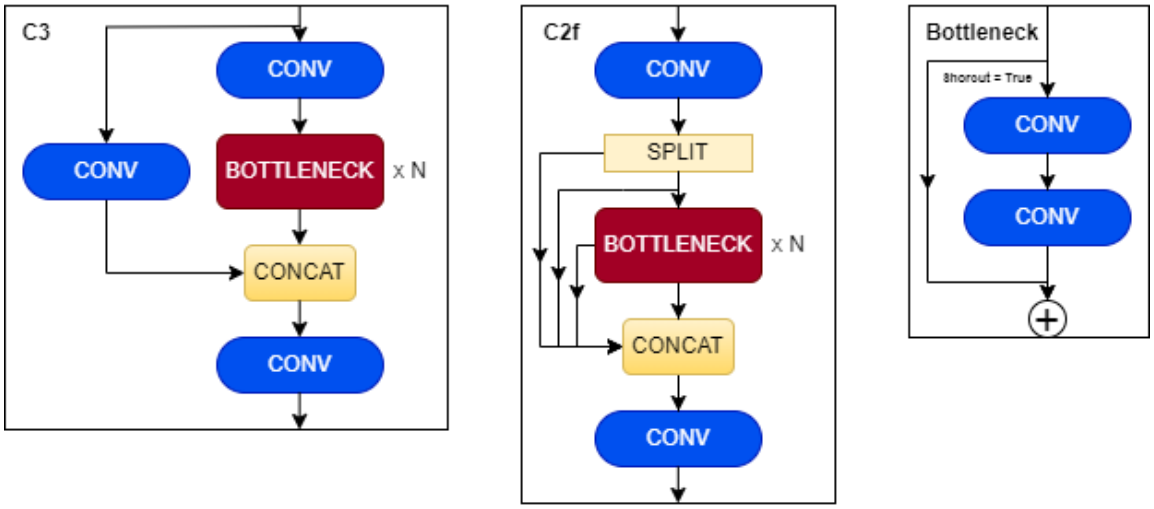

Figure 2.1.2.1 C2f and C3 building CSP blocks of the YOLOv8 and YOLOv5.

YOLOv8 has 3,6,6,3 repeats of the CSP blocks in contrary to 3,6,9,3 in YOLOv5. These stages present different levels of features called path aggregation networks (PANet) [47]. Furthermore, the last part of the backbone is the spatial pyramid pooling fast (SPPF), a modified version of the spatial pyramid pooling network (SPP). The idea of SPP is to compute feature maps from the entire image in one step and then crate the pooling step. This method outperforms the R-CNN method on Pascal VOC 2007 [48]. SPPF is a faster version of SPP due to fewer kernel sizes and fewer hidden channels. Both YOLOv5 and YOLOv8 are using the SPPF module as proposed by Glenn Jocher, CEO of Ultralytics. Figure 2.1.2.2 illustrates the SPPF module.

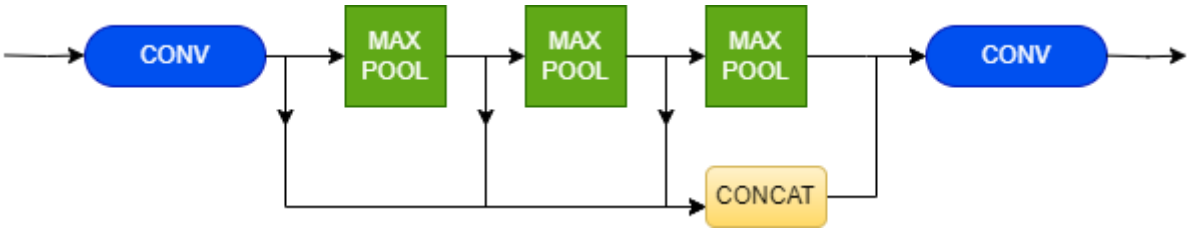

Figure 2.1.2.2 SPFF module.

2.1.3. Head

The first layer of the head in the YOLOv8 YAML () configuration file is the upsampling, unlike YOLOv5 which has another convolutional layer before upsampling in the first two stages of the head. The YOLOv8 head utilizes an anchor-free approach which directly predicts the center of the bounding box. Anchor-free detections reduce the number of generated prediction boxes that speed up the non-maximum suppression algorithm [49].

Objectness loss is no more part of the head in YOLOv8. The head consists of regression and classification branches with the distributed focal loss (DFL) for regression (localization) and binary cross-entropy (BCE) loss for classification [50]. DFL is intended for one-stage object detectors with individual branches for localization and classification to improve detection performance. DFL pushes the network to focus on the values near the ground truth label by modeling box locations as general distributions [51]. Furthermore, the YOLOv8 head utilizes a task-aligned assigner that combines both classification and localization information to assign ground truth objects to predicted bounding boxes [50, 52].

2.1.4. The proposed model

The proposed model is based on the efficient-lightweight YOLO (EL-YOLO) [53] architecture invented to improve the detection of small objects. Hu M. et al. (2023) designed three models based on YOLOv5. These models have modified SPPF, called efficient SPP (ESPP), and a new loss function, called an alpha-complete intersection over the union. Figure 2.1.4.1 illustrates the ESPP module.

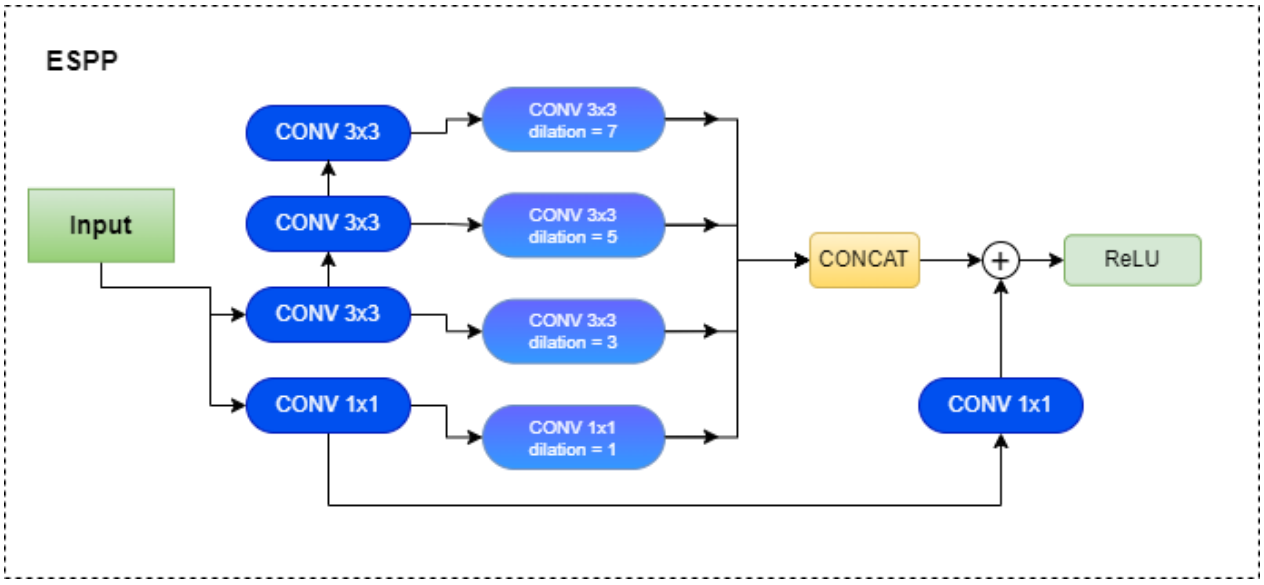

Figure 2.1.4.1 ESSP module proposed in EL-YOLO.

EL-YOLO achieves better performance on VisDrone than basic YOLOv5. The main idea is to give a chance to initial features that could be lost in deeper stages. Generally, YOLOv8 takes P3, P4, and P5 pyramidal stages into further upsampling and concatenation. P1 and P2 as initial feature stages are not taken into further concatenation. VisDrone contains high-resolution pictures (1920x1080) that are resized to 640x640 before the training step. Hence,

features from small objects can easily disappear. By including these pyramidal stages in further computation, the chance to keep small object features is increased. Our model based on the EL-YOLO contains the ESPP and EL-YOLO backbone. The proposed loss function is not implemented in this architecture. The EL-YOLOv8 (ours) architecture is presented in Figure 2.1.4.2.

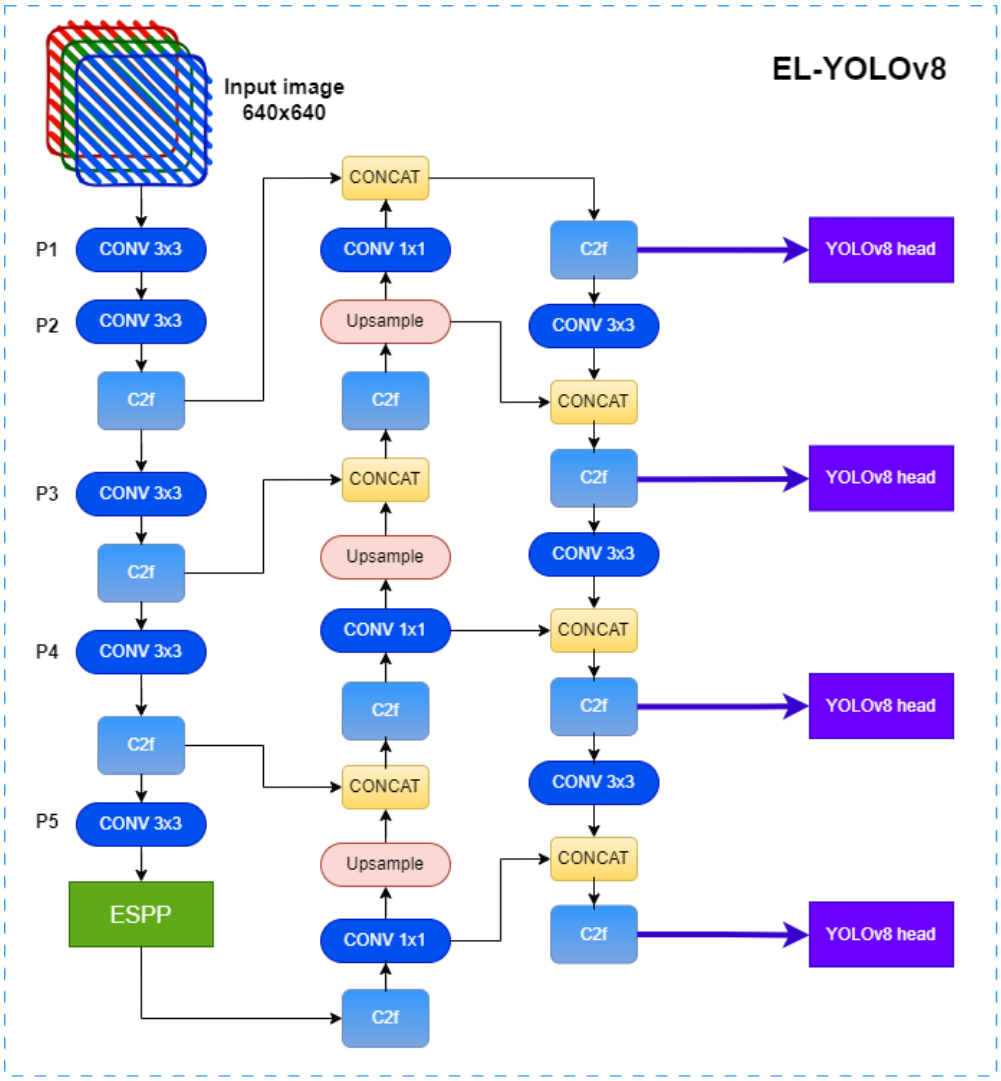

Figure 2.1.4.2 EL-YOLOv8 architecture.

The model is implemented into the Ultralytics framework by adding the ESPP block and changing the configuration file for the YOLOv8 models. As the source code for the EL-YOLOv5 proposed in [53] is not publicly available, we implement our code based on network visualization and description in [53]. With the proposed model we are ending the technical overview of object detection and moving to MOT.

2.2. Multiple object tracking

Nowadays, fast and accurate tracking in real-time is accomplished with deep learning-based trackers. Hardware improvements enable fast processing even with more neural networks for one task. Therefore, we chose the StrongSORT tracker which is an improved DeepSORT. Due to YOLOv8 possibilities to fine-tune detection, we select a tracker with a tracking-by-detection paradigm. This approach is modular, and each component has an opportunity to adapt to the desired task. We believe that the StrongSORT tracking approach is adaptable to the various problems because of its re-identification metric which can be additionally fine-tuned.

2.2.1. SORT – simple online and real-time tracking

SORT [54] was initially proposed by Alex Bewley et al. as a pragmatic approach to multiple object tracking (MOT). With a combination of Kalman filtering [55] techniques and Hungarian algorithms [56] as main tracking components, it outperforms other state-of-the-art online trackers. SORT is mainly focused on frame-to-frame tracking, which means that it considers only the current and previous frames to achieve tracking. The proposed object tracker is based on detection quality, and its main task is to associate corresponding bounding boxes on the current frame by adding an ID. The center and the aspect ratio of the detected bounding box are the principal information for estimating velocity components via the Kalman filter. To assign detections to existing targets, SORT calculates intersection-over-union (IOU) distance for optimally assigning new predictions to existing targets. Occluded objects that are not detected will lose their identity until being detected again. Due to poor re-identification possibility, occluded objects will probably appear with new IDs. According to the [54] benchmark, SORT outperforms online MOT competitors mostly by MOT accuracy, false alarm per frame (FAF), the number of mostly lost trajectories, and the number of false detections. An overview of the SORT MOT is in Figure 2.2.1.1.

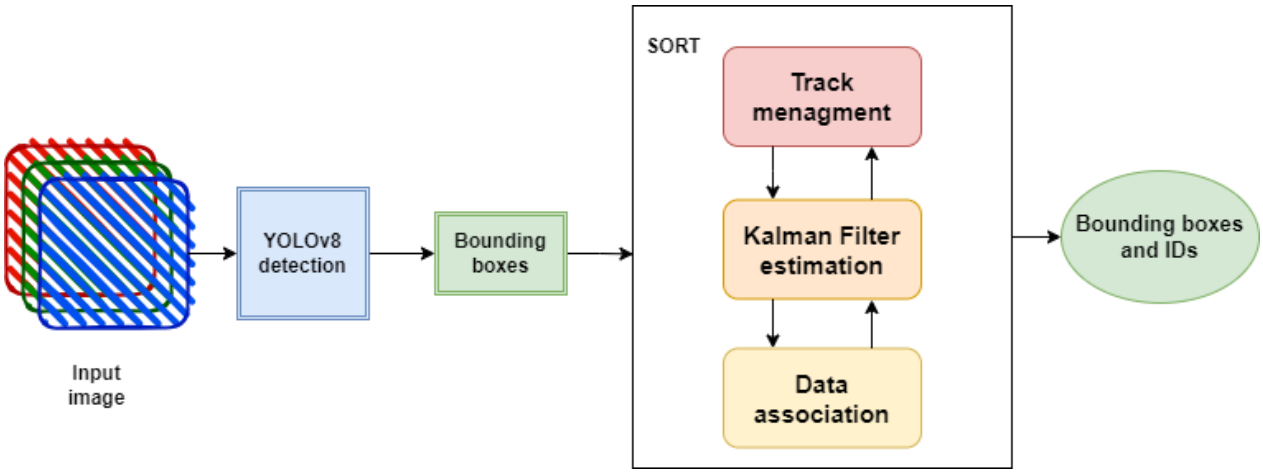

Figure 2.2.1.1 SORT multiple object tracking approach.

2.2.2. DeepSORT – simple online and real-time tracking with a deep association metric

N. Wojke et al. (2018) expand SORT algorithms with appearance information [28]. Appearance information enables the tracking of objects through longer periods of occlusions and reduces the number of generated identities. Due to the high number of identity switches caused by high estimation uncertainty, the association metric is replaced. Upgraded metric relies on a convolutional neural network (CNN) trained on motion analysis and re-identification dataset (MARS) [57]. MARS is a large-scale person re-identification dataset that contains 1,261 identities and around 20,000 tracks. This approach brings more information by combining motion and appearance information. The Kalman filtering framework is almost the same as in SORT [54]. The main idea is to count the number of frames while the track has not been associated with a measurement. Tracks that pass the pre-defined maximum age are considered left from the scene and deleted from the track set. Initialized tracks must be successfully associated with a measurement within the first three frames, for they will otherwise be deleted. To overcome a conventional way to solve the association between the predicted Kalman states and newly arrived measurements, authors incorporate the squared Mahalonobis distance:

$$d^{(1)}(i,j) = \left(d_j - y_i\right)^T S_i^{-1} (d_j - y_i)$$

where the i-th track distribution is projected into measurement space by $(y_i,\ S_i)$ and j-th bounding box by d_j. The Mahalonobis distance measures how many standard deviations are a detection away from the mean of track. Furthermore, associations that are unlikely are excluded by thresholding the Mahalonobis distance at a 95% confidence interval of the

inverse χ^2 distribution. Due to limitations caused by rapid displacements, N Wojke et al. integrated the second metric into the assignment problem. The second metric is called appearance descriptor, calculated from the detection bounding box. For each bounding box, the proposed CNN calculates the appearance descriptor by generating a 128-dimensional feature map. The feature map is preceded by a CNN consisting of two convolutional layers, six residual blocks [58], and a dense layer. Due to this output vector, the smallest cosine distance between tracks and new detections will determine which new detection corresponds to the past track. Tracks that have disappeared for some frames, but are still less than the maximum age, can thereby be correctly re-identified as existing ones. Figure 2.2.2.1 illustrates the flow of input image through an object detector whose output generates the appearance feature descriptor through a simple CNN model [28].

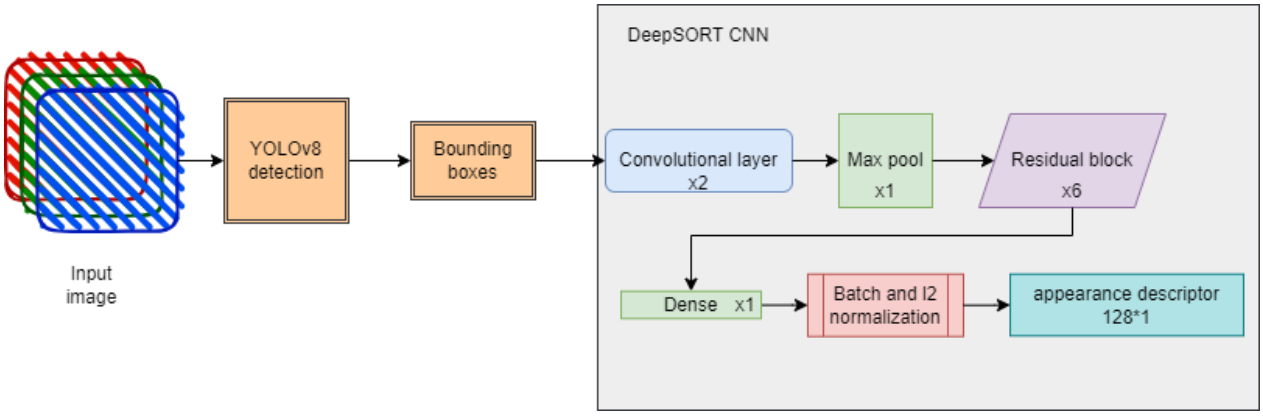

Figure 2.2.2.1 Computing the appearance feature descriptor by pre-trained CNN.

To conclude the assignment problem, the Mahalonobis distance and cosine distance of appearance feature descriptors are combined into a weighted sum:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j)$$

where $d^{(1)}$ is squared Mahalonobis distance and $d^{(2)}$ is the cosine distance of appearance feature descriptors. λ is a hyperparameter that controls the influence of each metric. The testing result shows that DeepSORT outperforms competitors mostly in identity switches and mostly lost (ML) which is the percentage of ground truth tracks that are tracked for at most 20% of their lifetime. More detailed descriptions of networks, improvements, and default tracking parameters are found in [28].

2.2.3. StrongSORT: Make DeepSORT Great Again

The domination of the MOT tracking-by-detection approach inspired Yunhao Du et al. (2022) to improve the standard DeepSORT (StrongSORT) and propose novel and lightweight algorithms (StrongSORT++) [59]. Due to outdated techniques, DeepSORT underperformed its state-of-the-art competitors in the past. The simple CNN model is replaced by BoT [60] with ResNeSt50 [61] as the backbone, and it was pre-trained on the Duke University multi-target multi-camera reidentification dataset (DukeMTCM-reID) [62]. ResNetSt (Hang Zhang et al., 2020) is presented as a modularized network structure that applies channel-wise attention through different network branches. This approach allows for capturing cross-feature interactions and learning diverse representations [61]. MTMC re-ID consists of more than 2,700 identities within 85 minutes through multi-camera, leading to more distinctive features. Furthermore, the appearance state is updated in an exponential moving average (EMA) manner that also reduces the time consumption, unlike the feature bank mechanism that utilizes features of last 100 frames. The enhanced correlation coefficient (ECC) [63] is adopted to fix camera motion compensation. The ECC is a low computational complexity solution for more accurate alignments under noisy conditions and photometric distortions. The vanilla Kalman filter is replaced by the noise scale adaptive (NSA) Kalman algorithm proposed in [64] due to its susceptibility to low-quality detections to low-quality detections. The NSA Kalman algorithm adaptively updates the noise covariance *k:*

$$\acute{R}_k = (1 - c_k)R_k$$

where R_k is the measurement covariance and c_k is the detection confidence score at state k. During the matching, both appearances cost A_a and motion cost A_m are being considered into cost matrix C as a weighted sum as follows:

$$C = \lambda A_a + (1 - \lambda)A_m$$

λ is a weight factor set to 0.98. To make the tracker stronger and more robust, the matching cascade is replaced by vanilla global linear assignment. StrongSORT++ brings new algorithms such as the appearance-free link (AFLink,) which stands for the first global link model without appearance information, and Gaussian-smoothed interpolation (GSI) which deals with missing detections. A more detailed description of StrongSORT++ is referenced in [59]. The framework comparison is in Figure 2.2.3.1.

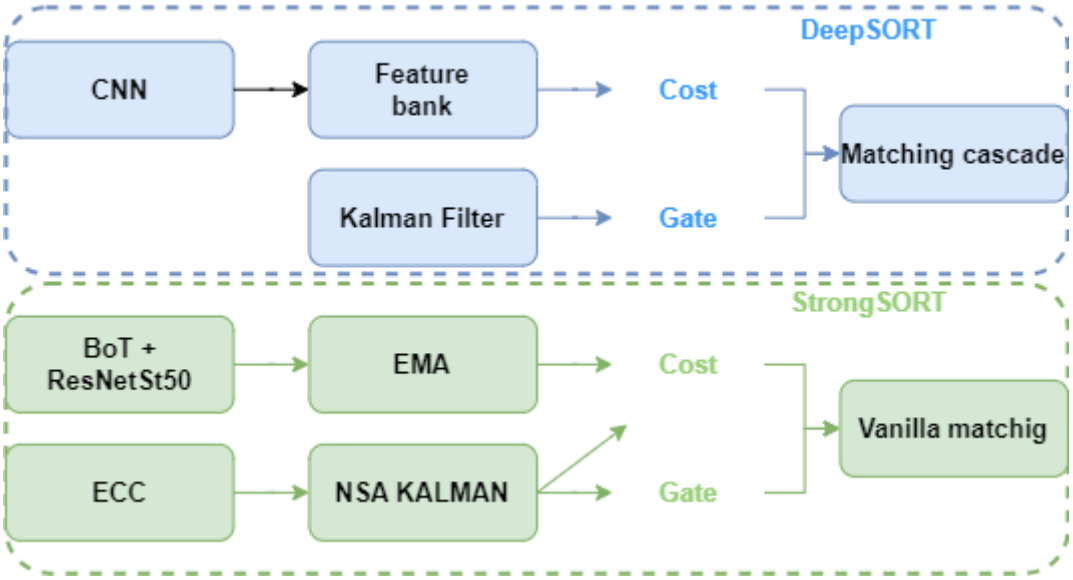

Figure 2.2.3.1 DeepSORT and StrongSORT MOT framework comparison.

2.3. VisDrone 2019

The dataset chosen for multiple object tracking purposes is VisDrone 2019. VisDrone 2019 dataset contains 288 videos, 261,908 frames, and 10,209 static images. The dataset is captured by various drone cameras in 14 different cities in China by the AISKYEYE team. Frames are manually annotated with more than 2.6 million bounding boxes. The dataset is focused on pedestrians and vehicles divided into 10 classes: pedestrian, person, car, van, bus, truck, motor, bicycle, awning tricycle, and tricycle. Pedestrians are humans who maintain a standing pose or walk while others are considered as class people.

VisDroneDET – 2019 is the dataset prepared for the object detection tasks we are interested in. It consists of 8,599 images with more than 540,000 bounding boxes. The training set consists of 6,471 images, the validation set contains 548 images, and 1580 images are intended for the test set [43]. The fact that the number of images in the test set is almost three times higher than in the validation set may indicate that VisDroneDET – 2019 is designed for challenge purposes. Figure 2.3.1 illustrates a histogram of VisDroneDET 2019 instances per class [43].

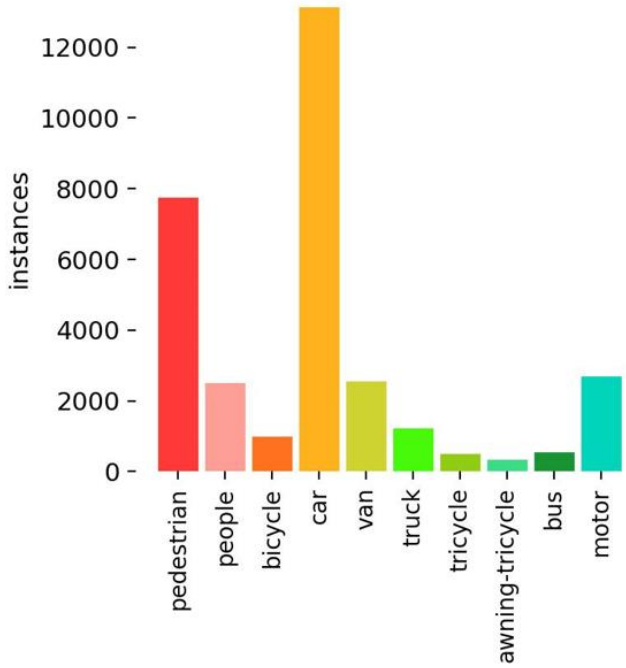

Figure 2.3.1 Instances of VisDroneDET-2019 detection dataset generated at the start of the training by the YOLOv8.

3. The dataset methodology

The dataset methodology includes a sampling approach to demonstrate models' behavior on the dataset samples. Sampling is needed to facilitate the process of selecting the most suitable models for the training on the entire dataset. This methodology approach is mainly chosen to enable reproductivity of the entire experiment.

3.1. The dataset – VisDrone 2019

VisDrone 2019 object detection dataset is selected because of its availability. The latest versions, such as VisDrone 2021, demand registration on the main page, which is avoided due to the reproducibility of the experiment. The dataset is downloaded from the official GitHub repository [43]. The downloaded dataset contains annotations in VisDrone format which need to be converted into YOLOv8 format, where each file represents normalized annotations on the corresponding image. Fortunately, Ultralytics provides Python scripts for downloading and converting VisDrone annotations into YOLOv8 format [19]. Due to hardware constraints, the train set is randomly divided into a base sample and five secondary samples. Each sample contains 10% of the train set, randomly selected by the Python random module [65]. Train samples enable faster training leading to a higher possibility of tuning hyperparameters. Metrics comparison in Figure 3.1.1 shows that each sample is a representative sample for experiments.

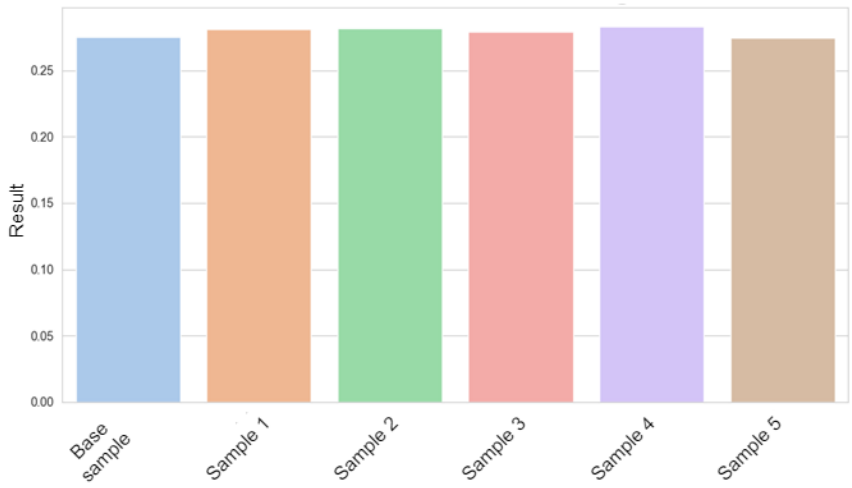

Figure 3.1.1 Histogram of metric comparison for the base sample and secondary samples.

Hence, we can assume that samples are representative samples of the dataset.

The command line code in Code 3.1.1 is used to train each sample for 20 epochs:

```
yolo detect train data="data_yaml_path" model=yolov8s.pt
epochs=20 imgsz=640 batch=-1 project="VisDrone" name=sampleN
seed=123 val=false cache=true workers=3
```

Code 3.1.1 – Command line code used for the sample training.

The parameter *val=False* will validate the best weights after the training. The number of epochs is 20 due to the pre-trained yolov8s.pt model that enables fast convergence. The batch size is set to be automatically calculated and the batch size of 16 is the most suitable for used GPU (T4 with 15GB in Google Colaboratory). The next section describes the experiment which includes training, results, and the final model for real-time detection and tracking.

4. The experiment

After selecting the candidate models, the experiment section contains an explanation of the training approach for the selected models. The results sections compare the efficiency of the validation set with metrics and examples. The last section presents an approach to building a model that detects and tracks objects in real-time. The programming language, frameworks, packages, and modules are contained in the last section.

4.1. Training approach

Due to the easy-to-use Ultralytics framework containing configuration files for YOLOv8 models, three models are trained to achieve the MOT tracking model:

1. Pre-trained MS COCO weights: pre-trained models enable fast convergence due to already learned features from MS COCO classes. Training on base and secondary samples shows that the model achieves the best results within 20 epochs.
2. EL-YOLOv8s (ours): our model achieves slightly worse results than the pre-trained YOLOv8s model through 100 epochs on the base sample. Due to that, we trained our model for an additional 100 epochs and presented performance in the results section.
3. YOLOv8-p2 + ESPP: we modified YOLOv8 which includes the p2 stage into further computation as EL-YOLO but blocks and modules are constructed as a basic YOLOv8. Additionally, we implement ESPP into YOLOv8-p2 and test it on the base sample. This model is also trained for 100 + 100 epochs.

The YOLOv8 (ultralytics) framework is still under development so the training is not optimized. RTX 3060 (6GB) mobile can handle batch of size 8 and more than 30 minutes is required per epoch which is extremely slow. Training in Google Colaboratory brings new types of problems like constant GPU memory problems. Free T4 with 15 GB cannot handle a batch of size greater than 6, although setting *batch=-1* parameter calculates a batch size of 32 as ideal for the proposed GPU. Gradient Paperspace also provides GPUs within notebooks similar to Google Colaboratory. Hence, the third option for training is via gradient notebooks. The standard YOLOv8s model with pre-trained weights is trained in Google Colaboratory while models that demand training from scratch are trained in Gradient notebooks.

Code 4.1.1 is used to train the models.

```
yolo detect train model=model.yaml/model.pt data="data.yaml"
epochs=100 imgsz=640 batch=16 project="Drone" name="name"
seed=123 cache=True
```

Code 4.1.1 – Command line code for training models in Gradient Paperspace terminal.

4.2. Results

Training results show the performance of the YOLOv8 and its modified versions. Table 4.2.1 contains data about modified network layers, parameters, and results.

Metric / Model	Precision (%)	Recall (%)	mAP_{50} (%)	$mAP_{50:95}$ (%)	Layers	Parameters (M)
YOLOv8s	55.3	42.0	44.4	27.1	225	11.1
YOLOv8s-P2 + ESPP	52.2	42.3	42.9	26.2	279	11.7
EL-YOLOv8s	49.5	40.2	41.3	25.2	274	9.5

Table 4.2.1 YOLOv8s-based architecture summary and metrics.

Results are evaluated on the validation set after 100 epochs. The model with pre-trained weights outperforms the proposed models. According to parameters in the network, EL-YOLOv8s has a slightly faster inference speed.

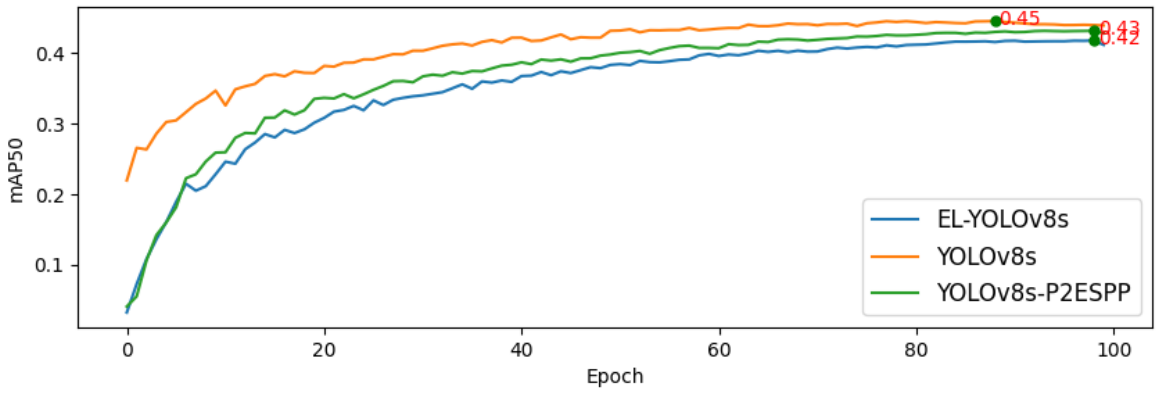

Figure 4.2.1 mAP50 metric comparison with peak value (green dot and red value)

Figure 4.2.1 shows slightly higher mAP_{50} peaks due to the fitness function that determines the best model weights as shown in Code 4.2.1.

```
def fitness(self):
      w = [0.0, 0.0, 0.1, 0.9]# weights for [P, R, mAP@0.5,
      mAP@0.5:0.95]
      return (np.array(self.mean_results()) * w).sum()
```

Code 4.2.1 YOLOv8 fitness function that determines the best weights.

Consequently, mAP_{50} contributes 10% while $mAP_{50:95}$ contributes 90% to the combined evaluation. Furthermore, EL-YOLOv8s and YOLOv8s-p2 + ESPP are trained for an additional 100 epochs due to the previous training from scratch. Additional training is started with pre-trained weights for each model with the same training strategy. Results are presented in Figure 4.2.2.

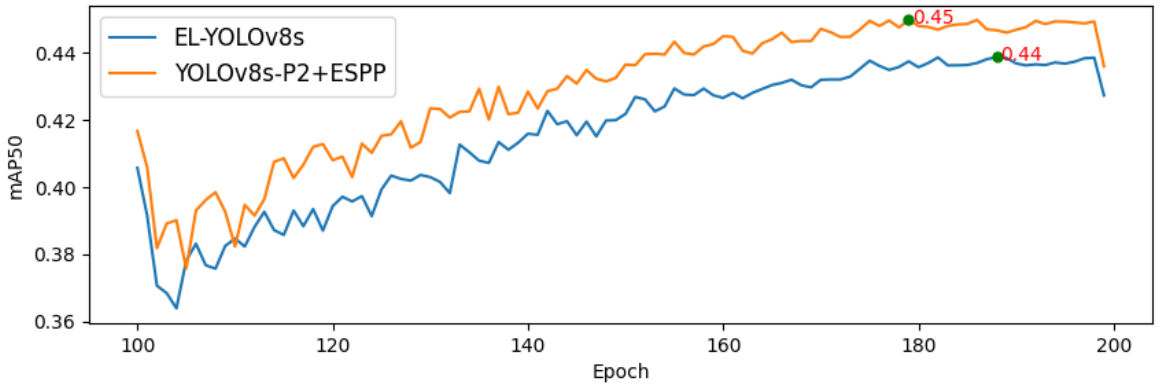

Figure 4.2.2 Training on EL-YOLOv8s and YOLOv8s-p2+ESPP for additional 100 epochs.

The experiments with YOLOv8-based models have no significant difference in metrics. The efficiency of the proposed EL-YOLOv8s and YOLOv8-ps + ESPP got closer to the basic pre-trained YOLOv8s model. Results with an additional 100 epoch are in table 4.2.2.

Metric / Model	Precision (%)	Recall (%)	mAP_{50} (%)	$mAP_{50:95}$ (%)	Layers	Parameters (M)
YOLOv8s	55.3	42.0	44.4	27.1	225	11.1
YOLOv8s-P2 + ESPP	54.6	42.7	44.3	26.9	279	11.7
EL-YOLOv8s	51.0	42.2	43.8	26.5	274	9.5

Table 4.2.2 Results after additional 100 epochs of training for EL-YOLOv8s and YOLOv8s-p2 + ESPP.

Although EL-YOLOv8s has the lowest value for $mAP_{50:95,}$ it has approximately 1.6 million parameters less than the other proposed models. Accordingly, EL-YOLOv8s is selected for further experiments for real-time MOT. Table 4.2.3 presents metrics evaluated for selected EL-YOLOv8 for each class.

Class / Metric	All	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	Awning-tricycle	Bus	Motor
Precision (%)	51.0	54.1	54.4	28.7	75.1	52.9	46.1	44.5	27.7	69.6	57.2
Recall (%)	43.2	48.6	38.1	17.0	82.6	51.0	39.2	32.7	20.1	53.4	49.0
mAP_{50} (%)	48.3	50.1	40.7	14.9	85.5	50.4	37.1	31.6	16.1	61.5	50.5
$mAP_{50:95}$ (%)	26.5	23.9	16.6	6.7	62.2	36.1	24.4	18.2	9.6	44.1	23.5

Table 4.2.3 Metric evaluated for each class in the dataset with the EL-YOLOv8s model.

According to Table 4, the class bicycle and awning-tricycle ruin the performance of the model. Tricycle is a bit strange class and awning-tricycle sounds like a subclass of tricycle. The VisDrone [43] dataset is captured in China, and we believe these classes are specific to this country. Therefore, we decided to merge bicycle, tricycle, and awning-tricycle into a single-class bicycle. The model with 10 classes where two classes have significantly lower scores leads to less accurate detections. Mentioned classes are also less presented in the dataset (Figure 15). Additionally, these classes are not the most significant in drone traffic detection scenarios. Classes pedestrian and people are obvious classes that share the same features but in different contexts in traffic scenarios. Due to that, we repeat the same training strategy with the dataset where tricycle and awning-tricycle labels are mapped into class bicycle and class people is mapped into class pedestrian. The results are in Table 4.2.4.

Class / Metric	All	Pedestrian	Bicycle	Car	Van	Truck	Bus	Motor
Precision (%)	61.8	64.5	46.8	78.0	56.1	50.7	74.3	62.5
Recall (%)	49.1	51.3	26.8	80.7	49.7	39.6	50.2	45.4
mAP_{50} (%)	52.8	56.2	28.6	84.0	50.1	38.5	61.8	50.5
$mAP_{50:95}$ (%)	33.0	24.9	15.6	61.3	36.0	25.3	44.3	23.6

Table 4.2.4 Metric evaluated on the EL-YOLOv8s with classes tricycle and awning-tricycle merged into the class bicycle.

Now the model evaluates higher $mAP_{50:95}$ for 4.0% overall classes and bicycle is still the class with the lowest metrics. To summarize the Results section, Figure 4.2.3 visualizes feature maps generated by the SPPF and ESPP blocks.

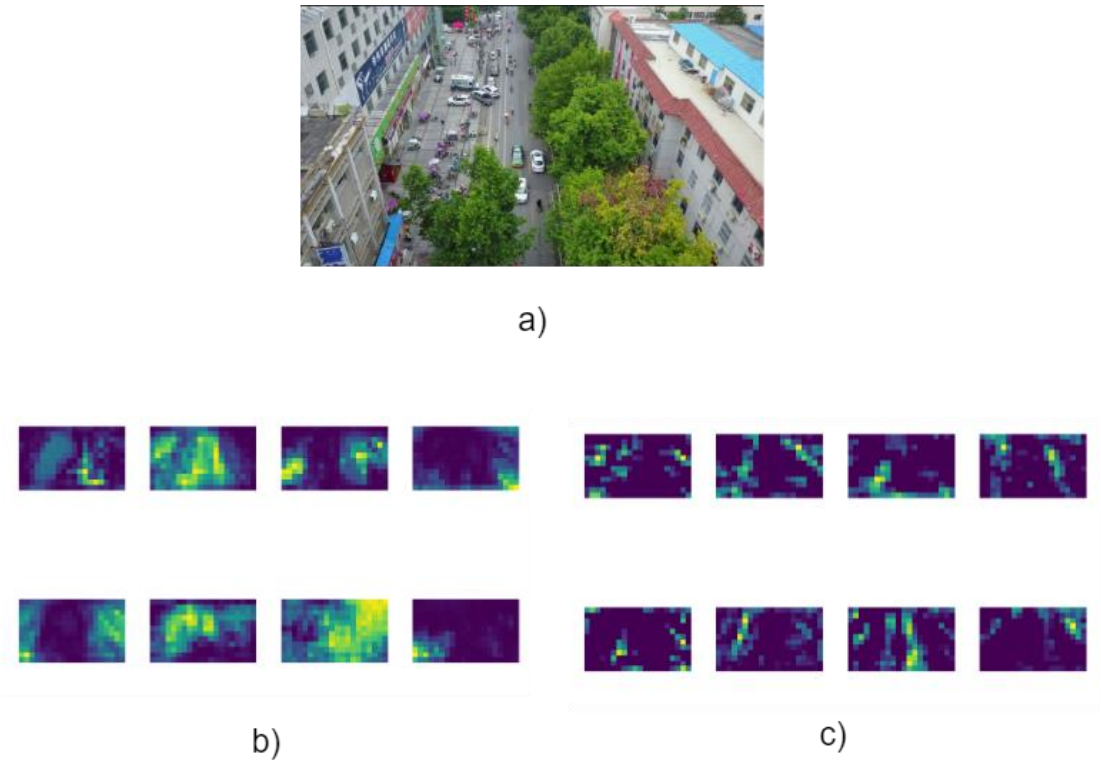

Figure 4.2.3 a) input image from the VisDroneDET-2019 validation set. b) SPPF feature maps (YOLOv8s). c) ESPP feature maps (EL-YOLOv8s).

SPPF output feature maps put focus on wider area of object features while ESPP relies on searching for individual small object features. A few predictions on the test set, that is not utilized for training or metrics evaluation, are shown in Figure 4.2.4.

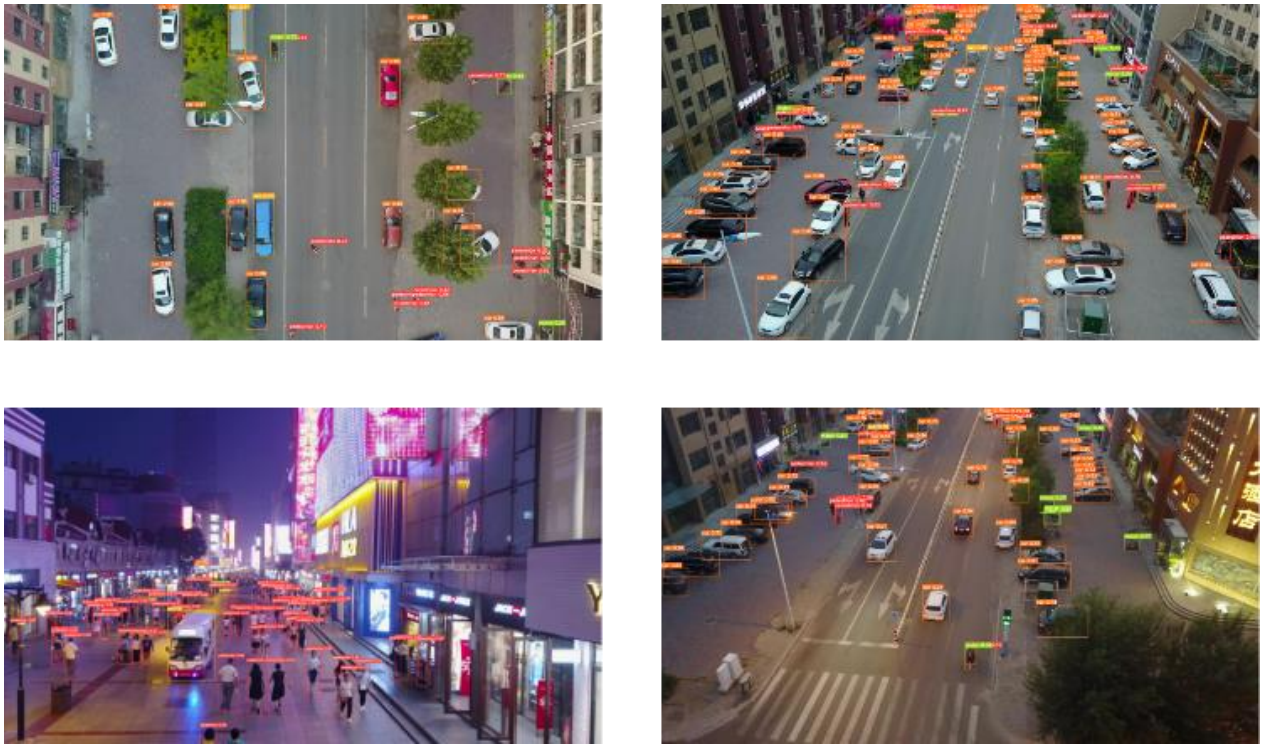

Figure 4.2.4 Predictions on test set images with EL-YOLOv8s which has merged classes.

4.3. Model for object detection and tracking

This work is publicly available in [66]. Python is the most popular programming language, especially in the Machine learning field and it is selected for this purpose. YOLOv8 and StrongSORT are implemented in Python, so there is no need for additional implementation. The main python libraries and frameworks utilized for this purpose are:

- **Ultralytics** [19] contains YOLOv8 models with corresponding scripts for training, validation, test etc. It enables computer vision workflow in Python code or by using command-line interface (CLI). Besides object detections, the Ultralytics framework provides the possibility to try diverse computer vision tasks such as instance segmentation, object tracking, pose estimation and image classification. The framework also contains YOLOv5 and YOLOv3 models and it is implemented in PyTorch.
- **PyTorch** is a framework that accelerates deep learning on the graphics processing unit (GPU). Having Nvidia GPU which supports the Cuda toolkit is necessary to utilize GPU acceleration. By installing the Ultralytics package, PyTorch is contained in dependencies, and modules for the CPU will be installed. According to that, PyTorch for GPU should be installed first [67].
- **Boxmot** is a collection of state-of-the-art object trackers which includes various trackers for the tracking-by-detection paradigm. It simplifies utilizing trackers such as StrongSORT with YOLOv8 due to the number of algorithms and additional neural networks for re-identification purposes. This work is mainly focused on StrongSORT object tracking, though utilizing other trackers would not be complicated [68].
- **OpenCV** is the computer vision and machine learning library that provides common infrastructures for applications. It provides classes and functions to struggle with the streams and videos that are necessary for end-to-end object detection and tracking [69].

Detailed installation instructions are in [66]. After preparing a virtual environment with necessary dependencies, the object-oriented paradigm is chosen as a programming approach in Python. Simple and straightforward object detection and tracking contain only one main class *ObjectDetection()*. This class contains the basic functions for model loading and prediction, *VideoCapture()*, and *tracker* initialization. The idea is to construct a simple general object detection and tracking model that could be easily upgraded for a specific purpose. We use the built-in *__call__()* function that handles the model pipeline. Firstly,

ObjectDetection() initializes the model and tracker with corresponding weights for detection and re-identification. Re-identification weights can be found on the Boxmot GitHub page in the ReID model zoo section. When the tracker and detector are ready, *VideoCapture()* opens the received video file, and inference starts. We detect objects for each frame that is successfully loaded by *VideoCapture()* and prepare data for the tracker. The tracker joins IDs to the bounding boxes and then draws them on the frame. To measure the performance, we calculated the time needed to load the frame, do detections and tracking on the frame, and visualize bounding boxes. Furthermore, time calculated by the Python performance counter is converted into frames per second FPS. Figure 4.3.1 shows a straightforward object detection and tracking approach with FPS calculation.

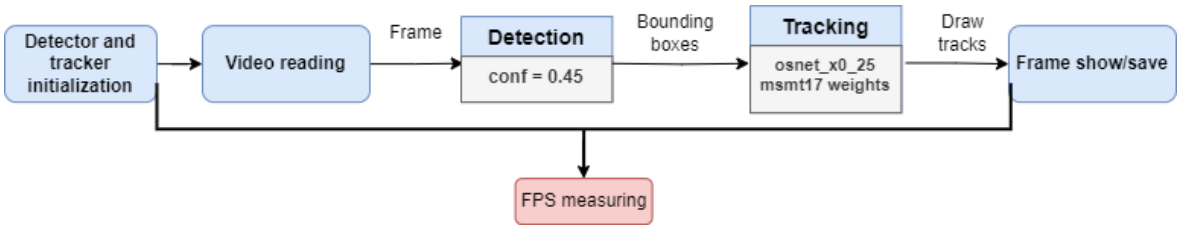

Figure 4.3.1 The pipeline of the object detection and tracking model with main hyperparameters.

To validate our model on videos we downloaded the VisDrone 2019 detection in videos validation set which contains 8 folders with sequences from the video. Sequences are converted to video by merging frame by frame with OpenCV's *VideoWriter()* class by using FPS rate of 20 and the same video resolution as image dimensions. FPS of 20 produces the most realistic objects movement speed and we chose it without additional testing. The reason why we are not training the model on this dataset is because of the images and label structure. Ground truth labels are in a single textual file for each sequence. Sequence consists of multiple images so searching for corresponding labels could be demanding. MOT metrics are not evaluated because we want to achieve real-time object detection and tracking in general instead of outperforming competitors on VisDrone tracking. The following figures show tracking results on the VisDrone videos:

Figure 4.3.2 10th frame extracted from the video1.

Figure 4.3.3 150th frame extracted from the video 1.

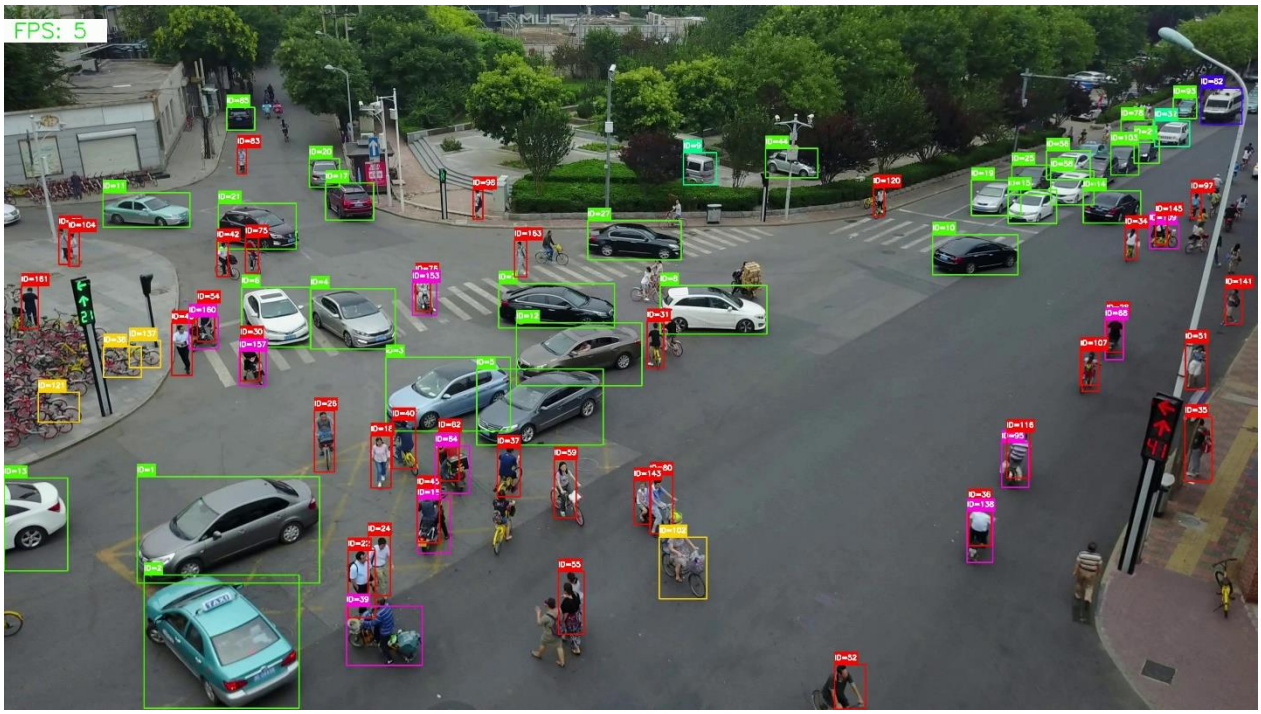

Figure 4.3.4 10th frame from the video 2.

Figure 4.3.5 150th frame from video 2.

As we can see, calculated FPS (top left corner) is not so real-time, but it depends mostly on the hardware. For the inference on videos, we utilize:

- Intel(R) Core (TM) i7-11800H @ 2.30GHz
- NVIDIA GeForce RTX 3060 Laptop GPU, 6GB
- 16,0 GB RAM
- CUDA Version: 12.2
- Windows 10 Education

Speed inference also depends on the number of objects that need to be further processed. The traffic situation in China usually with a high number of vehicles and other traffic scenarios with a lower number of objects leads to a higher FPS rate. According to Figures 4.3.2, 4.3.3, 4.3.4, and 4.3.5, we can conclude that YOLOv8 and StrongSORT detect and track most objects even in drone-moving scenarios. Re-identification has difficulties with similar objects that overlap in video. Due to that, some objects could be misidentified. Frames are in high resolution and traffic is crowded, so it would be exhausting to comment on each trajectory.

Conclusion

Reviewing the literature, we adopted the YOLOv8 algorithm and a StrongSORT framework as the most suitable to achieve the goal of this study: Real-time multiple object tracking for the objects captured by drone. With the help of the VisDrone 2019 dataset, we prepare weights for our goal. We additionally proposed a new EL-YOLOv8 model that unfortunately did not outperform the basic YOLOv8 algorithm in metrics. The difference in $mAP_{50:95}$ is only 0.6%, but our model has 1.6 million parameters less (~15% less). Hence, we chose EL-YOLOv8 as a model to accomplish real-time tracking. It is important to note that pre-trained YOLOv8 models demonstrate a considerably faster convergence than the models trained from scratch. Therefore, our model required additional time to achieve the best performances. To present our overall real-tracking model, we built it in Python and measured how much time the model needs to process in the period of loading a frame, object detection, and tracking step, and the period ends after bounding boxes visualization on the frame. The metric is additionally converted to frames per second and visualized on each frame. During the testing phase on VisDrone videos, the model's performance was notable in identifying challenges posed by crowded scenarios. However, it is anticipated that an increase in processing demands for detection, tracking, and visualization tasks could result in a potential slowdown of the model's overall performance. To conclude this thesis, we added some crowded scenario images and briefly described them. Due to the high image resolution and crowded scenario, our model is not the fastest one, but we believe that less crowded situations and hardware more powerful than mobile GPU would speed up inference. Therefore, our model is publicly available, and everyone can test their own weights and trackers and improve the model with even more computer vision tasks.

Literature

[1] Systematic Reviews (OPEN ACCESS) Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. Systematic Reviews 2021;10:89

[2] Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 Years: A Survey. Proceedings of the IEEE, 111(3), 257–276. https://doi.org/10.1109/JPROC.2023.3238524

[3] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 580–587. https://doi.org/10.1109/CVPR.2014.81

[4] Everingham, M., Eslami, S. M. A., van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The Pascal Visual Object Classes Challenge: A Retrospective. International Journal of Computer Vision, 111(1), 98–136. https://doi.org/10.1007/s11263-014-0733-5

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2014). LNCS 8691 - Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. http://arxiv.org/abs/1406.4729v1.

[6] R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.

[7] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556

[8] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[9] T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.

[10] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[11] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[12] Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement (cite arxiv:1804.02767 Comment: Tech Report)

[13] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. http://arxiv.org/abs/2004.10934

[14] Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2020-June, 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

[15] Jocher, G. (2020). Ultralytics YOLOv5 (Version 7.0) [Software]. AGPL-3.0. https://github.com/ultralytics/yolov5. doi: 10.5281/zenodo.3908559

[16] Lin, TY. et al. (2014). Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

[17] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. http://arxiv.org/abs/2209.02976

[18] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 2022, arXiv:2207.02696. https://github.com/

[19] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Software]. AGPL-3.0. https://github.com/ultralytics/ultralytics.

[20] Liu, W. et al. (2016). SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

[21] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2020). Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), 318–327. https://doi.org/10.1109/TPAMI.2018.2858826

[22] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 734–750.

[23] Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as Points. http://arxiv.org/abs/1904.07850

[24] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 213–229

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)

[26] Pal, S. K., Pramanik, A., Maiti, J., & Mitra, P. (2021). Deep learning in multi-object detection and tracking: state of the art. Applied Intelligence, 51(9), 6400–6429. https://doi.org/10.1007/s10489-021-02293-7

[27] Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple Hypothesis Tracking Revisited. 2015 IEEE International Conference on Computer Vision (ICCV), 4696–4704. https://doi.org/10.1109/ICCV.2015.533

[28] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 3645-3649, doi: 10.1109/ICIP.2017.8296962.

[29] Leal-Taix´e L, Canton-Ferrer C, Schindler K (2016) Learning by tracking: Siamese cnn for robust target association. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 33–40

[30] Tang S, Andriluka M, Andres B, Schiele B (2017) Multiple people tracking by lifted multicut and person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3539–3548

[31] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai, "Online multi-object tracking with convolutional neural networks," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 645-649, doi: 10.1109/ICIP.2017.8296360.

[32] Chu Q, Ouyang W, Li H, Wang X, Liu B, Yu N (2017) Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In: Proceedings of the IEEE International Conference on Computer Vision, pp 4836–4845

[33] A. Pramanik, S. K. Pal, J. Maiti and P. Mitra, "Granulated RCNN and Multi-Class Deep SORT for Multi-Object Detection and Tracking," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 6, no. 1, pp. 171-181, Feb. 2022, doi: 10.1109/TETCI.2020.3041019.

[34] Son J, Baek M, Cho M, Han B (2017) Multi-object tracking with quadruplet convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5620–5629

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[36] Ma C, Yang C, Yang F, Zhuang Y, Zhang Z, Jia H, Xie X (2018) Trajectory factory: Tracklet cleaving and re-connection by deep siamese bi-gru for multiple object tracking. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). IEEE, pp 1–6

[37] Fang, K., Xiang, Y., Li, X., & Savarese, S. (2018). Recurrent Autoregressive Networks for Online Multi-object Tracking. Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, 2018-January, 466–475. https://doi.org/10.1109/WACV.2018.00057

[38] L. Shafarenko, M. Petrou and J. Kittler, "Automatic watershed segmentation of randomly textured color images," in IEEE Transactions on Image Processing, vol. 6, no. 11, pp. 1530-1544, Nov. 1997, doi: 10.1109/83.641413.

[39] Milan, A., Rezatofighi, S. H., Dick, A., Reid, I., & Schindler, K. (2017). Online multi-target tracking using recurrent neural networks. 31st AAAI Conference on Artificial Intelligence, AAAI 2017, 4225–4232. https://doi.org/10.1609/aaai.v31i1.11194

[40] Sadeghian A, Alahi A, Savarese S (2017) Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In: Proceedings of the IEEE International Conference on Computer Vision, pp 300–311

[41] Kim C, Li F, Rehg JM (2018) Multi-object tracking with neural gating using bilinear lstm. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 200–215

[42] Schulter S, Vernaza P, Choi W, Chandraker M (2017) Deep network flow for multi-object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 6951–6960

[43] D. Du et al., "VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea (South), 2019, pp. 213-226, doi: 10.1109/ICCVW.2019.00030.

[44] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 2010, pp. 3485-3492, doi: 10.1109/CVPR.2010.5539970.

[45] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[46] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zis-serman, “The PASCAL visual object classes (VOC) challenge,” IJCV, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[47] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, "Path Aggregation Network for Instance Segmentation," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 8759-8768, doi: 10.1109/CVPR.2018.00913.

[48] Everingham, M., Van~Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2007). The Pascal Visual Object Classes Challenge 2007 (VOC2007) Results. Retrieved from http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

[49] Roboflow. What's New in YOLOv8. Roboflow Blog. URL: https://blog.roboflow.com/whats-new-in-yolov8/

[50] Ju, R.-Y., & Cai, W. (2023). Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm. http://arxiv.org/abs/2304.05071

[51] Li, X., Lv, C., Wang, W., Li, G., Yang, L., & Yang, J. (2023). Generalized Focal Loss: Towards Efficient Representation Learning for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(3), 3139–3153. https://doi.org/10.1109/TPAMI.2022.3180392

[52] Feng, C., Zhong, Y., Gao, Y., Scott, M. R., & Huang, W. (2021). TOOD: Task-aligned One-stage Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[53] Hu, M., Li, Z., Yu, J., Wan, X., Tan, H., & Lin, Z. (2023). Efficient-Lightweight YOLO: Improving Small Object Detection in YOLO for Aerial Images. Sensors, 23(14), 6423. https://doi.org/10.3390/s23146423

[54] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 2016, pp. 3464-3468, doi: 10.1109/ICIP.2016.7533003.

[55] R. Kalman, “A New Approach to Linear Filtering and Prediction Problems, ”Journal of Basic Engineering, vol. 82, no. Series D, pp. 35–45, 1960

[56] H. W. Kuhn, “The Hungarian method for the assignment problem,” Naval Research Logistics Quarterly, vol. 2, pp. 83–97, 1955.

[57] Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C., Wang, S., & Tian, Q. (2016). Mars: A video benchmark for large-scale person re-identification. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9910 LNCS, 868–884. https://doi.org/10.1007/978-3-319-46466-4_52

[58] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in BMVC, 2016, pp. 1–12.

[59] Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., & Meng, H. (2023). Strongsort: Make deepsort great again. IEEE Transactions on Multimedia, Publisher: IEEE.

[60] Luo, H., Jiang, W., Gu, Y., Liu, F., Liao, X., Lai, S., Gu, J.: A strong baseline and batch normalization neck for deep person re-identification. IEEE Transactions on Multimedia 22(10), 2597–2609 (2019)

[61] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., Li, M., & Smola, A. (2020). ResNeSt: Split-Attention Networks. http://arxiv.org/abs/2004.08955

[62] Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: European conference on computer vision. pp. 17–35. Springer (2016)

[63] Evangelidis, G. D., & Psarakis, E. Z. (2008). Parametric image alignment using enhanced correlation coefficient maximization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(10), 1858–1865. https://doi.org/10.1109/TPAMI.2008.113

[64] Du, Y., Wan, J., Zhao, Y., Zhang, B., Tong, Z., Dong, J.: Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2809–2819 (2021)

[65] Python Software Foundation. "random — Generate pseudo-random numbers." Python 3.10.6 Documentation. Available online: https://docs.python.org/3/library/random.html

[66] Farkas, L. (2023). Realtime Object Detection and Tracking with YOLOv8 and StrongSORT [GitHub repository]. Available online: https://github.com/lakyfarky/Realtime-object-detection-and-tracking-with-YOLOv8-and-StrongSORT

[67] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran

Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[68] Broström, M. BoxMOT: A collection of SOTA real-time, multi-object trackers for object detectors (Version 10.0.18) [Computer software]. https://doi.org/https://zenodo.org/record/7629840

[69] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools

[70] Fernando, T., Denman, S., Sridharan, S., & Fookes, C. (2018). Tracking by Prediction: A Deep Generative Model for Mutli-person Localisation and Tracking. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, *2018-January*, 1122–1132. https://doi.org/10.1109/WACV.2018.00128

List of figures

[1] Input image for Figures 1.1.1, 1.1.3, 1.1.5, 1.2.1 - https://pixabay.com/photos/highway-road-trucks-vehicles-3392100/

[2] Base image for Figure 1.1.4 - https://www.pexels.com/photo/blue-mini-cooper-11572937/

[3] Figure 1.2.2 is derived and modified based on the Fig. 3 found in [36]

[4] Figure 1.2.3 is derived and modified based on the Fig. 2 found in [40]

[5] Figure 2.1.4.1 is derived and modified based on the Fig. 5 found in [53]

[6] Figure 2.1.4.2 is derived and modified based on the Fig. 1 found in [53]

[7] Figure 2.2.3.1 is derived and modified based on the Fig. 2 found in [59]

[8] Figures 2.3.1 and 4.2.4 are generated by the Ultralytics framework [19]