Advanced Vision Systems for Dynamic Pedestrian Attribute Recognition and Tracking

Group 08: Andrea Vincenzo Ricciardi (2009), **Andrea Zinno** (2064), **Giovanni Rolando** (2018)

[1]Dept. of Information Eng., Electrical Eng. and Applied Mathematics - University of Salerno, Italy

Abstract This project revolves around creating a sophistical system capable of processing video inputs alongside a specified configuration, undertaking a sequence of pivotal operations. This includes the implementation of algorithms to accurately identify all individuals within a given scene, deployment of advanced tracking mechanisms to monitor and trace the movements of each individual over time, implement a pedestrian attribute recognition model for the acknowledgment of crucial attributes. Moreover, the project involves conducting in-depth analyses of the behavior of tracked individuals. This analysis encompasses tracking their movements through predefined Regions of Interest (ROIs) and determining the duration of their presence and the total number of passages in these areas.

For correspondence:

a.ricciardi38@studenti.unisa.it (AVR); a.zinno6@studenti.unisa.it (AZ); g.rolando1@studenti.unisa.it (GR)

1 | Introduction

In the rapidly evolving field of computer vision and artificial intelligence, the development of systems capable of interpreting and analyzing video feeds in real time represents a significant step forward in applications ranging from surveillance to customer behavior analysis in retail environments. The core of our system lies in its ability to identify individuals within a video scene, track their movements over time, and recognize important pedestrian attributes. These attributes include gender, the presence of bags or hats, and the colors of garments. Furthermore, the paper delves into the methodologies employed for tracking individuals across predefined ROIs and analyzing their behavior. This includes monitoring the duration of their presence in these areas and counting the total number of passages. This capability is crucial for a variety of applications, from enhancing security measures to providing insights into consumer behavior in retail settings.

2 | Detector: YOLOX

Object detection, a crucial aspect of computer vision, involves identifying and locating objects within images. The YOLO (You Only Look Once) series, a landmark in object detection models, has undergone several iterations, each improving upon its predecessor. **YOLOX** [3] is an object detection algorithm that stands out from other algorithms in the YOLO series due to its anchor-free design and decoupled head. Traditional YOLO models relied on predefined anchor boxes to predict object locations, which often led to limitations in detecting objects of varied sizes and shapes. YOLOX, by moving away from this anchor-based approach, simplifies the detection process and enhances the model's adaptability to different object dimensions. In other words, YOLOX can detect objects of various sizes and shapes more precisely and flexibly compared to other algorithms in the YOLO series.

The YOLOX consists of three parts: the backbone, the neck, and the head.

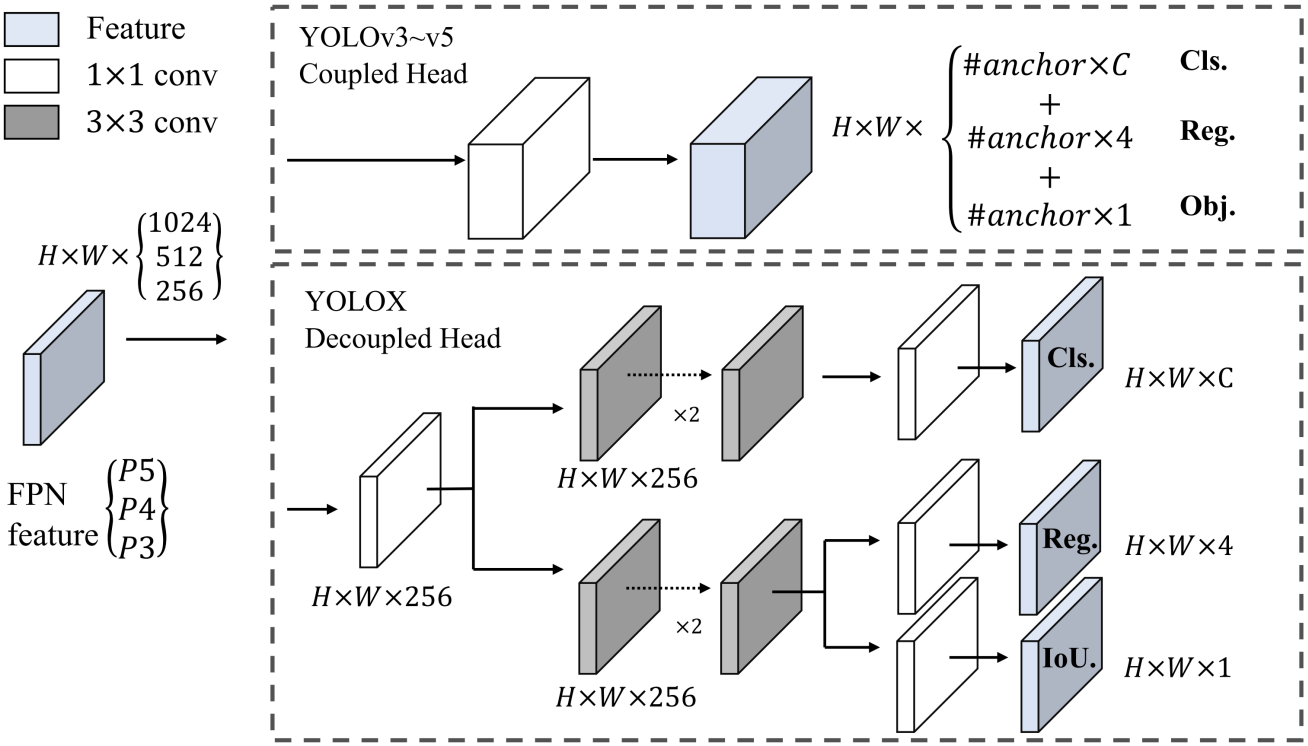


Fig. 1. Illustration of the difference between the YOLOv3 head and the YOLOX decoupled head [3]. For each level of FPN feature, YOLOX adopts a 1 × 1 conv layer to reduce the feature channel to 256 and then add two parallel branches with two 3 × 3 conv layers each for classification and regression tasks respectively. IoU branch is added on the regression branch.

2.1 | Darknet-53: The backbone of YOLOX

To understand the backbone of YOLOX, it's essential to first grasp the conceptual framework of **YOLOv3**, as YOLOX is deeply rooted in the innovations introduced by YOLOv3. YOLOv3 is known for its unique approach to object detection, utilizing a single neural network to predict bounding boxes and class probabilities directly from full images in one evaluation. This method contrasts sharply with traditional techniques that often involve separate stages for generating region proposals and classifying those regions. YOLOv3's architecture is characterized by its use of Darknet-53, a 53-layer network trained on ImageNet. For detection, YOLOv3 employs three scales, each with its own set of anchor boxes, to better detect objects at different sizes.

Anchor-free YOLOX evolves from this point, aiming to address certain limitations of YOLOv3 and its successors while introducing novel features to enhance performance and efficiency. One of the critical innovations in YOLOX is the abandonment of the anchor box mechanism. YOLOv3, along with its subsequent versions, YOLOv4 and YOLOv5, used predefined anchor boxes to improve detection performance, especially for small objects. However, these anchor boxes introduced complexities and limitations in the training process. YOLOX replaces this with an *anchor-free approach*, simplifying the model and making it more flexible in handling objects of various shapes and sizes.

simOTA simOTA, or Simplified Optimal Transport Assignment, is a novel label assignment strategy used in YOLOX. Traditional object detection models, especially those using anchor-based methods, often struggle with accurately assigning labels to bounding boxes in complex scenarios like occlusions. simOTA addresses this by focusing on the centers of objects rather than predefined anchors. In situations of occlusion, where objects overlap, it becomes challenging to assign ground truth labels to the correct bounding boxes. simOTA refines this process by optimizing

the assignment based on the Intersection over Union (IoU) values. It calculates the IoU between the predicted boxes and the ground truth, and then optimizes these assignments to ensure that each ground truth box is best represented, even in challenging conditions like overlapping objects.

Mixup and Mosaic Augmentation Moreover, the model incorporates several advanced strategies and augmentations to enhance its performance, notably the simOTA label assignment strategy, and the Mixup and Mosaic data augmentations.

- **Mixup augmentation**: it is a technique originally used in image classification tasks, and YOLOX adapts it for object detection. In this method, two images are blended together, creating a single composite image. This blending is not just a simple overlay but a weighted addition, where each pixel's value is a weighted sum of the corresponding pixels in the two original images. This technique can help the model learn more robust features by training on images that combine elements from different contexts, thereby improving its generalization capabilities.
- **Mosaic augmentation**: it is an efficient augmentation strategy proposed by ultralytics-YOLOv3. This strategy involves creating a single composite image from four different images. These images are arranged in a mosaic-like structure, and then a certain region of this composite image is cropped. This method is particularly effective in training the model to detect small objects. By combining multiple images, the model is exposed to a variety of object sizes and scales within a single training instance.

2.2 | Feature Pyramid Network

The neck connects the backbone and the head. It is composed of a *feature pyramid network* (FPN), which generates feature maps and corresponding grids at multiple scales, and a path aggregation network which combines the low-level and high-level features. The neck concatenates the feature maps from the backbone layers and feeds them as inputs to the head at three different scales (1024, 512, and 256 channels).

2.3 | Decoupled head

YOLOX simply has the model directly predict the bounding box dimensions as opposed to predicting an offset from an anchor box. To directly predict a bounding box, YOLOX uses a *decoupled head*. In conventional object detection models, the head of the network simultaneously handles both classification and bounding box regression (determining the location and size of the object). The decoupled head design addresses this issue by creating two separate heads or pathways within the network: one for classification and the other for bounding box regression. This separation allows each head to specialize and optimize for its respective task. The classification head can focus on extracting and processing features that are more relevant to identifying object classes, while the bounding box regression head can concentrate on spatial features critical for accurately locating and sizing the objects. Conctrely, the YOLO detect head is replaced with a lite detecouple head, that contains a 1×1 conv layer to reduce the channel dimension, followed by two parallel branches with two 3×3 conv layers respectively, as show in Fig. 1.

Output of YOLOX Both YOLOv3 and YOLOX output predictions are encapsulated in tensors that contain information about *object classes* (Cls), *bounding box coordinates* (Reg), and *object presence confidence* (IoU or Obj). In both models, each pixel in the height and width of the output is a different bounding box prediction. So, there are $h \times w$ different predictions. Both models utilize an FPN to make predictions at three different scales. In YOLOv3, this results in a single large tensor for each scale, combining all prediction aspects. In contrast, YOLOX outputs three separate tensors for Cls, Reg, and IoU at each scale. Therefore, if we consider all scales, YOLOX effectively generates nine separate outputs (3 Cls, 3 Reg, 3 IoU), one set for each scale.

Model	Size (px)	mAP^{val} 0.5:0.95	mAP^{test} 0.5:0.95
YOLOX-S	640	40.5	20.6
YOLOX-M	640	47.2	26.9
YOLOX-L	640	50.0	30.3
YOLOX-Darknet53	640	47.3	26.4
YOLOX-X	640	51.2	33.3

Table 1. Performance comparison of various YOLOX models.

In our implementation of the tracking algorithm, we have chosen to use **YOLOX-X** as the object detector, primarily due to its superior performance among the YOLOX series.

3 | Tracking: Deep SORT

While detection is effective for individual frames, it encoounters challenges when dealing with sequential images in a video. When we track an object using detection alone, we are essentially identifying it in each frame independently. However, without establishing a correlation or connection between the tracked features across sequential frames, issues arise, particularly in scenarios involving occlusion (when the object is partially or fully hidden). In such cases, if the detection is solely frame-based, there is a risk of losing track of the object during occlusion, and your target may slip out of the frame undetected. **Tracking**, on the other hand, aims to maintain continuity by linking the features of the object across frames, providing a more robust and consistent understanding of its movement over time.

One of the most notable contributions to this area is the development of **Deep SORT** (Simple Online and Realtime Tracking with a Deep Association Metric) [6], an extension of the original SORT (Simple Online and Realtime Tracking) algorithm. Deep SORT combines classic tracking methods with modern deep learning techniques to improve tracking accuracy, especially in scenarios where objects are occluded or move unpredictably.

3.1 | Background: SORT Algorithm

The **SORT** [2] algorithm is the foundation of the Deep SORT system. It is a simple yet effective algorithm for performing data association and track initiation in a multi-object tracking scenario. Specifically, it is an approach to Object tracking where rudimentary approaches like Kalman filters and Hungarian algorithms are used to track objects and claim to be better than many online trackers. SORT is made of 4 key components which are as follows:

- **Detection**: The initial phase in the SORT tracking process involves object detection within a frame. High-performance object detectors such as Faster R-CNN (FrRCNN), YOLO, and others are employed to identify objects of interest. These detections serve as the input for the tracking module, providing a set of bounding boxes along with their respective confidence scores.
- **Estimation**: Following detection, the SORT algorithm estimates the future positions of detected objects using a constant velocity model. This estimation is crucial for predicting where an object will appear in the next frame, aiding in the smooth tracking of objects across frames. The Kalman filter, a powerful tool for state estimation in linear systems subject to Gaussian noise, is applied here. When a new detection is associated with an existing target, the detected bounding box is used to update the target's state within the Kalman filter framework. This update optimally adjusts the velocity components of the target, enhancing the prediction's accuracy for the next frame.
- **Data Association**: Data association is the process of matching detected objects in the current frame with existing tracks based on their predicted states. SORT computes a cost matrix representing the Intersection over Union (IOU) distance between each detection and all predicted bounding boxes from the existing targets. The Hungarian algorithm, an optimization algorithm that solves the assignment problem in polynomial time, is then used to find the optimal assignment between detections and tracks. If the IOU between a detection and a track falls below a predefined minimum

threshold (IOUmin), the association is rejected to avoid incorrect matchings. This step is critical for handling occlusions and maintaining consistent object identities across frames.

- **Creation and Deletion of Track Identities**: The final component of SORT involves managing the lifecycles of track identities. New tracks are initiated for detections that cannot be associated with existing tracks, indicating the presence of previously untracked objects. Conversely, tracks are terminated if they fail to be associated with a detection for a specified number of frames (TLost), allowing the system to adapt to objects entering and leaving the scene. This dynamic management of track identities helps maintain an accurate and current representation of the tracked objects.

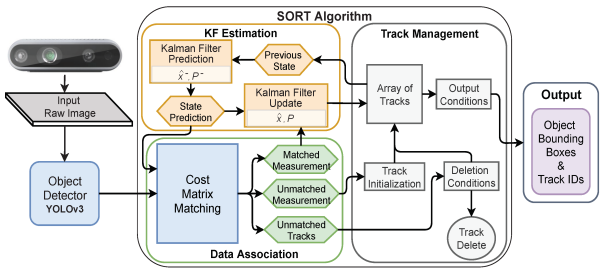


Fig. 2. SORT algorithm [5].

3.2 | Why DeepSORT?

Despite its efficiency, SORT has limitations, particularly in handling long-term occlusions and distinguishing between similar-looking objects. This is where Deep SORT comes into play. **Deep SORT** is an improvement of the SORT algorithm, integrating appearance information of objects to enhance associations. Data association integrates an additional appearance metric based on pre-trained CNNs allowing re-identification of tracks, after a long period of occlusion. The KF Estimation and the Track management modules are similar to the corresponding SORT modules. An overview of the method is presented in Figure 3. Deep SORT is made of 5 key components which are as follows:

- **Detection**: Similar to SORT, the first step in Deep SORT is the detection of objects in each frame. This process can be accomplished using any state-of-the-art object detector that provides bounding boxes around the detected objects.
- **Feature Extraction**: Once objects are detected, Deep SORT employs a CNN to extract feature embeddings from each detected bounding box. These feature embeddings capture the appearance information of the objects, which is crucial for distinguishing between different objects, especially in scenarios where they are closely located or partially occluded.
- **Estimation**: Each tracked object's future position is estimated using a Kalman filter. This prediction is based on the object's previous state, including its position and velocity, to forecast its movement and location in the next frame. The Kalman filter is also used to update the object's state when a new detection is associated with it. This update adjusts the object's estimated position and velocity based on the new detection, improving the accuracy of future predictions.
- **Data Association**: Data association in Deep SORT is significantly enhanced by the introduction of deep appearance features. Instead of relying solely on the IOU (Intersection over Union) metric between predicted and detected bounding boxes (as in SORT), Deep SORT computes a similarity score that combines both motion information (from the Kalman filter prediction) and appearance information (from the CNN feature embeddings). This similarity score is used to create a cost matrix for associating detected objects with existing tracks. The association problem is solved using the Hungarian algorithm, which finds the optimal assignment between detections and tracks that minimizes the overall cost.
- **Creation and Deletion of Track Identities**: Similar to SORT, Deep SORT manages the lifecycle of track identities to maintain an accurate representation of objects within the scene. However, Deep SORT enhances this process by leveraging deep appearance features, which allows for more robust handling of occlusions and identity management.

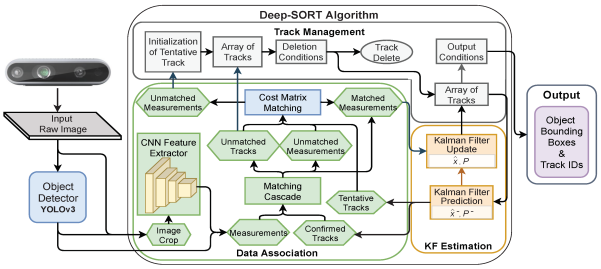


Fig. 3. DeepSORT algorithm [5].

3.3 | DeepSORT Implementation

The parameters of the Deep SORT implementation in PyTorch play crucial roles in determining how detections are associated with tracks, how tracks are maintained, and how the algorithm performs overall in terms of tracking accuracy and robustness. Below is a detailed explanation of each parameter within the context of Deep SORT:

- `max_dist = 0.15` (defualt 0.2): This parameter defines the maximum distance allowed between features to associate a detection with an existing track. Lowering this value makes the association criterion stricter, potentially reducing false positives but increasing the risk of missing correct matches. Increasing this value relaxes the matching criterion, which can reduce missed matches but may increase the rate of incorrect associations.
- `min_confidence = 0.45` (defualt 0.3): This parameter defines the minimum detection confidence for a detection to be considered by the tracker. Increasing this threshold ensures that only detections with higher confidence are considered for tracking, potentially reducing false positives but also risking the omission of valid objects with lower confidence scores. Decreasing this value allows more detections to be considered, which can increase recall but also the likelihood of tracking noise or irrelevant detections.
- `nms_max_overlap = 0.9` (defualt 0.5): This parameter defines the maximum overlap threshold used in Non-Maximum Suppression (NMS) to filter out detections. Lowering this threshold results in a more aggressive suppression of overlapping detections, which can help reduce duplicate tracks but may also miss tracking multiple objects that are close together. Increasing this threshold allows more overlapping detections to be retained, which can be useful in dense scenes but may result in more duplicate tracks.
- `max_iou_distance = 0.7` (defualt 0.7): This parameter defines the maximum IOU distance for matching detections to track objects. Decreasing this value makes the IOU matching criterion stricter, reducing the likelihood of associating detections to tracks based on spatial overlap. This can help in scenarios with closely spaced objects but might increase fragmentation of tracks. Increasing this value allows for looser associations based on spatial overlap, which can reduce track fragmentation but increase the risk of incorrect associations, especially in crowded scenes.
- `max_age = 100` (defualt 70): This parameter defines the maximum number of frames to keep a track without a matching detection. Increasing this value allows tracks to be maintained for longer periods without detection, which can be beneficial for handling occlusions or temporary loss of detection. However, it can also result in retaining outdated or irrelevant tracks. Decreasing this value makes track deletion more aggressive, which can keep the set of active tracks cleaner but may prematurely terminate valid tracks.
- `n_init = 5` (defualt 3): This parameter defines the number of consecutive detections required to establish a new track. Increasing this value requires more evidence (in terms of consecutive detections) to confirm a new track, which can reduce false positives but may also delay the initiation of valid tracks. Decreasing this value makes track initiation more responsive, which can be beneficial for quickly capturing new objects but may also lead to more false tracks.
- `nn_budget = 200` (defualt 100): This parameter defines the size limit for the feature cache, used to store features for track re-identification. Increasing this value allows the system to store more features per track,

potentially improving re-identification performance by considering a wider history of appearances. However, it also increases memory usage and computational cost. Decreasing this limit reduces the computational load but may impair the ability to re-identify objects over long periods or after occlusions, especially in environments with significant appearance changes.

Box 1. Kalman Filter

The **Kalman filter** is an efficient and versatile algorithm designed for estimating the state of a dynamic system that is subject to noise. The core principle of the Kalman filter is to combine noisy measurements of a system with predictions based on a model of the system itself, to produce optimal estimates of the system's current state. The operation of the Kalman filter can be divided into two main phases:

- **Prediction Phase**: The filter uses a mathematical model of the system to predict the future state of the system from its current state. This prediction also encompasses an estimate of the uncertainty associated with the state prediction.
- **Update Phase**: Upon acquiring new measurements, the filter updates its state estimates by blending the predictions with the newly obtained information. This update is based on the difference (or residual) between the actual measurements and those predicted by the system model.

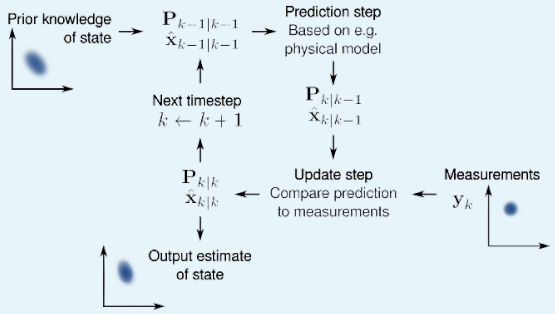


Fig. 5. The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate.

4 | PAR: MultiTask Attention Network

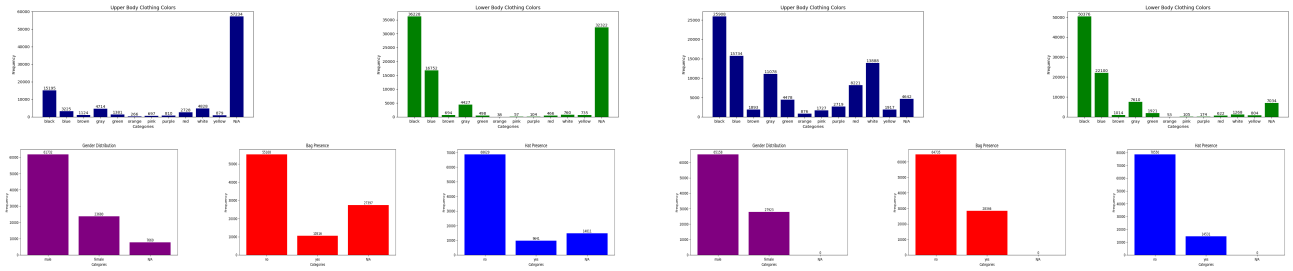
Pedestrian attributes recognition from images is nowadays a relevant problem in several real applications, such as digital signage, social robotics, business intelligence, people tracking and multi-camera person re-identification. To this concern, there is a great interest for recognizing simultaneously several information regarding pedestrians. The approach we propose leverages a **Multi-Task Attention Network** (`MTANet`), which is adept at handling the complexity and variability inherent in PAR tasks.

4.1 | Mivia PAR Dataset 2.0

To train our network to recognize pedestrian attributes, we trained it with the MIVIA PAR dataset [4]. This dataset comprises 105,244 images of cropped individuals, see Fig. 4a, separated into training (93,082) and validation (12,162) samples. Each sample is completely or partially annotated with numeric labels. The presence of a negative label for any sample refers to a non-annotated feature. The different features annotated are the following:

- **Color of the upper and lower clothes**. Two labels correspond to a single color associated with upper and lower-body clothes. Eleven possible colors are considered in the annotations: black (1), blue (2), brown (3), gray (4), green (5), orange (6), pink (7), purple (8), red (9), white (10), and yellow (11).
- **Gender of the foreground person**. The labels considered are male (0) and female (1).
- **Bag and Hat presence.** The respective labels considered are absence (0) and presence (1).

As visible from the distribution in the Fig. 4a, it is evident that this dataset presents several challenges typical of real-world data, including the presence of many unannotated labels and an imbalance in the data distribution. To address these issues, we employed **VILT** (Vision-and-Language Transformer) combined with **VQA** (Visual Question Answering) as a novel data imputation technique. By applying this technique, we were able to obtain a dataset with significantly fewer unannotated labels, as visible from Fig. 4b.



(a) (b)

Fig. 4. (a) Mivia PAR Dataset and (b) Mivia PAR Dataset 2.0.

The integration of VILT with VQA operates on a sophisticated multimodal analysis framework, combining visual and textual data processing to infer missing information. VILT, adept at understanding and integrating both image content and textual descriptions, enables the system to generate contextually relevant questions about the image content. Subsequently, VQA processes these questions against the visual data to predict the missing annotations. This method hinges on the system's ability to accurately interpret visual cues and answer questions about them, thus effectively "imputing" missing data based on visual evidence and contextual understanding. To utilize VILT with VQA for imputing missing annotations, we prepared a set of questions aimed at identifying the listed attributes in the images. For instance, if an image's annotation lacks the color of the person's shirt, our system automatically generates a question such as "*What color is the person's shirt?*" If the VQA component predicts a color with high confidence, this information is used to update the missing annotation. This conditional update strategy significantly reduces the likelihood of introducing errors into the dataset.

4.2 | Architecture

Our approach, encapsulated within the MTANet architecture, leverages the power of Multi-Task Learning (MTL) to simultaneously recognize multiple pedestrian attributes. The `MTANet` (see Fig. 7) is designed to handle the inter-task correlations and the unique challenges posed by the pedestrian attribute recognition problem, particularly the varying levels of difficulty in attribute prediction and the potential imbalance in attribute distribution.

Feature Extraction At the heart of MTANet is a feature extraction module based on a pre-trained `ResNet-50` model. This network has been shown to perform well on a variety of vision tasks and provides a rich representation of the input image. We utilize the feature map generated by the `ResNet-50` model, excluding its final classification layer, to maintain a high-dimensional feature space that can capture detailed information relevant to various attributes.

CBAM The concept of *attention* in neural networks is inspired by the cognitive attention mechanism in humans, which refers to the ability to focus on specific elements of our environment while ignoring others. This process helps the network to 'attend' to the most informative parts of the input, which is particularly useful in tasks where certain features are more relevant than others for making accurate predictions. In the MTANet, attention is applied at two levels: *channel-wise* and *spatial*. This dual focus allows the network to discern not only which features are important but also where they are located in the input space.

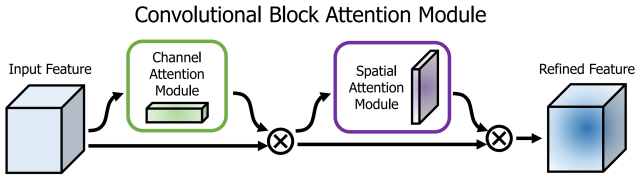


Fig. 6. Convolutional Block Attention Module [7].

The **CBAM** (Convolutional Block Attention Module) [7] is designed to be modular and can be seamlessly integrated into any CNN architecture. The module operates in two distinct stages: first, it processes the input feature map through

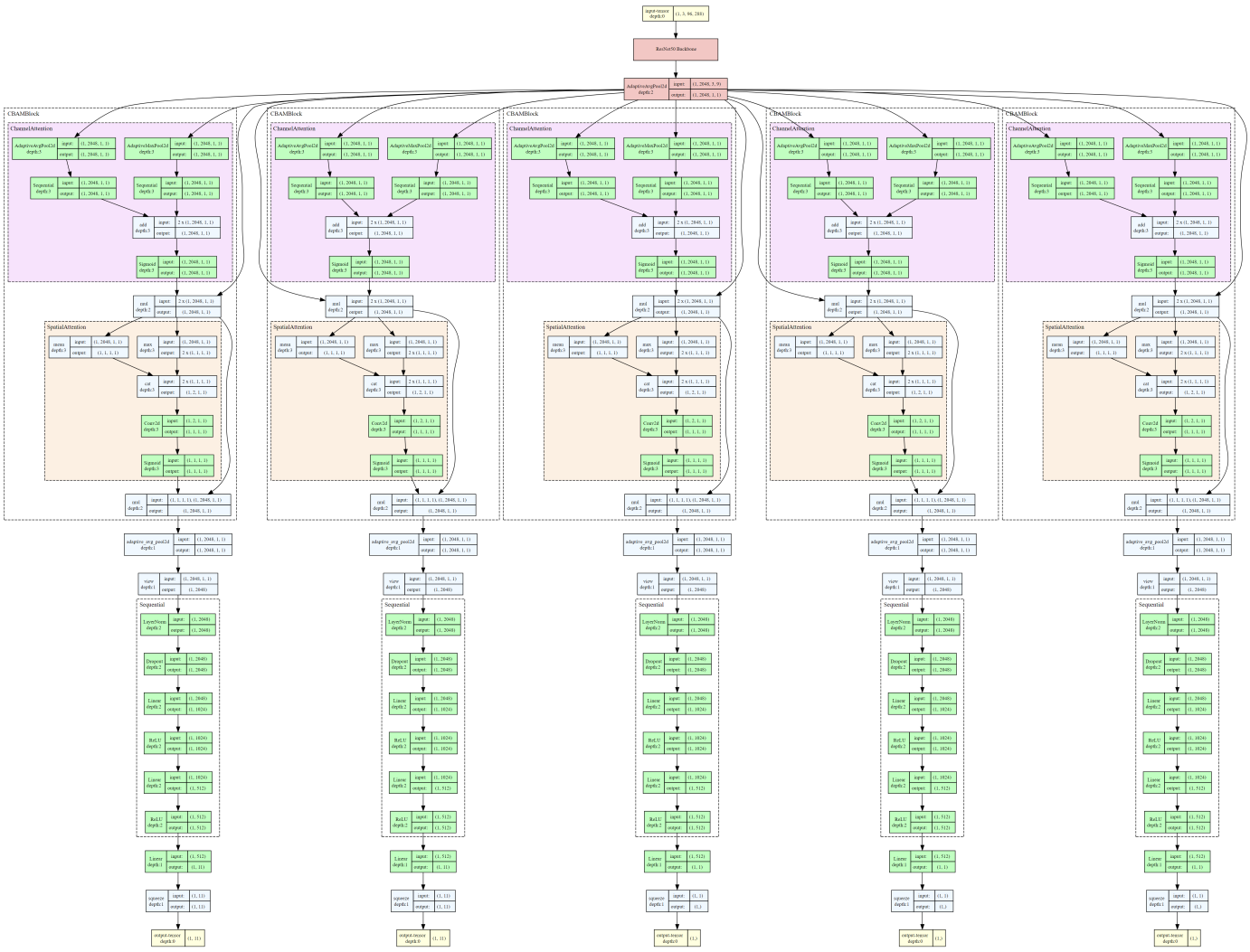


Fig. 7. MTAN Architecture.

the Channel Attention mechanism, and then it refines this feature map further with the Spatial Attention mechanism. Specifically:

- **Channel Attention**: This sub-module focuses on '*what*' features to emphasize or suppress across the channel dimension. It leverages global average and max pooling to highlight inter-channel relationships and determine which channels are more important.

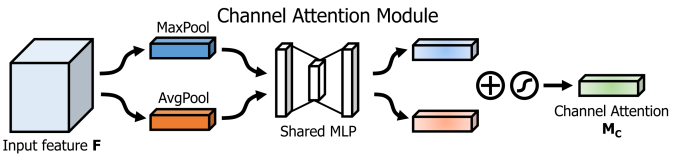


Fig. 8. Channel Attention Module [7].

- **Spatial Attention**: This submodule concentrates on the '*where*' by producing a spatial attention map. It uses the outputs of average and max pooling across the channel axis to highlight salient spatial regions in the feature map. This allows the network to focus on the most informative areas within the given context of the image.

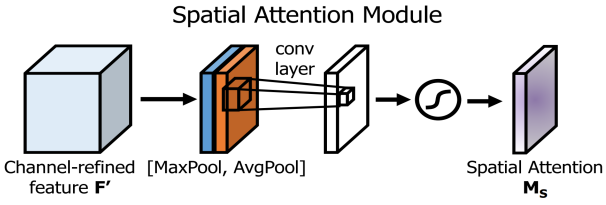


Fig. 9. Spatial Attention Module [7].

Task-Specific Processing After the attention modules, the features are pooled and flattened before being passed through task-specific layers. These layers consist of normalization and dropout for regularization, followed by linear layers with ReLU activations. This process tailors the attention-modified features for each classification task, ensuring that the representation is optimal for the attribute of interest.

Classifiers At the end of the pipeline, for each attribute, there is a classifier responsible for making the final prediction. These classifiers are `FCL` tailored to the number of classes for each attribute. For binary attributes (like gender, bag, and hat presence), the classifier outputs a single value that is thresholded to determine the class. For multi-class attributes (like upper and lower body color), the classifier outputs a probability distribution across all possible classes.

4.3 | Asymmetric Loss

In PAR, some attributes are less common than others, leading to class imbalance (see Fig 4b). Such imbalance is particularly common in real-world datasets and can significantly skew the training process, leading to models that perform well on majority classes but poorly on minority classes, which are often of greater interest. The typical loss functions, such as cross-entropy loss, assume that all classes are equally important, which often isn't the case in practical applications.

The **Asymmetric Loss** [1] function is designed to counteract this issue by adjusting the loss contribution of each class. It does so by applying different weights to the loss associated with the majority and minority classes. This helps in preventing the model from being biased towards the majority class and encourages it to pay more attention to the minority class.

Asymmetric Focusing Let γ_- and γ_+ be the negative and positive focusing parameters, respectively. Let $p = \sigma(z)$ be the network's output probability. We obtain asymmetric focusing by redefining the loss:

$$\begin{cases} L_- = (p)^{\gamma_-} \log(1-p) & \text{for negative samples} \\ L_+ = (1-p)^{\gamma_+} \log(p) & \text{for positive samples} \end{cases} \quad \textbf{1}$$

However, with extreme class imbalances, soft thresholding might not be sufficient. Thus, the concept of probability shifting is introduced, which performs a hard thresholding on very easy negative samples. This means that if the predicted probability for a negative sample is very low (below a certain margin), it's fully disregarded during loss calculation, as if to say the model is confident enough about this prediction that it doesn't need to focus on it. The shifted probability, p_m, is defined as:

$$p_m = \max(p - m, 0) \quad \textbf{2}$$

Here, m represents the probability margin, a tunable hyperparameter that decides the threshold for disregarding easy negatives (e.g., $m = 0.2$).

Asymmetric Loss By incorporating the shifted probability into the negative part of the focal loss, we obtain an asymmetric probability-shifted focal loss. Finally, the Asymmetric Loss is defined by integrating both mechanisms of asymmetric focusing and probability shifting into a single formula:

$$ASL = \begin{cases} L_+ = (1-p)^{\gamma_+} \log(p) & \text{for positive samples} \\ L_- = (p_m)^{\gamma_-} \log(1-p_m) & \text{for negative samples} \end{cases} \quad \textbf{3}$$

In short, the ASL applies two types of asymmetry to reduce the contribution of easy negative samples: soft thresholding via the focusing parameters (where typically $\gamma^- > \gamma^+$) and hard thresholding via the probability margin m.

4.4 | Results

Dataset Processing The `MTANet` was specifically trained on images resized to 96×288 pixels, a dimension choice made to preserve the natural aspect ratio of pedestrian bounding boxes, which is approximately one-third. This aspect ratio maintenance is crucial for ensuring that the pedestrian figures are not distorted, thereby preserving the integrity of visual attributes relevant to the task at hand. The training process was conducted using batches of 64 images. During training, images labeled with -1 in the Mivia Par 2.0 dataset were excluded to ensure the quality and relevance of the training data. Moreover, we employed `Horizontal Flip` augmentation in order to enhance the model's robustness to variations in pedestrian orientation.

However, we consciously chose not to apply brightness or contrast transformations as part of our augmentation strategy to avoid altering the intrinsic color properties of the subjects in the images.

Optimizer The optimization of the network was conducted using the `AdamW` optimizer, with a learning rate set to 1×10^{-4} and a weight decay of 1×10^{-3}. AdamW is an extension of the Adam optimizer that incorporates weight decay separately from the learning rate, offering a more effective regularization method and potentially leading to better training stability and model performance.

Scheduler To manage the learning rate throughout the training process, we employed the `OneCycleLR` scheduler. This scheduler dynamically adjusts the learning rate at each step of every batch, starting with a lower rate, gradually increasing it to a maximum, and then annealing down to a minimal rate. The OneCycleLR approach is designed to help the model quickly converge to a good solution early in training and then refine its parameters for optimal performance.

	Mean	U	L	G	B	H
Loss	0.031	0.046	0.033	0.021	0.041	0.013
Acc	0.925	0.837	0.875	0.977	0.954	0.980

Table 2. Results on the training set (Legend: U = Upper Color, L = Lower Color, G = Gender, B = Bag, H = Hat).

	Mean	U	L	G	B	H
Loss	0.058	0.056	0.036	0.069	0.104	0.026
Acc	0.895	0.828	0.875	0.919	0.897	0.956

Table 3. Results on the validation set (Legend: U = Upper Color, L = Lower Color, G = Gender, B = Bag, H = Hat).

5 | System Evaluation

To evaluate the integrated performance of our system, we developed the `HandleVideo` class. This foundational component is tasked with orchestrating the system's operational logic, enabling a comprehensive assessment of its capabilities. The initiation of a system test is facilitated by executing a specific command-line instruction, as show in Listing 1. This command underscores the necessity of specifying three parameters: `--video`, which delineates the path to the video file designated for testing; `--configuration`, which points to the JSON file containing configurations for ROIs; and `--results`, which identifies the file path where the generated outputs are to be stored.

System Evaluation Command

```
python group08.py --video test.mp4 --configuration configuration.json
↪  --results results.txt
```

Listing 1. Bash command to test the system.

The architecture of the system's logic, encapsulated within the `track_cap` method, is engineered to adaptively process video frames through a mechanism of dynamic frame skipping. For instance, in a video with 30 fps, the system will analyze one frame for every three frames encountered. Such a methodical frame processing regimen is designed to optimize the system's efficiency while ensuring thoroughness in video analysis.

ROI Logic A critical aspect of our system's logic pertains to the ROI Logic. This logic is predicated on the principle that an individual is deemed to be within an ROI if, and only if, the center of their bounding box falls within the defined coordinates of the ROI. The implementation of this logic is manifested in the `handle_roi` method. This method plays a pivotal role in updating the count of individuals detected within the ROI and is structured to accommodate three distinct scenarios for parameter updates: the transition of an individual out of an ROI while remaining in the frame, the absence of an individual from both the ROI and the current frame (which may imply either an exit from the camera's view or a loss of tracking, such as in densely populated environments), and

Fig. 10. Example of application of the system.

the conclusion of the video, necessitating an update of parameters for all individuals last observed within an ROI.

PAR Update Logic The PAR update logic within our system is engineered through the use of a Priority Queue mechanism. This approach is strategically designed to address and mitigate potential inaccuracies in attribute recognition that may arise from singular, momentary observations. Instead of reflecting attributes detected at a specific instant, both the resultant file and the Graphical User Interface (GUI) designated for real-time attribute updates display attributes that have been consistently recognized over multiple observations. This methodology enhances the robustness of the attribute recognition process, significantly reducing the likelihood of false negatives. However, while this approach offers substantial benefits in enhancing attribute recognition accuracy, it may also introduce specific challenges or limitations. For instance, the reliance on cumulative observations over time may delay the recognition of sudden changes in a pedestrian's attributes, such as when a person puts on or removes a piece of clothing or an accessory.

References

[1] Baruch, E. B., Ridnik, T., Zamir, N., Noy, A., Friedman, I., Protter, M., and Zelnik-Manor, L. (2020). Asymmetric loss for multi-label classification. *CoRR*, abs/2009.14119.

[2] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. *CoRR*, abs/1602.00763.

[3] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430.

[4] Greco, A. and Vento, B. (2023). Par contest 2023: Pedestrian attributes recognition with multi-task learning. In *20th International Conference on Computer Analysis of Images and Patterns: CAIP 2023*. Springer.

[5] Pereira, R., Carvalho, G., Garrote, L., and Nunes, U. J. (2022). Sort and deep-sort based multi-object tracking for mobile robotics: Evaluation with new data association metrics. *Applied Sciences*, 12(3).

[6] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402.

[7] Woo, S., Park, J., Lee, J., and Kweon, I. S. (2018). CBAM: convolutional block attention module. *CoRR*, abs/1807.06521.