Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-19356-9

A computer vision system for detecting motorcycle violations in pedestrian zones

Nicolás Hernández-Díaz[1] · Yersica C. Peñaloza[2] · Y. Yuliana Rios[3] · Juan Carlos Martinez-Santos[1] · Edwin Puertas[1]

Received: 1 December 2023 / Revised: 18 April 2024 / Accepted: 2 May 2024

Abstract

This paper presents a system that relies on computer vision to identify instances of motorcycle violations in crosswalks utilizing CNNs. The system was trained and evaluated on a novel public dataset published by the authors, which contains traffic images classified into four categories: motorcycles in crosswalks, motorcycles outside crosswalks, pedestrians in crosswalks, and only motorbike outside. We demonstrate the viability of leveraging deep learning models such as YOLOv8 for this purpose and provide details on the training and performance of the model. This system has the potential to enable intelligent traffic enforcement to mitigate accidents in pedestrian zones; to develop the system, a dataset comprising over 6,000 images was amassed from publicly available traffic cameras and subsequently annotated. Several models, including YOLOv8, SSD, and MobileNet, were trained on this dataset. The YOLOv8 model attained the highest performance with a mean average precision of 84.6% across classes. The study presents the system architecture and training process. Results illustrate the potential of utilizing deep learning to detect traffic violations in pedestrian zones, which can promote intelligent traffic enforcement and improved safety.

Yersica C. Peñaloza, Y. Yuliana Rios, Juan Carlos Martinez-Santos and Edwin Puertas these authors contributed equally to this work

✉ Nicolás Hernández-Díaz
nihernandez@utb.edu.co

✉ Edwin Puertas
epuerta@utb.edu.co

Yersica C. Peñaloza
yersica.carrillo@unipamplona.edu.co

Y. Yuliana Rios
yeriosdi@uis.edu.co

Juan Carlos Martinez-Santos
jcmartinezs@utb.edu.co

1 Facultad de Ingeniería, Universidad Tecnológia de Bolívar, Km 1 via a Turbaco, Cartagena 130010, Bolívar, Colombia

2 Facultad de Ingeniería y Arquitectura, Universidad de Pamplona, Km 1 Via Bucaramanga, Pamplona 543050, Norte de Santander, Colombia

3 Facultad Ingenierías Fisicomecánicas, Universidad Industrial de Santander, Calle 9, Carrera 27, Bucaramanga 680005, Santander, Colombia

Keywords IA model · Computer vision · Machine learning · Pedestrian areas · Autonomous traffic control system

1 Introduction

Pedestrian zones, designated by crosswalks, are of utmost importance for ensuring safety in urban areas. Nonetheless, violations of crosswalks by motor vehicles pose a significant risk. In Cartagena, Colombia, motorcycles drive through crosswalks, putting pedestrians in danger. Traffic accidents at crosswalks often happen due to noncompliance with right-of-way rules.

Automated systems for identifying crosswalk violations might allow enforcement and enhance pedestrian safety. Computer vision techniques hold promise for this application by enabling the training of models to recognize diverse objects in traffic camera images. Techniques such as convolutional neural networks (CNNs) demonstrate proficiency in object detection and classification tasks.

The improper use of pedestrian areas by private motor vehicles, especially motorcycles, is one of the leading causes of vehicular recklessness that represents a great danger to society and pedestrians in general. A clear example of this type of recklessness occurs in Cartagena de Indias, a city located in the north of Colombia, which experiences daily the recklessness of motorcycles. These motorcycles invade sidewalks and crosswalks (also known as "zebras"), disrespecting the space and violating the lives of pedestrians due to their irrational driving and lack of road safety education.

Despite the existence of a law that punishes this type of imprudence (law 1801 of 2016, National Code of Police and Coexistence), in the particular case of Cartagena de Indias, the Town has addressed multiple strategies to counteract such imprudence committed in pedestrian areas by motorcycles. For example, in the Cartagena 2020/2023 development plan "Let's Save Cartagena Together", the following strategies have been addressed to counteract such recklessness committed in pedestrian zones by motorcycles. de Cartagena [1] and in the 2021 action plan [2] of the Administrative Department of Transit and Transportation - DATT, different actions are evidenced that seek to reduce road accidents, taking as a strategy the implementation of training programs in road education. However, recklessness on the part of motorcyclists and pedestrians is still in force.

A strategy that would contribute to reducing recklessness in Cartagena de Indias by motorcyclists in pedestrian zones corresponds to using technologies to generate value-added information that would allow local authorities to exercise control and impose sanctions on offenders. These technologies use devices such as cameras and processing systems to report new developments. For example, some research highlights the importance of establishing smart roads for real-time monitoring of events such as traffic or accidents [3]. However, there is no evidence of studies or solutions to detect violations by motorized vehicles in pedestrian areas.

The main objective of this project is to develop an intelligent system to strengthen traffic regulation monitoring in Cartagena. This innovative system uses artificial vision techniques combined with machine learning to properly define whether the vehicle or the pedestrian has a preference over the space. Suppose the pedestrian has a choice based on the pedestrian traffic light. In that case, the system determines whether or not the vehicle in question is in the space that corresponds to the pedestrian at that moment.

The paper follows the next organization: Section 2 describes the methodology applied to acquire the dataset. Section 3 presents the analysis and post-processing of the dataset. Section 4 explains the benchmarks of Object Detection algorithms with the dataset, and Section 5 analyzes the results in detail. Finally, Section 6 summarizes our contributions and future works.

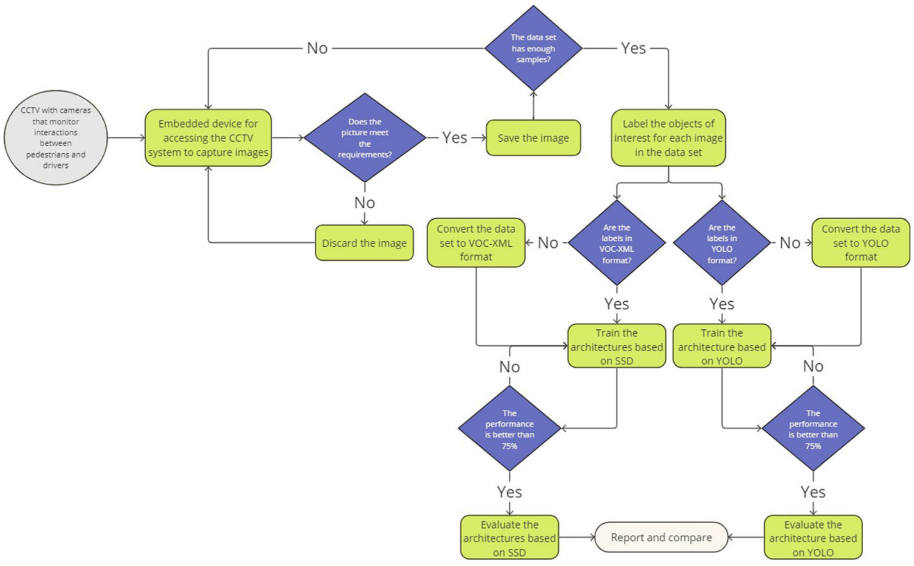
Our contributions presented with this manuscript are,

1. A simple methodology to build a dataset for object detection using a public CCTV.
2. A way to solve unbalanced datasets.
3. A dataset for future work related to this field.
4. Some benchmarks using object detection algorithms like ***YOLO V8n***, ***SSD + VGG16***, ***SSD + MobileNetV1*** and ***SSD + MobileNetV2***.
5. A model based on ***YOLO V8n*** with a 84,5% of accuracy able to classify 4 classes, **M**otorcycle with **M**otorcyclist **N**o in **A**rea for **P**edestrians, **M**otorcycle with **M**otorcyclist in **A**rea for **P**edestrians, **M**otorcycle without motorcyclist **N**o in **A**rea for **P**edestrians and **P**edestrian in **C**ross for **P**edestrians.

2 Methodology

This section describes the work environment used, the elements that comprise it, the data acquisition, and the methodology developed for the construction of the Data Set and training of the IA model; the methodology follows works like [4–6] and pretend to provide to the reader with a more specific perspective of the workspace and the focus of the project.

Figure 1 roughly describes the procedure to obtain an IA model that satisfies the project's objective.



CCTV with cameras that monitor interactions between pedestrians and drivers
Embedded device for accessing the CCTV system to capture images
Does the picture meet the requirements?
Yes
No
Save the image
Discard the image
The data set has enough samples?
No
Yes
Label the objects of interest for each image in the data set
Are the labels in VOC-XML format?
Are the labels in YOLO format?
No
No
Convert the data set to VOC-XML format
Convert the data set to YOLO format
Yes
Yes
Train the architectures based on SSD
Train the architecture based on YOLO
No
No
The performance is better than 75%
The performance is better than 75%
Yes
Yes
Evaluate the architectures based on SSD
Report and compare
Evaluate the architecture based on YOLO

Fig. 1 General methodology to follow

2.1 Data set construction

This work is an extension of the work in [7], so to have a better background about how we built the dataset, please double-check the work cited.

2.1.1 Work environment

About the work environment and the source of the data, it is necessary to clarify that since the object of the project is the development of an artificial intelligence system for the detection and identification of objects in pedestrian areas, it is urgent to determine the types of objects to be classified, always ensuring that they belong to the set of objects near or on pedestrian areas and are related to the detection of motorcyclist offenders.

As a result of this discussion, it is brought to light that the desired type of images corresponds to those containing the following objects:

1. **MMNAP** $<-$ **M**otorcycle with **M**otorcyclist **N**o in **A**rea for **P**edestrians
2. **MMAP** $<-$ **M**otorcycle with **M**otorcyclist in **A**rea for **P**edestrians
3. **MNAP** $<-$ **M**otorcycle without motorcyclist **N**o in **A**rea for **P**edestrians
4. **PCP** $<-$ **P**edestrian in **C**ross for **P**edestrians

They are also available in the context of national use, that is to say, in Colombia, considering that as a future work, the local government plans to implement this intelligent system in Cartagena, Colombia.

While searching for possible images that met these criteria, we reviewed public and open-access servers such as InseCam and Google Maps (Street View). We made attempts to talk to government entities for traffic control.

From this last approach, we found that Medellín has a pioneer road project that offers access to a CCTV server called SIMM, which is public and easily accessible.

SIMM project Medellín Mayor's office The SIMM (Sistema Inteligente de Movilidad de Medellín) project is a pioneer program in Colombia of the Secretary of Mobility of Medellín, which emerged in 2020 in response to the need to optimize the city's road network; this system has public access closed circuit television system composed of 80 cameras arranged in strategic areas that agents from the Traffic Control Center operate. Thanks to this component, any person with internet access can check the state of the roads in real-time and plan their movements along the different routes [8].

The nature of the data represents a challenge for its proper acquisition because access to the CCTV server is accessible. There is a unique URL to access each camera, and the information update in the server occurs every 5 minutes and asynchronously between the available cameras.

Elements that make up the work environment Figure 2 shows a schematic of the proposed methodology and its elements. As we can discern, the system depends on the availability and status of the data present in the CCTV. The request for information (RGB Image) is made to the URL of each camera independently through a Raspberry Pi Zero 2W with an Internet connection.

This embedded device, using a "Comparative Chronological Analysis", downloads the images in .jpg format to build the raw dataset, either during the time indicated by the user or until the execution is interrupted.

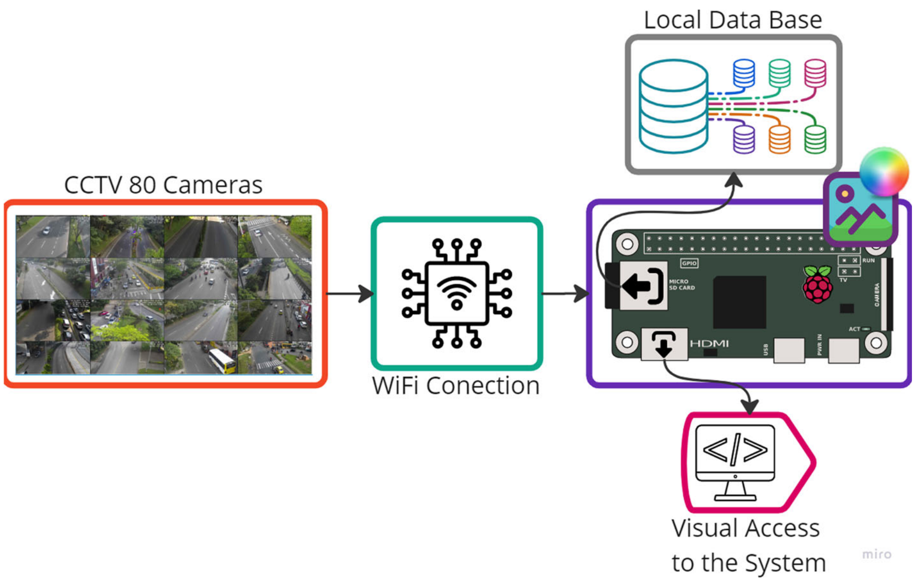


Fig. 2 Methodology implemented to acquire data

This data is stored locally on the micro SD card,sharing space with the Raspberry Pi operating system; since the weight per image is less than 120 KB, a 32 GB SanDisk class 10 card is more than enough.

2.1.2 Data acquisition

The methodology developed for constructing the data set necessary for the project consists of two main lines of execution. However, only the line of image acquisition from the CCTV server uses a Comparative Chronological Analysis (1) to determine whether it is necessary to download the image observed by the nth camera at an instant t of time.

The first line follows the next equation,

$$ACC(A, B) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(|A_{(i,j)} - B_{(i,j)}|\right)_{(i,j)} \tag{1}$$

Where:

- $ACC(A, B)$ Comparative Chronological Analysis (Scalar Value)
- n is fxc the number of rows per the number of columns of the matrix.
- (i, j) the indexing state over the image's matrix.
- A the current image observed by the nth camera.
- B the previous image stored by the nth camera.

And,

$$BD = \begin{cases} True & \text{if } ACC(A, B) > threshold \\ False & \text{if } ACC(A, B) \leq threshold \end{cases} \tag{2}$$

Where:

- BD Flag to develop the download.
- $threshold$ threshold that establishes the criterion of download.

Equation 1 seeks to establish the level of the pixel's color and how different the previous image is from the next one for the same data source. The result is a scalar value that the system compared with a threshold using the (2). This comparison gives us a simple boolean value that acts as a flag to indicate when the system needs to download an image and when, in contrast, it is necessary to discard it.

The second line consists of the labeling process using rectangular-type ROIs and hotkeys in conjunction with the verification process of the labels performed.

The second line of execution follows the next,

$$Asig = \begin{cases} MMNAP & \text{if } keyState{=}{=}A \\ MMAP & \text{if } keyState{=}{=}B \\ MNAP & \text{if } keyState{=}{=}C \\ PCP & \text{if } keyState{=}{=}D \\ continue & \text{if } keyState{=}{=}Q \end{cases} \tag{3}$$

Where:

- $Asig$ corresponds to the path within a dictionary where the snippet, the ROI coordinates, and the class label should be stored.
- $MMNAP$, $MMAP$... are the distant classes recognized during the project.
- $keyState$ stored the hotkey pressed by the user.
- $continue$ indicates to the system that the information of the general sample does not contain classes or that there are no more classes to be classified manually.
- $[A, B, C, D, Q]$ represent the hotkey assigned to each class in order.

The (3) allows the user via a key's state (Keyboard's Keys) to indicate during the labeling process to which one class belongs the nth object marked using an ROI and present inside the nth image.

The work in [9] explains the Comparative Chronological Analysis and the labeling process using rectangular-type ROIs and hotkeys.

2.2 Object detection process

The state-of-the-art [10–14] presents AI models based on CNNs and transformations with very high performances. However, the results obtained are a function of the data set applied, i.e., in general, each author uses a generic or pre-developed data set, which does not allow establishing whether it is possible to guarantee which model is the most appropriate. In addition, their implementations do not seek to be applied in low-resource embedded devices, as this is one of the objectives planned as future work for the project. Therefore, we proposed the following criteria that seek to define the set of models applied in the project,

1. How well documented is the model for its reproducibility?
2. What computational cost does the model require for its respective training?
3. Is this model designed to perform inference under embedded low computational resources?
4. Is the format in which the model requires the data conventional?
5. Does the model have any pre-trained model to apply transfer-learning [15] and the reported performance is high?

Based on these five elements and according to those reported in the state of the art [16–18], the following architectures are selected and reproduced, [19, 20]

1. YOLO version 8 (Formatting: YOLO)
2. Single Shot Multibox Detector with VGG16 (Formatting: VOC) [21]
3. Single Shot Multibox Detector with MobileNet V1 (Formatting: VOC) [22]
4. Single Shot Multibox Detector with MobileNet V2 Lite (Formatting: VOC) [23, 24]

Each one of these architectures will be re-trained by applying transfer learning since training from scratch with so little data is not very convenient due to the available models already knowing millions of data and perfect understanding the nature of different detection environments; the SSD pre-trained models applied are available in [25] and their respective documentation and code are available in [26], the model and the code for YOLO is available in [27].

3 Processing and analysis of the data set

While analyzing the data obtained, a common problem when working with data acquired from a non-artificial environment is that the data needs to be more balanced initially.

The first data set obtained contained 11361 images in .jpg format. The original size was 99GB. The resolution of each image was 1280x720 (HD). The photos come from the same CCTV server captured during night and day hours. The time range for the acquisition is from 04/10/2022 to 28/10/2022, a total of 24 days. However, another class different from the original was considered during the annotation process, as shown in Fig. 3 and (4).

$$Asig = \begin{cases} MMNAP & \text{if } keyState{=}{=}A \\ MMAP & \text{if } keyState{=}{=}B \\ MAP & \text{if } keyState{=}{=}C \\ MNAP & \text{if } keyState{=}{=}D \\ PCP & \text{if } keyState{=}{=}E \\ continue & \text{if } keyState{=}{=}Q \end{cases} \tag{4}$$

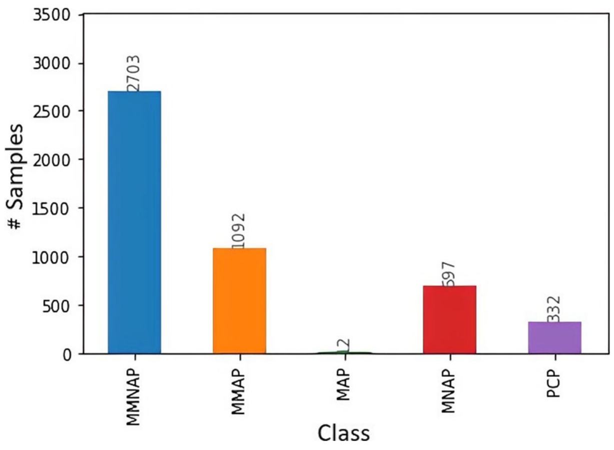


Fig. 3 First distribution of the DataSet

Where:

1. MAP **M**otorcycle without **M**otorcyclist in **A**rea for **P**edestrians

We assigned this class in the context of the project in the 3 position of the group of classes and called MAP (**M**otorcycle without **M**otorcyclist in **A**rea for **P**edestrians), this class was considered as a fundamental class for the project. However, there was an issue when we finished the post-processing due to the low frequency of class samples (see column 3 Fig. 3), precisely 12 samples. It was impossible to balance the whole dataset to fit this outlier. Therefore, we have decided to abandon this class.

We mentioned and explained that, until this point, without the class MAP, the data set can't correctly generalize the approach of this research, considering that a motorcycle blocking the pedestrian cross without a motorcycle is part of the violations in pedestrian zones. Hence, we expect to add this class as future work and improve the ML model behavior.

It is why Fig. 4 doesn't present this class; now, another problem is if we review Fig. 3, we can see that the DataSet is unbalanced.

Algorithm 1 Algorithm to apply the simplified Down-Sampling.

```
Input: x    ▷ Dictionary which contains the boundary boxes and its classes of the nth object in the nth image.
Output: nuevas_claves    ▷ Array which contains the new tabulated names of the images.

1: count_clases ← [0, 0, 0, 0, 0]    ▷ Array that allows to store the count of the number of samples per class
2: nuevas_claves ← []    ▷ DataSet obtain after to apply Down-Sampling to x
3: for $i \in$ x.keys() do
4:     if sum(x[i]["clase"]) $\neq 0$ then    ▷ ***
5:         nuevas_claves.append(i)
6:         for $j \in$ x[i]["clase"] do
7:             count_clases[j] += 1    ▷ Count of the samples per class
8:         end for
9:     end if
10: end for
```

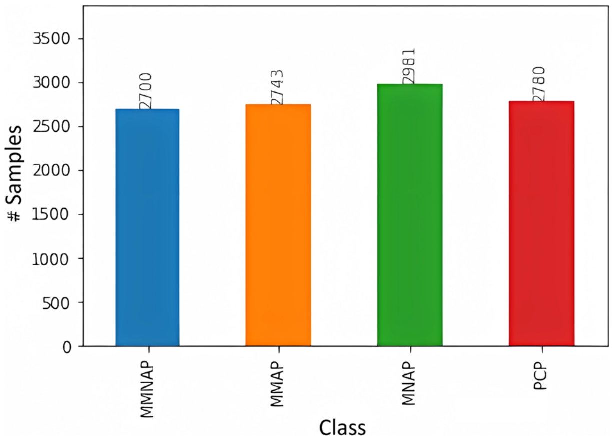


Fig. 4 Final distribution of the DataSet

According to the literature, one way to mitigate this situation is by applying techniques such as Down-Sampling, Up-Sampling, or similes; since the problem presented by the data set is an overflowing frequency in the $MMNAP$ class concerning the other classes, we do it to reduce by Down-Sampling the number of samples of said class regarding a reference that allows balancing the general behavior of the classes; the Algorithm 1, that is a simplification of the original technique allows explaining the process performed and the objective achieved.

The comment *** in the Algorithm 1 indicates that if on the **nth key** that represents the name of the **nth image**, only the class **0 = MMNAP** is present, this image is not contemplated in the **set of new keys**, since this effect means that the contribution of the **nth image** for the rest of the classes other than **0** does **not contribute to the frequency of samples reported by the different classes, it forces to the original data set to reduce proportionally".**

After applying the Algorithm 1, the classes' final sampling and distribution are shown in Fig. 4.

Therefore, the dataset finally developed consists of 6324 images from the 11361 previously stored (just 55.6% of the first data set built). This percentage of images represents the total number of images with at least a different class than the MMNAP class, which, as explained, is the class with the most significant count of samples. For example, suppose the x image only contains objects belonging to the class MMNAP. In that case, the image is disposable because it only contributes samples to the most significant class, increasing the unbalanced situation with the DataSet in this way.

The format of each image is .jpg, the resolution is 1280x720 (HD), and the weight per image is less than 120 KB, so the total weight of the data set is 643.9MB. Most of the images preserved were those taken mainly during daylight hours and can be downloaded as free from [28].

Figure 5 illustrates the correlation between the different samples of each class as a function of the $width$, the $height$, and the centers in x, y coordinates of each notation performed.

split the DataSet at this stage into two groups: a total of 75% of the DataSet (4743) with their respective labels as training data, and the resulting percentage 25% of the dataset (1581) as test, testing, or validation data, the Table 1 summarize the characteristics.

4 Identification and classification of the classes

The dataset obtained by following Section 2.1 is formatted under a pickle format. This formatted file contains an embedded Python dictionary, which isn't compatible with the Object Classifications algorithm listed in Section 2.2; indeed, it is necessary to transform the data set developed from its original format to a VOC XML or YOLO format as appropriate. This task is possible to carry on using different resources available on the internet; to cite some, we have [29–32].

4.1 Yolo V8

According to the model's authors, there are five YOLO models. However, each model requires specific and demanding computational resources depending on the complexity of the model, either for training or inference. We use **Ultralytics** because it is the recommendation to run in systems with low computational resources. In terms of the authors, this model is called *YOLOv8n* because it is a model that weighs less than 7MB; it is the smallest of the family.

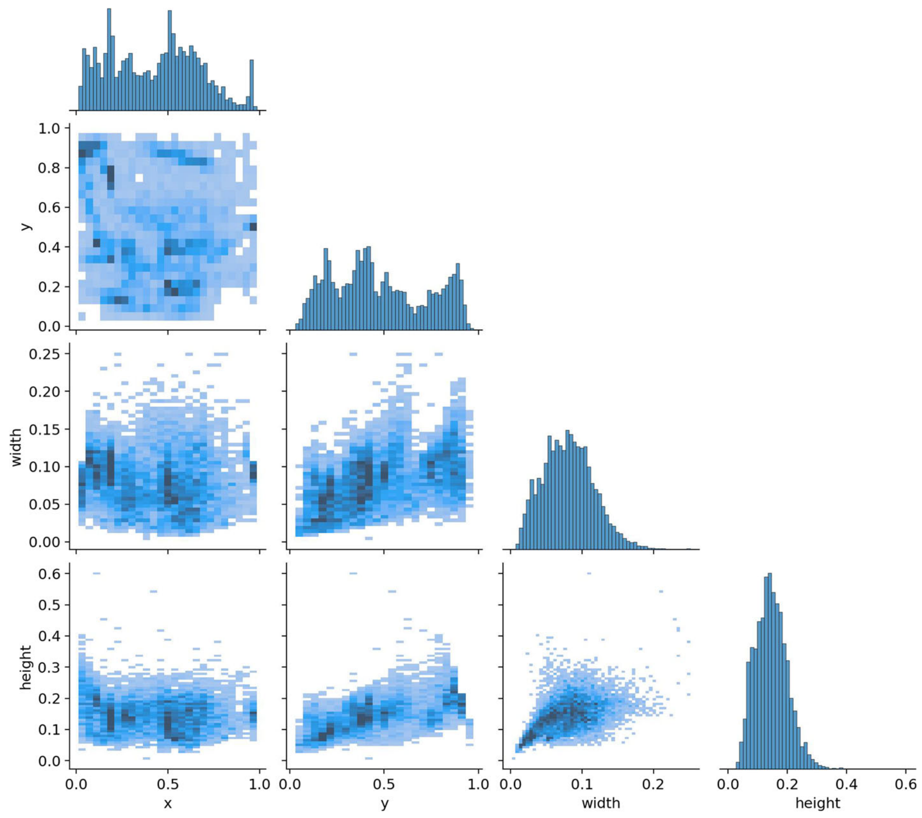


Fig. 5 Correlation of DataSet labels

It presents the characteristics of Table 2, and the type of format this architecture receives is VOC XML.

The model training process is configured in the following way through the command console and using the command ***yolo***;

```
yolo task=detect mode=train epochs=200
data=data\_custom.yaml model=yolov8n.pt imgsz=640
patience=8 workers=5 lr0=0.01 batch=16 optimizer=Adam
```

Table 1 Summary of dataset characteristics

Labels/Item	MMNAP	MMAP	MAP	PCP	Total Samples
Train	2025	2057	2236	2085	8403
Test	675	686	745	695	2801
Total Samples Per Label	2700	2743	2981	2780	11204

Table 2 YOLO's Model Used

Model	Size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7

About the configuration of the model, the value of the applied ***batch*** is 16. Therefore, we reviewed if the training layers delivered to the architecture, what is shown in Fig. 6 is observed.

The training lasted about **58.26 minutes**. It ended because there was no improvement in the model's performance during eight consecutive epochs, culminating the training strictly at **epoch 63**. Therefore, we found the best performance presented at **epoch 55**. The results are summarized in Table 3.

The **recorded validation time was 23.7 [s]**, **it is 15 [ms] per image**, and the ***Validation Loss obtained was 1.607*** reported under the best epoch obtained.

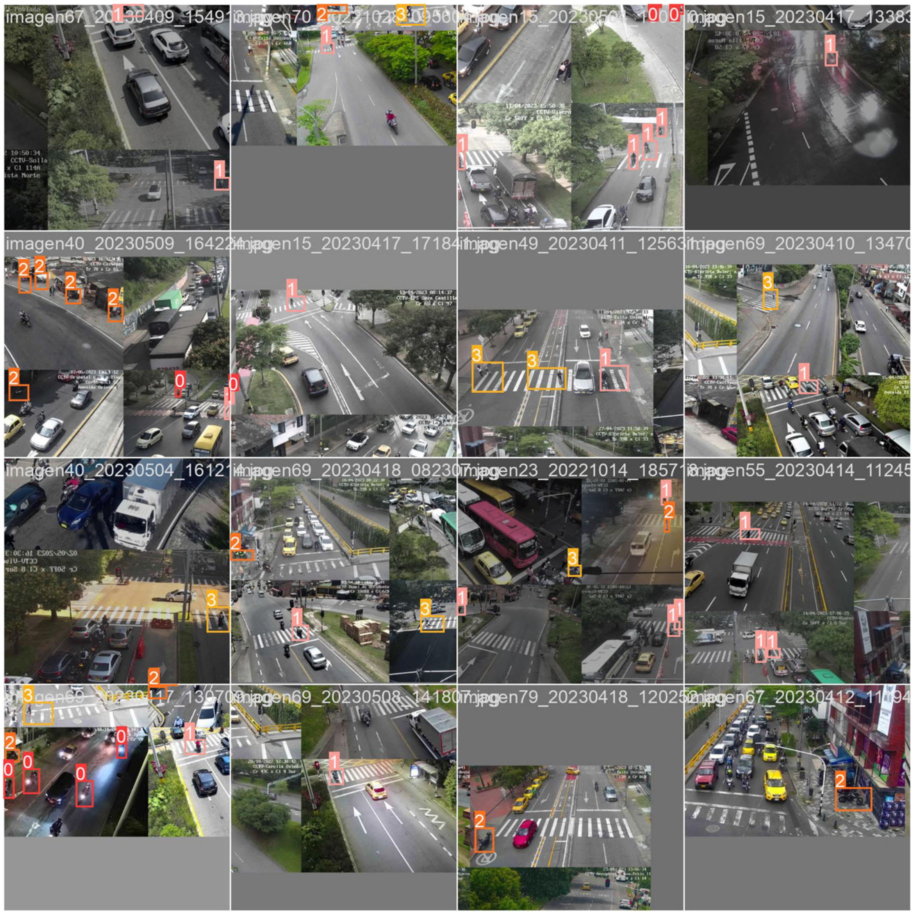


Fig. 6 Nth training batch for YOLO

Table 3 Results of the YOLO model obtained after evaluation

Class	Images	Instances	Box(P	R	mAP50)
all	1581	2836	0,809	0,797	0,846
MMNAP	1581	715	0,805	0,745	0,826
MMAP	1581	699	0,83	0,805	0,849
MNAP	1581	737	0,789	0,883	0,881
PCP	1581	685	0,812	0,753	0,829

The Complete results and Precision-Recall curve of the model obtained are presented in Figs. 7 and 8, respectively.

Based on the results, the YOLO Version 8 model is taken as a reference or baseline to compare the performance of the remaining models. Hence, we set the parameters following the configuration conditions established in YOLO.

```
mode=train epochs=200 workers=5 lr0=0.01 batch=14
                tmaxToTrain=1 hour optimizer=Adam
```

The parameter ***tmaxToTrain*** is set at 1 hour, assuming that with the YOLO architecture, the time used for training was one hour. The expected results were achieved, in addition to setting up a maximum training time for the rest of the models, allowing us to explain under equal conditions which model presents the best performance and which one, on the contrary, is the worst.

However, it is worth mentioning that, as we can see inside the box above, the parameter **batch** has changed. It is due to limitations of Hardware, specifically the memory *VRAM* of the central processing equipment to train the models; in practice, this seems to be because the architectures based on SSD require a better VRAM to store, a batch of images is greater

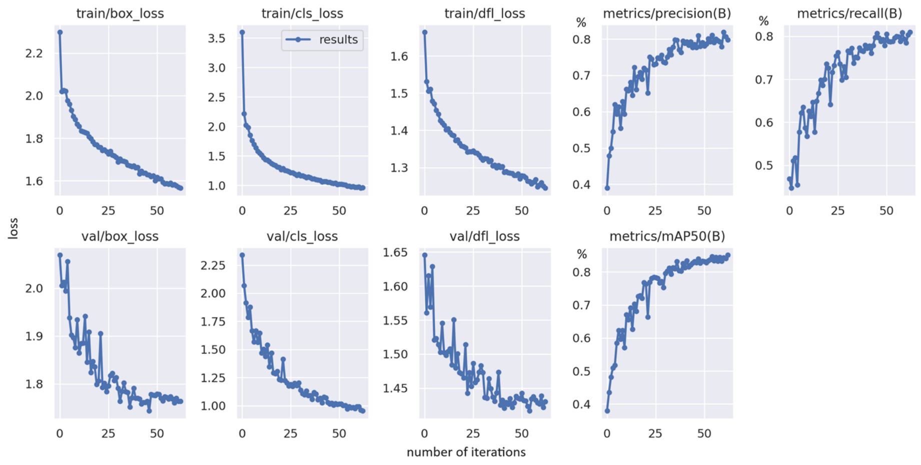


Fig. 7 Complete results, YOLO model obtained

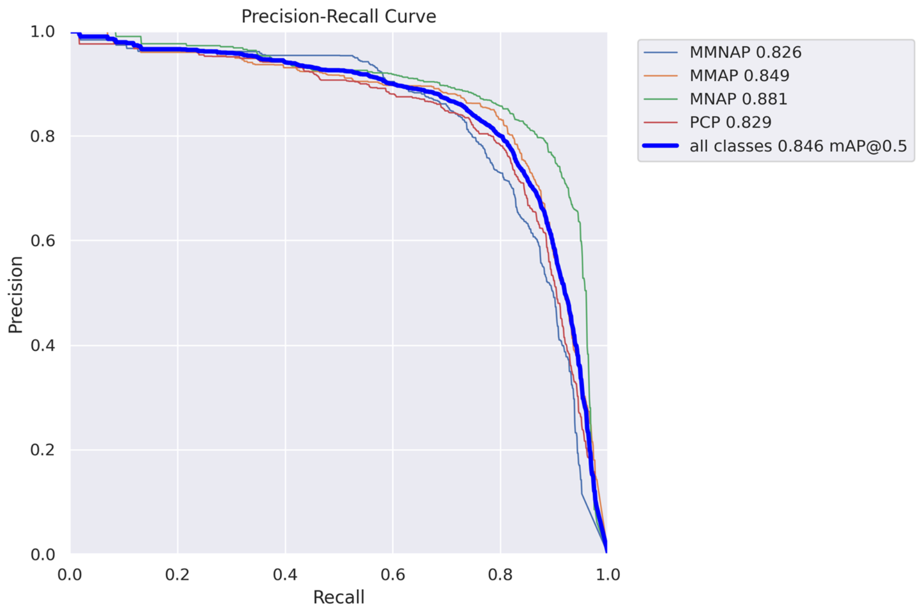


Fig. 8 Precision-Recall curve of YOLO's model obtained

than 14 can't be processed in the GPU side, forcing the training's stoppage; of course, this only applied for our case of study and limitations.

So, according to this section, the parameters used for the whole experiment analysis are described in Table 4.

4.2 SSD-based architectures

The next step is to check the performance of each object detector based on SSD, so the idea is,

- First Extract feature maps,

Table 4 Summary of the parameters used for the analysis

Architecture/Parameters	YOLO Versión 8	VGG16 + SSDLite	MobileNetV1 + SSDLite	MobileNetV2 + SSDLite
Task	Detection	Detection	Detection	Detection
Epochs	200	200	200	200
Image Size	640	300	300	300
Patience	8	8	8	8
Workers	5	5	5	5
Initial Learning Rate	0.01	0.01	0.01	0.01
Batch	16	14	14	14
Optimizer	Adam	Adam	Adam	Adam
Max Time to Train (min)	60	60	60	60

- then apply convolutional filters to detect the objects in the nth image.

SSD uses techniques other than Region Proposal Network or SuperPixels, which require less time to perform. Instead, it resorts to a straightforward method. It computes location and class scores using small convolution filters from image classifiers. It is why we can use architectures like VGG, MobilNet, SqueezeNet, and similar to detect objects.

4.2.1 SSD+VGG16

The first architecture tested for this subset of architectures based on SSD is the SSD+VGG16. This architecture presents the training results shown in Fig. 9; as we can see, this model reached a validation loss value of 3.599. This value is the best validation loss value for the whole SSD-based architecture. However, it was able to get this result in only 11 epochs, which is fewer epochs than the other models here during the same time for training, indicating the architecture converges better. Still, the time it takes to finish one epoch of training is the greatest here obtained.

Table 5 presents the behavior of this model after evaluating the validation data set; the whole process took exactly 56 seconds to process the total amount of data, 1581 images, which is the most significant time obtained during the evaluation process, it means that the model is slower for inference than the other SSD-based models analyzed. The best performance was 49.7%, the second-best result obtained in the study.

4.2.2 SSD + MobilNetV1

We trained the following SSD architecture based on MobilNetV1, which is a CNN network developed to run inside mobile devices, especially in cell phones; this architecture, although

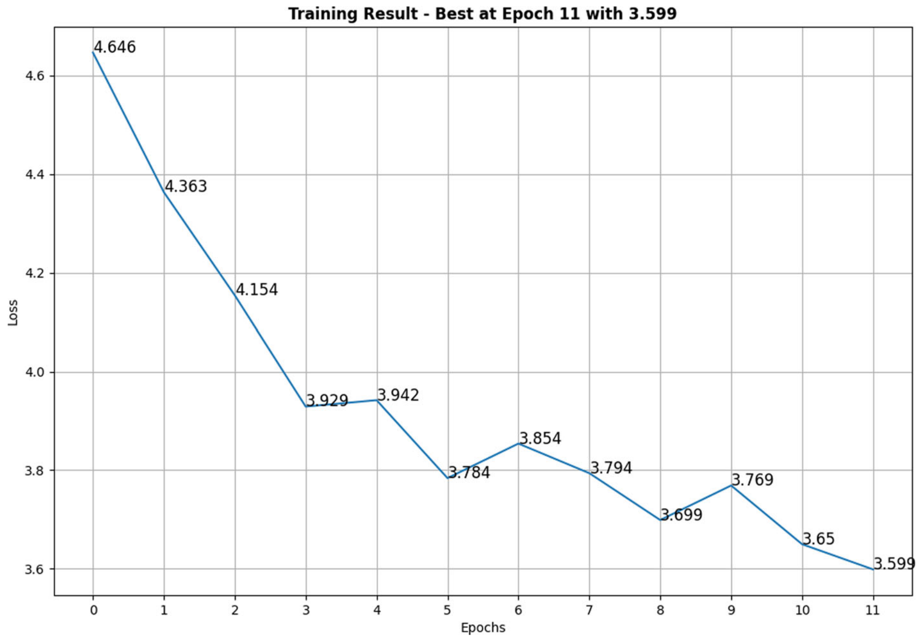


Fig. 9 SSD + VGG16 model training results

Table 5 Results after evaluating SSD + VGG16 architecture

Class	Images	Instances	mAP50
all	1581	2836	0,497
MMNAP	1581	715	0,198
MMAP	1581	699	0,515
MNAP	1581	737	0,673
PCP	1581	685	0,601

is the lightest one compared with the other SSD-based architectures here analyzed, due to it being able to reach 35 epochs, the validation loss obtained is worse than the previous architecture with 3.61, which explain why usually this architecture is used with mobile devices (Fig. 10).

Table 6 shows the results after evaluating the model; the overall performance is 40,7%, and the time spent for inference was 53 seconds. These results are located in the middle of the previous and the next architecture and make sense, considering that the next architecture is the second version of this one.

4.2.3 SSD + MobilNetV2

This architecture presents some enhancements to the previous one. The results show these improvements, with 33 epochs reached and a validation loss value of 3.653 as is illustrated in Fig. 11, this architecture seems to be able to achieve better performance and convergence with much more time of training, otherwise this architecture is almost the same than the

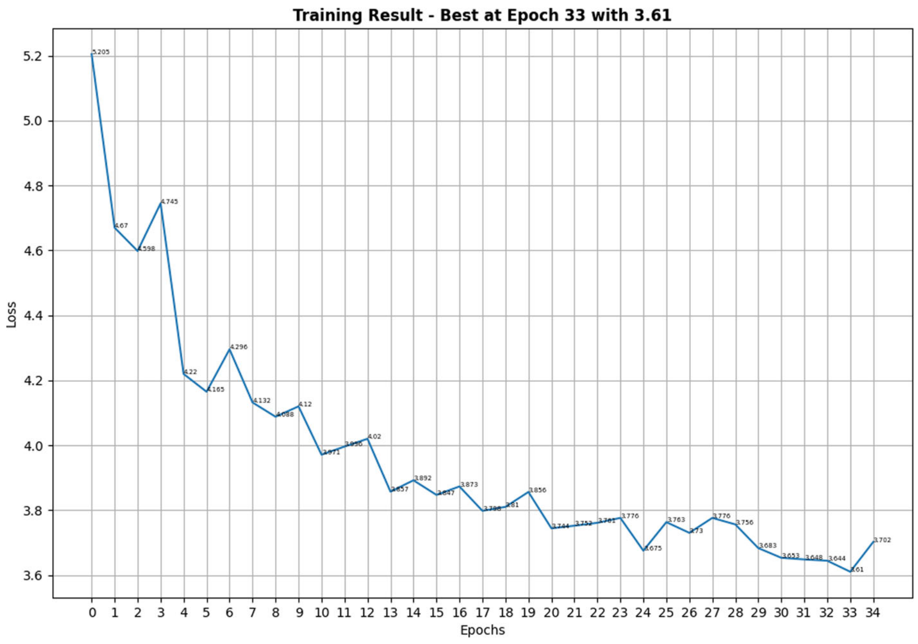


Fig. 10 Results obtained with SSD + MobilNetV1 architecture

Table 6 Results after evaluating SSD + MobilNetV1 architecture

Class	Images	Instances	mAP50
all	1581	2836	0,407
MMNAP	1581	715	0,194
MMAP	1581	699	0,398
MNAP	1581	737	0,474
PCP	1581	685	0,563

previous one with the unique difference that according to the Table 7 and the experiment done the inference time is better with 47 seconds, which is the best time obtained for this SSD-based architectures.

5 Summary

Table 8 presents the specific results concerning each model applied and the results obtained after evaluating the subset of test data (1581) images.

Notably, YOLO Version 8 exhibited the longest training time, taking approximately 3495.6 seconds to complete, while MobileNetV2 + SSDLite was the fastest, completing training in 3393 seconds. This variance in training times reflects differences in model complexity and computational requirements.

Analyzing the training process further, YOLO Version 8 had the longest time per epoch at 55.5 seconds, compared to MobileNetV2 + SSDLite's 102.6 seconds. Despite this, YOLO

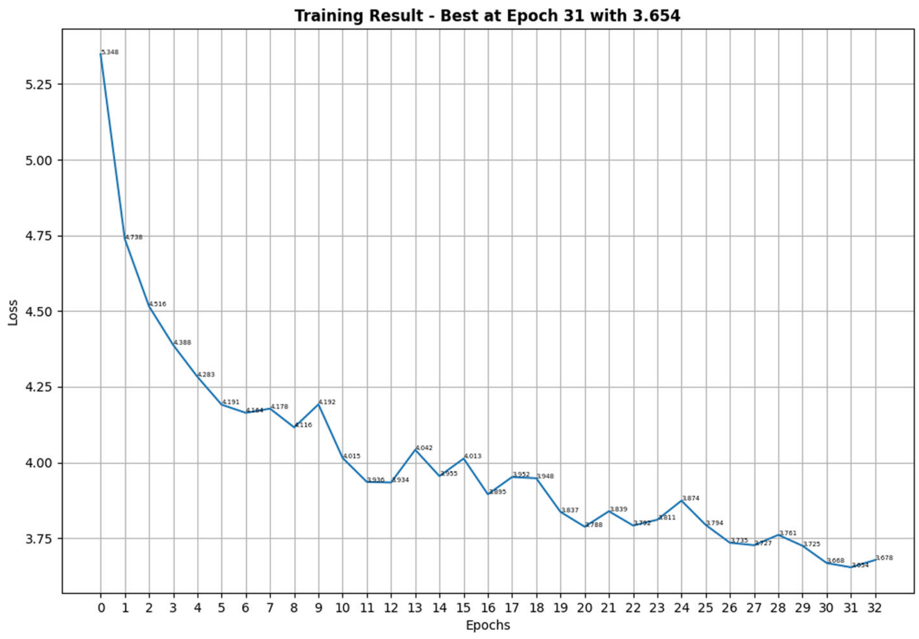


Fig. 11 Results obtained with SSD + MobilNetV2 architecture

Table 7 Results after evaluating SSD + MobilNetV2 architecture

Class	Images	Instances	mAP50
all	1581	2836	0,410
MMNAP	1581	715	0,175
MMAP	1581	699	0,417
MNAP	1581	737	0,499
PCP	1581	685	0,541

Version 8 required the fewest epochs to achieve convergence, reaching the best performance in 63 epochs, followed by MobileNetV2 + SSDLite at 33 epochs.

In terms of evaluation time, YOLO Version 8 again demonstrated efficiency, taking only 23.7 seconds, while VGG16 + SSDLite required 56 seconds. This discrepancy in evaluation times may impact the practical deployment of these models in real-time applications.

Additionally, we provide the names of the custom models saved during training, along with their corresponding validation losses. YOLO Version 8 achieved the lowest validation loss of 1.607, indicating superior performance compared to the other models. This is further corroborated by the performance metrics, with YOLO Version 8 attaining the highest 50@mAP score of 84.6%, outperforming VGG16 + SSDLite, MobileNetV2 + SSDLite, and MobileNetV1 + SSDLite, which achieved 49.7%, 41.0%, and 40.7% respectively.

From the results recorded in Table 8 cyanit is possible to affirm that the best model obtained corresponds to the model based on YOLO, since it presents a performance of 84.6% with a total inference time of 23.7 [s] and its inference time per image is 15 [ms], in addition, it perfectly meets the search criteria set out in the project, since it is a lightweight model, compatible with low-resource embedded devices and its implementation, thanks to the large community behind its development, is available and very well documented for different operating systems, compared to the other models studied.

Finally, we indicated that the model ***best.pth*** based on YOLO V8n. The architecture presented in Table 9 corresponds to the final product of this project. We anticipate a possible future use under a real-time offender detection environment.

Table 8 Summary of project results

Model/Results	YOLO Version 8	VGG16 + SSDLite	MobileNetV1 + SSDLite	MobileNetV2 + SSDLite
Time's Train [s]	3495,6	3437	3544	3393
Time per Epoch [s]	55,5	310	107,4	102,6
Got it Epochs	63	11	35	33
Best Epoch	55	11	33	31
Time's Eval [s]	23,7	56	53	47
Time per Image [ms]	15	35,42	33,52	29,73
Name's Custom Model	best.pth	vgg16-ssd- Epoch-11-Loss- 3.599.pth	mb1-ssd-Epoch-33- Loss-3.61.pth	mb2-ssd-Epoch-31- Loss-3.653.pth
Validation Loss	1,607	3,599	3,61	3,653
Performance 50@mAP	84,6%	49,7%	40,7%	41,0%

Table 9 Layout of the applied YOLO architecture

ID	From	n	Params	Module	Arguments
0	-1	1	464	ultralytics.nn.modules.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.Concat	[1]
21	-1	1	493056	ultralytics.nn.modules.C2f	[384, 256, 1]
22	[15, 18, 21]	1	752092	ultralytics.nn.modules.Detect	[4, [64, 128, 256]]

A ***batch*** of the evaluation data is tested by asking the best model obtained to perform the object detection process on 16 consecutive images. The results are in Fig. 12.

The confusion matrix after the evaluation is illustrated in Fig. 13; from the figure, it is possible to state that the model is sufficiently robust with an overall performance of **84.5%** and the minimum reported performance corresponds to the $MMNAP$ class with a total of **78%**.

5.1 Discussion

Regarding the results achieved through the model obtained based on YOLO, it is possible to ensure that the identification and classification of the different classes proposed in the project have succeeded. It results from a systematic process of evaluating different algorithms and strategies until reaching an accuracy more significant than 75%. However, the main objective of this project is to contribute to reducing accident rates, specifically in crosswalks. Therefore, we must discuss what we currently have and how we expect to generate some positive impact on crosswalks and their respective use.

Although we are talking about a second stage of this project because we only have a general idea of the implementation scheme, it is feasible to indicate that, according to the

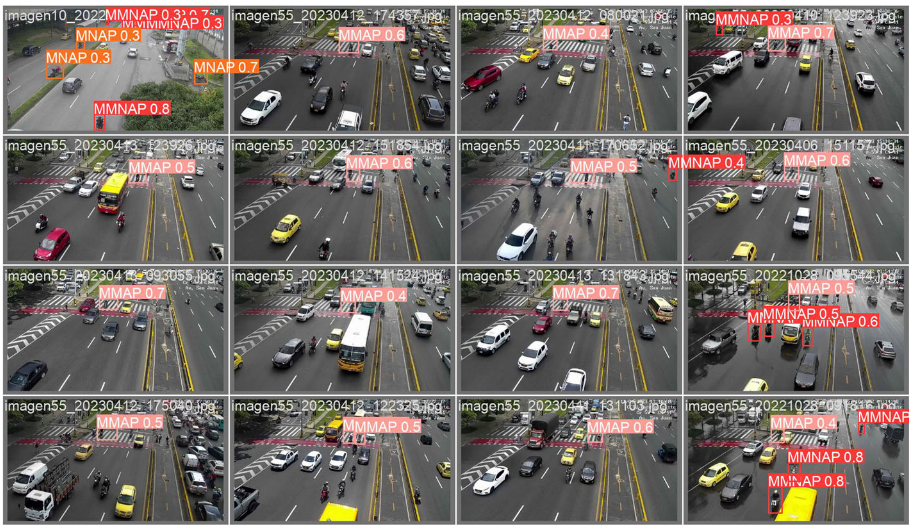


Fig. 12 YOLO model results from over 16 images

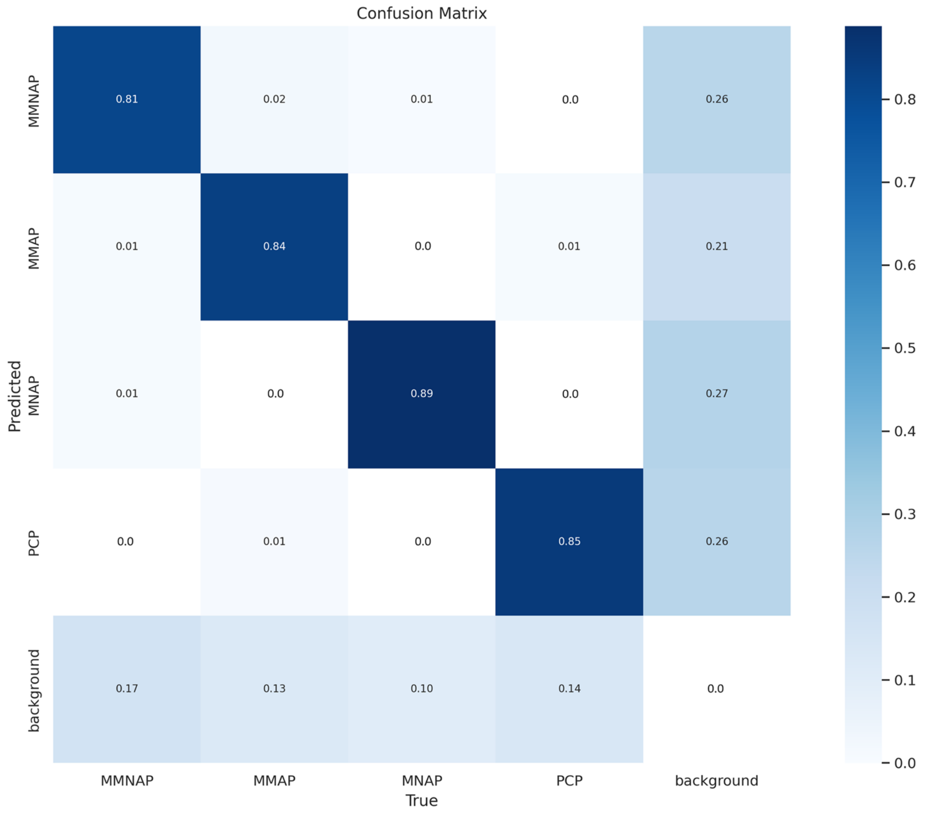


Fig. 13 Confusion matrix after evaluating the YOLO model obtained with the test data set

criteria listed in the model selection, we expect to integrate the model in an embedded device such as the Raspberry Pi or similar devices, it would facilitate the processing of the data in situ, i.e., on a real vehicular and pedestrian traffic light environment where the same device as a control unit is responsible for both feeding back the traffic light times and reporting the current status of the crosswalk depending on the detection of a violation.

Now, the question is deciding when a class goes from being a pedestrian or motorcyclist making regular use of the crosswalk to being a violator; in the context of use, a variable that depends entirely on the state of the traffic light at an instant *t* of time, since it is not the same a motorcycle crossing the crosswalk when the traffic light is redder than a pedestrian on the crosswalk when the pedestrian light is red, so it is necessary to define which object or class represents priority during the decision-making process to determine which traffic light to manipulate and when to generate an infraction state.

The next question is, what is in the state of the art about the concept presented in this work? Table 10 shows a comparative analysis of some works available in the literature and the results obtained through our research. As we can see, the literature doesn't present a multi-class algorithm for object detection. Even if they aren't worried about running the model under embedded solution for practical purposes, they are focused only on detecting the pedestrian discarding other elements that interact directly and indirectly with the pedestrian while the pedestrian is trying to use the pedestrian cross, of course, the accuracy reported compared with ours is better, however, it is only for one of the four classes here analyzed and they ignore a possible integration of this model with a real and integrated surveillance system.

5.2 Limitations of the study

The PC used to train and evaluate the different models planned is a ***Lenovo Ideapad Gaming 3 with an Intel Core i5, Geforce RTX3050 with a VRAM of 4GB and a RAM of 8 GB*** running Ubuntu 22.04.3.

About the model obtained and according to the lowest performance recorded (***MMNAP*** class with 78%), technical limitations at the hardware level represent a problem for this type of architecture that require computing units with very high technical characteristics that allow

Table 10 Comparative analysis with existing methods

Algoritms	Accuracy [%]					Time per Image [ms]
	Overall	MMNAP	MMAP	MNAP	PCP	
Literature [10]	0	0	0	0	100	19
Literature [12]	0	0	0	0	100	NR
Literature [16]	0	0	0	0	97,2	NR
Literature [17]	0	0	0	0	65,57	NR
Literature [20]	0	0	0	0	94.8	1.86
Literature [33]	0	0	0	0	90,51	NR
Literature [34]	0	0	0	0	96,8	NR
This Work	84,6	82,6	84,9	88,1	82,9	15

The Labels Represent: NR = No Reported, 0 = No performance reported for this class, MMNAP = Motorcycle with Motorcyclist No in Area for Pedestrians, MMAP = Motorcycle with Motorcyclist in Area for Pedestrians, Motorcycle without motorcyclist No in Area for Pedestrians, Pedestrian in Cross for Pedestrians in less time to reach convergence and improve performance. For the document, we explained that a recurring problem was the capacity of the ***GPU's VRAM*** installed since this was 4GB and the training team closed the session, indicating that it required more memory to continue, this also depended a lot on the size of the data and the type of architecture.

It is only one of the drawbacks that occurred since, as indicated throughout the document, given that the resolution of the images was so low (HD), the samples per class were even lower. It induces error and represents a second problem to address. Since the data quality is so low, the information contained in a sample of one class could be confused with the reference of another class. This situation occurred regularly, even with manual labeling, as there was some uncertainty about the class to which the nth sample belonged.

Other aspects that possibly affected the performance obtained are,

1. ***Environment***, If you look at the images in the data set. However, most of them are during daylight hours; there are images where rain, shadows, changes in light intensity, strong winds, and out of focus, among others, environmental aspects that cause errors when characterizing and classifying a sample.
2. ***Size of the data set***, The constructed dataset only consists of 6324 images and about 2750 samples per class. For this type of model, even when applying transfer learning, this dataset is tiny; therefore, working with datasets at least ten times larger is necessary to achieve a correct generalization based on the data.
3. ***Hiperparameters***, This approach was not explored for the paper, so there is a great deal of uncertainty as to whether a design of experiments where the hyperparameters of the network in question are varied would improve the performance obtained.
4. ***Data Acquisition Frequency***, As previously indicated, the CCTV reported asynchronously every 5 minutes the change of state of the image observed by the nth/80 camera. It does not facilitate having a large set of samples under a high similarity index environment so that, in the end, each sample is unique.

Finally, regarding the applicability of the issue, it is indicated that the model obtained is robust (although the degree of uncertainty is still high). The current embedded devices can make inferences using this model type. In these environments, the problem of the proper use of the crosswalk is reiterative and regular, so, in theory, it would be enough to synchronize the traffic lights to an embedded device that controls the traffic light times and closes the loop between the observed by the cameras and the behavior of both the pedestrian and vehicular traffic lights.

6 Conclusion

In conclusion, this study introduces a novel system leveraging computer vision technology to detect motorcycle violations in pedestrian crosswalks using Convolutional Neural Networks (CNNs). Our research showcases the feasibility of employing deep learning models, particularly YOLOv8, for this purpose, utilizing a meticulously curated dataset comprising over 6,000 annotated traffic images.

Key findings of our work include the successful training and evaluation of multiple models, with YOLOv8 demonstrating the highest performance, achieving an impressive mean average precision of 84.6% across classes. This highlights the potential of deep learning in intelligently identifying traffic violations within pedestrian zones, thereby contributing to enhanced safety measures and traffic enforcement efforts.

The contributions presented in this manuscript encompass the development of a systematic approach to dataset creation for object detection using public CCTV, alongside addressing the challenges of unbalanced datasets and providing benchmarks for various object detection algorithms.

Looking ahead, future research directions include enhancing data acquisition methods, further optimizing AI models for improved accuracy and reduced training time, and exploring real-time inference capabilities to handle larger datasets. Additionally, evaluating the system's performance under diverse environmental conditions and exploring its applicability to other cities or regions will be crucial for broader implementation.

However, it is essential to acknowledge the limitations of our work, including hardware constraints, data quality issues, and the need for larger datasets and hyperparameter exploration. Addressing these limitations through interdisciplinary collaborations and continuous updates and refinements to the AI-based system will be imperative for maintaining its effectiveness over time.

In summary, this study lays the groundwork for advancing research in urban surveillance and traffic enforcement, with the potential to contribute to safer and more efficient urban environments through the integration of AI-driven technologies into comprehensive management strategies.

7 Future work

Future work could focus on enhancing data acquisition methods to access CCTV servers more efficiently through collaborations with city authorities or private entities. Additionally, further optimization of the AI model could improve accuracy and reduce training time by exploring alternative architectures or incorporating additional training data. Investigating real-time inference capabilities and scalability to handle larger datasets would also benefit practical deployment in urban surveillance systems. Moreover, evaluating the system's performance in diverse environmental conditions and exploring its applicability to other cities or regions could provide valuable insights for broader implementation. Additionally, considering the evolving nature of traffic patterns and urban infrastructure, continuous updates and refinements to the IA-based system would be essential to maintain its effectiveness over time. Finally, exploring interdisciplinary collaborations with urban planners, policymakers, and transportation experts could facilitate the integration of AI-based surveillance systems into comprehensive urban management strategies aimed at enhancing safety and mobility.

Funding Open Access funding provided by Colombia Consortium.

Data Availability The author confirms that all data analyzed during this study are published in dataset for Detecting Motorcyclists in Pedestrian Areas [9]. The dataset used in this study can also be found directly in the following repository https://doi.org/10.5281/zenodo.7935298.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical standard This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. de Cartagena A (2023) Plan de Desarrollo Cartagena 2020-2023. sedcartagena.gov.co. http://www.sedcartagena.gov.co/plan-de-desarrollo-cartagena-2020-2023/
2. Cartagena T (2021) Plan de acción Departamento Administrativo de Transito y Transporte - DATT&nbsp; 2021. DATT. https://www.transitocartagena.gov.co/normatividad/decretos-y-resoluciones.html
3. Toh CK, Sanguesa JA, Cano JC, Martinez FJ (2020) Advances in smart roads for future smart cities. Proc R Soc A Math Phys Eng Sci 476(2233):20190439. https://doi.org/10.1098/rspa.2019.0439
4. Hernández Díaz N, Peñaloza YC, Ríos YY, Magre Colorado LA (2022) Software to assist visually impaired people during the craps game using machine learning on python platform. In: Narváez FR, Proaño J, Morillo P, Vallejo D, González Montoya D, Díaz GM (eds) Smart Technologies, Systems and Applications, pp 175–189. Springer, Cham. https://doi.org/10.1007/978-3-030-99170-8_13
5. Suarez OJ, Hernández Díaz N, Pardo Garcia A (2020) A real-time pattern recognition module via Matlab-Arduino interface, Virtual. https://doi.org/10.18687/LACCEI2020.1.1.646
6. Suarez OJ, Macias-Garcia E, Vega CJ, Peñaloza YC, Díaz NH, Garrido VM (2023) Design of a segmentation and classification system for seed detection based on pixel intensity thresholds and convolutional neural networks. In: Orjuela-Cañón AD, Lopez J, Arias-Londoño JD, Figueroa-García JC (eds) Applications of computational intelligence, pp 1–17. Springer, Cham. https://doi.org/10.1007/978-3-031-29783-0_1
7. Hernández-Díaz N, Pañaloza YC, Rios YY, Martinez-Santos JC, Puertas E (2023) Intelligent system to detect violations in pedestrian areas committed by vehicles in the city of cartagena de indias, 6. https://doi.org/10.18687/LACCEI2023.1.1.1447
8. Alcaldía de Medellín (2023) Cámaras de CCTV. https://www.medellin.gov.co/simm/camaras-de-circuito-cerrado Accessed 11-Dec-2020
9. Díaz NH, Peñaloza YC, Rios YY, Martinez-Santos JC, Puertas E (2023) Dataset for detecting motorcyclists in pedestrian areas. Data in Brief 50:109610. https://doi.org/10.1016/j.dib.2023.109610
10. Gao Zh (2016) Fast pedestrian crossing boundary detection method. Softw Eng Appl 05:146–153. https://doi.org/10.12677/sea.2016.52016
11. Deruytter M, Peter J, Versavel J (2009) A detector for detecting traffic participants. https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20090527&DB=&locale=en_EP&CC=EP&NR=2063404A1&KC=A1&ND=1
12. Fascioli A, Fedriga RI, Ghidoni S (2007) Vision-based monitoring of pedestrian crossings. In: 14th International Conference on Image Analysis and Processing (ICIAP 2007), pp 566–574. https://doi.org/10.1109/ICIAP.2007.4362838
13. Hariyono J, Jo K-H (2015) Detection of pedestrian crossing road. In: 2015 IEEE international conference on image processing (ICIP), pp 4585–4588. https://doi.org/10.1109/ICIP.2015.7351675
14. Perdana MI, Anggraeni W, Sidharta HA, Yuniarno EM, Purnomo MH (2021) Early warning pedestrian crossing intention from its head gesture using head pose estimation. In: 2021 International seminar on intelligent technology and its applications (ISITIA), pp 402–407. https://doi.org/10.1109/ISITIA52817.2021.9502231
15. Hudson M, Martin B, Hagan T, Demuth HB (2023) Neural Network ToolboxTM 7 User's Guide. https://dcc.ufrj.br/sadoc/machinelearning/nnet_ug.pdf
16. Zhao C, Chen X (2018) The study of pedestrian re-identification with the illumination change. In: 2018 IEEE 3rd advanced information technology, electronic and automation control conference (IAEAC), pp 133–137. https://doi.org/10.1109/IAEAC.2018.8577489
17. Pop DO, Rogozan A, Chatelain C, Nashashibi F, Bensrhair A (2019) Multi-task deep learning for pedestrian detection, action recognition and time to cross prediction. IEEE Access 7:149318–149327. https://doi.org/10.1109/ACCESS.2019.2944792

18. Porouhan P, Premchaiswadi W (2020) Proposal of a smart pedestrian monitoring system based on characteristics of internet of things (iot). In: 2020 18th International conference on ICT and knowledge engineering (ICT KE), pp 1–4. https://doi.org/10.1109/ICTKE50349.2020.9289891
19. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: Single shot multibox detector. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer Vision - ECCV 2016. Springer, Cham, pp 21–37
20. Wang Z, Feng J, Zhang Y (2022) Pedestrian detection in infrared image based on depth transfer learning. Multimed Tools Appl 81. https://doi.org/10.1007/s11042-022-13058-w
21. Ferguson M, ak R, Lee Y-T, Law K (2017) Automatic localization of casting defects with convolutional neural networks. In: 2017 IEEE international conference on big data (Big Data), pp 1726–1735. https://doi.org/10.1109/BigData.2017.8258115
22. Kadam K, Ahirrao S, Kotecha K (2022) Efficient approach towards detection and identification of copy move and image splicing forgeries using mask r-cnn with mobilenet v1. Comput Intell Neurosci 2022:1–21. https://doi.org/10.1155/2022/6845326
23. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2019) MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://doi.org/10.48550/arXiv.1801.04381
24. Sandler M, Howard A (2018) MobileNetV2: The next generation of on-device computer vision networks. Google Research. https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html
25. hao Q (2024) Qfgaohao/pytorch-SSD: Mobilenetv1, mobilenetv2, VGG based SSD/SSD-lite implementation in pytorch 1.0 / pytorch 0.4. out-of-box support for retraining on open images dataset. ONNX and caffe2 support. experiment ideas like coordconv. GitHub. https://github.com/qfgaohao/pytorch-ssd
26. Dusty-Nv N (2024) Dusty-NV/Jetson-inference: Hello AI World Guide to deploying deep-learning inference networks and deep vision primitives with TENSORRT and Nvidia Jetson. NVIDIA. https://github.com/dusty-nv/jetson-inference
27. Ultralytics Y (2024) Ultralytics/ultralytics: New - yolov8 in PyTorch; ONNX; CoreML; TFLite. Ultralytics. https://github.com/ultralytics/ultralytics
28. Hernández-Díaz N, Puertas E, Martinez-Santos JC, Archbold G, Rios Y, Peñaloza Y (2023) Dataset for Detecting Motorcyclists in Pedestrian Areas. Zenodo. https://doi.org/10.5281/zenodo.7935299
29. Flow R (2024) Yolov5 Pytorch TXT annotation format. RoboFlow. https://roboflow.com/formats/yolov5-pytorch-txt
30. Flow R (2024) Pascal VOC XML annotation format. RoboFlow. https://roboflow.com/formats/pascal-voc-xml
31. Foong NW (2022) Convert Pascal VOC XML to Yolo for Object Detection. Towards Data Science. https://towardsdatascience.com/convert-pascal-voc-xml-to-yolo-for-object-detection-f969811ccba5
32. Wang Y, Jia Y, Chen W, Wang T, Zhang A (2024) Examining safe spaces for pedestrians and e-bicyclists at urban crosswalks: An analysis based on drone-captured video. Accident Anal Prev 194:107365. https://doi.org/10.1016/j.aap.2023.107365
33. Han B, Wang Y, Yang Z, Gao X (2020) Small-scale pedestrian detection based on deep neural network. IEEE Trans Intell Trans Syst 21(7):3046–3055. https://doi.org/10.1109/TITS.2019.2923752
34. Han R, Xu M, Pei S (2024) Crowded pedestrian detection with optimal bounding box relocation. Multimed Tools Appl. https://doi.org/10.1007/s11042-023-18019-5