DEEP LEARNING CLASSIFICATION APPLIED TO TRAFFIC ACCIDENTS PREDICTION

Richard Coll-Josifov richard.coll@upc.edu
Albert Masip-Álvarez albert.masip@upc.edu
David Lavènia-Ferrer david.lavernia@upc.edu

Abstract

In this paper, YOLOv4 neural networks are trained with the goal of detecting and classifying objects from a street as seen from a drone. These have been trained on the VisDrone dataset, which is firstly validated through a custom-made graphic user interface. Then, several tests regarding performance, dataset composition and contrast have been carried out on the trained models. Results are compared to those from other previously existing models in order to evaluate their performance and analyse their shortcomings. The trained models are then used to detect and classify objects in a city scenario in real-time. Finally, an algorithm is proposed to track the objects, infer their future trajectories and predict potential collisions from the expected trajectories.

Keywords: You Only Look Once (YOLO), computer vision, deep learning, accident prediction.

1 Introduction

Driving errors are the main cause of traffic accidents, as stated in [15], [26], [20] and [29]. Recent studies show that lack of visibility, especially at intersections, is one of the factors with the highest incidence of traffic accidents, as indicated in [2].
The use of Deep Learning techniques applied to prevent traffic accidents can be found in recent contributions such as [27]. This publication uses these techniques to detect different traffic signs and elements of the road during traffic. Convolutional Neural Networks (CNN) are used in [28] to detect and classify animals that invade the circulation pathway. It has been necessary to carry out new trainings of the YOLO network on a customised dataset before addressing the main contribution of the work shown in this paper: the detection of potential traffic accidents from identification and classification techniques through Deep Learning tools.
In Section 2, the problem statement and the state of the art are presented, as well as a performance test of previous pretrained YOLO models on the customised dataset. In Section 3, the dataset is presented and analysed. In Section 4 the neural networks are trained and tested. In Section 5, the different pre-existing models are compared to the trained models. Finally, the accident prediction algorithm is explained in Section 6.
All the code related to this paper can be found on: https://github.com/RichardCollJosifov/YOLO

2 Problem statement

This paper analyses the training of a convolutional neural network for the systematic detection and classification of objects on a street into the following 6 categories: car, truck, pedestrian, bike, motorbike and bus. Video stream is provided by a drone flying over the scenario. Output results from the classification procedure are used by an algorithm that, given the current and previous frame location of the objects, will make predictions on their paths and will consider potential collisions between the elements.
Real time execution is a strong requirement of the system to be built.

2.1 State of the art

The most common approaches for object detection and classification that can be found in the literature are those based on deep learning techniques using convolutional neural networks. There are multiple approaches that achieve real-time processing speed. Out of the two-stage detection methods, the main ones for this area are Faster-RCNN[23] as well as its extension Mask R-CNN[9], whereas for the one-stage detection methods, the most used one is "You Only Look Once"[21] and, particularly, version YOLOv4 [6]. Besides YOLO architectures, another common one-stage method is Single Shot MultiBox Detector (SDD) [17].
A recent study by Kim, J. et al., [14] compares R-CNN, YOLOv4 and SDD for real-time vehicle recognition, which is very similar to the task of pedestrian and car detections. In their study they found that in terms of processing speed in frames per second (FPS), SDD was the fastest,

 https://doi.org/10.17979/spudc.9788497498418.0964

with 105.14FPS in comparison to 82.1FPS for YOLOv4 and 36.32FPS for Faster R-CNN. But when it comes to the mean average precision, YOLOv4 performed the best with 98.19%, in front of 93.40% for Faster R-CNN and 90.56% for SDD. Thus, as a good combination of both speed and precision, YOLOv4 structure is determined to be the best model structure for this usage.
Another study, by Benjdira, B. et al. [5], compared YOLOv3 against Faster R-CNN for car detections from drone aerial-view images, and found that YOLOv3 outperformed Faster R-CNN in all relevant metrics, including precision and processing speed. Given that YOLOv4 improves on YOLOv3, it can be assumed that these results also hold for YOLOv4. Other studies have found similar results [30] [11] [1].
As a consequence, in this paper YOLOv4 models are trained. Furthermore, the Darknet framework [22] commands have been used for the training, usage and test of YOLO based neural networks in order to reduce the coding requirements.
YOLOv4 has been used in front of other more advanced YOLO architectures, such as YOLOv5 [12] or PP-YOLO[18] due to the fact that there are more pre-trained models ours could be compared in front of, as well as there being several comparisons of YOLOv4 to other methods for tasks very similar to ours (and no such analysis yet for more advanced architectures). YOLOv6[19] and YOLOv7[31] were not used as well as they were published when this project was already at an advanced stage.
There has been previous research in the area of deep neural networks in order to predict possible accidents. Chavan, D. et al.[7] built a model, COLLIDE-PRED, involving a YOLOv4 neural network for object detection and classification, with the results of it being fed to a siamese region proposal network (SiamRPN), which is a network architecture for object tracking. After this, the trajectories are fed to a Transformer, which is a type of deep learning network, which then makes predictions for the future paths of each object. Their system then compares the predicted paths of the different objects (as seen from surveillance cameras), and a collision is predicted if at the same instant the model predicts two objects will be in the same position. Another study, by Yao, Y. et al.[32], looked at an unsupervised system to detect traffic anomalies based on cameras mounted on cars, and thus being non-stationary. Another study by Huang, X. et al. [10] used two neural networks, one for spatial data and one for temporal data to predict paths and possible collisions. Other models, such as the one devised by Kataoka, H. et al. [13], train models directly with a dataset of (near-)accident scenarios.

2.2 Background

This work is built on previous research. In [25], several YOLOv4 models were trained on the Stanford Drone Dataset (SDDs) [24], which is a drone based dataset with all images in zenithal view. But the models trained on this dataset encountered problems relating to performance as well as generalisation to other scenarios.
The research in the present paper is part of a larger research project being carried out at the Advanced Control Systems research group at the Universitat Politècnica de Catalunya (UPC BarcelonaTech), focusing on the coordination of autonomous vehicles. The approach here taken would be useful in coordinating the autonomous vehicles for collision avoidance.

2.3 Previous models testing

A testing of pre-existing publicly available models on the VisDrone test set was carried out to check if there were any that fulfilled the requirements in the average precision (measured in mAP, **m**ean **A**verage **P**recision). Precision P_i for a specific class i is calculated by the quotient:

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{1}$$

TP_i are the true positives on class i; they are valued 1 when the intersection over union is greater than 50% and 0 otherwise. FP_i are the false positives on class i; they are valued 1 when the intersection over union is lower than 50% or the detected class is not coherent with that one of the object and 0 otherwise. The mean average precision is then determined by the mean of the average precision for all classes to be detected and classified:

$$mAP = \frac{1}{6} \cdot \left(P_c + P_p + P_{bi} + P_{bu} + P_m + P_t\right) \tag{2}$$

being the subscripts: c for cars, p for pedestrians, bi for bicycles, bu for buses, m for motorbikes and t for trucks.
The tested models were those obtained in the previous study by Roset [25] as well as those published by Bochkovskiy, author of YOLOv4 [6], and whose models are COCO [16] based. In Table 1 the results of the best performing networks from each source is shown. The recall measures the proportion of ground truth objects that the model is able to detect, categorize and size correctly. It is defined by the quotient Recall $= \frac{TP}{TP+FN}$, being FN the false negatives in those situations where there is an object in the ground truth but the network has no prediction on it. The F1 metric combines into a single value the Precision and Recall:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Table 1: Previous models comparisons.

	Model	mAP	Recall	F1
COCO	yolov4-csp-swish	43.88%	45%	0.444
	yolov4-csp-x-swish	45.61%	48%	0.467
	yolov4-p5	44.93%	54%	0.490
	yolov4-p6	52.48%	61%	0.564
SDDs	Roset Model 5	4.19%	7%	0.052
	Roset Model 6	4.63%	6%	0.052

It can be observed that the models trained on the Stanford Drone Dataset perform poorly. The COCO based models perform good, albeit their overall precision levels are brought down by their low precision levels in the bicycles and motorbikes categories. Nonetheless, their overall low mAP values make it necessary to train custom-made networks for this purpose in order to increase performance.

3 Dataset generation

The selected dataset to start with was the VisDrone dataset [33]. This dataset consists of several sets of drones videos. The ones useful are the Multi-Object Tracking set, which consists of annotated videos containing multiple objects.

The dataset was not used directly, but first it was pre-processed through a custom Matlab graphical user interface (GUI) created to validate datasets, as the authors of the dataset recognize in their webpage that it contains several annotation errors. In addition, their classes differed from the ones being used here, and so the class numbering system had to be adapted from 10 classes to the 6 classes used in this paper. In addition, some objects had been misclassified, in which case with the GUI these errors were corrected, as well as there being objects badly located, meaning they did not cover the entirety of the object they were to identify. After having used the GUI to correct these errors in around 11500 images, an analysis on the changes was made, with between 5.99% and 9.35% of changes in the objects annotations being made to the used dataset, which seem to be high enough values as to justify the need of the validation of the data.

The validated dataset was then split into the train/dev/test sets, following a rule of about 60%-20%-20%. In total, around 7500 images were used for training, and 2000 each for dev and test. This division was done with whole video settings going to each (instead of just part of a video in every set), by trying to have a diverse population of classes and lighting and orientation setups in all the sets.

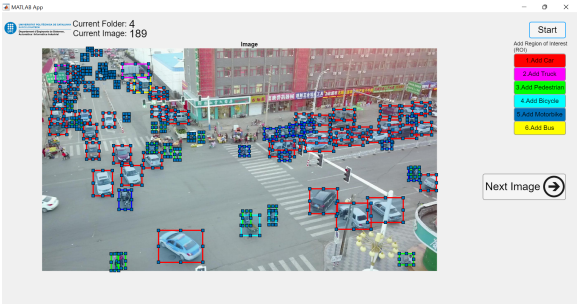


Figure 1: General view of the GUI for the dataset generation procedure

4 Neural network training

4.1 Darknet and YOLO networks

The Darknet framework allows, once it is set up, to train a network easily just through a single command. The only necessary things besides the annotations for the dataset are to have a file with the directories of the images to be used for training and a file for the configuration of the YOLO network. For all the different models the same training set will be used, so the only difference will be in the configuration of the networks. Two YOLO networks will be trained, with an additional two more so called tiny-YOLO, which are a light version of YOLO with less neurons and layers, that has much higher processing speed but lower precision. The parameter to be changed will be the input size of the image (and as a consequence, the number of neurons and weights and bias to be trained). For the YOLO, one of 480×480 input image size and one of 640×640 are trained, to be able to compare the increase in performance due to this larger size of the input of the image, whereas for the tiny-YOLO it is one with input of 416×416 and one with input 640×640. All the training has been undertaken through Google cloud services. Data augmentation has been used, as by default Darknet includes it already (it randomly rotates images, changes saturation, brightness and colour and uses Mosaic data augmentation).

4.2 Validation results

The models have then been tested firstly with the dev set, with results on Table 2. The precision of the yolov4-640 is high with high recall as well.

In front of the test set, the models have the following performance, as seen in Table 3, as well as

https://doi.org/10.17979/spudc.9788497498418.0964

Table 2: Trained models performance on dev set.

Model	mAP	Recall	F1
tiny-yolov4-416	41.00%	31%	0.353
tiny-yolov4-640	53.50%	47%	0.500
yolov4-480	69.73%	66%	0.678
yolov4-640	75.97%	70%	0.728

the performance on the three main categories seen in Table 4. The mAP drops for all the models but the recall increases, with an F1 metric for most models just slightly below the one on the dev set.

Table 3: Trained models performance on test set.

Model	mAP	Recall	F1
tiny-yolov4-416	34.08%	34%	0.340
tiny-yolov4-640	51.62%	64%	0.571
yolov4-480	60.35%	75%	0.668
yolov4-640	62.28%	79%	0.696

Table 4: Trained models' testing results on test set categories.

Model	AP		
	car	pedestrian	bike
tiny-yolov4 416	56.44%	4.30%	28.13%
tiny-yolov4 640	83.17%	37.61%	66.17%
yolov4 480	87.87%	63.10%	80.59%
yolov4 640	89.24%	71.87%	82.47%

5 Detection Results

Given the performance differences between the models trained on the Stanford Drone Dataset and the VisDrone dataset, an analysis of potential factors explaining it is here presented.

5.1 Image size

Given the low input size in relation to the VisDrone's data resolution, a test was made to determine the effects of the original image resolution and size in the prediction. The same high-resolution image was fed to the system in its original resolution (1900×1000), as well as that image being reduced to 480×480, 320×320 and also by reducing the image by adding a dark frame around it (so that when the image was rescaled to the input size of the network, the image would be smaller).

The results indicate that in the original format, the high resolution one, the system performs the best. As the image's resolution is decreased, the number of predictions the system makes also decrease. An interesting point is the case of the image with the dark frame, where the system does make a lot of predictions, even more than in the original image, but it misclassifies them, by predicting bicycles instead of the ground truth of cars.

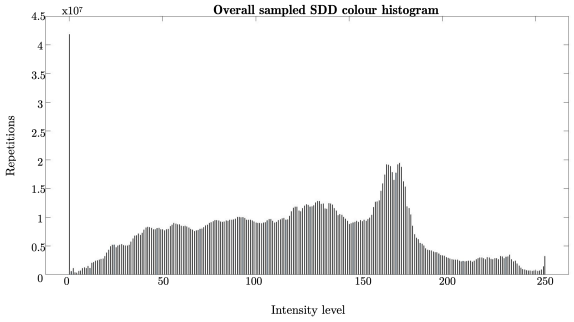
(a) original (b) 480x480

(c) 320x320 (d) dark frame

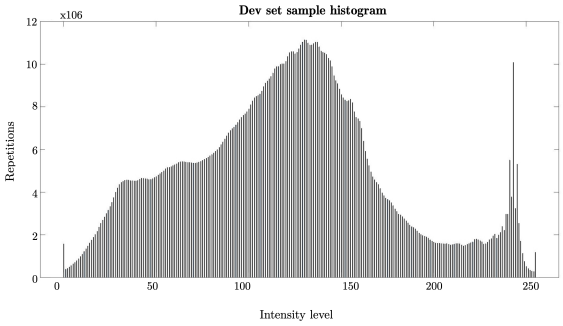
Figure 2: Results of Roset Model 6 YOLOv4 in modified images.

5.2 Contrast

Another aspect that might be relevant to a model's performance is the contrast in the images, both in light intensity and in colour range (as in a flat versus a concentrated luminance distribution). The overall contrast of samples of each dataset differs slightly, as seen in Figure 3.



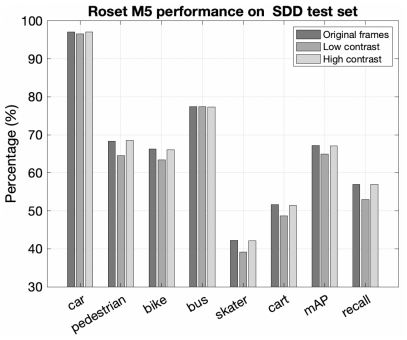
(a) SDDs



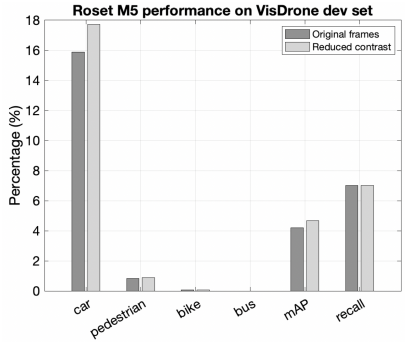
(b) VisDrone

Figure 3: Colour histograms from the datasets.

In particular, in the SDDs dataset, some videos have very low contrast, either by values lower than 200 or higher than 50 in the colours range. To test if this is having an effect on the performance, the VisDrone and the SDDs datasets were modified to have a colour range between 50 and 200. In the case of the yolov4-480 trained model, in the VisDrone dataset, the performance was reduced under this conditions, but for the Roset's models the performance increased slightly. But when testing for the SDDs dataset, the Roset's models in this conditions negatively impact the performance as well, as can be seen in Figure 4.



(a) SDDs



(b) VisDrone

Figure 4: Roset model 5 performance with contrast change.

5.3 Dataset view orientation

Another regard that has an impact on the performance is the angle of the camera the images are taken from. Purely bird-view to angled view images, the good performance on one does not imply the good performance on the other. For example, the trained network yolov4-480 has an overall precision of 60.35% in the overall test set, but when testing only in front of the zenithal view images of the set, the mAP drops to 14.42%, due in part to the low availability of zenithal view images in the training set. A question arises whether the zenithal and angled view images taken together makes the system precision increase or decrease, as compared to training just with one type or the other. To test this, different models have been trained with different amounts of zenithal view images as percentage of the total images, as seen in Table 5. The original model, model 1, contained 861 zenithal view images, which represented 11.66% of all the training data. For the other models, an additional video from the test set was added to training (as it is then tested in the dev set just for the purpose of this query) bringing the total of zenithal view images to 1146. But for each of the models 2, 3 and 4, the amount of angled image was reduced (by eliminating similar videos and reducing the frames from some videos) to bring the percentage of zenithal to the levels of 33%, 57% and 70%.

Table 5: Number of images of each type.

Model	Angled	Zenithal	% zenithal
1	7379	861	11.6682%
2	3471	1146	33.0164%
3	1985	1146	57.7329%
4	1625	1146	70.5231%

Then the models were trained with the Darknet framework and tested in front of the dev set, with results seen in Table 6. Applying a correlation analysis on the mAP in front of the amount of angled view images reveals a correlation coefficient of 0.9706 and a p-value of 0.0294, meaning that the null hypothesis of no correlation between the variables can be dismissed. Thus, it seems to be the case that the more angled view images there are, the better the performance on zenithal view is, regardless of a small addition of zenithal images. So a network trained in a bigger combination of both angled and zenithal images would seem to have greater performance than a system trained on just one or the other. These results, though, are not conclusive given the small amount of data (just 4 data points).

Table 6: Trained models of Table 5 performance on VisDrone zenithal dev set.

Model	**mAP**	**Recall**	**F1**
1	14.42%	43%	0.215
2	5.10%	6%	0.055
3	2.47%	4%	0.030
4	4.26%	11%	0.061

6 Accident prediction algorithm

This paper looks into applying just one neural network, YOLOv4, for object detection and classification followed by logic based and equation solvers

https://doi.org/10.17979/spudc.9788497498418.0964

algorithms for the task of accident prediction from non-moving drone view images.

6.1 Object Tracking

For object tracking the algorithm looks at one previous frame and considers an object to be the same one if the centre of the current one is within the bounds of the region of an object of the same class in a previous frame. If the method finds the current object in the preceding frame, it then tries to find with the same algorithm the object in the second preceding frame (it only does this for 2 previous frames).

Algorithm 1 Object Tracking

function OBJECTTRACK(cObj, previousObjectsList)
  for *pObj in previousObjectsList* **do**
    if $cObj.class == pObj.class$ **then**
      if $pObj.x < cObj.x < (pObj.x + pObj.width)$ & $pObj.y < cObj.y < (pObj.y + pObj.height)$) **then return** *pObj*
      end if
    end if
  end for
return *null*
end function

6.2 Path Prediction

Making use of the function for object tracking, the path prediction algorithm works by looking at the previous frames where the object is located, and does a linear regression with the centres of the object at the different frames to make the prediction of the path.

Algorithm 2 Path prediction

for $i = 1 : length(currentObjectsList)$ **do**
  $cObj \leftarrow currentObjectsList(i)$
  $previous \leftarrow ObjectTrack(cObj, previousObjectsList)$
  if $previous \neq null$ **then**
    $paths(i).path \leftarrow$ linearRegression($(cObj.x, cObj.y), (previous.x, previous.y)$)
    $paths(i).position \leftarrow (cObj.x, cObj.y)$
    $paths(i).speed \leftarrow (cObj.x - previous.x, cObj.y - previous.y)$
    $paths(i).class \leftarrow cObj.class$
    $paths(i).exists = true$
  end if
end for

6.3 Accident Prediction

With the predicted paths, a linear equation solver is used to, one to one, check for the intersection point between all the paths, and if it is within the dimensions of the image, and the direction of both objects is towards that point (as computed through their x, y speeds), an alert is generated as the system predicts a collision.

Algorithm 3 Collision prediction

for $i = 1 : length(paths) - 1$ **do**
  for $j = i + 1 : length(paths)$ **do**
    if $paths(i).exists$ & $paths(j).exists$ **then**
      $(x, y) = solve(paths(i), paths(j))$
      if ((x, y) are Numbers & $0 \leq x \leq imWidth$ & $0 \leq y \leq imHeight$) **then**
        if $sign(path(i).speed) \neq sign(path(i).position - (x, y))$ & $sign(path(j).speed) \neq sign(path(j).position - (x, y))$ **then**
          $collisWarn(path(i), path(j))$
        end if
      end if
    end if
  end for
end for

6.4 Validation

The system has been validated in front of videos recorded from a static drone, in general it works correctly, in regard to detecting possible collisions, but it generates a lot of false predictions of possible collisions between objects that do not come close to collision. In Figure 5 an example with the outputs of the yolov4-480 network (coloured rectangles) plus the predicted paths (lines) and a predicted collision (red lines) can be observed.

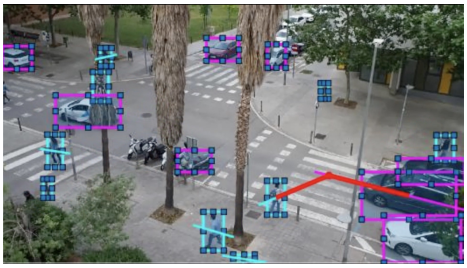


Figure 5: Example of collision prediction.

This system, though, is too sensitive to noise from the predictions, as small deviations can mean that

the predicted path is very different from the real path, as well as the fact that predicted paths are linear and cannot represent the most complex paths usually done by the vehicles and pedestrians.

Regarding the processing time, the algorithm is still too slow, with Figure 6 showing the histogram of the computing times in a test set. The average processing time, executed as Matlab code and run on an average laptop, is of 0.5954 seconds, but with certain frames taking considerably longer with up to 2.5 seconds in some frames.

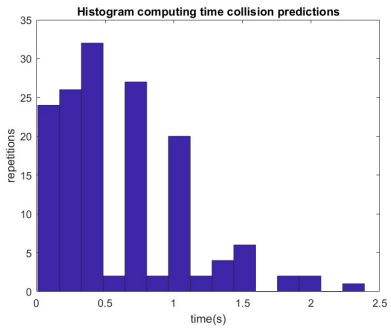


Figure 6: Histogram of computing time in a laptop.

7 Conclusions and future work

In this paper, YOLOv4 neural networks have been trained on the VisDrone dataset, which have then been used for an accident prediction algorithm. The collision prediction algorithm works correctly in anticipating possible accidents but with some drawbacks, mainly the system is too sensible to noise and overpredicts potential accidents.

The effects of angled and birds-eye view data mixing have been tested, with the conclusion that additional angled view images increase the performance of the system in birds-eye view.

Future work should be carried out in the field of integration and development of more advanced path prediction algorithms, such that it combines real-time processing speed with adaptive and complex path proposals. Most current techniques, though, either focus on people path prediction, such as [3], or on vehicle path prediction, such as [8] (although current methods face problems generalising to multiple settings [4]). So a system using multiple algorithms depending on the object could be devised.

Acknowledgement

This work has been co-financed by the Spanish State Research Agency (AEI) and the European Regional Development Fund (ERFD) through the project SaCoAV (ref. MINECO PID2020-114244RB-I00), by the European Regional Development Fund of the European Union in the framework of the ERDF Operational Program of Catalonia 2014-2020 (ref. 001-P-001643 Looming Factory) and by the DGR of Generalitat de Catalunya (SAC group ref. 2017/SGR/482).

References

[1] Abdulghani, A. M. A., Menekse Dalveren, G. G. (2022) "Moving Object Detection in Video under Different Weather Conditions Using YOLO and Faster R-CNN Algorithms", *European Journal of Science and Technology*, No. 33, pp. 40-54.

[2] Adanu, E. K. *et al.*, (2021) "An Analysis of the Effects of Crash Factors and Precrash Actions on Side Impact Crashes at Unsignalized Intersections", *Journal of Advanced Transportation.*

[3] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. and Savarese, S., (2016) "Social LSTM: Human trajectory prediction in crowded spaces", *Computer Vision and Pattern Recognition.*

[4] Bahari, M., Saadatnejad, S., Rahimi, A., Shaverdikondori, M., Shahidzadeh, A., Moosavi-Dezfooli, S. and Alahi, A., (2022) "Vehicle trajectory prediction works, but not everywhere" *IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[5] Benjdira, B. et al., (2019) "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3", *International Conference on Unmanned Vehicle Systems*, pp. 1-6.

[6] Bochkovskiy, A., Wang, C. and Mark Liao, H., (2020) "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv.

[7] Chavan, D., Saad, D. and Chakraborty, D., (2021) "COLLIDE-PRED: Prediction of On-Road Collision From Surveillance Videos", arXiv.

[8] Deo, N. and Trivedi, M., (2018) "Convolutional Social Pooling for Vehicle Trajectory Prediction", *IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

https://doi.org/10.17979/spudc.9788497498418.0964

[9] He, K., Gkioxari, G., Dollar, P. and Girshick, R., (2017) "Mask R-CNN", *International Conference on Computer Vision (ICCV)*, pp. 2980-2988.

[10] Huang, X. et al., (2020) "Intelligent Intersection: Two-Stream Convolutional Networks for Real-time Near Accident Detection in Traffic Video", *ACM Transactions on Spatial Algorithms and Systems, June 2020*, pp 1-28.

[11] Jiao, L. et al., (2019) "A Survey of Deep Learning-Based Object Detection", *IEEE Access*, vol. 7, pp. 128837-128868.

[12] Jocher, G., (2020) "YOLOv5", GitHub.

[13] Kataoka, H., Suzuki, T., Aoki, Y. and Satoh, Y., (2018) "Anticipating Traffic Accidents with Adaptive Loss and Large-scale Incident DB", *IEEE Conference on Computer Vision and Pattern Recognition.*

[14] Kim, J., Sung, J. and Park, S., (2022) "Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition", *IEEE International Conference on Consumer Electronics - Asia*, pp. 1-4.

[15] Lenard, J., and Hill, J., (2004) "Interaction of Road Environment, Vehicle and Human Factors in the Causation of Pedestrian Accidents", *International Expert Symposium on Accident Research (ESAR).*

[16] Lin, TY. et al. (2014) "Microsoft COCO: Common Objects in Context", *Computer Vision - ECCV 2016*, pp 740-755.

[17] Liu, W. et al., (2016) "SSD: Single Shot MultiBox Detector", *Computer Vision - ECCV 2016*, pp 21-37.

[18] Long, X. et al., (2020) "PP-YOLO: An Effective and Efficient Implementation of Object Detector", arXiv.

[19] Meituan, (2022) "YOLOv6", GitHub.

[20] Papantoniou, P. *et al*, (2019) "Which factors lead to driving errors? A structural equation model analysis through a driving simulator experiment", *IATSS Research 43.*

[21] Redmon, J. et al., (2016) "You Only Look Once: Unified, Real-Time Object Detection",*IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788.

[22] Redmon, J. and Farhadi,A., (2018) "YOLOv3: An Incremental Improvement", arXiv.

[23] Ren, S. et al., (2015) "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *Advances in Neural Information Processing Systems*, Vol. 28.

[24] Robicquet, A. et al., (2016) "Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes", *European Conference on Computer Vision* .

[25] Roset, J., (2021) "Estudi sobre la categoritzacio d'elements terrestres a partir d'imatges aeries de pla zenital", Universitat Politecnica de Catalunya, url: http://hdl.handle.net/2117/360728.

[26] Rumar, K., (2011) "The basic driver error: late detection", *Ergonomics, 33:10-11, 1281-1290.*

[27] Sanjeewani, P. and Verma, B., (2021) "Single class detection-based deep learning approach for identification of road safety attributes", *Neural Computing and Applications.*

[28] Santhanam, S. *et al.*, (2021) "Animal Detection for Road safety using Deep Learning", *International Conference on Computational Intelligence and Computing Applications.*

[29] Santoso, G. P. and Maulina, D., (2019) "Human Error in Traffic Accidents: Differences between Car Driver and Motorcyclist Experiences", *Psychological Research on Urban Society.*

[30] Shakir Sumit, S. et al., (2020) "In object detection deep learning methods, YOLO shows supremum to Mask R-CNN", *Journal of Physics: Conference Series.*

[31] Wang, C., Bochkovskiy, A. and Mark Liao, H., (2022) "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", arXiv.

[32] Yao, Y. et al., (2019) "Unsupervised Traffic Accident Detection in First-Person Videos", *International Conference on Intelligent Robots and Systems (IROS).*

[33] Zhu P., Wen L., Du D., et al., (2021) "Detection and Tracking Meet Drones Challenge", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp 1-1.