Neural Computing and Applications (2024) 36:5847–5860
https://doi.org/10.1007/s00521-023-09401-3

ORIGINAL ARTICLE

Efficient CNN-based low-resolution facial detection from UAVs

Julio Diez-Tomillo[1] · Ignacio Martinez-Alpiste[1] · Gelayol Golcarenarenji[2] · Qi Wang[1] · Jose M. Alcaraz-Calero[1]

Received: 22 June 2023 / Accepted: 13 December 2023 / Published online: 13 January 2024

Abstract

Face detection in UAV imagery requires high accuracy and low execution time for real-time mission-critical operations in public safety, emergency management, disaster relief and other applications. This study presents UWS-YOLO, a new convolutional neural network (CNN)-based machine learning algorithm designed to address these demanding requirements. UWS-YOLO's key strengths lie in its exceptional speed, remarkable accuracy and ability to handle complex UAV operations. This algorithm presents a balanced and portable solution for real-time face detection in UAV applications. Evaluation and comparison with the state-of-the-art algorithms using standard and UAV-specific datasets demonstrate UWS-YOLO's superiority. It achieves 59.29% of accuracy compared with 27.43% in a state-of-the-art solution RetinaFace and 46.59% with YOLOv7. Additionally, UWS-YOLO operates at 11 milliseconds, which is 345% faster than RetinaFace and 373% than YOLOv7.

Keywords Face detection · UAV · Deep learning · YOLO · RetinaFace

1 Introduction

Detecting and tracking down people on the ground from a drone or unmanned aerial vehicle (UAV) is critical for a myriad of applications such as remote monitoring and surveillance, search and rescue to find missing people [1, 2], people counting in dense crowds, emergency and vigilance (e.g., the Drone Guard Angel in the EU H2020 project ARCADIAN-IoT) and public safety (e.g., surveillance application in the EU H2020 project 5G-INDUCE) [3]. Face detection is an active research area in the field of computer vision to identify specific individuals. It is the first step to develop a robust facial recognition system by detecting and locating the human face from the obtained image. Automatic face detection system plays a vital role in face identification, facial expression recognition, head-pose estimation and human–computer interaction among others [4]. Although there has been a lot of research conducted on facial detection, there are still several issues to be addressed owing to various challenging operational conditions being involved in facial detection from drones including a high degree of variability, distance from the camera, face orientation, illumination, face occlusion, complex backgrounds, low resolution to name a few. These challenges have a detrimental effect on the detection rate and accuracy of the detection [4].

In addition, most of the research work is just focused on accomplishing the task of detecting faces in the scene with high accuracy. Nevertheless, the complete use case when follow-up tasks are executed is not generally considered in the research community. Figure 1 presents a typical example of a flowchart in order to perform some of the mentioned follow-up tasks, such as face identification, expression recognition and head pose estimation. These tasks are computationally expensive, and thus, their inference time is high. Therefore, the face detection task should be fast and light, in order to allow other tasks to provide results close to real time. This research is focused on improving the inference time in the face detection stage.

Although having a low execution time is mandatory for the success of different use cases, developing and executing a low computationally expensive algorithm is key to reduce the required amount of memory needed from the GPU (Graphics Processing Unit). Current research work should focus on lowering the rate of operations per second

✉ Julio Diez-Tomillo
julio.diez-tomillo@uws.ac.uk

1 University of the West of Scotland, Paisley, UK

2 University of Portsmouth, Portsmouth, UK

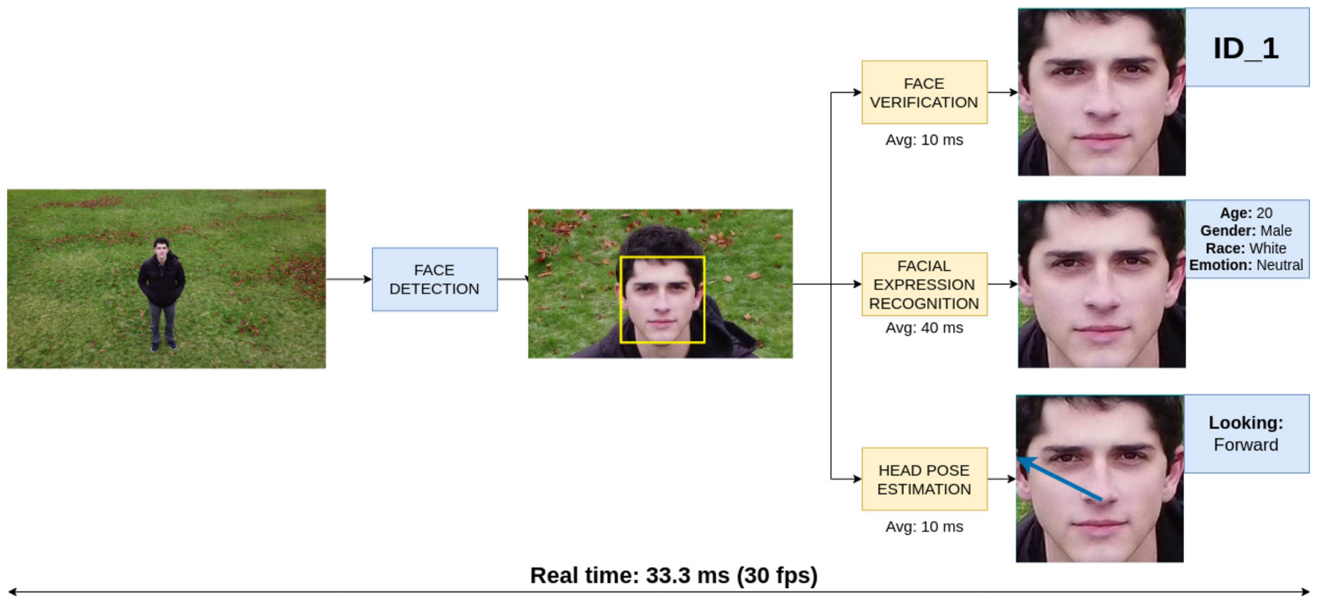


Fig. 1 Flowchart with examples of the use of face detection with follow-up tasks with their average inference time

to allow other algorithms to perform in parallel in the same GPU.

The aim of this study is to create a convolutional neural network (CNN) specifically designed for deployment with unmanned aerial systems (UAS). The CNN targets for a trade-off among accuracy, speed and portability. The solution should achieve high accuracy able to detect small faces within large images even if the faces consist of only a few pixels. In addition, in order to overcome the highlighted challenges, the CNN is optimized in terms of speed to achieve an execution time beyond real time (faster than 30 frames per second). Finally, this algorithm is power-efficient with low computational demands. As a result, our contributions are summarized as follows:

- Up-to-date literature review on face detection techniques and their limitations.
- Design and implementation of a novel CNN-based algorithm (UWS-YOLO) to perform facial detection from a UAV.
- Creation, processing and labeling of a new dataset with people’s faces recorded from a UAV at different distances.
- Performing qualitative and quantitative evaluation, and comparison with state-of-the-art algorithms (RetinaFace and YOLOv7) in terms of performance.

The organization of the paper is as follows: Sect. 2 reviews the related work. Section 3 describes the design of the proposed solution to perform facial detection, followed by the implementation setup in Sect 4. Section 5 presents the results of the proposed solution. Finally, Sect. 6 concludes the paper.

2 Related work

The aim of this section is to explain the techniques used in this paper. In addition, it reviews state-of-the-art work related to facial detection.

2.1 Machine learning techniques

The methods commonly used to detect faces from images or videos are divided into two main categories: feature-based and image-based approaches. The feature-based approaches such as Viola-Jones [5] and Gabor features-based methods [6] focus on extracting main facial features which result in sub-optical facial detection [4]. These methods are fast but fail when the detections are from different angles and lighting conditions [7]. Image-based approaches including deep convolutional neural networks have inspired face detection in recent years. There are two categories for CNN-based face detectors, region-based (two-stage) and single-stage methods. Fast RCNN [8], Faster RCNN [9] and R-FCN [10] are common region-based methods. They have achieved high accuracy at the cost of speed. The single-stage methods including YOLO series [11–13] and SSD (single-shot detector) [14] have achieved high inference time but lower accuracy compared to two-stage approaches. RetinaFace [15] using Resnet-152 as backbone achieves great accuracy but has a high inference time while processing HD (1920×1080) or 4k (4096×2160) images.

2.2 Advanced network configurations

The aforementioned single-stage CNNs may be configured in order to achieve higher accuracy for low-pixel size object detection. Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PAN) are two advanced techniques that keep the small features extracted in the preliminary CNN's layers to increase the accuracy at almost no cost in execution speed.

2.2.1 Spatial pyramid pooling (SPP)

The SPP [16] uses large max poolings to improve the receptive field and consequently increases the accuracy of the architecture. It is robust to object deformation and carries out information cumulation at a deeper stage of the neural network. To increase the receptive field and get better accuracy, the strides in the SPP module were changed to 7 x 7, 10 x 10 and 13 x 13 in the backbone of the proposed architecture.

2.2.2 Path aggregation network (PAN)

PAN is a method for the improvement of the detection of small objects. To achieve this, the features of the top layers with more information are combined with the deeper layers with more shallow features [17]. In this study, the PAN architecture was modified using an additional upsampling to keep more meaningful information which is essential for the detection of small objects.

2.3 Facial detection

This subsection compares state-of-the-art facial detection results with the proposed approach in this article. The results have been obtained from the papers of each algorithm. Table 1 shows the comparative analysis.

In one study, [18] proposed a face detection technique implemented on a Raspberry Pi. Haar cascade classifier algorithm was implemented using OpenCV [19]. The Droneface dataset [20] was used as the facial image dataset. Although high accuracy was obtained (98%, 93%, 86% and 80%) for distances of 1.5, 3, 4 and 5 m, respectively, the detections were not tested from an altitude of more than 5 m. In addition, the speed of the model has not been reported. In another study, [21] proposed an enhanced YOLOv3 algorithm for face detection being comparable with YOLOv3 and YOLOv4. The mAP of 51.9% was achieved using WIDER FACE dataset [22] which is less than our model with mAP of 72.26%. The speed of the model has not also been reported.

In [7], a fast customized CNN was implemented suitable for UAV use cases. In [23] and [24], although high performance has been achieved for face detection and identification, they are not fast enough for our use case and they have been tested up to a distance of 10 m. In [25], a Mobilenet-SSD was implemented in TensorFlow for facial detection. Although very high accuracy was obtained (91.92%), the algorithm was not tested on the drone. In another study [26], YOLOv3 was implemented on Raspberry Pi for facial detection. This study is slow for our use case (6.7 frames per second) and has been tested at a distance of 3.8 m. In [27], Local Binary Pattern Histogram (LBPH) was used as a face recognizer for anti-theft and surveillance applications. Although it has obtained a high accuracy of 89.1%, the speed and the distance have not been reported.

In [28] a high accuracy is achieved in the WIDER FACE dataset but without specifying what input size was used to obtain this accuracy. Moreover, it achieves almost real-time inference speed being close to 30 frames per second (FPS) which is slower than our model (91 FPS). In [29] and [30] a high accuracy is achieved in the WIDER FACE dataset, but the inference time has not been reported. In [31] a modification of RetinaNet to improve accuracy and speed is presented. However, our model achieves better results in terms of accuracy and inference time in the WIDER FACE dataset.

In summary, current literature does not consider deeply the challenges involved in facial detection from UAVs including altitude, illumination and face orientation among others and the models used in these studies are computationally expensive. However, our proposed solution reduces the computational power to make it more suitable for portable, resource-constrained devices.

3 Design of the proposed algorithm

The proposed algorithm is designed to strike a balance between accuracy, speed and portability. The UWS-YOLO integrates three key components to create an effective solution for face detection. First, a robust backbone architecture to achieve high accuracy, followed up by an SPP module with modified strides of 7, 10 and 13 in the max poolings. Finally, an enhanced PAN increases the accuracy for small object detection. Figure 2 presents the architecture of the proposed solution combining all the techniques implemented.

3.1 The backbone

The baseline of the proposed solution is the Tiny-YOLOv4 [32] backbone whose main strength is the low execution

Table 1 Comparison of facial detection solutions

References	Algorithm	Exec Env.	Platform	Accuracy	Speed (FPS)	Model size	Distance
[18]	Haar cascade	RPi	OpenCV	98.00%	NG	NG	Up to 5 m
[21]	Enhanced YOLOv3	PC	Darknet	51.90%	NG	NG	WF distance
[7]	Customzed CNN	PC	Caffe	77.40%*	31.0	NG	WF distance
[23]	LSTM	PC	NG	99.20%	00.7	NG	3–6 m
[24]	SVM	PC	OpenCV	97.50%	13.0	NG	2–10 m
[26]	YOLOv3	RPi	NG	NG	06.7	NG	3.8 m
[25]	Mobilenet-SSD	PC	TensorFlow	91.92%	39.0	NG	NG
[27]	LBPH	RPi	NG	89.10%	NG	NG	NG
[28]	IRNet	PC	PyTorch	76.60%	31.3	1.68 MB	WF distance
[29]	Enhanced YOLOX	PC	PyTorch	87.38%	NG	NG	WF distance
[31]	Enhanced RetinaNet	PC	PyTorch	41.00%	11.8	NG	WF distance
[30]	EfficientFace	PC	PyTorch	90.10%	NG	NG	WF distance
TP	UWS-YOLO	PC	Darknet	59.29%	91.0	49 MB	2–30 m

TP = This Paper; NG = Not Given; RPi = Raspberry Pi; WF = WIDER FACE; *=Recall

speed (4 ms). This backbone serves as the foundation of the UWS-YOLO algorithm; nevertheless, its main flaw is the low accuracy that it presents. Our solution includes Cross-Stage Partial Networks (CSPBlock) modules which replace the ResBlock module in the residual network (marked as a). CSPBlocks [33] were chosen in order to improve the correlation difference of gradient information and enhance the learning ability of the convolution network. The feature maps are divided into two branches to apply different transformations and then concatenated back together. UWS-YOLO includes three CSPBlock modules with 64, 128 and 256 filters. The parameters of the CSPBlocks are adopted according to [34]. Experiments with various filter numbers have been carried out in order to choose the best.

To implement this architecture, firstly similar to YOLOv4, the CSPBlock modules were used in the backbone of the proposed architecture. Cross-stage residual edge is used in the CSPBlock module to combine the two divided feature maps (groupid=1/2 in the figure). This enhances the learning ability of convolution networks compared with the ResBlock module.

The filter number of 64 in the first CSPBlock module is divided into two convolutions of 32 filters and then combined with 3x3 convolution at the beginning of the backbone with 32 filters. The results are then passed to a 3x3 convolution with a filter number of 64 and combined with the shortcut from the first convolution with the filter number of 64.

The filter number of the second CSPBlock module is 128 and divides into two convolutions of 64 filters and then combined with a 3x3 convolution of the backbone with 64 filters. The results are then passed to a 3x3 convolution with a filter number of 128 and combined with the shortcut from the first convolution with the filter number of 128.

The last module starts with a convolution of 256 filters and divides into two convolutions with filters of 128 and then combined with a shortcut from the convolution with 128 filters. The results are then passed to a 3x3 convolution with filter number of 64. The result is combined with the shortcut from the first convolution with filter 256.

Secondly to improve the receptive field of the backbone, an SSP module was modified concatenating the max-pooling outputs with kernel sizes of 7, 10 and 13, respectively. The concatenation of these max-pooling outputs improved the accuracy of the architecture.

Thirdly, to be able to detect small faces from long distances, a modified PAN module was included to the backbone of the architecture with an additional bottom-up path augmentation added to the FPN architecture to aggregate features from low-level layers with more detailed information and the higher-level layers with more semantic information. An extra upsampling (3 downsampling with factors of 16, 8 and 4) was added to the PAN architecture in comparison with YOLOV4 to keep shallower features. The concatenation of the features from the bottom-up path goes through a 1×1 convolution.

3.2 Enhanced spatial pyramid pooling (SPP)

The second component, an enhanced spatial pyramid pooling (SPP) [16], is included at the end of the backup (marked as b) at the end of the backbone. This technique provides the capability to handle different sizes of the

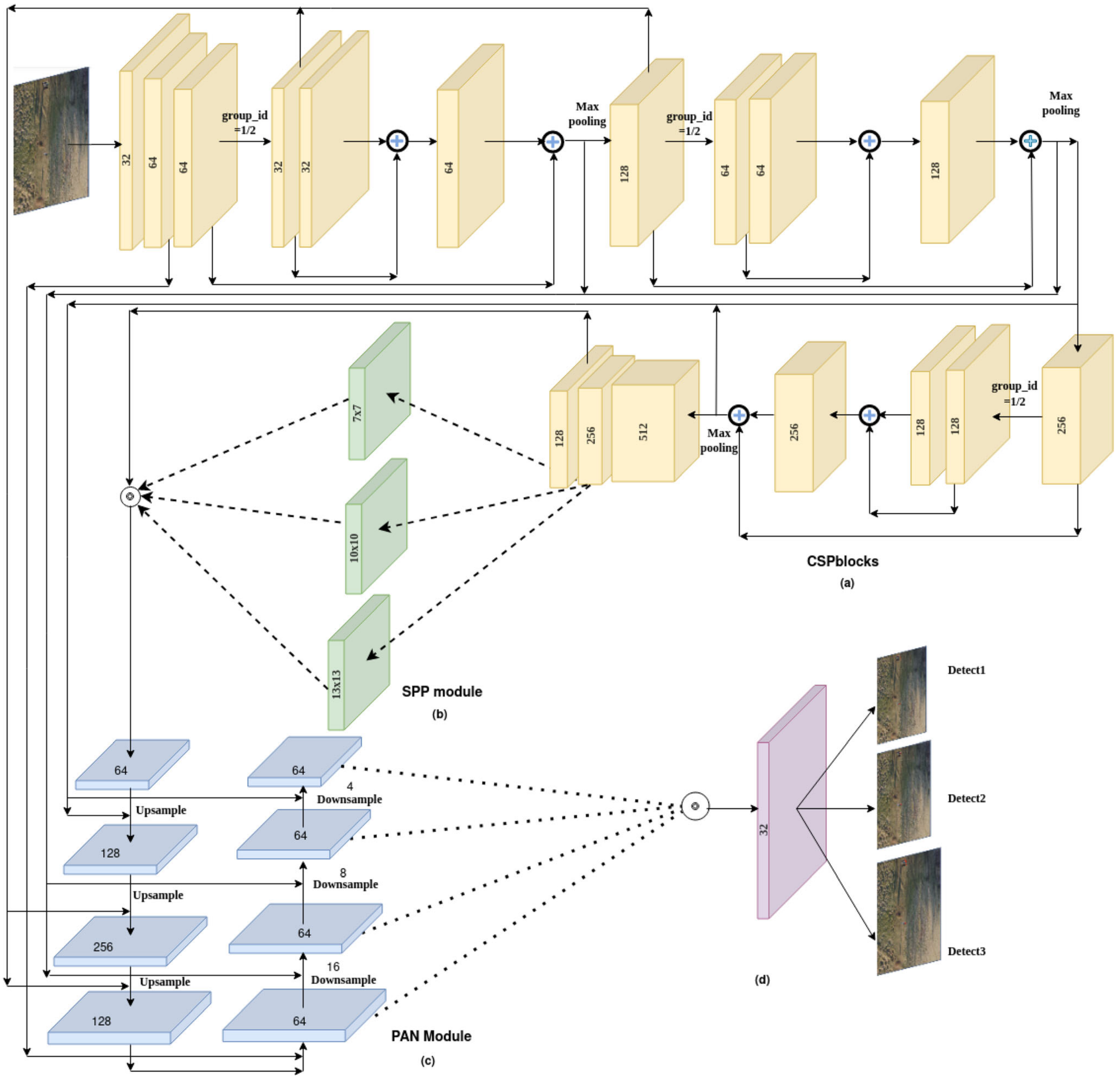


Fig. 2 The architecture of UWS-YOLO face detector

image input sizes and this capture multi-scale information from different spatial resolutions being this a key factor when detecting object at low resolutions from large images. By deploying this technique, the CNN become more robust when the UAV is flying at different altitudes because the algorithm is less affected by variations in the size of the faces.

3.3 Enhanced path aggregation network (PAN)

A modified path aggregation network (PAN) [17] module was also added (Marked as c) in the algorithm with three upsamplings (compared to YOLOv4). This technique enhances the capability of keeping the fine-grained features from the first layers along the computation of the following ones. This becomes an important matter when the details extracted from low-pixel size faces may be lost along the CNN.

The number of filters selected was 64, 128, 256, 128 and 64, respectively. The extra bottom-up path augmentation was down-sampled (3 downsamplings compared to two downsamplings in YOLOv4) with factors of 16, 8 and 4, respectively. The number of filters selected was 64. The features from the bottom-up path were then concatenated, and 1×1 convolution was run on the result. The coordinates, probability and confidence were obtained using YOLOv3 headers. The YOLOv3 headers were used to output the coordinates, probability and level of confidence. 19×19, 38×38 and 76×76 are the feature maps of the YOLOv3 header.

4 Implementation and deployment

This section presents the implementation and deployment of the UWS-YOLO in order to detect faces from UAV images and videos. It describes how the proposed solution is trained and tested in addition to the datasets used to evaluate results. Besides, further explanation is given regarding standards algorithms commonly employed to perform face detection in order to be compared against our algorithm.

4.1 Face detection datasets

Several algorithms are compared in terms of accuracy and speed. Every algorithm presented should be trained and evaluated over the same dataset in order to perform a fair comparison. In this manuscript, two datasets are presented. First, a publicly available dataset named WIDER FACE is mainly composed of images taken at ground level. Secondly, the authors of the manuscript created a face detection dataset recorded from a UAV: UAV-UWS dataset.

4.1.1 WIDER FACE

One of the largest datasets for face recognition is WIDER FACE dataset. It contains 32,203 images and 393,703 faces which is an average of 12 faces per image. The different illumination, scales, poses and occlusions included in this dataset make it a challenging task to achieve very high accuracy. WIDER FACE [22] is divided into three levels of difficulties, namely easy, medium and difficult. The accuracies in this research have been obtained by joining all three subsets together.

In order to allow other researchers to compare their algorithms against UWS-YOLO, this manuscript uses this dataset as the benchmark to compare our proposed solution with the state-of-the-art architectures; however, UWS-YOLO is optimized for low-pixel face recognition and not for close range images.

4.1.2 UAV-UWS dataset

In the scope of this research work, the authors have also created a dataset to test the algorithms in videos captured from a drone. This dataset was recorded using a DJI Mini 2 UAV. The frames obtained have a 4K resolution (3840×2160 pixels) and a frame rate of 30 fps.

In total, 20 people from different ethnicities were recorded for the dataset. For each person, a video of 30 s was recorded at 8 different distances from the drone to the face (2, 5, 7, 10, 15, 20, 25 and 30 m). Therefore, the dataset is composed of 144,000 different frames. The UAV was positioned at 30° above the face at each distance. Furthermore, the people recorded were asked to do different head movements to get a complete view of the whole face. Figure 3 shows an example of the head movements asked to every volunteer. Initially, participants were asked to execute a lateral head movement, looking first to the left and then to the right. Following this, a full circular movement with the head was made to cover the rest of the positions, including upward and downward positions. Finally, the volunteers were asked to stare directly at the drone, looking forward.

Due to the similarity of the faces in consecutive frames, the test dataset has been created extracting only one frame per second. Therefore, the dataset used for testing is composed of 4,800 images with a ratio of one face per image.

It is worth noting that GDPR (General Data Protection Regulation) compliance is a critical consideration in our

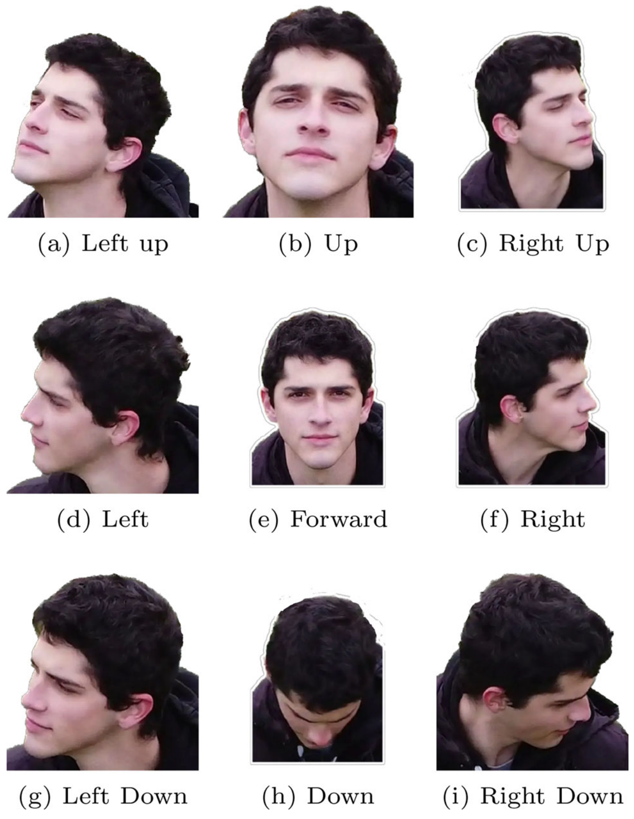


Fig. 3 Examples of the different face positions recorded in the UAV-UWS dataset

data-gathering process, as we are committed to respecting individuals' privacy and adhering to the regulations. While the dataset may be relatively small in comparison with some other studies, it has been meticulously curated to cater specifically to the objectives of our research. We emphasize that the dataset's size is appropriate for the testing and validation of our proposed model and offers a representative sample for the intended use case.

Table 2 shows the average face size, face at each distance of our dataset with two image sizes: the original 4K resolution and after being resized to 608 × 608. As can be seen, at higher than 15 m, the size of the faces in the resized image is less than 10 pixels. The smallest face is at 30 m where the average size is only 2 × 5 pixels in the resized image.

4.2 Face detection algorithms

UWS-YOLO is compared against RetinaFace [15] and YOLOv7 [35] in terms of accuracy, inference time, loading time of the model and model size. To compare all algorithms, these were trained in the same conditions over the same dataset. First, every algorithm was trained with the general dataset WIDER FACE. Then, each algorithm is evaluated over the testing dataset of WIDER FACE. Finally, to compare the accuracy of the algorithms when detecting faces from UAV images at high altitudes, the UAV-UWS dataset is used as a testing dataset; therefore, this dataset is only used for verification of the algorithms. This process is useful for comparing how each algorithm with the same training behaves at different distances.

Table 2 Distance from the camera to the face versus the size of the detected face in pixels (px) for the original resolution and the resized image

Distance	Size face	
	3840 × 2160 px	608 × 608 px
2 m	125 × 170 px	20 × 28 px
5 m	80 × 98 px	13 × 28 px
7 m	50 × 59 px	8 × 17 px
10 m	38 × 44 px	6 × 12 px
15 m	25 × 31 px	4 × 9 px
20 m	20 × 24 px	3 × 7 px
25 m	17 × 19 px	3 × 5 px
30 m	14 × 19 px	2 × 5 px

4.3 Hyperparameters

Table 3 shows the execution hyperparameters used to train UWS-YOLO, RetinaFace and YOLOv7.

4.3.1 RetinaFace–ResNet50

A momentum coefficient of 0.9 was used for training. The input size of the training images was 640 × 640 px, and the batch size used was 8. Moreover, the machine learning execution platform for RetinaFace was TensorFlow 2.5.3 [36].

4.3.2 UWS-YOLO

We utilize a momentum coefficient of 0.9 as the learning policy. An image input size of 608 × 608 px was used in our training with a batch size of 64. The machine learning execution platform used was Darknet [37].

4.3.3 YOLOv7

YOLOv7 was trained with a momentum coefficient of 0.9. 608 × 608 px is the size as the images are fed in YOLOv7. The batch size chosen is 8. The machine learning execution platform is Darknet [37].

4.4 Intersection over Union (IoU)

The Intersection over Union (IoU) is a fundamental metric used in object detection. The IoU measures the overlap between the predicted bounding box and the ground truth bounding box, providing a quantitative assessment of the accuracy of the detection. As shown in Fig. 4, the IoU is defined as the area of overlap (intersection) between the bounding boxes divided by the area of union.

Face detection is only one stage in the pipeline of our use case. Therefore, subsequent stages rely heavily on obtaining accurate face images to achieve high accuracy. For instance, in face verification tasks, having precise face images is crucial for obtaining successful results. Hence, an accurate bounding box around the face is of utmost

Table 3 Execution hyperparameters

Hyperparameters	RF-ResNet50	UWS-YOLO	YoloV7
Image size in pixel	640 × 640	608 × 608	608 × 608
Number of iteration	161000	40000	120000
Batch size	8	64	8
Momentum coefficient	0.9	0.9	0.9

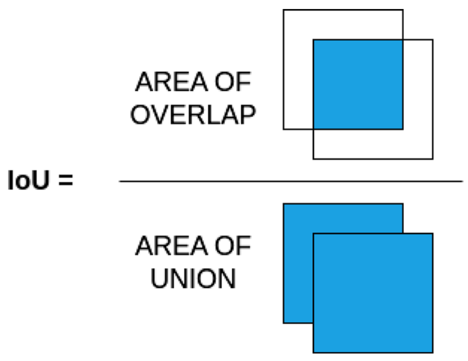


Fig. 4 IoU formula with example figures

importance, and this is where the IoU threshold becomes crucial.

Figure 5 illustrates three different face detections varying the IoU threshold. The green bounding box represents the ground truth, while the red one corresponds to the face detected by the model. With an IoU of 90% (Fig. 5a) the detected face closely matches the ground truth. When using an IoU of 75% (Fig. 5b) a small portion of the face is lost, but the results are still satisfactory for subsequent stages. However, with an IoU of 50% (Fig. 5c) a significant part of the face is lost, possibly even an eye from the bounding box. This makes it challenging for the next stages to achieve excellent results; however, the impact is not significant. In cases with an IoU below 50%, it becomes impossible to achieve good results in the next stages as more than half of the face may be lost.

Therefore, in this study, the accuracy of the models will be compared using these three IoU values: 90%, 75% and 50%.

4.5 Pipeline

The pipeline is divided into three stages as depicted in Fig. 6. First, the image is preprocessed, and then it is fed into the neural network. Finally, the results are post-processed to obtain the bounding boxes with confidence scores for each detection. The input to the pipeline is a single frame.

4.5.1 Preprocessing

The preprocessing stage consists of three steps aimed at preparing the captured frame for the neural network.

1. Resize: The image is resized to match the input size required by the neural network. In this process, the OpenCV resize function is utilized.
2. Scaling: The pixel values in each channel are scaled to fit within the range expected by the neural network.
3. Channel conversion: The image is converted from the BGR (Blue-Green-Red) color space to RGB (Red-Green-Blue). Moreover, the channel dimensions are transposed from (N, H, W, C) to (N, C, H, W), where N represents the number of images in a batch, C is the number of channels in the image, H is the height of the image, and W is the width.

4.5.2 Convolutional neural network (CNN)

The preprocessed image is then passed through a CNN. For this research, three different CNN models were employed: RetinaFace, YOLO-UWS and YOLOv7. Each model produces detection candidates along with their corresponding confidence scores.

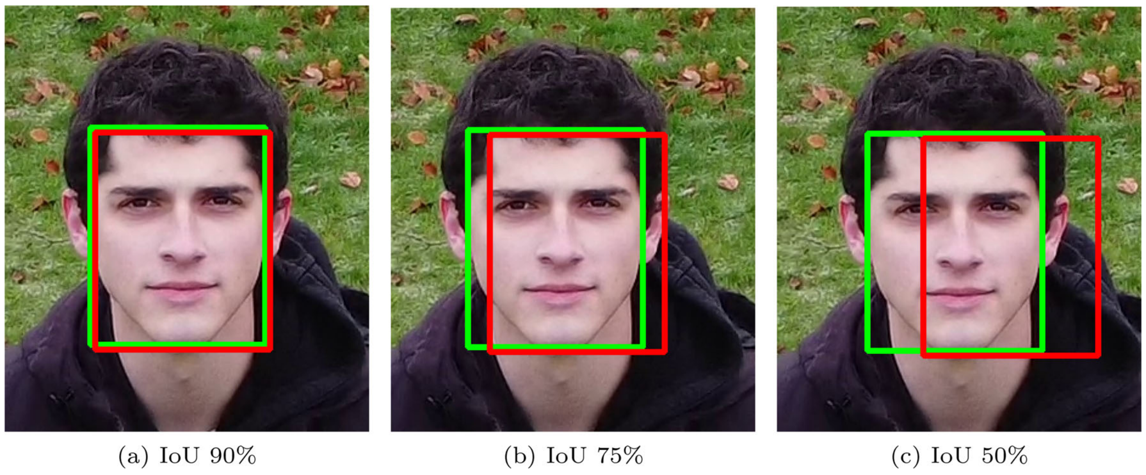


Fig. 5 Examples of face detections using three different IoU thresholds: 90%, 75% and 50%

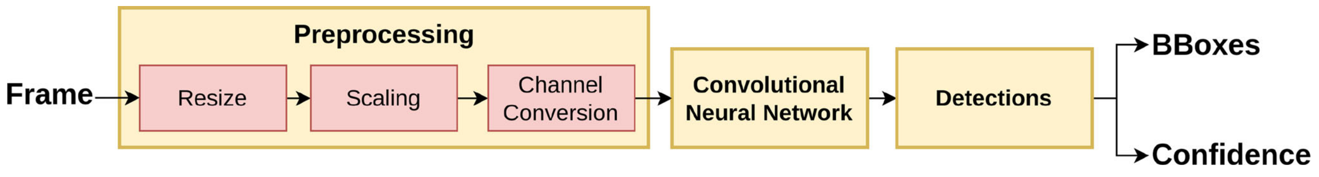


Fig. 6 IoU formula with example figures

4.5.3 Detections

A detection threshold is defined to determine the acceptance of a detection. If the confidence score of a detection is higher than the threshold, it is considered a valid detection; otherwise, it is discarded. Lowering the threshold will increase the number of true detections but also lead to an increase in false detections; therefore, it may reduce the accuracy. The choice of an appropriate threshold is crucial to achieve high accuracy with each model. Once the detections are confirmed, the bounding boxes are resized to match the dimensions of the original size.

The output of the pipeline consists of the bounding boxes for each detection, defined by the coordinates of the top-left corner of the bounding box, as well as its width and height. Furthermore, the pipeline provides the confidence score for each detection.

5 Empirical results and discussion

This section presents both quantitative and qualitative results from various experiments conducted to facilitate a comprehensive comparison of the algorithms. The quantitative results subsection shows the comparison of RetinaFace (with three different input sizes), YOLOv7 and UWS-YOLO on different metrics such as the inference time, the build time, the weights size and the accuracy of the algorithms. The last one has been analyzed on two different datasets—WIDER FACE and UAV-UWS. The qualitative results show what can our algorithm (UWS-YOLO) achieve on images at different distances.

5.1 Experimentation environment

The experiments have been carried out on a computer with an Intel(R) Xeon(R) E5-2630 v4 at 2.20 GHz with 20 cores and 32 GB of RAM. In addition, an NVIDIA GeForce GTX TITAN X with 12 GB of onboard memory with CUDA compatibility [38] was used. The Operative System (OS) used is Focal Ubuntu 20.04.3 with a Kernel version of 5.11.00.

5.2 Quantitative results

5.2.1 Accuracy

The accuracy of the models has been evaluated using three different approaches. Firstly, the WIDER FACE dataset was utilized, employing three IoUs (Intersection Over Union): 0.5, 0.75 and 0.9. Subsequently, the models were tested on the UAV-UWS dataset employing the same IoUs. Finally, they were tested on the UAV-UWS dataset but at every distance with a fixed IoU of 0.5. All the results have been obtained from experiments conducted specifically for this study.

Table 4 shows the results for the WIDER FACE dataset. Among the models, YOLOv7 exhibits the lowest accuracy. UWS-YOLO demonstrates modest performance and matches the accuracy of RetinaFace only when using an input size of 416 px and the most permissive IoU (50%). RetinaFace achieves the highest accuracy across all IoU, particularly with an input size of 1600 px.

Table 5 presents the accuracy of the models on the UAV-UWS dataset. Our model (UWS-YOLO) demonstrates exceptional accuracy, second only to RetinaFace with an input size of 1600 px. Compared with models with the same input size, UWS-YOLO achieves +12.7% better accuracy than YOLOv7 and +31.86% improvement over RetinaFace with an IoU of 50%. Even with a stricter IoU of 90%, UWS-YOLO outperforms YOLOv7 by +3.96% and RetinaFace by +2.33%.

Table 6 shows the accuracy of the models on the UAV-UWS dataset for each of the eight distances recorded: 2, 5, 7, 10, 15, 20, 25 and 30 m. At a distance of 2 m, all models

Table 4 Accuracy of UWS-YOLO and RetinaFace for WIDER FACE dataset varying IoU percentage

Models	IoU		
	90%	75%	50%
RetinaFace—416 px	2.35%	43.61%	72.40%
RetinaFace—608 px	3.80%	53.60%	81.96%
RetinaFace—1600 px	4.76%	63.60%	93.05%
YOLOv7—608 px	0.69%	29.62%	65.43%
UWS-YOLO—608 px	1.66%	32.98%	72.26%

Table 5 Accuracy of UWS-YOLO and RetinaFace for UAV-UWS dataset varying IoU percentage

Models	IoU		
	90%	75%	50%
RetinaFace—416 px	1.34%	9.52%	14.85%
RetinaFace—608 px	4.81%	19.17%	27.43%
RetinaFace—1600 px	24.14%	51.24%	64.74%
YOLOv7—608 px	3.18%	32.91%	46.59%
UWS-YOLO—608 px	7.14%	37.86%	59.29%

achieve similar results, but both YOLO variants (YOLOv7 and UWS-YOLO) achieve the highest accuracy, exceeding 87% of accuracy. At 5 m, better results are observed, except for RetinaFace with an input size of 416 px, which fails to detect faces beyond 5 m. Moreover, RetinaFace with an input size of 608 px can only detect faces up to 10 m, where it achieves an accuracy of merely 0.35%. This indicates that RetinaFace with a small input size is only able to detect faces accurately at short distances.

RetinaFace with an input size of 1600 px, YOLOv7 and UWS-YOLO is the only model that achieves good results at longer distances (more than 10 m) as indicated in the table. At 25 and 30 m, YOLOv7 struggles to detect any face, resulting in significantly low accuracy (0.15% and 0.01%, respectively). In contrast, UWS-YOLO achieves an accuracy of more than 8% at 25 m and 2.22% at 30 m, surpassing even RetinaFace with an input size of 1600 px by +1.83%.

These results demonstrate that our model UWS-YOLO performs well at both short and long distances, surpassing, at some distances, models with nearly three times larger input sizes. Furthermore, when compared to models with the same input size, UWS-YOLO consistently accomplishes superior results across all distances.

Table 7 Build time, weights size and inference time for the three compared models with input size of 608 px

Model	Build time	Weights size	Inference time
RetinaFace	2426 ms	128 MB	26 fps (38 ms)
YoloV7	4040 ms	140 MB	24 fps (41 ms)
UWS-YOLO	1552 ms	49 MB	91 fps (11 ms)

5.2.2 Built time, weights size and inference time

Table 7 provides a comparison of the built time, the weights size and the inference time for the three models under consideration. The values shown are with the same input size of 608 px for all three models. YoloV7 exhibits the longest building time, taking approximately 4 s, and it has also the biggest weights size of 140 MB. In contrast, RetinaFace has a build time of around 2.5 s with a weights size close to 130 MB. Notably, our proposed model, UWS-YOLO, demonstrates the fastest build time of only 1.5 s, showcasing its speed and stands out as the most lightweight model with only a weights size of 49 MB.

Regarding inference time, a correlation can be observed with the weights size. Our proposed model achieves the fastest inference time, merely 11 ms, equivalent to 91 fps. In contrast, RetinaFace has an inference time of 38 ms (26 fps) while YoloV7 is the slowest with 41 ms (24 fps). Therefore, our model stands as the only one capable of achieving real-time processing, defined at a frame rate of 30 fps.

Our proposed model is +345% faster than RetinaFace achieving better accuracy when evaluated on the UAV-UWS dataset. Furthermore, in comparison with YoloV7, our model outperforms it significantly, being +373% faster and showcasing better accuracy across all tested datasets, including WIDERFACE.

Table 6 Accuracy at different flying distances = IoU 50%

Models	Metric type	Distance							
		2 m	5 m	7 m	10 m	15 m	20 m	25 m	30 m
RetinaFace - 416 px	mAP	82.64%	29.71%	0%	0%	0%	0%	0%	0%
RetinaFace - 608 px	mAP	83.29%	84.75%	45.04%	0.35%	0%	0%	0%	0%
RetinaFace - 1600 px	mAP	83.60%	92.18%	98.63%	90.81%	82.44%	55.54%	10.62%	0.39%
YOLOv7 - 608 px	mAP	87.67%	92.05%	84.65%	60.42%	36.23%	4.95%	0.15%	0.01%
UWS-YOLO - 608px	mAP	87.58%	93.52%	92.90%	75.62%	60.15%	37.35%	8.54%	2.22%

5.2.3 Inference time and mAP comparison over different input sizes

Figure 7 shows the relation between the accuracy and the inference time in our UAV-UWS dataset. It has been compared using the three models—RetinaFace, YOLOv7 and UWS-YOLO—with three different input sizes—416, 608 and 1600 px. In the figure, the leftmost area indicates the fastest models, while the area at the top indicates models with higher accuracy. The figure also shows where is the real-time processing defined as 33 ms (30 fps). Any model on the left of this vertical line will execute real-time detections.

YOLOv7 and RetinaFace models only achieve real-time processing with their smallest input size (416), while our model achieves it even with an input size of 608. Furthermore, not only our model is faster than the others at these input sizes (416 and 608) but also it achieves a higher accuracy. Using an input size of 416, our model is +371% faster than RetinaFace and has an accuracy improvement of +18.31%. Moreover, it is +314% faster than YOLOv7 and better accuracy by +3.54%.

Our model was trained with an input size of 608 px; therefore, the best results will be when this input size is used. Figure 7 reflects this. Our model is the upper leftmost compared to the other models. As mentioned in previous sections, our model with this input size is +345% faster than RetinaFace and +373% than YOLOv7. It also has +31.86% and +12.7% higher accuracy than RetinaFace and YOLOv7, respectively.

On the other side, with an input size of 1600 px, our model is faster than the other models (+330% than RetinaFace and +365% than YOLOv7), but it has lower accuracy (−3.68% than RetinaFace and −8.07% than YOLOv7).

Although RetinaFace-1600px achieves the highest accuracy (69.16%), its inference time is also really high (208 ms). Moreover, UWS-YOLO does not show a great accuracy difference when increasing the input size to 1600 px (1.8% more), but the inference time increased significantly (46 ms more), being slower than real time. This can be seen in Fig. 7. Therefore, based on the trade-off between high speed and high accuracy and that real-time processing is needed, our algorithm UWS-YOLO-608px will achieve the best results.

5.3 Qualitative results

Figure 8 shows frames from our UAV-UWS dataset at four different distances and head positions. Each frame contains only one person and therefore one face. Figure 8a shows a volunteer at 2 m with the head facing right down. The face was detected by our model with a confidence score of 58%. The low confidence score can be attributed to the person's face facing down, resulting in an obscured facial appearance. This demonstrates the capability of our algorithm to detect faces under conditions where not all facial features are readily visible. Figure 8b shows the same volunteer in the same position but at a distance of 7 m. Our model is able to detect the face with a confidence score of 61%. Again, this confidence score can be attributed to the downward-facing orientation of the face.

In Fig. 8c the person is at 20 m and is looking directly at the drone. The face is detected with a confidence score of 81%. The model achieves a high confidence score even at far distances as the person is facing the drone and all the face features can be appreciated. Finally, Fig. 8d shows the same person staring at the drone but at 30 m, the maximum distance of our dataset. Our model is capable of detecting the face, but only with a confidence score of 13%. It is worth reminding that at 30 m, the size of the face in the

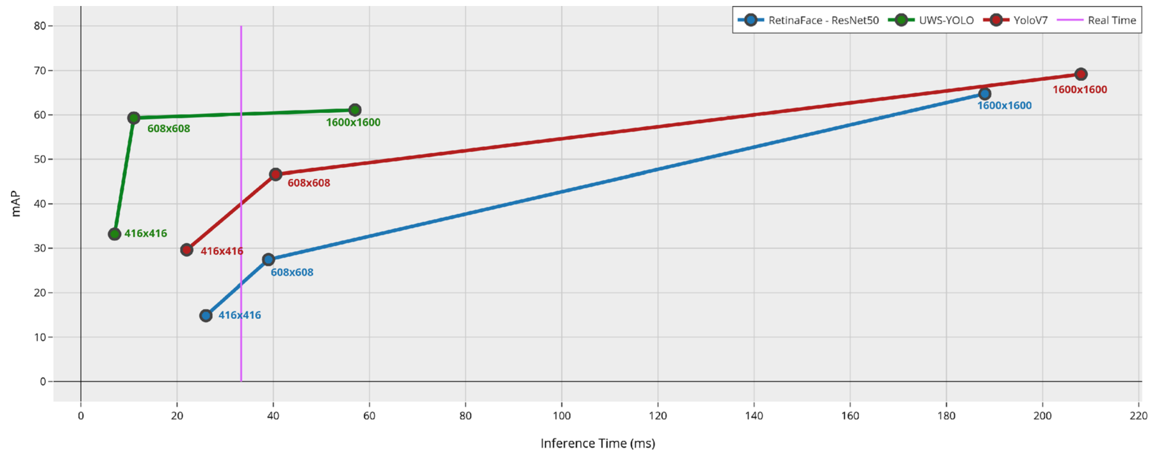
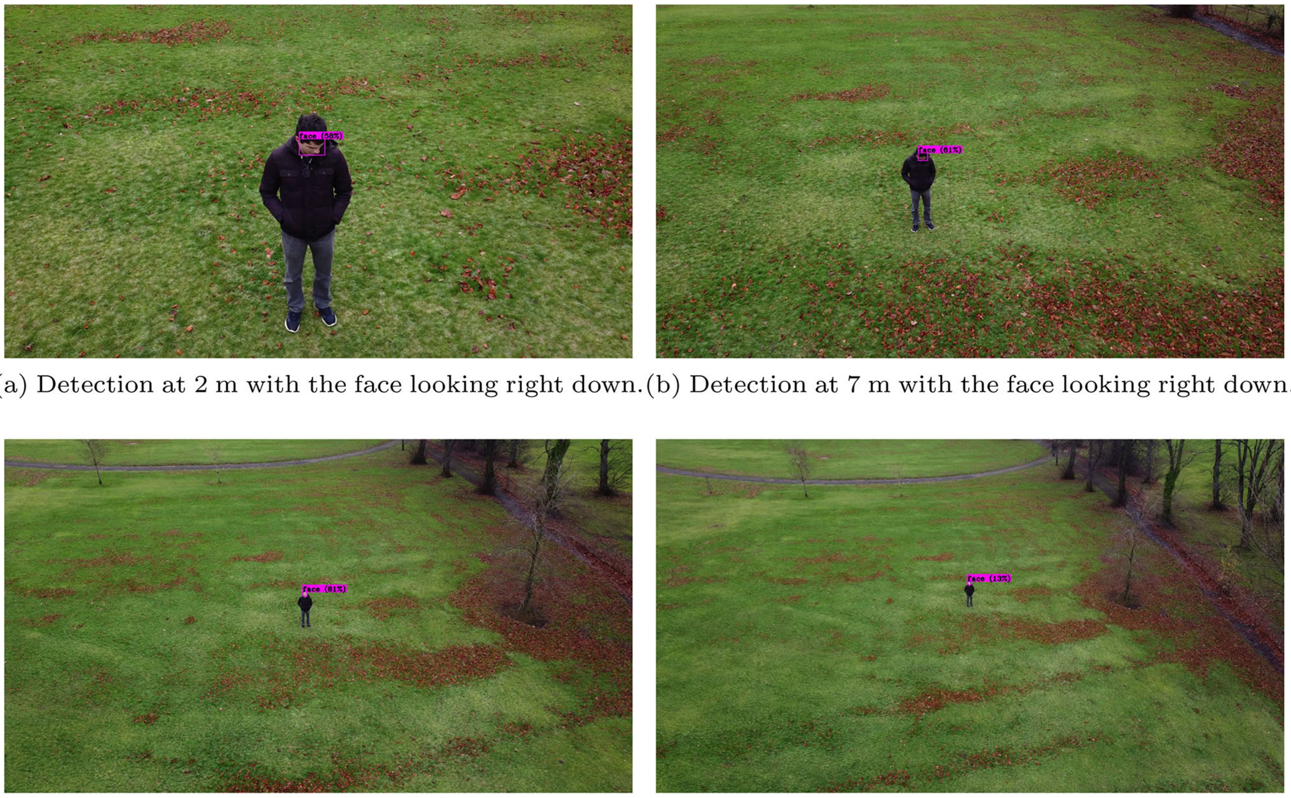


Fig. 7 mAP accuracy with IoU 0.5 in the UAV-UWS dataset versus its inference time in milliseconds using three different input sizes: 416, 608 and 1600



(a) Detection at 2 m with the face looking right down.

(b) Detection at 7 m with the face looking right down.

(c) Detection at 20 m with the face looking forward directly to the drone.

(d) Detection at 30 m with the face looking forward directly to the drone.

Fig. 8 UWS-YOLO detections at 4 four distances in frames from our UAV-UWS dataset

resized image (608×608) is 2×5 px. Therefore, the confidence score is low as the resolution of the face is small and most of the face features have been lost.

6 Conclusions

Face detection is a widely studied task in research, primarily focusing on achieving high accuracy. However, most existing solutions neglect the crucial aspect of low execution times, which are essential for real-time performance and subsequent face identification processes. Moreover, face detection from UAV imagery poses additional challenges due to factors such as face posing, angle variations, camera inclination, and drone vibrations.

This manuscript introduces a novel CNN-based algorithm, UWS-YOLO, specifically designed to address these limitations. Our algorithm demonstrates high-accuracy face detections with minimal inference times from UAV videos. To evaluate the performance of UWS-YOLO, we conducted comprehensive training and evaluation experiments, comparing it against SOTA algorithms, including RetinaFace. The evaluations were performed on both a standard dataset (WIDER FACE) and a dataset collected by the authors using a UAV. Empirical evaluation has been conducted, and UWS-YOLO outperforms RetinaFace by a significant margin, achieving an accuracy rate of 59.29% compared to RetinaFace's 27.43%. Notably, UWS-YOLO surpasses RetinaFace's capabilities by successfully detecting faces even beyond a distance of 10 m. Furthermore, in terms of speed, UWS-YOLO exhibits exceptional efficiency, performing 345% faster than RetinaFace.

The key advantages of UWS-YOLO can be summarized as follows: superior speed, remarkable accuracy and the ability to handle the complexities of face detection in UAV operations. These findings position UWS-YOLO as a balanced and highly suitable algorithm for face detection in UAV applications. In summary, the presented research has created a cutting-edge solution that surpasses previous algorithms in terms of speed and accuracy, while also addressing the unique challenges associated with face detection from UAV imagery.

Acknowledgements This work was in part funded by the EU Horizon 2020 5G-PPP 5G-INDUCE project ("Open cooperative 5G experimentation platforms for the industrial sector NetApps") under the Grant number H2020-ICT-2020-2/101016941, the EU Horizon 2020 ARCADIAN-IoT project ("Autonomous Trust, Security and Privacy Management Framework for IoT") under the Grant number H2020-SU-DS-2020/101020259 and the EU Horizon INCODE project

("Programming Platform for Intelligent Collaborative Deployments over Heterogeneous Edge-IoT Environments") under the Grant number HORIZON-CL4-2022-DATA-01-03/101093069. The authors would like to thank all the partners in these projects for their support.

Data availability The UAV-UWS dataset used in the current study is not publicly available due to the General Data Protection Regulation (GDPR) requirements as it contains personally identifiable information.

Declarations

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

References

1. Martinez-Alpiste I, Golcarenarenji G, Wang Q, Alcaraz-Calero JM (2021) Search and rescue operation using UAVs: a case study. Expert Syst Appl 178:114937. https://doi.org/10.1016/j.eswa.2021.114937
2. Golcarenarenji G, Martinez-Alpiste I, Wang Q, Alcaraz-Calero JM (2021) Efficient real-time human detection using unmanned aerial vehicles optical imagery. Int J Remote Sens 42(7):2440–2462
3. Golcarenarenji G, Martinez-Alpiste I, Wang Q, Alcaraz-Calero J-M (2022) Illumination-aware image fusion for around-the-clock human detection in adverse environments from unmanned aerial vehicle. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2022.117413
4. Kumar A, Kaur A, Kumar M (2019) Face detection techniques: a review. Artif Intell Rev 52(2):927–948
5. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, vol 1. IEEE
6. Liu C, Wechsler H (2003) Independent component analysis of Gabor features for face recognition. IEEE Trans Neural Netw 14(4):919–928
7. Triantafyllidou D, Nousi P, Tefas A (2018) Fast deep convolutional face detection in the wild exploiting hard sample mining. Big Data Res 11:65–76
8. Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
9. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in Neural Information Processing Systems, vol 28. Curran Associates Inc., New york, pp 91–99
10. Dai J, Li Y, He K, Sun J (2016) R-FCN: Object detection via region-based fully convolutional networks. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds) NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems, vol 29. Curran Associates Inc, New York, pp 379–387
11. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
12. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
13. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767
14. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) SSD: single shot multibox detector. In: Proceedings of the European conference on computer vision (ECCV), pp 21–37. Springer
15. Deng J, Guo J, Ververas E, Kotsia I, Zafeiriou S (2020) RetinaFace: single-shot multi-level face localisation in the wild. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5203–5212
16. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
17. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8759–8768
18. Daryanavard H, Harifi A (2018) Implementing face detection system on UAV using raspberry pi platform. In: Electrical engineering (ICEE), Iranian conference On, pp. 1720–1723. https://doi.org/10.1109/ICEE.2018.8472476
19. Bradski G, Kaehler A et al (2000) Opencv. Dr. Dobb's journal of software tools, vol 3(2)
20. Hsu H-J, Chen K-T (2017) DroneFace: an open dataset for drone research. In: Proceedings of the 8th ACM on multimedia systems conference, pp187–192
21. Chang J, Lu Y, Liu Y, Zhou B, Qiao H (2020) Long-distance tiny face detection based on enhanced YOLOV3 for unmanned system. arxiv: abs/2010.04421
22. Yang S, Luo P, Loy C-C, Tang X (2016) Wider face: a face detection benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5525–5533
23. Srivastava A, Badal T, Saxena P, Vidyarthi A, Singh R (2022) UAV surveillance for violence detection and individual identification. Autom Softw Eng 29(1):1–28
24. Pu Y-H, Chiu P-S, Tsai Y-S, Liu M-T, Hsieh Y-Z, Lin S-S (2022) Aerial face recognition and absolute distance estimation using drone and deep learning. J Supercomput 78(4):5285–5305
25. Tang J, Peng X, Chen X, Luo B (2021) An improved mobilenet-SSD approach for face detection. In: 2021 40th Chinese control conference (CCC), pp 8072–8076. IEEE
26. Chen BLS, Cheah DS, Chan KW, Nugroho H (2020) Person identification system for UAV. In: International conference on innovative technology, engineering and science, pp 325–335. Springer
27. Wang L, Siddique AA (2020) Facial recognition system using LBPH face recognizer for anti-theft and surveillance application based on drone technology. Meas Control 53(7–8):1070–1077
28. Jiang C, Ma H, Li L (2022) IRNet: an improved retinanet model for face detection. In: 2022 7th international conference on image, vision and computing (ICIVC), pp 129–134. IEEE
29. Liang P, Wu W, Liao S, Liu S, Duan Y, Zhou Z, Zhang Y (2022) Face detection using YOLOX with attention mechanisms. In:

2022 10th international conference on information systems and computing technology (ISCTech), pp 457–462. IEEE

30. Wang G, Li J, Wu Z, Xu J, Shen J, Yang W (2023) Efficientface: an efficient deep network with feature enhancement for accurate face detection. arXiv preprint arXiv:2302.11816
31. Mamieva D, Abdusalomov AB, Mukhiddinov M, Whangbo TK (2023) Improved face detection method via learning small faces on hard images based on a deep learning approach. Sensors 23(1):502
32. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934
33. Wang C-Y, Liao H-YM, Yeh I-H, Wu Y-H, Chen P-Y, Hsieh J-W (2020) CSPNet: a new backbone that can enhance learning capability of CNN. In: 2020 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 1571–1580
34. Wang C-Y, Bochkovskiy A, Liao H-YM (2021) Scaled-yolov4: scaling cross stage partial network. In: Proceedings of the IEEE/cvf conference on computer vision and pattern recognition, pp. 13029–13038
35. Wang C, Bochkovskiy A, Liao H (2022) Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arxiv 2022. arXiv preprint arXiv:2207.02696
36. ...Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. https://www.tensorflow.org/
37. Redmon J. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/ (2013–2016)
38. Nickolls J, Buck I, Garland M, Skadron K (2008) Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? Queue 6(2):40–53