Results in Engineering 29 (2026) 109495

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

FLOOD-DEPTH-ML: Machine learning-driven python application for estimation of urban flood depths through submerged vehicles detection

Mayank Mishra [a], Raffaele Albano [b,*]

[a] Department of Engineering, University of Basilicata, 85100, Italy
[b] Department of Health Science, University of Basilicata, 85100, Potenza, Italy

ARTICLE INFO

Keywords:
Flood depth
Machine learning
YOLO algorithm
Computer vision
Climate change
Urban floods

ABSTRACT

Climate change has caused an increase in floods worldwide that affect the lives of people and cause extensive property damage in urban areas due to high flood depth levels. Machine learning-based computer vision applications have been extensively used for the estimation of flood depth levels in urban environments. However, most applications are restricted to research purposes with their on-site usability remaining low and often fail to communicate the flood risk to the public in simple terms. In this study, we present a Python application that uses backend a you look only once (YOLO)-based detector with a simple graphical user interface (GUI) to classify vehicle inundation levels in five classes and help communicating flood risk. The Python application called FLOOD-DEPTH-ML available as open access allows users to analyze flood images/videos, online YouTube links, and most importantly its webcam feature, which users can use to easily integrate it with monitoring cameras to provide early warning for flood depths based on car submergence levels.

1. Introduction

With constant climate changes and urban expansion, the frequency and magnitude of flood events in urban areas have increased markedly, resulting in property damage and loss of life. Urban expansion generally leads to an increase in impervious surfaces and an expansion of artificial drainage networks; this results in significant changes in the magnitude of water and the velocity of surface runoff phenomena during extreme rainfall events [1]. The presence of less permeable surfaces increases the amount of precipitation that transforms into surface runoff because the infiltration is reduced, and it alters the development times of these phenomena, as the pathways followed by water flow change and exacerbate flood events. For example, flash floods in the Valencia region of Spain in October 2024 turned the narrow passages of century-old streets into death traps, resulting in more than 200 deaths. Also in Europe, floods occurred recently in Spain in October 2024 [2], succeeding those in Slovenia (August 2023) and Germany, the Netherlands, and Belgium (July 2021). In Italy, flash floods erupted in 2011 in Genoa [2]. The most recent ones in Texas, U.S.A in July 2025 where river rose by about 8 m within 45 min resulting in more than 100 fatalities.

In these flood events, the velocity of the water flows and the depth levels of the water are the main hydraulic characteristics to assess the direct consequences of flooding associated with damage to infrastructure, the toppling of pedestrians, the submersion of vehicles and the loss of life [3]. In fact, recent events have shown that the depth of the water is directly proportional to the extensive damage to the infrastructure and the loss of life [4]. In addition, urban flash flood casualties typically involved people trapped in their cars on flooded roads or trying to escape the rapidly rising water [5]. Hence, it is paramount to measure depth of water in flood events and warn the general public in real-time about flood risk owing to rising water levels.

However, water level sensors are not installed in urban flooded locations to measure flood depth. In this context, non-contact techniques such as computer-vision (CV) have recently received attention for measuring water levels and serve as an alternative to traditional water level sensors [6]. Also, videos taken by citizens and traffic cameras video feed when processed using CV techniques can provide comprehensive information about the depth of water and help calculate the associated risks.

These CV-based applications have several barriers, such as ease of use, as knowledge of coding etc. is needed as a prerequisite to use them, making their applicability by practitioners quite low. Some software applications in the field of hydrology are available, these applications have value for the analysis of flow velocity; for example, the SSIMS-Flow software, which is used for the estimation of the velocity in channel flows [7]. Most software applications in this field lack several key features, such as live counting of cars with various risk levels, generation of an analysis report that has all the information of the analyzed video feed for visualization purposes, ability to decrease analyzed frames per second

* Corresponding author.
E-mail addresses: mayank.mishra@unibas.it (M. Mishra), raffaele.albano@unibas.it (R. Albano).

https://doi.org/10.1016/j.rineng.2026.109495
Received 12 December 2025; Received in revised form 22 January 2026; Accepted 7 February 2026
Available online 10 February 2026

(FPS) for faster processing, and most importantly ease of use of the application. Furthermore, our application can be used directly on site for real-time analysis with Webcam feature without the training required for the model.

To address this gap, we developed a web-based customizable application in the Python environment with backend YOLO framework. We developed our own code that can address the limitations of current software and be easily used with a GUI. FLOOD-DEPTH-ML comes with a.py file and readme file where all requirements are stated, and the tool can be directly installed and ready-to-use immediately once these packages are installed. Other applications based on YOLO include improved object detection for autonomous vehicles [8], segmentation of concrete cracks in bridges [9], detecting indoor objects [10] and evaluating guardrails presence for bridge safety applications [11].

The objective of the present research is to build and validate a python tool based on CV technique that detects vehicle inundation levels and maps them to practical risk categories from video feeds obtained from generic surveillance cameras, citizen-recorded videos, or video footages from the internet and/or webcams. In addition, the tool is tested on uploaded images, videos and live camera input for the Matera city case study, reporting car counts by risk levels, and demonstrating the tool workflow.

2. Related works on flood-depth measurement

According to a literature review, some studies have deployed object detection models based on deep learning techniques for processing flooded car images to estimate flood depth, while some have focused on processing submerged pedestrian images [12]. For example, moving objects, such as vehicles, solid objects, and pedestrians, which can be easily detected using object detection techniques serve as an indirect indicator of flood depths. In this regard, video feeds provide detailed information on flood or hydrological characteristics, and ML models provide accurate forecasts for early warning systems that could provide the necessary time for the public to react in the event of flood events [13] and powerful tool to inform emergency management services [14]. In addition, using simple terminology, such as “water is up to the ankle” and “water has reached the window level of a car”, can help understand water levels in flooded regions, thus facilitating better risk communication to the general public. Communication of risk levels in simple terminology will help people accurately judge the degree to which they should prepare based on live gauged information. In addition, measuring the depths of the current water level is of paramount importance for risk management and several other applications to calculate the stability of vehicles and pedestrians in moving water and damages to buildings/streets.

Liu et al. [15] reviewed and summarized various machine learning (ML) applications for estimating flood depth. The authors reviewed about 108 relevant articles and concluded the necessity of integrating of flood-depth monitoring with smart city framework and disaster prevention. The flood-depth measurements are mainly divided into machine learning-based regression models that have governing factors that act as input and flood depths as output, secondly the vision-based methods and hybrid models that combine ML with physical models and models combining ML with remote sensing data.

2.1. Quantitative methods for flood-depth estimation

These methods give a number or a range instead of a textual description of flood depth. Hao et al. [16] were the first to use YOLO version 3 for detecting urban pond levels based on surveillance videos. They classified the pond level into the following: Level 1, corresponding to a height below 1/3 of the wheels; Level 2, corresponding to the water level below the fog lights, and Level 3 reflecting a depth above the fog lights, potentially interfering with the opening of car doors. The section summarizes quantitative approaches that provide an estimate of the flood depth or an water depths range. This water depths range is very important in predictions of water levels dangerous for cars in the streets as they give time users to evacuate as there is gradual transition from safe level to unsafe levels. Relevant studies are summarized in Table 1. In a recent study, [17] used a deep neural network based on state-of-the-art YOLOv8 and its enhanced version bidirectional feature pyramid network (BEW)-YOLOv8 to estimate flood depths, with “submerged cars” images as the input. The authors also used images from social networks and search engines to enhance their dataset for training models. They used five classes from Class 0 to Class 4, with each class corresponding to a depth level; for example, Level 2 indicates a depth of 35-75 cm and that the upper and lower parts of the wheels of a car are submerged in water. In terms of performance metrics, their model could achieve a mean average precision (mAP) score (which is a performance metric for YOLO and other DL models that encapsulates precision and recall across multiple classes) of approximately 0.831. Wan et al. [18] focused on “citizen sensor objects,” such as vehicles in flooded environments, and used a deep learning technique such as YOLO and its advanced version distributed shifted convolution (DSC)-YOLOv8b to estimate flood levels, achieving an mAP50 (precision at an intersection over union i.e. IoU threshold of 0.50) of 75.4%. In their previous research, [19], used YOLOv8 and the same 5 levels of flood hazard to demonstrate how video images can be used as the data source for monitoring urban flood levels[20]. used a pre-trained, large multimodal model, namely generative pre-trained transformers (GPT-4) Vision or FloodDepth-GPT, to measure flood depths in urban environments, using standard objects such as cars, stop signs, and persons, and by inputting the right query, obtaining flood depth with reasonable accuracy. FloodDepth-GPT, can be used for estimating flood depth [20]; however, they are better suited for images and cannot be integrated with surveillance videos. Sazara et al. [21] used a novel deep learning method based on the images of vehicles in side view; however, their calculation approach was limited to measuring flood depth at tire full height only. Notarangelo et al. [22] used STURM-FloodDepth tool based on DL-framework was applied to urban flood depth measurement based on levels of submerged cars on 2021 Luxembourg floods real-case study. The authors also used super-resolution enhancement (EDSR) to improve low-resolution vehicle images into high resolution which in turn increased models performance. Apart from using car as reference objects, [23] used instead buses for estimating flood depths using four levels and also took into account complex site conditions where the scene was interfered by other objects and multiple camera angles. Although, light and poor image quality significantly affected the accuracy of the model to detect flood depth levels. Furthermore, Lin et al. [24] considered data collection from social media images via automated web crawler that offered data collection and works with the descriptions of flood images. The authors used DL-framework YOLO world that was able to work not only for cars, but also for other reference objects such as bicycle, SUV, bus etc. and achieved a mean square error in the range of 85.3%-94.4%.

2.2. Qualitative methods for flood-depth estimation

Qualitative approaches provide information in terms of visual criteria that could later be used to estimate the depth of the flood, rather than providing a direct estimate. These approaches are summarized in Table 2. These investigations estimate the depths of the water based on qualitative information rather than giving a number. Zhong et al. [25] deployed YOLO version 4 with real-time capabilities of 30.49 frames per second (FPS) for submerged objects such as vehicle exhaust pipes and pedestrian legs, achieving a mean average precision of approximately 89%. They concluded that larger objects, such as cars, serve as a more reliable measure of water depths than smaller objects, such as pedestrians. Based on the findings, the authors reported that in addition to giving actual number descriptions of water levels such as ‘Car none’, ‘Car pipe’, ‘Car handle’ and ‘Car roof’, these methods also provide an estimate of urban flood level levels. Wu et al. [26] evaluated the performance of five deep learning models: AlexNet, MobileNet-V2,

Table 1
Studies employing quantitative methods based on deep learning frameworks for flood depth estimation.

Reference	Case study and dataset size	DL-technique used	Flood levels	Accuracy/mAP/Precision/recall/MAE
[16]	840 pictures of sedan cars from three sources: media images, synthetic images, and surveillance video images	YOLO, version 3	Mainly four levels, level 0, 1 (0–15 cm), 2 (15–40 cm), 3 (40–60 cm)	mAP = 78%
[17]	Main pictures of submerged damaged cars from flooding events in Henen, China and other images from search engines, 1244 images	Mainly 2 YOLOv8 and BEW-YOLOv8	Five levels 0 (0), 1 (0–35), 2 (35–75), 3 (75–105), 4 (>105 cm)	mAP@0.5 = 0.831
[18]	Flooded vehicle dataset with 2000 images and 6300 labeled vehicle (Sedan, SUV, and Van) provided by traffic cameras or searched from social networks	DSC-YOLOv8n, YOLOv8n, and other YOLO family models	Five categories corresponding to five flood levels Level 0, 1, 2, 3, 4	mAP50 = 75.4%
[19]	2000 images featuring 6300 vehicle objects obtained from Microsoft Bing, Google, and Baidu searching with the key words “urban flooding” and “urban flood.”	YOLOv8	Vehicle flooding status into 5 levels	Median mAP50% = 66.4-70%
[20]	150 flood photos from various online sources	A pre-trained large multimodal model	Only number estimates	Mean average error (MAE) of 25 cm
[21]	663 images for semantic segmentation and 1567 images for object detector	Fully Convolutional Networks FCN (VGG16) + U-Net	Level of water depth as a fraction of the wheel size	MAE = 0.038 with 45 real-world images
[22]	2000 flood photos and proprocessing applied to increase size to 6300 image patches	vehicle detection (YOLO-World and Slicing Aided Hyper Inference SAHI) and Fine-tuned ResNet-50 for flood level classification	Level 0–4	Precision = various depending on level (0.66, 0.43 0.64, 0.67, 0.91)
[23]	1008 images of buses augmented to 2184 images	YOLOv8 models	level 1 i.e. 0-20 cm, 2 i.e. 20-45 cm, 3 i.e 45-100 cm and 4 > 100 cm	61.2–66.0
[24]	2492 high quality flood images collected via data mining	YOLO-World for multi-instance detection	depth-labeling guidelines vary for various objects considered such as car, SUV, MPV, bicycle and bus	mean square error = 85.3% (for car) to 94.4% (for bicycle)

GoogleNet, ResNet50, and CBAM improved ResNet50, among which CBAM-improved ResNet50 offered the highest precision of 92.45% in the test dataset. The authors used grades A, B, and C to quantify the level of submergence for vehicles, pedestrians, and bicycles. Extending these findings, Sun et al. [27] used car types, such as ordinary, sports, and sport utility vehicle (SUV) in their labeling standards and incorporated pictures with rainfall in the background to determine flood levels using the YOLOv9 framework. Data were labeled safe, dangerous, and unsafe according to the risk of flood levels for cars, with precision levels of 74.6%, 69.1%, and 93.9% obtained for the three datasets used. Du et al. [28] used up to 10 levels to classify the risk to pedestrians and cars in flood events. They used processed images not only of cars but also of other standard objects, such as pedestrians, to estimate flood depths in real time. Jiang et al. [29], in a recent study, used the YOLOv5 model to detect and segment key points in the human body using images of pedestrians submerged in floodwater, reporting them as an indicator to detect depths of floodwater with intuitive interpretability and to assess pedestrian risk and emergency rescue. Pally and Samadi [30] developed a new ’FloodImageClassifier’ based on various deep learning frameworks, such as YOLOv3. Region-based CNNs, which can not only perform identification and segmentation tasks and but also calculate flood water levels and provide real-time monitoring of flood conditions by embedding input feeds from traffic cameras. Nevertheless, these studies do not give a range of flood depth levels but they can be estimated by considering heights of standard objects such as, car height, person height etc.

2.3. Related works using other reference objects

Furthermore, in addition to vehicles as reference objects, other objects are also used to estimate flood depth such as staff gauges [31], flooded traffic sign images [32], human objects [33–35] and buildings [36]. Zhang and Tong [31] used CV methods to estimate water levels by taking the staff gauge as reference object. The authors also took into account complex site conditions such as, lighting, appearance of staff gauge, weather effects, night scenes, and other related image distortions in their model and applied several methods such as mapping method to obtain transformed images and then obtain extended region of interest for CV-model. Song and Tuo [32] used flooded images of traffic signs in their deep-learning framework called FloodMask to estimate flood depths and the approach they deployed could be easily integrated with CCTV cameras and cameras mounted on vehicles. Meng et al. [33] and Vallimeena et al. [34] deployed region-based convolutional neural networks on flood-related images and segmented human objects identifying their key points and using attributes like age, ethnicity, gender to estimate height of person, which in turn was multiplied by ratio to computer flood depth. The approach was beneficial in planning rescue missions in flood events. Liang et al. [35] developed software V-FloodNet that can take input as images/video files and detect flood inundation depths based on the dimensions of the reference object present in the images / videos and using the template matching technique. Their deep learning model was also validated with the field measurements data from lakes, rivers, campus creek, and harbor for the estimation of inundation depths. Zou et al. [36] collected pairs of pre- and post-flood images from google street images, rectified them and extracted building height using CV-based segmentation techniques to calculate flood depth and evaluated the method against real datasets of floods from Japan and USA.

2.4. Miscellaneous flood-related applications of DL

In addition to their application in detecting flood depths, deep learning techniques in the computer vision domain such as YOLO have been employed to track objects in floods, river flows, or debris. Some of these techniques facilitate the analysis of debris without providing information about its depth in water, flow velocity, and flow direction (mainly velocity vectors). Some research focuses more on determining the extent of the flood using semantic segmentation methods [37] rather than flood depths. Gómez et al. [38] deployed several state-of-the-art convolutional neural networks (CNNs) to track floating plastic debris from satellite images and tested the learning approach in areas that were not unknown to the model. Lin et al. [39] used the commonly used ’object detection’ algorithm YOLO to identify floating debris in waterways as an indirect indicator of water quality. In addition to estimating flood

Table 2
Qualitative studies involving deep learning frameworks for flood depth estimation.

Reference	case study and dataset size	DL-technique used	Flood levels	Accuracy/mAP/recall/prediction score
[25]	1177 of flooded streets with pedestrians or vehicles (with 2095 objects) and street photos/surveillance videos from traffic cameras, Images downloaded from Google search engine using keywords such as "urban waterlogging" and "urban floods."	YOLOv4	Car (three levels none, pipe (> 5cm but below exhaust pipe and handle (between exhaust pipe and handle), car roof) and person (none, leg (below waist and < 5 cm), above waist), as benchmarking	mAP = 89.29%
[26]	6294 images related to urban waterlogging collected via web scraping tools such as Google, Baidu, Facebook, and other images and video plat- forms; data augmentation was also applied to increase dataset size.	Convolutional block attention module (CBAM)-improved ResNet50	Grade 0 (no water), A, B and C for submerged pedestrians, bicycles, electric vehicles, and cars.	accuracy = 92.45%
[27]	20,152 images (various datasets ISE-UFDS, UADETRAC dataset etc.)	YOLOv9	three (safe, dangerous, unsafe)	mAP = 69.4–82.2%
[28]	5676 images flooded image dataset	Coordinate Attention into Residual Neural Network (ResNet)	flooded depths classes 0-9	accuracy = 83.14%
[29]	Dataset constructed from Internet images (250 images), several datasets such as Sazara, Deepflood, WebCOOS and evaluation dataset comprising images with 183 pedestrians.	YOLOv5 and water surface segmentation model based on SegFormer	Based on parts of pedestrians submerged, total 15 types of keypoints (knees, thigh, hips, eyes, ankle etc.)	accuracy = 90.71%
[30]	More than 9000 images from social media platforms, relevant search engines, and live river cameras from US Geological Survey (USGS)	Various CNNs architectures such as YOLOv3 (You look only once version 3), Fast R-CNN (Region-based CNN), Mask R-CNN, SSD MobileNet (Single Shot MultiBox Detector MobileNet), and EfficientDet (Efficient Object Detection)	Detection of objects, water levels (Levels 1–11 denoting flood severity from mild to severe) and areas of water surfaces	Variable prediction score for different objects and depends on CNN used (e.g. 99.9% for YOLOv3 for vehicles and 99% for Fast R-CNN with person classification)

levels, 'object detection models' have been used in various applications; for example, identification of human bodies in catastrophic conditions (accuracy of 96%) by [40]; tracking of flood-borne objects that block canals and rivers by [41]; and real-time flash flood detection in South Korea [42]. In addition to cars and people, objects such as a staff gauge [43], submerged stop signs in residential neighborhoods [44], and submerged houses during floods can be used to monitor water levels and enhance response to flood disasters [45]. In addition to cars and objects, other techniques are being used in this domain. For example, Qin and Shen [46] deployed a novel refraction-based waterlogging depth estimation, with standard road markings of images from street cameras as input data. Liu et al. [47] proposed the flood depth measurement technique based on surveillance video images and a self-designed floating ruler intentionally designed to estimate flood depths, where YOLOv5 was used to identify the upper and lower pixel positions of the rulers, achieving an average relative error of approximately 5.25%.

3. Materials and methods

3.1. Case study and data collection

The city of Matera was chosen as a pilot case study to test the application tool. Located in the Basilicata region of south Italy, Matera is a UNESCO World Heritage Site. The city has been hit by flash flooding events during 2013, 2014, 2018, 2019, 2023, and 2024. These events reported several cases where people were trapped in their cars and pedestrians lost balance due to the high speed of water on narrow steep streets of the city. Fig. 1 shows some of the pictures together with the cars that were used to train the YOLO model for the deployment of the flood depth monitoring system.

The image data was collected from several sources. Photos of parked cars were taken, and the previous repository of the case study of Matera Italy, which is the focus of the investigation. Photos of flooded cars were mixed with previous data of flooded objects collected from repositories and other sources on the Internet to train the model and then tested on unknown datasets. Data were also collected through an Internet search using keywords such as 'Matera flood cars', 'Italian flood submerged cars', and 'Matera floods' to obtain a more representative picture of the car typologies in Italy and the background images. Data collection was also carried out in the city of Potenza, South Italy. Furthermore, data were collected from Matera city as background, etc. to train the model, as Italian city centers are different due to their ancient appearance. Approximately 2030 images were labeled with total 15,484 bounding boxes. The flood car data set used in our study was mixed with data gathered by Wan et al. [19], where they annotated 2000 images, with a total of 6300 annotations of vehicle objects. The total number of images we used are 2030, with 6219 annotations with 0 level annotations (water depth 0 cm) for 1979 images, level 1 annotations (water depth estimation between 0 - 35 cm) for 3261 images, Level 2 (water depth estimation between 35 -75 cm) having 2686 annotations, level 3 (water depth estimation between 75 - 105 cm) having 1273, and lastly the most submerged cars with water reaching up to window level i.e. level 4 (water depth > 105 cm) with 1339 annotations respectively as per classification of flood inundation levels by Liu et al. [17].

Labeling levels range from 0 to 5 (non-submerged to fully submerged cases), as shown in Fig. 2. Although in the application level 0 is color coded green, level 1 and 2 are color coded yellow, and levels 3 and 4 are color coded red for ease of interpretation of risk levels for cars. The same criteria as those used by Liu et al. [17] were used for the labeling of the images into five levels: Level 1, without water, up to mid-type depth; Level 2, from mid- to full-tire; Level 3, type up to the bottom of the window; and Level 4, above the window. Level 3–4 in our application represents severe and threatening conditions warranting instant halting of traffic for drivers and are shown by red boxes in the application interface. The images were re-labeled from scratch by the authors according to the submergence criteria defined in the study by Liu et al. [17]. Strict quality control in labeling was followed and the labeling scheme was checked by both authors via inter annotator agreement check, where they checked each other labeling scheme to avoid errors. The first author had manually annotated all the pictures and the second author had checked sample images to check if quality control is maintained. To summarize, we have used the aforementioned dataset of car depths in various submergence conditions, with various backgrounds and cars at

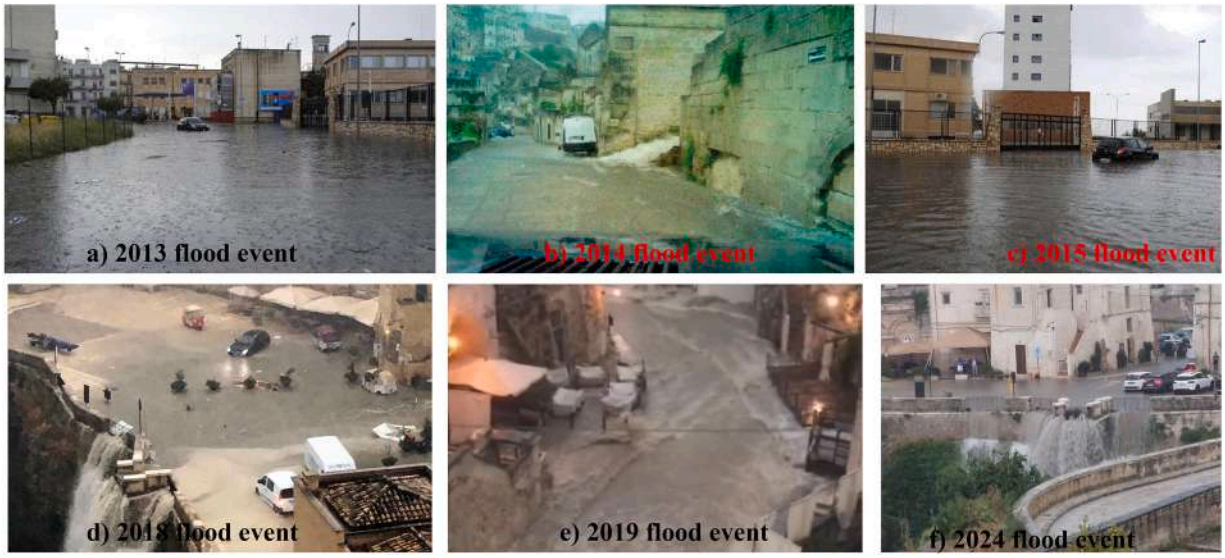

Fig. 1. Case study area in Matera, Basilicata, Italy for flood depth monitoring system based on car depths for various year flood events a) 2013, b) 2014, c) 2015, d) 2018, e) 2019, f) 2024 (sources/weblinks of all photos used are attached with the appendix A).

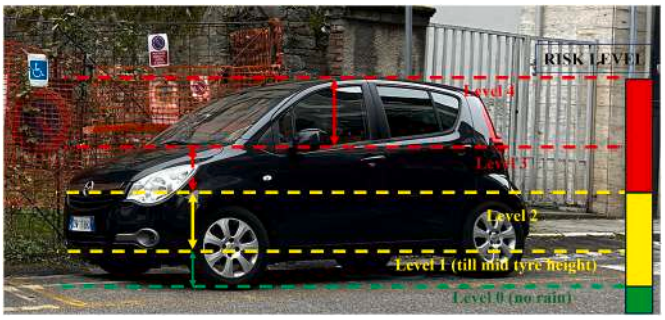

Fig. 2. Labeling criteria and the corresponding risk levels (adapted from Liu et al. [17]).

some distance, either far from or near the viewpoint (Fig. 3). In this regard, more than 15,000 bounding boxes were labeled for various levels of submergence from self-collected data, data current Tech4You project database, internet downloaded photos of floods in Italy, and pictures downloaded from previous papers [17].

However, for blurred images and cars that were far and small for the eye to check the water level, some uncertainty do exist in the annotation accuracy as it was manually done. For the ambiguous/boundary cases, where it was difficult to assign labels, let's say level 2 or 3, then level 3 was assigned to be on conservative side. Furthermore, muddy water and fuzzy backgrounds, cars toppled on each other, and situations where they cannot be decided which level to club them into were left in no class. For including decision rules for borderline cases, if it is visible by eye then it was included, and if the eye cannot distinguish the car image then it was excluded in the labelling scheme.

However, the testing is mainly based on data collected from Matera, previously literature articles and other European cities that were previously affected by flood events. In addition, pictures and videos available online of previous flood events such as the Genova floods were used to test the performance of the model.

3.2. Methods

The main principle of object detection algorithms for flood depth detection is that they provide an estimate of the submergence level of cars in the water based on bounding boxes, which can be correlated with the water level. Object detection algorithms can draw "bounding boxes" on objects, which are interpretable from tables. For example, if they give a bounding box with Level 0, that is, covering the full view of a car, it can be interpreted that the car is at 0 cm depth of water. Similarly, Level 1 implies that a car's half tires are submerged in water, and if we take average values of the half height of tires (over several car models), then flood depth can be estimated with reasonable certainty levels The aim of the present study was to train a machine leaning model, particularly an object detection model, namely YOLO [49], for determining the submergence levels, and accordingly provide an estimate of the depth of water with some uncertainty, which can be attributed to sources, such as the car model, model uncertainties, and false positives. For example, a small car at a depth of 75 cm can be considered to have Level 3 submergence, whereas a big car at a substantial height from the ground is considered to have Level 2 submergence, which is not as serious as for the small car. The current approach detects submergence levels solely based on the depth of water covering car parts such as tires, and does not take into account the sizes of cars in the labeling process. Further classifications, which can also predict car sizes, can help to determine flood depth accurately, but the model predicts the risk of submergence correctly, as it only depends on the water height covering the car.

Although not directly for car depths, standalone DL-based software applications exist with backend YOLO framework in other domains that can automate the processes such as employing YOLO for monitoring of road cracks [50] with RIDER software. The aforementioned RIDER software has several features that are included in current FLOOD-DEPTH-ML tool such as the option to manually choose the input file (image or video), save options of processed files, class information using bounding boxes in display and image itself, etc. Not only YOLO, other DL frameworks such as Mask R-CNN (region-based convolutional neural networks) models are used for software-based applications such as ART-DET software [51] for detection of deterioration in paintings whose feature of display original and processed file we have included in our FLOOD-DEPTH-ML tool.

The current application with backend YOLO algorithm [52] is deployed to estimate flood levels based on the submergence of cars (both image and video feed data) for various flood events in the past in Italy and particularly Matera city, chosen as the case study. The framework followed to develop the application is shown in Fig. 4. The tool offers various functions such as dragging and dropping images and video files (Fig. 6-1) with the analyze button; screen to show analyzed files (Fig. 6-3); a button to stop and pause the video analyzed (Fig. 6-5). Most

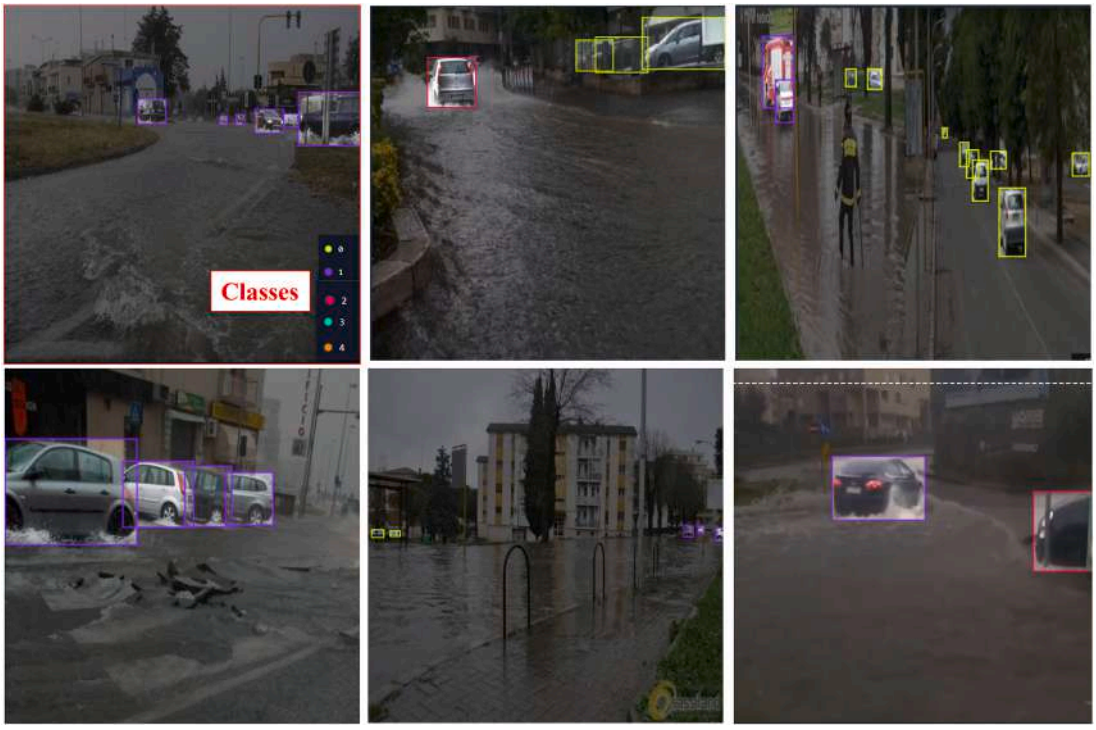

Fig. 3. Some images of the labeling of cars with their label classes coded in various colors in Roboflow [48].

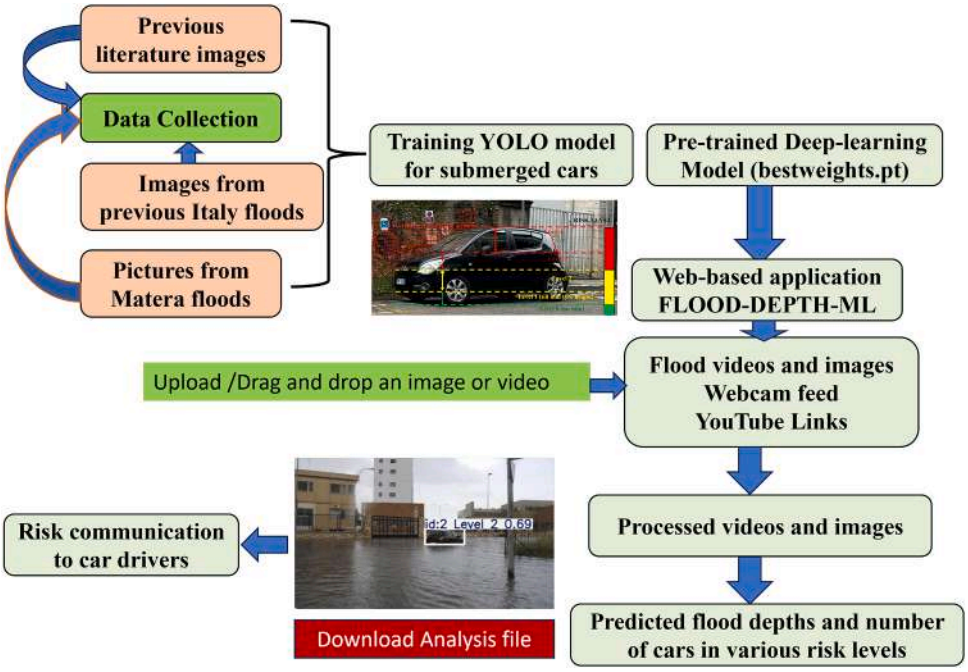

Fig. 4. General framework of the flood depth application (GUI is shown in detail in Fig. 6).

importantly, for the output, the tool offers the option of downloading the processed file (Fig. 6-4) and seeing the total number of cars at various levels in real time (Fig. 6-7). The count of cars at various levels was included as a feature in accordance with a previous study on YOLO-based RescueNet by Prabhu et al. [53], which included the count of flood survivors (both persons and animals) as a feature, with the mAP of 98%.

3.3. FLOOD-DEPTH-ML engine and method

The web-based application was developed in the Python environment, where the labeled images were first used to train the model using the YOLOv8 DL model having a refined CSPDarknet53 architecture illustrated in Fig. 5 consists of three parts, backbone, neck and head, each

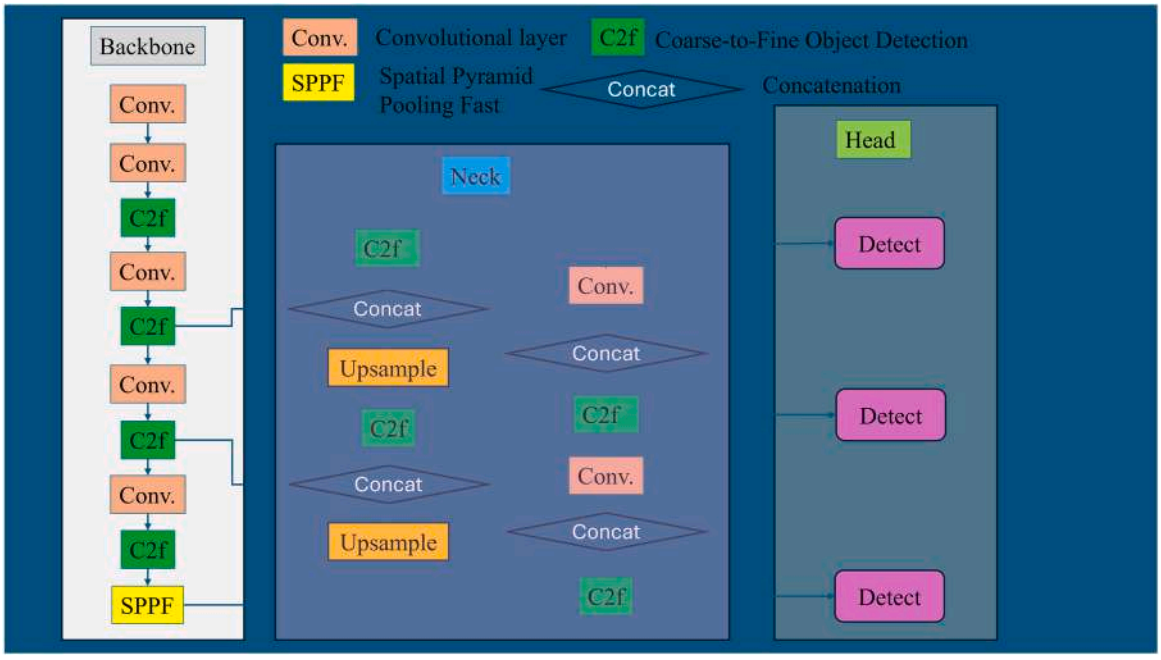

Fig. 5. Architecture of the YOLOv8 used in the application tool.

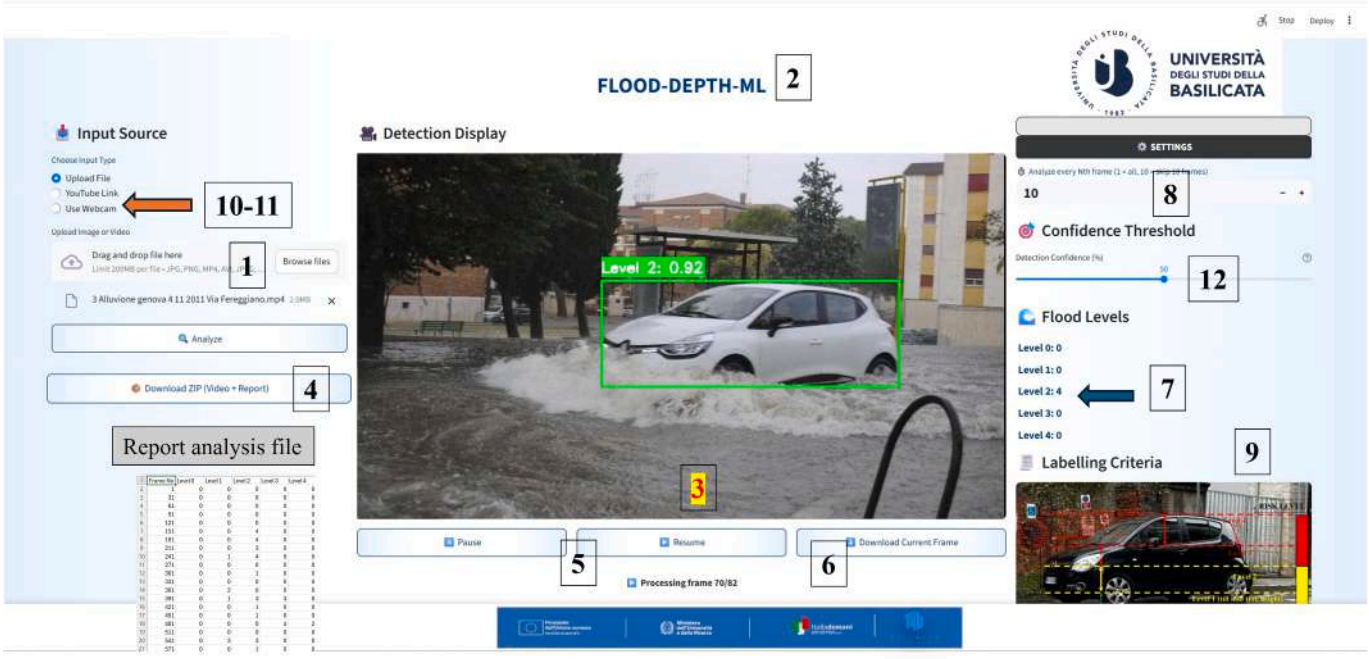

Fig. 6. Graphical user interface of the Python tool: 1) Drag and drop image/video files, 2) Front cover of the app, 3) Analyzed video/picture showing bounding boxes with class probabilities, 4-5-6) Buttons for downloading processed file, analysis button, start/resume video 7) Real-time display of car levels of a particular frame analyzed 8) Selection of FPS (e.g., 5 = every 5th frame analysis and 1 = each frame analyse) 9) Scheme of labelling and the criteria for various depth levels, 10–11) GUI with possibility to use webcam of computer and youtube video link as an input feed making it application for real-time applications 12) slider for selecting confidence threshold).

having their specific functions aiding in the recognition of flooded cars. The backbone part is mainly for extraction of features that can help to detect cars in various depth levels, neck for combining the features, the final identification and classification is done in the head via bounding boxes for levels 0–4 of car submergence depths. Additionally, the head is composed of parallel branches/multilayered pyramid that are able to detect cars in various scales with respect to the input, as some cars are seen as small part of the image while some cars occupy a sizable portion of the input image. For the train validation split strategy, 30% of the data set was then used for validation purposes. For the YOLO part, the hyperparameters used were set to learning rate of default, epochs = 100, input resolution for the images as 640 x 640, basic augmentation methods such as, horizontal flip, grayscale, blur were used. Also, later trails we have also used the yolov11-s small model whose weights also we have shared with the Python code.

The YOLO models trained previously by the authors [54] in Google colab were modified into an application form for ease of use by practitioners. Subsequently, the best weights of the model were stored

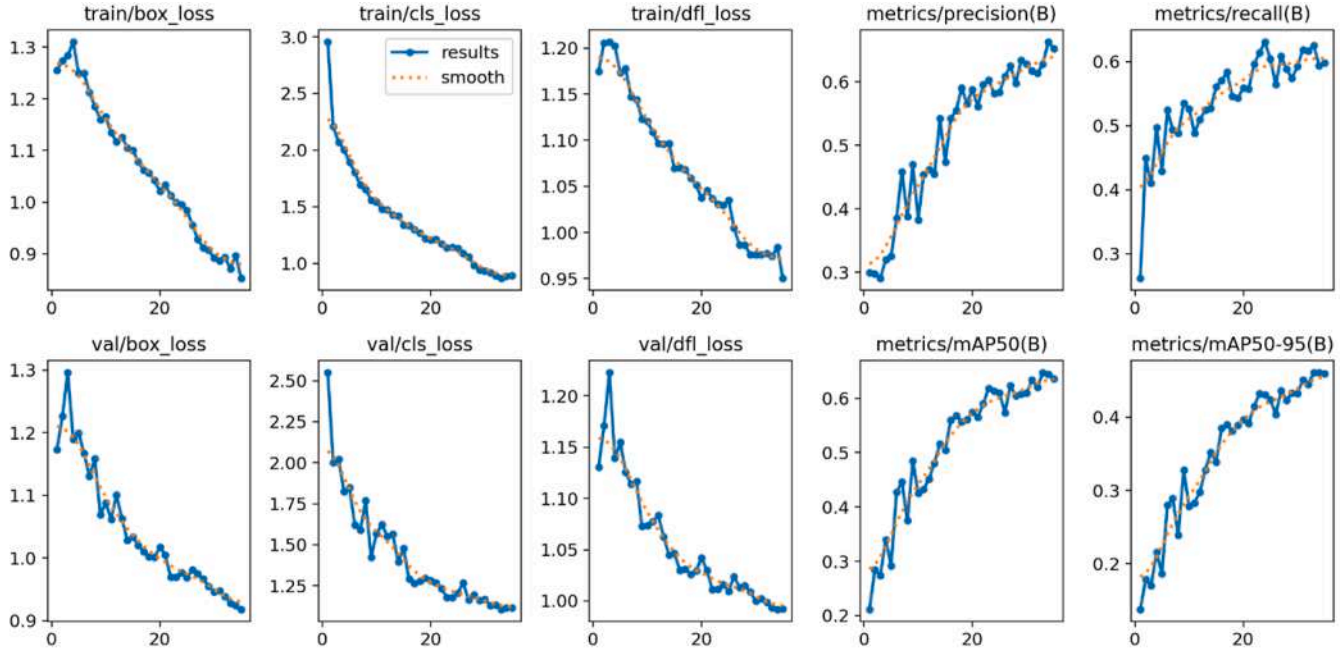

Fig. 7. Loss function for the YOLOv8 model for the car depth data.

and the code was transformed with a GUI interface using the Python Streamlit package that allows GUI-based web applications [55] and an easy-to-use interface with video upload options. Several libraries were used in developing the application such as streamlit for develping GUI, ultralytics for YOLO models, opencv-python for image and video analysis, numpy for computation, yt-dlp for downloading youtube videos for further analysis, pywebview for light GUI applications and Pillow for image processing capabilities. The video was analyzed by dividing into several frames, followed by the detection of the submergence level of the cars and the depth of the water in each frame. A snapshot of the FLOOD-DEPTH-ML tool is shown in Fig. 6. The tool first asks the user to select images and videos from flood events. Then, the YOLO algorithm is applied to determine the depth levels of the car that provide an estimate of the water depths. The probability values and the tracker IDs for counting of car levels are hidden in the GUI display, and the risk levels are color coded in green, yellow, and red for ease of interpretation. The FLOOD-DEPTH-ML TOOL was run on 12th Gen Intel(R) Core(TM) i5-1235U (1.30 GHz) processor with 16.00 GB RAM and there was almost no lag in the computation since the weights were stored after training. The application is customizable and can be modified by other users in terms of the best weights and the GUI according to their preferences, thus increasing the indicating model transferability of the model. FLOOD-DEPTH-ML is currently a fully functional.py file that can run on any machine, once the prerequisite packages in the Python are installed. The application (Figs. 6-2) can perform the following tasks and has the following features and functionalities:

- Process both images and video files of common formats such as.jpg,.png,.mp4, etc. (Fig. 6 -1).
- Display the output file as it is processed on a screen (Fig. 6 -3)
- Generation of the report file of the analyzed file and downloading of processed video after the analysis as an additional step to provide support for data visualization and further analysis (Fig. 6 -4)
- Feature to pause the video and download that particular frame only (Fig. 6 -6). After downloading that pause frame, the user can again start the video from that frame (Fig. 6 -5) and perform the same operation between until the end of the video (Fig. 6 -5).
- Counting the levels of risk of cars in parallel with video processing to estimate the risk over time (Fig. 6 -7).
- Option to select frame rate from 1 to 30, 1 = process each frame in the video, and 30 means process frame at 30 frame interval (Fig. 6 -8). This option is incorporated to decrease processing time as for practical reasons the scene do not change in 1 second, so even processing of 1 FPS/sec is fine for analysis instead of full-fledged analysis of 30 FPS. This also decreased the computational load as instead of changing the YOLO model, changing the FPS computations retains the accuracy of the model.
- Labelling criteria for guidance (Fig. 6 -9) is also shown in same GUI for better interpretation of results.
- The input feed is also modified by giving the user choices, such as including the camera of the laptop in the feed and the link for youtube video files (Fig. 6 -10, 11). The webcam analysis can be saved as a video file making the application step further with previous developed applications in this field.
- The slider to choose the threshold is introduced instead of having a default value of 50% (Fig. 6 -12). This makes the application customizable and with low confidence value the detections will increase, with a tradeoff in the accuracy.
- Inclusion of sound alerts if cars are in levels 3–4 in the analyzed frame. Now in the current version, a 'beep alarm sounds' is played when cars at level 3–4 are detected to better warn users about the risk of submergence.

The performance metrics used illustrated the loss function in Fig. 7. The formulation of the loss function is shown in Eq. (1), which is for YOLO,v1, but the basic concepts are the same for other versions including YOLOv8 [56,57].

$$\begin{aligned}\text{Loss Function} = {} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{\omega_i} - \sqrt{\hat{\omega}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2. \end{aligned} \tag{1}$$

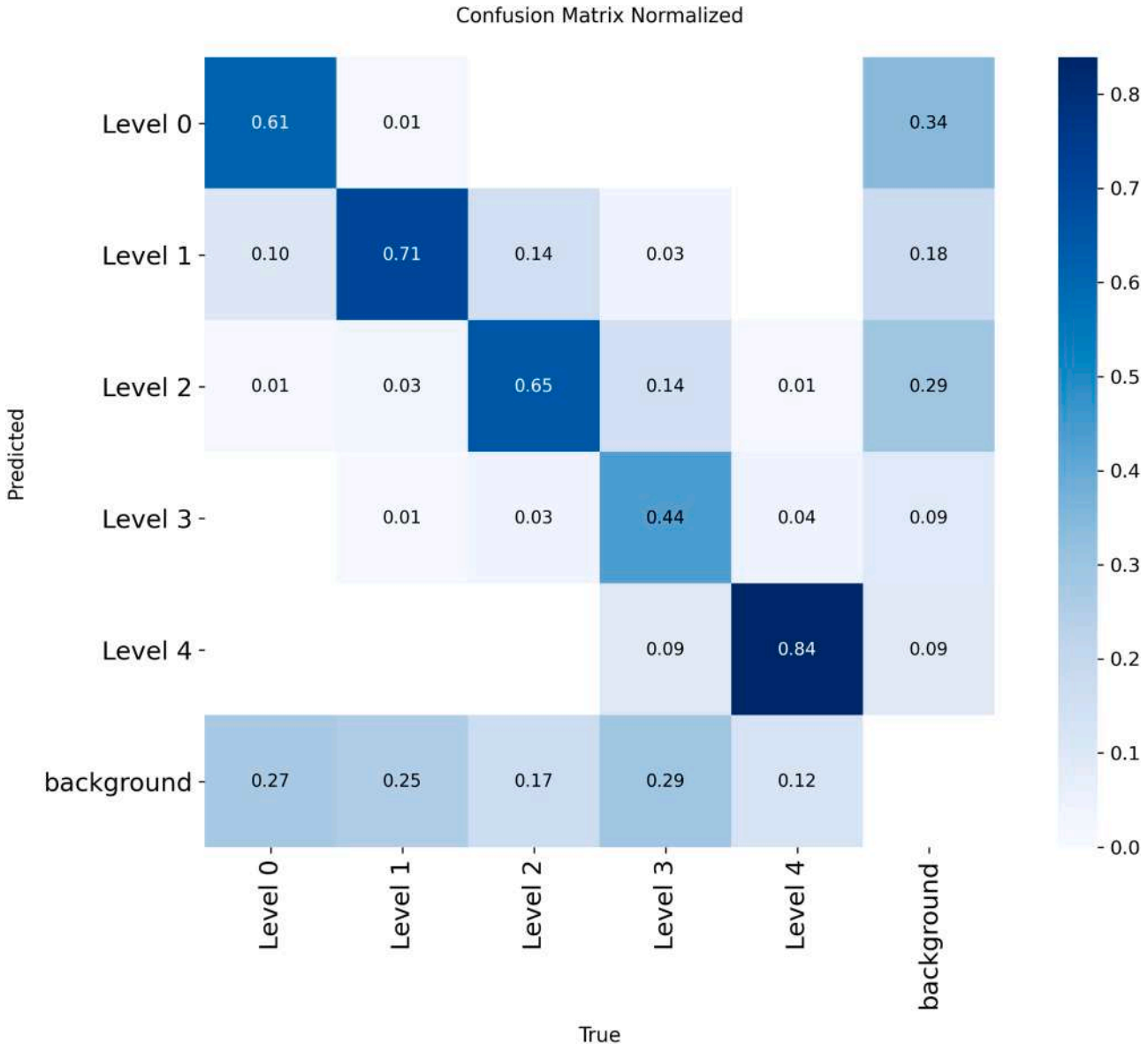

Fig. 8. Confusion matrix normalized for the YOLOv8 model for the car depth data.

Mainly three components of loss functions are there (Fig. 7), box loss for bounding boxes that is the difference between the predicted bounding models and the actual boxes of objects for identifying the car depth levels for coordinates (x_i, y_i, $\hat{x}_i$, $\hat{y}_i$) and their heights and widths of bounding boxes predicted and ground truth (ω_i, h_i, $\hat{\omega}_i$, $\hat{h}_i$), classification loss for object classes (C_i, $\hat{C}_i$) that make sure it's identifying car depth level objects correctly and the third one being objectness loss for detecting the presence of objects referring gauging how confident it is about detecting an object in the first place with probabilities ($\hat{p}_i(c)$, $p_i(c)$). The loss function decreases rapidly in the first 20 epochs and then gradually in the next 20 epochs.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$mAP = \frac{1}{n} \sum_{k=1 \in N} AP_k \tag{4}$$

where TP, FP, TN, and FN denote true positive, false positive, true negative, and false negative for the detection of submerged cars, respectively.

4. FLOOD-DEPTH-ML results

The closest comparison is with [19] who deployed YOLOv8l model and achieved the mAP50 of approximately 66.4%-70.0%, slightly lower than 68.40–70.8% obtained in this study with YOLOv8n. The dataset we used was mixed with dataset which [19] and we have added more photos of Matera case study and other images from floods across world and carried out our own independent annotations Table 3 reports the per class precision, recall and average precision values for the five levels with mean precision of 0.6994, mean recall of 0.6414 and mAP50 of 0.7082 respectively. The confusion matrix normalized is reported in Fig. 8 across all five levels with level 1, 4 performing better than other levels. Liu et al. [17] used the BEW-YOLOv8 model, obtaining the mAP50 of 83.1%, precision of 82.5%, and recall value of 72.9%. In addition, researchers [26] have achieved an accuracy of about 90% with three classes. For example, Wu et al. [26] used about 6294 images with a much greater number of bounding boxes compared with our dataset

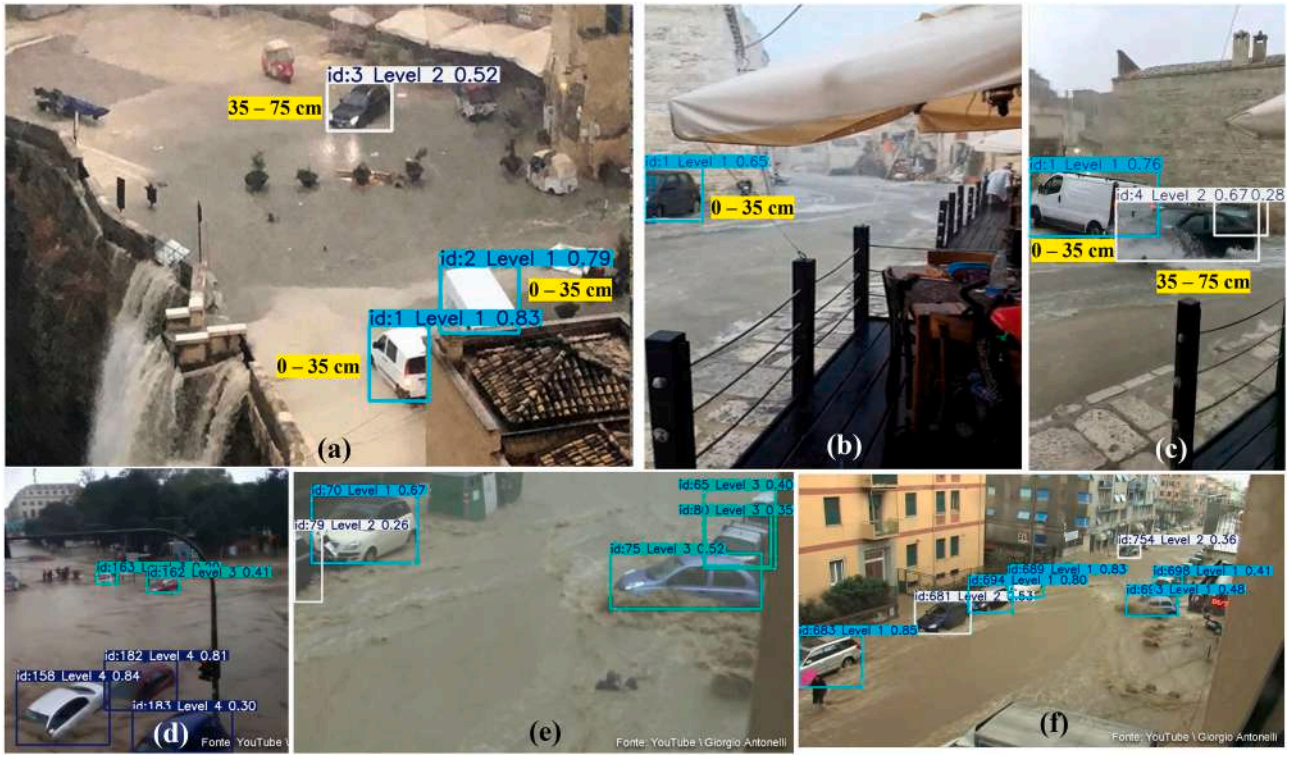

Fig. 9. Flood depth detection results on YOLOv8n trained algorithm. a-c) Sample results on the Matera dataset with boxes showing confidence values for prediction (Yellow highlight indicate range of depths and boxes indicate confidence scores); d-f) Results from other videos of Genova floods obtained from the internet.

Fig. 10. a) Demonstration of object measurement with method of Milanesi et al. [58], b) FLOOD-DEPTH-ML results in estimation of Level 2 submergence which is 35–75 cm of depth level.

of approximately 2100 images. The studies by Wan et al. [18,19], Hao et al. [16] and Liu et al. [17] did not offer an easy-to-use application that could be used by practitioners on site and without the prerequisite to learn Python and offers a variety of features in the application such as the ability to change FPS processing, pausing analysis files, downloading processed videos, log file of analysis carried out etc. while they [17,19] mainly dealt with images. Furthermore, some studies such as [16], were limited to particular car models such as the Sedan, making them difficult to adapt for other car models. In addition, the FloodDepth-GPT application for estimating flood depth [20] was mainly limited to images and to describing the depth calculation rather than providing video feed calculations and color-coded risks for faster communication to the public.

4.1. FLOOD-DEPTH-ML validation with Matera case study

Fig. 9a–c reports some of the images of Matera floods and Fig. 9d–f are snapshots from the video feeds of Genoa 2011 floods (downloaded from youtube video of channel Mi LEGO al Territorio) tested with the developed application. It can be seen that the model is reasonably ac-

Fig. 11. Green tick symbols signify correct identification by application and human estimates. Red crosses indicate incorrect identification while arrows indicate missed identifications. location of flood events: a) Uttarkashi, India (August 2025), b) Uttarkashi, India (August 2025), c) Uttarkashi, India (August 2025), d) Haridwar, India (June2024), e) Haridwar, India (June2024), f) Texas, USA (March 2025), g) Spain (October 2024), h) Spain (October 2024) (The sources of images are provided in the supplementary material attached with this article (Appendix A)).

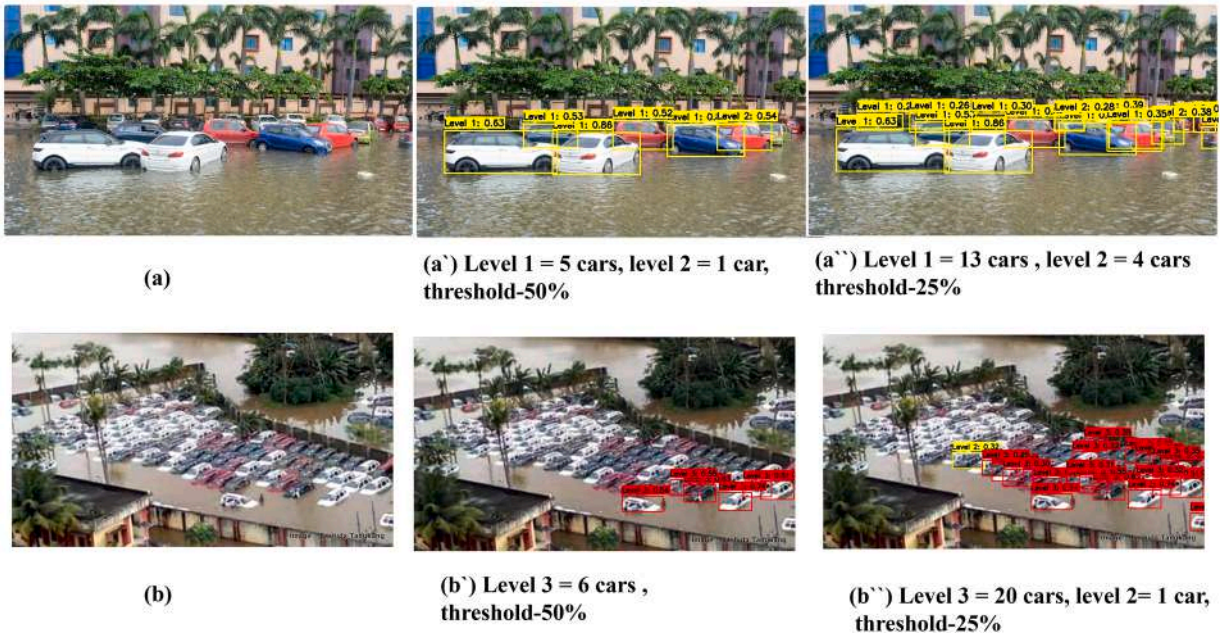

Fig. 12. Performance of the application tool in crowded environments. Left side is original picture and ‘,“ are the processed file detection threshold levels of 50% and 25% (number in boxes indicate confidence scores).

curate and can identify the cars at submergence risk by bounding boxes for various submergence levels.

4.2. Validation of application with recent flood events

Apart from the Matera case study, we have used several images from European floods and videos with a particular focus on Matera. The FLOOD-DEPTH-ML application was tested for recent images and videos available on the internet from various sources such as news channels capturing flood events, and then the images are then classified into three color-coded boxes, green (no risk), yellow (intermediate), and red color boxes meaning risk of submergence for identified cars in the videos/images. The web-based application can thus be deployed and actual values of levels identified from application and human estimates were estab-

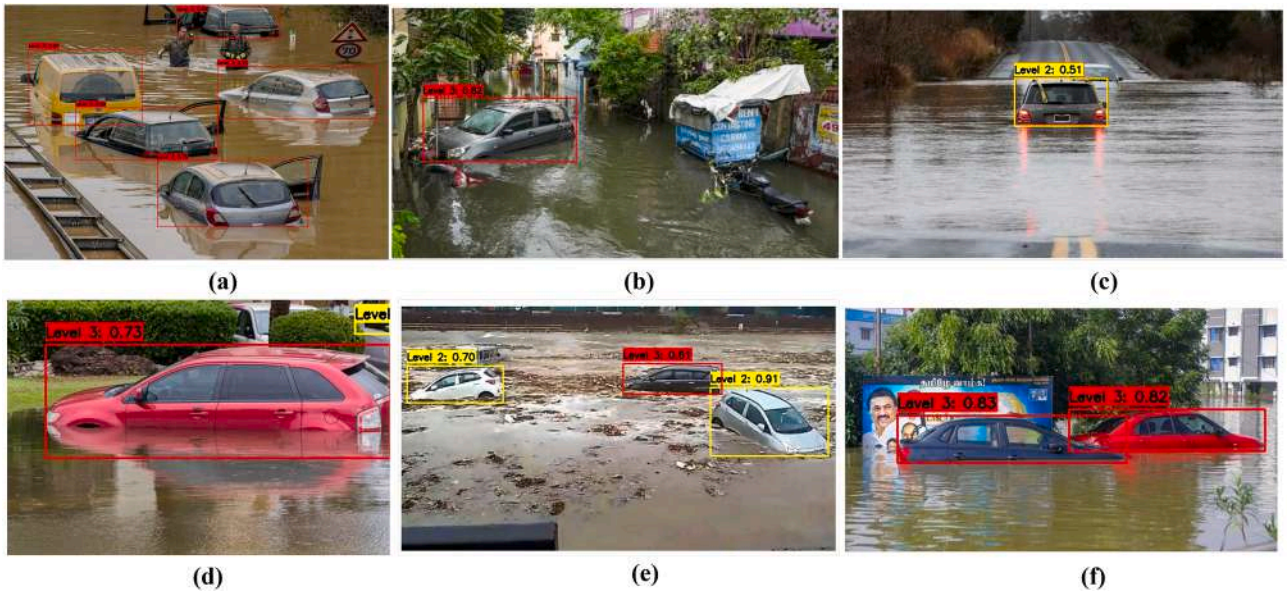

Fig. 13. Performance of the application tool in less-crowded environments with 1–5 cars in a picture (number in boxes indicate confidence scores).

Fig. 14. a) Webcam interface for real-time monitoring, b-c) testing of webcam camera with mobile shown pictures of flood events, possible location for installation of camera for monitoring in d) Matera case study street of Bruno Buozzi in Matera, e) at roof of the University of Basilicata, Potenza, Italy.

Table 3
Evaluation results for YOLOv8 model across five car levels.

Class/Level	Precision	Recall	AP50
Level 0	0.611	0.591	0.636
Level 1	0.669	0.662	0.687
Level 2	0.724	0.675	0.755
Level 3	0.718	0.464	0.594
Level 4	0.772	0.815	0.869

lished and checked for the accuracy. We have tried to include as much variety in our pictures as possible, for example, different angles (some front, some side view, some top view, some oblique view), different background (clear water, muddy water), and also varied number of cars in the images to check the efficiency of the deep learning model in identifying the submerged cars.

The water depth estimated by current FLOOD-DEPTH-ML application was also compared with the previous literature method of Milanesi et al. [58] for estimating height of objects with reference to height of the persons. The flood water depth was estimated from videos or images by scaling the reference object or human body parts with known dimensions, as following the web-based approach proposed by Milanesi et al. [58]. This method proved particularly useful in estimations in the complex urban environments where the monitoring systems were not available, enabling the rapid reconstruction of flood severity from real-based events. The reliability of this method has been validated by comparing the computed flood depths from images with known reference scaling lengths, exhibiting that the estimations fall within calculated range. The real length identification of object utilizing the body segment to measure the water depth for citizen-recorded 24 august 2018 flood event in street of Bruno Buozzi in Matera. In this case, the forearm length (blue line, Fig. 10 a) was, on average 16% of the total height of the person, and the screen length of the unsubmerged part of the body (yellow line, Fig. 10 a) was obtained. The water depth in Fig. 10b was utilized in estimating the flood water depth using FLOOD-DEPTH-ML as between 35–75 cm and is reported by as 37 cm by following web-based approach proposed by Milanesi et al. [58] in the article by Dal Sasso et al. [59]. Although the FLOOD-DEPTH-ML gives a range of 35–75 cm, but by eye estimated it can be ascertained that the depth is in the starting range of 35 cm, making it close to the estimation by the web-based approach proposed by Milanesi et al. [58], and some uncertainty exists due to car models etc. For calculation details and methods, the papers of Milanesi et al. [58] and Dal Sasso et al. [59] can be referred.

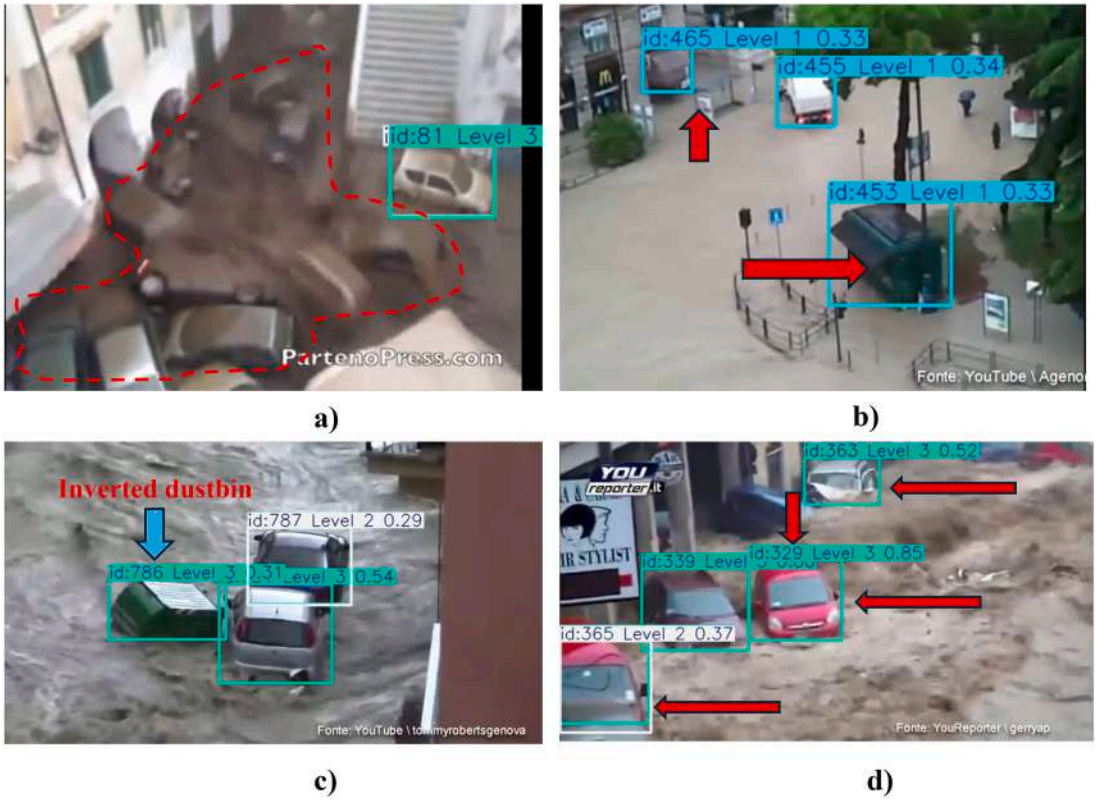

Fig. 15. a) Limitations in detection in crowded environments and difficult backgrounds. b) Detection results on video feeds with false positives for the sheds of restaurants, and the sheds for waiting for bus marked with red arrows, c) inverted dustbin detected as car, d) cars swept away by water detected as Level 3 (videos downloaded from youtube channel (Mi LEGO al Territorio), link shared in the appendix A).(For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Additionally effects of shallow flood mixed multi-scale vehicles scenes, multi-scale mixed partly visible vehicles scenes and night blur scenarios of previous papers [18] are attached as separate Appendix B (supplementary material). Figs. , 7, 8 and 9 in the previous paper of [18] are compared with FLOOD-DEPTH-ML results. FLOOD-DEPTH-ML was able to recognize all cars in Level 1 for the scene in Fig. 7 in the appendix B, two cars in levels 1,2,3 for Fig. 8 and for the night blurred images the two cars were correctly identified with confidence scores of 0.94 and 0.85 respectively for Figure 9 (See supplementary material attached with this article).

Fig. 11 shows the application results tested in pictures of various flood events around the world. The flood events are usually of the recent past, and the captions have described the picture of the events with its date and location in the world. They describe difficult conditions for the deep learning algorithm to detect the correct risk level for submerged cars, for example, in Fig. 11 b the background is muddy, and the background is a cluttered scene with many cars crowded in one place, often on top of each other due to the aftermath of the flood event. It should be noted that in most cases the application results match closely with the estimated human expert (Fig. 11a,b,d,e,f,g), while in the cases where the predictions are incorrect, it is also due to partial visibility of cars and due to cluttered environments. For example, Fig. 11b although the results is correct and the submergence of the car is level 2, but the car in fact is being swept away by water and is not stationary; hence we also need to take the velocity of the water into account for correct risk. Also, in Fig. 11b the threshold value was changed to 25% to be able to detect the car. For Fig 11c, the car on the right side of the image is not visible on the lower part, so it is not possible to identify the risk of submergence, as it could be parked in a dry space or could be submerged and can only be seen if we have the view in correct angle of view and from the view taken it is not possible to judge the level of risk of the car. The rest subfigures are good matches to determine the risk level for the cars as seen. Fig 11a has only one car and is correctly identified as Level 1, Fig 11g all three cars are correctly identified as Level 3 by red color-coded bounding boxes. Fig 11f the cars are level 2 as the water is near the full height of the tire and is easily identified by our web-based application. Based on the levels that the depth criteria highlighted by [17], the depth of the flood water in these locations can be easily estimated. The color-coded boxes are easy to identify as [27], used safe, danger, and unsafe labeling which was also in three classes. We also have further clubbed level 0 as green color (i.e. safe), level 1–2 (i.e. unsafe), and level 3–4 as danger.

Fig. 12 shows the performance of the FLOOD-DEPTH-ML tool for two sample pictures. Fig. 12a shows the cars at level 1 for almost all the cars and in 12b the cars are in levels 2–3. As seen in Fig. 12 a',” many cars are missed in detection due to crowded environments, and also the quality of the pixels is not good, which makes detection difficult. The detection threshold is modified from default 50% to 25% to make the detection of cars more efficient for crowded environments where tiny/small objects can be missed. The results of detection can be seen in Fig 12 a ',” the number of cars detected increase to 17 from 6 and in Fig. 12 b ',” the increase is 21 to previously detected 6 cars. This confidence threshold slider gives users more control of the tool as depending on the site, they can tune the tool for their needs. Furthermore, there are several researches Arias-Martínez et al. [60], Kalfas et al. [61] that recommend using a threshold of 25%. However, the best weight file of the FLOOD-DEPTH-ML application can further be modified, and the efficiency of the algorithm will increase and these images can be used as a benchmark to further improve the application tool.

However, the application tool performs well in less-crowded environments where less than one-five cars are present in the image frame, as can be seen in Fig. 13. In all the sub-figures in Fig. 13a-f, the FLOOD-

DEPTH-ML is able to recognize the submergence risk levels easily with good accuracy as seen in boxes highlighting confidence score for the detection, and hence we can conclude that its application in real environments can be satisfactory. However, the one car in Fig. 13c,e is missing either because it is in low resolution and is far from the view. Additionally, the threshold can be adjusted based on the scene and the FLOOD-DEPTH-ML is able to detect cars with low confidence scores.

Fig. 14a shows the webcam interface that was activated while testing it for photos that were shown to the model with mobile. As can be seen in Figs. 14b-c the model is able to detect the cars with good accuracy in urban flood environment. Furthermore, since the application for the current research work and testing of the application site has been identified in the Matera case study, the site was the most susceptible to flooding, also due to its steep slope (Fig. 14 d) for the installation of the monitoring camera and a test camera on the university roof (Fig. 14 e).

4.3. Limitations

It is difficult to identify cars in "crowded environments," and false positives also come into play due to similarity with car shapes (such as the shades used in restaurants are detected as Levels 3 and 4 car flooding as shown in Fig. 15a-b and inverted dustbins detected as level 3 of car submergence (Fig. 15c). Similar limitations were also pointed out by Reddy et al. [62] where the authors used YOLO and streamlit-based application to detect theft activity in highly cluttered spaces. However, there is need to incorporate modifications that could detect if the car is moving on its own or is being swept away by water as both cases it has a velocity, the one shown in Fig. 15d is being moved away by water and is more risky than level 3 as the car has no control of its own.

5. Discussion

There is uncertainty as we did not divide cars into small, medium, and large classes, as for each car size, the tyre size is different and so are the levels of submergence. According to [17] criteria, we used five levels, namely Level 0 (0), Level 1 (0–35), Level 2 (35–75), Level 3 (75–105), and Level 4 (>105 cm), and considered the mean value of the depths for future analysis of the levels of car instability depending on the level of labeling details. For example, 10 levels of labeling can decrease the uncertainty and provide a better estimate of car depths, and simultaneously including details such as car model and levels of depths can diversify the classification and increase the estimate levels. For example, a small car that has Level 3 submergence is a more serious threat than a big car at a substantial height from the ground level and having bigger tires, for which Level 3 submergence is not as serious as for the small car. Although sufficient data are needed to train these models to give better estimates and reduce the time needed for labeling, we obtain a balanced number of labels in each category to prevent biases.

From the results, we can infer that the model can detect the submergence levels of cars that are even partially visible, which is paramount because in many cases, their view is blocked by cars, poles, and other obstacles, and only partial images are available to make inferences regarding the flood levels. The point of view differs as cars turn, and in many cases we have to infer from a portion of the image only, as an ideal full view of a car is not always available. Modeling is helpful, as it automatically detects risk levels and eliminates the need for a person to stand on the side watching cars and predict the level of water. DL models automate the prediction.

However, these models have some limitations, as the stability of cars is detected solely on the basis of only the depths of water, and so the velocity is not considered. For the Fig. 9c, it can be noted although it is showing Level 1–2 but due to steep and slippery streets of Matera the risk of toppling of pedestrians is more and in future calculations velocity of water should also be taken into account. For example, some models can predict risk level 3, which is correct; however, cars tend to flow with water due to the high flow velocity of water.

The FLOOD-DEPTH-ML also counts the number of cars in various levels of submergence as did [53] counted number of survivors in flood devastated regions in their deep-learning-based application tool named RescueNet), making it one step further to previous models, and option for choosing either laptop camera as source input or local files from the laptop presents the users with the ability to perform tasks such as downloading analysis reports and changing the frame processing rate, making it state-of-the-art tool that can be easily used by practitioners at site. For example, [50] used the You look only once (YOLO) framework for the RIDER software they developed, and the GUI had several features which are included in our application, such as 1). Option of choosing the trained model (best-weights). While our has just one model which was trained previously based on collected dataset, but the users can use their trained weights, 2) changing the threshold for performance parameters (intersection over union i.e. IOU) and confidence scores (CF), which is included in FLOOD-DEPTH-ML via a slider that can be changed to desirable threshold. The feature of saving the processed file and labels is in both applications, while our application has the additional feature of downloading the report, saving analyzed webcam feed and also counting the various levels with each passing frame. However, [50] single-threading approach for up-to 10 FPS and multithreading for stabilizing frames at 30 FPS for boosting efficiency of the software is not used; instead our approach of choosing skipped frames is much simpler and lets the control to the user and practical for urban flood environments.

Furthermore, the ability to reduce the processing of frame rates in the application will make it possible to run it even on local computers without any additional resources. Choosing N = 2, is paramount to choosing ½ the frames which will decreases the computation time into almost ½, but N = 5 –10 is a good number for applications as it will choose frames at 5 no. interval, reducing computing time by almost 1/5 – 10th making it computationally efficient, as in general 1 second of video has 30 Frames to process and the urban flood environment do not change drastically and even 1 frame/second of processing can also work fine.

6. Conclusions and future works

The FLOOD-DEPTH-ML tool provides reasonably accurate flood depth estimates, comparable to the existing literature, and offers real-time processing capabilities with its webcam feature providing option to save webcam analysis feed that can be used directly at the site. The feature of adding the webcam feed and the feed from you tube, which will enhance the capability of the software to be able to use directly at site. The software can be easily integrated with surveillance cameras providing support to the flood early warning systems, such as those in Matera or any flood-related application that involves monitoring submerged cars, allowing continuous monitoring of vehicle safety during floods, and the current version runs smoothly on a laptop. The system aims to improve the estimation of the depth of the flood using existing objects in the environment, facilitating the identification of safe zones in real time. In addition, early warning system plans to includes flood depth forecasting capabilities using FLOOD-DEPTH-ML and a separate mobile app FLOWS developed for the city of Matera that can support public preparedness and enable rescue strategies, helping to mitigate property damage and loss of life during flood events. The anticipated impact is to support flood monitoring and to aid both preventive measures and emergency response focused mainly on car traffic.

Future works can be to implement the software into the installed cameras for the city of Matera to warn about higher levels of water in the streets and further enhance the capability of the application by incorporating satellite data, provision to include user feedback, inputting velocity of flow in the application, provision to include online video feed via shareable link to further enhance its integration with live cameras, and real-time weather data. Furthermore, modifications can be made

where a camera that is on the smartphone or drone can be connected to the application to improve the functionality of the application. In this way, the smartphone can be wirelessly connected to the computer and the software can be used.

CRediT authorship contribution statement

Mayank Mishra: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization; **Raffaele Albano:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Data availability

This is the link to the code: https://github.com/mayankmi/FLOOD-DEPTH-ML The Python code is provided for testing using the following GitHub link (https://github.com/mayankmi/FLOOD-DEPTH-ML.git) The authors can also provide the Python codes for several versions developed and the trained weights for YOLOv8 and v11 for the replication of results upon reasonable request by email also.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests.

Raffaele Albano reports financial support for this work was provided within the scope of the “Tech4You” Innovation Ecosystem project, Notice no. 3277 of 28.12.2021 - Intervention proposals for the creation and strengthening of “innovation ecosystems” PNRR - MUR project code: ECS0000009 - CUP H23C22000370006Tec4you. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work has been carried out within the scope of the “Tech4You” Innovation Ecosystem project, Notice no. 3277 of 28.12.2021 - Intervention proposals for the creation and strengthening of “innovation ecosystems” PNRR - MUR project code: ECS0000009 - CUP H23C22000370006Tec4you.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.rineng.2026.109495.

References

[1] A. Sole, L. Giosa, R. Albano, A. Cantisani, The laser scan data as a key element in the hydraulic flood modelling in urban areas, Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 40 (2013) 65–70.
[2] I. Castro-Melgar, T. Falaras, E. Basiou, I. Parcharidis, Assessment of the October 2024 cut-off low event floods impact in Valencia (Spain) with satellite and geospatial data, Remote Sens. 17 (13) (2025) 2145.
[3] H. Kreibich, K. Piroth, I. Seifert, H. Maiwald, U. Kunert, J. Schwarz, B. Merz, A.H. Thieken, Is flow velocity a significant parameter in flood damage modelling?, Nat. Hazards Earth Syst. Sci. 9 (5) (2009) 1679–1692.
[4] B. Merz, H. Kreibich, R. Schwarze, A. Thieken, Review article" Assessment of economic flood damage", Nat. Hazards Earth Syst. Sci. 10 (8) (2010) 1697–1724.
[5] R. Albano, A. Sole, D. Mirauda, J. Adamowski, Modelling large floating bodies in urban area flash-floods via a smoothed particle hydrodynamics model, J. Hydrol. 541 (2016) 344–358.
[6] Z. Wu, Y. Huang, K. Huang, K. Yan, H. Chen, A review of non-contact water level measurement based on computer vision and radar technology, Water 15 (18) (2023) 3233.
[7] R. Ljubičić, S.F. Dal Sasso, B. Zindović, SSIMS-Flow: Image velocimetry workbench for open-channel flow rate estimation, Environ. Modell. Softw. 173 (2024) 105938.
[8] K. Vinoth, P. Sasikumar, VINO_EffiFedAV: VINO with efficient federated learning through selective client updates for real-time autonomous vehicle object detection, Res. Eng. 25 (2025) 103700.
[9] J. Zhang, B. Zhu, H. Qiu, Real-time concrete crack segmentation for bridge structural health monitoring: a lightweight YOLOv11-based approach with multi-scale feature fusion, Res. Eng. 28 (2025) 108004.
[10] G.D. Deepak, S.K. Bhat, Maximizing YOLOv2 efficiency: a study on multiclass detection of indoor objects, Res. Eng. 26 (2025) 105405.
[11] G. Santarsiero, Automated assessment of bridge guardrails for regional prioritization based on open-source data and deep learning algorithms, Res. Eng. 26 (2025) 105210.
[12] J. Li, R. Cai, Y. Tan, H. Zhou, A.-M. Sadick, W. Shou, X. Wang, Automatic detection of actual water depth of urban floods from social media images, Measurement 216 (2023) 112891.
[13] M. Asif, M.M. Kuglitsch, I. Pelivan, R. Albano, Review and intercomparison of machine learning applications for short-term flood forecasting, Water Resour. Manage. 39 (5) (2025) 1971–1991.
[14] D. McSpadden, S. Goldenberg, B. Roy, M. Schram, J.L. Goodall, H. Richter, A comparison of machine learning surrogate models of street-scale flooding in Norfolk, Virginia, Mach. Learn. Appl. 15 (2024) 100518.
[15] B. Liu, Y. Li, M. Ma, B. Mao, A comprehensive review of machine learning approaches for flood depth estimation, Int. J. Dis. Risk Sci. (2025) 1–13.
[16] X. Hao, H. Lyu, Z. Wang, S. Fu, C. Zhang, Estimating the spatial-temporal distribution of urban street ponding levels from surveillance videos based on computer vision, Water Resour. Manage. 36 (6) (2022) 1799–1812.
[17] B. Liu, Y. Li, X. Feng, P. Lian, BEW-YOLOv8: a deep learning model for multi-scene and multi-scale flood depth estimation, J. Hydrol. 645 (2024) 132139.
[18] J. Wan, Y. Shen, F. Xue, X. Yan, Y. Qin, T. Yang, G. Yang, Q.J. Wang, DSC-YOLOv8n: an advanced automatic detection algorithm for urban flood levels, J. Hydrol. 643 (2024) 132028.
[19] J. Wan, Y. Qin, Y. Shen, T. Yang, X. Yan, S. Zhang, G. Yang, F. Xue, Q.J. Wang, Automatic detection of urban flood level with YOLOv8 using flooded vehicle dataset, J. Hydrol. 639 (2024) 131625.
[20] T. Akinboyewa, H. Ning, M.N. Lessani, Z. Li, Automated floodwater depth estimation using large multimodal model for rapid flood mapping, Comput. Urban Sci. 4 (1) (2024) 12.
[21] C. Sazara, B. Salahshour, M. Cetin, K. Iftekharuddin, A deep learning method for floodwater depth prediction on roadways from side-view real and synthetic images of vehicles, J. Big Data Analyt. Transp. 4 (1) (2022) 85–101.
[22] N.M. Notarangelo, C. Wirion, F. van Winsen, STURM-FloodDepth: a deep learning pipeline for mapping urban flood depth using street-level and oblique aerial imagery, Geomatica (2025) 100061.
[23] Y. Qiu, X. Zhou, J. Wan, T. Yang, L. Zhang, Y. Zhong, L. Shen, X. Ji, Automated urban flood level detection based on flooded bus dataset using YOLOv8, Nat. Hazards Earth Syst. Sci. 25 (9) (2025) 3525–3544.
[24] L. Lin, Z. Zeng, C. Tang, Y. Xie, Q. Liang, Robust and fast sensing of urban flood depth with social media images using pre-trained large models and simple edge training, Hydrology 12 (11) (2025) 307.
[25] P. Zhong, Y. Liu, H. Zheng, J. Zhao, Detection of urban flood inundation from traffic images using deep learning methods, Water Resour. Manage. 38 (1) (2024) 287–301.
[26] L. Wu, Y. Liu, J. Zhang, B. Zhang, Z. Wang, J. Tong, M. Li, A. Zhang, Identification of flood depth levels in urban waterlogging disaster caused by rainstorm using a CBAM-improved ResNet50, Expert Syst. Appl. 255 (2024) 124382.
[27] J. Sun, C. Xu, C. Zhang, Y. Zheng, P. Wang, H. Liu, Flood scenarios vehicle detection algorithm based on improved YOLOv9, Multimedia Syst. 31 (1) (2025) 74.
[28] W. Du, M. Qian, S. He, L. Xu, X. Zhang, M. Huang, N. Chen, An improved ResNet method for urban flooding water depth estimation from social media images, Measurement 242 (2025) 116114.
[29] J. Jiang, X. Feng, J. Huang, J. Chen, M. Liu, C. Cheng, J. Liu, A. Xue, Identification of pedestrian submerged parts in urban flooding based on images and deep learning, Environ. Modell. Softw. 183 (2025) 106252.
[30] R.J. Pally, S. Samadi, Application of image processing and convolutional neural networks for flood image classification and semantic segmentation, Environ. Modell. Softw. 148 (2022) 105285.
[31] D. Zhang, J. Tong, Robust water level measurement method based on computer vision, J. Hydrol. 620 (2023) 129456.
[32] Z. Song, Y. Tuo, Automated flood depth estimates from online traffic sign images: explorations of a convolutional neural network-based method, Sensors 21 (16) (2021) 5614.
[33] Z. Meng, B. Peng, Q. Huang, Flood depth estimation from web images, in: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities, 2019, pp. 37–40.
[34] P. Vallimeena, B.B. Nair, S.N. Rao, Machine vision based flood depth estimation using crowdsourced images of humans, in: 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), IEEE, 2018, pp. 1–4.
[35] Y. Liang, X. Li, B. Tsai, Q. Chen, N. Jafari, V-FloodNet: a video segmentation system for urban flood detection and quantification, Environ. Modell. Softw. 160 (2023) 105586.
[36] B. Zou, B. Peng, Q. Huang, Flood depth assessment with location-based social network data and google street view-A case study with buildings as reference objects, in: IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium, IEEE, 2022, pp. 1344–1347.
[37] Y. Wang, Y. Shen, B. Salahshour, M. Cetin, K. Iftekharuddin, N. Tahvildari, G. Huang, D.K. Harris, K. Ampofo, J.L. Goodall, Urban flood extent segmentation and evaluation from real-world surveillance camera images using deep convolutional neural network, Environ. Modell. Softw. 173 (2024) 105939.
[38] À.S. Gómez, L. Scandolo, E. Eisemann, A learning approach for river debris detection, Int. J. Appl. Earth Obs. Geoinf. 107 (2022) 102682.

[39] F. Lin, T. Hou, Q. Jin, A. You, Improved YOLO based detection algorithm for floating debris in waterway, Entropy 23 (9) (2021) 1111.
[40] J.R. Thota, A. Padala, Human remains detection in natural disasters using YOLO: a deep learning approach, Eng. Technol. Appl. Sci. Res. 14 (6) (2024) 17678–17682.
[41] U. Iqbal, M.Z.B. Riaz, J. Barthelemy, N. Hutchison, P. Perez, Floodborne objects type recognition using computer vision to mitigate blockage originated floods, Water 14 (17) (2022) 2605.
[42] N.H. Quang, H. Lee, N. Kim, G. Kim, Real-time flash flood detection employing the YOLOv8 model, Earth Sci. Inform. 17 (5) (2024) 4809–4829.
[43] L. Sabbatini, L. Palma, A. Belli, F. Sini, P. Pierleoni, A computer vision system for staff gauge in river flood monitoring, Inventions 6 (4) (2021) 79.
[44] B.A. Kharazi, A.H. Behzadan, Flood depth mapping in street photos with image processing and deep neural networks, Comput. Environ. Urban Syst. 88 (2021) 101628.
[45] J. Teoh, Z.B. Zulkoffli, K.M. Yap, H.S. Chua, Exploring generative AI for YOLO-based object detection to enhance flood disaster response in Malaysia, IEEE Access 12 (2024) 173686–173699.
[46] J. Qin, P. Shen, Refraction-based waterlogging depth measurement using solely traffic cameras for transparent flood monitoring, J. Hydrol. 655 (2025) 132917.
[47] S. Liu, W. Zheng, X. Wang, H. Xiong, J. Cheng, C. Yong, W. Zhang, X. Zou, A novel depth measurement method for urban flooding based on surveillance video images and a floating ruler, Natural Hazards 119 (3) (2023) 1967–1989.
[48] B. Dwyer, J. Nelson, J. Solawetz, (2022). Roboflow. Roboflow (Version 1.0)[Software].
[49] T. Diwan, G. Anirudh, J.V. Tembhurne, Object detection using YOLO: challenges, architectural successors, datasets and applications, Multimed. Tools Appl. 82 (6) (2023) 9243–9275.
[50] R. Fu, Y. Zhang, K. Zhu, A. Strauss, M. Cao, Real-time detection of concrete cracks via enhanced you only look once network: algorithm and software, Adv. Eng. Softw. 195 (2024) 103691.
[51] F.M. Garcia-Moreno, J.C. Alcaraz, L.R. Rodríguez-Simón, M.V. Hurtado-Torres, et al., ARTDET: Machine learning software for automated detection of art deterioration in easel paintings, SoftwareX 28 (2024) 101917.
[52] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[53] B.V.B. Prabhu, R. Lakshmi, R. Ankitha, M.S. Prateeksha, N.C. Priya, RescueNet: YOLO-based object detection model for detection and counting of flood survivors, Model. Earth Syst. Environ. 8 (4) (2022) 4509–4516.
[54] M. Mishra, R. Albano, A. Sole, Deep learning-assisted flood depth measurement using flooded car images and video feeds from urban flood events, Giornate dell'Idrologia 2025, Politecnico Di Bari, 8–10 Settembre 2025 (2025).
[55] M. Khorasani, M. Abdou, J.H. Fernández, Web application development with streamlit, Softw. Develop. (2022) 498–507.
[56] J. Terven, D.-M. Córdova-Esparza, J.-A. Romero-González, A comprehensive review of yolo architectures in computer vision: from yolov1 to yolov8 and yolo-nas, Mach. Learn. Knowl. Extract. 5 (4) (2023) 1680–1716.
[57] M. Mishra, T. Barman, G.V. Ramana, Artificial intelligence-based visual inspection system for structural health monitoring of cultural heritage, J. Civil Struct. Health Monit. 14 (1) (2024) 103–120.
[58] L. Milanesi, M. Pilotti, B. Bacchi, Using web-based observations to identify thresholds of a person's stability in a flow, Water Resour. Res. 52 (10) (2016) 7793–7805.
[59] S.F. Dal Sasso, G. Andrulli, V. Cambio, L. Mita, M. Asif, V. Totaro, R. Albano, Urban flooding analysis for event reconstruction in historical cities: the case study of matera (Italy) (manuscript under revisions) " Front. in Build Env. (2026) .
[60] L. Arias-Martínez, F. Jáñez-Martino, E. Fidalgo, P. Rodríguez-González, A.I. Fernández-Abia, E. Alegre, J. Barreiro, Automated shrinkage and gas porosity detection using YOLO model in additive-manufactured aluminum alloy parts, Integr. Mater. Manuf. Innov. 14 (2025) 320–332.
[61] I. Kalfas, B. De Ketelaere, K. Bunkens, W. Saeys, Towards automatic insect monitoring on witloof chicory fields using sticky plate image analysis, Ecol. Inform. 75 (2023) 102037.
[62] K.U.K. Reddy, F. Shaik, V. Swathi, P. Sreevidhya, A. Yashaswini, J.U. Maheswari, Design and implementation of theft detection using YOLO based object detection methodology and gen AI for enhanced security solutions, in: 2025 International Conference on Inventive Computation Technologies (ICICT), IEEE, 2025, pp. 583–589.