Autonomous Intelligent Systems

ORIGINAL ARTICLE Open Access

Enhanced pothole detection system using YOLOX algorithm

Mohan Prakash B[1*] and Sriharipriya K.C[1]

Abstract

The road is the most commonly used means of transportation and serves as a country's arteries, so it is extremely important to keep the roads in good condition. Potholes that happen to appear in the road must be repaired to keep the road in good condition. Spotting potholes on the road is difficult, especially in a country like India where roads stretch millions of kilometres across the country. Therefore, there is a need to automate the identification of potholes with high speed and real-time precision. YOLOX is an object detection algorithm and our main goal of this article is to train and analyse the YOLOX model for pothole detection. The YOLOX model is trained with a pothole dataset and the results obtained are analysed by calculating the accuracy, recall and size of the model which is then compared to other YOLO algorithms. The experimental results in this article show that the YOLOX-Nano model predicts potholes with higher accuracy compared to other models while having low computational costs. We were able to achieve an Average Precision (AP) value of 85.6% from training the model and the total size of the model is 7.22 MB. The pothole detection capabilities of the newly developed YOLOX algorithm have never been tested before and this paper is one of the first to detect potholes using the YOLOX object detection algorithm. The research conducted in this paper will help reduce costs and increase the speed of pothole identification and will be of great help in road maintenance.

Keywords: YOLO, YOLOX, Object detection, Pothole detection, Machine learning

1 Introduction

The roads serve as an integral part of human society and it is the most common means of transportation. The construction and maintenance of roads dates back to 6000 BC and the presence of roads always indicates the presence of a developed civilization. The roads have changed a lot since then and our dependence on them has greatly increased. Therefore, it becomes very important to maintain the roads properly and smoothly to avoid unnecessary accidents and transportation delays.

One of the most common types of damage that occurs on a road is potholes. They occur on roads due to poor maintenance and a combination of various factors that include environmental factors such as rain, snow, etc. and human factors that causes an increase in road stress and ultimately causes pothole formation. One cannot predict the location of pothole formation, since there are many factors leading to their occurrence. There are approximately 5 million km of roads in India, which greatly complicates the task of identifying potholes and maintaining the roads. Detecting road damage takes a lot of time for the government because the process of road inspection by a professional worker takes a lot of time and resources. A lot of money is spent every year to keep the roads maintained and in working order. Therefore, there is a need to automate the process of detecting potholes with improved speed and at low cost without manual intervention.

A lot of research has been published using different methods to automate the process of detecting potholes in the roads. Some of the commonly used pothole detection techniques are (i) vibration technique by using accelerometers [1, 2] and other sensors [3] that can detect a bump in the road once the vehicle drives through it, (ii) 3D reconstruction technique [4, 5] by using laser scanner method, stereo vision method or kinetic sensor method, (iii) im-

*Correspondence: mohanprakash.b2019@vitstudent.ac.in
1 SENSE, VIT University, Vellore, 632014, Tamil Nadu, India

age processing techniques [6, 7] using GLCM (Gray-Level Co-Occurrence Matrix), RBF (Radial Basis Function) [6] and other morphological methods [8, 9], and (iv) then the computer vision engineering approach using artificial intelligence, machine learning [10] and various deep learning techniques to detect potholes from a camera input in the form of an image or a video.

The vibration technique [1, 2, 11] is employed by using sensors that can detect sudden impacts or movements. They are installed in a moving vehicle and as soon as the vehicle drives through a pothole, the sensor detects the pothole and data on the location of the pothole is collected. This method is not efficient because we cannot predict the potholes in advance and the vehicle needs to be able to drive on the pothole, which also makes the system more error-prone. Also, they cannot predict the difference between potholes and other road artifacts, including speed bumps, reflectors, etc.

The second method is based on 3D reconstruction techniques [4, 5]. These techniques include stereo vision based methods and laser based scanning methods. Using these methods, a 3D image of the road is created and the position and depth of the pothole can be analysed in advance, making this a valid candidate for a real-time application. However, the main disadvantage of this technique is that these techniques are expensive to implement and the system itself is very sensitive. The cameras need to be calibrated properly and they are sensitive to vibration, so a stabilizer is required to be attached to the camera. The 3D reconstruction of the road requires a lot of computational power [12], which makes the cost of the system costly as powerful processors are required to be set up. Multiple cameras are required to analyse the depth and width of the pothole, and the reconstruction range is very short. These disadvantages make the implementation of pothole detection using 3D reconstruction techniques unyielding.

The image processing technique uses various statistical methods like GLCM, RBF, etc. to obtain and differentiate different textures and features of an image. These techniques are often used alongside other machine learning approaches because of the massive computational power required. GLCM is an image pre-processing technique that can examine and separate textures that take into account the spatial relationships of pixels. RBF or radial basis function is a real-valued function whose main task is to approximate multivariable functions by linear combination. It is an ANN (Artificial Neural Network) [13] that can transform input signals into other forms. Morphological image processing [8] is a variety of image processing techniques that focus on the shape of features in a gray-scale image. Each pixel indicates the contrast intensity of the surrounding neighbouring pixels.

Another technique that is gaining popularity these days is the vision-based techniques. They use a single camera [14] for collecting data in the form of videos or images and are somewhat immune to shakes. Computer vision [15] is an area of artificial intelligence that can derive meaningful information and data from images or videos. Typically, a camera is placed in front of the vehicle facing the road and the video that is being recorded is analysed and potholes are detected by the AI. There are a number of different deep learning algorithms [16, 17] that can be used for object detection. Algorithms like CNN (Convolution Neural Network) [18, 19], RNN (Recurrent Neural Network) [20], faster R-CNN [21] are some of the most commonly used systems for training and processing. One such object detection algorithm is the YOLO (You Only Look Once) algorithm.

YOLO is an acronym for the term You Only Look Once. It's a one-shot detector and the input image is only sampled once, hence the name. It is one of the most widely used object detection algorithms and has received a lot of attention since its release. It is extremely fast and is able to outperform other modern object detectors. It has a wide range of applications such as autonomous vehicles, video analytics, object identification, etc. including pothole detection. Various versions of the YOLO algorithm have been released over the years, all building on their predecessors and being vastly improved compared to their older versions. YOLOX is the latest released object detection model, which is an improvement on the previous YOLO algorithms.

Various methods have been used to tune the algorithm to improve both its speed and accuracy. Although YOLO algorithms are one of the commonly used object detection models for pothole detection, YOLOX is a newly introduced model whose pothole detection capabilities have yet to be justified. Therefore, this paper aims to train and analyse this newly developed YOLOX algorithm to detect potholes on the road. The next section reviews some of the commonly used YOLO algorithms for pothole detection.

2 Related work

The first version of YOLO, the YOLOv1 [22], was published in 2015 by authors Joseph Redmon and Ali Farhadi. It is considered to be one of the first object detectors to fall into the category of single-shot object detectors. Here, a single branchless convolutional neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. The entire detection pipeline is a single network, which allows the algorithm to directly optimize for end-to-end detection performance. Yolov1 is one of the first object detection frameworks that locates objects without the use of anchor boxes. This anchorless mechanism predicts bounding boxes by strictly considering points that are close to the centre of the objects. This YOLO base model was capable of processing images in real time at 45 frames per second. In terms of speed, it far surpasses other detection methods [23] such as DPM

(Deformable Parts Model) [24] and R-CNN. However, because it only uses centre points to predict bounding boxes, YOLOv1 makes more localization errors and the lack of anchor boxes also results in a significant drop in recall value. Also, since the entire image is captured, it becomes significantly more difficult to predict small or dense objects in the image. This makes it more difficult to spot clusters of potholes, which are small and occur in large numbers.

The second version, YOLOv2 [25], was released in 2017 by improving the training and performance of YOLOv1. It was published by authors Joseph Redmon and Ali Farhadi. Since the lack of anchor boxes caused the Precision drop in YOLOv1, the YOLOv2 model uses the anchor boxes in convolutional layers to achieve good localization accuracy and to predict k-mean clustering. Built into this model are high and low-level feature maps, as well as fine-grained features, which are very useful in detecting small objects. At 67 FPS, the YOLOv2 was able to reach 76.8 mAP on VOC 2007 [26]. That was enough to outperform the other cutting-edge methods like Faster R-CNN, ResNet and SSD [27] while still running significantly faster [28]. Although a relative increase in recall was observed compared to YOLOv1, there was a slight decrease in mAP.

YOLOv3 [29] by authors Joseph Redmon and Ali Farhadi was released in 2018. Compared to v2, the ability to detect small objects, accuracy and real-time ability have been further improved. A lot of works has been done extensively using the YOLOv3 algorithm [30–33] to design an object detection system that can be used to detect potholes from a video feed. Dharneeshkar et al. [30] proposed a YOLOv3-based pothole detection system and the results were obtained and analyzed against YOLOv2 and YOLOv3-tiny. YOLOv3-tiny's mAP@0.25 and mAP@0.5 values were found to be larger compared to its counterparts. A precision of 0.76 was obtained compared to the precision of 0.52 obtained on YOLOv2. However, the main disadvantages of these papers are that YOLOv3 is slow and many advances in object detection methods have been made since then.

YOLOv4 [34] was published in April 2020 by Alexey Bochkovskiy et al. They integrated cross-iteration batch normalization and path aggregation network, making it more suitable for single GPU training. CSPDarknet, a combination of CSPNet and Darknet, was used as the backbone of the network, increasing the learning ability of the convolutional layer and also reducing storage costs. The researchers also improved their pothole detection mechanism by using the YOLOv4 algorithm for various object detection applications. Various pothole detection mechanisms using YOLOv4 [32, 35, 36] have been proposed. Authors Sung-sick et al. [36] used different modules from YOLOv4 and YOLOv5s for pothole detecting applications. The research concluded that YOLOv4-tiny has higher precision compared to YOLOv5. The main disadvantage of this paper is that it does not provide a parameter to find the time it takes to detect the pothole, as it is an important parameter in real-time pothole detection. In the paper released by Omar and Kumar [35], uses YOLOv4 to detect potholes. The model's accuracy, recall and mAP@0.25 were determined, which were then used to evaluate the model. However, the results obtained were lower than the predicted values.

YOLOv5 [37] was released on May 27, 2020, just one month after the release of YOLOv4, and was released by authors Glenn Jocher et al. The Pytorch framework was used instead of Darknet. It has been said to be a significant improvement over both the YOLO v3 and v4 in terms of both speed and precision. Several research projects were conducted on YOLOv5 [36, 38] to test the precision and accuracy of the algorithm. Ahmed [38] proposed to find a best method to detect potholes by evaluating and comparing different parameters of different image recognition algorithms like YOLOv5, YOLOR [39] and Faster R-CNN with five different backbone structures. The results obtained show that Faster R-CNN with ResNet50 backbone has the highest precision, followed by YOLOv5. The YOLOv5 has the fastest recognition speed and the smallest model size compared to all other algorithms. It is concluded that YOLOv5-s model is more suitable for pothole detection as it has satisfactory detection speed. The main disadvantage of YOLOv5 is that no research paper has been published that can be used to clearly understand how the algorithm works. This complicates the analysis of the algorithm and reduces the reliability of works created using YOLOv5.

YOLOX Zheng Ge et al. [40] was introduced on July 22, 2021 and is one of the latest object detection method in the YOLO series. This object detection algorithm moves from the traditional anchor-based image detection technique to an anchor-free method. The authors Zheng Ge et al. belonging to Megvii technology, has won 1st place in the Streaming Perception Challenge (Workshop on Autonomous Driving at CVPR 2021) [41] by using the YOLOX-L model. The YOLOv3 model was used as a baseline and the Darknet53 is used as backbone. It achieved 50.0% AP on COCO dataset at a speed of 68.9 FPS on Tesla V100, exceeding YOLOv5-L by 1.8% AP. The system presented by Teerapong et al. [42] uses YOLOX algorithm to detect road assets on the highways. To avoid using multiple cameras to detect objects from all four sides of the vehicle, the authors developed a solution that uses a 360-degree spherical camera to detect objects and other approaching vehicles. The camera is mounted on top of the vehicle and the 360-degree image obtained is converted into a 2D panoramic image. The objects present in the panorama are then recognized with YOLOX. The proposed system is found to have an AP of 61.5% which is higher than the previously used system by an AP of 2.56%. Road objects such as kilometer signs, milestones, and other vehicles can be detected, but the size of objects found in the panoramic angle varies drastically, and detecting small objects that are located far away from the camera is very difficult. Although the panoramic camera solves the problem of requirement of multiple cameras for a vehicle to look in all directions, the image or video feed from the panoramic camera is distorted and curved, resulting in lower levels of precision and dampening the detection mechanism.

3 Methodology

3.1 Dataset pre-processing

Datasets were downloaded from the Roboflow public library [43], which contains 665 pothole images, annotated in various formats and free to use. Some of the images in the dataset used were shown in Fig. 1. We used the Pascal VOC annotation format to denote the bounding boxes that indicate the potholes in an image. The annotations of the images were saved in XML file format. The number and type of classes present in the dataset is specified in a separate protoBuf text file, which greatly simplifies class editing. Here we have only one class named 'pothole' stored in the pbtxt file. YOLOX model requires the annotation format as per the COCO dataset format which is formatted in json file. The limitation with the json format is that it is difficult to add more images to the existing formatted dataset. Splitting the data between training and testing also becomes difficult once the data has been formatted in COCO format. This is because the annotation information of all images is stored in a single JSON file, making the data difficult to edit. To solve this problem, annotation files were created according to the Pascal VOC format. In the Pascal VOC format, each image has its annotation information stored in its own XML file. If there are 'n' images in a dataset, 'n' annotation files, or in this case XML files, are created. While in the case of COCO, for n images in a dataset, only one annotation file or JSON file is created, storing all the annotation information of the whole dataset.

After receiving the dataset, it is divided into training and testing in a ratio of 80:20. After a model is trained using the training set, the model is tested by making predictions on the test data set. It is easy to determine whether the model's guesses are correct because the data present in the test set already contain known values for the attribute that is needed to be predicted.

Once the split is done, the training and testing data set needs to be converted in accordance with the COCO annotation format. This step is required because YOLOX can only be trained and validated if the dataset annotations are in COCO format. To facilitate the XML file format to JSON conversion, the filenames of the XML file and the images must be the same and the filename can only be in

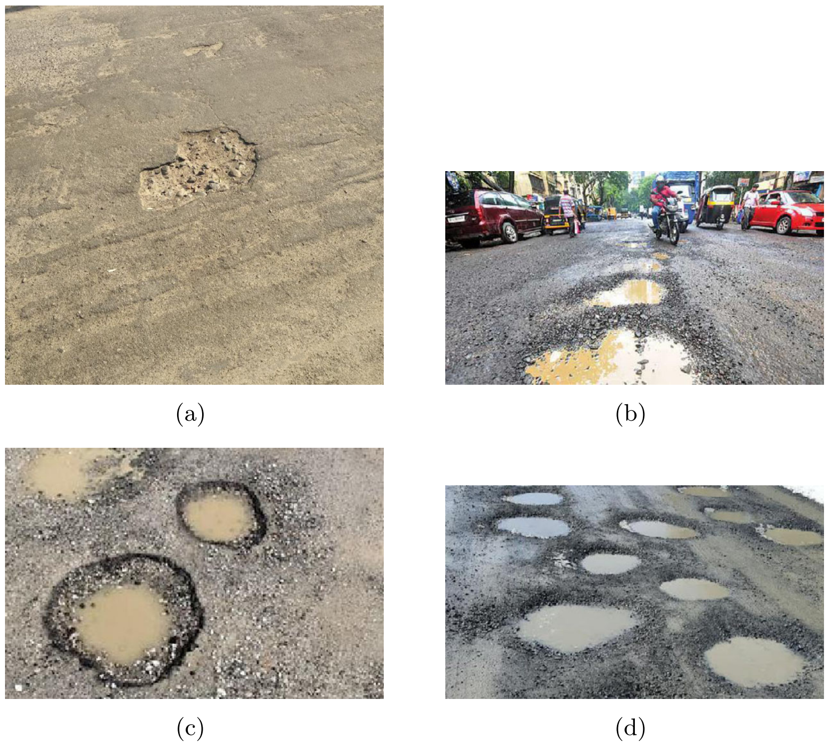

Figure 1 Sample pothole images used for training and testing the model

string format. In addition, the XML file also stores the initial file name when it was created. Therefore, we need to edit the content of each file of the data set so that all file names are integers and the corresponding image name and XML file name are the same.

After the pre-processing is done and the files are split, the annotation format is converted from Pascal VOC to COCO format. After conversion, a single JSON file is created containing all the annotation details of the images. This converted dataset is then used to train the model.

3.2 YOLOX

One of the most important requirements for object detectors for real-time applications is that the model must be fast and accurate. Speed and accuracy are often inversely related in most applications, and this is true to some extent in object detection models. But it is still possible to increase both the speed and accuracy of an object detector once a good understanding of the various components of object detectors is obtained. YOLO series detectors are always considered the best compromise between accuracy and speed and are very well suited for real-time applications. The YOLOX algorithm is the latest in the series and is created by utilizing and updating the YOLOv3 algorithm. In order to understand why YOLOX is much better for real-time applications and how it achieves its speed and precision, we need to understand the structure of the YOLOv3 algorithm and what are the changes made by YOLOX algorithm from the basic YOLOv3 algorithm.

3.2.1 Backbone

Both YOLOX and YOLOv3 use the Darknet-53 backbone architecture. For object detectors, the backbone is the first layer that receives the data image. Darknet-53 is a huge backbone composed of multiple layers of 1×1 convolution and 3×3 convolution layers to help extract more features from the original data. YOLOX also uses an additional SPP (Spatial Pyramid Pooling) layer to get the best characteristics of the data from the max pooling layers of the backbone. The model also uses Feature Pyramid Network (FPN) which extracts features from the image with multiple aspect ratios, i.e., with multiple widths and heights. This layer releases three channel outputs with different feature sizes. The first output has 256 features, second output has 512 features, and the third output has 1024 features. The main goal of this process is to facilitate the process of encoding the data used by the detector layer head to make final predictions. This FPN layer sits at the bottom of the backbone layer and all output is sent to the top layer of the object detection model.

3.2.2 Head

Now that the feature extraction part is complete, we need to use these features to perform actual tasks like localization, detection, pose estimation, etc. In object detectors, Head is a sophisticated architecture used to analyse the features received from the backbone to give concrete outputs. The input of the head is received from the FPN network consisting of three different outputs at three different channel scales which has 1024, 512 and 256 features.

Although the backbone layer of YOLOv3 and YOLOX is quite similar, the head of YOLOX model is completely different from that of YOLOv3 model. YOLOX uses what is known as a decoupled head, while YOLOv3 uses a coupled head. The difference made by these two types of head is clearly noticeable in the output predictions. The YOLOX output consists of three different matrices that contain different information, while the YOLOv3 output consists of a huge matrix that summarizes all information and output results. It should be noted that the three feature outputs of YOLOX contain the same information as the single huge feature output given by the head of YOLOv1. Using the decoupled head significantly increases the convergence speed of the model compared to the coupled head structure.

Figure 2 shows the structure of the YOLOX head. The output from the FPN layer is first given to the 1×1 convolution layer. This layer is used to reduce the channel dimension of the input data. Then the output of the 1×1 conv layer is passed to two parallel 3×3 conv layers. One branch is used for the classification task and the other for regression. The regression branch is further split into two parallel layers to get a regression output and an IoU (object) output. The first branch that performs the classification task gives the class (Cls.) output. This output detects the object in the bounding box and tells what object is inside the bounding box. The Regression (Reg.) output determines the position of the bounding box. There are four main components that must be determined to locate the bounding box, they are width, height and the x, y coordinates. The IoU output shows the confidence percentage of each bounding box. It determines how confident the network is with respect to the specified object that exists within the bounding box.

The number of outputs obtained for each image is different in YOLOX compared to YOLOv3. Note that there are three different outputs received from the FPN layer of the backbone network. So each image will get are three different outputs from each head instead of one. So in the case of YOLOv3 we get three outputs and in the case of YOLOX we get 3 outputs for each (Cls.), (Reg.) and (Iou.) output, making a total of nine outputs for each image.

3.2.3 One-stage object detection

Object detection algorithms using deep learning can be classified between one-stage detectors and two-stage detectors [44]. There are two tasks that the object detector must solve in order to successfully classify an object. First it has to find a random number of objects that are present in

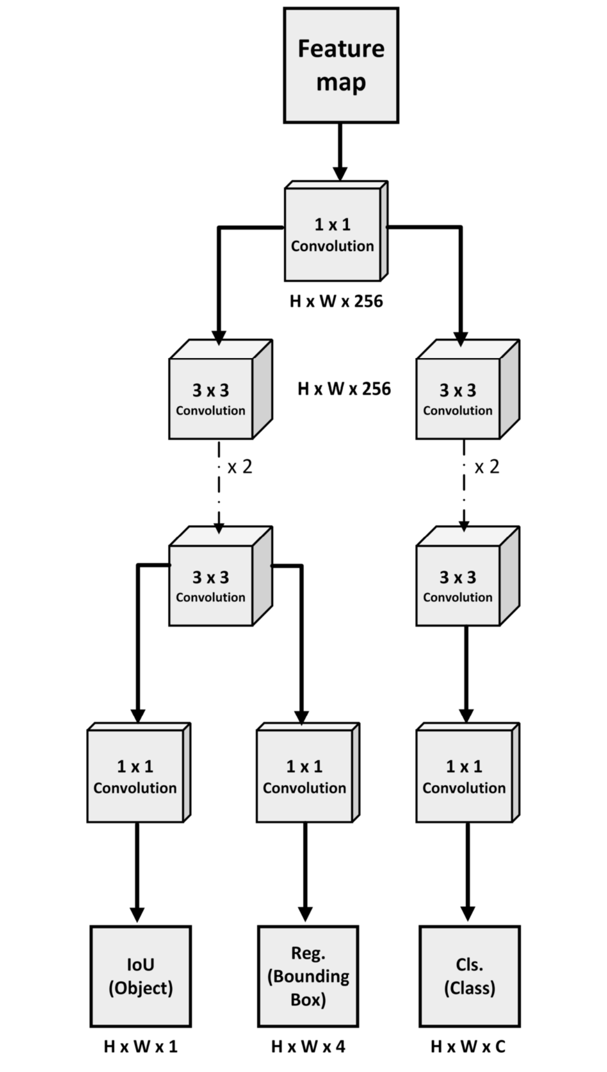

Figure 2 Schematic structure of YOLOX head

the image. The number of detections need not be exact as it is chosen arbitrarily, but it must follow certain functions in order for the number chosen to make sense as it moves further towards the convolutional layers. Second, the algorithm must classify each individual object and evaluate the object's size and position using a bounding box.

Two-stage object detectors, as the name suggests, treat these two problems in two stages. In the first stage, a specific algorithm is used to provide the region proposals [45]. For example, the Fast RCNN object detection technique uses the selective search algorithm for regional proposals, and this algorithm suggests about 2000 regions for each image. All proposed regions will be applied in the pipeline down to the second stage. The second stage consists of a CNN network which has to classify the object and process the bounding box position by performing a pose estimation. For example, in the case of Fast R-CNN, the convolution neural network layer considers the 2000 suggested regions in the image and attempts to identify a region that best matches the existing object. Once the object is identified, a separate network estimates the best positions and dimensions for the bounding box.

Thus, an image is sampled at least three times in the two-stage object detectors, once for region proposals, once for object detection, and once for bounding box prediction. Some of the older object detection models based on CNN typically belong to the two-stage object detection algorithm. Algorithms like SPP-Net, R-CNN (Region based CNN) and its successors like Fast R-CNN, Faster R-CNN, Mask R-CNN and R-FCN (Region-based Fully Convolution Network) all belong to Two stage object detectors [46].

Unlike two-stage detectors that work with one problem per stage, single-stage detectors like YOLOX try to do both tasks in a single step. Instead of adding the pose estimation in the secondary stage, it is integrated directly into the object detection model, increasing the speed of the model by being up to 100 times faster than the R-CNN method. Here, the computational cost for pose estimation and object detection is shared, eliminating the need for a separate stage for pose estimation, thereby reducing computational cost and increasing speed. Another important factor that increases speed is the fact that the image does not have to be sampled several times [47]. The image is only sampled once since single-stage detectors rely on convolutions for detection and estimation instead of relying on sampling.

Now that there is a possibility to reduce the time delay by using the single-shot detector model [48] for YOLOX, experiments were conducted to further reduce the delay in other venues. Research shows that using anchors to determine localization costs more time and removing these anchors could further help eliminate excessive timing delays and computational costs.

3.2.4 Anchor

The YOLO series detectors are always considered the best compromise between accuracy and speed and are very well suited for real-time applications. Anchors have played a key role in the progress and advancement of object detectors. However, the results show that anchors consume significant computational power of the model [49]. This in turn increases the time it takes to recognize objects. To counteract this problem, several advances have been made in recent years to make object detectors anchor-free.

Anchors are basically predefined training samples with many hyperparameters that need to be carefully tuned depending on the application. The larger the anchor boxes, the more difficult it is to predict the smaller objects. The smaller the anchor boxes, the greater the prediction time and smaller the IoU (intersection over union). The number of anchor boxes used will also affect the final performance of the detector. Even the smallest change in these hyperparameters has a significant impact on the end result. Anchor boxes are present so the model does not need to predict bounding boxes directly. A set of predefined anchor boxes are spread across an image. They are already defined with certain height-width ratios. After detection, the anchor box closest to the object in terms of distance and size is tailored to fit tightly with the detected object. The anchor also increases the complexity of the detection head. Thus, removing the anchor and implementing the anchor-free mechanism helps to reduce the complexity of the model as no clustering techniques are required to determine the hyperparameters, which makes the training time is significantly shorter [50]. For example, in YOLOv3, a total of 9 anchor clusters are used by default for all 3 levels. The size of these anchors depends on the size of the image used.

To reduce the number of design parameters that require heuristic tuning and to avoid excessive computational costs incurred by other methods such as anchor clustering, etc., the YOLOX detection algorithm shifts to an anchor-free model. Instead of predicting an offset from the anchor box, YOLOX attempts to directly predict the bounding box for an object. To do this, the number of predictions allowed for each region is reduced from 3 to 1 and the positions of the bounding boxes are calculated by directly predicting four values—two offsets to the top corner of the grid, height and width of the predicted bounding box.

Direct predictions are not as direct as the name might suggest, as the YOLOX algorithm uses a method called striding to create an offset in the image. FCOS (Fully Convolutional One-Stage Object Detection) [51] is one of the first object detectors to use striding instead of anchors. Basically, a two-dimensional grid-like structure is created and the intersections of two orthogonal lines are considered as offsets of the image. Depending on the image size, object and requirements, the number of orthogonal lines are changed which changes the location and number of offset points. Using these offset points, the bounding boxes are easy to scale and predict. For each FPN layer in YOLOX algorithm, offset strides of 8, 16, 32 are used.

The main benefit of using striding instead of anchor boxes is that anchor boxes change with each image depending on the dimension and size of the image and the object, which costs an enormous amount of computational cost due to the excess hyperparameters that require precise tuning. Whereas the offset points are fixed and don't vary much with the resizing of the image and the object. The computational effort is greatly reduced by using striding since there are no hyperparameters in the model that require tuning.

3.2.5 Augmentation

Data augmentation is a well-used technique in object detection that helps diversify the training dataset by applying different transformations to the images. This helps increase the data sample of the dataset, which helps train the model to achieve higher accuracy. The transformations applied to the images in the data augmentation step should ensure that the model becomes more robust by creating variations on the images that the model can see in the real world. Gathering and labelling the data for records is a time-consuming and costly process, and utilizing data augmentation techniques reduces these costs significantly. There are two main augmentation methods used by YOLOX to boost its performance. Some of the main augmentation methods are:

Mosaic

Mosaic [52] is an augmentation method that has already proven to be efficient and is already used in YOLOv3. First, 4 different images are chosen from the training data set. These images are scaled to the same dimension (usually square in shape). These four images are placed side by side like the 2×2 matrix shape to form a larger square. The bounding box present in each training image is adjusted to fit this newly created larger image. Then a piece of this image is randomly cut out, such that the resulting image contains a part of all 4 image samples. Now the bounding boxes that exist outside of this newly cropped image are removed and this newly augmented image is used to train the model.

Mixup

Mixup's [53] intended purpose was to perform classification tasks, but it also works well for object detection problems. The Mixup algorithm averages two images over each other based on an averaging parameter. Once the images are overlapped, the bounding boxes corresponding to both images are placed on top of each other in the final overlapped image. Experimental results show that Mixup improves the generalization process of various cutting-edge neural network architectures. Mixup also reduces the memorization of corrupt labels, increases robustness against adversarial examples, and stabilizes the training of generative adversarial networks.

3.2.6 Label assignment

Label assignment is an important task in the training phase of object detection model. Label assignment divides the predicted bounding boxes into positive and negative groups. Not all bounding boxes predicted by the model are good. Some of the predictions are really bad and we don't want the model to believe that these wrong predictions are correct. It is important that the model learns from the mistakes and errors so that the same mistakes do not happen again in the future. Good predictions are placed in a positive group and the bad predictions are placed in a negative group. The negatively labelled group is used by the loss function, which attempts to assess how bad

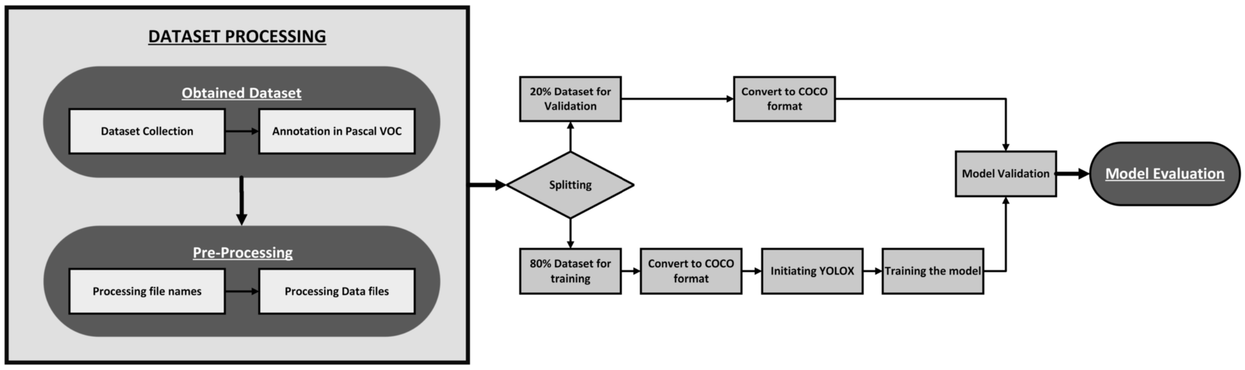

Figure 3 Flowchart of the System Workflow

these predictions are and feeds back into the network. The positively labelled group is used to assign a ground truth bounding box to an object. These ground truth bounding boxes are nothing more than the original bounding box that perfectly encapsulates the given object. They are used as standards for future predictions of objects. In order to be able to distinguish between positive and negative predictions, YOLOX uses the dynamic label assignment tool SimOTA.

The SimOTA labelling tool is based on a widely used global context label assignment tool known as OTA. This OTA algorithm [54] approaches the label assignment problem as a discrete OT (optimal transport) problem. The main goal of an OT problem is to move one mass distribution to another as efficiently as possible. The goal of the discrete OT problem is to find a match between two data points x and y, belonging to data X and Y, respectively, such that the average cost of all matches from data X and Y is as small as possible.

The OTA algorithm is designed to have high computational power and increase time delay. Therefore, as such, it cannot be used in an algorithm that prioritizes high speed and less complexity. So, the authors developed the SimOTA algorithm, which basically removes some iteration steps of the OTA algorithm and replaces it with approximation functions instead. This contributes significantly to the algorithm working with higher speed and reduced complexity.

3.2.7 Non-max suppression

During the detection of an object in an image, multiple bounding boxes pointing to the same object often appear. This group of bounding boxes are placed closer together and their size and location are slightly different from each other. The main problem faced by the detector model is that the ground truth of the image is difficult to locate and assign in these scenarios. In these cases, these clusters of overlapping bounding boxes are removed, leaving only a single bounding box. For this purpose, the YOLOX model uses what is known as non-max suppression.

The non-max suppression algorithm [55] removes the bounding boxes by calculating the IoU value of these predictions. The box with the lowest IoU value is removed from the prediction cluster. The mean of the remaining bounding boxes is calculated and the boxes whose IoU value is below the scoring threshold are removed from the cluster. This process is repeated until only one bounding box remains.

3.3 Workflow

The object detection model was trained on a laptop with access to the Google Colab virtual machine, which allows calculations to be performed on the Tesla K80 GPU with 12 GB of RAM memory. As the YOLOX dataset must be formatted according to the COCO 2017 dataset. The preprocessed data set is therefore only assigned to certain folders from which the YOLOX model takes the inputs for the training. The model was designed to identify only one class—pothole. The visual representation of the workflow is shown in Fig. 3.

4 Results

Figure 4 shows the detection results obtained using the weight with the highest AP value. The resolution of the first image shown in Fig. 4 is 400 × 300. This image contains multiple potholes that overlap. There are also many vehicles and people interacting with the potholes. This makes it difficult to spot potholes. We can see that the YOLOX model failed to spot some potholes present around the centre of the image. Nevertheless, the model was able to achieve a highest confidence threshold of 91.8%. The lowest confidence threshold is 35.0% and is present around the centre of the image. The percentage of the number of potholes identified compared to the total number of potholes present in the image is approximately 55%. The average confidence threshold that the image achieves is 81.3%.

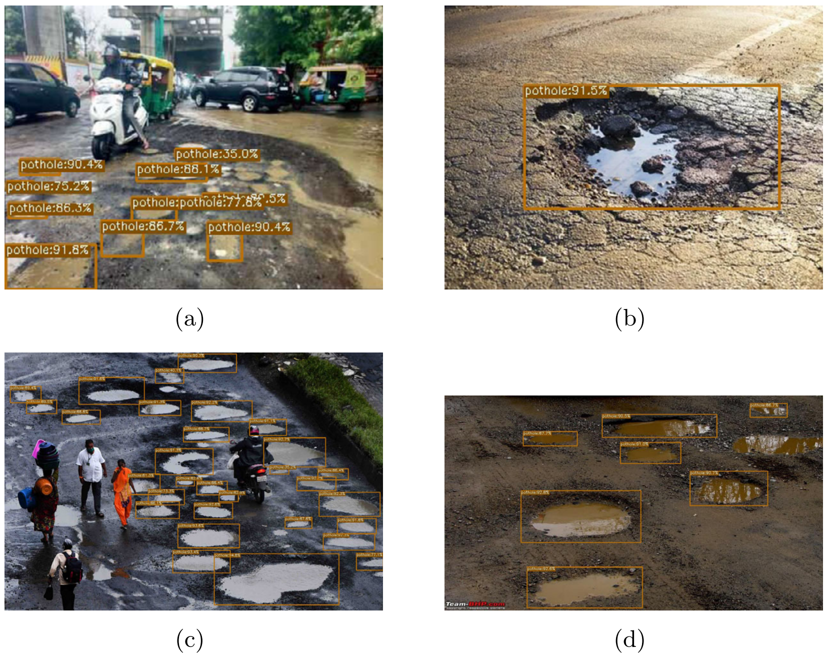

Figure 4 Detection results from YOLOX

The second image in Fig. 4 also has a resolution of 400 × 300. This picture contains only one pothole. The confidence threshold achieved for the image is 91.5%.

The third image shown has a resolution of 1280 × 872. This image also shows human-vehicle interactions. Some people block the camera's view of the pothole, making the pothole only partially visible and more difficult to predict. Despite this, the model was able to predict most of the potholes in the image. Of the 36 potholes in the image, the model was able to identify 30. The percentage of predicted potholes is about 83.3%. The highest confidence level obtained was 94.8% and the lowest was 35.2%. The average confidence threshold obtained from the model is 83.9%.

The fourth image in Fig. 4 has a resolution of 1186 × 673. There are several potholes in the image, but there is no interference or pothole obstruction in the image. But the presence of tree shadows and reflections in the potholes reduces the contrast between the pothole and the road, making it difficult for the model to detect potholes. The model predicts 7 out of 8 existing potholes. The percentage of predicted potholes is around 87.5%. The highest confidence threshold reached was 92.8% and the lowest confidence threshold was 86.7%. The average confidence threshold of the image is about 90.3%.

The analysis of the results shows that parameters such as image resolution, pothole size and the interaction of potholes with the environment play an important role in achieving higher accuracy of our model. If the image resolution is low, potholes are small, and various factors obstruct the potholes, like the first image in figure, the recognition process will be hindered and the resulting precision will be low. While in the case of the fourth image of the figure, the image resolution is high, potholes are of reasonable size and there are not many obstacles, resulting in a higher level of accuracy.

During the training of the YOLOX-nano model for pothole detection, the training was done in batches, each having trained for 100 to 300 epochs per batch and after successful training of a batch, the model is stored for future reference. This trained batch is then used for further training of the model for higher epochs. The analysis of the image presented above were conducted on a model that had the highest AP values. Here, at epoch 3200, the obtained weight model gives the best AP of 85.6%. The model is trained on a pothole dataset for multiple iterations and epochs, and the mAP values are recorded. The module is trained for 5000 epochs in multiple batches. The total training time of the module was more than 28 hours. It should be noted that in the evaluation model of the COCO dataset, no distinction is made between the values of IoU and mAP. So AP, mAP @0.50, mAP @0.75 and AR of the model that are calculated. Table 1 shows the results obtained.

Figure 5 shows the AP values in relation to the epochs. It should be noted that at the top of the AP plot w.r.t. to epochs, the average AP value at 250 epochs is plotted at the top of the chart to better visualize the changing trends in the value of AP across epochs. The largest AP value was

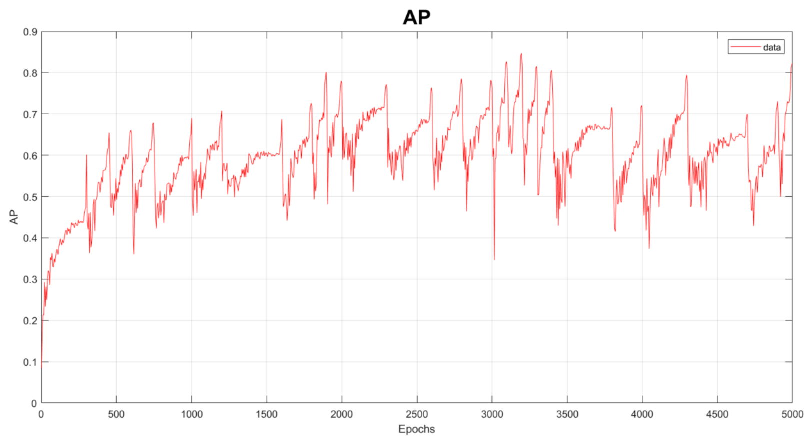

Figure 5 Change in AP with respect to Epochs

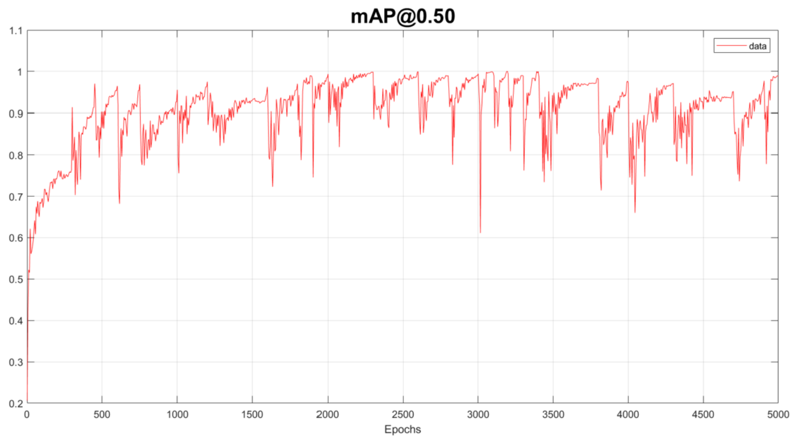

Figure 6 Change in mAP@0.50 with respect to Epochs

Table 1 Results obtained for YOLOX-nano model trained to detect pothole

Model	AP	mAP@0.50	mAP@0.75	AR	Epoch	Size
YOLOX-nano	0.856	1.00	0.99	0.42	5000	7.22 mb

found to be 0.856, which was reached at about 3200 training epochs of the model. The AP was found to steadily decrease after 3200 epochs due to over-fitting of the model.

Training was stopped at 5000 epochs to avoid further overfitting of the model. The model trained up to 3200 epochs was used for further pothole predictions. Here it should be noted that AP is taken as the average mAP over various IoU thresholds from 0.5 to 0.95 at a step interval of 0.05. We use the MS COCO Challenge scoring metric, denoted as Average Precision (AP)@[IoU=0.50:0.95] as AP [56].

There are also several research papers that use the Pascal VOC evaluation metric to calculate the AP. Because the COCO evaluation metric states that there is no difference between AP and mAP, the results obtained in this section are used for comparison and analysis with other research that has followed the Pascal VOC rating metric.

The mAP for object detection is the average of the AP calculated for all classes. mAP@0.5 means it is the mAP calculated with the IoU threshold configured to 0.5. The mAP is a good measure of the sensitivity of the neural network. Figure 6 shows the mAP@0.50 values in relation to the epochs. As in Fig. 5, the mean mAP values were plotted at 250 at the top of the graph. 100% mAP@0.50 values were reached at around 2000 epochs. It is quite surprising to find an mAP@0.50 value of 100%, and this value was also reached long before other parameters such as AP, mAP@0.75 could reach their respective maximum values. The 100% mAP at a threshold of 0.50 indicates that the

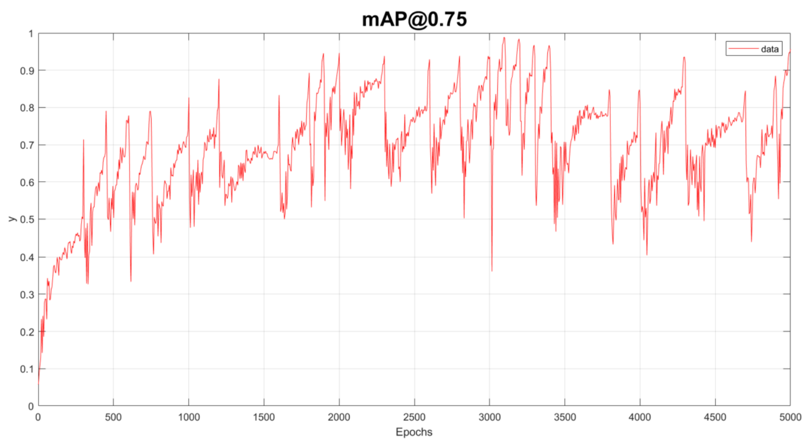

Figure 7 Change in mAP@0.75 with respect to Epochs

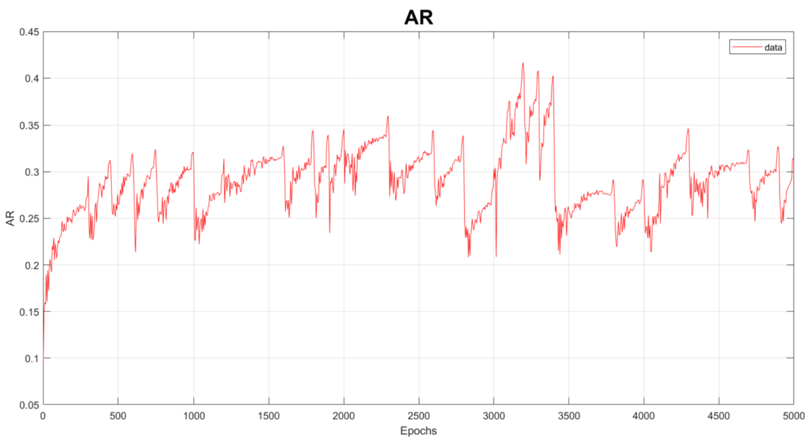

Figure 8 Change in AR with respect to Epochs

value of the intersection over the union (IoU) of the predicted bounding box that overlaps the ground truth of the object present in the image is greater than 50%. i.e., at least 50% of the object is within the predicted bounding box. The model must predict all images of the training dataset such that at least 50% of the object is covered by the bounding box to achieve a value of 100% mAP@0.50. Since the size of the data set is not that large and since the mAP threshold is only 50%, both factors could have allowed the model to achieve an mAP value of 100%.

Figure 7 shows mAP@0.75 values in relation to epochs. The mean values at 250 were plotted at the top of the chart for better visualization and the highest mAP value was 0.99 after 3100 training epochs. Note that the highest mAP@0.50 value was reached faster than AP and mAP@0.75. Figure 8 shows the average recall value of the model in relation to the number of epochs. The maximum AR value was found to be 0.42 at around 3200 training epochs.

5 Result comparison

Some of the main properties that the model must meet in order to use the model successfully in real world applications are that the model must be light and should have a considerably high precision, i.e., the trained model must be small in size and should have less computational effort for easy deployment for mobile applications. The trained model must also be comparable to other cloud-based heavyweight models that have high computational requirements in terms of accuracy. Here we compare our model with other mobile models like YOLOv5s, YOLOv4tiny, YOLOv3-tiny. The lightweight models are chosen for comparison because these models have less processing power and hence are small in size and faster in terms of speed. This comparative analysis shows how

Table 2 Comparison of mAP@0.50 value of various Algorithm

	YOLOXnano	YOLOv5s	YOLOv4tiny	YOLOv4	YOLOv3	YOLOv3tiny
mAP@0.50	100.0	74.8	78.7	77.7	70.24	49.71

Table 3 Comparison of AP, Size, Inference Speed and epoch values of various Algorithm

Model	Epoch	AP	Size	Inference speed
YOLOX-nano	5000	85.6%	7.22 mb	0.0380 s
YOLOR-P6	100	43.20%	291.8 mb	0.03 s
YOLOR-W6	100	44.60%	624.84 mb	0.032 s
YOLOv5s	1200	58.90%	14.8 mb	0.009 s
YOLOv5m	1200	61.54%	43.3 mb	–

well the YOLOX model compares to other state-of-art mobile detectors. The model is also compared with heavy weight models like YOLOR-P6, YOLOR-W6, YOLOv5m, YOLOv4, YOLOv3. By comparing the YOLOX model with heavy models, we can analyse the robustness of our model in terms of accuracy and precision.

The mAP values at 0.50 of different models were compared in Table 2. In Table 3, the YOLOX model was compared to other state of art models based on parameters such as AP, size and inference speed.

The mAP@0.50 values of YOLOv5s, YOLOv4tiny, YOLOv4 used in Table 2 were from the results of Park et al [36]. In the same table, the mAP@0.50 value for the YOLOv3 model was reported by Lin et al. [33]. Similarly, the corresponding mAP@0.50 values of YOLOv3tiny were obtained from research by Dharneeshkar et al [30]. In Table 3, the corresponding result values of YOLOR-P6, YOLOR-W6, YOLOv5m and YOLOv5s were taken from the results of the research work done by Ahmed [38]. For each parameter, the corresponding bar charts have been plotted to better understand the strengths and weaknesses of our model compared to others.

Figure 9 shows the comparison of the Average Precision of different models. Here the model is compared with YOLOR-P6, YOLOR-W6, YOLOv5s, YOLOv5m, YOLOv4. Our YOLOX model has the highest accuracy of any other model, outperforming the heavyweight YOLOv5m model by 8.2%. The Mean Average Precision value at 0.50 (mAP@0.50) is calculated for YOLOX-nano and is compared to YOLOv5s, YOLOv4tiny, YOLOv4, YOLOv3, YOLOv3-tiny. The YOLOX-nano model has the highest mAP value of 100% and outperforms the YOLOv4tiny by 21.3%. Figure 10 shows the mAP@0.50 comparison of the above models. The size of our model after training is 7.22 megabytes, which is significantly smaller compared to other models shown in Fig. 11. On the other hand, Fig. 12 shows that our model has an inference period of 0.038 seconds, which is slow compared to other models.

The results of the comparative analysis for the mAP@0.50 values in Table 2 show that the YOLOX-nano model has the highest mAP value, followed by the YOLOv4-tiny, YOLOv4, YOLOv5s and YOLOv3-tiny models in the same order. The YOLOX-nano model outperforms the YOLOv4-tiny by 21.3%. The mAP value of the YOLOv3 tiny model is only 49.71%, which is very small and is unsuitable for use in real world applications. The YOLOv4-tiny and YOLOv4 models have almost similar accuracy values. In particular, the YOLOv4 Tiny model is small and performs well with an mAP of 78.7% compared to its YOLOv4 counterpart at 77.7%. YOLOv5s has an acceptable mAP value of 74.8%. The YOLOv3 model has a mAP value of 70.24%, which is just above the 70% mark. Figure 10 shows the mAP@0.50 comparison of the above models for better visual understanding.

Figure 9 shows the comparison of the Average Precision of different models. Here the model is compared with YOLOR-P6, YOLOR-W6, YOLOv5s, YOLOv5m, YOLOv4. Our YOLOX model has the highest accuracy of any other model at 85.6%, outperforming the heavyweight YOLOv5m model by 8.2%. The YOLOv5s model, which is a lightweight model, also has a similar AP value to its YOLOv5m counterpart. The YOLOR-W6 and YOLOR-P6 models have corresponding AP values of 44.6% and 43.2%, respectively.

The size of our model after training is 7.22 megabytes, which is significantly smaller compared to other models shown in Fig. 11. This indicates that the computational effort of the YOLOX-nano model is significantly lower compared to other models. The YOLOv5s model follows the YOLOX-nano model, having a size of 14.8 megabytes which is more than twice the weight of the YOLOX-nano model. YOLOv5m has a size of 43.3 megabytes, which is reasonable for a heavy weight model. The YOLOR-P6 model has a weight of 291.8 megabytes and the YOLOR-W6 has a weight of a whopping 624.84 megabytes, making them exclusively usable only on cloud-based detection mechanisms.

From Fig. 12 we can see that the YOLOX-nano model has an inference time of 0.038, which is significantly larger compared to the YOLOv5s model. YOLOR models also have an inference time of around 0.03 s, which is acceptable as they focus on cloud-based object detection. The YOLOv5s model has the shortest inference time of 0.009 s, which is more than 4 times faster than that of the YOLOX-nano model. We conclude that the value of our model for the time period of inference is worse than the expected result.

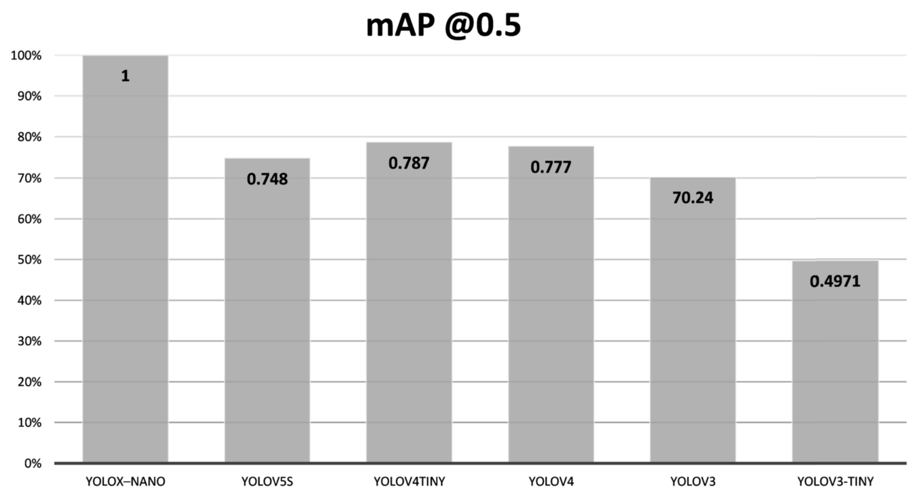

Figure 9 Comparison of Average Precision (AP) of different trained models

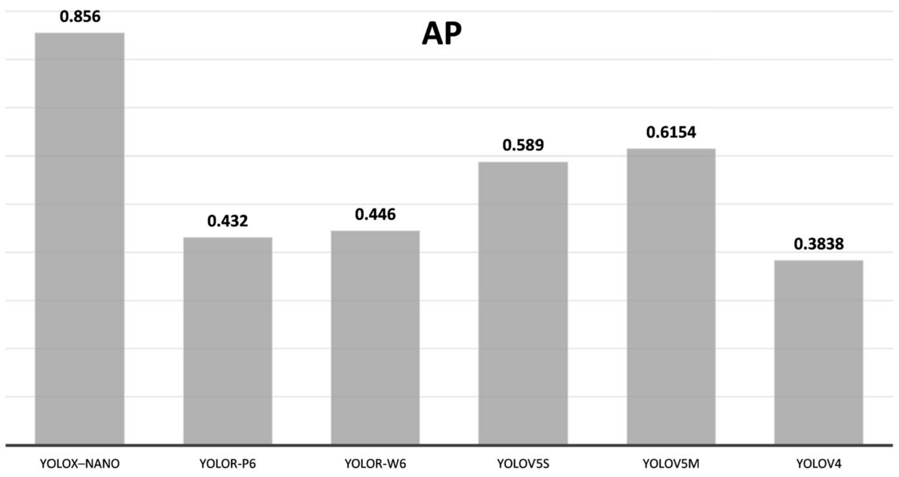

Figure 10 Comparison of mean Average Precision of the different models

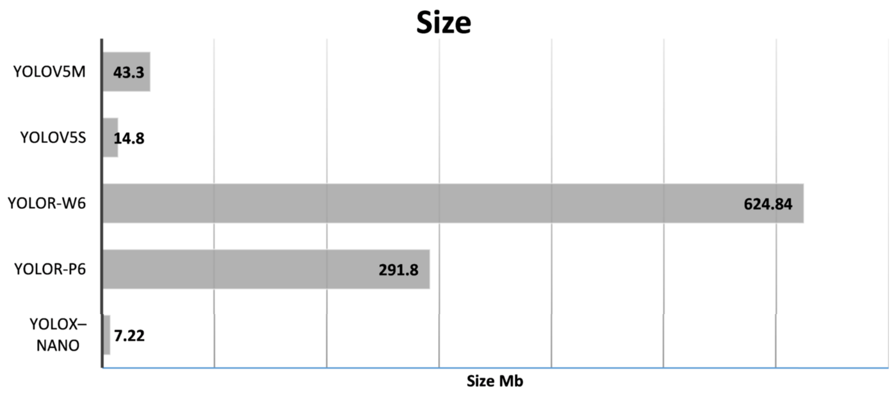

Figure 11 Comparison of size of the trained model. YOLOX-nano has the lowest size

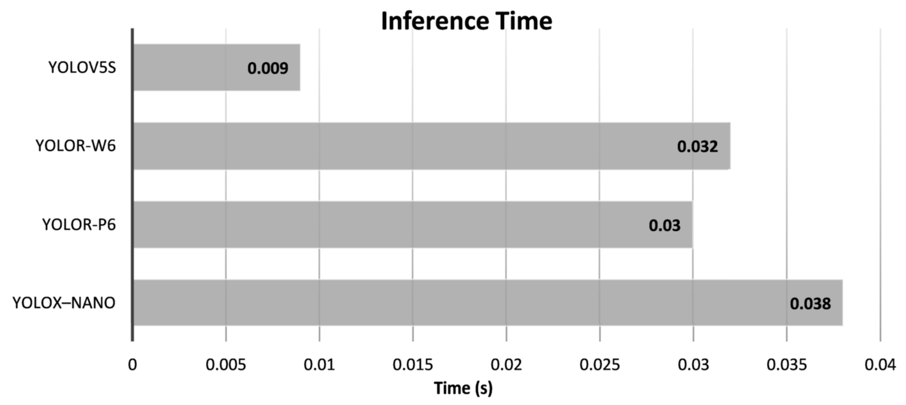

Figure 12 Comparison of inference speed of the different model. YOLOv5s has the lowest time

Some of the limitations of this comparative study are that the different models used in this study were not strictly trained under similar conditions. Although all pothole detection models have been trained using datasets with pothole images, various conditions including the size of the dataset, the images included in the dataset, the GPU used to train the model, no. of the epochs varies to some extent. A closer look at these results reveals some anomalies and these results should be interpreted carefully so as not to discount or disregard other object detection methods. The comparative analysis performed on parameters such as the inference speed and the size of the model holds true because these are innate properties of the object recognition model and do not change much even after a large effort of training the model. Whereas the parameters like AP and mAP values change over a long period of training. In our analysis, in Table 3, the number of training epochs performed on YOLOv5 models is 1200 epochs and the training performed on YOLOR models is only 100 epochs. While in the case of YOLOX Nano mode, the number of training epochs performed is 5000 and the best AP value was obtained at around 3200 epochs. This difference in epoch values could be the reason for the resulting AP values.

It is true that it is unfair to compare models trained with different epochs for accuracy. But, we can still compare YOLOR models with the YOLOX nano model trained for 100 epochs and compare YOLOv5 models with the YOLOX nano model trained for 1200 epochs. These comparison results show that the YOLOX-nano model far outperforms the YOLOv5 models. At 1200 training epochs, the YOLOX-nano achieves an AP of around 71.6%, which is more than 10% higher than that of the two YOLOv5 models. At 100 training epochs, the YOLOX-nano model achieves about 38%, which is less than the YOLOR models but is still comparable to these heavyweight detection models.

These results demonstrate the potential advantage of YOLOX-nano model over other established methods. The pothole detection model based on YOLOX-nano model outperforms all other lightweight detectors by a large margin and performs less when compared with heavyweight models only by a small margin. The YOLOX model was able to achieve higher AP, mAP values and the size of the model was also low, which can be easily seen in the comparative table and figures shown in this section. The only shortcoming of the model is that the inference time taken is high compared to other models. This will definitely affect the application usability of the model. But still, the higher AP values and considerably lower storage space taken by the model counteracts the problem and drawbacks of the model. This makes our model more appropriate for pothole detection applications compared to other models.

6 Conclusion

Pothole detection, unlike the detection of other objects, is difficult and unique because potholes do not have a specific shape and size. They come in a variety of forms, making the detection process much more difficult. The improved pothole detection system based on the YOLOX model proposed in this article has been trained for 5000 epochs. Mean precision, mean Average Precision and Average Recall values of the model are calculated and all parameters are visualized. The data set used to train the model contains pothole images captured in various shapes and there are also multiple potholes in the images. This pothole dataset was used to train the YOLOX-nano model and compared to different YOLO models. Two models of YOLOv3 (YOLOv3 and YOLOv3-tiny), two models of YOLOv4 (YOLOv4 and YOLOv4-tiny), two models of YOLOv5 (YOLOv5s and YOLOv5m) and two models of YOLOR (YOLOR-W6 and YOLOR-P6) are taken up and used for comparison with our trained model. The experiment shows that the YOLOX-nano model has the highest precision compared to all other models. The model was able to achieve a value of 85.6% Average Precision, which is 24.06% higher than the YOLOv5m model. The mean Average Precision, evaluated using the mAP@0.5 value, also shows that the YOLOX-nano model far outperforms all other models. The model size of the YOLOX-nano model is 7.2 MB, which is less than half the size of the YOLOv5s model. The inference speed of the trained model is very slow compared to other models. It has a speed of 0.038 seconds, which is 4 times slower than the YOLOv5s model. The analysis result shows that the YOLOX nano model is very suitable for pothole detection because it can be easily deployed with less storage space and low power consumption as the size of the model is small, therefore the computing power required by the model is also less. The precision is also very high compared to other models. Although the model's inference speed is slower than other models, the trade-off between precision and speed falls in favour of the YOLOX-nano model. Therefore, our proposed model can be used to detect potholes with higher precision and is very suitable for real-time applications.

7 Future work

In the future, the system can be implemented on a real-time dashboard camera. The system can also be activated with a GPS module that will mark the coordinates once the pothole has been detected by the dashboard module. The coordinates can then be used to identify the potholes by maintenance workers who then work on the damaged road. To achieve greater accuracy, the model can also be trained on a specialized dataset, with the images in the dataset being from the vehicle's dashboard view. In addition, the size and width of the identified potholes can be measured with various image processing methods or with calibrated stereo cameras. There are some images in the dataset that were taken in different lighting conditions due to the presence of shadows. Some of the images were also taken from different camera angles. These factors may have affected the performance of the model while analysing with the test dataset. Potholes that are far away and potholes that are smaller make the detection process more difficult, and the accuracy of detection also decreases for these smaller potholes. The detection accuracy of cracks in roads is also not addressed in this research. These issues and further improvement of this model can be addressed in future studies.

Acknowledgements
The authors are grateful to our colleagues and friends of School of Electronics Engineering (SENSE), VIT University, Vellore.

Funding
Not applicable.

Availability of data and materials
Pothole dataset is collected from public library at https://public.roboflow.com/object-detection/pothole.

Code availability
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author contribution
Dataset preparation, conducting experiments, Analysing the results and Manuscript preparation was done by MPB under the guidance and supervision of SKC. The authors read and approved the final manuscript.

Publisher's Note

Received: 10 June 2022 Revised: 4 August 2022 Accepted: 5 August 2022
Published online: 31 August 2022

References

1. S.K. Sharma, R.C. Sharma, Pothole detection and warning system for Indian roads, in *Advances in Interdisciplinary Engineering*, ed. by M. Kumar, R.K. Pandey, V. Kumar (Springer, Singapore, 2019), pp. 511–519
2. H.-W. Wang, C.-H. Chen, D.-Y. Cheng, C.-H. Lin, C.-C. Lo, A real-time pothole detection approach for intelligent transportation system. Math. Probl. Eng. **2015**, 869627 (2015). https://doi.org/10.1155/2015/869627
3. S.K. Sharma, H. Phan, J. Lee, An application study on road surface monitoring using dtw based image processing and ultrasonic sensors. Appl. Sci. **10**(13), 4490 (2020). https://doi.org/10.3390/app10134490
4. A. Ahmed, M. Ashfaque, M.U. Ulhaq, S. Mathavan, K. Kamal, M. Rahman, Pothole 3D reconstruction with a novel imaging system and structure from motion techniques. IEEE Trans. Intell. Transp. Syst. **23**(5), 4685–4694 (2022). https://doi.org/10.1109/TITS.2021.3054026
5. X. She, Z. Hongwei, Z. Wang, J. Yan, Feasibility study of asphalt pavement pothole properties measurement using 3D line laser technology. Int. J. Transp. Sci. Technol. **10**(1), 83–92 (2021). https://doi.org/10.1016/j.ijtst.2020.07.004
6. R.H. Pramestya, D.R. Sulistyaningrum, B. Setiyono, I. Mukhlash, Z. Firdaus, Road defect classification using gray level co-occurrence matrix (GLCM) and radial basis function (RBF), in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)* (2018), pp. 285–289. https://doi.org/10.1109/ICITEED.2018.8534769
7. I. Sutrisno, A. Syauqi, M. Hasin, R.M. Iskandar, I. Asmara, I. Suwondo, W.D. Ardiansyah, E. Setiawan, Design of pothole detector using gray level co-occurrence matrix (GLCM) and neural network (NN). IOP Conf. Ser., Mater. Sci. Eng. **874**, 012012 (2020). https://doi.org/10.1088/1757-899X/874/1/012012
8. M. Muslim, D. Sulistyaningrum, B. Setiyono, Detection and counting potholes using morphological method from road video. AIP Conf. Proc. **2242**, 030011 (2020). https://doi.org/10.1063/5.0008282
9. S.-K. Ryu, T. Kim, Y.-R. Kim, Image-based pothole detection system for its service and road management system. Math. Probl. Eng. **2015**, 968361 (2015). https://doi.org/10.1155/2015/968361
10. M.H. Yousaf, K. Azhar, F. Murtaza, F. Hussain, Visual analysis of asphalt pavement for detection and localization of potholes. Adv. Eng. Inform. **38**, 527–537 (2018). https://doi.org/10.1016/j.aei.2018.09.002
11. C. Wu, Z. Wang, S. Hu, J. Lepine, X. Na, D. Ainalis, M. Stettler, An automated machine-learning approach for road pothole detection using smartphone sensor data. Sensors **20**(19), 5564 (2020). https://doi.org/10.3390/s20195564
12. S. Hoque, M.Y. Arafat, S. Xu, A. Maiti, Y. Wei, A comprehensive review on 3D object detection and 6D pose estimation with deep learning. IEEE Access **9**, 143746–143770 (2021). https://doi.org/10.1109/ACCESS.2021.3114399
13. N.-D. Hoang, An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter based feature extraction. Adv. Civ. Eng. **2018**, 7419058 (2018). https://doi.org/10.1155/2018/7419058
14. T. Liu, Y. Liu, Moving camera-based object tracking using adaptive ground plane estimation and constrained multiple kernels. J. Adv. Transp. **2021**, 8153474 (2021). https://doi.org/10.1155/2021/8153474

15. M. Bajammal, A. Stocco, D. Mazinanian, A. Mesbah, A survey on the use of computer vision to improve software engineering tasks. IEEE Trans. Softw. Eng. **48**(5), 1722–1742 (2022). https://doi.org/10.1109/TSE.2020.3032986
16. S. Minaee, Y.Y. Boykov, F. Porikli, A.J. Plaza, N. Kehtarnavaz, D. Terzopoulos, Image segmentation using deep learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(7), 3523–3542 (2022). https://doi.org/10.1109/TPAMI.2021.3059968
17. R. Bibi, Y. Saeed, A. Zeb, T. Ghazal, R. Said, S. Abbas, M. Ahmad, M. Khan, Edge AI-based automated detection and classification of road anomalies in VANET using deep learning. Comput. Intell. Neurosci. **2021**, 6262194 (2021). https://doi.org/10.1155/2021/6262194
18. M. Anandhalli, A. Tanuja, V.P. Baligar, P. Baligar, Indian pothole detection based on CNN and anchor-based deep learning method. Int. J. Inf. Technol. (2022). https://doi.org/10.1007/s41870-022-00881-5
19. Y. Lu, Y. Guo, M. Liang, CNN-enabled visibility enhancement framework for vessel detection under haze environment. J. Adv. Transp. **2021**, 5598390 (2021)
20. D. Luo, J. Lu, G. Guo, Road anomaly detection through deep learning approaches. IEEE Access **8**, 117390–117404 (2020). https://doi.org/10.1109/ACCESS.2020.3004590
21. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
22. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection (2015). arXiv:1506.02640
23. M. Kawano, K. Mikami, S. Yokoyama, T. Yonezawa, J. Nakazawa, Road marking blur detection with drive recorder, in *2017 IEEE International Conference on Big Data (Big Data)* (2017), pp. 4092–4097. https://doi.org/10.1109/BigData.2017.8258427
24. P. Felzenszwalb, R. Girshick, D. Mcallester, D. Ramanan, Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 1627–1645 (2010). https://doi.org/10.1109/TPAMI.2009.167
25. J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger. (2016). arXiv:1612.08242
26. M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The Pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010). https://doi.org/10.1007/s11263-009-0275-4
27. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot MultiBox detector, in *Computer Vision—ECCV 2016* (Springer, Cham, 2016), pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
28. R. Sumalatha, R.V. Rao, S.M.R. Devi, Pothole detection using yolov2 object detection network and convolutional neural network, in *Applied Information Processing Systems*, ed. by B. Iyer, D. Ghosh, V.E. Balas (Springer, Singapore, 2022), pp. 293–300
29. J. Redmon, A. Farhadi, YOLOv3: an incremental improvement (2018). arXiv:1804.02767
30. J. Dharneeshkar, V. Dhakshana, S. Aniruthan, R. Karthika, L. Parameswaran, Deep learning based detection of potholes in Indian roads using YOLO, in *2020 International Conference on Inventive Computation Technologies (ICICT)* (2020), pp. 381–385. https://doi.org/10.1109/ICICT48043.2020.9112424
31. E.N. Ukhwah, E.M. Yuniarno, Y.K. Suprapto, Asphalt pavement pothole detection using deep learning method based on yolo neural network, in *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)* (2019), pp. 35–40. https://doi.org/10.1109/ISITIA.2019.8937176
32. P.A. Chitale, K.Y. Kekre, H.R. Shenai, R. Karani, J.P. Gala, Pothole detection and dimension estimation system using deep learning (YOLO) and image processing, in *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)* (2020), pp. 1–6. https://doi.org/10.1109/IVCNZ51579.2020.9290547
33. Y.-C. Lin, W.-H. Chen, C.-H. Kuo, Implementation of pavement defect detection system on edge computing platform. Appl. Sci. **11**(8), 3725 (2021). https://doi.org/10.3390/app11083725
34. A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: optimal speed and accuracy of object detection (2020). arXiv:2004.10934
35. M. Omar, P. Kumar, Detection of roads potholes using YOLOv4, in *2020 International Conference on Information Science and Communications Technologies (ICISCT)* (2020), pp. 1–6. https://doi.org/10.1109/ICISCT50599.2020.9351373
36. S.-S. Park, V.-T. Tran, D.-E. Lee, Application of various YOLO models for computer vision-based real-time pothole detection. Appl. Sci. **11**(23), 11229 (2021). https://doi.org/10.3390/app112311229
37. G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, V. Abhiram, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, M.T. Minh, ultralytics/yolov5: v6.1—TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. https://doi.org/10.5281/zenodo.6222936
38. K.R. Ahmed, Smart pothole detection using deep learning based on dilated convolution. Sensors **21**(24), 8406 (2021). https://doi.org/10.3390/s21248406
39. C.-Y. Wang, I.-H. Yeh, H.-Y.M. Liao, You only learn one representation: unified network for multiple tasks (2021). arXiv:2105.04206
40. Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, YOLOX: exceeding YOLO series in 2021 (2021). arXiv:2107.08430
41. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in *Computer Vision—ECCV 2016*, ed. by B. Leibe, J. Matas, N. Sebe, M. Welling (Springer, Cham, 2016), pp. 21–37
42. T. Panboonyuen, S. Thongbai, W. Wongweeranimit, P. Santitamnont, K. Suphan, C. Charoenphon, Object detection of road assets using transformer-based yolox with feature pyramid decoder on Thai highway panorama. Information **13**(1), 5 (2022). https://doi.org/10.3390/info13010005
43. Pothole Dataset. https://public.roboflow.com/object-detection/pothole. Accessed: 2022-02-05
44. M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, J. García-Gutiérrez, On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. Remote Sens. **13**(1), 89 (2021). https://doi.org/10.3390/rs13010089
45. X. Lu, Q. Li, B. Li, J. Yan, Mimicdet: bridging the gap between one-stage and two-stage object detection, in *Computer Vision—ECCV 2020*, ed. by A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Springer, Cham, 2020), pp. 541–557
46. A. Lohia, K. Kadam, R. Joshi, D. Bongale, Bibliometric analysis of one-stage and two-stage object detection (2021)
47. P. Soviany, R.T. Ionescu, Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction (2018). arXiv:1803.08707
48. P. Poirson, P. Ammirato, C. Fu, W. Liu, J. Kosecka, A.C. Berg, Fast single shot detection and pose estimation, in *2016 Fourth International Conference on 3D Vision (3DV)* (IEEE Computer Society, Los Alamitos, 2016), pp. 676–684. https://doi.org/10.1109/3DV.2016.78
49. S. Liu, H. Zhou, C. Li, S. Wang, Analysis of anchor-based and anchor-free object detection methods based on deep learning, in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)* (2020), pp. 1058–1065. https://doi.org/10.1109/ICMA49215.2020.9233610
50. T. Zhang, Z. Li, Z. Sun, L. Zhu, A fully convolutional anchor-free object detector. Vis. Comput. (2022). https://doi.org/10.1007/s00371-021-02357-2
51. Z. Tian, C. Shen, H. Chen, T. He, FCOS: fully convolutional one-stage object detection (2019). arXiv:1904.01355
52. W. Hao, S. Zhili, Improved mosaic: algorithms for more complex images. J. Phys. Conf. Ser. **1684**(1), 012094 (2020). https://doi.org/10.1088/1742-6596/1684/1/012094
53. H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, mixup: beyond empirical risk minimization (2017). arXiv:1710.09412
54. Z. Ge, S. Liu, Z. Li, O. Yoshie, J. Sun, OTA: optimal transport assignment for object detection (2021). arXiv:2103.14259
55. N. Bodla, B. Singh, R. Chellappa, L.S. Davis, Soft-NMS—improving object detection with one line of code (2017). arXiv:1704.04503
56. T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C.L. Zitnick, P. Dollár, Microsoft COCO: common objects in context (2014). arXiv:1405.0312