IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Deep Learning Based Approach to Detect Potholes Using YOLO Version 7

[1]Kaushik Goswami, [2]Soumyadip Chattopadhyay, [2]Arka Kundu

[1]Faculty, [2]Post Graduate Student, Post Graduate and Research Department of Computer Science
[1] Department of Computer Science
[1]St. Xavier's College (Autonomous), Kolkata, India

Abstract: This paper is intended to study and develop a custom trained machine learning model which can detect potholes on road in real-time. Potholes and Road Craters are one of the key contributors of the numbers of accidents take place each year. A comparison for selection of object detection algorithm has been done in this paper. The proposed solution uses You Only Look Once (YOLO) version 7 object detection algorithm to train and be able to detect potholes and road craters. After training the model with the custom dataset, the model has achieved a precision of 0.94, recall value of 0.98. The mAP of the proposed solution 94.76%. The solution is able to detect potholes in both daylit and low-light real-time scenarios.

***Index Terms* - Potholes, YOLO, CNN, Single Shot Detection, IoU, Detection, Craters, Road, Highway, Accident, Deep Learning**

I. Introduction

For a nation to offer nationwide commuting options, roads are a necessity for transportation. Road infrastructure makes it possible to link people and move commodities, which improves the availability of jobs, business possibilities, and the health care system nationwide. As top-notch roads contribute to the GDP of the nation, the disastrous infrastructure of roads can be fatal for the safety of passengers and the condition of automobiles. Roads are typically built of asphalt pavement, which over time is susceptible to various structural problems. Authorities have been concerned about the asphalt pavement distresses in order to prevent unfavorable situations. These pavements are prone to two significant pavement failures, cracks and potholes, due to factors like age, poor building materials, traffic volume, weather, and an inadequate drainage system. Figure 1 illustrates a pothole as a concave-shaped depression in the road surface. Potholes are dangerous because they can cause accidents, unpleasant driving experiences, and car malfunctions.

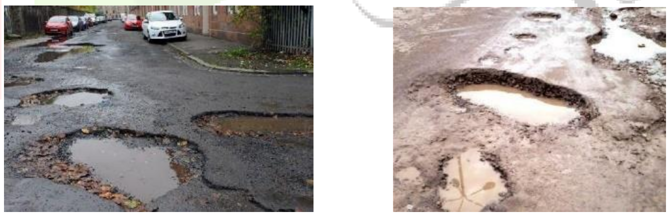

(a) (b)

Fig. 1: Road Damage causing potholes and craters

Death tolls caused by accidents have increased by almost 7% from January 2022 to May 2022, in comparison with the same for 2021 [29]. A recent statistics state that more than 2000 people have lost their lives in highway road accidents during the first five months of 2022. A data from West Bengal Traffic Police portrays that during 2018 to 2020, the number of accidents and fatalities are significant. [30]

Table 1: Accidents Statistics in West Bengal (2018-2020)

Year	*Accidents Reported*	*Souls Killed*	*Souls Injured*	*Injured Severity (Per 100 accidents)*	*Death Severity (Per 100 deaths)*
2018	10042	5417	9835	97.9	53.9
2019	10158	5500	9757	96.0	54.1
2020	9180	4927	8314	90.5	53.6

Potholes claims a large share in the numbers of accidents cause each day. As per a report of Times of India, a case-wise analysis reveals that, almost 56% of fatalities in accidents were caused by over-speeding and 28% accidents took place due to careless driving or overtaking, and more than 10% accidents' cause were bad road conditions. The highest contributor to the number of death tolls is the National Highways (34.5%), followed by State Highways (25.1%) [29]. These figures amply demonstrate the necessity for a system that can foresee and address situations of encountering potholes at high speed that can occur when driving on roads and expressways and result in deadly accidents. Although the issue cannot be resolved from the ground up, the proposed solution's development and implementation can help to some extent reduce the numbers.

II. LITERATURE SURVEY

Several studies have been conducted on deep learning-based pothole detection approaches using various machine learning-based object detection algorithms. A comparative study by Roopak Rastogi [1] et al. Al. An accuracy of 87% is achieved using the YOLO version 2 object detection algorithm. Muhammad Harun Assad et al. [31] developed an algorithm that uses YOLO v4 as the base object detection algorithm, trained on a custom dataset, and achieved 90% accuracy at 31.76 FPS. Zhang et al. [32] proposed an embedded system for road obstruction detection integrated with CNN using the Montreal Pavement dataset. The model shows that the true positive rates for potholes, patches, marks, linear cracks, and crack networks are 75.7%, 84.1%, 76.3%, 79.4%, and 83.1%, respectively. Oche et al. [33] used five binary classification models (Naive Bayes, Logistic Regression, SVM, K-Nearest Neighbors (KNN), and Random Forest Tree) to apply various classifications to data collected via smartphone and car routes. They presented a comparison of machine learning approaches. The Random Forest Tree and KNN achieved the highest accuracy of 0.8889 on the test set. To improve the accuracy of the Random Forest Tree, they tuned hyperparameters and increased accuracy up to 0.9444. The model has shown promising results on different routes and out of sample data. Arbawa et al. [34] proposed a method for detecting road potholes using the gray-level co-occurrence matrix (GLCM) feature extractor and support vector machine (SVM) as a classifier. They analyzed three features such as contrast, correlation, and dissimilarity. The results have shown that a combination of contrast and dissimilarity features exhibits better results with an accuracy of 92.033% and computing time of 0.0704 seconds per frame. Chen et al. [35] proposed a novel location-aware convolutional neural network and trained on a public pothole dataset that consists of 4,026 images as training samples and 1,650 images as test samples. The proposed model is based on 2D-vision techniques and location-aware convolutional networks. CNN networks consist of two main subnetworks; the first localization subnetwork (LCNN) finds as many candidate regions as possible by employing a high recall network model, and the second part-based subnetwork (PCNN) performs classification on the candidates on which the network is expected to focus. The proposed method achieved high precision 95.2% and recall 92.0%.

From the computational problem approach, the main problem in this domain, that has to be solved, is to detect and classify various scenarios from the environment and predict them after identifying. There are many algorithms that have been developed for object detection like SSD (Single Shot Multi box Detection) [2,7], YOLO (You Only Look Once) etc. [2,9 ,10, 12, 13 ,14, 15, 16]. The aim of this section is to study and compare these algorithms, and observe the best algorithm which is has the following qualities:

Fast:	To be able to produce output in almost real-time.
Accurate:	The accuracy of the output must be high.
Light weight:	To cut down the computational overhead in real-time environment, the algorithm must be lightweight.

In the following section, several object detection algorithms are discussed. Comparing all the pros and cons of the algorithms, the best fit algorithm which performs well in our problem space is chosen for design and development.

1. Histogram of Oriented Gradients (HOG):

One of the earliest techniques for object detection is the Histogram of Oriented Gradients [4]. It was initially released in 1986. Despite modest advancements in the following decade, the strategy did not become widely used in computer vision applications until 2005, when it began to do so. A feature extractor is used by HOG to locate items in a picture. The feature descriptor employed in HOG is a representation of a portion of an image from which we only take the most important details into account and ignore everything else. The feature descriptor's job is to transform the image's total size into an array or feature vector. To locate the most important areas of an image in HOG, we employ the gradient orientation technique.

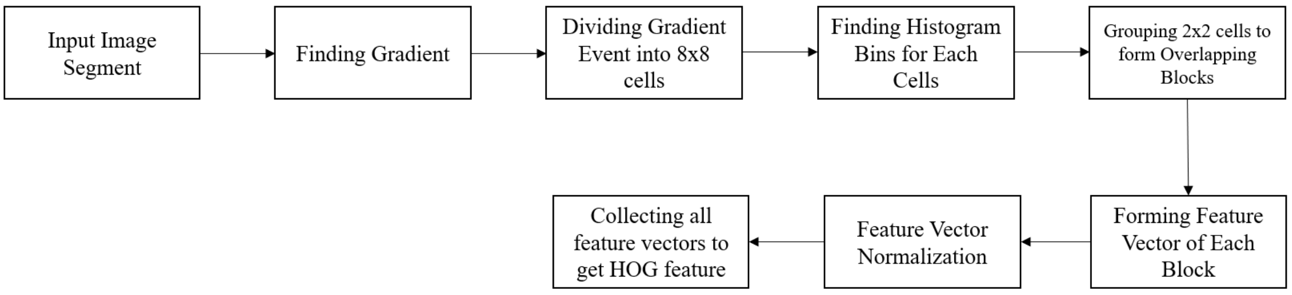

Input Image Segment
Finding Gradient
Dividing Gradient Event into 8x8 cells
Finding Histogram Bins for Each Cells
Grouping 2x2 cells to form Overlapping Blocks
Forming Feature Vector of Each Block
Feature Vector Normalization
Collecting all feature vectors to get HOG feature

Fig. 2: Steps in HOG Algorithm

The histogram of the gradient for a specific pixel in an image is computed by taking into account the vertical and horizontal values to get the feature vectors. A clear value for the present pixel is acquired by investigating the other entities in their horizontal and vertical surrounds with the aid of the gradient magnitude and gradient angles. As seen in Fig. 2, a picture segment of a specific size is taken into consideration. By breaking up the complete calculation of the picture into gradient representations of 88 cells, the gradient is first determined. We may divide each cell into angular bins and compute the histogram for the specific area using the 64 gradient vectors that are obtained. The 64 vectors are shrunk down to 9 values by this technique. We have the option to build overlaps for the blocks of cells after we have the size of the 9-point histogram values (bins) for each cell. The final steps are to form the feature blocks, normalize the obtained feature vectors, and collect all the features vectors to get an overall HOG feature.

Limitations –

The Histogram of Oriented Gradients (HOG) [4], though relatively ground-breaking in the early days of object identification, had a lot of flaws. It takes a long time to compute complicated pixels in photos and is useless in some cases involving tighter spaces and object recognition.

2. Region-based Convolutional Neural Networks (R-CNN):

The object detection process has been improved over the past approach of HOG using region-based convolutional neural networks [4]. We employ selected features in the R-CNN models [5] to try to extract the most important characteristics, which are typically about 2000 features. A selective search algorithm that can produce these more significant regional recommendations may be used to calculate the process of choosing the most important extractions.

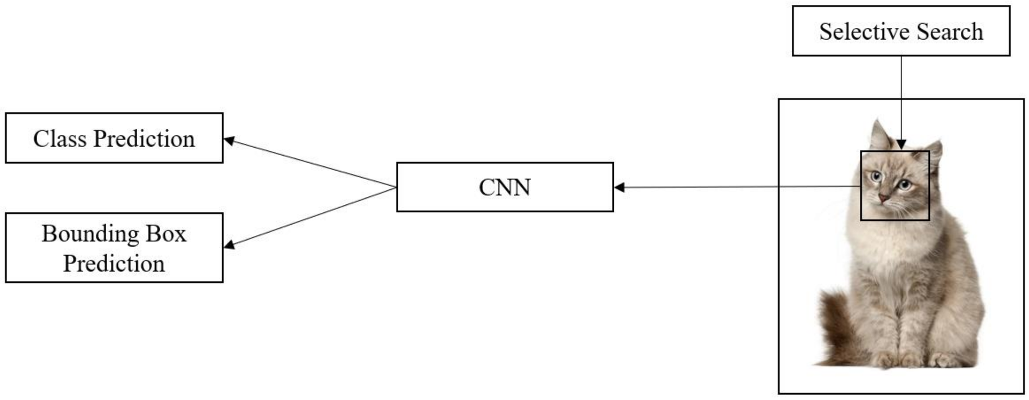


Fig. 3: Working principle of R-CNN [6]

The working procedure of the selective search algorithm to select the most important regional proposals is to ensure that you generate multiple sub-segmentations on a particular image and select the candidate entries for your task. The greedy algorithm can then be made use of to combine the effective entries accordingly for a recurring process to combine the smaller segments into suitable larger segments. Our next objectives are to extract the features and create the necessary predictions after the selective search method has been successfully performed. The convolutional neural networks can then be utilized to generate an n-dimensional (2048 or 4096) feature vector as an output once we have made the final candidate suggestions. The task of feature extraction can be easily completed with the aid of a pre-trained convolutional neural network.

Making the necessary predictions for the image and labelling the corresponding bounding box in accordance with those predictions is the R-final CNN's step. The predictions are created by computing a classification model for each task in order to get the best results possible. A regression model is then utilized to adjust the bounding box categorization for the suggested regions.

Limitations –

i. Despite the pre-trained CNN models generating good feature extraction results, the total extraction process of all the area recommendations, and ultimately the best regions using the present techniques, is incredibly sluggish.

ii. The R-CNN model's high prediction time, in addition to its sluggish training rate, is another significant flaw. The requirement for extensive computational resources raises the process’s overall viability. Because of this, the entire architecture might be seen as pricey.

iii. On occasion, the first step's candidate choices might go wrong because there aren't any room for change at that point. This has a lot of potential to cause issues with the trained model.

3. Faster R-CNN:

The R-CNN [5] model was able to compute object detection and produce desired results, however there were numerous significant flaws, particularly the model's speed. As a result, it was necessary to develop quicker approaches to some of these challenges in order to fix the flaws with R-CNN. First, the Fast R-CNN was developed to address some of the R-underlying CNN's problems.

Instead of taking into account all of the sub-segments, the quick R-CNN approach runs the entire picture through the pre-trained Convolutional Neural Network. A unique technique called region of interest (RoI) pooling uses two inputs—a pre-trained model and a selective search algorithm—to produce an output for a fully connected layer. We will learn more about the Faster R-CNN network, an advancement over the Fast R-CNN model, in this part. One of the best R-CNN [5] iterations, the Faster R-CNN model significantly increases performance speed over its forerunners. The region proposals are computed using a selective search algorithm in the R-CNN and Fast R-CNN models, while the Faster R-CNN [5] technique substitutes this current approach with a better region proposal network. To provide useful outputs, the region proposal network (RPN) computes pictures at a variety of sizes.

The local proposal network shortens the computation time for the margin, which is typically 10 ms per picture. The convolutional layer of this network is what allows us to extract the key feature mappings for each pixel. We have many anchor boxes with various scales, sizes, and aspect ratios for each feature map. We anticipate the specific binary class for each anchor box and then produce a bounding box for it. Since there are a lot of overlaps created during the feature map creation process, the resulting data is then run through the non-maximum suppression to remove any extraneous data. The region of interest is traversed by the output of the non-maximum suppression, and the remaining steps and calculation are carried out similarly to how Fast R-CNN operates.

Limitations – The degree of time delay in the suggestion of various items is one of the Faster R-CNN method's primary drawbacks. The speed can occasionally vary depending on the system being utilized.

4. Single Shot Detection (SSD):

One of the quickest methods to complete object identification jobs in real time computing is the single-shot detector for multi-box predictions. While the Faster R-CNN techniques are capable of producing predictions with high accuracy, the total process is time-consuming and necessitates a real-time job that runs at a rate of less than 7 frames per second, which is not ideal. This problem is resolved by the single-shot detector (SSD) [15], which increases the frames per second to nearly five times that of the Faster R-CNN model. It does away with the region proposal network and uses multi-scale features and default boxes instead.

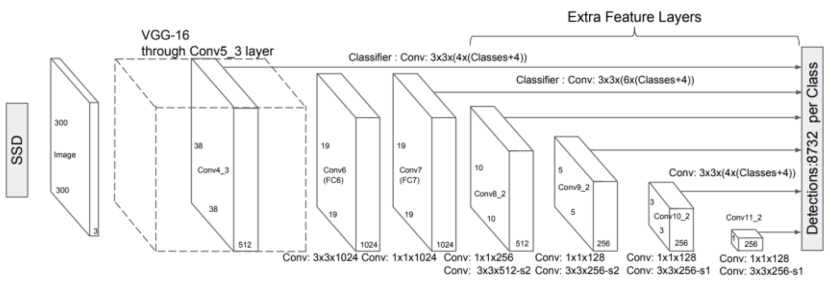


Fig. 4: Architecture of SSD Algorithm [8]

There are primarily three parts to the single shot multi-box detector design. The feature extraction phase, which is the initial step of the single-shot detector, is when all of the essential feature maps are chosen. There are no further layers in this architectural zone, just completely convolutional layers. The procedure of identifying heads comes next after extracting all of the necessary feature maps. Additionally, fully convolutional neural networks are used at this stage. Finding the semantic meaning of the pictures is not the problem at hand in the second step of the detecting heads, though. Instead, the main objective is to provide the best bounding maps possible for all of the feature maps. The next step is to feed it through the non-maximum suppression layers after computing the two crucial steps to lower the error rate brought on by repeated bounding boxes.

Limitations –

i. The SSD [8] suffers from lowering the resolution of the photos to a poorer quality, albeit greatly improving performance.

ii. For small-scale objects, the SSD design will often perform worse than the Faster R-CNN [5].

5. You Only Look Once (YOLO):

One of the most well-liked model architectures and object identification algorithms is YOLO [9,12,17, 18]. The YOLO architecture is frequently the first idea returned in a Google search for object detection algorithms. There are other YOLO variations, which we shall go through in the next parts. The YOLO model achieves great accuracy and processing speed by utilizing one of the finest neural network archetypes. The key factors influencing its appeal are its quickness and precision.

The YOLO [15] architecture achieves object detection with the use of three main terminologies. To fully understand why this model performs so rapidly and precisely in comparison to other object identification algorithms, it is important to comprehend these three strategies. Residual blocks are the initial notion in the YOLO model. In the first architectural design, grids were made in the specific picture using remaining 7 × 7 blocks.

Each of these grids serves as a center point, and a specific forecast is created for each grid in accordance with that. In the second method, the bounding boxes are created by taking into account each of the central points for a certain forecast. Although the classification tasks are successful for each grid, it is more difficult to separate the bounding boxes for each forecast. The intersection of union (IOU), which is used to determine the optimal bounding boxes for the specific object identification job, is the third and final strategy.

Advantages of YOLO –

i. When compared to most training techniques and object detection algorithms, YOLO has a calculation and processing speed that is relatively fast, especially in real-time.

ii. In addition to having a quick computation time, the YOLO technique also achieves a high level of accuracy overall by reducing background mistakes compared to other approaches.

iii. The YOLO design makes it possible for the model to learn and comprehend various items more quickly.

Constraints of YOLO –

i. Due to the decreased recall rate, tiny objects in an image or video are not detected.

ii. Due to the restrictions of bounding boxes, it is impossible to identify two items that are extremely near to one another.

Versions of YOLO –

One of the most well-known and successful object detection algorithms is the YOLO [11] architecture. After the YOLO architecture was released in 2016, its subsequent iterations, YOLO v2 [14] and YOLO v3 [36], debuted in 2017 and 2018. In contrast to 2019, which saw no new releases, 2020 had three swift releases: PP-YOLO, YOLO v4 [15], and YOLO v5 [16]. Each iteration of YOLO made a small advancement over its predecessor. To guarantee that object detection could be supported on embedded devices, the small YOLO was also published. YOLO published its version 6 [12] in October 2021, and version 7 [13] on June 2nd, 2022. The quickest and most precise object identification system created to date is YOLOv7 [13].

III. Proposed Method:

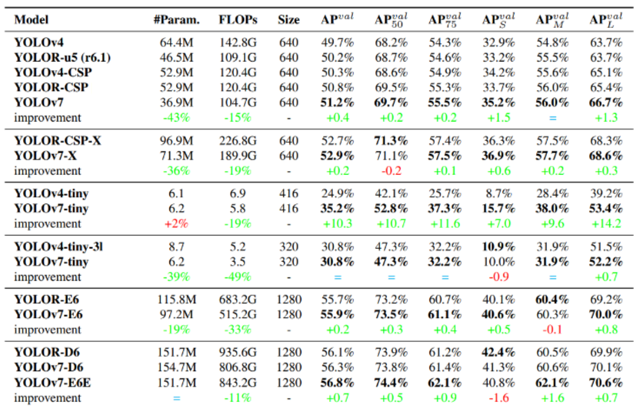
The proposed method uses custom trained YOLOv7 [13] model to detect potholes. Like the other modules, it takes input from the dashcam and passes the frames through the trained model. If any pothole is detected in any frame, and its confidence value passes the confidence threshold, then the pothole is located and identified, and an alert is show to the driver.

The mathematical representation for the function of detection of any pothole is as follows:

$$f(n) = \begin{cases} 0, & if\ confidence < confidence\ threshold \\ 1, & if\ confidence \geq confidence\ threshold \end{cases}$$

Where, f(n) is the deterministic function for identifying a pothole in a frame, *confidence* is the similarity of the detected object with the objects registered in custom trained model, and *confidence threshold* is the threshold for counting the detected pothole as a true pothole. This component predicts the detection of any pothole based on the binary decision made by the deterministic function.

The reason behind choosing YOLO v7 [13] as the main framework for our proposed solution is because YOLOv7 infers faster and with greater accuracy than its previous versions (i.e., YOLOv6, YOLOv5, etc.), pushing the state of the art in object detection to new heights.



Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1)	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Fig. 5: Comparison of all YOLO versions [13]

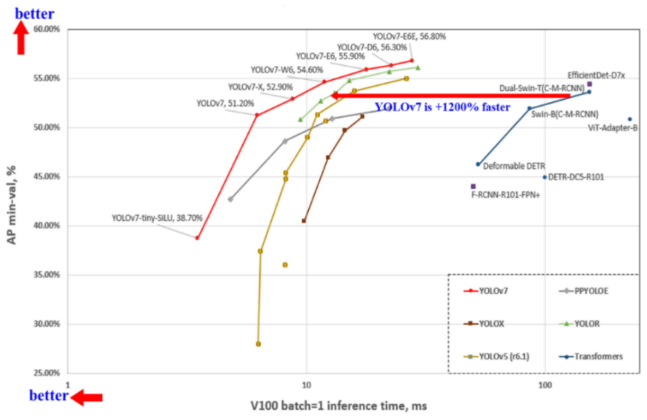


Fig. 6: The evaluation of YOLOv7 models show that they infer faster (x-axis) and with greater accuracy (y-axis) than comparable real-time object detection models. [13]

ALGORITHM FOR POTHOLE DETECTION

INPUT	:	Live video feed from dashcam.
OUTPUT	:	Warnings of pothole if detected.

ALGORITHM:

```
STEP – 1:   Start
STEP – 2:   Define confidence_threshold
STEP – 3:   Load YOLOv7 Model (custom-trained)
STEP – 4:   Get frame
STEP – 5:   Pass the frame through the model loaded in STEP – 3.
STEP – 6:   For each detected potholes in the frame, repeat,
            a.  If the confidence of detected pothole is greater than or equal to defined confidence_threshold in STEP-2,
                then,
                    i. Count the pothole as a real pothole and show warning
            b.  Else, go to STEP - 4
STEP – 7:   Stop
```

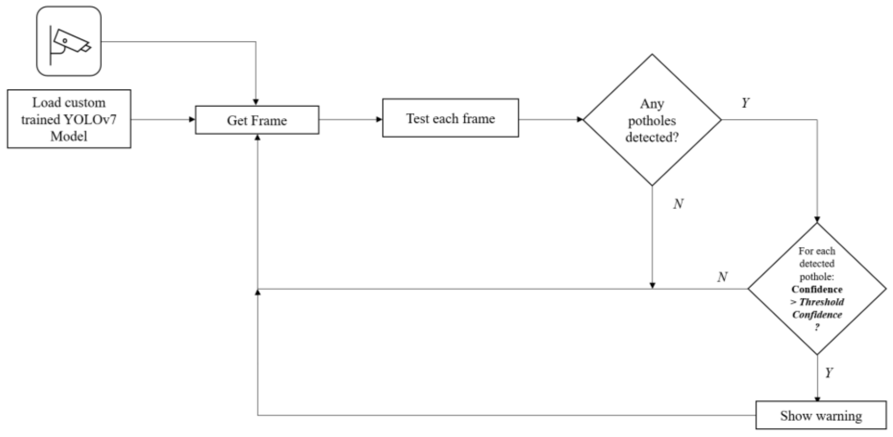

Load custom trained YOLOv7 Model
Get Frame
Test each frame
Any potholes detected?
Y
N
N
For each detected pothole: Confidence > Threshold Confidence ?
Y
Show warning

Fig. 7: Flow diagram of Pothole Detection System

Fig. 8 describes the all the steps that have to be done sequentially in order to design, develop, implement and test the proposed solution. The proposed solution has been divided into three main stages which are as follow:

1. Data Collection and Preprocessing:

   All the relevant data are collected from different sources and cleaned to produce an almost error free dataset. After data cleaning, all the images are annotated with relevant labels.

2. Model Training and Testing:

   The Machine Learning model is developed with YOLO v7 architecture, and the model is tested using inference.

3. Prediction in real-time:

   The final result is saved and is tested in real-time and real-life scenarios.

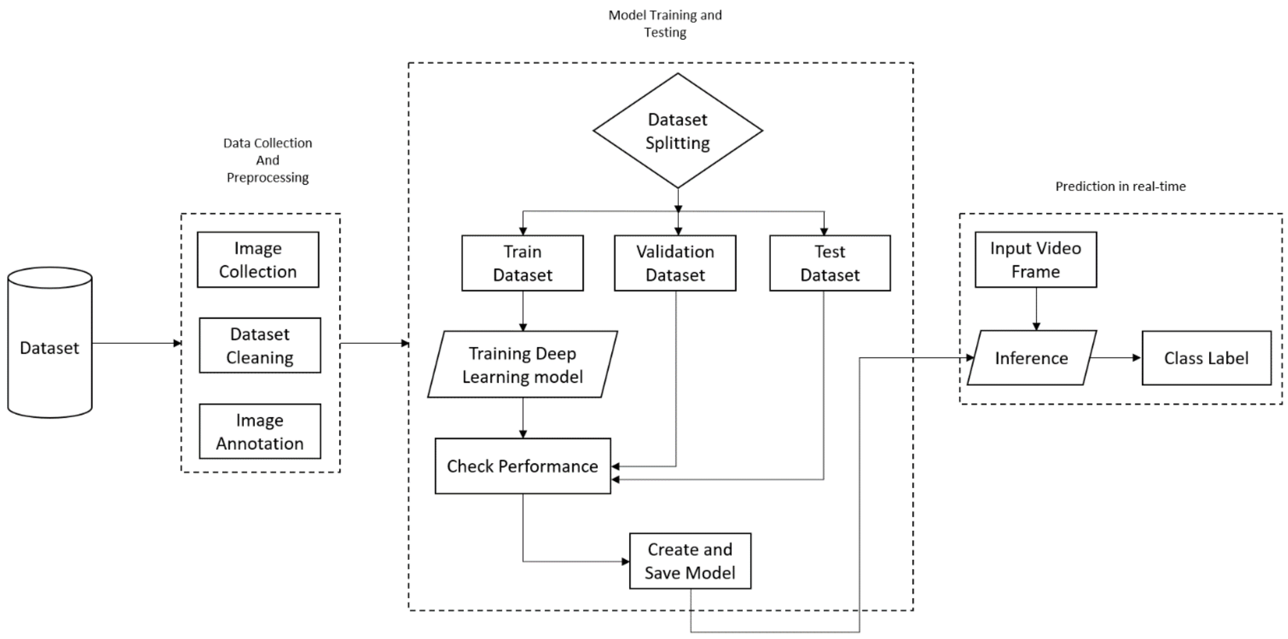


Fig. 8: Proposed methodology block diagram of real-time pothole detection.

IV. Implementation

1. Framework Used

You Only Look Once (YOLO) version 7 has been used as the main object detection framework in the implementation of the proposed solution. YOLO is trained using an opensource neural network framework which is fast due to CUDA and C language providing the real-time attribute for our detection.

2. Dataset Description

A custom dataset has been developed by combining the images of Roboflow Pothole Dataset and images captured by the authors which contains a wide variety of images of potholes, plain roads, and different shapes and sizes of craters.

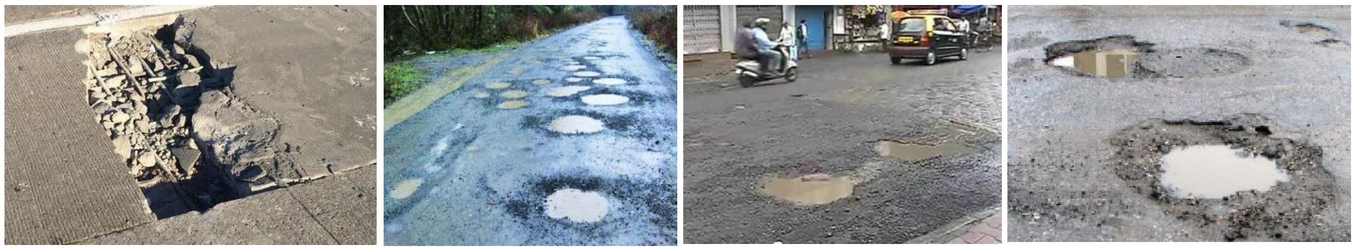


Fig. 9: Some images from the custom dataset

The whole dataset has been divided into three segments: train, test and validation. The training dataset contains 1265 images which will be used to train and develop the model, the validation dataset contains 401 images which will be used to validate our model performance during training, and the test dataset contains 118 images to test the model post training. The dataset contains images captured in day and night scenario.

3. Data Cleaning

After collection of the dataset, any irrelevant or unwanted images, images with low resolution, duplicate images are removed from the dataset by manually removing images for any unwanted error from the training using the dataset.

4. Image Annotation

After the cleaned dataset has been generated, all the images of the train and validation dataset have been annotated manually. The annotation and labeling work has been done using LabelImg framework, which is an open source Python library based on Graphical User Interface. LabelImg is easy to use, and it directly produces the image annotated label data as YOLO format in a text file named same as the corresponding annotated image. For training the YOLO model, the annotations should be in YOLO format as <class_id> <center_x> <center_y> <width> <height>, where class_id is an integer value starting from 0 up to the number of classes defined; in our case, the object class will be 0 as we have only one class, i.e., pothole and the remaining parameters are the coordinates, height, and width of the labeled object bounding box.

5. Experimentation Protocols

The training of the YOLO model on the custom dataset has been carried out on Google Colab system having Intel Xeon CPU at 2.30 GHz wit h2 cores, 12 GB of memory and Nvidia Tesla T4 GPU with 16 GB of video memory and 1250 MHz of memory clock and a GPU base clock of 585 MHz. The model has been trained with 100 epochs, a batch size of 32 and a learning rate of 0.001. The filters are set to 18, as filter size = (class + 5) * 3, since, we have only 1 class. The model as taken 2-hours 52-minutes approximately to train using the dataset.

V. EXPERIMENTAL RESULTS

The performance of the model has been evaluated with performance metrics (mAP, Precision, Recall, F1 Score and mean inference time per image) using the following mathematical expressions:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

$$Average\ Precision = \sum_{n=1}^{N}(R_n - R_{n-1})P_n \quad (4)$$

$$Mean\ Average\ Precision\ (mAP) = \frac{1}{N}\sum_{n=1}^{N} AP_i \quad (5)$$

The model is assessed by setting a threshold and checking the precision and recall values. For the calculations of precision and recall, N thresholds are assumed, each of which consists of a pair of precision (Pn) and recall (Rn) values (n 1, 2, ..., N). Equation (4) defines average precision (AP), while equation (1) defines mean average precision (mAP), which is the average of AP across all classes. Since we only have one class, the AP and mAP will be the same in our instance. The amount of overlap between the predicted bounding box and the actual bounding box is measured and compared to the Intersection over Union criterion (IOU). For this instance, we have devised the threshold to 0.4. Therefore, a prediction is correct if IOU score is greater than or equal to the threshold of 0.4 (40%). The IoU is calculated using the following formula:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (6)$$

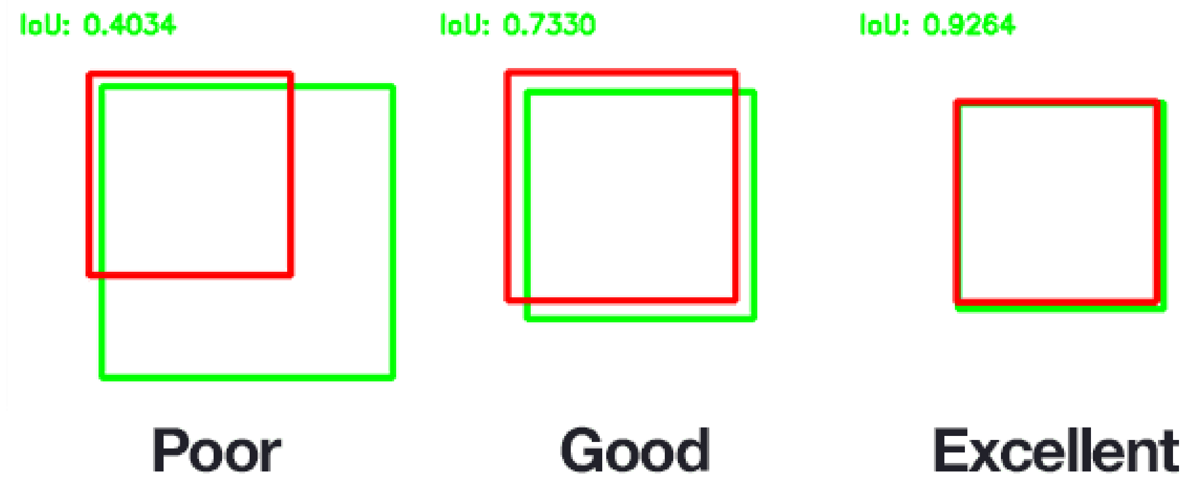


Fig. 10: Different of types of Intersection over Unions, where based on the IoU value, Poor, Good, Excellent – these three categories have been devised

After training the model, the model has achieved a PR (precision) of 0.94, Recall value of 0.98. The F1 Score achieved is 0.96.

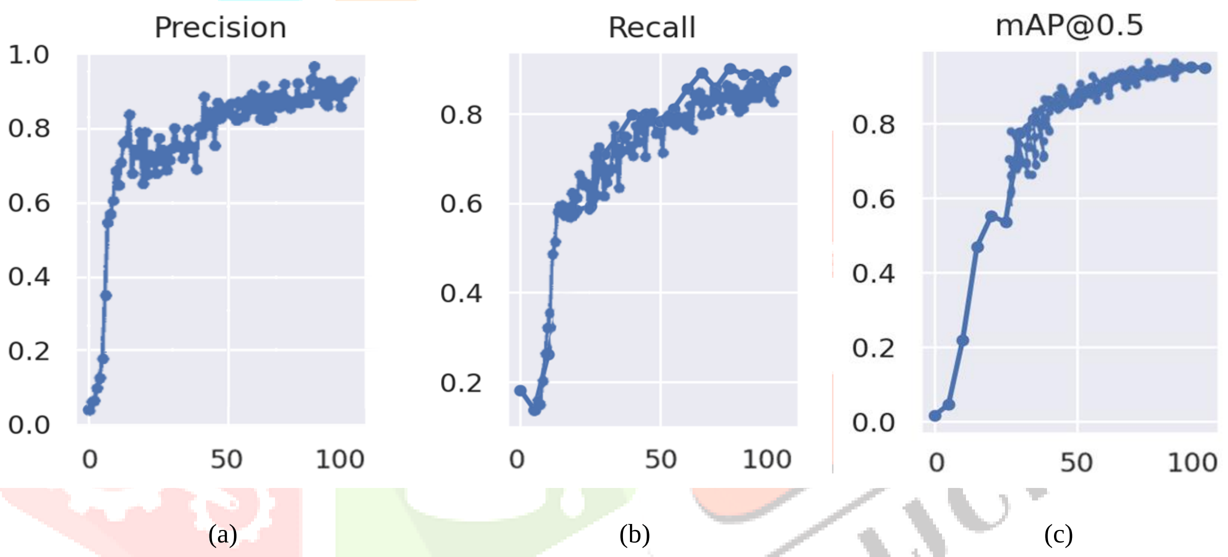


Fig. 11: (a) Precision graph, (b) Recall Graph, (c) mAP Graph obtained after 100 epochs

The mean average precision of the training has been finally calculated 94.76% as shown in Fig. 11(c).
The average detection accuracy of the developed model is calculated with the help of the following mathematical expression:

$$Avg.Detection = \frac{Detected\ Potholes}{Total\ Potholes} \times 100 \qquad (7)$$

Table 2: Real-time pothole detection performance analysis calculated using (7)

Scenario	Number of Potholes	*Detected Potholes*	*Accuracy*	*FPS*
Day	23	23	100%	27.9
Night	20	18	90%	21.65

Several inferenced frames from videos captured for real-time performance analysis has been shown in Fig. 12. It has been found out that the developed model has been performing with good accuracy in both daylit and night scenarios, though some false positive cases are coming up in night and low light environment, the quantity of which is negligible.

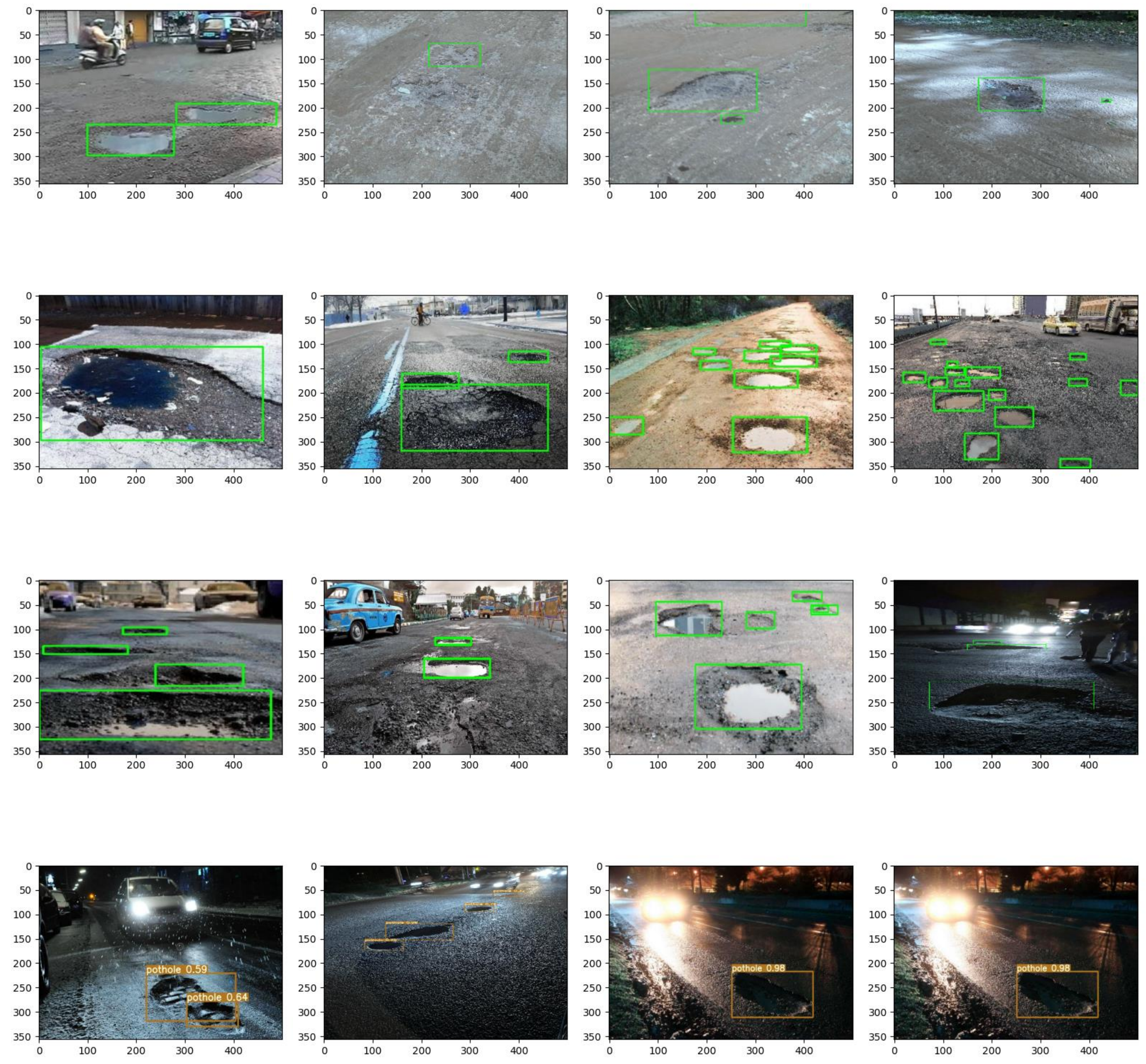


Fig. 12: Inferenced images using the developed model

V. Conclusion And Future Scope

The work presented in this paper comes up with the conclusion that, the YOLO v7 is the best fit for detecting potholes in images and as well as real-time scenarios. The proposed solution has been found working with an average accuracy of almost 95%. Although, there is many scopes of improvements in the solution. As it has been found out that the proposed solution is unable to detect some of the potholes in low-light environments or in night. So, the volume of the dataset can be increased by adding more night and low-light pothole images and train it later on with a greater number of epochs. New CNN and Deep Learning architectures can be used to implement and train the proposed solution and analyse the performance. Depth sensors and Stereo Vision Cameras can be used to perceive the environment for better accuracy which will be able to detect potholes more accurately. An ecosystem of detection and alerting process can be developed by storing the Geo Locations of the potholes detected by the proposed solution, with the help of GPS, and the locations can be stored on a cloud database, which can be used to alert any vehicle coming within the proximity of that particular pothole by constantly comparing the locations of the pothole and the vehicle. The proposed solution can be used to scan and collect road damage data, which can help Governments to automate the collection of all the road damage information, reducing human effort.

VI. References

[1] Rastogi, Roopak & Kumar, Uttam & Kashyap, Archit & Jindal, Shubham & Pahwa, Saurabh. (2020). A Comparative Evaluation of the Deep Learning Algorithms for Pothole Detection. 1-6. 10.1109/INDICON49873.2020.9342558.

[2] John, Anand & Meva, Dr. Divyakant. (2020). A Comparative Study of Various Object Detection Algorithms and Performance Analysis. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. 8. 158-163. 10.26438/ijcse/v8i10.158163.

[3] Suleiman, Amr & Sze, Vivienne. (2014). Energy-Efficient HOG-based Object Detection at 1080HD 60 fps with Multi-Scale Support. 10.13140/2.1.4752.0964. s

[4] Ren, Haoyu & Li, Ze-Nian. (2015). Object detection using edge histogram of oriented gradient. 2014 IEEE International Conference on Image Processing, ICIP 2014. 4057-4061. 10.1109/ICIP.2014.7025824.

[5] Liu, Bin & Zhao, Wencang & Sun, Qiaoqiao. (2017). Study of object detection based on Faster R-CNN. 6233-6236. 10.1109/CAC.2017.8243900

[6] Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

[7] Jia, Songmin & Diao, Chentao & Zhang, Guoliang & Dun, Ao & Sun, Yanjun & Li, Xiuzhi & Zhang, Xiangyin. (2019). Object Detection Based on the Improved Single Shot MultiBox Detector. Journal of Physics: Conference Series. 1187. 042041. 10.1088/1742-6596/1187/4/042041.

[8] Rohan, Ali & Rabah, Mohamed & Kim, Seunghae. (2019). Convolutional Neural Network-Based Real-Time Object Detection and Tracking for Parrot AR Drone 2. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2919332.

[9] Atik, Muhammed Enes & Duran, Z. & ÖZGÜNLÜK, Roni. (2022). Comparison of YOLO Versions for Object Detection from Aerial Images. International Journal of Environment and Geoinformatics. 9. 87-93. 10.30897/ijegeo.1010741.

[10] Adarsh, Pranav & Rathi, Pratibha & Kumar, Manoj. (2020). YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. 687-694. 10.1109/ICACCS48705.2020.9074315.

[11] Jiang, Peiyuan & Ergu, Daji & Liu, Fangyao & Ying, Cai & Ma, Bo. (2022). A Review of Yolo Algorithm Developments. Procedia Computer Science. 199. 1066-1073. 10.1016/j.procs.2022.01.135.

[12] Li, Chuyi & Li, Lulu & Jiang, Hongliang & Weng, Kaiheng & Geng, Yifei & Li, Liang & Ke, Zaidan & Li, Qingyuan & Cheng, Meng & Nie, Weiqiang & Li, Yiduo & Zhang, Bo & Liang, Yufei & Zhou, Linyuan & Xu, Xiaoming & Chu, Xiangxiang & Wei, Xiaoming & Wei, Xiaolin. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. 10.48550/arXiv.2209.02976.

[13] Wang, Chien-Yao & Bochkovskiy, Alexey & Liao, Hong-yuan. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 10.48550/arXiv.2207.02696.

[14] Xiaohong Han, Jun Chang, Kaiyuan Wang, Real-time object detection based on YOLO-v2 for tiny vehicle object, Procedia Computer Science, Volume 183,2021 Pages 61-72, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.02.031

[15] Bochkovskiy, Alexey & Wang, Chien-Yao & Liao, Hong-yuan. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[16] M. Karthi, V. Muthulakshmi, R. Priscilla, P. Praveen and K. Vanisri, "Evolution of YOLO-V5 Algorithm for Object Detection: Automated Detection of Library Books and Performace validation of Dataset," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2021, pp. 1-6, doi: 10.1109/ICSES52305.2021.9633834.

[17] L. Huidrom, L. K. Das, and S. Sud, "Method for automated assessment of potholes, cracks and patches from road surface video clips," Procedia-Social and Behavioral Sciences, vol. 104, pp. 312–321, 2013.

[18] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," Advanced Engineering Informatics, vol. 25, no. 3, pp. 507–515, 2011.

[19] R. Agrawal, Y. Chhadva, S. Addagarla, and S. Chaudhari, "Road surface classification and subsequent pothole detection using deep learning," in Proceedings of the 2021 2nd International Conference for Emerging Technology (INCET), pp. 1–6, IEEE, Belagavi, India, May 2021.

[20] C. Zhang, E. Nateghinia, L. F. Miranda-Moreno, and L. Sun, "Pavement distress detection using convolutional neural network (cnn): a case study in montreal, Canada," International Journal of Transportation Science and Technology, vol. 1, 2021.

[21] S. I. Hassan, D. O'Sullivan, and S. McKeever, "Pothole detection under diverse conditions using object detection models," IMPROVE, vol. 1, pp. 128–136, 2021.

[22] R. Kavitha and S. Nivetha, "Pothole and object detection for an autonomous vehicle using yolo," in Proceedings of the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1585–1589, IEEE, Madurai, India, May 2021.

[23] H. Chen, M. Yao, and Q. Gu, "Pothole detection using location-aware convolutional neural networks," International Journal of Machine Learning and Cybernetics, vol. 11, no. 4, pp. 899–911, 2020.

[24] M. Rani, M. Mustafar, N. Ismail, M. Mansor, and Z. Zainuddin, "Road peculiarities detection using deep learning for vehicle vision system IOP Conference Series: materials Science and Engineering," IOP Publishing, vol. 1068, no. 1, p. 012001, 2021.

[25] M. Omar and P. Kumar, "Detection of roads potholes using yolov4," in Proceedings of the 2020 International Conference on Information Science and Communications Technologies (ICISCT), pp. 1–6, IEEE, Tashkent, Uzbekistan, November 2020.

[26] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, "Real-time pothole detection using deep learning," 2021, https://arxiv. org/abs/2107.06356.

[27] K. Gajjar, T. van Niekerk, T. Wilm, and P. Mercorelli, Visionbased Deep Learning Algorithm for Detecting Potholes, 2021.

[28] S.-S. Park, V.-T. Tran, and D.-E. Lee, “Application of various yolo models for computer vision-based real-time pothole detection,” Applied Sciences, vol. 11, no. 23, p. 11229, 2021.
[29] https://timesofindia.indiatimes.com/city/bhubaneswar/road-accident-deaths-up-by-7-in-first-5-months-of 2022/articleshow/93062935.cms (Last accessed on December 22, 2022, at 3.40 PM)
[30] https://www.wbtrafficpolice.com/acccidenta-during-last-7-days.php (Last accessed on November 28, 2022, at 6.40 PM)
[31] Asad, Muhammad & Khaliq, Saran & Yousaf, Muhammad Haroon & Ullah, Muhammad & Ahmad, Afaq. (2022). Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective. Advances in Civil Engineering. 2022. 10.1155/2022/9221211.
[32] C. Zhang, E. Nateghinia, L. F. Miranda-Moreno, and L. Sun, “Pavement distress detection using convolutional neural network (cnn): a case study in montreal, Canada,” International Journal of Transportation Science and Technology, vol. 1, 2021.
[33] O. A. Egaji, G. Evans, M. G. Griffiths, and G. Islas, “Real-time machine learning-based approach for pothole detection,” Expert Systems with Applications, vol. 184, p. 115562, 2021
[34] Y. K. Arbawa, F. Utaminingrum, and E. Setiawan, “Three combination value of extraction features on glcm for detecting pothole and asphalt road,” Jurnal Tekn
[35] H. Chen, M. Yao, and Q. Gu, “Pothole detection using location-aware convolutional neural networks,” International Journal of Machine.
[36] Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement.