**JOIN (Jurnal Online Informatika)**
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 10 Number 1 | June 2025: 122-131
DOI: 10.15575/join.v10i1.1481

# Road Damage Detection Using YOLOv7 with Cluster Weighted Distance-IoU NMS

**Rudy Rachman[1], Nanik Suciati[2], Shintami Chusnul Hidayati[3]**
[1,2,3]Department of Informatic Engineering, Sepuluh Nopember Institute of Technology, Indonesia

**Article Info**

***Article history:***

Received October 10, 2024
Revised December 18, 2024
Accepted February 01, 2025
Published April 01, 2025

***Keywords:***

Cluster NMS
NMS
Object detection
Pothole
YOLOv7

**ABSTRACT**

Road damage can occur everywhere. Potholes are one of the most common types of road damage. Previous research that used images as input for pothole detection used the Faster Regional Convolutional Neural Network (R-CNN) method. It has a large inference time because it is a two-stage detection method. The object detection method requires post-processing for its detection results to save only the best prediction from the method, namely, non-maximum suppression (NMS). However, the original NMS could not properly detect small, far, and two objects close to each other. Therefore, this research uses the YOLOv7 method as the object detection method because it has better mean Average Precision (mAP) results and a lower inference time than other object detection methods; with an improved NMS method, namely Cluster Weighted Distance Intersection over Union (DIoU) NMS (CWD-NMS), to solve small or close potholes. When training YOLOv7, we combined a new, independently collected pothole dataset, with previous public research datasets, where the detection results of the YOLOv7 method were better than those of Faster R-CNN. The YOLOv7 method was trained using various scenarios. The best scenario during training is using the best checkpoint without using a scheduler. The mAP.5 and mAP.5-.95 value of CWD-NMS was 89.20% and 63.30% with 10.30 millisecond per image for inference time.

***Corresponding Author:***

Rudy Rachman
Informatic Engineering Department, Faculty of Intelligent Electrical and Informatics Technology, Sepuluh Nopember Institute of Technology, Indonesia
Jl. Teknik Kimia, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur, Indonesia. 60111
Email: rudyrachman16@gmail.com

## 1. INTRODUCTION

Road damage can occur at any road type. The main factors of road damage are rainwater, changes in temperature and weather, bad construction materials on the road, and the overlimit weight of vehicles passing on the road, etc., [1]. Potholes are one of the most common types of road damage. Potholes can disturb drivers and even cause accidents. Potholes that are not immediately detected and covered will cause more serious damage.

Many researchers are starting to search for the best method for detecting potholes [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. Previous research has used several media, such as images and vehicle sensors. However, detection using media other than images is difficult to adapt to daily life and is quite expensive. Research that uses image processing also encounters several problems if the image quality is poor (pothole covered by shadow or by another object).

Therefore, to solve this problem, several previous research have employed deep learning. A common deep learning method for detection is the Convolutional Neural Network (CNN) [13], [14], [15],

[16], [17]. The CNN can perform feature extraction even though the quality of the acquired image is poor; however, it can still detect potholes correctly. In previous research, research used Faster Regional CNN (R-CNN) to detect potholes and achieved a mean Average Precision (mAP) value of 79.70% [11]. However, Faster R-CNN is a two-stage detection method. Two-stage detection is slower than one-stage detection in real-time detection. Therefore, this research will use one of the state-of-the-art (SOTA) detection methods, namely You Only Look Once (YOLO) [18].

Several previous research have used Yolo version five (YoloV5) [16], [19], [20], [21], [22]. YoloV5 was also a SOTA method before YOLOv7 was proposed [17]. YOLOv7 provides faster inference speed and better mAP results than YoloV5. Therefore, in this research, YOLOv7 was used for pothole detection using image media.

The Yolo method requires post-processing for its detection results. The commonly used post-processing method is non-maximum suppression (NMS) [16], [17], [19], [21], [23]. NMS is used to delete detection results that have already been detected and only save the best detection results. The best detection results were obtained based on the detection probability value and Intersection over Union (IoU) value. However, NMS is still unable to detect small and close objects. This problem can be solved using another NMS method, namely Distance-IoU NMS (DIoU-NMS) [16], [19], [20], [21], [22]. Previous research used YoloV5 with the DIoU-NMS method, and the mAP results were better than those of the original NMS.

The DIoU-NMS also has new improvements by combining cluster NMS and weighted NMS [23]. The new combined method is called Cluster Weighted DIoU-NMS (CWD-NMS). The prediction results of CWD-NMS are better than those of other improved NMS methods [23]. In aerial object detection using the improved YoloV5 and CWD-NMS methods, the mAP results were also better than those of other methods [22].

In summary, our research makes three major contributions. First, we used the YOLOv7 method with CWD-NMS to better detect potholes. YOLOv7 is used in this study, because the results of previous studies using CWD-NMS, only used YoloV5. The implementation of YOLOv7 or YoloV9 is still unavailable with CWD-NMS. Second, the best training scenario, which does not use a scheduler, comes from the best checkpoint during training. Finally, we used a new pothole dataset collected independently to complement the previous research's dataset.

## 2. METHOD

This section presents the discussions on previous research. Their research is the theoretical basis of this research. The Faster R-CNN is a two-stage object detection. It can detect many objects, for example, potholes, small objects on the roadside, etc., [11], [14]. In previous research, Faster R-CNN achieved a mAP value of 79.90% to detect potholes with MobileNetV2 as the backbone. The dataset used in that research is available on Kaggle and contains 665 images of potholes [11]. Their study also implements augmentation of the training data to increase variation while training Faster R-CNN.

In real-time object detection, Faster R-CNN is slower than one-stage object detection; one detection of an object can take approximately 200 milliseconds (ms) [13]. One of the best one-stage object detection methods is Yolo [18]. We used YOLOv7 because it achieves better mAP values and has a smaller inference time than other methods [17]. Their research used the Common Object in Context (COCO) evaluation dataset, and they obtained better results than another object detection method [24].

As described in the previous section, Yolo requires a post-processing method, namely NMS. NMS also has an improvement that improvement also makes NMS faster and gives a better result than NMS, namely C-NMS [23]. C-NMS can also be combined with DIoU-NMS to create another NMS method, namely CD-NMS. C-NMS and CD-NMS can also be combined with weighted NMS, and that method is CW-NMS and CWD-NMS. Based on their research results using a Single Shot Detector (SSD), Faster R-CNN, and YoloV3 for their object detection methods and for the dataset were COCO and Pascal VOC. CWD-NMS achieved a better result than another NMS method.

This section discusses how to acquire pothole images, image preprocessing, and an explanation of YOLOv7. It is divided into three parts: first data acquisition, second data preprocess, and last YOLOv7 configuration.

### *2.1. Data Acquisition*

Data will be used to train and evaluate the model. In this research is using three type of dataset, primary, secondary, and combination (primary and secondary). Primary data comprise 400 pothole

p-ISSN: 2528-1682
e-ISSN: 2527-9165

images. Primary data were collected by recording driving trips in Samarinda using a GoPro Hero 7 camera. The resolution of the camera is 1920 times 1080 pixels. The primary dataset is shown in **Error! Reference source not found.**. Pothole image acquired from screenshots of a video frame in which a pothole appears in the image. Pothole images from the primary image have no label. For labeling, we used Label Studio [25] with a bounding box (BB) template, and the label format was YOLO.

Figure 1. Image Example from Primary Dataset

The secondary data are available on Kaggle. The original Kaggle image contains some lousy images, which are discarded. The secondary data comprise 665 pothole images. All labels were set to potholes in the Pascal VOC format. The secondary dataset is shown in **Error! Reference source not found.**. The combination dataset is acquired by combining image data from the primary and secondary datasets. The total number of pothole images was 1065.

Figure 2. Image Example from Secondary Dataset

### *2.2. Data Pre-processing*

The first step in pre-processing an image is to resize the image to the same size. We used a value of 576 pixels as the image size. Two interpolation methods are used when resizing an image. The bicubic interpolation method is used if the area of the image is larger than the area of the target image, whereas the area interpolation method is used if the area of the image is smaller than the area of the target image [26].

The label format also should be in the same format. The label format used is the Yolo format. The equation for converting Pascal VOC to the Yolo format can be seen in the equation (1)-(4). $x$ and $y$ in equations (1) and (2) are the coordinates of the center point of the BB.

$$x = \frac{\left(\frac{x_1 + x_2}{2}\right)}{\text{image width}} \quad (1)$$

$$y = \frac{\left(\frac{y_1 + y_2}{2}\right)}{\text{image height}} \quad (2)$$

$$\text{BB width} = \frac{(x_2 - x_1)}{\text{image width}} \quad (3)$$

$$\text{BB height} = \frac{(y_2 - y_1)}{\text{image height}} \quad (4)$$

The next step was to split the dataset into training and testing sets. The secondary training data are already available in the split.json file from Kaggle. The split amount was 80% for training data and

20% for testing. The splitting of the train, test, and validation data in the primary dataset was performed using the train-test split function from the sklearn library [27] with the same amount as the secondary dataset. Validation data can be generated from half of (50%) from test data.

Finally, data augmentation is required to increase data variety. It is also good for preventing overfitting [28]. The method used is geometric augmentation that consists of flipping vertically, fliping horizontally, and doing 90° rotation to right and left. Results from augmentation method can create new seven images, which are shown in **Error! Reference source not found.**. The ground truth box point coordinates of the labels were also augmented so that the data train could be trained using the YOLOv7 method [11].

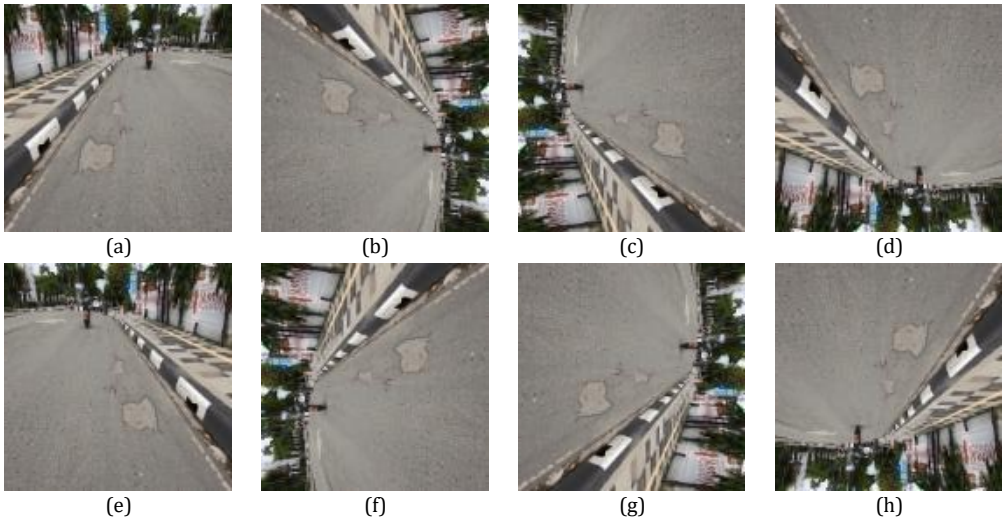


Figure 3. Example Image Results from Data Augmentation

**Error! Reference source not found.**(a) shows the original image, and the other images are the augmented images. **Error! Reference source not found.**(b) is the 90° rotation to right. **Error! Reference source not found.**(c) is the 90° rotation to left. **Error! Reference source not found.**(d) is flip vertical. **Error! Reference source not found.**(e) is flip horizontal. **Error! Reference source not found.**(f) is flip vertical and 90° rotation to right. **Error! Reference source not found.**(g) is flip vertical and 90° rotation to left. Finally, **Error! Reference source not found.**(h) shows a flip vertical and horizontal. Pothole image data after pre-processing for each part included 6816 images for training data, 107 images for validation data, and 106 images for test data.

### *2.3. Object Detection Method*

We used the Pytorch framework because YOLOv7 was built and configured in this framework; the version used in this research was 2.1.0 [29]. The pre-processed images and labels are placed into a folder according to the Yolo format so that they can be trained with the method, then create new .yaml file, so YOLOv7 can train, validate, and evaluate with a custom dataset. In the training phase, the training data need to be shuffled so that the method can learn and detect potholes correctly. The multi-scale method was also used, and thus, YOLOv7 was more robust because it was trained with different image sizes in one epoch.

Hyperparameter values in YOLOv7 need to be configured. Examples of hyperparameter values in YOLOv7 are in the optimizer, where there are learning rate (LR) and momentum; and in post-processing, there is an IoU threshold value in the NMS.

The training process begins after setting the hyperparameter values. Image data features are extracted using the YOLOv7 method with its backbone, and then features are sent to the head to facilitate detection. The detection results from the head should be processed again because the result still detects the same object twice with different prediction probability values. This problem can be solved using NMS. Our research flowchart is shown in Figure 4.

p-ISSN: 2528-1682
e-ISSN: 2527-9165

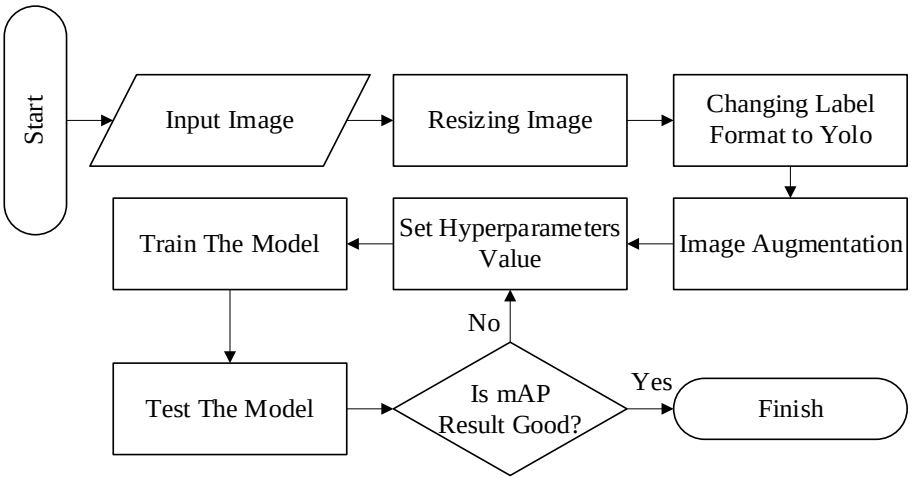


Figure 4. Research Flowchart

## 3. RESULT AND DISCUSSION

This session will discuss the results obtained by the YOLOv7 detection method according to the metrics discussed in section **Error! Reference source not found.**.

### *3.1. Evaluation Metrics*

The evaluation metrics used to measure the capabilities of the tested methods were precision, recall, mAP, F1-score, and NMS inference time (NMS). The NMS methods that will be used for comparison are original NMS (Torchvision NMS), C-NMS, CD-NMS, and CWD-NMS. The secondary dataset test data were used as a benchmark for comparison with Faster R-CNN [11]. Three YOLOv7 methods with training data from different datasets were evaluated: primary training data only, secondary training data only, and combined training data. The evaluation also considers the use of schedulers in the YOLOv7 method: training using a one-cycle LR scheduler, a linear LR scheduler, and no LR scheduler. When the method training process is complete, it creates two method checkpoints, the best and last. The results of these two checkpoints will also be evaluated for their metric values.

### *3.2. Implementation Details*

The GPU used in this research was an Nvidia RTX 3060 12 GB. The split amounts for the primary, secondary, or combination dataset is 80.00% for training, 10.00% for testing, and 10.00% for validation.

The hyperparameter values were fixed in this study are fixed (except LR which uses a scheduler). The optimizer method for training the YOLOv7 method was the Stochastic Gradient Descent (SGD). The momentum value was set to 0.80. The initial LR value was set at $9.20 \times 10^{-4}$ [11]. The LR target value when using the scheduler was set to the initial YOLOv7 setting value (0.10).

The maximum number of training epochs in this research was 100 epochs. We also compared two method checkpoints, the best mAP result during training and the last training epoch. During the training process, overfitting did not occur; thus, an early stopping was not required.

### *3.3. Discussion*

The training results of the YOLOv7 method under various scenarios were compared based on the evaluation metrics mentioned in the previous section. The YOLOv7 method was compared to various scenarios, such as the NMS, checkpoint (C.P.), and scheduler methods.

First, a comparison of the YOLOv7 method trained on only the primary dataset. The results of training in various scenarios are shown in Table 1.

Table 1. YOLOv7 Training Results with Only Primary Dataset

| NMS | C.P. | Scheduler | Precision (%) | Recall (%) | mAP .5 (%) | mAP 0.5:0.95 (%) | F1-Score (%) | Speed (ms) |
|---|---|---|---|---|---|---|---|---|
| Torchvision | Best | One-cycle | 46.30 | 27.20 | 28.30 | 16.10 | 34.30 | 5.10 |
| | Last | | 46.60 | 48.70 | 42.60 | 23.90 | 47.60 | 5.00 |
| | Best | Linear | 50.30 | 36.40 | 33.90 | 18.20 | 42.20 | 5.10 |
| | Last | | 70.40 | 31.80 | 40.70 | 23.00 | 43.80 | 7.00 |
| | Best | None | 50.70 | 36.90 | 36.10 | 19.70 | 42.70 | 8.00 |
| | Last | | 51.50 | 41.00 | 39.20 | 21.90 | 45.70 | 7.00 |
| C-NMS | Best | One-cycle | 46.30 | 27.20 | 28.30 | 16.10 | 34.30 | 7.00 |
| | Last | | 46.60 | 48.70 | 42.60 | 23.90 | 47.60 | 6.30 |
| | Best | Linear | 50.30 | 36.40 | 33.90 | 18.30 | 42.20 | 6.90 |
| | Last | | 70.40 | 31.80 | 40.70 | 23.00 | 43.80 | 5.60 |
| | Best | None | 50.70 | 36.90 | 36.10 | 19.70 | 42.70 | 6.10 |
| | Last | | 51.50 | 41.00 | 39.20 | 21.90 | 45.70 | 5.80 |
| CD-NMS | Best | One-cycle | 46.30 | 27.20 | 28.20 | 16.10 | 34.30 | 7.60 |
| | Last | | 46.60 | 48.70 | 42.60 | 23.80 | 47.60 | 6.00 |
| | Best | Linear | 50.30 | 36.40 | 33.90 | 18.20 | 42.20 | 6.70 |
| | Last | | 70.40 | 31.80 | 40.70 | 23.00 | 43.80 | 6.60 |
| | Best | None | 50.70 | 36.90 | 36.10 | 19.70 | 42.70 | 7.60 |
| | Last | | 51.50 | 41.00 | 39.20 | 21.80 | 45.70 | 6.00 |
| CW-NMS | Best | One-cycle | 45.50 | 26.70 | 28.20 | 16.20 | 33.60 | 8.20 |
| | Last | | 46.70 | 48.10 | 42.40 | 24.10 | 47.40 | 6.10 |
| | Best | Linear | 50.30 | 36.40 | 33.60 | 17.90 | 42.20 | 8.60 |
| | Last | | 70.40 | 31.80 | 40.80 | 23.10 | 43.80 | 6.90 |
| | Best | None | 50.70 | 36.90 | 36.20 | 20.20 | 42.70 | 7.30 |
| | Last | | 52.20 | 41.50 | 39.10 | 22.00 | 46.20 | 9.40 |
| CWD-NMS | Best | One-cycle | 45.50 | 26.70 | 28.10 | 16.20 | 33.60 | 9.60 |
| | Last | | 46.70 | 48.10 | 42.40 | 24.00 | 47.40 | 7.50 |
| | Best | Linear | 49.60 | 35.90 | 33.50 | 18.00 | 41.60 | 11.00 |
| | Last | | 70.40 | 31.80 | 40.80 | 23.10 | 43.80 | 10.40 |
| | Best | None | 50.70 | 36.90 | 36.00 | 20.20 | 42.70 | 11.90 |
| | Last | | 52.20 | 41.50 | 39.10 | 22.00 | 46.20 | 9.10 |

The training results of the YOLOv7 method using the primary dataset alone could not detect potholes originating from the secondary dataset (mAP value below 50.00%). This occurs because the images from the primary dataset were taken at a fixed angle, whereas the images from the secondary dataset are at different angles, this can be seen in **Error! Reference source not found.** and **Error! Reference source not found.**. Second, a comparison of YOLOv7 methods trained with only the secondary dataset. The results of training in various scenarios are presented in Table 2.

Table 2. YOLOv7 Training Results with Only Secondary Dataset

| NMS | C.P. | Scheduler | Precision (%) | Recall (%) | mAP .5 (%) | mAP 0.5:0.95 (%) | F1-Score (%) | Speed (ms) |
|---|---|---|---|---|---|---|---|---|
| Torchvision | Best | One-cycle | 84.90 | 80.50 | 85.40 | 61.00 | 82.60 | 6.40 |
| | Last | | 85.30 | 80.50 | 85.70 | 60.20 | 82.80 | 6.10 |
| | Best | Linear | 85.90 | 81.50 | 87.30 | 63.20 | 83.70 | 6.10 |
| | Last | | 90.80 | 81.50 | 86.70 | 61.60 | 85.90 | 6.30 |
| | Best | None | 88.10 | 80.00 | 86.90 | 62.40 | 83.90 | 5.60 |
| | Last | | 82.00 | 82.00 | 85.80 | 61.10 | 82.00 | 7.30 |
| C-NMS | Best | One-cycle | 84.90 | 80.50 | 85.40 | 61.00 | 82.60 | 7.00 |
| | Last | | 85.30 | 80.50 | 85.70 | 60.20 | 82.80 | 5.50 |
| | Best | Linear | 85.90 | 81.50 | 87.30 | 63.20 | 83.70 | 7.10 |
| | Last | | 90.80 | 81.50 | 86.70 | 61.70 | 85.90 | 8.40 |
| | Best | None | 88.10 | 80.00 | 86.90 | 62.40 | 83.90 | 7.60 |
| | Last | | 82.00 | 82.00 | 85.80 | 61.10 | 82.00 | 7.20 |
| CD-NMS | Best | One-cycle | 84.90 | 80.50 | 85.40 | 61.00 | 82.60 | 6.00 |
| | Last | | 85.30 | 80.50 | 85.70 | 60.20 | 82.80 | 6.40 |
| | Best | Linear | 85.90 | 81.50 | 87.30 | 63.20 | 83.70 | 8.20 |
| | Last | | 90.80 | 81.50 | 86.70 | 61.60 | 85.90 | 7.50 |
| | Best | None | 88.10 | 80.00 | 87.00 | 62.40 | 83.90 | 7.40 |
| | Last | | 82.70 | 81.00 | 85.80 | 61.10 | 81.90 | 9.00 |
| CW-NMS | Best | One-cycle | 84.90 | 80.50 | 85.40 | 60.90 | 82.60 | 5.70 |
| | Last | | 85.30 | 80.50 | 85.70 | 60.60 | 82.80 | 6.90 |
| | Best | Linear | 85.90 | 81.50 | 87.30 | 63.50 | 83.70 | 9.30 |
| | Last | | 90.80 | 81.50 | 86.70 | 61.50 | 85.90 | 7.40 |
| | Best | None | 88.10 | 80.00 | 86.90 | 63.20 | 83.90 | 9.10 |
| | Last | | 82.00 | 82.00 | 85.80 | 61.30 | 82.00 | 9.30 |

p-ISSN: 2528-1682
e-ISSN: 2527-9165

| NMS | C.P. | Scheduler | Precision (%) | Recall (%) | mAP .5 (%) | mAP 0.5:0.95 (%) | F1-Score (%) | Speed (ms) |
|---|---|---|---|---|---|---|---|---|
| CWD-NMS | Best | One-cycle | 84.90 | 80.50 | 85.40 | 60.90 | 82.60 | 8.30 |
| | Last | | 85.30 | 80.50 | 85.80 | 60.60 | 82.80 | 8.50 |
| | **Best** | **Linear** | **85.90** | **81.50** | **87.30** | **63.50** | **83.70** | **9.30** |
| | Last | | 90.80 | 81.50 | 86.70 | 61.50 | 85.90 | 8.90 |
| | Best | None | 88.10 | 80.00 | 87.00 | 63.20 | 83.90 | 9.00 |
| | Last | | 82.70 | 81.00 | 85.80 | 61.30 | 81.90 | 9.80 |

The YOLOv7 method trained with only the secondary dataset obtained better mAP values than Faster R-CNN [11]. The best result obtained by Faster R-CNN was only 79.70%, while YOLOv7, which was trained with only secondary datasets was >80.00% for each scenario.

Third, a comparison of YOLOv7 methods trained with the combination (primary and secondary datasets). The results of training in various scenarios are shown in Table 3.

Table 3. YOLOv7 Training Results with the Combination Dataset

| NMS | C.P. | Scheduler | Precision (%) | Recall (%) | mAP .5 (%) | mAP 0.5:0.95 (%) | F1-Score (%) | Speed (ms) |
|---|---|---|---|---|---|---|---|---|
| Torchvision | Best | One-cycle | 87.00 | 82.50 | 88.00 | 61.70 | 84.70 | 6.10 |
| | Last | | 93.60 | 75.90 | 87.00 | 62.20 | 83.80 | 6.70 |
| | Best | Linear | 91.30 | 80.50 | 88.80 | 63.60 | 85.50 | 5.90 |
| | Last | | 89.40 | 77.90 | 87.30 | 62.10 | 83.30 | 5.20 |
| | Best | None | 91.50 | 77.40 | 89.00 | 63.00 | 83.90 | 6.20 |
| | Last | | 88.00 | 83.10 | 89.00 | 61.50 | 85.50 | 6.60 |
| C-NMS | Best | One-cycle | 87.00 | 82.50 | 88.00 | 61.70 | 84.70 | 7.00 |
| | Last | | 93.60 | 75.90 | 87.00 | 62.20 | 83.80 | 4.50 |
| | Best | Linear | 91.30 | 80.50 | 88.80 | 63.60 | 85.50 | 5.70 |
| | Last | | 89.40 | 77.90 | 87.30 | 62.10 | 83.30 | 5.90 |
| | Best | None | 91.50 | 77.40 | 89.00 | 63.00 | 83.90 | 7.40 |
| | Last | | 88.00 | 83.10 | 89.00 | 61.50 | 85.50 | 5.10 |
| CD-NMS | Best | One-cycle | 87.00 | 82.50 | 88.00 | 61.70 | 84.70 | 7.60 |
| | Last | | 93.60 | 75.90 | 87.00 | 62.20 | 83.80 | 6.90 |
| | Best | Linear | 91.30 | 80.50 | 88.80 | 63.50 | 85.50 | 6.70 |
| | Last | | 89.40 | 77.90 | 87.30 | 62.10 | 83.30 | 5.90 |
| | Best | None | 91.50 | 77.40 | 89.00 | 63.00 | 83.90 | 8.80 |
| | Last | | 88.00 | 83.10 | 89.00 | 61.60 | 85.50 | 6.80 |
| CW-NMS | Best | One-cycle | 87.00 | 82.50 | 88.10 | 62.00 | 84.70 | 6.60 |
| | Last | | 93.60 | 75.90 | 87.20 | 62.40 | 83.80 | 6.90 |
| | Best | Linear | 91.30 | 80.50 | 88.80 | 63.20 | 85.50 | 7.50 |
| | Last | | 89.40 | 77.90 | 87.30 | 62.00 | 83.30 | 7.10 |
| | Best | None | 91.50 | 77.40 | 89.00 | 63.30 | 83.90 | 9.20 |
| | Last | | 88.00 | 83.10 | 88.80 | 62.00 | 85.50 | 6.80 |
| CWD-NMS | Best | One-cycle | 87.00 | 82.50 | 88.00 | 61.90 | 84.70 | 7.50 |
| | Last | | 93.60 | 75.90 | 87.20 | 62.40 | 83.80 | 7.60 |
| | Best | Linear | 91.30 | 80.50 | 88.80 | 63.20 | 85.50 | 11.10 |
| | Last | | 89.40 | 77.90 | 87.30 | 62.00 | 83.30 | 9.00 |
| | **Best** | **None** | **91.50** | **77.40** | **89.20** | **63.30** | **83.90** | **10.30** |
| | Last | | 88.00 | 83.10 | 88.80 | 62.00 | 85.50 | 6.40 |

The mAP results of the YOLOv7 method trained with a combination of datasets increased by 2.00% compared to those trained with only the secondary dataset. This occurs because the combination dataset complements the shortcomings of the secondary dataset, namely, small potholes and potholes that are distant. The best mAP result was 89.20%, which was obtained from the YOLOv7 method training scenario using CWD-NMS, not using a scheduler, and using the best checkpoint. However, the NMS inference speed using CWD-NMS was slower than that of other NMS methods (10.30 ms for each image detection). In contrast to C-NMS, it detects each image with the smallest NMS inference time (4.50 ms), but the mAP value is only 87.00%.

A comparison of the YOLOv7 method with the best and fastest results and the Faster R-CNN method is presented in Table 4.

Table 4. Comparison with Previous Research

| Method | mAP .5 (%) | Speed (ms) |
|---|---|---|
| Faster R-CNN (MobileNetV2) [11] | 79.70 | 52.00 |
| YOLOv7 (C-NMS, Last Checkpoint, Scheduler) | 87.00 | 18.80 |
| **YOLOv7 (CWD-NMS, Best Checkpoint, No Scheduler)** | **89.20** | **22.10** |

The Faster R-CNN method used in previous research had the best mAP result (79.70%) and the total inference time was much slower than that of the method used in this research. Even with the YOLOv7 method, which was trained using only secondary datasets or combination datasets, mAP was able to achieve ≥85%, with an inference time of ≤23 ms.

It can be concluded that the combined dataset can improve the mAP results for pothole detection compared to using only the secondary datasets. The checkpoint best achieved better mAP values than the last checkpoint. The results of training on the secondary dataset show that the linear scheduler has the best mAP value, but in the combination dataset, not using the scheduler has the best mAP value because the mAP value of the combination dataset is better than the secondary dataset; thus, it can be concluded that without a scheduler, it is better by a few percent for training the YOLOv7 method.

The best NMS method for pothole detection using YOLOv7 is CWD-NMS. However, CWD-NMS provides a greater (slower) NMS inference speed than other NMS methods. The C-NMS method is the fastest NMS method; however, the results are smaller than those of other NMS methods.

Lastly, a comparison of single images from previous Faster R-CNN research [11]. The scenarios used in the YOLOv7 method for this comparison were the best checkpoint, no scheduler, and CWD-NMS. The detection results of Faster R-CNN and YOLOv7 are shown in Figure 5. The first and second rows show the detection results of the Faster R-CNN and YOLOv7 methods, respectively. The comparison results are obvious in the second image, in which Faster R-CNN could not detect many potholes compared with the YOLOv7 method; only one pothole was not detected because the hole was completely covered by water. Therefore, the mAP result for a single image using the YOLOv7 method was 98.40%. The YOLOv7 method can detect almost all potholes because it applies a multi-scale method during training, and the combination dataset can complement the lack of secondary datasets for small or distant potholes.

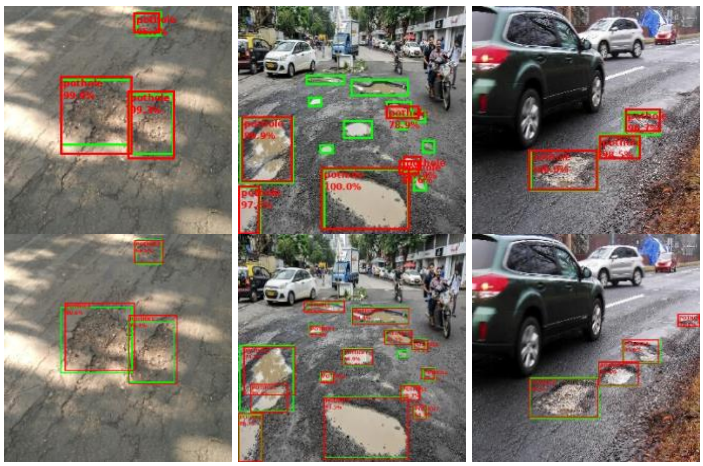

Figure 5. Comparison Between Faster R-CNN and YOLOv7 on a Single Image

## 4. CONCLUSION

In previous research, Faster R-CNN was used Faster R-CNN, which is a two-stage detector. The Faster R-CNN still could not detect small or distant potholes. Combining the YOLOv7 method with CWD-NMS solves this problem, as shown in Figure 5. YOLOv7 with CWD-NMS also obtained the best mAP value compared to other NMS methods and Faster R-CNN. A combined dataset that combines a newly, independently collected pothole dataset and previous research datasets makes YOLOv7 detection even better. The mAP value increased by 2.00% from a previous low of 85.00% to 87.00%. Various training scenarios were also tested to obtain the optimal YOLOv7 training scenario. It can be concluded that without using a scheduler in the combination dataset, it has the best mAP value. The training checkpoint with the best mAP value is the best checkpoint. Of the NMS methods tested in this research, CWD-NMS demonstrated the best mAP values compared to the others for each training scenario; however, the

inference time was slightly longer than that of the other NMS methods. The fastest NMS method for inference was C-NMS; however, the mAP results were not the best.

In the future, we will focus on using early stopping methods in training methods and reducing inference time. The reason for using early stopping was that, among all scenarios, the best mAP value was the checkpoint. The inference time problem can be solved in future research by using a better NMS method or shortening the method without reducing its detection ability.

## REFERENCES

[1] V. Kaushik and B. S. Kalyan, "Pothole Detection System: A Review of Different Methods Used for Detection," in *2022 2nd International Conference on Computer Science, Engineering and Applications, ICCSEA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICCSEA54677.2022.9936360.

[2] H. Majidifard, Y. Adu-Gyamfi, and W. G. Buttlar, "Deep machine learning approach to develop a new asphalt pavement condition index," *Constr Build Mater*, vol. 247, p. 118513, 2020, doi: 10.1016/j.conbuildmat.2020.118513.

[3] F. Meng and A. Li, "Pavement Crack Detection Using Sketch Token," *Procedia Comput Sci*, vol. 139, pp. 151–157, 2018, doi: 10.1016/j.procs.2018.10.231.

[4] F. Utaminingrum *et al.*, "Feature selection of gray-level Cooccurrence matrix using genetic algorithm with Extreme learning machine classification for early detection of Pole roads," *Results in Engineering*, vol. 20, p. 101437, Dec. 2023, doi: 10.1016/j.rineng.2023.101437.

[5] T. Siriborvornratanakul, "An Automatic Road Distress Visual Inspection System Using an Onboard In-Car Camera," *Advances in Multimedia*, vol. 2018, 2018, doi: 10.1155/2018/2561953.

[6] C. Van Geem *et al.*, "Sensors on Vehicles (SENSOVO) - Proof-of-concept for Road Surface Distress Detection with Wheel Accelerations and ToF Camera Data Collected by a Fleet of Ordinary Vehicles," *Transportation Research Procedia*, vol. 14, pp. 2966–2975, 2016, doi: 10.1016/j.trpro.2016.05.419.

[7] H. Wu *et al.*, "Road pothole extraction and safety evaluation by integration of point cloud and images derived from mobile mapping sensors," *Advanced Engineering Informatics*, vol. 42, no. March, p. 100936, 2019, doi: 10.1016/j.aei.2019.100936.

[8] S. K. Ryu, T. Kim, and Y. R. Kim, "Image-Based Pothole Detection System for ITS Service and Road Management System," *Math Probl Eng*, vol. 2015, 2015, doi: 10.1155/2015/968361.

[9] A. Dhiman and R. Klette, "Pothole Detection Using Computer Vision and Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3536–3550, Aug. 2020, doi: 10.1109/TITS.2019.2931297.

[10] M. H. Yousaf, K. Azhar, F. Murtaza, and F. Hussain, "Visual analysis of asphalt pavement for detection and localization of potholes," *Advanced Engineering Informatics*, vol. 38, no. July, pp. 527–537, 2018, doi: 10.1016/j.aei.2018.09.002.

[11] R. Rachman, A. Septiarini, and H. Hamdani, "Detection Of Road Damage Using Faster Regional-Convolutional Neural Network Method," in *2022 International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT)*, IEEE, Nov. 2022, pp. 54–59. doi: 10.1109/ICEECIT55908.2022.10030629.

[12] L. Huidrom, L. K. Das, and S. K. Sud, "Method for Automated Assessment of Potholes, Cracks and Patches from Road Surface Video Clips," *Procedia Soc Behav Sci*, vol. 104, pp. 312–321, 2013, doi: 10.1016/j.sbspro.2013.11.124.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[14] C. Cao *et al.*, "An Improved Faster R-CNN for Small Object Detection," *IEEE Access*, vol. 7, pp. 106838–106846, 2019, doi: 10.1109/ACCESS.2019.2932731.

[15] H. Chen *et al.*, "A deep learning CNN architecture applied in smart near-infrared analysis of water pollution for agricultural irrigation resources," *Agric Water Manag*, vol. 240, Oct. 2020, doi: 10.1016/j.agwat.2020.106303.

[16] Z. Xue, L. Zhang, and B. Zhai, "Multiscale Object Detection Method for Track Construction Safety Based on Improved YOLOv5," *Math Probl Eng*, vol. 2022, 2022, doi: 10.1155/2022/1214644.

[17] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, Jun. 2015, doi: 10.1109/CVPR.2016.91.

[19] T. Gao, M. Wushouer, and G. Tuerhong, "DMS-YOLOv5: A Decoupled Multi-Scale YOLOv5 Method for Small Object Detection," *Applied Sciences (Switzerland)*, vol. 13, no. 10, May 2023, doi: 10.3390/app13106124.

[20] H. Hongyu, K. Ping, F. Li, and S. Huaxin, "An Improved Multi-Scale Fire Detection Method based on Convolutional Neural Network," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, Dec. 2020, pp. 109–112. doi: 10.1109/ICCWAMTIP51612.2020.9317360.

[21] Y. Song *et al.*, "Intrusion detection of foreign objects in high-voltage lines based on YOLOv4," in *2021 IEEE 6th International Conference on Intelligent Computing and Signal Processing, ICSP 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 1295–1300. doi: 10.1109/ICSP51882.2021.9408753.

[22] R. Zhang, M. Jing, Y. Fan, and X. Zeng, "Small Object Detection in Aerial Images," in *Proceedings of International Conference on ASIC*, IEEE Computer Society, 2021. doi: 10.1109/ASICON52560.2021.9620245.

[23] Z. Zheng *et al.*, "Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation," May 2020.

[24] T. Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, May 2014, doi: 10.1007/978-3-319-10602-1_48.

[25] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov, "Label Studio: Data labeling software." Accessed: Aug. 14, 2023. [Online]. Available: https://github.com/heartexlabs/label-studio
[26] R. Rachman, M. Alfian, and U. L. Yuhana, "Classification of Illustrated Question for Indonesian National Assessment with Deep Learning," in *2023 14th International Conference on Information & Communication Technology and System (ICTS)*, IEEE, Oct. 2023, pp. 77–82. doi: 10.1109/ICTS58770.2023.10330865.
[27] L. Buitinck *et al.*, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.
[28] Y. Ding, C. Liu, H. Zhu, and Q. Chen, "A supervised data augmentation strategy based on random combinations of key features," *Inf Sci (N Y)*, vol. 632, pp. 678–697, Jun. 2023, doi: 10.1016/j.ins.2023.03.038.
[29] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Dec. 2019.