Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 5, October 2021, pp. 2724~2732
ISSN: 2302-9285, DOI: 10.11591/eei.v10i5.3062

Low resource deep learning to detect waste intensity in the river flow

Ferdinandus Fidel Putra, Yulius Denny Prabowo
Department of Informatics, Institut Teknologi dan Bisnis Kalbis, Indonesia

Article Info

Article history:

Received Oct 28, 2020
Revised Apr 3, 2021
Accepted Jul 8, 2021

Keywords:

Convolutional neural networks
Video processing
Waste detection
YOLO v3

ABSTRACT

Waste has become a significant problem in Indonesia, especially in the capital city of Jakarta due to many disasters caused by it. The one cause of flooding is the blockage of river flow by waste. The monitoring of litter is essential to find out the waste intensity in the river. The research was formed which aims to produce an application that can detect, track, and calculate river waste using YOLO v3 algorithm. This research was done in order to simplify the process of monitoring waste in the river and can calculate waste using video. This research uses 340 images directly from photos and videos, captured by researchers-detection of waste processed frame by frame by changing video into several structures. From the acquired result from the experiment, it's proven that YOLO v3 can be used for detection and counting waste recorded on video. The result of this research is an application that can detect waste and it is able to detect said objects with 98.74% of confidence from case video.

Corresponding Author:

Yulius Denny Prabowo
Department of Informatics
Institut Teknologi dan Bisnis Kalbis
JL. Pulomas Selatan Kav.22, Jakarta Timur 13210, DKI Jakarta, Indonesia
Email: yulius.prabowo@kalbis.ac.id

1. INTRODUCTION

In big cities, garbage is a logical consequence of residential areas, but garbage also often causes flooding. Based on research, Indonesia is the second-largest waste producer after China [1]. In a study conducted by Jambeck. in 2015 entitled plastic waste inputs from land into the ocean, Indonesia has contributed 3.22 million metric tons (millions of metric tons/MMT) of plastic waste [1]. President Jokowi revealed that the floods experienced at the beginning of the new year 2020 were caused by damage to the ecosystem and ecology and because there were still many people who littered [2]. The ministry of environment and forestry (KLHK), also said that the amount of waste piles produced nationally is 175,000 tons per day [3]. It's quite clear that waste is a big problem in Indonesia, especially in the capital. The concern of urban communities not to throw garbage in the river flow is essential, so that river flow is not obstructed which eventually causes flooding, on the other hand, the government needs to make efforts to monitor the discharge of waste in the river flow.

Several studies to detect waste have been conducted by other researchers before. Zhihong *et al*. [4] In the July 2018 article "multi-task detection system for garbage sorting base on high-order fusion of convolutional feature hierarchical representation", focus are on detecting waste through images with complex backgrounds. Succeeded in detecting even small objects on a complex scene, before being applied to the manipulator to perform the sorting by KUKA's robotic hand [4]. Other research on waste was carried out by Marlein Geraeds, Tim van Emmerik, Robin de Vries, and Mohd Shahrizal bin Ab Razak in an article entitled

"Riverine Plastic Litter Monitoring Using Unmanned Aerial Vehicles (UAVs)" in 2019. They monitored plastic waste using UAVs to obtain spatial data on rivers and garbage in the Klang River, Malaysia. This research succeeded in detecting debris during high tide on 30 April and 1 May, the density of waste occurred in the middle of the river, and on 30 April, and 1 May, the highest density of plastic was observed around 48m from the south riverbank [5].

A. Chung *et al*. [6] also carried out waste detection using micro-UAV in an article entitled "cloud computed machine learning-based real-time litter detection using micro-uav surveillance" in 2018. They compared the accuracy of waste detection using the convolutional neural network (CNN) method, support vector machine (SVM), single shot multibox detector (SSD, region-based fully convolutional network (R-FCN), you only look once (YOLO), custom ensemble results, and bagging ensemble results with captured data using UAV. This study found that the bagging ensemble method can perform real-time detection using UAVs with the highest accuracy compared with other methods [6].

Salimi, Dewantara, and Wibowo also presents trash detection with a smart trash bin robot using support vector machine (SVM) to classify the trash in an article "visual-based trash detection and classification system for smart trash bin robot". They first use the Haar-Cascade technique to detect the trash presence around the robot. Besides using Haar-Cascade, they also used gray-level Co-occurance matrix (GLCM) to obtain the characteristic of texture drawn from the statistic of gray intensity values from the image and the oriented gradient histogram (HOG) to calculate the appearance of gradient orientation in the localized image portion. SVM used in their experiment to classify the features into organic waste, non-organic waste, and non-waste. Their experiment found that offline testing of classification system using 5-fold Cross-Validation method obtain 82,7% of accuracy and online testing of detection and classification system obtain 63.5% of accuracy [7].

Trash detection is also researched by Mikami *et al*. in "DeepCounter: Using Deep Learning to Count Garbage Bags". The researchers calculate the amount of collected garbage (both flammable and non-flammable) taken from a camera placed behind the garbage truck, so they can see the distribution of the garbage from each city. Single shot multibox detector (SSD) algorithm is used on their experiment to detect the object. They use 1600 images of the dataset that have been annotated. A desktop NVIDIA GTX 1080 and NVIDIA Jetson in the experiment is utilized to compare the result between Original SSD and Tiny SSD. It was found that the results of Tiny SSD are better than the original SSD using the aforementioned hardware [8].

Litter detection in marine is also researched by Fulton *et al*. in the journal "robotic detection of marine litter using deep visual detection models". Along with the aforementioned research, some research that using robotics in marine for navigation and localization like "AEKF-SLAM: A new algorithm for robotic underwater navigation" journal [9] and navigation technologies for autonomous underwater vehicles journal [10] has been done to evaluate the feasibility of litter detection. The dataset that they use was sourced from the J-EDI (JAMSTEC E-library of Deep-sea Images) dataset of marine debris. J-EDI dataset provides type-specific debris data in short video format that contain images captured from real-world environments. Their training data was drawn from videos between 2000 and 2017 that labeled as containing debris. Four architectures networks were used in their experiment like YOLO V2, Tiny-YOLO, faster RCNN with Inception v2, and single shot multibox detector (SSD). From the experiment above, Faster R-CNN has the highest plastic AP with a score of 83,3 AP, followed by YOLO V2 (fine-tuned) with 82.3 AP, Tiny-YOLO with 70,3 AP, and SSD with 69,8 AP. But, Tiny-YOLO has the highest performance comparing with 3 other mentioned architecture with 205 frames per second using GTX 1080 [11].

This phenomenon has attracted researchers to apply technology to detect the amount of waste in river flows so that at least they can predict when the river should be cleaned. Experiments were carried out in watersheds where there was still visible garbage dumping. In observation, the waste that passes through the river is dominated by plastic and styrofoam waste of various sizes. Therefore, this study aims to create a system for the detection and counting of garbage in the river flow using the you only look once (YOLO) v3 method. The YOLO v3 method was chosen because there was small trash on the learning data record and through several experiments conducted by researchers, the YOLO v3 method was able to detect all objects on a large, medium, and small scale.

In this study, the researcher carried out the detection and calculation of waste passing through the river using the YOLO v3 method. The learning dataset used in this study was collected directly by taking photos of the watershed. The images used were 340 photos. The application was tested using video recordings showing trash in the river flow; the video used was taken from several river flow locations. From the experiments conducted, it can be seen that the resulting application can detect small size trash such as food wrappers to large sizes, the application can also calculate how much waste is seen in the video recording. This detection and counting waste application can work more effectively with a video containing footage of rivers with minimum sunlight reflection throughout the video.

2. RESEARCH METHOD

Experimental data were taken from river basins in Jakarta where trash was often seen passing through the river. Observations were made to determine the smooth flow of the river and the amount of waste seen in the river flow. This observation was carried out by researchers to find data collection locations. Some of the things that were taken into consideration were: the smooth flow, discharge, and variation of waste and river clarity, the clearer the river, object detection was difficult because the reflection of other objects in the water made it difficult for the algorithm to detect waste. The watershed that was selected as training data collection is a river flow area with a reasonably smooth flow, variations in the size and type of garbage, and clear water conditions (it can reflect other objects). Three hundred seventy photos of debris and video recording of river flow at several points as many as seven videos, which were used as training and testing data in this study, were taken using a smartphone. These photos will then be resized to reduce the resolution and size of the image; all this data will then be uploaded to Google Drive. Experiments were carried out using the help of the google collab engine. This experiment using the Google colab engine because the researcher can utilize its high-end GPU to process images faster, instead of used the researcher's local GPU and CPU [12]. GPU is very important in completing things related to graphics [13]. The researcher is preferred to use GPU than CPU because the GPU can run 20 to 40 times faster than the CPU [14], [15].

The literature review was carried out by researchers to determine the methods and algorithms that can be used to detect objects in the form of garbage. Unlike the YOLO v2 which often struggled with small object detection [16], the results of literature studies and experiments conducted have found that YOLO v3 is most suitable for waste detection because the algorithm can detect objects that vary from small to large [7]. The usage of YOLO v3 is also a good choice for this experiment because the majority of data used in this experiment also have a small sized object [17]. After the data acquisition was completed, the researcher annotated the data. Annotations are performed on all images in the dataset. Annotation is done to label objects and takes the coordinates of the items in the picture. The result of this annotation is object information in the form of the center point, coordinates, and class. Furthermore, this image dataset is divided into two groups: training data and validation data with a composition of 90% training data and 10% validation data. The next step is to create a model using YOLO V3. YOLO v3 training architecture design is using the architecture from Darknet-53. Darknet is the open-source neural network framework written in C and CUDA [18]. This Darknet 53 is much powerful than Darknet 19, but still more efficient than ResNet-101 [19]. The architecture of Darknet-53 can be seen in Figure 1.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 1. The architecture of Darknet-53 [19]

CUDA is used because this software has reliable architecture and the language is almost similar to C, so it is easy to understand [20]. Feature from the last three residual blocks will be used to feed them into the detector. The detector will produce 3 different results for each scale. The results are for large scale, medium scale, and small scale. The process of the detector can be seen in Figure 2. After the model is generated, the next step is to create a waste counting system. Waste counting is done by creating a counter line, objects that are detected as garbage, and crossing the counter line will be counted as garbage. In this detection and counting waste experiment, the researcher adjusted several YOLO V3 parameters to suit the research context. Changes were made to a batch, subdivision, max_bactches, and steps parameters. Changes were also made to filters and classes, [yolo] and [convolutional] sessions. In the waste counter detection and creation stage, object detection is carried out at each frame in the video. Objects that are detected in the frame

and have a confidence value> 0.5 are stored in a new variable to retrieve object offset information. Object Offset contains object coordinates, objectiveness, and object class (x, y. W, h, objectiveness (0/1), class). The system runs non-max suppression for all objects with more than one bounding box. This non-max suppression is done to take the bounding box with the highest confidence value. Bounding boxes and determining the object's center are drawn around the item from the offset obtained. The center point on the detected object will be used as a reference for calculating waste. If the midpoint is intersected with a counter line, it is counted as waste. The system path in this study can be seen in Figure 3 in appendix.

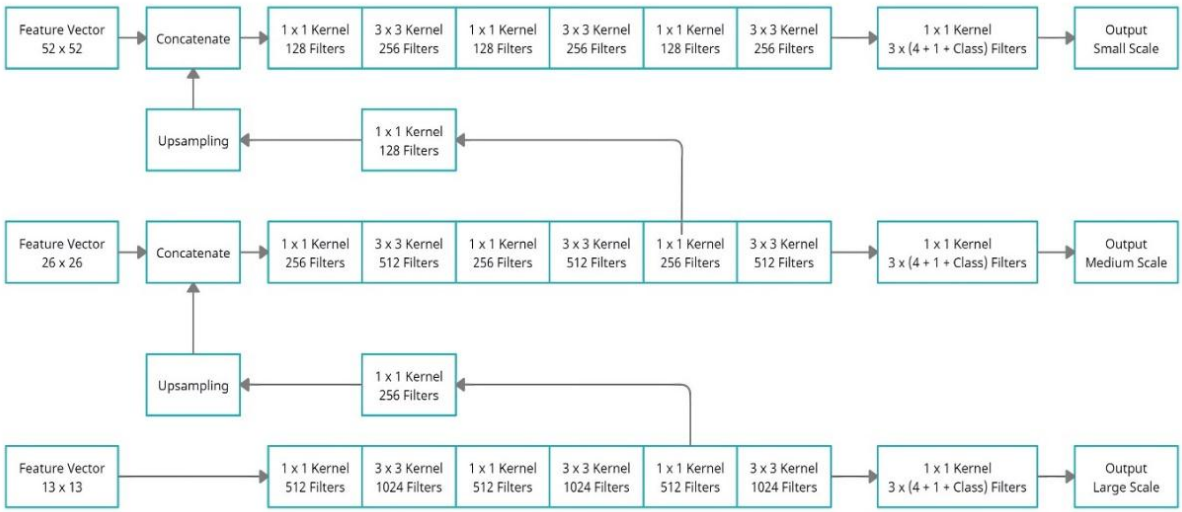

Figure 2. Multi-scale detector (with some changes) [21]

This waste detection and counting continue at every frame, from the beginning to the end of the video. The final result of waste detection and counting produces a new video containing elements of the amount of waste, counter lines, and trash objects surrounded by bounding boxes, class labels, and the size of their configuration. Testing of the application is carried out using input in the form of video. This waste detection and counting application were created using the Python 3.7 language. The results of the video processing that has been done before producing new videos with .avi format.

3. RESULTS AND DISCUSSION

Modeling in this study uses the YOLO v3 algorithm using the Darknet 53 architecture. Training is carried out on YOLO v3 using 106 layers with convolution, shortcuts, and feature combination (concatenate). The architecture used in YOLO v3 can be seen Figure 4. This study also prepares a pre-trained weight model of Darknet-53 architecture as a medium for transfer learning. Transfer learning is used in this study so that researcher did not have to re-train the entire network from scratch [22], [23]. The training data is carried out using 2000 iterations with a comparison of the train data and validation of 9: 1. The graph results of the training and data validation that has been done can be seen in Figure 5. YOLO v3 accuracy can be seen from the resulting average precision (AP) and mean average precision (mAP). The final AP result obtained in this training is 64.43%, while the best AP result obtained is 66.07%. The mAP obtained was 0.654692 or 65.46%. The loss gained during the training period is 0.9593.

Waste detection and calculation are carried out on video testing, video recording is entered into the application to detect waste that appears on the video. The detected trash is seen as an object with a box around it, on this detection, the confidence value is also displayed, the greater this value, the more confident the application will recognize the object as trash. For waste calculation, the researchers drew guidelines on the video. This guideline is used as a reference; if the centroid point of the object that is detected as waste intersects with the guideline, the application will count it as garbage. The calculation using the guideline in this experiment is using the sort method from Alex Bewley *et al*. [24] in their journal titled "Simple Online and Realtime Tracking" [24]. Initiation of the guideline is used to trigger counter from sort method. From the experimental results using data such as in Figure 6, the application is able to detect 26 objects as rubbish, including dry leaves that fall on the river surface. This detection is done frame by frame.

```
layer   filters  size/strd(dil)      input                output
  0 conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
  1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
  2 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
  3 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
  4 Shortcut Layer: 1,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
  5 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
  6 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  7 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
  8 Shortcut Layer: 5,  wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF
  9 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
 10 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
 11 Shortcut Layer: 8,  wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF
 12 conv    256       3 x 3/ 2    104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
 13 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 14 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 15 Shortcut Layer: 12,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 16 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 17 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 18 Shortcut Layer: 15,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 19 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 20 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 21 Shortcut Layer: 18,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 22 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 23 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 24 Shortcut Layer: 21,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 25 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 26 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 27 Shortcut Layer: 24,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 28 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 29 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 31 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 32 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 34 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 35 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  52 x  52 x 256 0.001 BF
 37 conv    512       3 x 3/ 2     52 x  52 x 256 ->   26 x  26 x 512 1.595 BF
 38 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 39 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 40 Shortcut Layer: 37,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 41 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 42 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 43 Shortcut Layer: 40,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 44 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 45 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 46 Shortcut Layer: 43,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 47 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 48 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 49 Shortcut Layer: 46,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 50 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 51 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 52 Shortcut Layer: 49,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 53 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 54 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 55 Shortcut Layer: 52,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 56 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 57 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 58 Shortcut Layer: 55,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 59 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 60 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 61 Shortcut Layer: 58,  wt = 0, wn = 0, outputs:  26 x  26 x 512 0.000 BF
 62 conv   1024       3 x 3/ 2     26 x  26 x 512 ->   13 x  13 x1024 1.595 BF
 63 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 64 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 65 Shortcut Layer: 62,  wt = 0, wn = 0, outputs:  13 x  13 x1024 0.000 BF
 66 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 67 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 68 Shortcut Layer: 65,  wt = 0, wn = 0, outputs:  13 x  13 x1024 0.000 BF
 69 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 70 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 71 Shortcut Layer: 68,  wt = 0, wn = 0, outputs:  13 x  13 x1024 0.000 BF
 72 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 73 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 74 Shortcut Layer: 71,  wt = 0, wn = 0, outputs:  13 x  13 x1024 0.000 BF
 75 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 76 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 77 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 78 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 79 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 80 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 81 conv     18       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  18 0.006 BF
 82 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
 83 route  79                                      ->   13 x  13 x 512
 84 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
 85 upsample                 2x    13 x  13 x 256 ->   26 x  26 x 256
 86 route  85 61                                   ->   26 x  26 x 768
 87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
 88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
 92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
 93 conv     18       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x  18 0.012 BF
 94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
 95 route  91                                      ->   26 x  26 x 256
 96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
 97 upsample                 2x    26 x  26 x 128 ->   52 x  52 x 128
 98 route  97 36                                   ->   52 x  52 x 384
 99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
105 conv     18       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  18 0.025 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
```

Figure 4. YOLO architecture used in this research

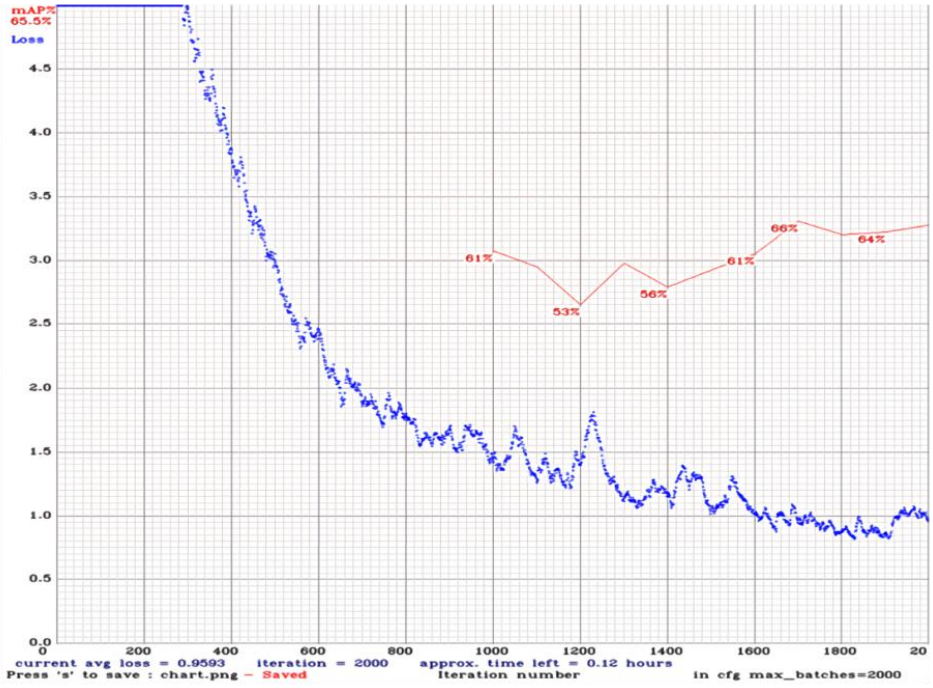

Figure 5. Training result graph

Figure 6. Example output from video used in the experiment

According to the result, the object that can be detected as waste is the object that has such a minimum or no sunlight reflection around its area. Most objects can not be detected as waste when the object passing through the reflection of sunlight even the train data of the object has been annotated under the reflection of sunlight. The results obtained are same as the statement in the journal "Estimation of Sunlight Direction Using 3D Object Models" which states about the object to sunlight [25]. The program may fail to keep track of the detected waste, this may be caused by the lack of training data variance for the waste images. This may pose a problem whenever the waste passing through the guideline when it is not detected. Hence, collecting objects from many various perspectives for train data and picking the right area for the guideline is necessary for effective counting. In this experiment, the guideline is set manually from several points of view (based on reflection), and the guideline which according to the author is optimal is chosen. The guideline used in Figure 6 is chosen because it has less reflection of sunlight and it is also the area mostly traversed by waste.

4. CONCLUSION

From the results of experiments conducted in the study, it can be concluded that the application of the Convolutional Neural Networks algorithm in the YOLO V3 architecture can be implemented to detect waste in river flows. The resulting application can also count the detected waste in the video recording. The experimental results show that the final AP value obtained in training is 64.43%, the best AP value obtained during the training period is 66.07%, and the average AP (mAP) received is 0.654692 or 65.46%. The highest waste detection accuracy obtained was 98.74% with a loss during the training period of 0.9593. These results indicate that the researchers succeeded in detecting and calculating waste in river flows using the CNN algorithm on the YOLO V3 architecture with low computation resources. Although this study succeeded in answering the research objectives, the resulting algorithms had not yet correctly detected objects that were hit by sunlight reflection; the researchers suggested that changes in parameters or the use of other learning algorithms were made to correct these deficiencies. This system can be try to develop using CCTV with the real-time case for future enhancement. It can try with another new detection algorithm like SSD, or YOLO V4 also.

APPENDIX

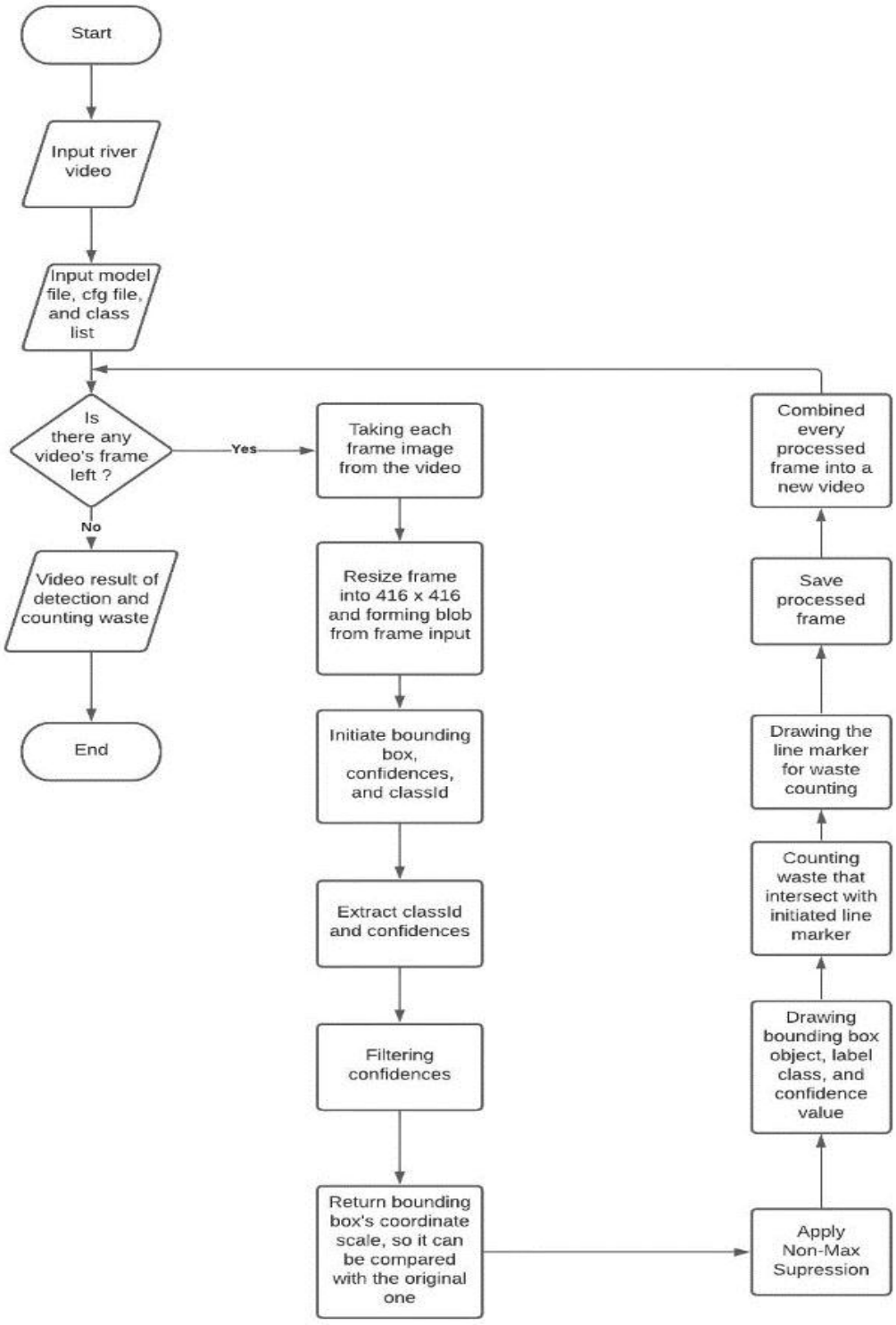

Figure 3. The path of waste detection system

ACKNOWLEDGEMENTS

Researchers would like to thank Institut Teknologi dan Bisnis Kalbis for funding this research and to Google for providing the free computing tools used during this research.

REFERENCES

[1] K. Sheth, "Countries Putting The Most Plastic Waste Into The Oceans", WorldAtlas, 2021. [Online]. Available: https://www.worldatlas.com/articles/countries-putting-the-most-plastic-waste-into-the-oceans.html. [Accessed: 08-Jul- 2021].

[2] C. I. Tim, "Jokowi: Banjir Jakarta karena Sampah dan Kerusakan Ekologi," *CNN Indonesia*, 02-Jan-2020. [Online]. Available: https://www.cnnindonesia.com/nasional/20200102100743-20-461770/jokowi-banjir-jakarta-karena-sampah-dan-kerusakan-ekologi. [Accessed: 15-Mar-2020].

[3] Nur Faizah Al Bahriyatul Baqiroh, "Timbulan Sampah Nasional Capai 64 juta ton per Tahun - Ekonomi Bisnis.com,"21-Feb-2019. [Online]. Available:https://ekonomi.bisnis.com/read/20190221/99/891611/timbulan-sampah-nasional-capai-64-juta-ton-per-tahun. [Accessed: 23-Dec-2019].

[4] C. Zhihong, Z. Hebin, W. Yan, W. Yanbo and L. Binyan, "Multi-task Detection System for Garbage Sorting Base on High-order Fusion of Convolutional Feature Hierarchical Representation," *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 5426-5430, doi: 10.23919/ChiCC.2018.8483842.

[5] M. Geraeds, T. van Emmerik, R. de Vries, and M. S. bin Ab Razak, "Riverine plastic litter monitoring using Unmanned Aerial Vehicles (UAVs)," *Remote Sensing*, vol. 11, no. 17, p. 2045, 2019, doi: 10.3390/rs11172045.

[6] A. Chung, D. Y. Kim, E. Kwok, M. Ryan, E. Tan and R. Gamadia, "Cloud Computed Machine Learning Based Real-Time Litter Detection using Micro-UAV Surveillance," *2018 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2018, pp. 1-4, doi: 10.1109/URTC45901.2018.9244800.

[7] Salimi, B. Bayu Dewantara, and I. Wibowo, "Visual-based trash detection and classification system for smart trash bin robot", *2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, pp. 378-383, 2018, doi: 10.1109/KCIC.2018.8628499.

[8] K. Mikami, Y. Chen, J. Nakazawa, Y. Iida, Y. Kishimoto and Y. Oya, "DeepCounter: Using Deep Learning to Count Garbage Bags," *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 1-10, doi: 10.1109/RTCSA.2018.00010.

[9] X. Yuan, J.-F. Mart´ınez-Ortega, J. A. S. Fernandez, and M. Eckert, "AEKF-SLAM: A New Algorithm for Robotic Underwater Navigation," *Sensors*, vol. 17, no. 5, 2017, doi: 10.3390/s17051174.

[10] L. Stutters, H. Liu, C. Tiltman and D. J. Brown, "Navigation Technologies for Autonomous Underwater Vehicles," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 4, pp. 581-589, July 2008, doi: 10.1109/TSMCC.2008.919147.

[11] M. Fulton, J. Hong, M. J. Islam and J. Sattar, "Robotic Detection of Marine Litter Using Deep Visual Detection Models," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5752-5758, doi: 10.1109/ICRA.2019.8793975.

[12] E. BUBER and B. DIRI, "Performance Analysis and CPU vs GPU Comparison for Deep Learning," *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, 2018, pp. 1-6, doi: 10.1109/CEIT.2018.8751930.

[13] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU Computing," in *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879-899, May 2008, doi: 10.1109/JPROC.2008.917757.

[14] G. Sethumadhavan, S. A. Narayanasamy and A. Gopalakrishnan, "Performance evaluation of image smoothing on CPU and GPU using multithreading — An experimental approach in high performance computing," *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2016, pp. 1-5, doi: 10.1109/ICCIC.2016.7919680.

[15] A. Syberfeldt and Tom Ekblom, "A Comparative Evaluation of the GPU vs The CPU for Parallelization of Evolutionary Algorithms Through Multiple Independent Runs," *International Journal of Computer Science and Information Technology*, vol. 9, no. 3, pp.01-14, 2017, doi: 10.5121/ijcsit.2017.9301.

[16] Z. Du, J. Yin and J. Yang, "Expanding Receptive Field YOLO for Small Object Detection," *Journal of Physics: Conference Series*, vol. 1314, p.012202, 2019, doi: 10.1088/1742-6596/1314/1/012202.

[17] Nhat-Duy Nguyen, T. Do, T. D. Ngo and Duy-Dinh Le, "An Evaluation of Deep Learning Methods for Small Object Detection," *Journal of Electrical and Computer Engineering*, vol. 2020, pp.1-18, 2020, doi: 10.1155/2020/3189691.

[18] J. Redmon, "Darknet: Open Source Neural Networks in C." [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 10-Aug-2020].

[19] J. Redmon and A. Farhadi, "YOLO v.3," Tech Rep., pp. 1-6, 2018.

[20] J. Darmawan and S. Mungkasi, "Performance of parallel computation using CUDA for solving the one-dimensional elasticity equations", Journal of Physics: Conference Series, vol. 801, p. 012080, 2017. doi: 10.1088/1742-6596/801/1/012080.

[21] E. Yanjia Li, "Dive Really Deep into YOLO v3: A Beginner's Guide - Towards Data Science," 31-Dec-2019. [Online]. Available: https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e. [Accessed: 09-May-2020].

[22] N. Best, J. Ott and E. J. Linstead, "Exploring the efficacy of transfer learning in mining image-based software artifacts," *Journal of Big Data*, vol. 7, no. 59, pp. 1-10, 2020, doi: 10.1186/s40537-020-00335-4.

[23] M. A. M. Fuad, M. R. Ab Ghani, R. Ghazali, T. A. Izzuddin, M. F. Sulaima, Z. Jano and T. Sutikno, "Training of Convolutional Neural Network using Transfer Learning for Aedes Aegypti Larvae," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 4, pp.1894-1900, doi: 10.12928/telkomnika.v16i4.8744.

[24] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464-3468, doi: 10.1109/ICIP.2016.7533003.

[25] Y. Liu, T. Gevers and X. Li, "Estimation of Sunlight Direction Using 3D Object Models," in *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 932-942, March 2015, doi: 10.1109/TIP.2014.2378032.

BIOGRAPHIES OF AUTHORS

Ferdinandus Fidel Putra is a final year student in the Informatics study program at Institut Teknologi dan Bisnis Kalbis majoring in soft computing. Currently working on a thesis in the field of computer vision.

Yulius Denny Prabowo is a lecturer and Head of Informatics Department at Institut Teknologi dan Bisnis Kalbis with a research focus in the fields of machine learning in general, natural language processing and document summarization.