Waste Object Detection and Classification using Deep Learning Algorithm: YOLOv4 and YOLOv4-tiny

Andhy Panca Saputra[a], Kusrini [b]
[a] Student, Universitas Amikom Yogyakarta, Indonesia
[b] Associate Professor, Universitas Amikom Yogyakarta, Indonesia
Email: [a]andhy.28@students.amikom.ac.id, [b]kusrini@amikom.ac.id

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: As industrialization, urbanization, and global population levels increased, so did the pollution and environmental degradation because of the waste. The common problem related to waste is the waste sorting that is still improperly handled from the household level to the final disposal site. The solution offered in this research is using deep learning algorithm for object detection using YOLOv4 and YOLOv4-tiny with Darknet-53. The dataset consists of 3870 waste images divided into 4 classes like glass, metal, paper, and plastic. At the testing stage, each model uses 3 different inputs such as images, videos, and webcams. In the YOLOv4-tiny model, experiments hyperparameter were also carried out on subdivision values and mosaic data augmentation. The result proves that YOLOv4 have better performance than YOLOv4-tiny for object detection, although in terms of computational speed the YOLOv4-tiny's scores are better. The best results from the YOLOv4 model reach mAP 89.59%, precision 0.76, recall 0.90, F1-score 0.82, and Average IoU 64.01%, while the best YOLOv4-tiny results are mAP 81, 84%, precision 0.59, recall 0.83, F1-score 0.69, and Average IoU 48.35%. This research also proves that the models with smaller subdivision values and uses a mosaic have an optimal performance.

Keywords: Deep Learning, Object Detection, YOLOv4, YOLOv4-tiny, Waste

1. Introduction

The data from the Ministry of Environment and Forestry (KLHK) shows that the amount of waste throughout Indonesia continues to increase from 175,000 tonnes per day or the equivalent of 64 million tonnes of waste per year (according to September 2019 data) to 67.8 million tonnes in 2020. The required time for waste degradation is longer than the production time **(Devi et al., 2018)**. If the waste is mixed without any sorting process, then the degraded waste pile in the landfill site is only the organic waste. Moreover, with the mixing of this waste, the inorganic waste degradation time is longer. That is why the sorting and recycling process of the waste is important, both from the household level to the level of the final processing site **(Sakr et al., 2016)**.

Object detection is a technique from computer vision that is used to identify objects in the form of images or videos. Object recognition is the main output of deep learning and machine learning algorithms. The objective is to teach the computer to gain a level of understanding of what is contained in an image. The recognition of these objects is an essential element in the waste classification system created. Several methods have been used in previous studies to detect and classify waste, namely SURF-BoW and Multi-Class SVM **(Y. Liu et al., 2018)**, Content-Based Image Retrieval (CBIR) systems **(Chinnathurai et al., 2016)**, K-NN Algorithm **(Torres-García et al., 2015)**, and K-mean Cluster Algorithm **(Deepa & Roka, 2018)**. But in research that using those methods, there are still facing several obstacles such as low accuracy and limited identifiable types of waste.

Convolutional Neural Network (CNN) is an algorithm that has become the standard used for image classification and object recognition. During its development, several algorithm developments based on CNN emerged, some of which are Region-based-CNN (R-CNN) **(Girshick et al., 2014)**, Fast R-CNN **(Girshick, 2015)**, Faster R-CNN **(S. Ren et al., 2017)**, and You Only Look Once (YOLO) **(Redmon et al., 2016)**.

In this research, YOLOv4 and YOLOv4-tiny algorithms are proposed. The YOLO algorithm is an algorithm used as a Real-Time Object Detection that is usable in detecting images directly. Based on the research conducted by Juan Du about Understanding of Object Detection Based on CNN Family and YOLO **(Du, 2018)**, it shows that the performance of YOLO algorithm is better than other CNN family algorithms, such as Fastest DPM, R-CNN Minus R, Faster R-CNN ZF, YOLO VGG-16, Fast R-CNN, Faster R-CNN VGG-16, Faster R-

CNN ResNet, with mAP (mean Average Precision) 78.6. To improve the performance of this algorithm, a hyperparameter experiment was also carried out on the subdivision value and the mosaic data augmentation.

The results of this research can be accessed through the website that the author created with the flask framework, google collab, or via a local PC. To input, you can also go through 3 different inputs like images, videos, and webcams. This research is structured as follows: section 1 introduction, after that it is continued with literature review and related works in section 2, then research methods and proposed experiments in section 3, followed by results and discussion in section 4, and finally in section 5, conclusive remarks and our future work.

2. Literature Review and Related Work

YOLO (You Only Look Once) is an algorithm developed by Joseph Redmon in 2016 **(Redmon et al., 2016)** to detect objects in real-time based on CNN (Convolutional Neural Network). This algorithm can detect through image input, video input, and real-time using a webcam. Then in 2017 during the CVPR (Conference on Computer Vision and Pattern Recognition), YOLOv2 was released by Joseph Redon and Ali Farhadi by increasing the accuracy and speed of the algorithm **(Redmon & Farhadi, 2017)**. Then YOLOv3 was released in April 2018 by Joseph Remon and Ali Farhadi with increased performance from the previous version **(Redmon & Farhadi, 2018)**. The next version of this algorithm, YOLOv4, was released on April 24, 2020. This version not written by Joseph Ramond, the author of versions 1 - 3 but Alexey Bochkovskiy who built the Windows version of YOLO, together with Chien-Yao Wang, and Hong-Yuan Mark Lio **(Bochkovskiy et al., 2020)**.

YOLO uses an ANN (Artificial Neural Network) approach to detect objects in an image. This network divides the image into several regions and predicts each bounding box and probability for each region. Then the bounding box is compared with each predicted probability.

In general, the YOLO algorithm has simple steps when compared to SSDs and the faster R-CNN. Most of the steps in this algorithm same as image classification using the CNN algorithm. YOLO performs bounding box calculations with one map feature scale. The stages of the YOLO algorithm are input images marked into several S x S grid boxes; each grid box create a few bounding boxes of various sizes. The bounding box size is bigger than the grid box size; If the bounding box contains objects, the grid box that create the bounding box is responsible for detecting that object; One grid box can only expose one object. If more than one bounding box for a certain grid box detects an object, it is only allowed to select one object with one bounding box that has the highest level of accuracy; No Max Suppression is used to determine when many bounding boxes are detected for the same object.

A modern detector is usually composed of a Backbone and Head. A backbone that is pre-trained on ImageNet. For those detectors running on GPU platform, their backbone could be VGG **(Simonyan & Zisserman, 2015)**, ResNet **(He et al., 2016)**, ResNeXt **(Xie et al., 2017)**, or DenseNet **(Huang et al., 2017)**. For those detectors running on CPU platform, their backbone could be SqueezeNet **(Iandola et al., 2016)**, MobileNet **(A. Howard et al., n.d.; A. G. Howard et al., 2017; Sandler et al., 2018; Tan et al., 2019)**, or ShuffleNet **(Ma et al., 2018; Zhang et al., 2018)**. Head which is used to predict classes and bounding boxes of objects. As to the head part, usually categorized into two kinds, the first one is a one-stage object detector or Dense Prediction like YOLO **(Bochkovskiy et al., 2020; Redmon et al., 2016; Redmon & Farhadi, 2017, 2018)**, SSD **(W. Liu et al., 2016)**, and RetinaNet **(Lin et al., 2017)**. And the second one is a two-stage object detector or Sparse Prediction like R-CNN **(Girshick et al., 2014)** series, including fast R-CNN **(Girshick, 2015)**, faster R-CNN **(S. Ren et al., 2017)**, R-FCN **(Dai et al., 2016)**, and Libra R-CNN **(Pang et al., 2019)**. Object detectors developed in recent years often insert the neck layers between the backbone and head, and these layers are usually used to collect feature maps from different stages. Networks equipped with this mechanism include Feature Pyramid Network (FPN) **(Xiaohan Li et al., 2019)**, Path Aggregation Network (PAN) **(S. Liu et al., 2018)**, BiFPN **(Tan et al., 2020)**, and NAS-FPN **(Ghiasi et al., 2019)**. YOLOv4 consists of, Backbone: CSPDarknet53 **(Wang et al., 2020)**, Neck: SPP **(He et al., 2014)** and PAN **(S. Liu et al., 2018)**, Head: YOLOv3 **(Redmon & Farhadi, 2018)**. For more details, the YOLOv4 architecture can be seen in Figure 1.

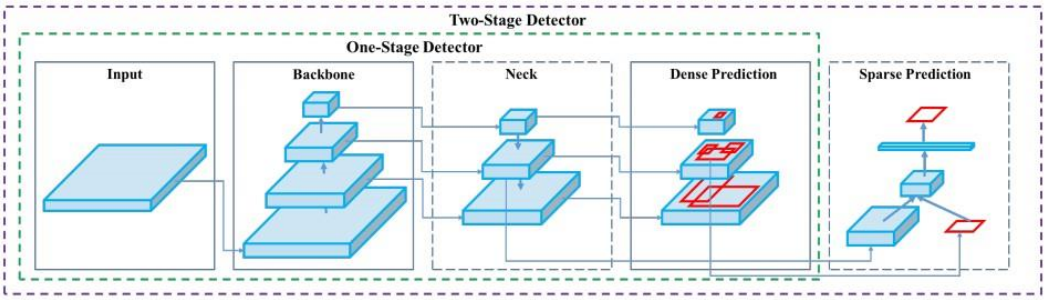


Figure 1. Architecture YOLOv4

Some of the new methods introduced at YOLOv4 are Bag of Freebies, Bag of Special, and Mosaic. Bag of Freebies is a training method that can make the object detector receive better accuracy without increasing the inference cost. Bag of Special is a method that only increases the inference cost by a small amount but it can significantly improve the accuracy of object detection. While Mosaic represents a new data augmentation method that mixes four training images. This thing reduced the need for a large mini-batch size significantly.

Yolov4-tiny is a model developed from the YOLOv4 algorithm. This model has a simpler network structure and reduces the parameters that exist in YOLOv4. With a network structure like this, YOLOv4-tiny is suitable for development on mobile devices and embedded devices **(Jiang et al., 2020)**.

Darknet-53 is one of the frameworks (a framework is used to make it easier for software developers to create and develop applications) that could be used on YOLO. Darknet-53 uses 53 convolutional layers **(Weijun Chen et al., 2021)**. It can be seen in Figure 2, the more convolutional layers, the more the filter will also increase as much as double. This was to search for a wide range of objects. For the first layer lack of the variant to be found, but with more layers added, the search scope could be wider.

Type	Filters	Size	Output	
Convolutional	32	3x3	256x256	
Convolutional	64	3x3/2	128x128	
Convolutional	32	1x1		1x
Convolutional	64	3x3		
Residual			128x128	
Convolutional	128	3x3/2	64x64	
Convolutional	64	1x1		2x
Convolutional	128	3x3		
Residual			64x64	
Convolutional	256	3x3/2	32x32	
Convolutional	128	1x1		8x
Convolutional	256	3x3		
Residual			32x32	
Convolutional	512	3x3/2	16x16	
Convolutional	256	1x1		8x
Convolutional	512	3x3		
Residual			16x16	
Convolutional	1024	3x3/2	8x8	
Convolutional	512	1x1		4x
Convolutional	1024	3x3		
Residual			8x8	
Avgpool		Global		
Connected		1000		
Softmax				

Figure 2. Darknet-53

Darknet-53 has a better performance compared to Darknet-19. In addition, Darknet-53 has better performance to ResNet-101 and 1.5 times faster. Compared to ResNet-152, Darknet-53 has the same performance but two times faster. BFLOP/s Darknet-53 also achieves the highest score than Darknet-19, ResNet-101, and ResNet-152. This suggests that Darknet-53's network structure utilizes the GPU well, making it more efficient and faster. That is mostly because ResNets have just way too many layers and is not very efficient.

In a single-stage object detection network, there is much interesting research: Research to detect wheat heads using YOLO was conducted by **(Gong et al., 2020)**. Some of the algorithms used for comparison are YOLOv3, YOLOv4, Faster R-CNN, and self-developed methods. The result was good with mAP 94.5% and FPS 88. There is also a study using YOLOv4 for detecting citrus in an orchard environment by **(Wenkang Chen et al., 2020)** who use the Kinect V2 camera, The Canopy, and the K-means ++ algorithm to produce RGB images. Meanwhile, **(P. Ren et al., 2020)** use YOLO for real-time people counting. YOLOv4 is also utilized by **(Gandhewar, 2020)** to scene text detection using multiple datasets such as ICDAR 2015, ICDAR 2013, ICDAR 2003, SVT, IIIT5K, MSRA-TD-500. **(Feng et al., 2021)** also use YOLO to create an Autonomous UAV System by comparing model performance based on input size.

Several studies utilizing YOLO for waste detection: Research by **(Kumar et al., 2021)** who use YOLOv3 to create Smart Waste Management. The experimental results were good with YOLOv3 mAP 94.99%, while YOLOv3-tiny was only 51.95%. The use of YOLO for waste classification and sorting was also carried out by **(Shah et al., 2020)** and the accuracy results up to 98%. **(Valente et al., 2019)** use YOLO for the identification of waste containers which can perform detection using images, video, or real-time video capture. The resulting accuracy is 90%. Another study by **(Jardosh et al., 2020)** discussed Modern Waste Management using Smart Bin. Besides being able to classify the types of waste, it can also track all the bins for their waste level, tracking all the trucks and generating optimized routes for the truck drivers. The mAP results in 84.44%. Other research by **(Xiali Li et al., 2020)** utilizing YOLOv3 for vision-based water surface waste detection using a robot with the resulting mAP reach 91.45%.

3. Research Method

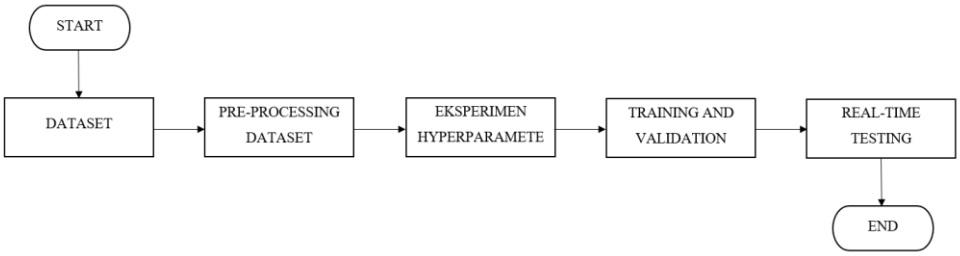


Figure 3. Workflow

Figure 3 shows the steps that will be carried out in this research. The first step is doing dataset collecting for research. The dataset is divided into two parts, the dataset for training and the other one is for validation. The data used is a dataset derived from Github obtained from Kaggle (Garbage Classification). The dataset is an image with .jpg format which has a size of 299 x 299 pixels. The data used in this research were 3870 with the distribution of training data as many as 3126 images and data validation as many as 744 images. This dataset consists of 4 classes of waste types such as glass, metal, paper, and plastic.

The second step is the pre-processing dataset that is labeling all images with annotation tools namely LabelImg, which produces a .txt file that containing image information. In addition to labeling, there is a data augmentation process in the pre-processing dataset. Data Augmentation is a technique of manipulation/modification of data without losing the essence of the data. In this research, data augmentation was carried out to reproduce the dataset that will be used. The augmentation used in this study is mosaic data augmentation. Mosaic is a new data augmentation method, if CutMix combines 2 images, mosaic combines 4 training images. This significantly reduces the need for a large mini-batch size.

The next stage is a hyperparameters experiment to be conducted in this research. There are two variables, the variable that is used as a standard in conducting this research called as variable of standard and the variable that is used as an experiment hyperparameter to compare the performance of the YOLO architecture called as variable input.

- Variable of standard: Batch 64, Network Size 406x406, Max Batches 8000, Steps 6400 dan 7200, Filters 27, Classes 4.
- Variable input: Subdivision 16 and 8, using and without using a mosaic.

Variable input experiments can be seen in Table 1.

Table 1. Variable Input Experiments

EKSPERIMENT	SUBDIVISION	MOSAIC
1	16	-
2	16	√
3	8	-
4	8	√

The next step is the training and validation stages by Google Colaboratory. The results of this process significantly affect the success of the YOLOv4 model in performing object detection. The input of this process is the training dataset and validation dataset that has been prepared in advance. While the output of this process is a weights file which will later be used during real-time testing. At this stage, the transfer learning process occurs, which is a method that utilizes a model that has been trained on a dataset to solve other similar problems. Transfer learning at this stage is obtained from the AlexeAB repository which has previously been trained on the COCO dataset, which is a dataset from Microsoft which has 80 different classes. For YOLOv4 use yolov4.conv.137 and for YOLOv4-tiny use yolov4-tiny.conv.29. The YOLOv4 and YOLOv4¬-tiny structures are shown in Figure 4 and Figure 5.

```
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 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     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                                    ->  208 x 208 x  64
   4 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   5 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   6 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
   8 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   9 route  8 2                                  ->  208 x 208 x 128
  10 conv     64       1 x 1/ 1    208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
                                                ...
 151 route  147                                   ->   26 x  26 x 256
 152 conv    512       3 x 3/ 2     26 x  26 x 256 ->   13 x  13 x 512 0.399 BF
 153 route  152 116                               ->   13 x  13 x1024
 154 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 155 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 156 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 157 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 158 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 159 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 160 conv     27       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  27 0.009 BF
 161 yolo
```

Figure 4. The YOLOv4 Structure

```
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 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     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                                    ->  208 x 208 x  64
   4 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   5 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   6 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
   8 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   9 route  8 2                                  ->  208 x 208 x 128
  10 conv     64       1 x 1/ 1    208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
                                                ...
 151 route  147                                   ->   26 x  26 x 256
 152 conv    512       3 x 3/ 2     26 x  26 x 256 ->   13 x  13 x 512 0.399 BF
 153 route  152 116                               ->   13 x  13 x1024
 154 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 155 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 156 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 157 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 158 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
 159 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
 160 conv     27       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  27 0.009 BF
 161 yolo
```

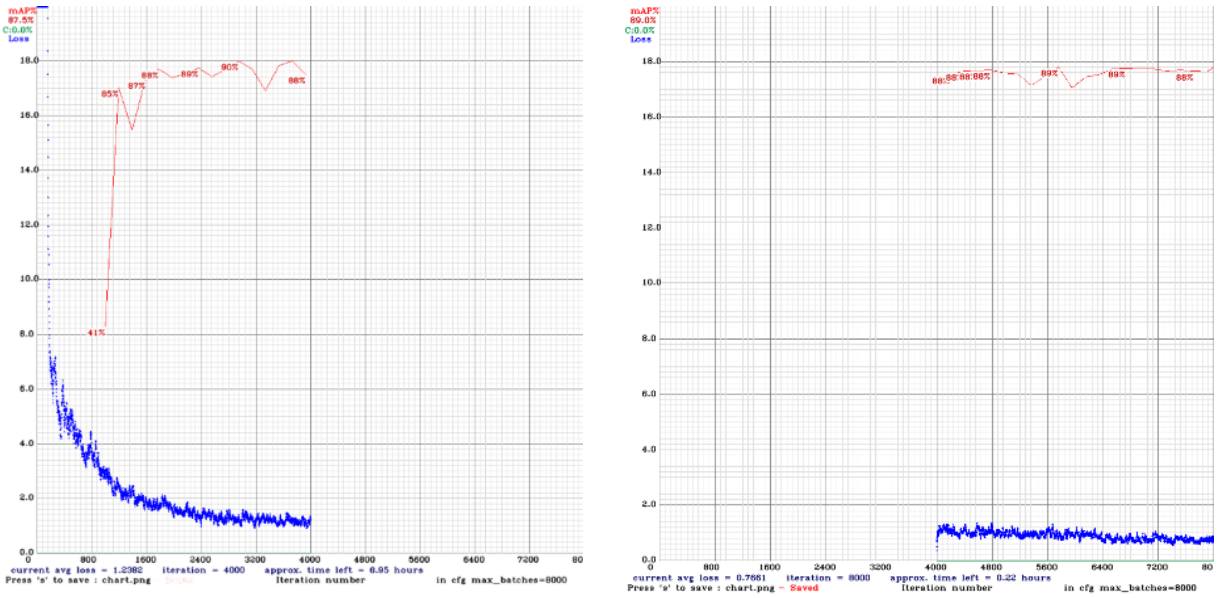
Figure 5. The YOLOv4-tiny Structure

After knowing the best weights model generated from the training and validation process, the next stage is real-time testing. This process involves input images, videos, and webcams in the yolov4 and yolov4-tiny models.

4. Result and Discussions

This training and validation process was carried out twice through Google Colab. The first is the training and validation process using the YOLOv4 configuration. Then proceed with the training and validation process using the YOLOv4-tiny configuration. The configuration parameters used are Batch 64, Subdivision 16, Network Size 416x416, Max_batches 8000, Steps 6400 and 7200, Filters 27, Classes 4. Both of these processes successfully perform 8000 iterations and the total time required to complete the training and validation process using YOLOv4 is 15 hours, while the time needed to complete the training and validation process using YOLOv4-tiny is 2 hours (13 hours faster than the YOLOv4 process). The YOLOv4 model produces the best mAP of 89.59% and an average loss of 0.766, while the best YOLOv4-tiny mAP is at 81.84% and an average loss of 0.218.

Based on the results of the training and validation that has been carried out, the best results from the YOLOv4 model are yolov4-obj_3000.weights with an AP value of Glass 96.89%, Metal 82.56%, Paper 95.60%, Plastic 83.30%. While the best results from the YOLOv4-tiny model are yolov4-obj_best.weights with AP Glass 91.15%, Metal 75.61%, Paper 82.58%, Plastic 78.00%. The Average Loss and mAP for YOLOv4 is shown in Figure 6 and for YOLOv4 Tiny is shown in Figure 7.



(a) (b)

Figure 6. Average Loss and mAP YOLOv4 (a) iteration 1000 – 4000 (b) iteration 4000 – 8000

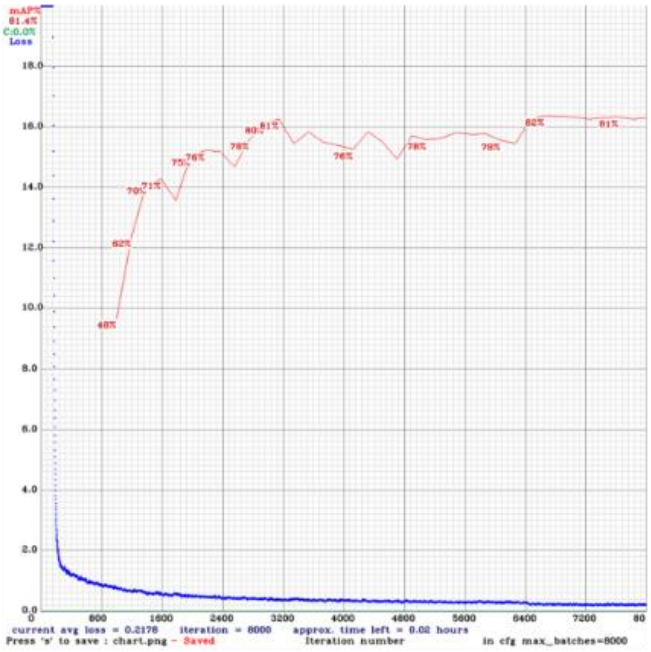


Figure 7. Average Loss and mAP YOLOv4-tiny iteration 1000 – 8000

In previous research, many researcher used Gary Thung's TrashNet Dataset (https://github.com/garythung/trashnet) to solve waste classification problems **(Jardosh et al., 2020; Mittal et al., 2020)**. In this research, the author also use the same dataset, with less class / category, but added the number of images for each class. The comparison of the dataset is shown in Table 2.

Table 2. Dataset Comparison

TrashNet	Dataset in This Research
Glass 501	Glass 953 (+ 452)
Paper 594	Paper 998 (+404)

Cardboard 403	-
Plastic 482	Plastic 900 (+418)
Metal 410	Metal 1019 (+609)
Trash 137	-

When compared with previous research, especially those using TrashNet Dataset, the YOLOv4 model that produced in this study has better accuracy. Research conducted by **(Jardosh et al., 2020)** using the TrashNet Dataset with the YOLOv3 algorithm. This research produces the best mAP value 84.44%, 5.15% lower compare to the model in this research that using YOLOv4 with Mosaic (new data augmentation). Another research also conducted by **(Mittal et al., 2020)** using the TrashNet Dataset which utilizes the CNN algorithm. The accuracy results obtained 87%, 2.59% lower compare to the model in this study that using YOLOv4 with Mosaic (new data augmentation). This study also displays the results of computation speed / predicted time that were not display in 2 previous researches.

After completing the training and validation, the next step is doing real-time testing. The test involves three inputs in which images, videos, and webcams. The three inputs were also testing using the YOLOv4 and YOLOv4-tiny models for comparison. All experiments are under the top weights of YOLOv4 and YOLOv4-tiny, which results from the training and validation process.

4.1. Input Images

Figure 8 shows the test using Google Colab using the command 'detector test' and using threshold 0.5. The input images used are images from each class of glass, metal, paper, and plastic. The results in Figure 8 show that the YOLOv4 prediction probability is better, but the prediction time is less than YOLOv4-tiny. YOLOv4 prediction time is 20.37 ms, while YOLOv4-tiny prediction time is faster with 4.89 ms.

In addition, the classification and detection of image input can also be run on a local laptop / PC which can be seen in Figure 9, with several files needed, such as file weights, .cfg, and obj. names.

Classification and detection of waste with image input can also be done through a website that the author has created using the flask framework and it can be seen in Figure 10.

(a) (b)

Figure 8. Testing Input Images Using Google Colab (a) YOLOv4 dan (b) YOLOv4-tiny

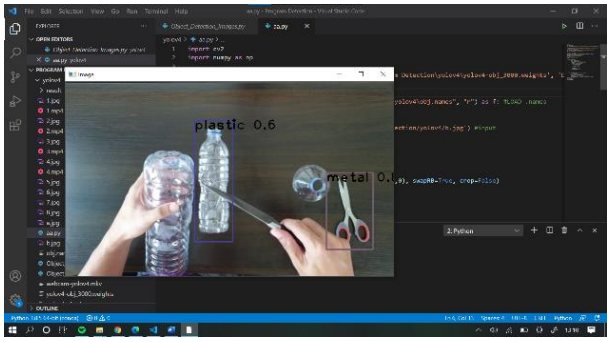


Figure 9. Testing Input Images Using Local PC

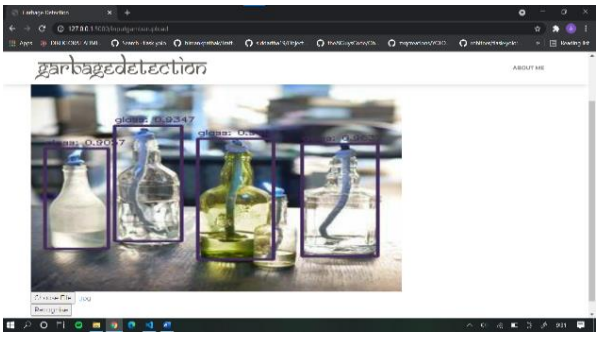


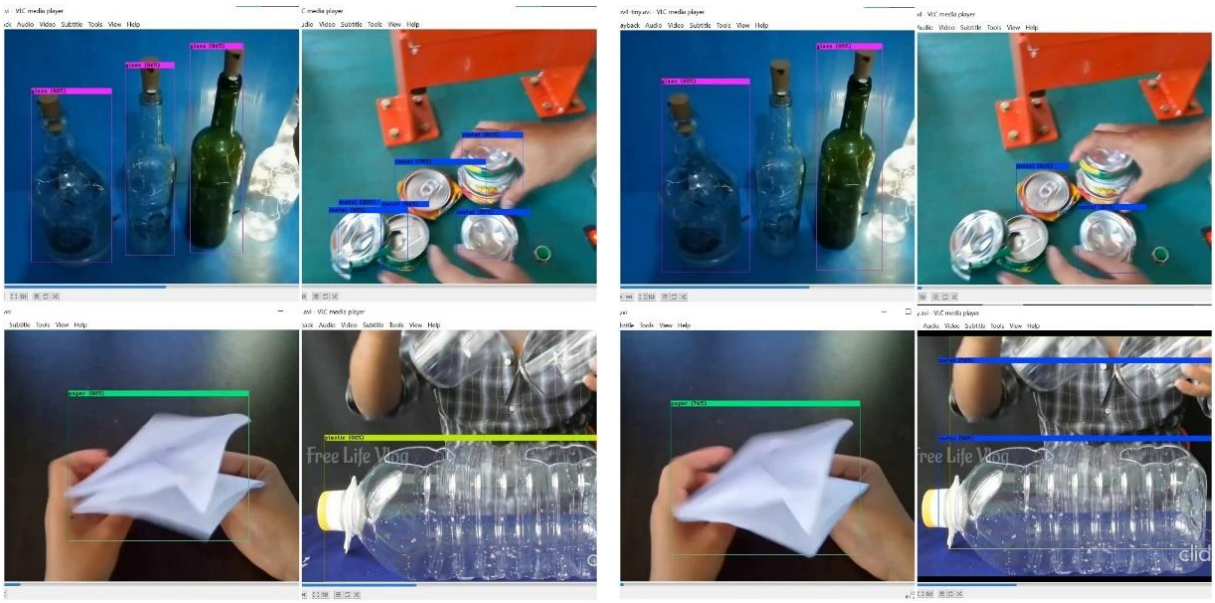
Figure 10. Testing Input Images Using Website

4.2. Input Videos

Figure 11 shows the test using Google Colab using the 'detector demo' command and a threshold of 0.5. The video inputs used pictures from every class of glass, metal, paper, and plastic. The results in Figure 11 also show that the YOLOv4 prediction probability is better, but the prediction time is still less than YOLOv4-tiny. YOLOv4 prediction time is 14s, 29s, 47s, 108s, while YOLOv4-tiny prediction time is faster with 11s, 27s, 39s, 98s. YOLOv4 FPS is 42, 39, 47, 48, while the YOLOv4-tiny FPS is better with 51, 43, 58, 54.

In addition, the classification and detection of videos input can also be run on a local laptop / PC which can be seen in Figure 12, with several files needed, such as files weights, .cfg, and obj.names.

Classification and detection of waste with videos input can also be done through a website that the author has created using the flask framework which can be seen in Figure 13.



(a) (b)

Figure 11. Testing Input Videos Using Google Colab (a) YOLOv4 dan (b) YOLOv4-tiny

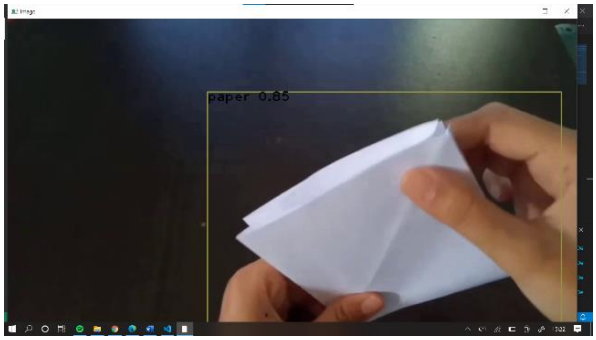


Figure.12 Testing Input Videos Using Local PC

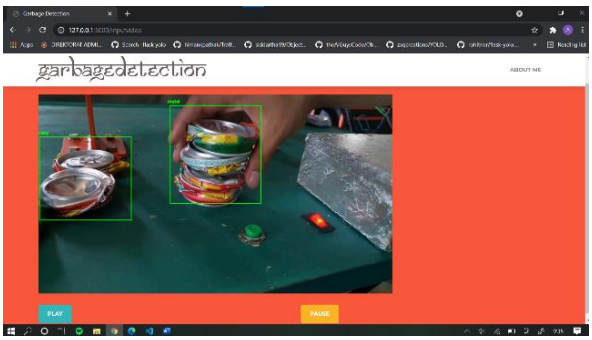
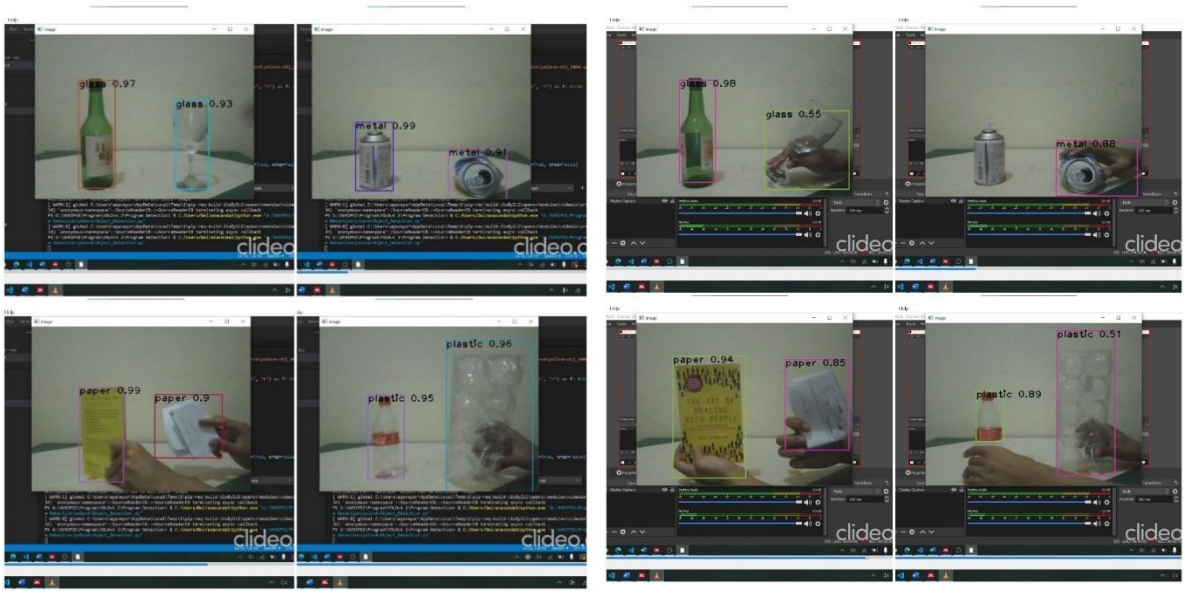


Figure.13 Testing Input Videos Using Website

4.3. Input Webcam

This test was carried out using a local laptop / PC. Some of the files that need to be prepared when testing on a local laptop / PC are the best weights files, .cfg files, obj.names files, various types of waste from all classes, and python files threshold of 0.5. Figure 14 below shows that YOLOv4 performance is still better than YOLov4-tiny in terms of prediction probability. All tested waste was detected with a value above 90%. Meanwhile, when using the YOLOv4-tiny model, real-time testing experienced several problems, such as difficulty recognizing glass cups that were included in the glass class category. The author must change the position of the glass cup until the glass cup can be detected. Then the YOLOv4-tiny model also experienced difficulties in recognizing air fresheners that were included in the metal class. This model is still unable to detect even though the author changes the position of air fresheners, and it even experiences false detection into a glass class. The same thing happens when it comes to detecting paper. The author needs to change the paper's position so that it can be detected in the paper class. Finally, when detecting a plastic class, the two types of waste are detected into the correct class, even though the prediction probability value is not bigger than when it was detected using the YOLOv4 model. By doing real-time testing using webcam input, also proves that the FPS value YOLOv4-tiny is better than the YOLOv4 model. When doing real-time detection, the webcam input with the YOLOv4 model is not that smooth, while using the YOLOv4-tiny model, real-time testing with the webcam input runs smoothly.

Classification and detection of waste using webcam inputs can also be done through a website that the author has created using the flask framework, shown in Figure 15.



(a) (b)

Figure 14. Real-time Testing Input Webcam (a) YOLOv4 dan (b) YOLOv4-tiny

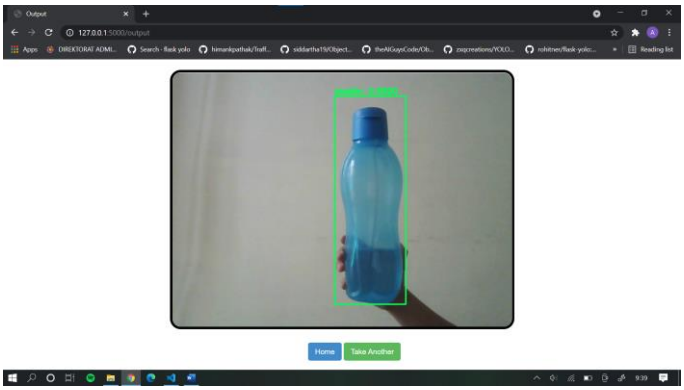


Figure 15. Real-time Testing Input Webcam Using Website

4.4. Subdivision and Mosaic Augmentation Parameters Effect

The author conducted the test by changing the subdivision and mosaic data augmentation parameters in the YOLOv4-tiny model. The configuration parameters used are Batch 64, Network Size 416x416, Max_batches 8000, Steps 6400 and 7200, Filters 27, Classes 4. Table 3 shows that using the subdivision 8 value creates a better mAP value of around 2% compared to the 16 subdivision value. By using mosaic data augmentation, the mAP value is 1% better than without mosaic data augmentation. This shows that smaller subdivision values (8) and using a mosaic data augmentation make the model more optimal.

Besides affecting the mAP value, the subdivision and data mosaic augmentation parameters also affect the computation speed. It can be seen in Table 3, that the smaller the subdivision value, the faster the computation time. In comparison, the model that uses the data mosaic augmentation requires a longer time compared to the model that does not use the augmentation data mosaic.

Table 3. Subdivision and Mosaic Data Augmentation Parameters Effect

Subdivision	Mosaic	mAP (%)	Time (minutes)
16	-	81,84	120
16	√	82,41	240
8	-	84,08	98
8	√	84,50	143

5. Conclusion

In this paper, the authors produce a classification model and real-time image detection of waste that can show the bounding box and prediction probability of objects in each image. The model can be run using Google Colab, local PC / laptop, or with a website built using the Flask framework. The model can also be run using 3 different inputs, images, videos, and a webcam.

The research results show that the YOLOv4-tiny architecture, which is simpler than the YOLOV4 architecture, affects the performance of the algorithm both in prediction probability and prediction time. Where YOLOv4 has better performance in prediction probability. The mAP value reaches 89.59%, precision 0.76, recall 0.90, F1-score 0.82, and Average IoU 64.01%. Meanwhile, YOLOv4-tiny has a faster computation speed when compared to the YOLOv4 model. Although YOLOv4-tiny has good computational performance, its detection ability and prediction probability are not as good as the YOLOv4 model. The YOLOv4-tiny model has an mAP value of 81.84%, precision 0.59, recall of 0.83, F1-score 0.69, and Average IoU 48.35%. The comparison of the model between YOLOv4 and YOLOv4-tiny shows the accuracy speed trade-off, that is, the faster a model detects, it will result in a reduction in the accuracy value and vice versa.

Then, several factors that affect the performance of the YOLO algorithm architecture, namely the smaller the subdivision value used, the mAP value will increase during the training and validation process. With an increase in the range of 2%. Then, by activating the data mosaic augmentation, the mAP value generated during the training and validation process has increased in the range of 1% even though the time needed is longer when compared to not activating the augmentation data mosaic.

Suggestions for future research are to increase the amount of data used. Because the more the number of datasets used, the model's performance will also increase. Then it can increase the number of classes/types of

waste that will be classified and detected in real-time. And future research can use the YOLO (You Only Look Once) algorithm version 5.

References

[1]. Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

[2]. Chen, Weijun, Huang, H., Peng, S., Zhou, C., & Zhang, C. (2021). YOLO-face: a real-time face detector. *Visual Computer*, *37*(4), 805–813. https://doi.org/10.1007/s00371-020-01831-7

[3]. Chen, Wenkang, Lu, S., Liu, B., Li, G., & Qian, T. (2020). Detecting citrus in orchard environment by using improved Yolov4. *Scientific Programming*, *2020*. https://doi.org/10.1155/2020/8859237

[4]. Chinnathurai, B. M., Sivakumar, R., Sadagopan, S., & Conrad, J. M. (2016). Design and implementation of a semi-autonomous waste segregation robot. *Conference Proceedings - IEEE SOUTHEASTCON*, *2016-July*. https://doi.org/10.1109/SECON.2016.7506679

[5]. Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems*, 379–387.

[6]. Deepa, T. P., & Roka, S. (2018). Estimation of garbage coverage area in water terrain. *Proceedings of the 2017 International Conference On Smart Technology for Smart Nation, SmartTechCon 2017*, 347–352. https://doi.org/10.1109/SmartTechCon.2017.8358394

[7]. Devi, R. S. S., Vijaykumar, V. R., & Muthumeena, M. (2018). Waste segregation using deep learning algorithm. *International Journal of Innovative Technology and Exploring Engineering*, *8*(2S), 401–403.

[8]. Du, J. (2018). Understanding of Object Detection Based on CNN Family and YOLO. *Journal of Physics: Conference Series*, *1004*(1). https://doi.org/10.1088/1742-6596/1004/1/012029

[9]. Feng, Y., Tse, K., Chen, S., Wen, C. Y., & Li, B. (2021). Learning-based autonomous uav system for electrical and mechanical (E&m) device inspection. *Sensors (Switzerland)*, *21*(4), 1–23. https://doi.org/10.3390/s21041385

[10]. Gandhewar, N. (2020). Scene Text Detection Using YOLOv4 Framework. *Bioscience Biotechnology Research Communications*, *13*(14), 181–184. https://doi.org/10.21786/bbrc/13.14/43

[11]. Ghiasi, G., Lin, T. Y., & Le, Q. V. (2019). NAS-FPN: Learning scalable feature pyramid architecture for object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 7029–7038. https://doi.org/10.1109/CVPR.2019.00720

[12]. Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 International Conference on Computer Vision*, *ICCV 2015*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

[13]. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

[14]. Gong, B., Ergu, D., Cai, Y., & Ma, B. (2020). A Method for Wheat Head Detection Based on Yolov4. *Research Square*, 1–16.

[15]. He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8691 LNCS*(PART 3), 346–361. https://doi.org/10.1007/978-3-319-10578-9_23

[16]. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 770–778. https://doi.org/10.1109/CVPR.2016.90

[17]. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. http://arxiv.org/abs/1704.04861

[18]. Howard, A., Wang, W., Chu, G., Chen, L., Chen, B., & Tan, M. (n.d.). *Searching for MobileNetV3*.

[19]. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

[20]. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. 1–13. http://arxiv.org/abs/1602.07360

[21]. Jardosh, P. M., Shah, S. S., & Bide, P. J. (2020). SEGRO: Key towards modern waste management. *2020 International Conference for Emerging Technology, INCET 2020*, 8–12. https://doi.org/10.1109/INCET49848.2020.9154113

[22]. Jiang, Z., Zhao, L., Shuaiyang, L. I., & Yanfei, J. I. A. (2020). Real-time object detection method for

embedded devices. *ArXiv*, *3*, 1–11.

[23]. Kumar, S., Yadav, D., Gupta, H., Verma, O. P., Ansari, I. A., & Ahn, C. W. (2021). A novel yolov3 algorithm-based deep learning approach for waste segregation: Towards smart waste management. *Electronics (Switzerland)*, *10*(1), 1–20. https://doi.org/10.3390/electronics10010014

[24]. Li, Xiali, Tian, M., Kong, S., Wu, L., & Yu, J. (2020). A modified YOLOv3 detection method for vision-based water surface garbage capture robot. *International Journal of Advanced Robotic Systems*, *17*(3), 1–11. https://doi.org/10.1177/1729881420932715

[25]. Li, Xiaohan, Lai, T., Wang, S., Chen, Q., Yang, C., & Chen, R. (2019). Weighted feature pyramid networks for object detection. *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, 1500–1504. https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217

[26]. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017). Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-October*, 2999–3007. https://doi.org/10.1109/ICCV.2017.324

[27]. Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8759–8768. https://doi.org/10.1109/CVPR.2018.00913

[28]. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[29]. Liu, Y., Fung, K.-C., Ding, W., Guo, H., Qu, T., & Xiao, C. (2018). Novel Smart Waste Sorting System based on Image Processing Algorithms: SURF-BoW and Multi-class SVM. *Computer and Information Science*, *11*(3), 35. https://doi.org/10.5539/cis.v11n3p35

[30]. Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet V2: Practical guidelines for efficient cnn architecture design. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11218 LNCS*, 122–138. https://doi.org/10.1007/978-3-030-01264-9_8

[31]. Mittal, I., Tiwari, A., Rana, B., & Singh, P. (2020). *Trash Classification: Classifying garbage using Deep Learning*. *11*(July). https://doi.org/10.13140/RG.2.2.19452.56969

[32]. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. (2019). Libra R-CNN: Towards balanced learning for object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*(2), 821–830. https://doi.org/10.1109/CVPR.2019.00091

[33]. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 779–788. https://doi.org/10.1109/CVPR.2016.91

[34]. Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[35]. Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. http://arxiv.org/abs/1804.02767

[36]. Ren, P., Wang, L., Fang, W., Song, S., & Djahel, S. (2020). A novel squeeze YOLO-based real-time people counting approach. *International Journal of Bio-Inspired Computation*, *16*(2), 94–101. https://doi.org/10.1504/ijbic.2020.109674

[37]. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[38]. Sakr, G. E., Mokbel, M., Darwich, A., Khneisser, M. N., & Hadi, A. (2016). Comparing deep learning and support vector machines for autonomous waste sorting. *2016 IEEE International Multidisciplinary Conference on Engineering Technology, IMCET 2016*, 207–212. https://doi.org/10.1109/IMCET.2016.7777453

[39]. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[40]. Shah, P. N., Panigrahi, L., Patel, A., & Tiwari, S. (2020). Classification and Segregation of Garbage for Recyclability Process. *International Journal of Science and Research (IJSR)*, *9*(4), 1757–1761. https://doi.org/10.21275/SR20426184751

[41]. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference*

Track Proceedings, 1–14.

[42]. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 2815–2823. https://doi.org/10.1109/CVPR.2019.00293

[43]. Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 10778–10787. https://doi.org/10.1109/CVPR42600.2020.01079

[44]. Torres-García, A., Rodea-Aragón, O., Longoria-Gandara, O., Sánchez-García, F., & González-Jiménez, L. E. (2015). Intelligent waste separator. *Computacion y Sistemas*, *19*(3), 487–500. https://doi.org/10.13053/CyS-19-3-2254

[45]. Valente, M., Silva, H., Caldeira, J. M. L. P., Soares, V. N. G. J., & Gaspar, P. D. (2019). Detection of waste containers using computer vision. *Applied System Innovation*, *2*(1), 1–13. https://doi.org/10.3390/asi2010011

[46]. Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2020-June*, 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

[47]. Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 5987–5995. https://doi.org/10.1109/CVPR.2017.634

[48]. Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6848–6856. https://doi.org/10.1109/CVPR.2018.00716