Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DETECTING OBJECTS FROM AERIAL IMAGES USING SINGLE-STAGE DETECTION METHOD

[1]ANGELICA FAUSTINE, [2]GEDE PUTRA KUSUMA

[1,2]Computer Science Department, BINUS Graduate Program – Master of Computer Science,
Bina Nusantara University, Jakarta, Indonesia, 11480
E-mail: [1]angelica.faustine@binus.ac.id, [2]inegara@binus.edu

ABSTRACT

One of the applications of object detection and recognition are detecting object from aerial images. There are various further applications of object detection in aerial images, it can be used for tracking objects, implementation for unmanned aerial vehicles (UAV), traffic surveillance, and calculating objects. As the applications of detecting objects from aerial images get wider and more common, the accuracy of object detection becomes more important. This paper evaluates the effects of changing the backbone of YOLOv4 (You Only Look Once), the current state-of-the-art single-stage object detection method, with EfficientNet and EfficientNetv2, the image classification method. The experiment was done with the CARPK dataset, a dataset of aerial images of cars in a parking lot taken by drones that suit car detection. The result of changing the backbone of YOLOv4 with EfficientNetB3 manage to increase the detection accuracy to 99.23% in the CARPK dataset.

Keywords: *Aerial Image, Car Detection, EfficientNet, EfficientNetv2, YOLOv4*

1. INTRODUCTION

One of the processes that a system can do with image data type is object detection and recognition. The data collected from this object detection and recognition process will then be used by the system to do other things. The object detection and recognition process itself has been used a lot in our daily life, such as in our mobile devices where we can use our face to unlock the phone lock, in a smart car where they can detect obstacles around the car, and even on a surveillance camera at a red light which then used by the government to track our plate number when we make a violation. Not only in our daily life, the object detection and recognition processes are also used in the healthcare area to detect abnormalities in bones and even cells. Object detection is also used on aerial images taken from drones to detect corps, cars, planes, and many other things. The important uses of object detection and recognition process make development in this area keeps ongoing.

For a system to run, there must be an algorithm, a bridge that connects the developer's idea to the system brain. In object detection itself, there're 2 types of algorithms which are single-stage algorithms and two-stage algorithms. Each algorithm has its own capabilities wherein single-stage algorithms speed has become their number one priority meanwhile, in two-stage algorithms accuracy has become their top priority. However, with the development of single-stage algorithms that keeps on happening, this algorithm has also increased their accuracy which led to the use of single-stage object detection to be used in real-time. You Only Look Once (YOLO) [1] is a single-stage object detection method that claims to outperform the two-stage detection method Region with CNN features (R-CNN) [2] and Deformable Parts Models (DPM) [3]. YOLO itself has been updated a few times with YOLO9000 [4], YOLOv3 [5], and even YOLOv4 [6]. In every update of YOLO, there are always changes on the backbone part of YOLO which interests a lot of people to try changing the backbone of YOLO.

From this variety of updates in YOLO, we attempt to evaluate the effect of changing the backbone of YOLO on detecting objects from aerial imaging. To do this, we decide to use CARPK [7] dataset, a dataset that contains captured cars in parking lots from drone-view. We decide to use EfficientNet [8], the backbone of EfficientDet [9], and EfficientNetv2 [10], an updated version of EfficientNet as the backbone of YOLOv4. This experiment aimed to detect cars in the CARPK

dataset using the bounding boxes method while trying to increase the detection accuracy or reduce the error score of YOLOv4.

2. RELATED WORKS

In order to find an idea to do an object detection using bounding boxes while increasing the detection accuracy or reducing the error score, we did some research on previously done methods in the object detection area. We collected papers from the last 5 years that do experiment with object detection methods. In a few research that has been done recently [11] [12] [13], it is proved that the current single-stage object detection method has higher accuracy while still able to run faster than the two-stage object detection method. From an experiment that has been done [11], YOLOv3 is found to have higher accuracy than Faster R-CNN in a dataset made out of images containing Hauler, Excavator, and Wheeled Loader. In an experiment done with PASCAL VOC2012 and MSCOCO dataset [12], SSD and SSOD seem to have similar accuracy with Faster R-CNN and Mask R-CNN but have significantly higher speeds than Faster R-CNN and Mask R-CNN. While another experiment on underwater video [13] shows that YOLOv3, Scheme 1, and Scheme 2 have a higher speed detection than Fast R-CNN and Faster R-CNN.

Some attempt to make an updated version on an existing single-stage object detector has been made before like Detection with Enriched Semantics (DES) [14], Recurrent Rolling Convolution (RRC) [15], Gradient Harmonizing Mechanism (GHM) [16], and AlignDet [17] or an attempt to just to increase the accuracy of an exist single-stage object detector like RetinaNet also happen before [18] [19]. In the experiment of making DES [14], the developer decided to use Single Shot Detector (SSD) for the detection and VGG-16 as the backbone in VOC2007, VOC2012, and MS COCO dataset. By adding semantic enrichment and global activation module, both DES300 and DES512 manage to have higher accuracy than SSD300 and SSD512 but have slower detection time.

Another object detection method is RRC [15], a single-stage end-to-end trainable object detection network. In the making process of RRC, the developer decided to use the reduced VGG-16 network as the backbone and train on the KITTI dataset with a few reference settings from SSD as SSD also use a reduced VGG-16 network as the backbone. From the experiment, it is shown that RRC has higher accuracy than SSD.

Another proposed object detection model is GHM [16], a proposed method to train a single-stage object detection model. The developer decides to use GHM-C (Gradient Harmonizing Mechanism into loss for classification) and GHM-R (Gradient Harmonizing Mechanism into loss for regression) with RetinaNet-FPN-ResNet-101 and RetinaNet-FPN-ResNeXt-101 as the backbone. When being trained on the MS COCO dataset, the proposed method with RetinaNet-FPN-ResNeXt-101 achieve higher accuracy than YOLOv3. In the experiment related to the RoIAlign operator and im2col operator, RoIConv operator and AlignDet is made [17]. AlignDet that uses ResNeXt-101 as backbone was trained on the MS COCO dataset managed to get higher accuracy than YOLOv3, SSD, and RetinaNet.

Some experiment intends to increase the performance of RetinaNet has been done before. One attempt to increase the performance of RetinaNet500 is by adding AP-Loss [18] This attempt resulted in the increase of RetinaNet500 accuracy by 3%. This experiment is tested on VOC2007, VOC2012, and MS COCO dataset. Another attempt to increase the performance of RetinaNet is by turning it into a multi-stage object detection which is then named as Cas-RetinaNet [19]. The result showed after training and testing using the MS COCO dataset indicate that Cas-RetinaNet using ResNet-101 as backbone manage to increase the accuracy of RetinaNet by 2%. Both of these experiments manage to surpass YOLOv2 and YOLOv3.

As we may know, YOLO is an example of a single-stage object detection method. The implementation of YOLO itself is quite common, especially in YOLOv2 and YOLOv3. There is an experiment using YOLOv2 with Grozi-3.2k dataset [20] where the experiment is done with modification to detect grocery products in customer use case and management use case. This experiment is done by changing the descriptors, local feature, threshold refinement, and reranking strategy in the category-based. This experiment reported with the most increase of mAP happened when all of the changes were applied.

Another experiment using YOLOv2 in shelf images [21], shows that the amount of iteration, threshold, and class model will give a different result when training. It is proven from the experiment that when the iteration number is too big there is a chance that overfitting will happen, a threshold value that's too big will make the object to be undetectable, and when the threshold value too small it will make wrong objects to be detected, and the number of classes will increase the number of iteration and time needed for training. In one experiment YOLOv3 and

YOLOv2 are used to detect weapons from surveillance system dataset [22]. The result shows that YOLOv3 outperformed YOLOv2 with higher accuracy.

The implementation of object detection in aerial image datasets also has been done a few times before. A few research has been conducted to detect objects on the sea like ships [23] or object on land like humans [24], and even vehicles like cars [25]. In research to detect ships [23], the authors make a comparison of accuracy between YOLOv3 and Mask-RCNN to a dataset that they make by themselves. The authors find that the Mask-RCNN has higher accuracy and faster detection than YOLOv3. In another research to detect humans from Unmanned Aerial Vehicle (UAV) [24], the author improved the accuracy of YOLOv3 on a dataset that they make by making a new backbone called UAV-YOLO.

In research to detect cars from aerial images VEDAI dataset [25], the author proposed a method to improve the performance of YOLOv3 by increasing the number of conv2D_BN_Leaky layers of the res block in the backbone area. The result shows that the proposed method manage to achieve higher accuracy than the original YOLOv3.

From the information that we got from evaluating research papers, we found that changing the backbone of an object detector is a common method to increase the detection accuracy of the said object detector. We also found that single-stage object detection was currently faster and has similar accuracy with two-stage object detection. YOLO was one of the single-stage object detectors that were commonly used and experimented on in the previous version. From this information, we decided to change the backbone of YOLOv4 in an attempt to increase detection accuracy.

3. THEORY AND METHODS

3.1 Object Detection and Recognition

Nowadays Artificial Intelligence (AI) is applied in robots and machines with the ability to think like a human or animal [26]. For this AI to work, it needs data to be inputted in a form of text, sound, video, or images. In a system where the data is inputted as images, the first action that needed to be done is the object detection and recognition process. For a system to be able to do an object detection and recognition process, it needs to go through a deep learning process where the data is no longer explicitly given by the developer. This process will make the machine learn how to process an image into data that they need.

In a learning process, there is a training process where the system will learn to take the information of object location and object name in a set of labeled images by processing it in repeat. The images from the dataset will be inputted in batches size of images which usually called as minibatch. When the whole dataset was trained by one iteration, it become an epoch of training. Usually, when training a dataset, the number of epochs will affect the accuracy of the machine learning. When the number of epochs is too small the model will be underfitted and when the number of epochs is too large the model will be overfitted. In the training process, there will also be a validation process which is done to evaluate the ability of the system in detecting object location and recognizing the object name. This validation process will also be used to determine the system's accuracy in the object detection and recognition process.

In the detecting process, the most common way for a system to detect and recognize an object is by making a bounding box around the said object. In order for the system to make this bounding box, the system will first make grids in the inputted image where each of the boxes inside the grid will have its own identity. This identity will then be used by the system to make possible bounding boxes. From this, the system will have a lot of bounding boxes around 1 object. Each bounding box will then have its own confidence score calculated based on the ground truth box. Using the threshold set by the developer, the best bounding box will then be chosen. This process can be seen in Figure 1. After the bounding box is made, the system will do the recognition process by trying to label the object based on its name. The result of the object detection and recognition process can be seen in Figure 2.

Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

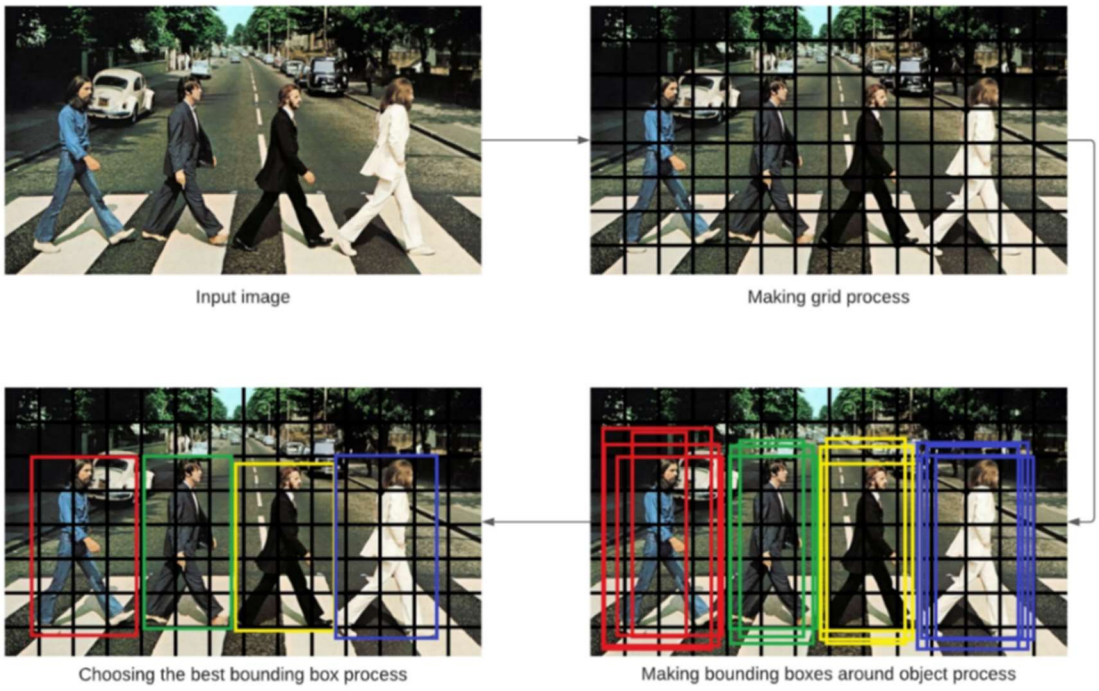

Figure 1. Object Detection Flow

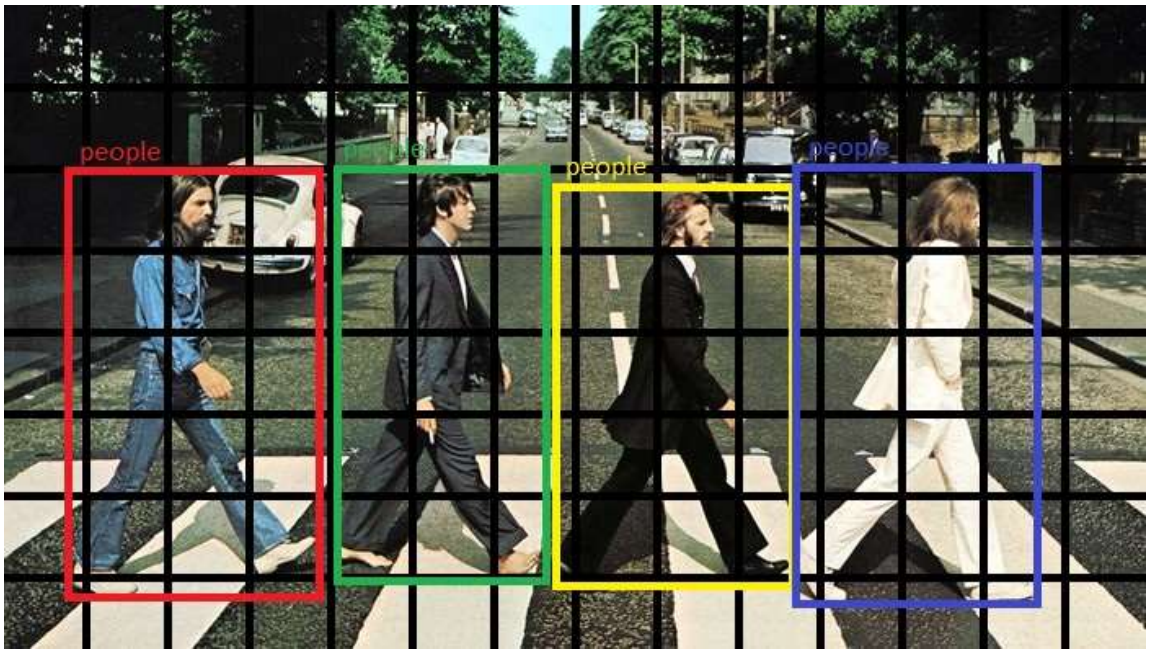

Figure 2. Final Result of Object Detection and Recognition

3.2 YOLO

You Only Look Once (YOLO) is the current single-stage detection method that is still being updated and have a great performance in speed. The detection in YOLO was done by using a moving window and area to detect objects more accurately

and have minimum error. In detecting an object, YOLO will divide the picture into grids and make bounding boxes as in Figure 1. Each bounding box in YOLO will carry object class, x, y, w, and h. The x and y will represent the center coordinate of the bounding box while the w and h represent the width and height of the bounding box. The bounding box generated from object detection will then has a confidence score which is calculated by the Pr(object)*IoU where IoU is the Intersection over Union.

YOLOv1 [1] was made with Googlenet as the backbone and manage to detect objects with 45 FPS on Titan X GPU which is fast at that time. In a faster YOLO version, it even manages to get to 150 FPS. However, YOLOv1 only manage to predict with 2 bounding boxes and 1 class in 1 grid cell. This resulted in YOLOv1 being unable to detect objects that are close to each other and have small sizes. YOLOv1 is also only able to detect 49 objects and has a relatively high localization error.

To update YOLOv1, YOLOv2 [4] was made. YOLOv2 was proposed together with the classification model Darknet-19 as the backbone. YOLOv2 itself was made to improve recall and localization while maintaining the classification accuracy of YOLOv1. To increase the number of objects can that can be detected by 1 grid cell, anchor boxes were used in YOLOv2. With the change of backbone, YOLOv1 and YOLOv2 also have different outputs from the network. If in YOLOv1 the output of the network was 7x7x30, in YOLOv2 the output of the network was 13x13x(anchor boxes*(1+4+20)) which will be 13x13x125 if the anchor boxes are 5.

Even after the making of YOLOv2, the detection of smaller objects is still become an obstacle for YOLO. In YOLOv3 [5] shortcut connection was used to fix this problem. However, this becomes a new problem as YOLOv3 has worse performance in detecting medium and large objects. To do this, there is a change in the backbone area as the developer proposed to be called Darknet-53. Darknet-53 is a hybrid approach of Darknet-19 with some residual network and shortcut connection. To fix the detection based on size, YOLOv3 proposed another method to make the predict boxes. Where the YOLOv1 and YOLOv2 only predict the output from the last layer, YOLOv3 makes 3 times prediction in 3 different scales. YOLOv3 will detect in the scale of 13x13x[3*(4+1+class)], 26x26x[3*(4+1+class)], and 52x52x[3*(4+1+class)] where class is the total number of classes in the dataset. In the case of the Pascal VOC dataset where there are 20 classes, it will be 13x13x75, 26x26x75, and 52x52x75. The 13x13 will be in charge of detecting large objects, 26x26 will be in charge of detecting medium objects and 52x52 will be in charge of detecting small objects.

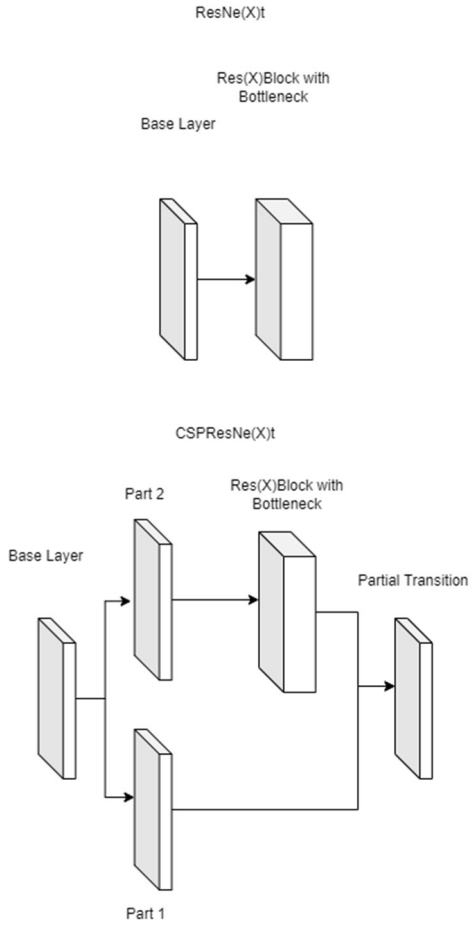

Figure 3 The Application of CSPNet to ResNe(X)t [27]

In the latest update of YOLO, YOLOv4 [6] makes quite a major change from the previous YOLOv3 from the backbone part to the neck part of YOLO. In the backbone part of YOLOv4, there is a slight update that the author did to the previous backbone using Cross Stage Partial Network (CSPNet) [27]. The example of changes using CSPNet can be seen in Figure 3. The method that was done in ResNe(X)t was also done in Darknet-53 with the new proposed name as CSPDarknet-53. YOLOv4 uses CSPDarknet-53 as the backbone. In the neck area, YOLOv4 uses Spatial Pyramid Pooling (SPP) [28], and Path Aggregation Network (PAN) [29]. Meanwhile, for the head area, YOLOv4 still uses the YOLOv3 head. The architecture of CSPDarknet-53 can be seen in Table 1. The proposes of making YOLOv4 itself is to make a powerful YOLO that's more efficient and suitable for single

GPU training while giving Bag of Freebies and Bag of Specials method of object detection.

Table 1 Structure of CSPDarknet-53

Type	Output	Layers
Convolutional	256 x 256	1
DarknetConv2D_BN_Mish	128 x 128	1
csp_resblock_body	128 x 128	1
csp_resblock_body	64 x 64	2
csp_resblock_body	32 x 32	8
csp_resblock_body	16 x 16	8
csp_resblock_body	8 x 8	4
Avgpool		
Connected		
Softmax		

3.3 EfficientNet

EfficentNet [8] is a group of models made out of convolutional neural networks which used a compound scaling method to uniformly increase the depth, width, and resolution of the network. These models are named EfficientNet-B0 until EfficientNet-B7 where the increase of the number at the model's name indicates the increase of the model size. The bigger the size of the model also resulted in the increase of accuracy of the model since the width, depth, and resolution of the network also increase. The EfficientNet uses MBConv blocks in their architecture that consists of a few layers that can be seen in Figure 4. The initial network of EfficientNet-B0 based on the reported paper can be seen in Table 2.

Table 2. Structure of EfficientNet-B0 [8]

Stage (i)	Operator ($\hat{F}_i$)	Resolution ($\hat{H}_i \times \hat{W}_i$)	#Channels ($\hat{C}_i$)	#Layers ($\hat{L}_i$)
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

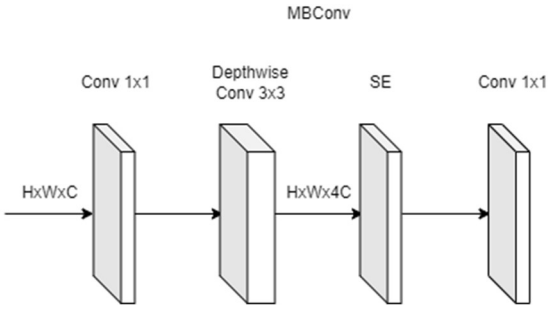

Figure 4 The Architecture of MBConv

To increase the size of the model, the developer of EfficientNet decide to use compound scaling where compound coefficient (ø) was used to increase depth, width, and the resolution of the network using equations 1-5.

$$depth: d = \alpha^{\emptyset}(1) \quad (1)$$

$$width: w = \beta^{\emptyset} \quad (2)$$

$$resolution: r = \gamma^{\emptyset}1 \quad (3)$$

$$s.t. \alpha \times \beta^2 \times \gamma^2 \approx 2 \quad (4)$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \quad (5)$$

3.4 EfficientNetv2

EfficientNetv2 [10] is an updated version of EfficientNet where the author found a way to increase the performance of EfficientNet while reducing the amount of time needed for training. The author proposed a Fused-MBConv as a replacement for some MBConv that was used in the previous version of EfficientNet. The proposed Fused-MBConv can be seen in Figure 5. However, this replacement was only done in a few stages of EfficientNet as the author found that changing all parts of the EfficientNet reduces the performance of the model. The author also changes the way they named the model into EfficientNetV2-S until EfficientNetV2-M/L. To make this variety of models, the author still uses the same scaling method as the old EfficientNet. The structure of EfficientNetV2-S can be seen in Table 3.

Table 3. Structure of EfficientNetV2-S [10]

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6

5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

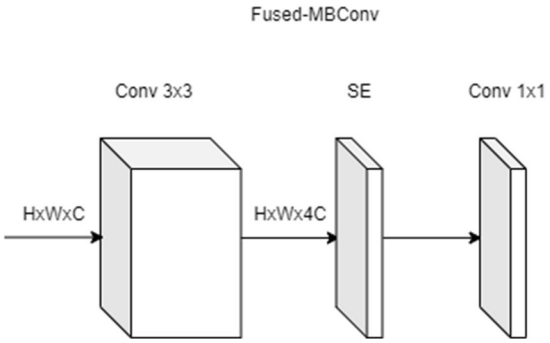

Figure 5 The Architecture of Fused-MBConv

4. PROPOSED METHOD

The dataset that was used in this experiment is CARPK [7] dataset. This dataset consists of aerial images of cars in 4 different parking lots taken by the drone. This dataset contains nearly 90000 cars with bounding box annotation. The bounding boxes given were recorded with top-left points and the bottom-right points. The detected cars from the dataset include cars on the edge of the image. This dataset was usually used as a counting object dataset. The dataset was divided into 989 training images and 459 testing images.

To detect cars, we use a single-stage detection method YOLOv4. An object detector architecture is usually divided into head, neck, and backbone. Usually, the backbone is in charge of the feature extractor, while the head is in charge of the object detection itself which will result in the making of bounding boxes and classification confidence. The neck however is in charge of processing the feature extracted by the backbone to connect with the object detection in the head area. Changing the backbone of an object detector is something that has been commonly done by other developers to improve the accuracy or speed of an object detector.

This is the reason why we want to evaluate the effect of changing the backbone of YOLOv4 in detecting cars from aerial images. The backbone of YOLOv4 is called CSPDarknet-53. To change this backbone, we decide to look at EfficientNet, the backbone of EfficientDet that has higher accuracy than YOLOv4 and EfficientNetV2. The reported result of EfficientNet and CSPDarknet-53 when being trained and tested on ImageNet shows that EfficientNet achieves better accuracy than CSPDarknet-53. Based on these results, we decided to try to use EfficientNet-B3 to replace CSPDarknet-53 as the backbone of YOLOv4. We also attempt to use EfficientNetV2 as the backbone of YOLOv4 as EfficientNetV2 was reported to have better results and run faster than the old EfficientNet. The result of classification on the ImageNet dataset of CSPDarknet-53 and EfficientNet can be seen in Table 4 while the result of EfficientNetV2 can be seen in Table 5. In the experiment of detecting objects and counting cars on the CARPK dataset, we decide to evaluate a few proposed models including the original YOLOv4. The architecture of the original YOLOv4 using CPSDarknet-53 with input images size 416x416 can be seen in Figure 6. The SPP and PANet in the architecture indicate the neck part of YOLOv4. From the architecture, we can see that in the head part of YOLOv4, the prediction was done in 3 different scales resolution just like the head part of YOLOv3.

From the feature extraction that happened on the CSPDarknet-53, there is concatenation from CSP_Resblock_Body-3 and CSP_Resblock_Body-4 as the output of CSP_Resblock_Body-3 has a resolution of 52x52 and the output of CSP_Resblock_Body-4 has a resolution of 26x26. Both of these concatenations happen at the end of the block.

Table 4. The Performance of CSPDarknet-53, and EfficientNet on ImageNet [30]

	CSPDarknet-53	B0	B1	B2	B3	B4	B5	B6	B7
Top-1	80.1	77.1	79.1	80.1	81.6	82.9	83.6	84.0	84.3
Top-5	95.1	93.3	94.4	94.9	95.7	96.4	96.7	96.8	97.0

Table 5. The Performance of EfficientNetv2 on ImageNet

	S	M	L	B0v2	B1v2	B2v2	B3v2
Top-1	83.9	85.2	85.7	78.7	79.8	80.5	82.1

Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

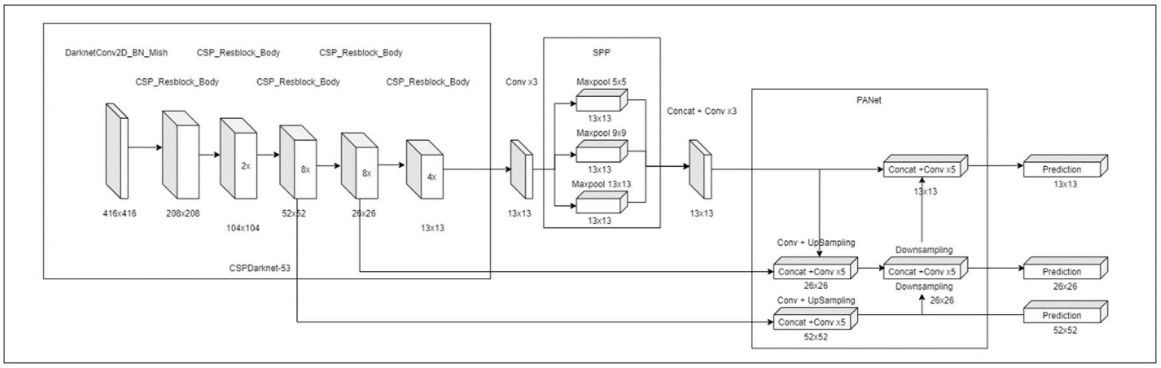

Figure 6. YOLOv4 Architecture with CSPDarknet-53 as The Backbone

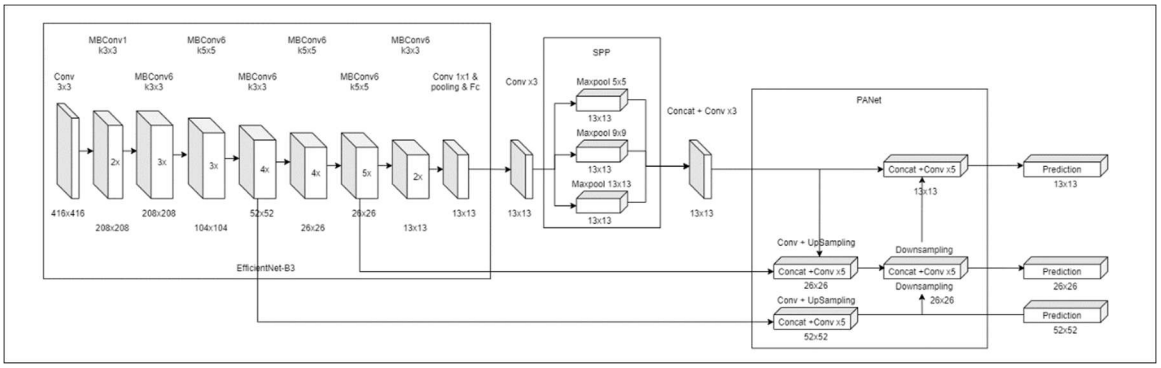

Figure 7. YOLOv4 Architecture with EfficientNet-B3 as The Backbone

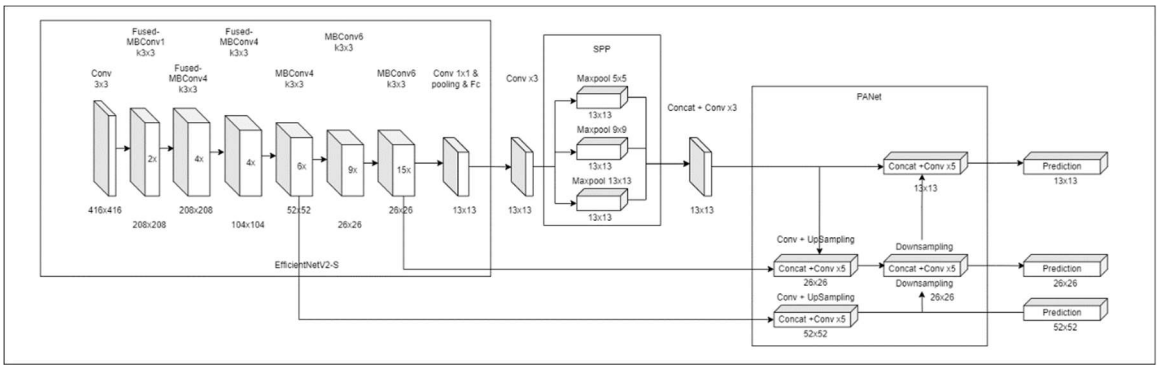

Figure 8. YOLOv4 Architecture with EfficientNetV2-S as The Backbone

As stated previously, our proposed method was to change the backbone area of YOLOv4, thus from the original YOLOv4, we only change the architecture of CSPDarknet-53. From the original architecture of YOLOv4, we can see that the image scale that was used to be extracted into the prediction process are the image with scale size 52x52, 26x26, and 13x13. Thus we decide to use the same image scale to extract the prediction from our proposed model. The architecture of YOLOv4 with EfficientNet-B3 as the backbone with input image size 416x416 can be seen in Figure 7. Since the architecture of EfficientNet is different from CSPDarknet-53, there are 2 proposed models that we try on EfficientNet-B3.

The first proposed models make a concatenation from the end of block MBConv6-3 to get the 52x52 resolution and the end of block MBConv6-5 to get the 26x26 resolution. The second proposed models make a concatenation from the middle of MBConv6-4 to get the 52x52 resolution and the middle of MBConv6-6 to get the 26x26 resolution.

Another experiment that we did was to change the backbone of YOLOv4 with EfficientNetV2-s. The architecture of YOLOv4 with EfficientNetV2-S as the backbone with input image size 416x416 can

be seen in Figure 8. As EfficientNetV2 has a similar architecture with EfficientNet, we also tried 2 proposed models with EfficientNetV2-S.

On the first proposed model, we use concatenation from the end of block Fused-MBConv4-3 to get the 52x52 resolution and the end of block MBConv6-5 to get the 26x26 resolution. For the second proposed model, we use concatenation from the middle of block MBConv4-4 to get the 52x52 resolution and the middle of block MBConv6-6 to get the 26x26 resolution.

5. RESULT AND DISCUSSION

In the training process, we implement YOLOv4 and our proposed method in the Keras environment. The whole training and testing process was run on the cloud in Kaggle. For each model, we used 8 batch sizes, SGD optimizer with no momentum, and a 0.001 learning rate. In the training process, we used the transfer learning method with a total of 40 epochs while freezing the backbone, and then continue training after unfreezing the backbone layer up to 250 epochs. We also applied a reduced learning rate with the patience of 10 epochs and early stopping with the patience of 50 epochs. For training, we shuffle and split a total of 989 training image into 80% of training dataset and 20% of validation dataset. While the testing dataset was consisted of 459 images. The size of the input image used in this experiment was 416x416 as this input size was the input size recommended by YOLOv4. When we test our model accuracy, we use 0.5 threshold.

From the implementation result of object detection using YOLOv4 and the proposed method, we also calculate the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of the model by calculating the difference number of predicted cars and number of ground truth cars in each test image using equation 6 and equation 7.

$$MAE = \frac{1}{n}\sum_i^n |f_i - y_i| \quad (6)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_i^n (f_i - y_i)^2} \quad (7)$$

The result of implementing YOLOv4 with the proposed model on the CARPK dataset when tested on the validation dataset is shown in Table 6 while the result, when tested on the testing dataset, is shown in Table 7.

Table 6. The mAP, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) from Validation Dataset

Model	mAP%	MAE	RMSE
YOLOv4	98.60	0.53	1.14
YOLOv4+EfficientNetB3 (end of the block)	98.79	0.57	1.26
YOLOv4+EfficientNetB3 (middle block)	**99.23**	**0.51**	**1.08**
YOLOv4+EfficientNetV2-S (end of the block)	96.88	1.01	1.89
YOLOv4+EfficientNetV2-S (middle block)	98.29	0.68	1.44

Table 7. The mAP, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) from Testing Dataset

Model	mAP%	MAE	RMSE
Layout Proposal Network [7]	-	23.80	36.79
YOLOv4	**74.91**	21.65	28.07
YOLOv4+EfficientNetB3 (end of the block)	70.05	23.40	29.41
YOLOv4+EfficientNetB3 (middle block)	74.22	**21.23**	27.69
YOLOv4+EfficientNetV2-S (end of the block)	63.93	25.58	30.98
YOLOv4+EfficientNetV2-S (middle block)	73.62	21.25	**27.52**

It is shown that there are big differences in mAP values when we do validation and testing. From what we learned, the dataset of training and testing turn out to be very different. The example of the training dataset can be seen in Figure 9 while the example of the testing dataset can be seen in Figure 10.

Figure 9 The Example of Training Image of CARPK Dataset

Figure 10 The Example of testing Image of CARPK Dataset

The effect of such different images makes the model have lower confidence in detecting the object. The contrast of the image also causes the model to have difficulty detecting the cars. However, the proposed models still have a good MAE and RMSE score as it's lower than the result given by the reported MAE and RMSE score of LPN.

6. CONCLUSION & FUTURE WORKS

From this experiment, we found that the detection result of YOLOv4 was initially good. The change of backbone to EfficientNetB3 and EfficientNetV2-S shows that there's a slight increase in the accuracy on validation and a slight decrease on MAE and RMSE. From this experiment, we manage to show that there is a possibility of an increase in YOLOv4 performance by changing the backbone area. We also managed to achieve the highest accuracy in validation with EfficientNetB3 middle block and achieved the lowest MAE and RMSE on testing by using the EfficientNetB3 middle block version as the backbone of YOLOv4. There is some limitation in our experiment such as the small number of dataset and the inconsistency between the training data and the testing data. As the original weight of EfficientNetV2 wasn't formatted in the Keras version, we might also not be using the latest weight version. These limitations cause our testing results lower than the validation result. In the future, we would like to try changing the backbone part of YOLOv4 with other classification models that were better than EfficientNetB3 and EfficientNetV2-S. We also would like to try to use the model that has been enhanced to be used on different datasets.

REFERENCE:

[1] J. Redmon, S. Divvala, R. Girschick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016,* pp. 779-788.

[2] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014,* pp. 580-587.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, 2010, pp. 1627-1645.

[4] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017,* pp. 6517-6525.

[5] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv,* 2018.

[6] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv,* 2020.

[7] M.-R. Hsieh, Y.-L. Lin and W. H. Hsu, "Drone-based Object Counting by Spatially Regularized Regional Proposal Networks," *ICCV,* 2017.

[8] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, PMLR, 2019, pp. 6105-6114.

[9] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020,* pp. 10778-10787.

[10] M. a. L. Q. Tan, "EfficientNetV2: Smaller Models and Faster Training," *Proceedings of the 38th International Conference on Machine Learning,* vol. 139, 2021, pp. 10096--10106.

[11] S. K. Chadalawada, "Real Time Object Detection and Recognition Using Deep Learning Methods," *Digitala Vetenskapliga Arkivet (DiVA),* 2020.

[12] B. Qiang, R. Chen, M. Zhou, Y. Pang, Y. Zhai and M. Yang, "Convolutional Neural Networks-Based Object Detection Algorithm

by Jointing Semantic Segmentation for Images," *MDPI Open Access Journals,* 2020.

[13] F. Han, J. Yao, H. Zhu and C. Wang, "Underwater Image Processing and Object Detection Based on Deep CNN Method," *Hindawi,* 2020.

[14] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang and A. L. Yuille, "Single-Shot Object Detection with Enriched Semantics," *arXiv,* 2017.

[15] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai and L. Xu, "Accurate Single Stage Detector Using Reccurent Rolling Convolution," *IEEE Explore,* 2017.

[16] B. Li, Y. Liu and X. Wang, "Gradient Harmonized Single-stage Detector," *arXiv,* 2018.

[17] Y. Chen, C. Han, N. Wang and Z. Zhang, "Revisiting Feature Alignment for One-stage Object Detection," *arXiv,* 2019.

[18] K. Chen, J. Li, W. Lin, J. See, J. Wang, L. Duan, Z. Chen, C. He and J. Zou, "Towards Accurate One-Stage Object Detection with AP-Loss," *arXiv,* 2019.

[19] H. Zhang, H. Chang, B. Ma, S. Shan and X. Chen, "Cascade RetinaNet: Maintaining Consistency for Single-Stage Object Detection," *arXiv,* 2019.

[20] A. Tonioni, E. Serra and L. D. Stefano, "A deep learning pipeline for product recognition on store shelves," *IEEE Explore,* 2018.

[21] C. G. Melek, E. B. Sonmez and S. Albayrak, "Object Detection in Shelf Images with YOLO," *IEEE explore,* 2019.

[22] S. Narejo, B. Pandey, D. Esenarro vargas, C. Rodriguez and M. R. Anjum, "Weapon Detection Using YOLO V3 for Smart Surveillance System," *Hindawi,* vol. 2021, 2021.

[23] D. Zhao and X. Li, "Ocean Ship Detection and Recognition Algorithm," *2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC),* 2020.

[24] M. Liu, X. Wang, A. Zhou, X. Fu, Y. Ma and C. Piao, "UAV-YOLO: Small Object Detection on Unmanned Aerial Vehicle Perspective," *Sensors,* vol. 20, 2020.

[25] B. Xu, B. Wang and Y. Gu, "Vehicle Detection in Aerial Images Using Modified YOLO," *2019 IEEE 19th International Conference on Communication Technology,* 2019.

[26] Z. Mohammed, "Artificial Intelligence Definition, Ethics and Standards," *ResearchGate,* 2019.

[27] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh and I.-H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020,* pp. 1571-1580.

[28] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, 2015, pp. 1904-1916.

[29] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, "Path Aggregation Network for Instance Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759-8768.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein and e. al, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, no. 3, 2015, pp. 211-252.