The Improvement Impact Performance of Face Detection Using YOLO Algorithm

Rakha Asyrofi
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
asyrofi.19051@mhs.its.ac.id

Yoni Azhar Winata
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
yoni.19051@mhs.its.ac.id

***Abstract*— Image data augmentation is a way that makes it possible to increase the diversity of available data without actually collecting new data. In this study, researchers have evaluated the application of image manipulation with the Thatcher effect, double illusion, and inversion on the performance of face detection for data augmentation needs where the data obtained has a weakness that is the limited amount of data to create a training model. The purpose of this research is to increase the diversity of the data so that it can make predictions correctly if given other similar datasets. To perform face detection on images, it is done using YOLOv3 then comparing the accuracy results from the dataset after and before adding data augmentation.**

***Keywords*— Thatcher Effect, Augmentation, YOLO, Object Detection.**

I. Introduction

In the field of computer vision, object detection has been widely applied in various areas, such as the medical field, activity recognition, security monitoring in companies, and automatic control robots. Popular object detection methods, at first, were known by using feature extraction methods such as Histogram of Oriented Gradients (HOG), Speeded-Up Robust Features (SURF), Local Binary Patterns (LBP), Haar wavelets, and also Color histograms. The process of capturing the characteristics of objects that can describe the characteristics of objects is the primary technique in the feature extraction method [1].

The object detection method using feature extraction has a problem related to achieving the best performance. It also has various other challenges, such as how to make feature extraction scalable when dealing with large amounts of data [2], [3].

Deep learning is part of the machine learning method in which it implements the concept of learning done by computers by mimicking the workings of the human brain. This concept is known as Artificial Neural Networks. At this time, deep learning is often used in object detection and can solve problems and challenges in feature extraction methods.

Although in various studies, the detection of deep learning objects can overcome obstacles that occur in the feature extraction method, but in fact, deep learning also has its challenges. First, to work well, deep learning is often carried out on large numbers of interconnected neurons so that it will impact on the need for large resources too and usually requires a long time for training [4]. Second, problems with overfitting and underfitting are typical when creating learning models. Third, there is a lack of training data [5], [6].

The main problem solved in this paper is to compare the appropriate manipulation techniques in face detection to add image data to the training model process so that it can identify the most suitable technique to use in this problem. In addition, a program function has been created to implement image manipulation.

II. Literature Review

A. Related Work

Artificial intelligence, at this time, has been developed specifically for object detection and object recognition by using machine learning and even deep learning. In its development, deep learning began from the development of ANN to CNN. CNN contains a convolution layer consisting of a non-linearity layer and a pooling layer. CNN architecture also has a filter that is useful when training to estimate the error value. The filter will be run using the update system, where after the update process is complete, the CNN will be issued. This CNN method has often been used in various studies [4], [7]–[9]. After that, there were developments from other CNN. Among these architectural developments are SPP-Net, R-CNN, Fast CNN, Faster R-CNN, and YOLO[10].

Object detection algorithms generally use regions to process objects in an image [11]. In other words, the working network does not see the image as a whole, and the part of the image that has the highest probability is considered to contain the detected object. YOLO or You Only Look Once is an object detection algorithm that is different from the region based algorithm where it uses a convolutional network to predict bounding boxes and their class probabilities [12].

YOLO v2 uses a custom deep architecture Darknet-19, the 19 layer networks that are originally equipped with 11 more layers for object detection. With 30 layer architectures, YOLO v2 often has difficulty with the detection of small objects. YOLO v3 uses the Darknet variant, which originally had 53 network layers on ImageNet. For object detection, 53 other layers are stacked on top of it, the architecture consisting of 106 layers makes it fully convolutional in YOLO v3. It is the reason behind the slowness of YOLO v3 compared to another YOLO version [10], [13].

In general classification is the process of identifying labels from the data being tested, whereas in YOLO, classification is by localization. There is an additional assignment of object locations in the form of bounding boxes (b_x, b_y, b_h, b_w), as in Figure 1. b_x and b_y are bounding annotation from the center of the object detected, b_h and b_w are bounding box height and weight annotations. YOLO algorithm stages can be seen in the architecture in Figure 2.

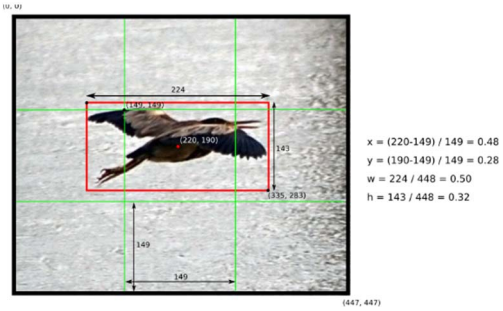

Figure 1. Coordinate *x,y,w,h* from detection box of the grid

Input image on the YOLO architecture is an image with a size of 448 * 448. It created a grid on the image with size *S* * *S* grids. If *S* = 7, then each grid cell will be 64 * 64 and have 49 grid cells.

Images also have confidence cf or class of objects. Suppose *nB* = 1 is a human, *nB* = 2 is a car. Each grid cell contains 5 values, which are the location of the *x* coordinate is assumed based on rows *(x_br)*. *y* coordinates based on columns *(y_kol)*, size and confidence values (*x, y, w, h, c*) of the existing *bbox nB*, which is *B1* and *B2* [13].

$$C(Object) = \Pr(Object) * IOU(Pred, Truth) \quad (1)$$

In equation 1, confidence *C* represents the number of bboxes. Pr indicates whether there are objects in the cell. *IOU* is Intersection Over Union or overlapping rates. *C* will always be 0 when the grid cell is background.

$$IOU(Pred, Truth) = \frac{area(box_{truth}) \cap area(box_{pred})}{area(box_{truth}) \cup area(box_{pred})} \quad (2)$$

In equation 2, the box area is the prediction of each box, if the value is low it will be deleted based on the threshold. If there are 2 bboxes referring to different classes or because there are overlapping objects, the tensor size results are (S * S * (*nB* * 5 + 2 * *nC*)) = (7 * 7 * (2 * 5 + 2 * 2)) = (7 * 7 * 14) tensors. Where, the 1st bbox and the 2nd bbox, refer to different object classes.

The loss function used in the YOLO V3 architecture is Mean Square Error (MSE), where it will contain 3 main things. 3 The main thing is classification error, coordinate error, and *IOU* error.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left\| {}^{obj}_{ij} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \right. \quad (3)$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left\| {}^{obj}_{ij} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \right.$$

B. The detection principle of YOLO

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \left\| {}^{obj}_{ij} (C_i - \hat{C}_i)^2 \right. \quad (4)$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left\| {}^{obj}_{ij} (C_i - \hat{C}_i)^2 \right. \quad (5)$$

$$+\sum_{i=0}^{S^2} \left\| {}^{obj}_{i} \sum_{C \in classes} (p_i(c) - \hat{p}_i(c))^2 \right.$$

From equation 3, there is a Localization Loss where it is the size of the error of the predicted bbox location and size. In the detection, only what is detected is taken into account. $1_{ij}^{obj} = 1$ when *j* from bbox in cell i detect objects in the grid. If it does not exist, then the value is set to 0. λ_{coord} will increase when the burden of the loss in the coordinates of the bbox also increases.

When an object in the cell grid is detected then its Confidence Loss is measured in equation 4. C_i is the socre confidence box of bbox j in cell grid i. $1_{ij}^{obj} = 1$ when *j* from bbox in cell *i* detects an object in the grid. If there isn't, then the value is set to 0.

If the object is not detected, then according to equation 5, the value is $\lambda_{noobj} \sum_{i=0}^{S2} \sum_{j=0}^{B}$ where Ci is the box Confidence socre of box *j* in cell grid *i* and λ_{noobj} is the weight down and loss when detecting the background of the image [10], [13].

III. Implementation

A. Face Detection Using YOLOv3

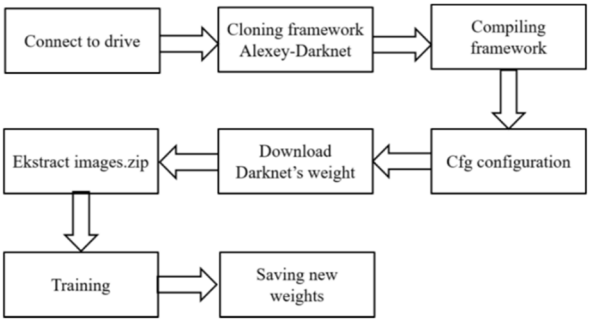

Figure 2. YOLOv3 Training Architecture Using Google Colab

In this study, Figure 2 explain the deep learning architecture used is YOLOv3. YOLOv3 uses the Darknet variant, which initially has 53 network layers on Imagenet. For object detection, another 53 layers are stacked on top of it. An architecture consisting of 106 layers makes YOLOv3 fully convolutional. Insert image in YOLO architecture is an image with size 416x416. After the image is inserted, a grid will be created on the image with SxS size [12]. The grid of each cell consists of x, y, w, h and has the confidence of the object. If there is confidence, it is defined in equation (1).

B. Augmentation Data

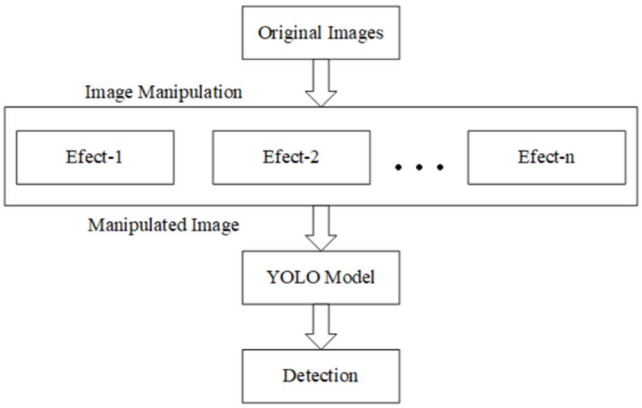

Figure 3. Process Block Diagram

Effect-1 is image manipulation using the Tatcher effect, effect-2 is Color Shifting (RGB), effect-3 is mixing images, while effect-5 is a combination of several manipulation effects. The experiment was designed to obtain a large number of samples from a face image with a small amount of data without compromising the integrity of the original image.

The implementation according to Figure 3, The image effect application has been implemented using libraries in Python. The results from the Image Data Generator have been used by inserting the original image into the generator with the aim of increasing the amount of data. The result of the implementation of this technique, the number of images with manipulated objects, is gradually increased, from 500 training data to 1000 images. For mixing images the effect is implemented by detecting the eyes, nose and mouth on the face using Haar Cascade then pasting cropped images such as facial accessories using the library from OpenCV.

C. Dataset

The experiment in this study was to use the Celeb dataset from the Kaggle repository. The dataset consists of more than 200000 human face data where only 500 data will be taken. The dataset is divided into 70% for training, 15% for validation, and 15% for testing. After that, for the data manipulation process, 70% of the 500 training data will be implemented with some image effects. In training conducted, the image data manipulation algorithm was implemented. The iteration is carried out as many as 3000 for the training process. The learning rate is 0.001, and the learning level is reduced at 5000 and 20000 times.

D. Experimental Environment

This research has been conducted in an Experimental environment using an Intel Core i5-8250U, CPU is 1.60GHz, GPU is MX250, 8GM RAM. The operating system running on a computer is Windows 10 Professional, 63-bit. While the architecture used is YOLO v3 with the framework is Darknet and the syntax is made using Python 3.

E. Image Manipulitaion For Data Augmentation

Based on experiment 1 where a dataset of 500 images without manipulating the image, it can detect several facial images in the form of paintings or comics but often it is also wrong in detecting facial objects. While experiment 2 with 500 images of data and image manipulation to 1000 images, has an increase in the detection rate of facial objects that is not too significant. Based on this experiment, according to Figure 4 to Figure 6, to detect facial objects, whether original images, comics, or writing, more than 1000 training data is needed and the data must be varied and at least reflect what kind of facial object you want to get.

Figure 4. Original Image Experiment Result, Non Augmentation – Augmentation – WIscDER pre-trained

Figure 5. Comic Experiment Result, Non Augmentation – Augmentation – WIDER pre-trained

Figure 6. (a) Art Painting Experiment Result, Non Augmentation – Augmentation – WIDER pre-trained

IV. Conclusion and Evaluation

The conclusions obtained are based on the confidence value of the testing results. The experimental results resulted in the fact that with data augmentation, the value of its confidence increased, although not significantly. Experiments using the Wider dataset tend to be able to detect real faces more accurately than using the Celeb dataset. This might happen because Wider has more variations in shooting angles than Celeb. Experiments using the Wider dataset can detect small objects, while using the Celeb dataset cannot detect small face objects.

References

[1] A. Priadana and M. Habibi, "Face detection using haar cascades to filter selfie face image on instagram," *Proceeding - 2019 Int. Conf. Artif. Intell. Inf. Technol. ICAIIT 2019*, pp. 6–9, 2019, doi: 10.1109/ICAIIT.2019.8834526.

[2] B. Meidyani and H. Ijandrasa, "Hybrid Denoising Development to Improve the," 2019.

[3] H. Zeng, F. Qin, and K. Lin, "An Optimized Face Detection Based on Adaboost Algorithm," *Proc. 2018 Int. Conf. Inf. Syst. Comput. Aided Educ. ICISCAE 2018*, pp. 375–378, 2019, doi: 10.1109/ICISCAE.2018.8666925.

[4] H. Jiang and E. Learned-Miller, "Face Detection with the Faster R-CNN," *Proc. - 12th IEEE Int. Conf. Autom. Face Gesture Recognition, FG 2017 - 1st Int. Work. Adapt. Shot Learn. Gesture Underst. Prod. ASL4GUP 2017, Biometrics Wild, Bwild 2017, Heteroge*, pp. 650–657, 2017, doi:

10.1109/FG.2017.82.

[5] S. Bang, F. Baek, S. Park, W. Kim, and H. Kim, "Image augmentation to improve construction resource detection using generative adversarial networks, cut-and-paste, and image transformation techniques," *Autom. Constr.*, vol. 115, no. September 2019, p. 103198, 2020, doi: 10.1016/j.autcon.2020.103198.

[6] M. A. Kutlugun, Y. Sirin, and M. Karakaya, "The effects of augmented training dataset on performance of convolutional neural networks in face recognition system," *Proc. 2019 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2019*, vol. 18, pp. 929–932, 2019, doi: 10.15439/2019F181.

[7] Q. Q. Tao, S. Zhan, X. H. Li, and T. Kurihara, "Robust face detection using local CNN and SVM based on kernel combination," *Neurocomputing*, vol. 211, pp. 98–105, 2016, doi: 10.1016/j.neucom.2015.10.139.

[8] H. C. Shin *et al.*, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016, doi: 10.1109/TMI.2016.2528162.

[9] R. Xie, Q. Zhang, E. Yang, and Q. Zhu, "A method of small face detection based on CNN," *Proc. - 2019 4th Int. Conf. Comput. Intell. Appl. ICCIA 2019*, pp. 78–82, 2019, doi: 10.1109/ICCIA.2019.00022.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[11] J. J. Lv, X. H. Shao, J. S. Huang, X. D. Zhou, and X. Zhou, "Data augmentation for face recognition," *Neurocomputing*, vol. 230, pp. 184–196, 2017, doi: 10.1016/j.neucom.2016.12.025.

[12] Y. Wang and J. Zheng, "Real-time face detection based on YOLO," *1st IEEE Int. Conf. Knowl. Innov. Invent. ICKII 2018*, vol. 2, pp. 221–224, 2018, doi: 10.1109/ICKII.2018.8569109.

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