Vol.6 No.1 Tahun 2020
Jurnal Teknologi dan Manajemen Informatika
http:// http://jurnal.unmer.ac.id/index.php/jtmi
P-ISSN: 1693-6604 E-ISSN: 2580-8044

Detection of Militia Object in Libya by Using YOLO Transfer Learning

Hatem Alsadeg Ali Salim [1], Yosi Kristian [2], Endang Setyati [3]

[1,2,3] Institut Sains dan Teknologi Terpadu Surabaya Indonesia

Info Artikel

Riwayat Artikel
Diterima: 05-03-2020
Disetujui: 26-03-2020

Kata Kunci
YOLO;
Daknet;
Tensorflow;
Militia

✉ Corresponding Author
Yosi Kristian
Institut Sains dan Teknologi Terpadu Surabaya Indonesia
Tel. +62 89680847005
yosi@stts.edu

ABSTRAK

Humans can recognize and classify shapes, names, and provide responses to object that are received by visually quickly and accurately. More importantly, it is expected that the system created is able to help provide response in all tasks and time, for example when driving, walking in the crowd even when patrolling as a member of the military on dangerous terrain.This has become a problem in the system used on the battlefield. In the proposed system, the object detection model must be able to sort out the objects of armed humans (militia) with unarmed human objects. To overcome the problem the author uses the YOLO transfer learning algorithm which currently has the third version. It is stated that YOLOv3 has very extreme speed and accuracy. In mean Average Precision (mAP) obtained by 0.5 IOU, YOLOv3 is equivalent to 4x faster than Focal Loss. Moreover, YOLOv3 also offers optimal speed and accuracy simply by changing the size of the model, without the need for retraining

INTRODUCTION

Various methods are used for object detection have been developed. The object detection system is made as closely as possible with the human observation model that is received visually. Histogram of Oriented Gradients (HOG) with SVM [1], although this method is strong at detecting humans in limited quality images, but this method fails when the main object is partially closed by another object like vehicles, trees and others. This has become a problem in the system used on the battlefield. In the proposed system, the object detection model must be able to sort out the objects of armed humans (militia) with unarmed human objects. To overcome the problem the author uses the YOLO transfer learning algorithm which currently has the third version. It is stated that YOLOv3 has very extreme speed and accuracy. General purpose robots need the ability to interact with and manipulate objects in the physical world. Humans see novel objects and know immediately, almost instinctively, how they would grab them to pick them up. Robotic grasp detection lags far behind human performance [2]. In mean Average Precision (mAP) obtained by 0.5 IOU, YOLOv3 is equivalent to 4x faster than Focal Loss [3]. Moreover, YOLOv3 also offers

optimal speed and accuracy simply by changing the size of the model [4], without the need for retraining. The objectives and benefits of the research that will be carried out on the topics taken are conduct training at YOLO to be able to recognize objects of armed humans (militias) in the image and helps classify militia objects and ordinary people. The usage contribution of research are:

a. Aims to detect human objects that are not merely labeled "Person" but also as "militia" according to the picture given.
b. This research is very important if developed in the future, for example, it is used to make special auto-turret technology for militia detection, automatic weapons that can shoot targets that have been arranged in the object category

RESEARCH METHOD

The following picture is a system diagram in this study, begin from input image until Militia Classification

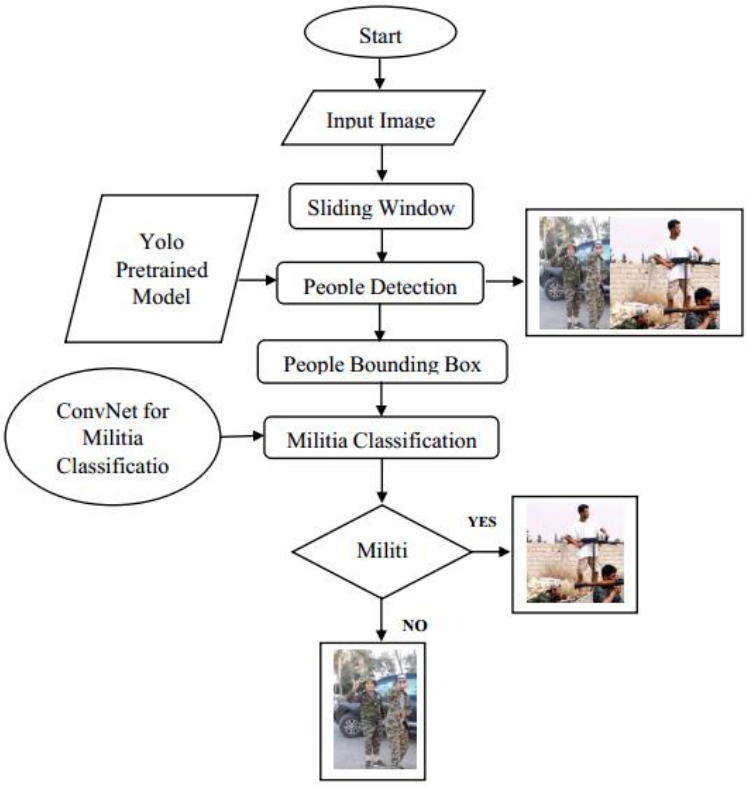

Figure 1. Architecture system diagram

In order to build up to object detection, we first learn about object localization.

a. Input Image

Start by defining with the image classification task where an algorithm looks at this picture an might be responsible for saying this is a man with gun (militia). So that was classification [1]. The problem we learn to build a neural network is classification with localization [5] which means not only do we have to label this as say a militia or not militia but the algorithm also is responsible for putting a bounding box [6] or drawing a red rectangle around the position of the militia in the image. So that is called the classification with localization problem [7] where the term localization refers to figuring out where in the picture is the militia we have detected.

Figure 2. People without Gun and Holding Gun (militia)

b. Sliding Window Detection (Convolutional)

If we have a test image like this, we start by picking a certain windows size [6], shown down there, for example using a man with gun (militia) image:

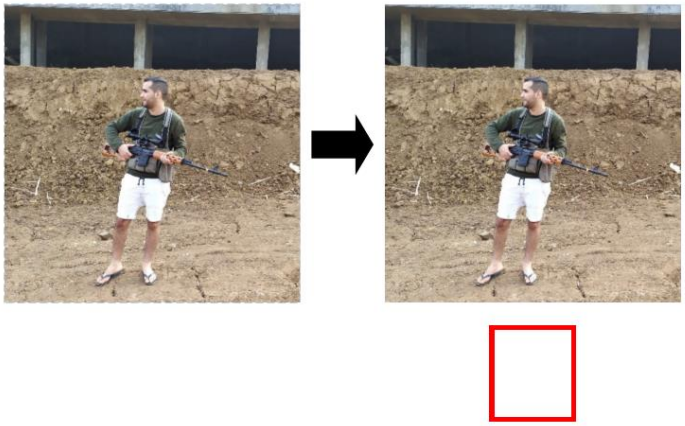

Figure 3. Sliding Window Detection (1)

Then we would input into this ConvNet [8] a small rectangular region [9]. So, take just this below red square, input that into the convNet and have a ConvNet make a prediction.

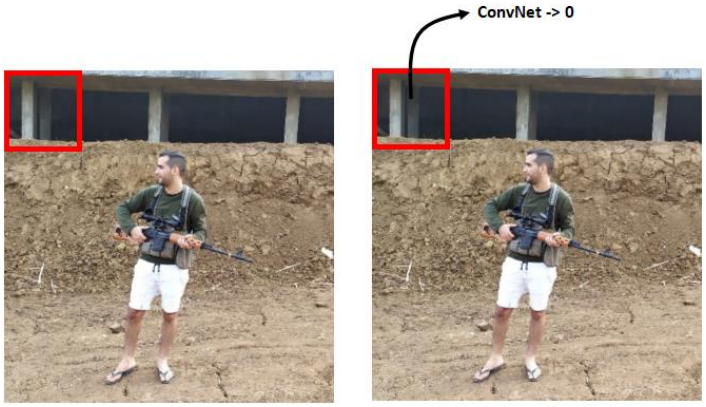

Figure 4. Sliding Window Detection (2)

Presumably for that little region in the red square, it will say NO, that little red square does not contain a militia. In the Sliding Windows Detection Algorithm [6], what we do is then pass as input a second image now bounded by this red square shifted a little bit over and feed that to the convNet. So we are feeding just the region of the image in the red square of the convNet and run the convNet again. And for third image and so on.

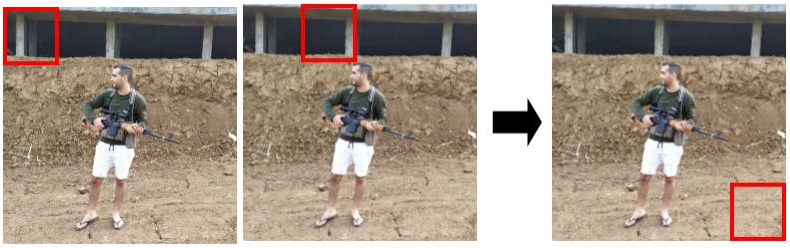

Figure 5. Sliding Window Detection (3)

Keep going until we have slid the window across every position in the image. Every region of this size and pass lots of little cropped images into the convNet and have it classified 0 or 1 for each position as some stride. Next step, repeat it use a larger window. So this algorithm is called Sliding Windows Detection because we take these windows [10], square boxes, and slide them across the entire image and classify every square region with some stride as containing a militia or not. In particular, the Sliding Windows Object Detector can be implemented convolutionally or much more efficiently [11].

c. People Detection

In order to build up to object (people) detection, we first do the object localization.

Image of People Classification

There might be multiple objects [12] in the picture and we have to detect them all and localize [13] them all. It might need to detect not just other cause but maybe other objects. The classification of localization problems usually have one object usually one big object in the middle of the image that we're trying to recognize or recognize and localize [14]. In contrast in the detection problem there can be multiple objects and in fact maybe even multiple objects of different categories within a single image. So the idea is we've learned about for image classification will be useful for classification with localization and then the idea is we learn for localization will then turn out to be useful for detection[15].

Militia or non militia classification with localization

For example there is an input image for detection below. There might be multiple objects in the picture and we have to detect them all and localize them all. It might need to detect not just other.

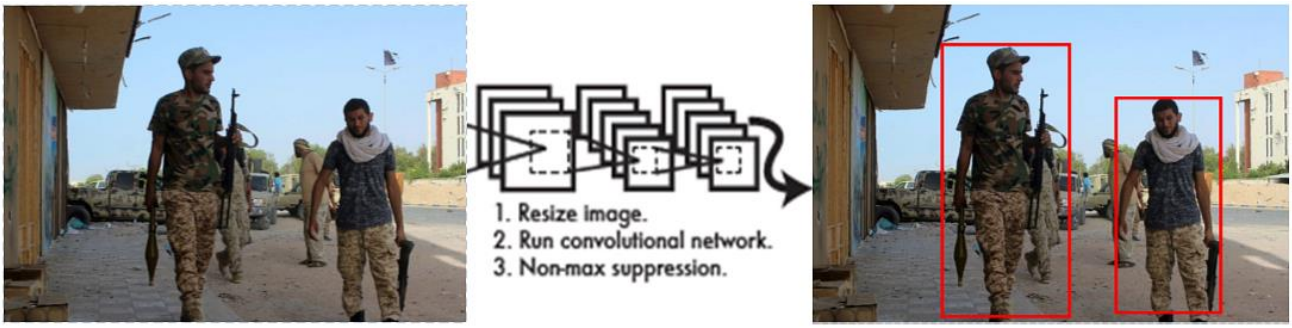

Figure 6. People detection

d. People Bounding Box

In part 2 (Sliding Windows), we use a convolutional implementation of sliding windows and that is more computationally efficient but the store has a problem of not quite outputting the most accurate bounding boxes. In this part, how we can get our bounding box predictions to be more accurate. In the example before, none of the boxes really match up perfectly with position of the people. So maybe that box is the best match. And also, it looks like in drawn through, the perfect bounding box is not even quite square. It is actually has a slightly wider rectangle or slightly horizontal aspect ratio.

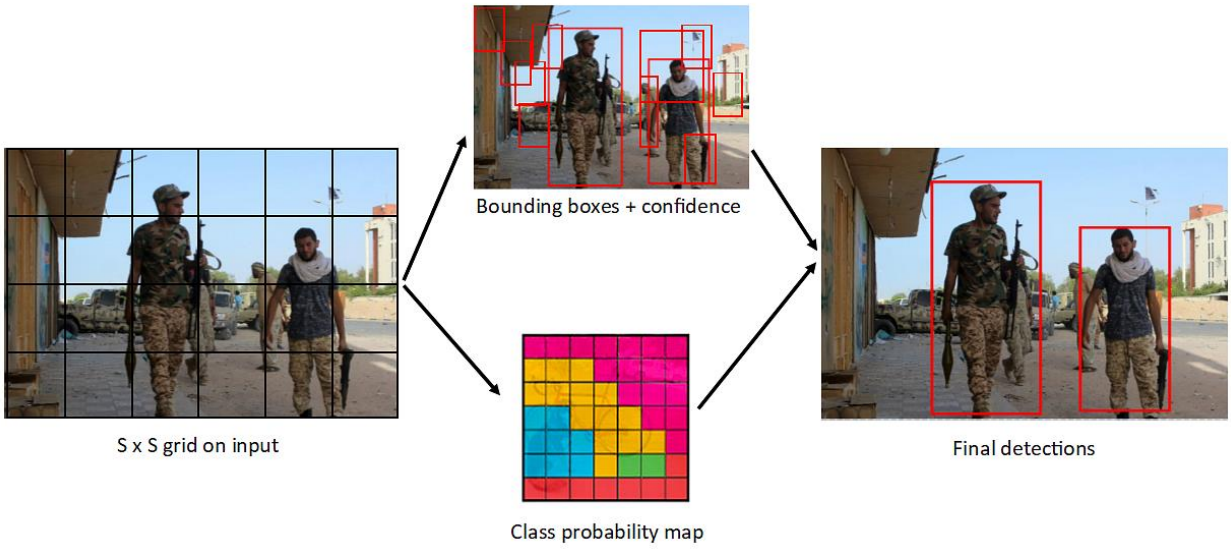

Figure 7. People Bounding Box with YOLO Algorithm

So we must get this algorithm to outputs more accurate bounding boxes. A good way to get this output more accurate bounding boxes is with the YOLO algorithm. When it comes to deep learning-based object detection, there are three primary object detectors we will encounter:

- R-CNN and their variants, including the original R-CNN, Fast R- CNN, and Faster R-CNN
- Single Shot Detector (SSDs)
- YOLO

R-CNNs are one of the first deep learning-based object detectors and are an example of a **two-stage detector.**

1. In the first R-CNN publication [13], Girshick et al. proposed an object detector that required an algorithm such as Selective Search [16] (or equivalent) to propose candidate bounding boxes that could contain objects.
2. These regions were then passed into a CNN for classification, ultimately leading to one of the first deep learning-based object detectors.

The problem with the standard R-CNN method was that it was *painfully slow* and not a complete end-to-end object detector. The Fast R-CNN algorithm made considerable improvements to the original R-CNN, namely increasing accuracy and reducing the time it took to perform a forward pass; however, the model still relied on an external region proposal algorithm [17]. R-CNNs became a true end-to-end deep learning object detector by removing the Selective Search requirement and instead relying on a Region Proposal Network (RPN) that is (1) fully convolutional and (2) can predict the object bounding boxes and "objectness" scores (i.e., a score quantifying how likely it is a region of an image may contain an image). The outputs of the RPNs are then passed into the R-CNN component for final classification and labeling [18].

YOLO is a great example of a single stage detector [19], details an object detector capable of super real-time object detection, obtaining **45 FPS** on a GPU. YOLO has gone through a number of different iterations, including YOLOv2, capable of detecting over 9,000 object detectors [20]. Redmon and Farhadi are able to achieve such a large number of object detections by performing joint training for both object detection and classification. Using joint training the authors trained YOLO9000 simultaneously on both the ImageNet classification dataset and COCO detection dataset. The result is a YOLO model, called YOLO9000, that can predict detections for object classes that don't have labeled detection data. YOLOv3 is significantly larger than previous models, the best one yet out of the YOLO family of object detectors [3].

e. Militia classification

For this box over here hopefully, the value of *y* to the output for that box at the bottom left, hopefully would be something like zero for bounding box one. And then just open a bunch of numbers, just noise. Hopefully, we'll also output a set of numbers that corresponds to specifying a pretty accurate bounding box for the person/militia. So that's how the neural network will make predictions. Finally, we run this through non-max suppression.Non-max Suppression (for multiple detection problem) is one of the problems of object detection is that the algorithm may find multiple detections of the same objects. Rather than detecting an object just once, it might detect it multiple times. Non-max suppression is a way for us to make sure that our algorithm detects each object only once.

Figure 8. Militia Classification

RESULT AND DISCUSSION

Several stages of implementation need to be done on 2 class objects, both militia and non-militia. Namely training dataset for each class, the process of labeling objects in the image dataset, training, and testing.

Dataset for training

The following are datasets for militia and non militia (person). Each dataset class consists of 100 image data, 100 images for militia, and 100 images for non militia.

(a) (b)

Figure 10. Image dataset (a) militia, (b) non militia

Labeling

Given this label training set, we can then train a ConvNet that that inputs an image, like one of these closely cropped images. And then the job of the ConvNet is to output y, 0 or 1, is there a militia or not.

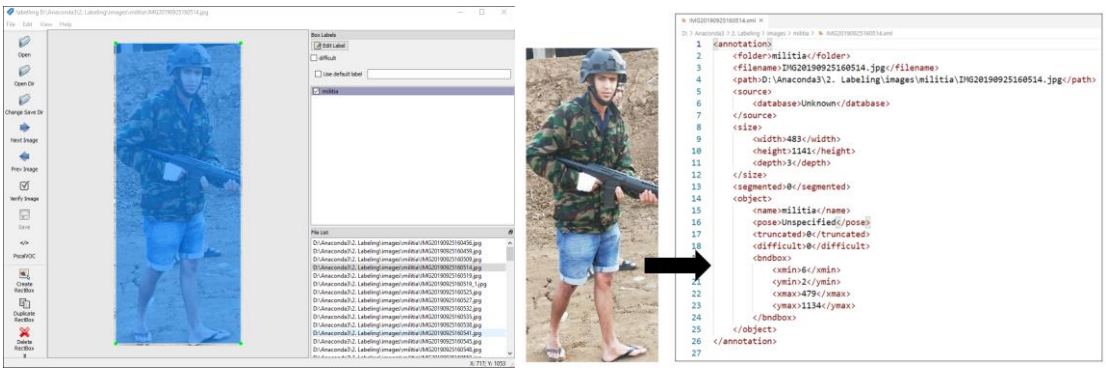

Figure 9. Labeling process and labeling result (xml format)

Training

Training command in darkflow with GPU:

```
python flow --model cfg/yolov2-1c.cfg --load bin/yolov2.weights --train --annotation train/annotations --dataset train/images --gpu 0.8 --epoch 300
```

300 epoches need 4.5 hours (GPU GTX1070 8GB, 16GB DDR4, Ryzen 5 2600 processor). For best test result, may need minimum 10000 epoch.

Testing

Testing command using darkflow with GPU:

```
python flow --imgdir sample_img/ --model cfg/yolov2-1c.cfg --load bin/yolov2.weights --labels labels.txt --gpu 0.8
```

Figure 10. Detection result as millitia (people with gun)

Figure 11. Detection result as person (people without gun)

CONCLUSION

For accuracy, For our model, we have got 0.803 which means our model is approx. 80% accurate. For precision, We have got 0.788 precision which is pretty good. For recall, We

have got recall of 0.631 which is good for this model as it's above 0.5 (militia or person). And for F1 score in our case is 0.701.

ACKNOWLEDGEMENT

Thank you to the Institut Sains Terapan dan Teknologi Surabaya which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "Detection = Classification + Localization," 2017.

[2] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2015-June, no. June, pp. 1316–1322, 2015, doi: 10.1109/ICRA.2015.7139361.

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

[4] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, 2014.

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net : ImageNet Classification Using Binary," *Eccv2016*, pp. 1–17, 2016.

[8] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 808–816, 2016, doi: 10.1109/CVPR.2016.94.

[9] J. Li, Y. Wu, J. Zhao, L. Guan, C. Ye, and T. Yang, "Pedestrian detection with dilated convolution, region proposal network and boosted decision trees," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017-May, pp. 4052–4057, 2017, doi: 10.1109/IJCNN.2017.7966367.

[10] D. Forsyth, "Non-Maximum-Suppression," *Computer (Long. Beach. Calif).*, vol. 47, no. 2, pp. 6–7, 2009, doi: 10.1109/MC.2014.42.

[11] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," *MM 2014 - Proc. 2014 ACM Conf. Multimed.*, pp. 675–678, 2014, doi: 10.1145/2647868.2654889.

[12] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 886–893, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[14] T. F. Gonzalez, "Handbook of approximation algorithms and metaheuristics," *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.

[15] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to segment object candidates," *Adv. Neural Inf. Process. Syst.*, vol. 2015-January, pp. 1990–1998, 2015.

[16] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *Proc. - 13th IEEE Int. Conf. Autom. Face Gesture Recognition, FG 2018*, pp. 357–364, 2018, doi: 10.1109/FG.2018.00058.

[17] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

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

[19] 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.

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.