Haibin Yu · Jinguo Liu · Lianqing Liu ·
Zhaojie Ju · Yuwang Liu · Dalin Zhou (Eds.)

LNAI 11743

Intelligent Robotics and Applications

12th International Conference, ICIRA 2019
Shenyang, China, August 8–11, 2019
Proceedings, Part IV

Springer

Real Time Object Detection Based on Deep Neural Network

Tarek Teama[1], Hongbin Ma[1(✉)], Ali Maher[2], and Mohamed A. Kassab[3]

[1] Beijing Institute of Technology, No. 5 South Zhong Guan Cun Street, Haidian, Beijing 100081, People's Republic of China
eng.tarek.teama@gmail.com, mathmhb@qq.com
[2] Military Technical College, Cairo, Egypt
ali_mtc@hotmail.com
[3] Beihang University, Beijing, China
mohamedkassab99@gmail.com

Abstract. In this research we focus on using deep learning for the training of real time detection of defected Nails and Nuts on a high speed production line using You Only Look Once (YOLO) algorithm for real time object detection and trying to increase the precision of detection and decrease the problems facing real time object detection models like Object occlusion, different orientation for objects, lighting conditions, undetermined moving objects and noise. A series of experiments have been done to achieve high prediction accuracy, the experimental results made on our costumed pascal visual object classes (VOC) dataset demonstrated that the mean Average Precision (mAP) could reach 85%. The proposed model showed very good prediction accuracy on the test dataset.

Keywords: Computer vision · Deep learning · Visual servoing · YOLOv2 · Convolutional Neural Network · Object detection

1 Introduction

Now a days visual perception is very important for robots to do some tasks like grasping [1], tracking and detection of objects, it provides rich and detailed information about the environment the agent is moving in. The main purpose of visual servoing [2] techniques is to control a dynamic system by means visual extracted features. Over the last years deep learning made great contributions in the field of computer vision like:

1. Object classification, detection and tracking
2. Direct human robot interaction
3. Object location prediction for grasping in visual servoing systems.

Most of the recent researches in visual servoing, are trying to propose different solutions based on deep learning throughout using different configurations of

H. Yu et al. (Eds.): ICIRA 2019, LNAI 11743, pp. 493–504, 2019.
https://doi.org/10.1007/978-3-030-27538-9_42

Convolutional Neural Networks (CNN). We briefly explain YOLO and YOLOV2. YOLO algorithm [3] is one of the frameworks that are used recently in most of applications for real time object detection and tracking. Object detection task is accomplished by detecting (localize) each target in the input image then recognize it, the localization task is carried out by the regression network while the recognition task is carried by the classification network, as a result of combining both networks in a unified system, the detection deep proposals [4,6,7], can not be utilized in real-time application system.

In the algorithms based on regression [3,5] instead of selecting interesting parts of an image (RCNN), they predict classes and bounding boxes for the whole image in one run of the algorithm. As YOLO's name exactly means that the algorithm is looking through the whole image only once and putting the bounding boxes on every object in the image where the algorithm thinks that there is a probability that this object exists in that area of the image and then remove the bounding boxes with lower existence probability percentage, the threshold may differs from application to another.

As shown in Fig. 1 firstly the image is divided into cells typically $S \times S$ grid, each cell is responsible for a constant number of bounding boxes (B) and class probability (c). Secondly YOLO puts all the bounding boxes inside the image, finally the bounding boxes with low object probability are been removed and only boxes with high object probability which contains the highest shared area still exist, this is done through a process called non-maximum suppression (NMS) and the amount of filters needed in the last layer is calculated through.

$$F = B \times (C + 5) \tag{1}$$

In Eq. (1), F represents the last layer total number of filters needed in the last layer, B is the number of bounding boxes, while C is the total number of classes, the number 5 represents the bounding box predictions related outputs.

Fig. 1. YOLO object detection sequence

Redmon et al. [8] proposed a new version YOLOv2 to improve YOLO accuracy, darknet-19 network was designed by removing the full connection layers of the network, and batch normalization [9] was applied to each layer. Referring to the anchor mechanism of Faster R-CNN, k-means clustering was used to obtain the anchor boxes. In addition, the predicted bounding boxes were retrained with

direct prediction. Since in this paper we are using a costumed dataset and that doesn't contain any general objects so we calculated the anchors for our dataset using K-means [10] clustering method.

2 Related Works

In this section, we will highlight some of the most recent state of arts on real time object detection, tracking and visual servoing.

2.1 Human Robot Interaction

The problem of direct human robot interaction [17] (HRI) still facing some challenges on how to make HRI much more safe for human workers. A proposed solution was introduced in [11] which focuses on the problem of the robots working with human directly and help transferring objects between human and robot with safe environment for the human worker and the other problem is detecting and tracking both the object position and the human position to grasp the object efficiently from the human worker.

2.2 Object Detection and Visual Servoing

A system was introduced [12] to locate and grasp a leaf from a plant using a single camera, the task of picking a leaf can be divided into three parts, firstly determining the position of the leaf and identifying it in the image space using CNN, secondly transforming its position into cartesian space, finally using of visual servoing to control the robot for picking the leaf using Monoscopic Depth Analysis (MDA) as a control scheme.

2.3 Defect Inspection

Defects inspection is one of the most important tasks in production lines, an efficient Automated Optical Inspection equipment (AOI) [13] was introduced which uses artificial neural network based on deep learning to detect features and classify defects on touch panel glass with area of $43 \times 229.4\,\mathrm{mm}^2$, the training and classification of the stored defect images are processed through the ZF-Net neural network. This system introduced excellent accuracy in detecting and classifying defects.

3 Proposed System

3.1 Problem Definition

In this research our purpose is to leverage the prediction accuracy of the 9 layers YOLOv2-tiny algorithm to be comparable with the prediction accuracy of the 24 layers YOLOv2 while maintaining the high speed detection advantage of

YOLOv2-tiny and keep tracking of the defected object on a production line until it is grasped and removed from the production line and obtaining very accurate predictions to the defected object with the exact location of that object.

During this work we faced three main challenges, the first one was the detection of very small objects on high speed production line, the second was the noise produced due to the mechanical parts movements over the production line, while the last one was the randomly placed objects with different orientations that also may results in having lots of occluded objects. Our dataset is composed of only two classes nuts and nails.

3.2 Dataset Creation

First a custom dataset of nuts and nails with different orientations, occlusions and lighting conditions have been created that contains 3000 jpg images, this dataset created by first taking real-time videos with video frame rate 60 fps and then annotating these videos using ViTBAT Video annotation tool [14], this dataset consists of six video sequences with 3000 fully annotated frames of resolution 1920×1080 pixels. Each target is annotated by a bounding box across non-successive frames; then the tool automatically interpolates the bounding box locations and sizes in skipped frames for each target as shown in Fig. 2.

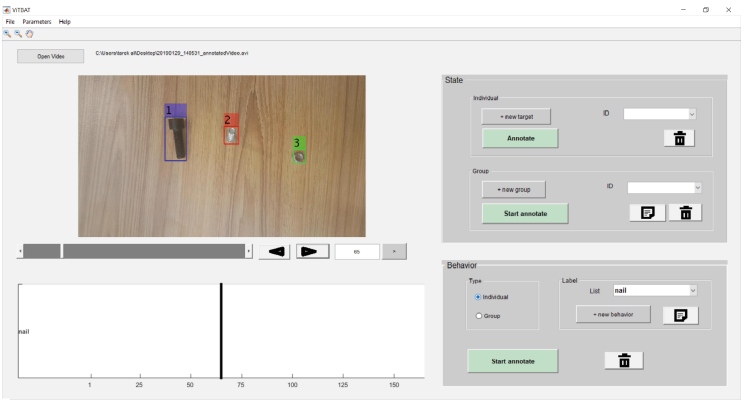

Fig. 2. The annotating process of nuts and nails by the video tracking and behavior annotation tool (ViTBAT)

Figure 3 shows a sample images from our dataset after cutting the video frames by MATLAB into sequence of images, the dataset images was divided randomly into 70% training, 15% validation and 15% testing.

3.3 Enhancing YOLOv2-Tiny Detection Accuracy

Since decreasing the number of layers will decrease the accuracy three approaches were applied to leverage YOLOv2-tiny 9 layers detection accuracy to be compa-

Fig. 3. Dataset sample

rable with the original 24 layers YOLOv2 network. The first one was to investigate the impact on the mAP and average loss throughout changing the network input image resolution from 416×416 to 608×608, which contributed to raising the mAP as shown in Table 1, however the anchor boxes that were used inside YOLOv2-tiny was not suitable for detecting very small objects.

Table 1. Input image resolution change impact on (mAP)

Model	mAP (%)	
	416×416	608×608
YOLOv2	71	72
YOLOv2-tiny	37	58

The second one was to calculate the anchor boxes and this is done through using k-mean clustering with Euclidean distance. We obtain anchors for the training dataset as follows:

As shown in Eq. (2) each bounding box (width and height) tuple is multiplied by the ratio between the network input resolution and the size of the input image contains that bounding box.

$$(w_r, h_r) = \left(\frac{W_n}{W_i} \times w_t, \frac{H_n}{H_i} \times h_t\right) \tag{2}$$

where, (w_t, h_t) and (w_r, h_r) are the ground truth bounding boxes and re-scaled bounding boxes width and height, respectively. (W_n, H_n) and (W_i, H_i) are the network input resolution and the input image width and height, respectively.

After that all of the re-scaled bounding boxes are being clustered through using k-mean clustering algorithm. The obtained k centroid tuples (w_c, h_c) were

divided by the feature map stride $feat_{st}$ as shown in Eq. (3) of the network, here the network contains 5 max pooling layers with stride 2 then $feat_{st} = 32$. We know have k (w_a, h_a) tuples, each one is the resulting anchor's width and height.

We calculated anchor's from $k = 1$ until $k = 22$, then through using silhouette average coefficient we managed to obtain the optimum number of k clusters that are suitable for our training dataset, where $k = 9$ so we choose 9 anchor boxes and the last convolutional layer output feature map (filter size) was set to 63 (Fig. 4).

$$(w_a, h_a) = \left(\frac{w_c}{feat_{st}}, \frac{h_c}{feat_{st}}\right) \tag{3}$$

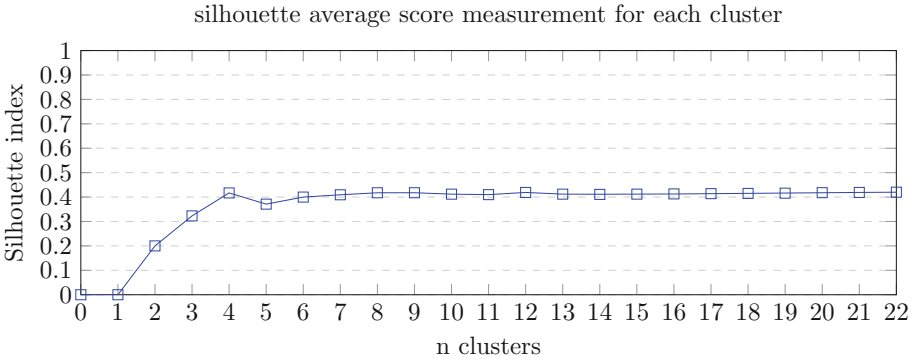

Fig. 4. Silhouette average score measurement

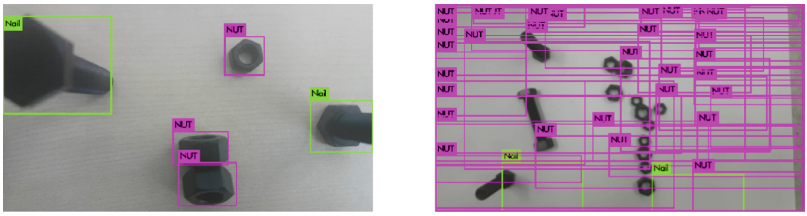

Fig. 5. Impact of our optimum number of anchors on YOLOv2-tiny detection accuracy

The impact of the silhouette measure [15] to estimate the optimum number of anchor's on the detection results is illustrated qualitatively in Fig. 5, where the right figure demonstrates the detection results with default visual object class (VOC) [16] anchors, while the left one with our evaluated anchors. The last step we needed to solve the issue of the decreased prediction accuracy due to the challenges we mentioned in Sect. 3.1.

We have done some dataset augmentations by choosing 15 images from our training dataset and using OpenCV on each image to add some transformations to those images, we included 34 image transformations which are cropping, resizing, padding each image, flipping the image, inverting image, superpixel image, adding light to the image from dark to full brightness, adding light color, saturation, hue image, multiply image for RGB colors with different ranges, gaussian blur, averaging blur, median blur, bilateral blur, opening image, closing image, morphological gradient image, top hat, black hat, sharpen image, embossing image, edge image, adaptive gaussian noise, salt noise alone, pepper noise alone and combined salt and pepper noise, contrast, applying canny edge detector, gray scale image transformation, image scaling, image translation, image rotation and image geometric transformation. Each image of the 15 images produces 139 new images which includes all of these image transformations and this led to an increase in our overall dataset images to reach 5085 images which can be found on that link https://github.com/tarek212/NUTS-and-NAILS-Dataset (Fig. 6).

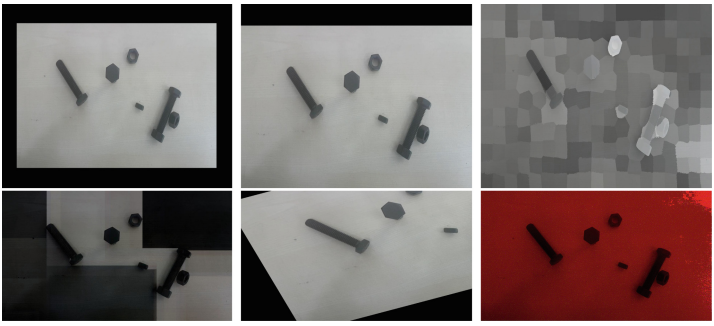

Fig. 6. Dataset augmentation samples (Color figure online)

4 Experiments

4.1 Environment

The hardware environment of the experiment is shown in Table 2, the operating system was Ubuntu 16.04, the models were implemented on the darknet platform framework.

4.2 Analysis of Training Stage

After combining all of the three approaches to train our model on YOLOv2 and YOLOv2-tiny on our training dataset. Figure 7 shows the average loss curve with the mean average precision (mAP%), where the upper one demonstrates

Table 2. The hardware environment. GPU; central processing unit (CPU).

Hardware	Environment
Computer	MSI GAMING LAPTOP
CPU	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz, 4 Cores
GPU	Nvidia GTX 1060
Video Memory	6 GB
CUDA	Version 9.0
CUDNN	Version 7.0

YOLOv2 average loss and (mAP) improvement, where the bottom figure demonstrate the effectiveness of combining the three approaches on YOLOv2-tiny with average loss 0.4966 and (mAP) almost between 82% to 83% with 40000 iterations of training, with average IOU that reaches to 82% and average recall that reaches to 90%.

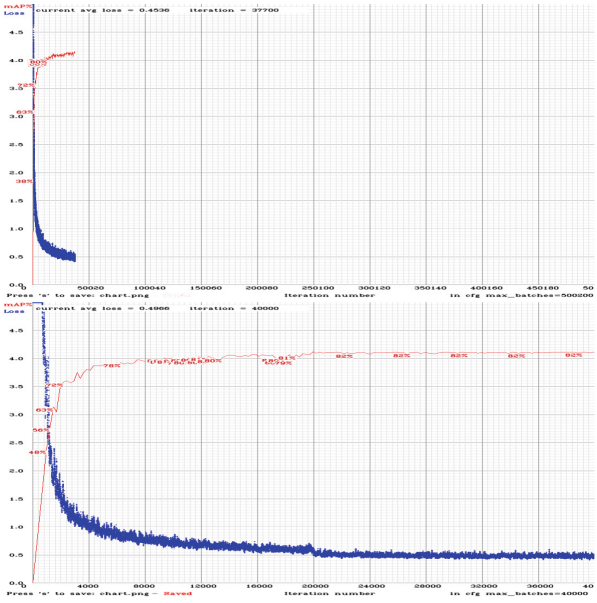

Fig. 7. YOLOv2 Vs YOLOv2-tiny training average loss and (mAP)

From the results obtained we managed to reach a comparable accuracy with YOLOv2 as shown in the comparison on Table 4, and with low computation cost and with maintaining the high speed detection advantage of YOLOv2-tiny as we needed high speed detection to work with robot arm. As shown in Table 3 the

used resources during the training process is high in YOLOv2 and the weights file size is large compared to the used resources in YOLOv2-tiny which is low and the weights file size is very small.

Table 3. Environment used resources during training process and weights file size

Network	Resources used	
	GPU DRAM (GB)	Weight file sized (MB)
YOLOv2 24 layers	4.5	202.8
YOLOv2-tiny 9 layers	1.418	44.2

Table 4. Training results

Model	Input resolution	Anchors	Average loss	mAP %
YOLOv2	608 × 608	9	0.4536	83
YOLOv2-tiny	608 × 608	9	0.4966	82

4.3 Analysis of Test Stage

The performances of YOLOv2 tiny using the test dataset with a IOU threshold = 0.5 were compared using the recall, precision and average IOU as the evaluation metrics, as shown in Table 5. For object detection, a very important metric for measuring the performance of a model is mAP. As shown in Table 6. Figure 9 shows the detection results of the YOLOv2-tiny model, it shows good results with both single and multiple object detection and during the test on real time videos the model shows high detection speed with minimum 50 fps and maximum 100 fps. with prediction accuracy that ranges from 65% to 99% depending on the orientation, lighting condition and occlusion. Finally, the proposed detector delivers the detection results to a decent linear velocity kalman filter to track the defective parts as presented in [17]. The defective parts rejection process can be accomplished by means of assemble robot arm derived with the tracked defective object under ROS transportation protocol (Fig. 8).

Table 5. Testing results on real time videos

Model	Recall (%)	Precision (%)	Avg. IOU (%)	Speed (fps)	mAP (%)
YOLOv2	80	78	75	5–20	72
YOlOv2-tiny	85	84	82	50–100	82.21

Table 6. Testing results on test dataset for images and videos

Model	Nail (%)	Nut (%)
YOLOv2	77	76
YOLOv2-tiny	79.02	85.41

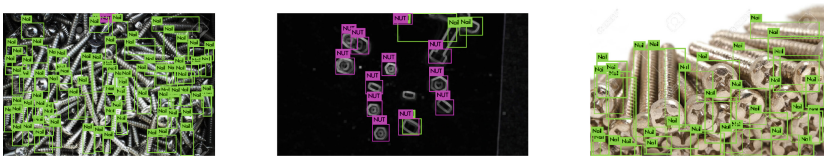

Fig. 8. YOLOv2 detection results

Fig. 9. YOLOv2-tiny testing results

5 Conclusion

A real-time object detection system has been presented through using YOLOv2-tiny, the presented solution has the ability to detect nails and nuts on high speed production line also we improved the overall prediction accuracy and maintained the detection speed advantage of YOLOv2-tiny by doing some dataset augmentations through using opencv library. These data augmentations were carried out to achieve efficient training that capable of increasing the detection prediction accuracy during testing on images and real time videos.

Through using cyclic learning rate during the training processes we managed to obtain the best learning rate for the model to converge faster which has been set to 0.00001, increasing the input image resolution from 416×416 to 608×608 and using K-means clustering to obtain the anchors and to obtain the optimum number of anchor boxes we used the silhouette coefficient measurement, which improved the overall feature extraction process.

In spite of the use of the 9 layers YOLOv2-tiny which should decrease the accuracy of the detection, we managed to maintain high speed real-time object detection and with comparable prediction accuracy with the 24 layers YOLO, and we managed also to maintain low computation cost as shown in Table 3, so this model is suitable to work in an embedded systems environment without using alot of its resources.

However due to our limited computing power we couldn't use YOLOv3 which is more accurate, and we still have one challenge which is the probability that an object appear on the production line during the detection that is not one of the two classes, since our dataset is relatively low so in future work we will collect more actual nuts and nails data to further study how to improve the accuracy and speed of defects detection on a high speed production line. Finally implementing a tracking system to be able to track the defected objects on the production line and apply our model on robot arm.

References

1. Sanchez-Lopez, J.R., Marin-Hernandez, A., Palacios, E.: Visual detection, tracking and pose estimation of a robotic arm end effector, April 2011. https://www.researchgate.net/publication/239918179
2. Tsarouchi, P., Michalos, G., Makris, S., Chryssolouris, G.: Vision system for robotic handling of randomly placed objects. Procedia CIRP **9**, 61–66 (2013). https://doi.org/10.1016/j.procir.2013.06.169
3. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 2016, pp. 779–788 (2016)
4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014). https://doi.org/10.1109/CVPR.2014.81
5. Girshick, R.: Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015). https://doi.org/10.1109/ICCV.2015.169
6. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39** (2015). https://doi.org/10.1109/TPAMI.2016.2577031
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988 (2017). https://doi.org/10.1109/ICCV.2017.322
8. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 2017, pp. 6517–6525 (2017)
9. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, Lille, France, July 2005, pp. 448–456 (2005)
10. Sang, J., et al.: An improved YOLOv2 for vehicle detection. Sensors **18** (2018). http://www.mdpi.com/1424-8220/18/12/4272. https://doi.org/10.3390/s18124272
11. Wang, Y., Ewert, D., Vossen, R., Jeschke, S.: A visual servoing system for interactive human-robot object transfer. Autom. Control Eng. J. **3** (2015). https://doi.org/10.12720/joace.3.4.277-283
12. Ahlin, K., Joffe, B., Hu, A.-P., Mcmurray, G., Sadegh, N.: Autonomous leaf picking using deep learning and visual-servoing. IFAC-PapersOnLine **49**, 177–183 (2016). https://doi.org/10.1016/j.ifacol.2016.10.033

13. Ye, R., Pan, C.-S., Chang, M., Yu, Q.: Intelligent defect classification system based on deep learning. Adv. Mech. Eng. **10**(03) (2018). https://doi.org/10.1177/1687814018766682
14. Biresaw, T.A., Nawaz, T., Ferryman, J., Dell, A.I.: ViTBAT: video tracking and behavior annotation tool. In: 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 295–301 (2016). https://doi.org/10.1109/AVSS.2016.7738055
15. Maher, A., Taha, H., Zhang, B.: Realtime multi-aircraft tracking in aerial scene with deep orientation network. J. Real-Time Image Proc. **15**(3), 495–507 (2018)
16. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
17. Maher, A., Li, C., Hu, H., Zhang, B.: Realtime human-UAV interaction using deep learning. In: Zhou, J., et al. (eds.) CCBR 2017. LNCS, vol. 10568, pp. 511–519. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69923-3_55