A time-saving automobile assembly state monitoring system based on channel-pruned YOLOv4 algorithm

Jiang Daqi

Northeastern University

Hong Wang (✉ hongwang@mail.neu.edu.cn)

Northeastern University

Research Article

Keywords: Assembly state monitoring, YOLO v4, Channel pruning algorithm, Real-time detection

Posted Date: November 3rd, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1026299/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

A time-saving automobile assembly state monitoring system based on channel-pruned YOLOv4 algorithm

Abstract

A time-saving automobile assembly state monitoring system in industrial environment is presented in this paper. The system only needs to input a video which contains the whole detected parts and manually label in the first frame. By finding the best point for tracking and tracking the point, the dataset can be automatically generated which saves time spent on manufacturing the dataset and makes the assembly state monitoring system easy to deploy into a practical industrial environment. The target detection algorithm uses the channel-pruned YOLOv4 neural network. The experimental result shows the algorithm balances speed and accuracy. Compared to original YOLOv4, our proposed method is two times faster and the mAP is nearly equal to it. It shows that the channel pruning process dynamically improves the speed of the forward propagation without sacrifice accuracy.

Keywords: Assembly state monitoring; YOLO v4; Channel pruning algorithm; Real-time detection.

Introduction

In automobile industry, assembly line requires the most manpower due to the complexity of the process¹. Despite that the workers are frequently professionally trained, there is still a tiny probability of error. As Murphy's law says: anything that can go wrong will go wrong if repeatedly. The assembly line covers thousands of steps. If an inaccurate step occurs, the factory will spend a lot of time to repair and even it will cause serious consequences.

In this paper, in response to this problem, a real-time and accurate automobile assembly state monitoring system is proposed. The system is able to detect the result of the assembly by obtaining the image and inputting it to the neural network. If a workpiece is wrongly installed or forgotten by mistake, it will remind the worker to check it, which will eliminate of ill effects of the mistake. In addition, the algorithm is based on deep learning which frees programming personnel from manually extracting image features. Normally, the deep learning algorithm needs numerous labeled images as dataset. Contrastively, the proposed algorithm needn't label the image by hand and the algorithm will automatically label the input image. Besides, operating personnel can shoot a video of the area to be tested and the datasets can be automatically generated. The generated datasets are subsequently used to train the neural network. This saves time spent on manufacturing the dataset and makes the intelligent grasping system easy to deploy into a practical industrial environment. Due to the accuracy and robustness of the convolutional neural network, the success rate of the gripping operation reached an astonishing 100% without interference. The approach is validated in a series of experiments. This method frees workers from monotonous work and improve factory productivity.

Target detection algorithms based on deep learning are divided into two categories². One is a

classification-based target detection algorithm (two-stage) and the other is a regression method for target detection (single-stage). As the name implies, the two-stage method uses two steps to perform target detection: 1. Generate a possible region (Region Proposal) and use CNN to extract features. 2. Use the classifier to classify and correct the position. The representative of this type of method is R-CNN³. In 2014, Ross et al. proposed R-CNN. The algorithm uses selective search to obtain candidate regions and then normalizes them as input to the CNN network. Then use the neural network to obtain the characteristics of the candidate area, and finally use multiple SVM classifiers for classification. R-CNN greatly improves the accuracy of target detection. However, because R-CNN performs feature extraction and calculation on all candidate regions, the overlapping regions between multiple candidate regions are repeatedly calculated many times, resulting in a large number of repeated calculations and reducing the computational efficiency. Due to the complexity of the calculation process, a large amount of intermediate data needs to be stored during the calculation process, which requires a large number of storage resources. Therefore, the real-time performance of R-CNN is not strong, and it takes up storage resources very much. In 2015, Fast R-CNN improved on the basis of SPP-Net, simplified the SPP layer, combined the classification and border regression problems, and introduced SVD decomposition to reduce the amount of calculation⁴. However, Fast R-CNN still uses the selective search method to select candidate regions which consumes a lot of computing resources and its calculation speed is still not satisfactory. In response to the problem of candidate region selection, Faster R-CNN uses RPN network instead of selective search method, marking that the target detection algorithm based on deep learning has embarked on a true end-to-end calculation⁵. However, because Faster R-CNN still uses the ROI layer in Fast R-CNN, its effect on the detection task of small targets is not satisfactory. And due to the limitation of two-stage itself, the fastest Faster R-CNN in the two-stage series of algorithms can only reach 5 FPS. Therefore, researchers have proposed another idea: single-stage. The problem of target detection and classification is directly transformed into a regression problem, and the step of selecting candidate regions and extracting features is removed, and the position and category of the detection object are directly obtained through the entire image. The representative of this series of algorithms is the YOLO algorithm⁶. The YOLO algorithm divides the entire picture into $S \times S$ grids, and each grid is responsible for detecting the detection object whose center falls within the grid, including the location and category information of the detection object. This algorithm trades for a significant increase in speed by discarding a certain degree of accuracy, and its detection speed can reach up to 45FPS. In 2017, the author of YOLO proposed the YOLOv2 algorithm, which improved the detection accuracy and speed by adding the batch normalization layer after the convolutional layer and adding the K-Means clustering method⁷. This algorithm represents the most advanced target detection algorithm in the industry this year. In 2018, YOLOv3 made some design details improvement based on YOLOv2, and once again improved the detection accuracy while maintaining a higher detection speed⁸. In short, for different application scenarios, the two-stage algorithm and the single-stage algorithm have their respective scopes of application. There is no

inferior algorithm, only suitable usage scenarios.

Result

Proposed system description

The procedure of intelligent grasping system proposed in this paper is illustrated in Fig. 1. In this system, a video including detected workpiece in various angles is the original input of the system. Operating personnel set the region of interest which only includes the detected workpiece. And then the system will work automatically. In the region of interest, at most 100 tracking points are extracted to realize the tracking of the detected workpiece in the first frame. The tracking points will update in the next frame by optical flow method. In each frame, the smallest orthogonal rectangle containing each tracking point is calculated. Each frame is saved to image and together with the coordinates and dimensions are integrate into datasets to train the neural network based on YOLOv4 model.

Fig. 1 Illustration of the proposed procedure of intelligent grasping system

Find the best point for tracking

First, take a video containing all the parts whose assembly state is inspected. Then, find the strongest corners to be tracked in the first frame. To find them, convert every frame of the video to 2-dimensional grayscale. Assume an area with coordinate (u, v) and shifted by (x, y) . The weighted sum of squared differences between these two areas which are denoted by $S(x, y)$ can be calculated by:

$$S(x, y) = \sum_u \sum_v w(u, v) (I(u+x, v+y) - I(u, v))^2 \quad (1.)$$

Perform Taylor expansion to $I(u+x, v+y)$ and let I_x and I_y be the partial derivatives of I . We have:

$$I(u+x, v+y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y \quad (2.)$$

Which produces the approximation:

$$S(x, y) \approx \sum_u \sum_v w(u, v) (I_x(u, v)x + I_y(u, v)y)^2 \approx (x, y)A \begin{pmatrix} x \\ y \end{pmatrix} \quad (3.)$$

where A is the structure tensor,

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x(u, v)^2 & I_x(u, v)I_y(u, v) \\ I_x(u, v)I_y(u, v) & I_y(u, v)^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \quad (4.)$$

A strong corner has a large value of S in two directions. Calculate the eigenvalues of A and select A with two large eigenvalues as the strong corner. Based on the magnitudes of the eigenvalues λ_1 and λ_2 , the points with $\min(\lambda_1, \lambda_2)$ are selected for tracking according to the Kanade-Tomasi corner detecting method⁹. As shown in Fig. 2, in the first frame of the video, a total of 100 points are selected to be used for the next track.

Fig. 2 The best points to be tracked

Track the point and generated the dataset

After the corner points are selected, use the optical flow method to track the trajectory of these points. In the video, assume that the displacement of the part between two nearby frames is small and almost constant in a little area. Hence, the optical flow velocity vector (V_x, V_y) always satisfies the following equation:

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ \vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \quad (5.)$$

Where $q_1, q_2, \dots, q_n$ are the pixels near the corner point, and $I_x(q_i), I_y(q_i), I_t(q_i)$ are the partial derivatives with respect to position x, y and time t . Convert these equations into matrix form:

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \quad (6.)$$

The system of equations is overdetermined. Based on the least-squares principle, the optimal solution is as follows¹⁰:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix} \quad (7.)$$

Iterating on every frame, the assembly part is tracked in the video. The tracking effect of the workpiece is shown in Fig. 3. By extracting the best tracking point and tracking by optical flow method, the position of the detected workpiece in each frame is accurately calculated, which is crucial to produce the dataset.

Fig. 3 The schematic of the tracing process by optical flow method

Neural networks configuration

Use the training image datasets generated by the above method to train the convolutional neural network. Experimental evidence demonstrates that the quantity is adequate for the convolutional neural network to provide precise object detection. A novel neural network whose inputs are 608×608 images resized from the original images and outputs are coordinate position of the workpiece is proposed. The neural network configuration is based on the YOLOv4¹¹ which is a CNN-based object detector. The architecture of the neural network is shown in Fig. 4, composed of an input layer, backbones, neck and heads. The backbone is the core of the network for extracting features whose role is to extract the information in the image for use by the subsequent network. These networks often use Resnet, VGG, etc. because these networks have proven that they have strong feature extraction capabilities on classification problems¹². In training process, the officially trained model parameters are directly loaded, followed by our own network. Let these two parts of the network be trained at the same time because the loaded backbone model already has the ability to extract features. During the training process, it will be fine-tuned to make it more suitable for the tasks. The neck is placed between the backbones and the head to make better use of the features extracted by the backbones. The head is the network that obtains the output content of the network, using the previously extracted features, and the head uses these features to make predictions. The tensor formula is as follows:

$$tensor = N * N * [(bounding\ box) * (offset + object + class)] \quad (8.)$$

Fig. 4 The architecture of the YOLOv4 neural network

Where $\text{bounding box}=2$, $\text{offset}=4$ and $\text{object}=1$. As the Fig. 5 shown, N is equal to 19, 38 and 76 respectively so the result can be predicted for the three scales in the network which improves the performance of detecting tiny objects. There are two types of datasets: successfully assembled and not assembled, so $\text{class}=2$.

Fig. 5 The model performs object detection on three scales which improves the performance of detecting tiny object

The network predicts 4 coordinates for each bounding box, t_x, t_y, t_w, t_h . If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h , then the predictions correspond to:

$$\begin{cases} b_x = \lambda \sigma(t_x) + c_x \\ b_y = \lambda \sigma(t_y) + c_y \\ b_w = p_w e^{t_w} \\ b_h = p_h e^{t_h} \end{cases} \quad (9.)$$

Where c_x and c_y are always whole numbers. b_x and b_y represent the predicted object coordinates. Therefore, extremely high or low t_x values are required if the b_x value approaches the c_x or c_x+1 value. Hence, this the sigmoid is multiplied by a factor $\lambda=2$, so simplifying detection of the objects on the edge of grid.

There are two factors that influence objectness score. First, whether the area has a target to be predicted. If there is a target to be measured, set $P_r(\text{object})$ to 1. If the opposite, set it to 0. Then

calculating the intersection over union (IOU) of the predicted and ground truth region. The product of the two factors is the objectness score which is calculated by:

$$\sigma(t_0) = Pr(object) * IOU_{pred}^{truth} \quad (10.)$$

Because the prediction box contains two categories and each prediction box can contain no more than one category, the softmax activation function is selected to output the probability. In order to balance the disparity between the number of positive and negative samples, a weighting factor $\alpha \in [0, 1]$ is imported¹³. In practice, we define the loss function as:

$$L = - \sum_{i=1}^N \alpha y^{(i)} \log \hat{y}^{(i)} + (1 - \alpha)(1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \quad (11.)$$

Where L represents the value of cross-entropy loss, $y^{(i)}$ is the label of the sample whose value is 0 or 1, $\hat{y}^{(i)}$ is the probability that the sample is predicted to be positive and α denotes the weight coefficients for different classes. This method is adopted in the experiments as it dramatically improves accuracy over the conventional cross-entropy function and found that $\alpha=0.8$ works first-rate in the experiments.

Discussion

In order to accurately detect the assembly state and deploy the system on industrial computers with inferior compute capability, a real-time target detection algorithm based on YOLOv4 and a channel pruning algorithm were proposed in this paper. Besides, the dataset is automatically generated by corner detection and optical flow method which saves the time of manually creating datasets. The main conclusions are as follows.

(1) Six algorithms are tested in this paper, including Faster R-CNN, SSD 300, Tiny YOLOv2, YOLOv3, original YOLOv4 and the proposed method. The mAPs of these six object detection algorithms were 84.89%, 83.19%, 82.49%, 91.87%, 97.48%, and 97.37% and the detection speeds were 6.5f/s, 32.8f/s, 64.8f/s, 30.4f/s, 31.0f/s and 61.3f/s, respectively. The result shows the algorithm balances speed and accuracy and is best suited for this task.

(2) On the basis of using the original YOLOv4 algorithm, the channels in which many scaling factors are near zero are pruned to decrease the number of parameters, model size and inference time. The channel pruning process dynamically improves the speed of the forward propagation on the premise of ensuring the detection accuracy of the model. Moreover, this research could provide some technical references for the deployment of automobile assembly state monitoring system.

Method

Training process

This section uses YOLOv4 algorithm for training and testing. To train the training set of 2400 images, the PC is equipped with 32.0 GB RAM, Core i7-9700F CPU 3.0 GHz, and the NVIDIA GeForce RTX 2060. A batch size of 32 and 4000 epochs are specified. The experiment includes the following three steps.

- Step 1: A dataset including 500 images is automatically generated. The images were used to

train the neural network.

- Step 2: The trained neural network was used to detect the images containing the assembly part.

- Step 3: In order to achieve the desired results, we finetuned the parameters and retrained.

Steps 1, 2, and 3 are repeated continuously. When all the states of automobile assembly are detected, the training process ends. The experimental process is shown in Fig. 6.

Fig. 6 The detailed experimental process

To execute the detection, the input size of the neural network was specified to 608 pixels $\times$ 608 pixels, which was mainly due to the relatively small size of assembly parts. Then perform subsequent processing. There will be 19*19*14, 38*38*14, and 76*76*14 feature maps in the network. The parameters of the training process are shown in Table 1.

After the training is completed, the loss curve during the training process is shown in Fig. 7. It shows that at the initial stage of the assembly parts detection model training, the model learning efficiency was higher, and the training curve convergence speed was faster. As the training deepened, the slope of the training curve gradually decreased. Finally, when the number of training iterations reaches about 2000, the model learning efficiency gradually reached saturation and the loss function fluctuated slightly around 0.05.

Parameters	Value
Input size	608 $\times$ 608
Learning rate	1.0×10^{-3}
Batch size	32
Classes	2
Iterations	4000

Table 1. The training process parameters of the assembly state detection model.

Fig. 7 Training loss curve of YOLOv4 assembly state detection model

We set a total of 20 sets of scene images containing the assembly process which have not appeared in the training dataset. Each scene contains one or two assembly parts which are randomly set as successfully assemble or not. The proposed algorithm successfully identifies these assembly states. The detection effect of the trained YOLOv4 algorithm on assembly parts was shown in Fig. 8. It could be seen that the algorithm could completely detect the assembly parts, indicating that the algorithm used in this research could be deployed in auto industry and perform excellently.

Fig. 8 The detection effect of the trained YOLOv4 algorithm on assembly parts

Channel Pruning and Fine-tuning

To accelerate the process of forward propagation and reduce requirements for compute capability, the original YOLOv4 is slimmed by pruning the channels in which many scaling factors are near zero¹⁴. The workflow is shown in Fig. 9. In the training process, the loss function is added a sparsity-induced penalty on the scaling factors to obtain a narrow network (see Fig. 10). The new loss function is calculated by:

$$L' = L + \lambda \sum_{\gamma \in \Gamma} \text{smooth}_{L_1}(\gamma) \quad (12.)$$

Where γ is a scaling factor for each channel multiplied to the output of that channel and λ is a parameter that balances two terms. In YOLO, γ is the scale affine transformation parameters which are responsible for linearly transforming normalized activations in batch normalization layer. Further experiments show that the training effect is the best when $\lambda=10^{-5}$. The smooth function is defined as:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (13.)$$

Compared with the L1 and L2 norms, the smooth function converges faster and is more robust to outliers.

Fig. 9 Flow-chart of network pruning procedure.

Fig. 10 Associate a scaling factor with each channel in convolutional layers and sparsity regularization is imposed on these scaling factors. Then the channels with small scaling factor values will be pruned into a narrow network

After completion of the training process, channel-level sparsity-induced regularization makes the scaling factors of many layers in the network tend to zero. Then, 70% channels with lower scaling factors are pruned. The operation obtains a narrower network with fewer parameters which improves the speed of the forward propagation.

Comparison of different object detection algorithms

In recent years, algorithms based on deep learning have achieved good results in the industry. More and more excellent algorithms are proposed. In order to compare the performance of these algorithms in recognizing assembly state, five algorithms are tested in this paper, including Faster R-CNN, SSD 300, Tiny-YOLOv2, YOLOv3 and original YOLOv4. The backbones of these four object detection algorithms are ResNet50, VGG16, compressed Darknet 19, Darknet53.

Use the same training dataset to train five types of neuron networks, and then measure the performance of them by the same test dataset. Evaluation index includes precision, recall, mAP, F1 score (also called Dice similarity coefficient) and detection speed. The calculation formula is as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (14.)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (15.)$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \quad (16.)$$

The accuracy of the detection reflects the model's ability to recognize the assembly state. Meanwhile, the speed of detection represents the potential of the algorithm to be deployed in portable devices or industrial computers with inferior compute capability. In this paper, a total of six different object neural networks including the Faster R-CNN, SSD 300, Tiny YOLOv2, YOLOv3, original YOLOv4 and the proposed method were tested to analyze the performance of assembly state detection. The test results were shown in Table 2.

Algorithms	<i>Precision</i> (%)	<i>Recall</i> (%)	<i>mAP</i> (%)	Detection speed (f/s)
Faster R-CNN	80.18	87.72	84.89	6.5
SSD 300	74.26	85.95	83.19	32.8
Tiny-YOLO v2	66.59	84.41	82.49	64.8
YOLO v3	82.99	93.14	91.87	30.4
YOLO v4	89.97	98.02	97.48	31.0
Ours	90.02	97.96	97.37	61.3

Table 2. Detection results of assembly state detection by different object detection algorithms.

It could be seen from the table that the mAPs of these six object detection algorithms were 84.89%, 83.19%, 82.49%, 91.87%, 97.48%, and 97.37%, respectively. With respect to the detection speed, the detection speeds were 6.5f/s, 32.8f/s, 64.8f/s, 30.4f/s, 31.0f/s and 61.3f/s, respectively. The Tiny-YOLOv2 was the fastest among six algorithms. However, the mAP of the Tiny-YOLOv2 was much lower than that of the proposed algorithm. Compared to original YOLOv4, our proposed method is two times faster and the mAP is nearly equal to it. It shows that the channel pruning process dynamically improves the speed of the forward propagation without sacrifice accuracy.

In addition, we restrict the algorithm to run on the CPU to simulate being deployed in industrial computers. The original YOLOv4 consumes 1 second for each forward propagation. Contrastively, the proposed channel-pruned network only consumes 0.6 seconds to complete the same process. The result shows that even if deployed in portable devices or industrial computers with inferior compute capability, the algorithm can also achieve good detection results.

Reference

1. Nee, A. Y. C., Ong, S. K., Chrysosolouris, G. & Mourtzis, D. Augmented reality applications in design and manufacturing. *CIRP Annals* **61(2)**, 657-679 (2012).
2. Wang, S., Efficient deep learning. *Nat. Comput. Sci.* **1**, 181–182 (2021).
3. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR* 580-587 (2014).
4. Girshick, R. Fast R-CNN. *ICCV* 1440-1448 (2015).
5. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards real-time object detection with

- region proposal networks. *IEEE T. Pattern Anal.* **39(6)**, 1137-1149 (2015).
6. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You only look once: Unified, real-time object detection. *CVPR*. 779-788 (2016).
 7. Redmon, J. & Farhadi, A. YOLO9000: better, faster, stronger. *CVPR*. 7263-7271 (2017).
 8. Redmon, J. & Farhadi, A. YOLOv3: An incremental improvement. arXiv preprint <https://arxiv.org/abs/1804.02767> (2018).
 9. Shi, J. Good features to track. *CVPR*. 593-600 (1994).
 10. Lucas, B. D. & Kanade, T. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop* 121-130 (1981).
 11. Bochkovskiy, A., Wang, C. Y. & Liao, H. Y. M. (2020) YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint <https://arxiv.org/abs/2004.10934v1>.
 12. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *CVPR*. 770-778 (2016).
 13. Lin, T. Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. *ICCV*. 2980-2988 (2017).
 14. Liu, Z. *et al.* Learning efficient convolutional networks through network slimming. *ICCV*. 2736-2744 (2017).

Author contributions

Jiang and Wang conceived and designed the methods. Jiang conducted the experiment, collected the data and performed the analysis. Wang had the supervision of all research and analysis. All authors read and approved the final manuscript.

Funding

The authors received the support from The National Key R&D Program of China (2021YFF0306405).

Competing interests

The authors declare no conflict of interest.

Data availability

The generated datasets and during and experimental data during the current study are available from every author on reasonable request.

Ethics declarations

This chapter does not contain any studies with human participants or animals performed by any of the authors. Authors acknowledge the work is original and all the relevant text from the literature has been properly cited.

Consent to participate

Not applicable.

Consent for publication

All authors consented to the publication.