Label Recognition and Detection Based on YOLO Algorithm

Authors: Lü Zeyu

Date: 2024-01-07T00:00:00+00:00

Abstract

This paper employs the YOLO vision algorithm for label recognition and detection, and discusses the experimental process and results.

Full Text

Preamble

With the rise of artificial intelligence and continuous development of information technology, automation and intelligence have become the inevitable direction of production revolution in the industrial sector. To address detection issues present in traditional PET bottle production lines, this paper proposes a novel and highly practical automated architecture, and constructs a test platform based on existing laboratory conditions.

For the visual implementation, the YOLOV5 algorithm is employed, with a sample set obtained through independent collection of PET bottles and production of labels. The total number of training samples is 1070, including 107 for testing and 963 for training.

2 Visual Model—YOLO Algorithm

Visual detection constitutes a critical component of automated production lines, aiming to correctly and rapidly identify the production quality of PET bottles under inspection. Commonly used target detection methods include template matching, R-CNN, Fast R-CNN, YOLO, SSD, among others. Since this research addresses a practical application in industrial production with inherent requirements for efficiency, the YOLO algorithm is selected as the visual detection method, as its fast and efficient characteristics are highly compatible with modern industrial production.

YOLO, short for You Only Look Once, is a target detection framework proposed by Joseph Redmon in 2015. Compared with the classic RCNN series algorithms, YOLO processes target detection in a unique manner: by treating the entire image as a single instance and predicting bounding box coordinates and class probabilities for these boxes. Its greatest feature, as the name suggests, is treating target detection as a single regression task, thereby achieving extremely fast response speeds and gaining widespread adoption in industry.

The original YOLOV1 divides a 416$\times$416$image into a 7\times7 grid, ultimately predicting a vector of length S\times(B\times$5 where S=7, B=2, and C represents the number of predicted classes.

YOLOV2 introduced minor improvements over the first generation. Referencing SSD's use of multi-scale feature maps for detection, the authors proposed a pass-through layer to achieve multi-scale detection. Compared with V1, YOLOV2's performance improved by 1% when using Fine-Grained Features. YOLOV3 emerged against the backdrop of rising popularity of fpn at the time. Since small targets might lose features or become diminutive after multiple convolutional layers, the approach considered combining shallow and deep features to obtain both surface and semantic features, a method that demonstrated good performance in YOLOV3.

The overall framework of YOLOV4 is based on V3, with several improvements over previous versions. For instance, it adds an SPP block on CSPDaknet53 and uses PANet instead of FPN for parameter aggregation as in YOLOV3. Beyond improving the backbone network, YOLOV4 innovated in data augmentation by introducing methods such as Mosaic and Mixup. Mosaic combines four images into one for target detection, while Mixup adds two images together. Both methods offer the advantage of rich background information and can aid object detection. Additionally, regarding the neck structure, it employs the PANet method, which not only uses two upsampling concatenation approaches but also incorporates two downsampling operations on this basis to fuse features more effectively.

The core concept of the YOLO algorithm is to use the entire image as network input and directly regress the position of bounding boxes and their class affiliations in the output layer.

First, YOLO's CNN network divides the input image into an S×S grid, with each cell predicting B bounding boxes and their confidence scores. Its accuracy is recorded as the intersection over union of the predicted box. Therefore, confidence is defined as: Pr(object), which is the IOU between the actual box and truth box pred. The position and size of the bounding box are recorded using (x, y, w, h), where (x, y) represents the center coordinates of the bounding box (relative to the top-left corner of each cell), and w and h represent the width and height of the bounding box (relative values). All four elements should remain within the range of 0-1. Finally, (x, y, w, h, c) can be used to represent the bounding box prediction values, where c represents confidence.

Specifically, in classification problems, each cell of the bounding box must pro-

vide C predicted class probability values, Pr(class_i|object). The class confidence for each bounding box is then: (x, y, w, h, c). YOLO uses a convolutional network model for feature extraction, with 24 convolutional layers and 2 fully connected layers. It primarily employs 1$×$1 convolutions as fully connected layers. For convolutional and fully connected layers, the activation function is max(x, 0.1x), while the final layer uses a linear activation function.

For the loss function, YOLO's configuration is as follows: The first term of the formula represents the error term for the bounding box center coordinates, indicating that the i-th cell contains a target and the j-th bounding box in that cell is responsible for predicting that target. The second term of the formula is the error term for the height and width of the bounding box. The third term is the confidence error term for bounding boxes containing targets. The fourth term is the confidence error term for bounding boxes not containing targets. The final term is the classification error for cells containing targets. 1_{ij}^{obj} indicates that the i-th cell contains a target. For the confidence value λ_{noobj}, it is generally set to 1.

During prediction implementation, multiple prediction values may appear, requiring the use of the non-maximum suppression algorithm to select the final result. This paper ultimately selects the yolov5 version to complete the visual component, with its structural block diagram shown in Figure 1 [Figure 1: see original paper].

3 Detection Sample Preprocessing

The collected images were annotated using labelimg, with five annotation categories defined: Label_{have}, Label_{loss}, Label_{broken}, Cap_{ture}, and Cap_{false}, corresponding to label present, label missing, label broken, cap normal, and cap missing, respectively.

After completing sample collection and data annotation, 1070 xml-format annotation files were obtained for preprocessing. (1) Files were renamed, with labels and images requiring sequential naming according to a specific format, as shown in the naming code in Figure 2 [Figure 2: see original paper]. (2) Label format conversion was performed, as illustrated in Figure 3 [Figure 3: see original paper], showing the xml format before conversion (left) and the txt format after conversion (right). (3) Yolov5 requires data division into training and test sets. This paper employs uniform distribution sampling with a training-to-test set ratio of 9:1. The final sample utilization is 107 test sets and 963 training sets.

4 Model Training and Results

After comprehensive consideration of performance and training time, YOLOV5s.pt was selected as the pretrained weights to ensure model detection accuracy and speed.

The yb.yaml file was configured with nc=2 and names: ['Cap_{ture}',

'Cap_{false}', 'Label_{have}', 'Label_{loss}', 'Label_{broken}']. The training epoch count was set to 300.

Table 1 shows the training parameters. Various mathematical metrics were calculated from the training results and plotted sequentially.

Figure 4 [Figure 4: see original paper] illustrates the relationship between the F1 score and confidence threshold from the training results. The F1 score is a classification metric representing the harmonic mean of precision and recall, ranging between 0 and 1. Generally, when the confidence threshold is low, many low-confidence samples are considered positive, resulting in high recall but low precision. When the confidence threshold is high, only high-confidence samples are considered positive, leading to more accurate detection. The F1 curve in the figure is "spacious" with its top approaching 1, indicating a large confidence threshold interval where the model performs well on the training dataset (achieving both good completeness and accuracy).

Figure 5 [Figure 5: see original paper] shows the relationship between the bounding box center point coordinates and dimensions. The top plot indicates the distribution of the center point's x-coordinate, showing most concentrations at the image's center. The first row of plots indicates the distribution of the center point's y-coordinate, with most points clearly concentrated at the image's center. The second row shows the distribution of box width, with training results indicating box widths generally less than half the image width. The final row illustrates the distribution of box height, showing box heights generally less than half the image height.

Figures 6 [Figure 6: see original paper] and 7 [Figure 7: see original paper] display the single-class precision and single-class recall for each label, respectively. Precision measures the probability that positive classes identified by a classifier are indeed positive. Its calculation formula is: Precision. In the formula, the numerator represents correctly identified positive classes, and the denominator represents all samples judged as positive. Precision = TP + FP. Recall represents the recall rate, and its calculation formula is: (3-5). As can be seen from the above figure, Recall = remains above 0.9, and Recall remains near 0.

The final epoch training result plot is shown in Figure 8 [Figure 8: see original paper]. Box represents the mean loss value, which can be seen to have reached an ideal state around epoch 200, with the curve trend slowing by epoch 280 and approaching 0.

The final two plots use precision and recall as axes, where m represents the mean, @ is a marker followed by a threshold value used to determine positive and negative samples. For example, @0.5:0.95 indicates taking thresholds from 0.5 to 0.95 in steps of 0.05 and then averaging.

After training metric analysis, real sample testing is required. For the five indicators, this paper uses both physical samples and web images for practical testing, with results shown in Table 2 .

Cap_{false} Cap_{true} Label_{lose} Label_{broken} Label_{have} Table 2 Physical Test Results Detected Label Count 93.7% As can be seen, except for Label_{broken}, which had a few undetected broken defects, the detection success rate for all other features is 100% with no missed detections.

Figure 9 [Figure 9: see original paper] shows the test results on web images (left) and physical images (right). For the visual implementation, the YOLOV5 algorithm is employed, with a sample set obtained through independent collection of PET bottles and production of labels. The total number of training samples is 1070, including 107 test sets and 963 training sets. The next step after model training is metric detection and physical sample testing, where both precision and recall of metric detection reach 0.9, and both loss mean and classification loss mean approach 0. The lowest accuracy in physical sample testing is 93.7%, with some tests achieving 100% accuracy. Both detection methods have achieved excellent results.

Note: Figure translations are in progress. See original paper for figures.