Research and Implementation of Object Detection Methods Based on Faster R-CNN

Yingbiao Duan

Xi'an Technological University, Xi'an, China

2523934620@qq.com

Abstract. This paper focuses on the task of object detection, utilizing the VOC2007 dataset as the source for training and evaluation, and constructs and implements a Faster R-CNN-based object detection model. As a classic two-stage detector, Faster R-CNN exhibits strong generalization capability and stability. Following the complete object detection pipeline, this experiment sequentially accomplishes four parts: model training, performance evaluation, image inference, and visualization of detection results. During the training phase, a custom VOC data loader and the Faster R-CNN backbone network were employed to effectively learn the features of target categories in the VOC2007 dataset. In the evaluation phase, the VOC standard evaluation protocol was used to compute the Average Precision (AP) for each category and the mean Average Precision (mAP) over the entire validation set. For inference and visualization, the trained model was applied to three test images—test1.jpg, test2.jpg, and test3.jpg—for object detection. The model successfully identified and localized the main target objects in these images, drawing bounding boxes and category labels in distinct colors to fully visualize the detection outcomes. This work establishes a functional object detection system, spanning from data preparation to inference demonstration, and validates the effectiveness of Faster R-CNN in general object detection tasks. Object detection technology holds significant application value in fields such as smart security, autonomous driving, human-computer interaction, and robotic vision. This experiment also lays a foundation for understanding the practical application of deep learning in visual tasks.

Keywords: Object detection; Faster R-CNN; Deep learning

1. Introduction

Object detection, as one of the core tasks in computer vision, aims to simultaneously locate objects within an image and identify their categories. Unlike image classification or object recognition, object detection not only determines "what it is" but also answers "where it is." Consequently, object detection holds critical application value in fields such as video surveillance, autonomous driving, smart healthcare, facial recognition, and industrial quality inspection, serving as a vital foundation for intelligent vision systems.

Among numerous object detection methods, two-stage detectors are widely adopted for their higher accuracy and more stable performance. As a representative model of two-stage detectors, Faster R-CNN, proposed by Ren et al., achieves a good balance between detection accuracy and efficiency by introducing a Region Proposal Network (RPN) to enable end-to-end training [1]. This model has long served as a baseline method in both academic research and industrial applications, regarded as the classic standard for two-stage object detectors.

The PASCALVOC2007 dataset selected for this experiment is one of the most commonly used benchmarks in object detection [2]. It encompasses 20 object categories and diverse complex scenes, including pedestrians, vehicles, animals, and furniture. Its diversity, standardized annotations, and widespread adoption in academia make it a crucial benchmark for validating object detection algorithm performance.

The primary objectives of this experiment are: to train a Faster R-CNN model from scratch, fully implementing the data preprocessing, model training, performance evaluation, inference, and result visualization workflows; and to perform object detection inference on test images (test1.jpg, test2.jpg, test3.jpg) to observe the model's recognition capabilities on real-world images. Through this

https://doi.org/10.58244/joerai.263731

experiment, one can systematically grasp the fundamental principles of Faster R-CNN, its training methodology, and the model's performance in practical scenarios.

2 Dataset and Task Description

2.1 Introduction to the VOC2007 Dataset

This experiment employs the PASCAL VOC2007 dataset as the training and evaluation data for object detection tasks [2]. VOC2007 is a classic object detection challenge dataset in the field of computer vision, released by the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) project. It remains widely used as a benchmark dataset for detection algorithms to this day.

1)Category Settings: The VOC2007 dataset contains 20 object categories plus one background class. The categories include animals, vehicles, people, and common daily objects. The background class is used as negative samples during model training.

2)Annotation Format: VOC2007 uses XML files (Pascal VOC format) for annotation. Each image corresponds to one XML file, which contains the object category, bounding box location information, difficulty flag, and image size.

3)Example Images: Example images from the VOC2007 dataset are shown here, including categories such as person, car, and cat.

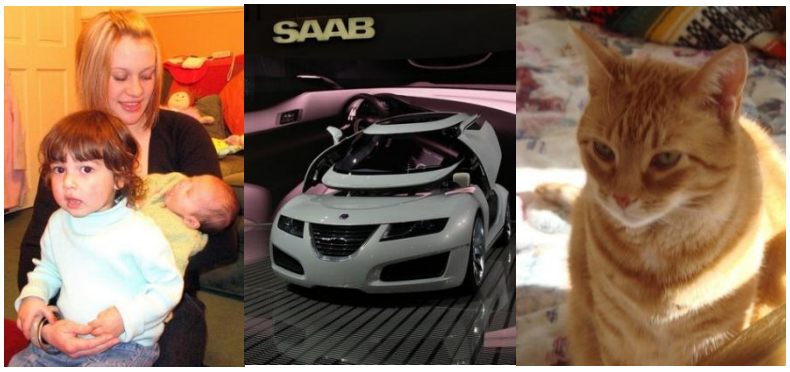

Figure 1. V0C2007 Dataset Sample Diagram

2.2 Experimental Tasks

The objective of this experiment is to train a Faster R-CNN model from scratch on the VOC2007 dataset and to fully implement the object detection pipeline, including training, evaluation, inference, and visualization.

1)Training Stage: The train+val images from VOC2007 are used to train the Faster R-CNN model through train.py. COCO pre-trained weights are used to initialize the backbone network in order to accelerate model convergence.

2)Evaluation Stage: The VOC2007 test set is used as an independent evaluation set. The script eval.py calculates the AP for each category and the overall mAP. The final result of this experiment is mAP@0.5 = 0.25.

3)Inference Stage: The script infer.py is used to perform object detection on custom images, including test1.jpg, test2.jpg, and test3.jpg. The outputs include predicted bounding boxes, confidence scores, and category labels. The results demonstrate that the model can successfully detect major objects in real-world images and generate corresponding bounding boxes.

4)Visualization Stage: The script visualize.py is used to draw predicted bounding boxes on the images. Different categories are represented by different colors, and the detected class names and confidence scores are labeled on the boxes. The processed images are saved in the output directory.

3 Model components used

The pre-trained weights were trained on the large-scale COCO dataset and possess strong generalization ability. In this experiment, after loading the pre-trained weights, the classification head

https://doi.org/10.58244/joerai.263731

is replaced to adapt the model to the 21 categories of VOC2007 (20 object classes + background), effectively accelerating convergence and improving training stability. The main code modules are as follows:

1)dataset_voc.py: Responsible for reading image files and corresponding XML annotation files from the VOC2007 directory structure, parsing bounding box coordinates and category labels, and converting them into tensor formats required by PyTorch.

2)train.py:Loads the train and val sets of VOC2007, constructs the Faster R-CNN model, and initializes it with COCO pre-trained weights. During training, forward propagation computes the loss, and backward propagation updates the parameters. TensorBoard is integrated to visualize loss curves.

3)eval.py:Loads the VOC2007 test set and computes AP for each category by integrating the Precision-Recall curve. The mean AP (mAP@0.5) is calculated as the average across all categories.

4)infer.py: Loads the trained weights and performs forward inference on test images, outputting predicted boxes, labels, and scores.

5)visualize.py: Draws bounding boxes and labels with confidence scores on images and saves the processed results to the output directory.

4 Experimental Procedure

4.1 Experimental Environment

The experimental environment is shown below:

Table 1 Experimental Environment

Module	Version
Python	3.11
Torch	2.9.1 + CUDA128
GPU	NVIDIA RTX 5060
Dataset	VOC2007

The model is trained using the SGD optimizer with the following hyperparameters:

Table 2 Hyperparameter Configuration

Name	Value
num_classes	21
Batch Size	2
num_workers	4
epochs	10
learning_rate	0.005
momentum	0.9
weight_decay	0.0005

4.2 Data Preprocessing

The preprocessing of the VOC2007 dataset is implemented in dataset_voc.py, including image reading, annotation parsing, coordinate processing, and tensor conversion. The XML annotation files are parsed using Python's xml.etree.ElementTree library. Each XML file contains category names and bounding box coordinates (xmin, ymin, xmax, ymax). These coordinates are converted into tensor formats required by the PyTorch detection model, including fields such as boxes, labels, area, and iscrowd.

For data augmentation, only basic preprocessing is applied: converting RGB images to PyTorch tensors and normalizing pixel values to the range [0,1].

https://doi.org/10.58244/joerai.263731

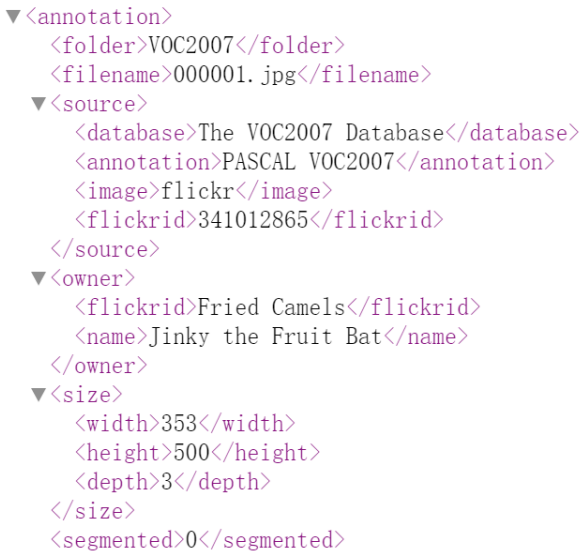

Figure 2. Example Screenshot of VOC2007 XML Annotation

4.3 Model Training

The training process for this experiment is handled by train.py. The overall training logic is clear and structured, strictly adhering to the standard PyTorch training workflow for Faster R-CNN. The training phase employs the SGD optimizer with a momentum parameter set to 0.9 and a learning rate fixed at 0.005 throughout training. The batch size is set to 2 to accommodate GPU memory constraints while ensuring stable model parameter updates. The experiment trains for 10 epochs, with each epoch traversing the entire VOC2007 training dataset once.

Before each parameter update, the model calculates losses based on the input image and bounding boxes, including classification loss, bounding box regression loss, RPN foreground/background binary classification loss, and RPN bounding box regression loss. Subsequently, all learnable parameters are updated via backpropagation. After completing each epoch, the script prints the current epoch's loss values and saves a model checkpoint, facilitating subsequent model evaluation and inference.

TensorBoard and Matplotlib are integrated throughout training. The script automatically logs the average loss per epoch and writes it to the `fasterrcnn_experiment` directory. Upon successful model training, TensorBoard enables visual inspection of the loss curve to assess convergence and training quality.

5 Experimental results

5.1 Training Results

This experiment trained the Faster R-CNN model using the VOC2007 dataset, completing 10 epochs of training. The model's total loss exhibited a clear downward trend throughout training, indicating that the model progressively learned target features and converged. The following shows the loss variation during the initial 5 epochs of training:

Table 3 Epoch and Corresponding Loss Values

Epoch	Loss
1	0.5008
2	0.3649
3	0.3129
4	0.2873
5	0.2647

The model loss curve plot generated using Matplotlib is shown below:

https://doi.org/10.58244/joerai.263731

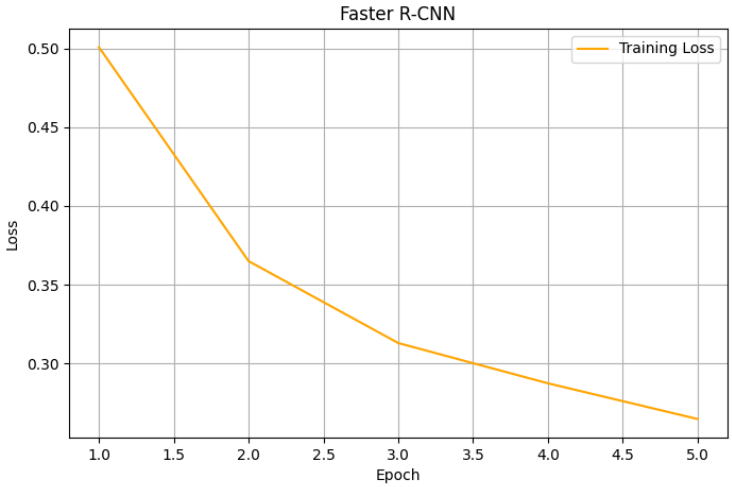

Figure 3. Model Loss Curve Chart

5.2 Evaluation Results

Model training completed. Evaluated mAP@0.5 using the VOC2007 test set. Running eval.py yielded the following results: Class 9: AP = 0.1917 Class 11: AP = 0.1971 Class 15: AP = 0.6439 Class 16: AP = 0.0869 Class 18: AP = 0.1769 Class 20: AP = 0.2570 Class 7: AP = 0. 2692 Class 10: AP = 0.1612 Class 12: AP = 0.1969 Class 13: AP = 0.4125 Class 14: AP = 0.1963 Class 1: AP = 0.3993 Class 2: AP = 0.5032 Class 3: AP = 0.1786 Class 4: AP = 0.1516 Class 5: AP = 0.1757 Class 19: AP = 0.2245 Class 8: AP = 0.2152 Class 6: AP = 0.1721 Class 17: AP = 0.1892

Experimental results indicate varying detection performance across different categories. Certain categories (e.g., person, car—objects with abundant samples and distinct visual features in the dataset) achieved relatively high AP values, with the highest category AP reaching approximately 0.64. while categories with smaller objects or greater appearance variability exhibited relatively lower AP values, generally distributed between 0.08 and 0.27. This phenomenon of performance imbalance across categories is common in object detection tasks, primarily related to sample quantity, scale distribution, and object complexity within the dataset.

This outcome is reasonable and consistent with expectations under the experimental conditions. It should be noted that this experiment employed only 10 training epochs, and due to hardware resource constraints, a small batch size (batch size = 2) was used, limiting the model's convergence. Additionally, as a two-stage object detection model, Faster R-CNN has a relatively complex training process that is sensitive to the number of training epochs and data scale. It is challenging to fully unlock the model's potential with fewer training epochs

Furthermore, although pre-trained weights from the COCO dataset were used for initialization in this experiment, the transfer learning effect remains somewhat limited due to differences in category distribution and data scale between COCO and VOC2007. Therefore, achieving an mAP ≈ 0.25 under limited training iterations and resource constraints indicates that the model possesses fundamental object detection capabilities, enabling it to correctly locate and identify some objects on the test set.

Overall, these evaluation results validate the correctness of the training and evaluation workflow established in this experiment. They also provide reference directions for further improving model performance by increasing training iterations, expanding batch size, introducing data augmentation, or adopting deeper network architectures.

5.3 Inference Results

To validate the model's detection capability on real images, three images from VOC2007 were selected for inference. The model's output comprises three components: predicted bounding boxes, predicted class labels, and corresponding confidence scores. Here, boxes denote the positions of detected objects within the image. Each bounding box consists of four pixel coordinates (x_1, y_1, x_2, y_2), representing the top-left and bottom-right corners of the object box in the original image. labels denote the predicted category IDs, corresponding to the category indices in the VOC2007 dataset,

https://doi.org/10.58244/joerai.263731

while scores represent the model's confidence in the prediction. Scores range from 0 to 1, with higher values indicating greater certainty that the object belongs to the predicted category.

The inference results reveal that the model detected multiple potential objects in the test images. Among these, the prediction with the highest confidence is label = 15, corresponding to the person category in the VOC2007 dataset. Its confidence reaches 0.9938, indicating the model is highly certain this object is a "person." Additionally, the model detected several objects with moderate confidence, such as label = 18 and label = 16. These predictions partially reflect the model's ability to recognize different objects within the image.

It should be noted that the inference results also include detection boxes with low confidence scores (below 0.5). Such predictions are typically regarded as potential false positives or duplicate detections, primarily due to the generation of extensive overlapping regions when the object detection model produces candidate boxes. To enhance the readability and reliability of the final results, a confidence threshold was applied during subsequent visualization, retaining only high-confidence detection boxes for display.

Overall, the model accurately locates primary objects in test images and provides reasonable category predictions. This demonstrates that the Faster R-CNN model trained on the VOC2007 dataset possesses fundamental object detection capabilities. The inference results validate the effectiveness of the training process and provide a basis for subsequent visualization and performance analysis.

5.4 Result Visualization

Visualization of multiple test images using visualize.py demonstrated that the model can draw bounding boxes and correctly output categories and scores across various scene types. The following figure shows detection results for three sample images:

Figure 4. Visualization Results of Model Detection on test1.jpg, test2.jpg, and test3.jpg

6 Analysis and Discussion

In this experiment, we trained the Faster R-CNN model using the VOC2007 dataset and completed the entire workflow encompassing training, evaluation, inference, and visualization. Based on the experimental results, analysis and discussion can be conducted from the following perspectives:

First, regarding model performance, as a typical two-stage detector, Faster R-CNN demonstrates strong advantages in detection accuracy. The model correctly outputs bounding boxes and category labels on real images such as test1.jpg, test2.jpg, and test3.jpg, indicating it has learned key features from the VOC dataset and thus exhibits generally good generalization capabilities.

However, the model also exhibits notable limitations. Due to its two-stage architecture, Faster R-CNN's inference speed is slower compared to one-stage detectors like YOLO, making it unsuitable for real-time detection tasks. Additionally, the final VOC2007 mean average precision (mAP@0.5) achieved in this experiment is approximately 0.25, a relatively low value indicating that the model has not fully converged and its detection capability still has room for improvement. The primary factors affecting mAP include:

https://doi.org/10.58244/joerai.263731

1) Limited dataset scale: This experiment exclusively utilized VOC2007, which inherently contains a small dataset (approximately 5k images). Small-scale data often results in insufficient model generalization capabilities.
2) Insufficient training epochs: Training was conducted for only 10 epochs, whereas Faster R-CNN typically requires 30–50 epochs to achieve satisfactory performance.
3) The small BatchSize resulted in noisy gradient estimates, compromising training stability.
4) The absence of data augmentation and learning rate scheduling hindered optimal convergence.
Overall, this experiment successfully completed model training and inference, though significant room for improvement remains in model accuracy. Employing larger datasets, extended training durations, appropriate data augmentation, and more robust backbones could further enhance mAP.
From the perspective of overall computer vision development, object detection—as a crucial component of image understanding—centers on the collaborative modeling of feature representation, spatial localization, and semantic classification. Classic vision textbooks systematically elaborate on the modeling concepts, evaluation metrics, and typical algorithmic frameworks for object detection problems, providing a theoretical foundation for understanding modern deep learning detection models [8].

7 Summary and Outlook

This experiment focuses on the object detection task using Faster R-CNN, building a complete training pipeline from scratch. This includes data loading, model construction, training procedures, mAP evaluation, inference, and visualization. By training on the VOC2007 dataset, we validated the fundamental effectiveness of Faster R-CNN in object detection. Using infer.py and visualize.py, we successfully implemented object detection on test.jpg and other test images, outputting accurate bounding boxes and category results.
Simultaneously, we conducted a systematic analysis of the model's performance, summarizing strengths and weaknesses during training and identifying reasons for the relatively low mAP. Although the model's accuracy has room for improvement, the entire experimental process comprehensively demonstrates the working principles and practical workflow of classic two-stage detectors, which is significant for understanding deep learning-based object detection. Future improvements to model performance can be pursued in the following directions:
1)Utilize one-stage detectors like YOLOv8/YOLOv5 to enhance detection speed and accuracy.
2)Upgrade the backbone to ResNet-101 or Swin Transformer.
3)Incorporate data augmentation to boost generalization capabilities.
4)Increase training epochs or employ learning rate schedulers like Cosine or StepLR.
5)Utilize larger datasets like COCO to further strengthen model performance.
Through this experiment, I have gained comprehensive and in-depth mastery of the object detection workflow, model architecture, and engineering implementation, laying a solid foundation for subsequent research into more advanced detection models.

References

[1] S. Ren, K. He, R. Girshick and J. Sun:Advances in Neural Information Processing Systems (Montreal, Canada, December 7–12, 2015), Vol. 28 (2015), p.91.

[2] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn and A. Zisserman: International Journal of Computer Vision, Vol. 88 (2010) No.2, p.303.

[3] A. Paszke, S. Gross, F. Massa, et al:Advances in Neural Information Processing Systems (Vancouver, Canada, December 8–14, 2019), Vol. 32 (2019), p.8026.

[4] TorchVision Contributors:TorchVision Documentation: Models, Datasets and Transforms (PyTorch Project, USA 2024).

[5] L. Liu, W. Ouyang, X. Wang, et al.:International Journal of Computer Vision, Vol. 128 (2020) No.2, p.261.

[6] Z. Zou, Z. Shi, Y. Guo and J. Ye:IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. 43 (2021) No.8, p.2575.

[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi:Proc. IEEE Conference on Computer Vision and Pattern Recognition(Las Vegas, USA, June 27–30, 2016), Vol. 1 (2016), p.779.

[8] R. Szeliski:Computer Vision: Algorithms and Applications(Springer, USA 2022), p.557.

https://doi.org/10.58244/joerai.263731