Intelligent Document Tampering Detection CVPR Experiment Report

Team Name — 专业课和我作队

郑光铭 22302010014

陈柏滔 22302010008

黄宸一 22302010011

Abstract

In this experiment, the Faster R-CNN and YOLO models were used as baselines and innovated upon to complete the document tampering detection task. By introducing region threshold filtering, multi-scale data augmentation, and adjusting parameters such as the number of training epochs, batch size, and unified image size, the YOLO11m model was used to achieve the best detection performance, with the highest Micro-F1 score reaching 93.68.

1. Task Introduction

1.1. Task Background

Verification of vouchers and documents is a perennial topic in financial scenarios and holds extremely important application value in various fields. In particular, the exploration of tampering detection in non-standard documents has always been an important research direction in both industry and academia. With the continuous development of information technology, a large amount of data and information is rapidly flooding into our lives, and the authenticity and reliability of these data and information are crucial. The rapid innovative growth in multimedia technology has made it easily accessible to play with image editing software, which could result in the desired manipulation of the original content. Images are often used as solid evidence in legal proceedings.[1, 4] In the context of digital financial services, the verification of vouchers and documents is even more important. The essence of digital finance is based on the flow and exchange of data and information, and the authenticity and reliability of these data and information are key to whether digital financial services can proceed smoothly. For example, in digital payment scenarios, the authenticity of user payment vouchers directly affects the security and efficiency of payments; in digital lending scenarios, the authenticity of the personal loan information provided by borrowers is the foundation for assessing their repayment ability and risk level. Moreover, the transaction processes and business models in digital finance also raise higher requirements for the verification of vouchers and documents. In digital financial transactions, the verification of vouchers and documents not only needs to ensure their authenticity and reliability, but also needs to guarantee that they are tamper-proof and cannot be forged, to ensure the security and credibility of digital financial transactions. Therefore, the verification of vouchers and documents holds critical application value in digital finance, and developing a set of universal tampering detection algorithms to ensure the authenticity of data collection in various complex scenarios is not only of great business value but also helps reduce various fraud risks, providing a reliable data and information foundation for digital financial services and promoting the stable development of the digital finance industry.

In recent years, deep neural networks (DNNs), such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Generative Adversarial Networks (GAN) have shown to be capable of characterizing the complex statistical dependencies from high-dimensional sensory inputs and efficiently learning their hierarchical representations, allowing them generalize well across a wide variety of computer vision (CV) tasks, including image forgery detection.[2, 3]

In this experiment, we will build upon existing detection models and make adjustments and innovations to design an algorithm that identifies tampered regions in voucher images.

1.2. Task Data

1.2.1. Data Format

The image data used for training and validation can be categorized into shooting documents, receipts, scanned documents/PDFs, and street view photos, etc. The formats of the images include jpg, JPEG, and png, and each image file has a corresponding index.

Some of these images may have been tampered with using common methods such as copy-move, splicing, or removal, as well as more recent deep learning-based image generation techniques.

Tampering localization has also become a research hotspot in image tampering detection tasks in recent years. In practical applications, merely detecting whether an image has been tampered with is not sufficient. It is necessary to also identify where the tampering has occurred in the image.[6] For simplicity, in this experiment, all tampered regions are assumed to be rectangular.

The label file stores the tampered region (which may be empty) corresponding to each image file, and the specific format is as follows:

[. . .

{“id”: “fileName”, “region”: [[x_1, y_1, x_2, y_2]]},

. . . ,

{“id”: “fileName2”, “region”: []},

. . .]

fileName refers to the corresponding image file name. If the image is tampered with, the region is $[x_1, y_1, x_2, y_2]$, representing the pixel coordinates of the top-left and bottom-right corners of the rectangular tampered area. If not tampered with, the region is empty.

1.2.2. Data Split

The images in the above format are divided into a training set and a test set, with the respective quantities of 13,000 images and 5,000 images.

The training set has a corresponding label file ‘label_train.json’ for training, which stores the tampered regions (which may be empty) for each image in the training set. The filenames in the training set are all in the format ‘train_$index$’, such as ‘train_8359.jpg’.

For the YOLO model, we split 20% of the training set as validation set, which is used for testing the model’s training results to avoid overfitting.

The test set is held by the competition organizers and serves as the final evaluation basis. It is not public to avoid participants directly training on the test set. The trained model will be evaluated on the test set, generating a prediction label file that will be compared with the actual tampered regions.

1.3. Evaluation Criteria

For the training results, the predictidon accuracy needs to be calculated. In this experiment, we use the micro-average $Micro-F1$ as the prediction accuracy metric. The predicted $region$ field corresponding to each id is compared with the true labels, and based on a threshold, the values for True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are computed. These values are then used to calculate precision (P) and recall (R), and finally, the $Micro-F1$ score is computed. The specific calculation formulas are as follows:

$$P_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{1}$$

$$R_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{2}$$

$$F1_{micro} = \frac{2 \times P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{3}$$

2. Experiment Steps

In this experiment, we mainly use Faster R-CNN and YOLO for testing. First, we match the existing models with the experimental tasks, complete the baseline reproduction, and then adjust and innovate the models. Through ablation experiments, we compare and find the best results.

2.1. Environment Configuration

Python: 3.11.8
GPU: RTX4090

2.2. Model Configuration

2.2.1. Faster R-CNN

Model: Faster R-CNN with ResNet-50-FPN backbone
Dataset:
Training image path:
/home/zgm2024/CVpj/train/images
Label file path:
/home/zgm2024/CVpj/train/label_train.json
Data Processing: Custom dataset class TamperingDataset is used to load images and annotations, and skip images with regions smaller than 10 pixels.
Training Settings:
Loss function:
Sum of classification and regression losses of Faster R-CNN
Optimizer:
AdamW, initial learning rate $lr = 0.001$, weight decay $weight\_decay = 0.01$
Scheduler:
StepLR, decay factor of 0.1 every 3 epochs, i.e., $step\_size = 3,\ gamma = 0.1$
Batch size: 4
Device: GPU (if available)
Pre-trained Weights: The model is initialized from torchvision's pre-trained weights
Output:
The trained model is saved at the path:
/home/zgm2024/CVpj/train/model/tampering_detection_model_updated.pth

2.2.2. YOLO

Model: YOLOv8n
Dataset Configuration:
YAML configuration file path:
/home/zgm2024/CVpj/yolo/datasets/yolo_config.yaml
Training Settings:
Initial uniform image size: 640×640 pixels

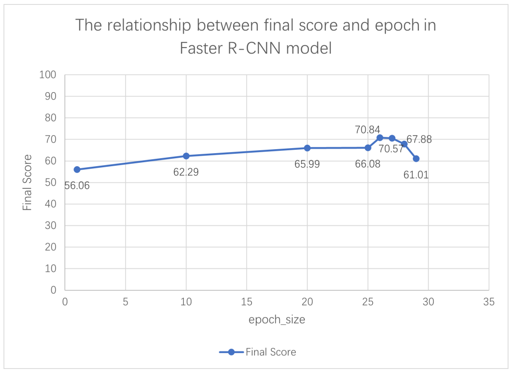


Figure 1. Micro-F1 vs. Epochs for Faster-RCNN Model

Batch size: 16
Device: GPU 0
Pre-trained Weights: Initialized from yolov8n.pt
Output:
Training logs and weights are saved at:
/home/zgm2024/CVpj/yolo/runs/train/tampering_detection

2.3. Baseline Reproduction

In this section, we perform baseline reproduction for the Faster R-CNN and YOLO models, changing only the number of training epochs to observe the training effects and select the better model.

2.3.1. Faster R-CNN

The model is trained with parameters $lr = 0.001,\ weight\_decay = 0.01,\ step\_size = 3,\ gamma = 0.1$. The training epochs are gradually increased, and the relationship between Micro-F1 score and the number of epochs for the Faster R-CNN model is shown in the line chart Fig. 1.

The initial Micro-F1 score of the Faster R-CNN model is 56.06. By increasing the epochs, we find that at epoch 26, the score reaches the maximum of 70.84. After further training, the score decreases, indicating that the model may have overfitted.

The Faster R-CNN model is easy to call and has a simple model definition. After 26 epochs, the model achieves the best score, but the training process is slow. Data augmentation significantly increases training time, and the results are not ideal. However, Faster R-CNN helps to under-

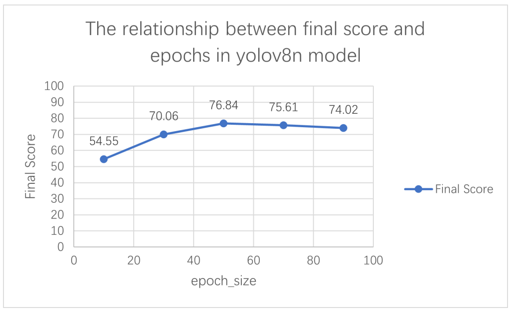


Figure 2. Micro-F1 vs. Epochs for YOLOv8n Model

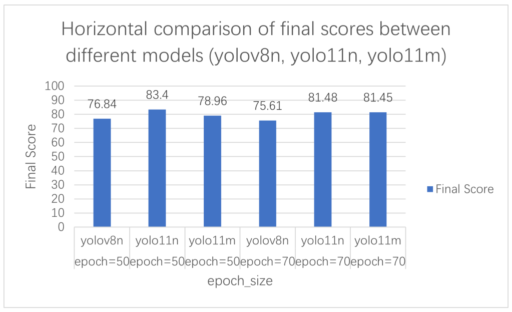
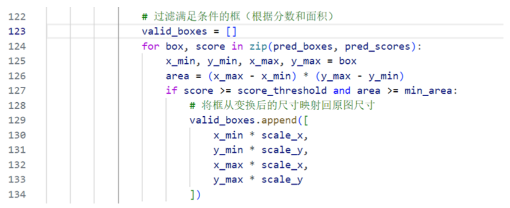


Figure 3. Comparison of YOLOv8n, YOLO11n, and YOLO11m



```
# 过滤满足条件的框（根据分数和面积）
valid_boxes = []
for box, score in zip(pred_boxes, pred_scores):
    x_min, y_min, x_max, y_max = box
    area = (x_max - x_min) * (y_max - y_min)
    if score >= score_threshold and area >= min_area:
        # 将框从变换后的尺寸映射回原图尺寸
        valid_boxes.append([
            x_min * scale_x,
            y_min * scale_y,
            x_max * scale_x,
            y_max * scale_y
        ])
```

Figure 4. Code Modification for Region Threshold Filtering

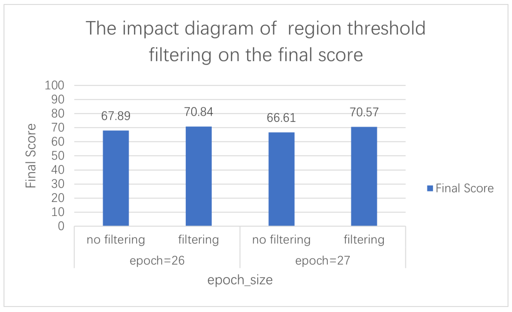


Figure 5. Impact of Region Threshold Filtering in Faster R-CNN

stand the ResNet-50 network structure and the significance of hyperparameter tuning for model training.

2.3.2. YOLOv8n

Training with parameters $imgsz = 640$, $batch = 16$ and predicting with parameter $conf = 0.25$, we gradually increase the training epochs. The relationship between Micro-F1 score and epochs for the YOLOv8n model is shown in the line chart Fig. 2.

The number of training epochs has a significant effect on the score. After rough estimation, the model reaches its maximum score of 76.84 after around 50 epochs, surpassing Faster R-CNN. The YOLOv8n model trains much faster than Faster R-CNN, proving that the YOLO model is more flexible. Therefore, we consider using a larger model for training based on YOLO.

2.3.3. YOLO Model Version Comparison

Based on the YOLO model, we attempt to train with larger and newer models. Based on the baseline experiment results of YOLOv8n, we train YOLO11n and YOLO11m, and compare the results of epoch 50 and epoch 70 in a bar chart Fig. 3.

From the chart, it is clear that for the updated YOLO11 model, the accuracy is significantly higher than YOLOv8 when all other parameters are consistent. Although YOLO11m has a larger number of parameters, its accuracy is lower than YOLO11n in the chart. This suggests that the accuracy of the model is not only related to the number of parameters but also to other parameters (such as epoch and batch). When the epoch and batch are smaller, YOLO11m does not necessarily outperform YOLO11n. The subsequent SubSec. 2.4 section will carefully compare the effects of epoch and batch on YOLO11m, as the highest score is achieved by YOLO11m.

2.4. Method Innovation

2.4.1. Innovation 1: Region Threshold Filtering

To improve the model's detection performance on different images and provide multiple detection intervals, we modify the Faster R-CNN evaluation logic to set a confidence threshold for generating tampered regions. The code modification is shown in Fig. 4.

From the highest score point in Fig. 1, we compare the 26th and 27th epochs with and without region threshold filtering. The results are shown in the bar chart Fig. 5.

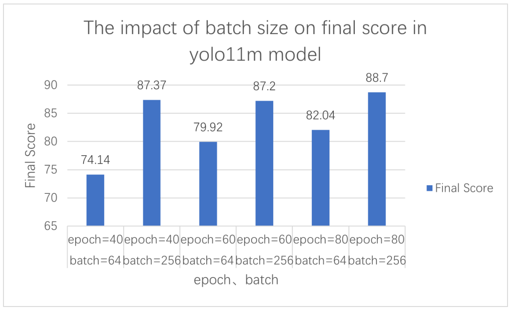


Figure 6. Impact of Batch Size in YOLO11m Model

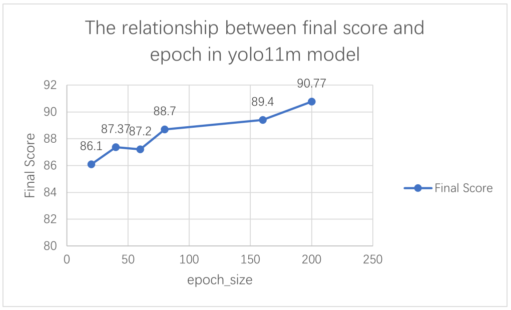


Figure 7. Micro-F1 vs. Epochs for YOLO11m Model

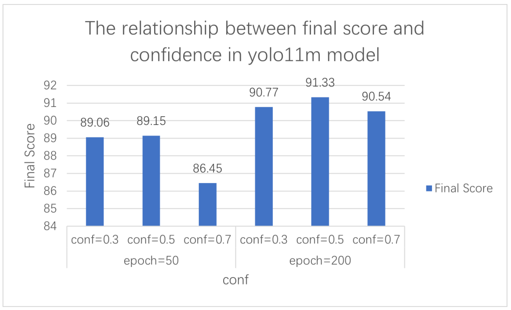


Figure 8. Micro-F1 vs. Confidence Threshold in YOLO11m

From the chart, we can see that the score has significantly improved with the innovation of region threshold filtering, indicating that this method has a clear effect in this experiment.

2.4.2. Innovation 2: Adjusting Training Parameters and Configuration Files

In this section, we adjust the parameters for YOLO11m and gradually converge to the highest possible score for the YOLO model.

First, we modify the batch size and increase the number of training epochs. These two parameters have a significant impact on the prediction score. We train with batch sizes of 64 and 256 for 40, 60, and 80 epochs, and the results are shown in the bar chart Fig. 6.

From the chart, it is clear that increasing the batch size greatly helps the model's score, with a significant improvement. Increasing the batch size greatly enhances the model's ability under the same number of epochs.

With a batch size of 256, we increase the training epochs and obtain the line chart of prediction scores as epochs increase Fig. 7.

From the chart, it can be seen that the YOLO11m model with batch = 256 is very powerful. While there are slight fluctuations during the increase in epochs, the overall trend is an upward one. At epoch = 200, the Micro-F1 score exceeds 90, which is a very high accuracy.

After adjusting the batch size and training epochs to make the model converge, we fine-tune the model by adjusting the confidence threshold ($conf$) and uniform image size ($image\_size$) to achieve a slight improvement in the score.

We set confidence thresholds to 0.3, 0.5, and 0.7, and compare the predicted results for epochs 50 and 200, as shown in the bar chart Fig. 8.

From the chart, it can be roughly seen that there is no definite pattern or conclusion between the confidence threshold and the model's score. Both too high and too low confidence thresholds lead to a decrease in model performance. The optimal confidence threshold varies after each training session. Therefore, confidence can only be used as a trial-and-error method to improve the model after finding the best one, and cannot directly determine the optimal confidence threshold for the model.

The results obtained from further testing, not shown in the figure, indicate that the model score reaches its optimal value of 93.68 when adjusting $conf = 0.2$.

Finally, we increase $image\_size$ to 800, which is the uniform size to which the images are scaled before training in the YOLO model. We compare the predicted results for epochs 150 and 200, as shown in the bar chart Fig. 9.

Increasing the image size, though requiring more com-

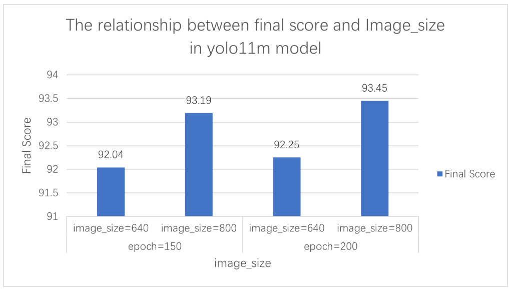
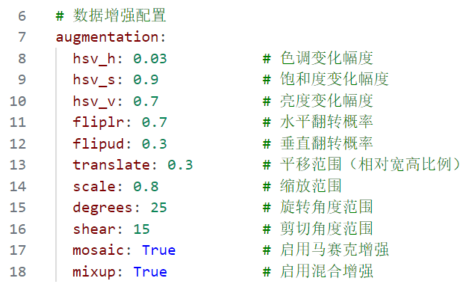


Figure 9. Micro-F1 vs. Image Size in YOLO11m Model



```
# 数据增强配置
augmentation:
  hsv_h: 0.03                    # 色调变化幅度
  hsv_s: 0.9                     # 饱和度变化幅度
  hsv_v: 0.7                     # 亮度变化幅度
  fliplr: 0.7                    # 水平翻转概率
  flipud: 0.3                    # 垂直翻转概率
  translate: 0.3                 # 平移范围（相对宽高比例）
  scale: 0.8                     # 缩放范围
  degrees: 25                    # 旋转角度范围
  shear: 15                      # 剪切角度范围
  mosaic: True                   # 启用马赛克增强
  mixup: True                    # 启用混合增强
```

Figure 10. Data Augmentation Parameter Configuration

puting power, leads to better model performance because more information is preserved in the images.

2.4.3. Innovation 3: Multi-scale Data Augmentation

Due to data augmentation operations such as image flipping, the data volume can be expanded to make the network-trained model more accurate[5], which proved feasible for optimizing the scores. Therefore, we also tried multi-scale data augmentation.

To enhance the model's robustness to tampering at different scales, we introduce multi-scale image resizing during the training of YOLO11m. The data augmentation parameter configuration is shown in Fig. 10.

We compare the prediction scores for epochs 150 and 200 with and without data augmentation, as shown in the bar chart Fig. 11.

Before data augmentation, no matter what tuning method was applied, the highest score the model could achieve was 90.71. However, with data augmentation and the same training parameters, the model score reached 92.25, indicating that data augmentation has a significant effect on the YOLO model.

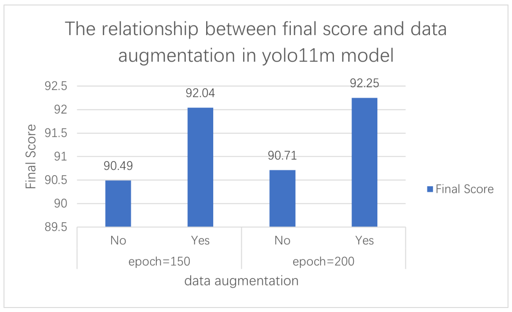


Figure 11. Micro-F1 vs. Data Augmentation in YOLO11m Model

Model	Filter	Adjustment	Micro-F1
Faster R-CNN	No	No	56.06
	No	Yes	67.89
	Yes	Yes	70.84

Table 1. Ablation Experiment Results Table for Faster R-CNN

Model	Augmentation	Adjustment	Micro-F1
YOLO	No	No	76.84
	No	Yes	90.71
	Yes	Yes	93.68

Table 2. Ablation Experiment Results Table for YOLO

2.5. Ablation Experiment Results

Based on the comparisons in SubSec. 2.4 between Faster R-CNN models with and without region threshold filtering and parameter adjustments, and YOLO models with and without multi-scale data augmentation and parameter adjustments, we present the results in tables Tab. 1 and Tab. 2 using Micro-F1 as the metric.

Region threshold filtering significantly improved Faster R-CNN's performance on smaller tampered regions.

Multi-scale data augmentation improved YOLO11m's generalization ability across different tampering patterns.

Adjusting training parameters and configuration files also resulted in an increase in score over the model's baseline.

3. Experimental Results and Analysis

3.1. Best Results

The highest Micro-F1 score achieved so far is 93.68. This score is shown in the competition website as Fig. 12.
The experimental parameters that yielded the highest score are as follows:
Model: YOLO11m
Number of Training Epochs: 200
Unified Image Size during Training: 800×800 pixels
Batch: 256
Prediction Confidence: 0.2

3.2. Result Analysis

Based on the analysis and comparison of the performance of each model during the experiments in Sec. 2, the following conclusions can be made:

3.2.1. Performance Analysis of Faster R-CNN

Faster R-CNN is a precision-first two-stage object detection model that demonstrated high reliability and accuracy in this experiment, particularly in locating and classifying small targets. The experiment showed the following:

The baseline Micro-F1 score was 56.06, with limited performance when no region threshold filtering or parameter tuning was applied.

After adding region threshold filtering, the Micro-F1 score improved to 70.84, significantly enhancing the model's ability to detect small tampered regions.

However, Faster R-CNN has a slower training speed and higher computational resource demands, especially when data augmentation is used, leading to significantly increased training time. While the performance is acceptable, it is not suitable for efficient large-scale detection solutions.

3.2.2. Performance Analysis of the YOLO Model

YOLO models are known for their fast detection performance and strong generalization capability, particularly in small object detection and real-time applications. In the experiment:

YOLOv8n achieved a baseline Micro-F1 score of 76.84, significantly higher than Faster R-CNN.

Although YOLO11m has a larger number of parameters, it performed optimally with sufficient training (such as increasing epochs and batch size), with the Micro-F1 score exceeding 93. This indicates that YOLO models have strong adaptability to large-scale training.

3.2.3. Horizontal Comparison Between Models

Faster R-CNN is suitable for detection tasks that require high precision, but it has significant disadvantages in training speed and scalability.

YOLO series models, especially YOLO11m, strike a better balance between speed and accuracy, outperforming both Faster R-CNN and YOLOv8.

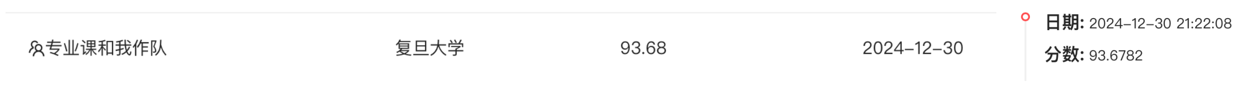
3.2.4. Effectiveness of Innovative Methods

Region Threshold Filtering: Introducing region threshold filtering for Faster R-CNN significantly improved the model's ability to recognize small targets, which was a key method for enhancing its performance.

Training Parameter Adjustment: For YOLO11m, adjusting the batch size (from 64 to 256), increasing the number of training epochs (up to 200) and increasing the $image\_size$ (up to 800) gradually improved the Micro-F1 score, indicating that larger batch training aids in enhancing model performance.

Multi-Scale Data Augmentation: By incorporating multi-scale data augmentation with different scaling sizes, the robustness of the model in detecting various tampering patterns was significantly improved, with the Micro-F1 score of YOLO11m rising from 90.71 to 92.25. This confirmed the importance of multi-scale multi-scale data augmentation in improving the model's generalization ability.

In summary, this experiment validated the efficiency and strong performance of YOLO11m, and the innovative methods (such as region threshold filtering and multi-scale data augmentation) significantly improved the model's Micro-F1 score, ultimately reaching 93.68. This demonstrates that through careful parameter tuning and the introduction of innovative techniques, it is possible to effectively optimize model performance and improve the accuracy of document tampering detection.


&专业课和我作队 复旦大学 93.68 2024-12-30

日期: 2024-12-30 21:22:08

分数: 93.6782

Figure 12. Screenshot of Final Score in the Competition Website

3.3. Future Work

Explore hybrid architectures combining the precision of Faster R-CNN and the speed of YOLO.

Experiment with larger models (*e.g*. SAM) and evaluate their performance in more complex scenarios.

3.4. About the Code

A complete codebase and configuration files are provided to ensure result reproducibility.

The scripts have been optimized to accommodate various hardware environments. When using the code, the user needs to change its relative path.

The purposes of the code files in the "Codes" folder are as follows:

YOLO/eval_yolo.py generates json file according to the YOLO model.

YOLO/transform.py turns the origin data into the right format for YOLO model.

YOLO/train.py is the YOLO training code.

YOLO/split.py divides the training data into training set and validation set randomly.

YOLO/coco.yaml is the configuration file for YOLO.

Faster R-CNN/Faster-RCNN-eval.py generates json file according to the Faster R-CNN model.

Faster R-CNN/Faster-RCNN-train.py is the Faster R-CNN training code.

4. Team Information

4.1. Team Members Information

郑光铭 22302010014
陈柏滔 22302010008
黄宸一 22302010011

4.2. Team Name

专业课和我作队

4.3. Contribution Distribution

郑光铭: 40%
陈柏滔: 40%
黄宸一: 20%

References

[1] 刘昊岳 Liu Haoyue, 马文伟 Ma Wenwei, 付晓 Fu Xiao, 沈程秀 Shen Chengxiu, and 王亚领 Wang Yaling. 基于改进 rgb-n 的图像操纵检测算法. *Laser & Optoelectronics Progress*, 2021. 1

[2] Devjani Mallick, Mantasha Shaikh, Anuja Gulhane, and Tabassum Maktum. Copy move and splicing image forgery detection using cnn. *ITM Web of Conferences*, 44:03052, 2022. 1

[3] Yuan Rao, Jiangqun Ni, and Huimin Zhao. Deep learning local descriptor for image splicing detection and localization. *IEEE Access*, 8:25611–25625, 2020. 1

[4] Sandhya and Abhishek Kashyap. A novel method for real-time object-based copy-move tampering localization in videos using fine-tuned yolo v8. *Forensic Science International: Digital Investigation*, 48:301663, 2024. 1

[5] Xiaoyan Wei, Yirong Wu, Fangmin Dong, Jun Zhang, and Shuifa Sun. Developing an image manipulation detection algorithm based on edge detection and faster r-cnn. *Symmetry*, 11(10), 2019. 6

[6] 魏晓燕, 左鑫兰, 但志平, 吴义熔, 董方敏, and 孙水发. 提高图像篡改检测区域选取性能的 fcr-cnn 模型. 计算机辅助设计与图形学学报, 33(4):560–568, 2021. 2