Visual Recognition using Deep Learning HW2 Report

110550093 蔡師睿

1 Introduction

The task focuses on digit recognition using the SVHN dataset[1], which comprises 30,062 training images, 3,340 validation images, and 13,068 testing images. It involves two subtasks: digit classification with bounding box detection, and multi-digit number prediction. The challenge imposes two constraints, including the use of only the Faster R-CNN[2] model, and no reliance on external data. To enhance performance, I implemented several modifications and training strategies. The implementation can be found at *https://github.com/Ray-1026/Visual-Recognition/tree/main/HW2*.

2 Method

2.1 Faster R-CNN[2]

Faster R-CNN is a two-stage object detection framework that efficiently detects objects in images by combining region proposal and classification into a unified architecture. As shown in the Fig. 1, it consists of a backbone, region proposal network (RPN), and a head. In the first stage, a RPN scans the convolutional feature map to generate potential object regions (region proposals). In the second stage, these proposals are refined and classified by a Fast R-CNN head, which performs bounding box regression and object classification. The entire model is end-to-end trainable and uses a shared backbone for feature extraction, making it both accurate and relatively fast compared to earlier detectors like R-CNN[3] and Fast R-CNN[4].

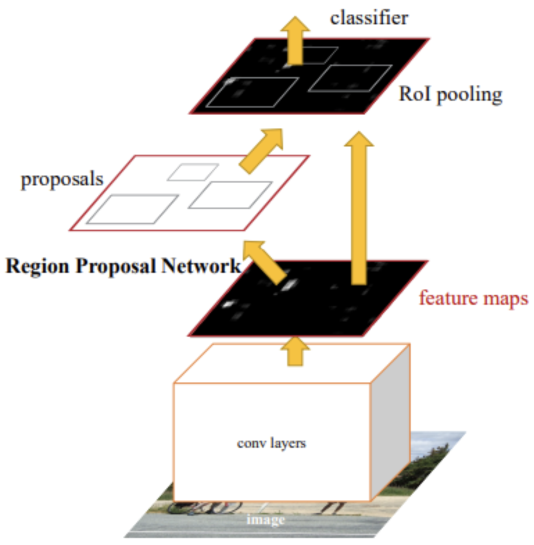


Figure 1: .

Backbone. In Faster R-CNN, the backbone functions as the feature extractor, processing raw input images to generate feature maps that emphasize key visual information. As the quality of these representations directly impacts both object localization and classification, the backbone plays a crucial role in overall detection performance. Common choices for the backbone include ResNet-50[5] and MobileNet[6], which offer a balance between accuracy and computational efficiency.

Region Proposal Network (RPN). The RPN is a fully convolutional subnetwork that generates candidate object regions directly from the feature map produced by the backbone. It slides a small window over the feature map and, at each position, predicts objectness score and bounding box coordinates. These proposals are then passed to the second stage of Faster R-CNN for classification and further bounding box refinement.

Head. In Faster R-CNN, the head refers to the second stage of the network that performs object classification and bounding box refinement on the proposals generated by the RPN. Each region proposal is first aligned with the feature map using RoI Align. This feature is then passed through a series of fully connected layers to produce class scored and bounding box regression.

2.2 Modification

To efficiently build the Faster R-CNN model, I leveraged the torchvision library[7], which provides both the architecture implementation and pretrained weights.

Min max size. Modifying the min_size and max_size parameters in Faster R-CNN during fine-tuning affects how input images are resized before being passed to the network. Here are som benefits of modifying them during fine-tuning. First, it can match dataset characteristics, and prevent overfitting on small datasets. Most of all, it can reduce GPU memory usage as well.

Data augmentation. During the experiments, I observed that fine-tuning Faster R-CNN tends to overfit after just a few epochs. To mitigate this, I applied data augmentation techniques during training. As shown in Algorithm 1, I found that applying only Color Jittering yields the best results for this challenge.

Algorithm 1 Data Augmentation

```
$train\_tfm = T.Compose([$
    $T.ColorJitter(brightness = 0.2, contrast = 0.2, saturation = 0.2, hue = 0.1),$
    $T.ToTensor(),$
])
```

Box score threshold. Modifying the box score threshold during inference in Faster R-CNN directly affects which predicted bounding boxes are kept and which are discarded based on their confidence scores. I initially experimented with a threshold of 0.5 but found that both accuracy and mAP improved when increasing it to 0.7.

2.3 Hyperparameters

- random seed: 42
- epochs: 10
- batch size: 8
- AdamW:
  - learning rate: $1e{-4}$
  - weight_decay: $1e{-2}$
- Cosine Annealing Schedule:
  - T_max: 10
  - eta_min: $1e - 6$
- box score threshold: 0.7
- min_size: 400
- max_size: 1000

3 Results

3.1 Training Curve

Although the training loss decreases consistently throughout the training process, the validation mAP begins to worsen after five epochs. This may be due to the limited size of the dataset, which makes the model prone to overfitting. As a result, I selected the model checkpoint with the highest validation mAP as the final model for evaluation, which is at epoch five here.

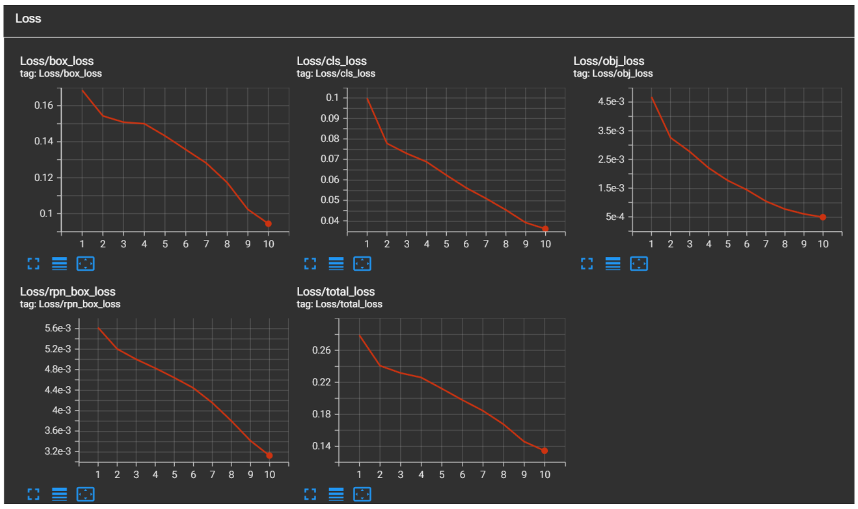


Figure 2: Training losses

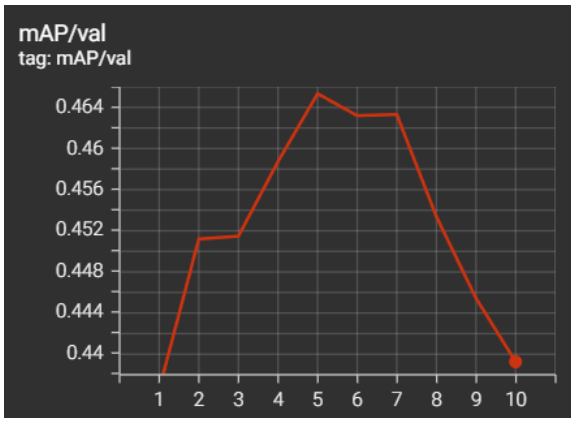
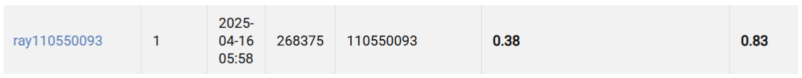


Figure 3: Validation mAP

3.2 Public Score



ray110550093	1	2025-04-16 05:58	268375	110550093	**0.38**	**0.83**

Figure 4: Public score on Codabench

4 Additional Experiments - Ablation Studies

4.1 backbone

I replaced the backbone with a more powerful pretrained ResNet-based model; however, the results suggest that ResNet-50 is better suited for this task.

Backbone	mAP	Accuracy
ResNeSt-50 [8]	0.37	0.81
ResNet-50	**0.38**	**0.83**

4.2 Data augmentation

Data augmentation	mAP	Accuracy
None	0.37	0.81
Horizontal flip	0.37	0.82
Color jitter	**0.38**	**0.83**

4.3 Training strategies

Data augmentation	Adapt minmax size	mAP	Accuracy
X	X	0.36	0.79
V	X	0.37	0.81
X	V	**0.38**	0.82
V	V	**0.38**	**0.83**

Blue curve: vanilla; **green curve**: only data augmentation; **orange curve**: only adapt minmax size; **red curve**: data augmentation+adapt minmax size.

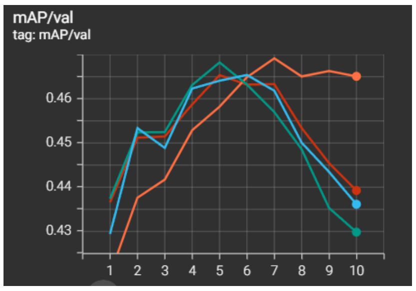


Figure 5: Validation mAPs

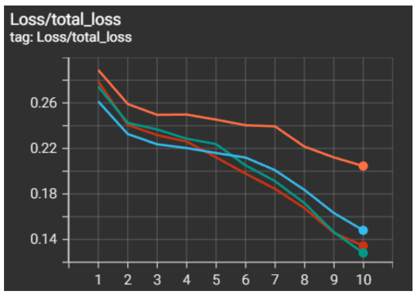


Figure 6: Training losses

References

[1] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[4] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] A. G. Howard, M. Zhu, B. Chen, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[7] T. maintainers and contributors, *Torchvision: Pytorch's computer vision library*, `https://github.com/pytorch/vision`, 2016.

[8] H. Zhang, C. Wu, Z. Zhang, *et al.*, "Resnest: Split-attention networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 2736–2746.