Visual Recognition Using Deep Learning HW2

110550085 房天越

GitHub Repo Link

https://github.com/TianYueh/NYCU-Visual-Recognition-Using-Deep-Learning-2025-HW2/

Introduction

The goal of this homework is to detect and recognize handwritten digits in images using a two-stage object detector. We need to apply Faster R-CNN framework with a ResNet-50 backbone and Feature Pyramid Network (FPN) to generate high-quality region proposals and classify each digit instance. Data augmentation is applied to improve robustness to variation. The core idea is to combine multi-scale feature extraction (via FPN) with anchor-based region proposals, then refine and classify each candidate region in a dedicated head, achieving both precise localization and accurate digit classification.

Method

1. Describe how you preprocess the data

   All images are loaded in RGB format using PIL.

   Data Augmentation for training set is applied with:

   ColorJitter(brightness=0.3, contrast=0.3, saturation=0.2),

   RandomAffine(degrees=2, translate=(0.02, 0.02)),

   ToTensor(),

   Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),

   RandomErasing(p=0.5).

   COCO annotations are transformed from [x, y, w, h] to [x_min, y_min, x_max, y_max]. Category IDs and other fields (area, iscrowd) packaged into a dict per image.

2. What is your model architecture

   I. Backbone

      The backbone is ResNet50 with FPN v2, which is pretrained on ImageNet, truncated after conv5_x, and outputs a pyramid of feature maps at stride of {4, 8, 16, 32}.

   II. Neck

The neck is AnchorGenerator and a RPN Head, with anchor sizes (8, 16, 32) pixels, and aspect ratios (0.5, 1.0, 2.0).

The RPN Head shared 3×3 conv over each FPN level. Two sibling 1×1 convs for objectness score and bounding-box regression.

III. Head

The head is Faster R-CNN Predictor, it pools each proposal into a fixed 7×7 feature map. This predictor is with 2 FC layers, each size of 1024, and a output classification layer with softmax over classes 0~9 + background. The regression has 4k outputs for bounding-box refinement.

3. Hyperparameters

Epochs: 15

Batch size: 8

Number of workers: 4

Optimizer: SGD with lr=0.005, momentum=0.9, weight_decay=0.0005

LR Scheduler: CosineAnnealingLR with T_max=10

Mixed Precision: Enabled via torch.cuda.amp

Backbone Freezing: Optional flag to freeze ResNet weights for faster convergence

Post preprocessing for Task 2 includes sorting by x-axis, filtering predictions with a confidence > 0.6, and a concatenation for digits to make numbers.

Also, IOU detection and distance detection are used to avoid repetitive detections.

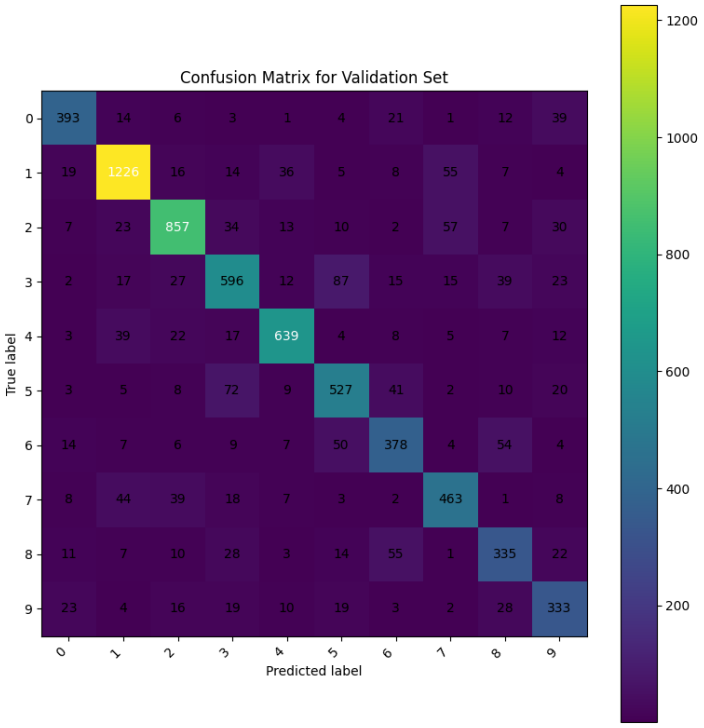
Results

1. Your Findings

I. Confusion Matrix

This is the confusion matrix of the model with score threshold = 0.6.

We can see that on the diagonal, the model classifies the numbers successfully. However, we can also see that some numbers pairs are often misclassified, such as 3/5, 1/7, and 2/7. I think this is because they have similar shape, and is sometimes written or printed alike.

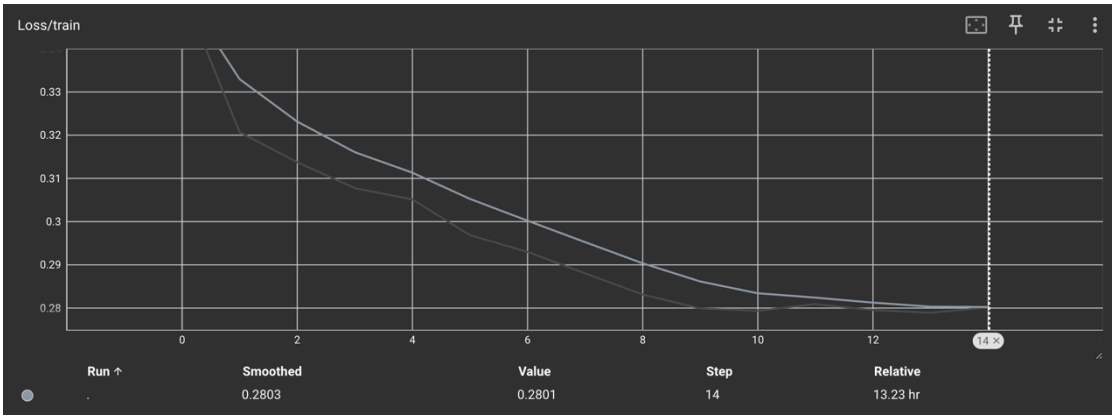


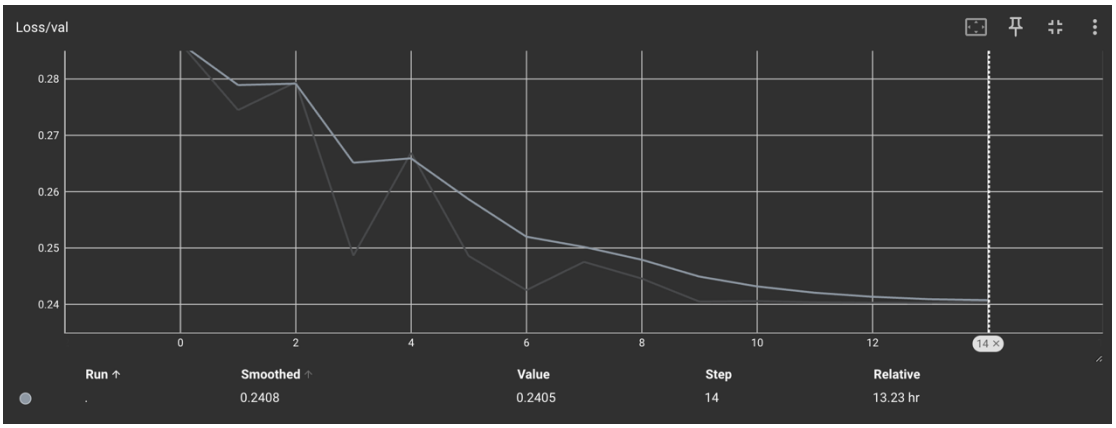
2. Model Performance

I. Training/Validation Loss

Here are the curves of training set and validation set (the darker line are the real values), we can see that the training loss gets down quickly at first and slows down at epoch 9, the loss almost does not change there.

For the validation loss, we can see a vibration at first, and it also converges to 0.24 at about epoch 9. I think this is because our predicting classes are significantly different between classes compared to other tasks, so the model recognizes the patterns soon and the loss does not change much.





Additional Experiment

1. Different Score Thresholds

   I tried to use different score thresholds, they are 0.5, 0.6, and 0.7.

   The overall performance is 0.6>0.7>>0.5.

   This is because there is a serious repetition for threshold = 0.5.

   However, if IOU detection (If IOU is larger than a specific value, discard the number with less confidence score) is enabled, we have the overall performance 0.6>0.5>0.7.

   This is possibly because that for threshold = 0.7, it ignores too many numbers, a lot of results are output as -1.

2. Different IOU detection for the second stage of evaluation.

   The area of IOU detection threshold is set as 0.3 in my final submission. I tried area = 0, or 0.3, 0.5. For threshold = 0.5, a serious repetition can be observed. However, if it is set as 0, we can see some situations such as in test_2.png, the ground truth is 112, one of the 1's is ignored.

   The overall performance is 0.3>>0>0.5.

3. With/Without Data Augmentation

   I tried not to use data augmentation, and the validation loss is about 0.29 at epoch = 4. If data augmentation is used, the validation loss becomes 0.25 at epoch = 4.

References

1. Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In Advances in Neural Information Processing Systems.

2. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). *Feature Pyramid Networks for Object Detection*. In IEEE Conference on Computer Vision and Pattern Recognition.
3. PyTorch Team. (2024). *Torchvision Faster R-CNN Implementation*. https://github.com/pytorch/vision
4. Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal Loss for Dense Object Detection*. In IEEE International Conference on Computer Vision.