Homework 2:
Digit Recognition using Faster R-CNN

Marie Picquet
Student ID: 313551818
National Yang Ming Chiao Tung University
Selected Topics in Visual Recognition
using Deep Learning (535521)

April 16, 2025

1 Introduction

The objective of this assignment is to develop a digit recognition system based on object detection and sequence classification. The task is divided into two stages: detection of individual digits within an image, followed by recognition of the full digit sequence.

The model architecture is based on Faster R-CNN [1] with a MobileNetV3 Large backbone and Feature Pyramid Network (FPN), pretrained on the COCO dataset [3]. This setup provides a balance between detection accuracy and computational efficiency. MobileNetV3 combines resource-efficient depthwise separable convolutions with architecture search and lightweight attention modules to improve performance on mobile and embedded devices [2].

Model training was conducted on Google Colab using GPU acceleration. Learning rate scheduling and automatic mixed precision (AMP) were applied to improve training stability and efficiency.

The complete code for this assignment is available at the following GitHub repository: `https://github.com/mariep00/535521-HW2.git`

2 Method

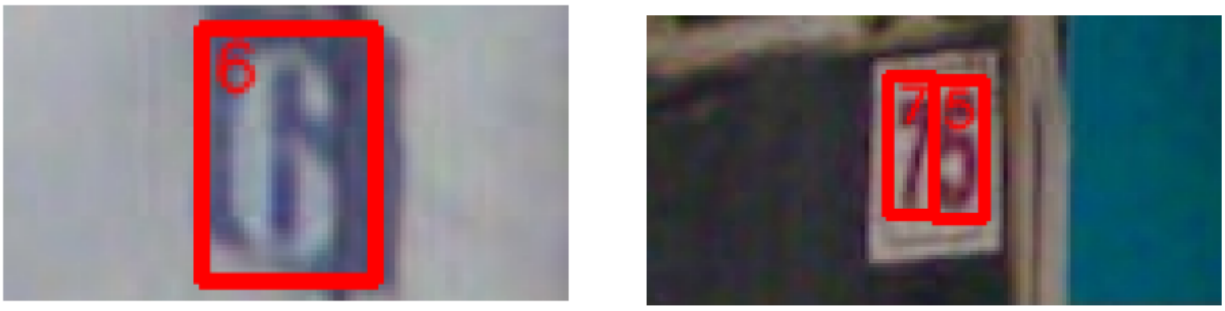
Dataset

The dataset consists of RGB images depicting digit sequences. It is divided into three subsets: 30,062 images for training, 3,340 for validation, and 13,068 for testing. In the training and validation sets, each digit is annotated with a bounding box and a class label ranging from 0 to 9. All annotations are provided in COCO format [3].

Image type	RGB
Digit classes	10 (digits 0–9)
Training set	30,062 images
Validation set	3,340 images
Test set	13,068 images

Table 1: Dataset summary

Figure 1a and Figure 1b illustrate examples of correctly annotated digits, where the bounding boxes and class labels align with the visible content of the image. However, it was observed during manual inspection that certain digits are occasionally missing from the annotation files, as demonstrated in Figure 2. This incompleteness must be taken into account when evaluating model predictions.



(a) Annotated digit "6" with ground truth bounding box.

(b) Annotated digits "7" and "5" with ground truth bounding boxes.

Figure 1: Example images from the dataset with COCO-style annotations visualized using the provided JSON files.

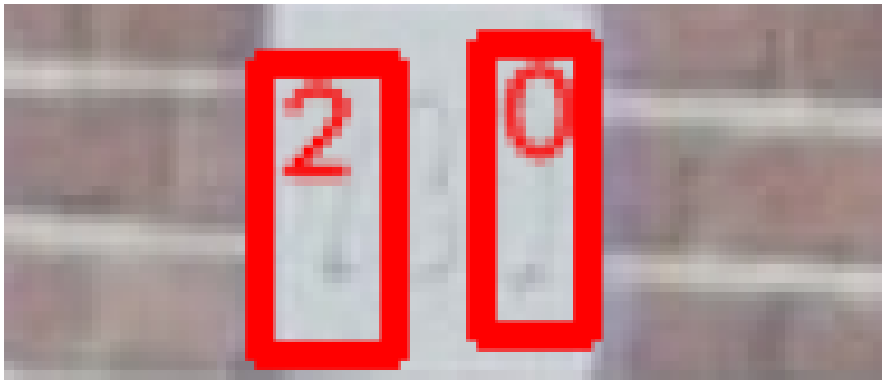


Figure 2: Example images from the dataset with COCO-style annotations visualized. Annotated digits "2" and "0" with ground truth bounding boxes and visibly missing digit.

Data Preprocessing

Images were resized such that the shorter side is 512 pixels while preserving the aspect ratio, for better results. Bounding boxes were scaled accordingly. All images were converted to tensors, with pixel values scaled to the range $[0, 1]$.

Data Augmentation

No data augmentation was applied during training. This decision was made to reduce computational overhead and due to the observation that most digits in the dataset appeared upright and mostly

visible.

Task 1: Digit Detection

Model Architecture

The model follows the standard Faster R-CNN architecture and consists of three main components: a backbone for feature extraction, a Region Proposal Network (RPN) as the neck, and a detection head for classification and bounding box regression. The model was implemented using the `torchvision.models.detection` module [4].

- Backbone:

The backbone network is MobileNetV3 Large with a Feature Pyramid Network (FPN). MobileNetV3 is a lightweight convolutional neural network designed for efficient inference on mobile devices [2]. It combines depthwise separable convolutions with neural architecture search techniques and attention modules to improve the balance between performance and computational efficiency.

The FPN enhances the backbone by producing multi-scale feature maps, improving detection performance on objects of varying sizes. The backbone was initialized with pretrained weights from the COCO dataset and used without normalization adjustments to the input images.

This choice of MobileNetV3 over deeper architectures like ResNet-50 was motivated by practical considerations. While ResNet-50 may give slightly higher accuracy on large-scale datasets, it requires significantly more computational resources. In contrast, MobileNetV3 offers much faster inference and lower memory usage, which is critical when training on platforms like Google Colab. For the task of digit detection, where the input patterns are relatively simple, the trade-off in accuracy is likely minimal. The addition of FPN helps further mitigate the performance gap by improving multi-scale feature representation. Overall, the selected architecture achieves a strong balance between speed, efficiency, and accuracy.

- Neck: Region Proposal Network (RPN):

The RPN takes feature maps from the backbone and generates a set of object proposals (regions likely to contain digits). It uses anchor boxes at multiple scales and aspect ratios to predict objectness scores and bounding box coordinates. The proposals are filtered using non-maximum suppression (NMS) and top-k selection to retain the most likely candidate regions.

- Head:

The detection head receives the proposals from the RPN and performs classification and bounding box regression for each region of interest (ROI). The final classifier was modified to output 11 classes (10 digits plus 1 background class). The head consists of two fully connected layers followed by separate branches for class scores and box refinements. During training, the model learns to assign accurate class labels and regress refined box coordinates for each detected digit.

Training Configuration - Hyperparameters

The model was trained for one epoch using the SGD optimizer with momentum (0.9), an initial learning rate of 0.005, and weight decay of 0.0005. A step learning rate scheduler with decay factor 0.1 and step size of 3 was used. Training was performed on Google Colab with GPU acceleration.

To improve efficiency, Automatic Mixed Precision (AMP) was used via `torch.amp` and `GradScaler` [5]. Checkpoints were saved and the best model (based on validation loss) was used for inference.

Implementation Details

Custom `Dataset` classes were implemented for both training and inference. Annotations were parsed from COCO-style JSON files. Bounding boxes were converted to `[x1, y1, x2, y2]` format. A special `TestDataset` class preserved original image sizes for proper coordinate scaling during post-processing.

During inference, the trained model was evaluated on the test set using batch size 4. Results were collected and stored in COCO format for Task 1 and as full digit predictions for Task 2.

Task 2: Full Number Recognition

The second task aimed to predict the entire digit sequence visible in each test image. The model performed this by using the individual digit detections from Task 1 to reconstruct full numbers. To generate full digit predictions for each image (Task 2), predictions from Task 1 were grouped by image. Each predicted digit was assigned a horizontal center position based on its bounding box, and all digits were then sorted from left to right. Finally, the predicted digits were concatenated to form the full number. If no digits were detected for an image, ...a fallback value of -1 was used, following the evaluation protocol in COCO-style object detection [3].

3 Results

Task 1: Digit Detection

The output for Task 1 consists of a JSON file `pred.json` in COCO format, containing all predicted digits in the test images.

The model outputs were filtered with a confidence threshold of 0.5 and non-maximum suppression (NMS) was applied to remove overlapping predictions. An example prediction is illustrated in Figure 3. The predictions correspond to digits "0" and "1", with overlapping bounding boxes, demonstrating the model's ability to localize and classify adjacent digits with varying confidence.

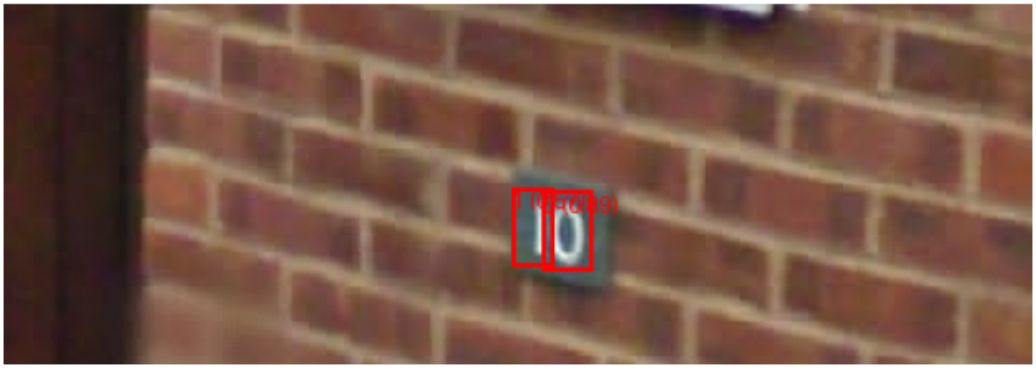


Figure 3: Sample detection results on the test set. Two bounding boxes were detected:
- **BBox 1:** [250.99, 86.74, 22.23, 37.19], Category ID: 1 (Label: 0), Confidence: 0.8863
- **BBox 2:** [236.22, 85.01, 18.86, 36.59], Category ID: 2 (Label: 1), Confidence: 0.5681.

Detection accuracy was evaluated using mean Average Precision (mAP). The final model achieved a competition score of 38% mAP on the test set.

Task 2: Full Number Recognition

This task involved reconstructing full digit sequences from the individual detections obtained in Task 1. The approach proved reliable across a wide range of inputs, effectively handling multi-digit compositions and varying spatial layouts. By sorting detected digits from left to right based on their horizontal positions, the model was generally able to assemble sequences accurately.

While some failure cases occurred due to missing detections or incorrect ordering, the system performed well on the majority of examples. Figure 4 illustrates a successful prediction, where digits “1” and “4” were detected in close proximity and correctly ordered to form the number “14”.

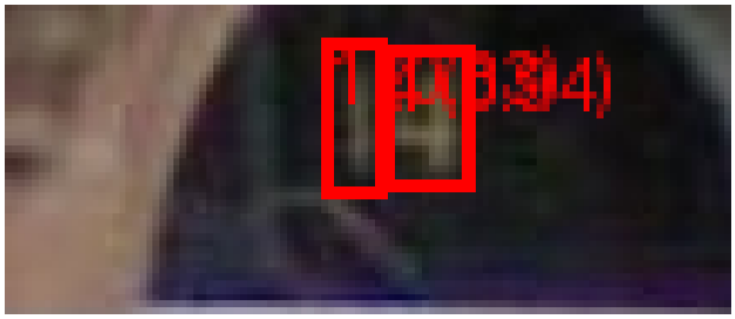


Figure 4: Task 2 example: Digit detection and reconstruction into the number ”14”. Two digits were detected in close proximity:
- **BBox 1:** [56.07, 6.41, 13.34, 20.82], Category ID: 5 (Label: 4), Confidence: 0.9368
- **BBox 2:** [47.65, 5.37, 8.85, 23.09], Category ID: 2 (Label: 1), Confidence: 0.8252

Overall, the model demonstrated a strong performance on Task 2, particularly for clearly separated and well-aligned digits. However, its accuracy was sometimes impacted by challenges such as occlusion, annotation inconsistencies, or degraded image quality (e.g., motion blur or strong lighting contrast). Despite these limitations, the final system achieved a competitive score of 81% accuracy on the test set.

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In Advances in Neural Information Processing Systems (NeurIPS), 2015.

[2] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. *Searching for MobileNetV3*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 1314–1324, 2019.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. *Microsoft COCO: Common Objects in Context*. In European Conference on Computer Vision (ECCV), pages 740–755, 2014.

[4] PyTorch Vision Team. *TorchVision: Object Detection Finetuning Tutorial (Faster R-CNN)*. Available at: `https://github.com/pytorch/vision`

[5] PyTorch Team. *Automatic Mixed Precision (AMP) for Training Deep Neural Networks*. Available at: `https://pytorch.org/docs/stable/amp.html`