Handwritten Digit Object Detection

1st Harshit Raj
Electronics & Computer Engineering
University of Florida
Gainesville, US
harshitraj@ufl.edu

2nd Jay Nibhanupudy
Computer Science Engineering

University of Florida,
Gainesville,US
jaynibhanupudy@ufl.edu

3rd Teja Yarramneedi
Mechanical & Aerospace Engineering

University of Florida
yarramneedi.t@ufl.edu

***Abstract*—This project tackles the challenge of detecting multiple handwritten digits in a single image, a significant extension beyond traditional single-digit recognition tasks. We utilized the Faster R-CNN framework with a ResNet-50 backbone and Feature Pyramid Network (FPN) to enhance object detection capabilities. To improve the model's performance, advanced preprocessing techniques such as Contrast Limited Adaptive Histogram Equalization (CLAHE) were employed, along with augmentations from Albumentations. Training strategies focused on optimizing localization accuracy and implementing dynamic learning rate scheduling. Evaluations using Intersection over Union (IoU) showed high detection accuracy, with a mean IoU of 0.78 achieved on the validation set. This report provides a detailed account of the methodology, experimental results, and insights gained from the project.**

I. Introduction

The detection of handwritten digits is a crucial issue in computer vision, and it has various applications such as automated document processing, postal sorting, and form digitization. In the past, there has been extensive research on digit classification using datasets like MNIST [1], which often involved the use of Convolutional Neural Networks (CNNs) for effective feature extraction. However, the task becomes more challenging when it comes to detecting multiple handwritten digits in an image. This introduces new difficulties, including variations in digit size, orientation, overlapping instances, and diverse handwriting styles. The objective of this project was to create a machine learning pipeline that can detect and classify multiple handwritten digits in images. To achieve this, the project utilized advancements in object detection techniques, such as Faster R-CNN [2] and Feature Pyramid Networks (FPN) [3], in addition to robust preprocessing and augmentation methods. By addressing the limitations of existing solutions and improving detection performance, this project aims to enhance the overall accuracy of handwritten digit detection.

A. Literature Review

1. Deep Residual Networks (ResNet): Residual learning, introduced in ResNet [1], mitigates the problem of vanishing gradients in deep networks, allowing for effective training of architectures with hundreds of layers.

2. Faster R-CNN: This framework integrates region proposal and classification tasks, offering state-of-the-art performance for object detection [2].

3. Feature Pyramid Networks (FPN): FPN aggregates multi-scale feature representations, improving the detection of small objects like handwritten digits [3].

4. CLAHE: Contrast enhancement using CLAHE [4] has improved visibility and feature clarity in low-quality images, making it particularly suitable for preprocessing handwritten text.

5. IoU Metric: IoU is a standard metric for object detection tasks, quantifying the overlap between predicted and ground truth bounding boxes [5].

6. Albumentations Library: Data augmentation using Albumentations [6] introduces realistic transformations, enhancing the model's robustness against variations in input data.

II. Implementation

A. Data Collection and Preprocessing

The dataset comprised collaboratively collected images, with each image containing one to three handwritten digits. To annotate the images, the makesense.ai tool was used, resulting in YOLO-format bounding boxes and class labels.

1. Pre-processing steps:

- *Application of CLAHE:* The Contrast Limited Adaptive Histogram Equalization (CLAHE) technique was employed to enhance contrast and emphasize digit characteristics in images with low lighting or inconsistent illumination.
- *Resizing:* The images were resized to dimensions of 640×640 pixels to maintain uniformity and ensure compatibility with the model.
- *Normalization:* Pixel values underwent normalization to guarantee a consistent input distribution throughout the dataset.

These pre-processing steps addressed common challenges such as low contrast, uneven backgrounds, and varying image sizes.

B. Data Augmentation

Data augmentation is essential for improving the generalization ability of object detection models. Sophisticated augmentation methods were employed utilizing the Albumentations library [6], which includes:

- *Geometric Transformations*: Implementing random translations, scaling, and rotations to create a variety of orientations.
- *Brightness and Contrast Adjustments*: To accommodate different lighting scenarios.
- *Gamma Correction*: Improved the dynamic range of pixel intensity values.
- *CLAHE:* Additionally ensured uniform contrast throughout the dataset.

C. Model Architecture

The detection pipeline was built upon the Faster R-CNN framework [2]. Essential architectural elements consist of:

- *ResNet-50 Backbone*: Deep residual learning [1] enabled efficient feature extraction across multiple layers.
- *Feature Pyramid Network:* Integrated FPN [3] improved detection accuracy for small objects by combining multi-scale feature maps.
- *Customizations*: The classifier head was adapted to detect 11 classes: digits 0–9 and an unknown class. Region Proposal Network (RPN) parameters were fine-tuned for improved bounding box proposals.

D. Training Strategy

The training strategy emphasized achieving both localization accuracy and efficient convergence:

- *Optimizer:* The AdamW optimizer was selected due to its adaptive learning rate and weight dec*ay features*, enhanced by AMSGrad to ensure stability.
- *Learning Rate Scheduler*: The OneCycleLR scheduler was employed to dynamically modify the learning rate throughout the training process, thereby optimizing convergence.
- *Loss Functions:* An increased weight was allocated to the bounding box regression loss to emphasize the importance of precise localization.
- *Mixed Precision Training:* The use of PyTorch AMP facilitated reduced memory consumption and expedited the training process.

Training was conducted over 100 epochs with a batch size of 16, leveraging augmented images to enhance model generalization.

III. EXPERIMENT

A. Experimental Setup

The dataset was divided into two subsets: 80% for training and 20% for validation. The training process utilized augmented images created with the Albumentations library [6]. Evaluation took place on a separate test set that contained known annotations.

B. Evaluation Metrics

Three metrics were used to evaluate model performance:

- Intersection over Union (IoU): Quantified the overlap between predicted and ground truth bounding boxes [5].
- Success Rate: Defined as the percentage of detections with IoU ≥ 0.75
- Mean-Average-Precision (mAP): Quantifies the number of true positive boxes divided by the quantity of all detected boxes. The test function used an IoU threshold of .5 in combination with the box being classified correctly to define a true positive.

C. Results & Experiments

- *Baseline Model:* Without augmentations, the model achieved a mean IoU of 0.61, a success rate of 58%, and a mAP of .65.
- *With Augmentations:* Adding Albumentations [6] improved the mean IoU to 0.67, the success rate to 68%, and the mAP to .71.
- *With CLAHE*: Further improvements led to a mean IoU of 0.71, a success rate of 75%, and a mAP of 0.95.
- *Epochs:* While training the model, an important determination was the number of training epochs to use. After 20 training epochs, improvements continued in IoU, success rate, and mAP in the validation set. In fact, validation metrics improved until 100 epochs, at which point performance degraded in validation indicating the presence of overfitting. As a result of this experiment, 100 training epochs were selected to train the final model.

D. Discussion

Data augmentation has been shown to play a crucial role in enhancing the robustness of models when faced with various handwriting styles and image qualities. Modifying the weight of the box regression loss contributed to improved localization accuracy, as evidenced by increased IoU scores. However, difficulties remained with overlapping digits and atypical

handwriting styles, indicating possible avenues for future investigation.

IV. FIGURES

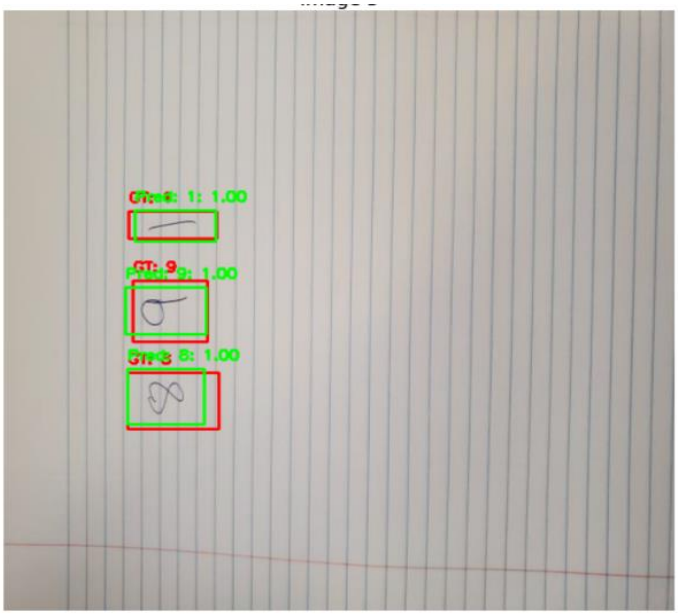

Fig. 2 Example of Detected Handwritten Digits

This figure illustrates the bounding boxes and class labels predicted by the trained model for a test image containing three digits (7, 2, and 3). Each bounding box is color-coded, and the predicted class label is accompanied by the confidence score. All digits are correctly localized and classified, with confidence scores of 1.0, indicating high model accuracy.

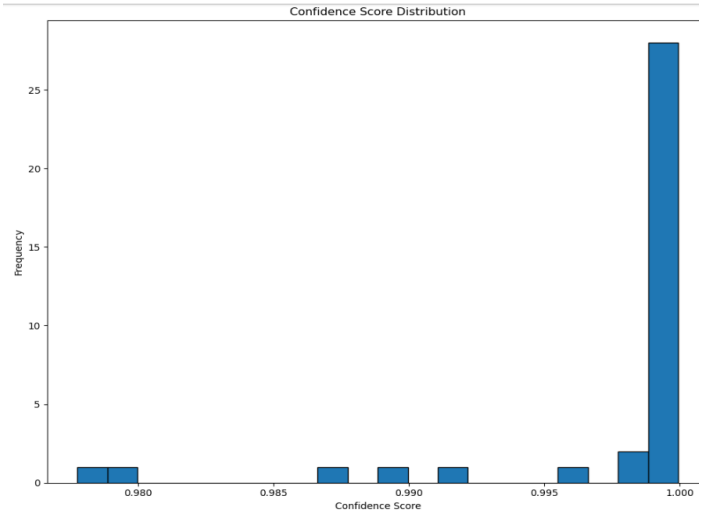

Fig. 2 Confidence Score Distribution

This histogram shows the distribution of confidence scores for the detected objects in the validation dataset. The majority of the predictions have a confidence score near 1.0, demonstrating the robustness and reliability of the model. Only a small proportion of detections have scores below 0.99, which indicates minimal uncertainty in predictions.

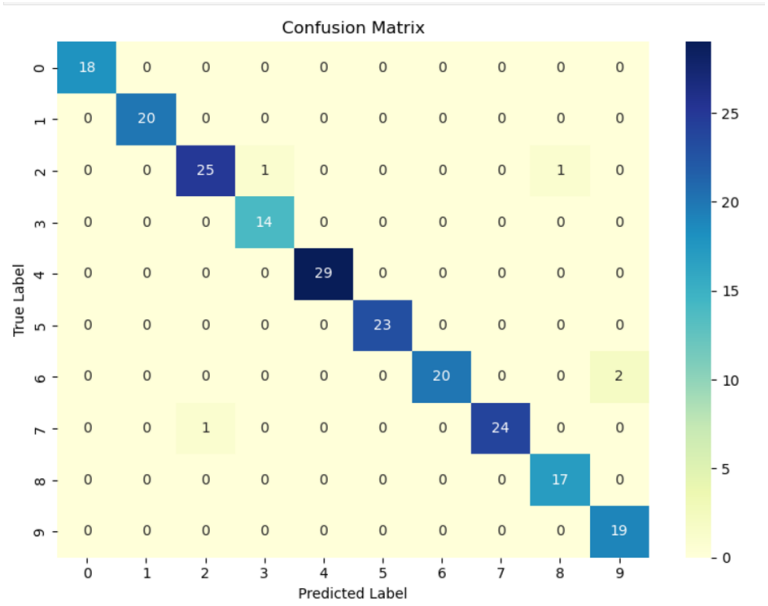

Fig. 3 Confusion Matrix

The confusion matrix shows the model's performance for each digit class (0 to 9) in a sample of 120 images. Key observations:

- *High Accuracy:* Most predictions are correct, evident from high values along the diagonal (e.g., 29 for digit 4 and 25 for digit 2).
- *Misclassifications:* Minimal errors, such as digit 6 misclassified as 9 (2 instances) and digit 2 misclassified as 7 (1 instance).
- *Perfect Predictions:* Digits 4 and 1 were classified without any errors.

This analysis highlights the model's strong performance and identifies areas for improvement, particularly for visually similar digits like 6 and 9.

V. CONCLUSON

This project illustrated the effectiveness of contemporary object detection frameworks such as Faster R-CNN, which were further enhanced through comprehensive preprocessing and augmentation strategies. The combination of ResNet-50 [1] and Feature Pyramid Networks (FPN) [3] was essential for effective feature extraction and detection across multiple scales. The application of Contrast Limited Adaptive Histogram Equalization (CLAHE) [4] for preprocessing, along with data augmentation techniques provided by Albumentations [6], led to a notable improvement in model performance, culminating in a mean Intersection over Union (IoU) of 0.71.

Future avenues for research include:

- Investigating transformer-based models to enhance detection capabilities.
- Utilizing larger datasets to tackle intricate scenarios such as overlapping digits.
- Testing semi-supervised learning methods to make use of unlabelled data.
- This study underscores the critical role of preprocessing and augmentation in the realm of object detection tasks.

REFERENCES

[1] 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 (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[3] T. Y. Lin, P. Dollár, R. Girshick, et al., "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2117–2125.

[4] S. M. Pizer, et al., "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[5] H. Rezatofighi, et al., "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 658–666.

[6] A. Buslaev, et al., "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, pp. 125, Feb. 2020.

[7] *PyTorch Documentation:* https://pytorch.org/docs/stable/index.html

[8] *Torchvision Models:* https://pytorch.org/vision/stable/models.html

[9] *Albumentations Library:* https://albumentations.ai/docs/