**ECE 364 Project Report**

**Instructor: Prof. Kani**

**May 08, 2025**

Junyi Cheng [junyi10@illinois.edu](mailto:junyi10@illinois.edu)

Chaohua Yao [chaohua4@illinois.edu](mailto:chaohua4@illinois.edu)

Shengyu Xu [sx27@illinois.edu](mailto:sx27@illinois.edu)

1. **Introduction**

The project focuses on detecting handwritten Chinese characters embedded within composite images containing both digits and characters—a task made challenging by visual clutter, overlapping instances, and small object sizes. The dataset, CCD-ND, combines samples from MNIST and Chinese-MNIST, requiring the model to distinguish and localize Chinese characters while ignoring digit distractors.

To address this, we adopted YOLOv8s, a lightweight yet powerful object detection framework, and developed a pipeline that includes data augmentation, test-time enhancements, and advanced post-processing techniques. Our goal was to build an efficient detection system that satisfies the project’s parameter constraints while achieving robust performance under real-world conditions.

1. **Approach, Model Selection & Other details**

We chose YOLOv8s as our base model for several reasons. First, compared to other popular detection frameworks such as RT-DETR or MMDetection, YOLOv8 provides an exceptionally user-friendly setup and training pipeline via the Ultralytics API, which greatly simplifies experimentation and reproducibility. Additionally, the YOLOv8s variant has a relatively small number of parameters, making it well-suited for our project constraint of keeping the model under 30 million parameters.

Despite its lightweight nature, YOLOv8s delivers state-of-the-art detection performance, especially for small objects, which aligns well with our task of identifying handwritten Chinese characters in cluttered and noisy backgrounds. Its balance of accuracy, efficiency, and ease of deployment made it an ideal choice for this project.

We focused on tuning key hyperparameters to balance model performance and computational limits. Image size and freeze were adjusted to manage GPU memory while preserving accuracy. Epoch count and patience were used to avoid overfitting or underfitting. A warm-up phase was included to stabilize initial training. Data augmentations such as mosaic, horizontal flip, and Mixup improved generalization and robustness against distractors. These choices were essential for achieving reliable detection under constrained conditions.

To enhance robustness and improve final detection accuracy, we employed a model ensemble strategy during inference. Instead of relying on a single trained model, we combined predictions from two different YOLO models trained with distinct settings. These models were evaluated under multiple test-time augmentations, including multi-scale resizing and horizontal flipping. All predictions were then merged using a Weighted Box Fusion (WBF) algorithm, which considers both confidence scores and box overlap to generate final bounding boxes. We observed that this ensemble-TTA-WBF pipeline consistently outperformed any single model on the Kaggle leaderboard, demonstrating its effectiveness in handling noisy and diverse image inputs.

For the final output generation, we employed Soft-NMS (Gaussian variant) instead of traditional NMS. This approach reduces the confidence of overlapping boxes rather than suppressing them outright, which is particularly beneficial for dense character layouts. Combined with multi-scale and flipped inference (TTA), we aggregated multiple predictions per image and applied Soft-NMS to select high-confidence, low-redundancy bounding boxes. Compared to standard NMS or single-pass inference, this method yielded more stable results in cases of character overlap or visual ambiguity.

1. **Training Hyperparameters**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Model 1 | Model 2 | Comment |
| imgsz | 1024 | 1280 | Input image size |
| epochs | 150 | 250 | Number of training epochs |
| batch | 8 | 8 | Batch size |
| freeze | 5 | 0 | Freeze first xx layers for fine-tuning |
| patience | 20 | 30 | Early stopping after xx rounds without improvement |
| box | 0.1 | 0.1 | Weight of the box regression loss |
| lr0 | 0.003 | 0.003 | Initial learning rate |
| lrf | 0.1 | 0.1 | Final learning rate |
| momentum | 0.937 | 0.937 | SGD momentum |
| weight\_decay | 0.001 | 0.001 | Weight decay (L2 regularization) |
| warmup\_epochs | 3.0 | 3.0 | Number of warmup epochs |
| warmup\_momentum | 0.8 | 0.8 | Warmup momentum |
| warmup\_bias\_lr | 0.1 | 0.1 | Warmup learning rate for bias layers |
| hsv\_h | 0.015 | 0.015 | Hue augmentation |
| hsv\_s | 0.5 | 0.5 | Saturation augmentation |
| hsv\_v | 0.4 | 0.4 | Value (brightness) augmentation |
| translate | 0.2 | 0.2 | Translation (shift) augmentation |
| scale | 0.3 | 0.3 | Scaling augmentation |
| fliplr | 0.4 | 0.4 | Left-right flip probability |
| mosaic | 1.0 | 0.6 | Left-right flip probability |
| mixup | 0.1 | 0.15 | Mosaic augmentation probability |

1. **Results and Discussion**

Our final model achieved strong performance on the validation set, with an mAP@0.5 of 0.987 and an mAP@0.5:0.95 of 0.945, indicating high precision across various IoU thresholds. On the Kaggle test set, our model achieved an mAP@0.5:0.95 of 0.926, ranking first on the public leaderboard at the time of submission.

The slight performance drop from local validation to the Kaggle test set may be attributed to overfitting, as the model was trained on a relatively small dataset. To further improve generalization and robustness, future work could incorporate data augmentation through generative models (e.g., GANs) to expand the training set and capture more diverse character representations and layouts.