Handwritten Digit Classfier

YOLOv8-Based Handwritten Digit Classification with Optimized Data Augmentation Strategies

Bharat Gandhi
ECE Department
University of Florida
Gainesville, USA
b.gandhi@ufl.edu

Nicholas Sily Vasconcellos
CISE Department
University of Florida
Gainesville, USA
nicholas.silyvas@ufl.edu

Yongkyoon Park
ECE Department
University of Florida
Gainesville, USA
yo.park@ufl.edu

***Abstract*—This project presents the development of a handwriting digit classifier leveraging a comparative analysis between two pre-trained models, YOLO and CRNN, using a custom dataset. Through evaluation, the YOLO model was identified as the most suitable for the dataset format, achieving superior performance. Consequently, the YOLO model was employed to construct the final classifier, which achieved an accuracy of approximately 90%. The results highlight the necessity of data preprocessing to further enhance model performance.**

***Keywords*—Image Processing, YOLO(You Only Look Once), CRNN(Convolutional Recurrent Neural Network), Data Pre-processing, Data Augmentation**

I. Introduction

Handwritten digit recognition and prediction is a fundamental application of machine learning and deep learning techniques, demonstrating their practicality and effectiveness in solving real-world problems. This project aims to implement models for handwritten digit identification and prediction to better understand the application of machine learning and deep learning algorithms. By engaging in this project, we explore how algorithms operate in practice, learn strategies to enhance model performance, and mitigate overfitting.

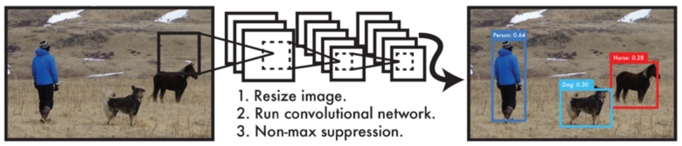


Fig. 1. The YOLO Detection System.

To achieve these goals, we studied related research and literature on YOLO (You Only Look Once) and CRNN (Convolutional Recurrent Neural Network) models. These studies provided insights into the data structures used for training these models and the principles underlying their prediction mechanisms. This prior knowledge guided the development and evaluation of our models.

The primary focus of this project is to gain a deeper understanding of how CNNs (Convolutional Neural Networks) and other deep learning models function, while also identifying effective methods for improving their prediction performance. Additionally, we emphasize learning techniques to prevent overfitting, ensuring that the models generalize well to unseen data.

The structure of this paper is as follows. Section 2 provides an overview of related research and theoretical definitions, Section 3 describes the experimentation and implementation in this project, Section 4 presents the experimental results and key findings, Section 5 is the conclusion.

Through this project, we aim to acquire hands-on experience with machine learning and deep learning models, enhancing our understanding of their operational mechanisms and strategies for improving their performance in practical applications.

II. Overview of Related Research

A. *YOLO (You Only Look Once)*

The YOLO (You Only Look Once) model, as the name suggests, performs object detection by analyzing an image only once. Introduced in 2015 by Joseph Redmon, YOLO revolutionized object detection by offering a faster and more efficient approach compared to prior models. Before YOLO, deep learning models required multiple passes over an image, significantly increasing the time needed for processing. This limitation made it challenging to use such models for real-time object detection tasks.

In contrast, the YOLO model detects objects in an image in a single pass, allowing it to perform object detection at high speeds. This advantage makes YOLO particularly effective for applications requiring real-time processing, such as video surveillance, where traditional models may fall short due to slower processing times.

When an image is input, it is divided into grid regions of equal size. For each grid cell, the model predicts the presence of an object, bounding boxes, and the confidence scores for the boxes. The higher the confidence score, the tighter the bounding boxes will be. Simultaneously, the model performs classification tasks to determine which objects are present in the respective regions. Afterward, only the bounding boxes with high confidence scores are retained, while those with low probabilities are filtered out. Finally, the remaining bounding boxes are processed using the Non-Maximum Suppression

(NMS) algorithm, which removes redundant boxes and retains only the most relevant ones.

For confidence score, the model uses the following prediction:

$$Pr(Object) * IOU_{Truth}$$

Where The value of Pr is {0, 1} depending on the existence of object (it there is no object in that cell, the Pr scores should be zero), and the IoU (Intersection over Union) is a metric used to evaluate the positional accuracy of bounding boxes. It is calculated as the ratio of the intersection area to the union area of the predicted bounding box and the ground truth bounding box. A higher IoU value indicates greater accuracy.

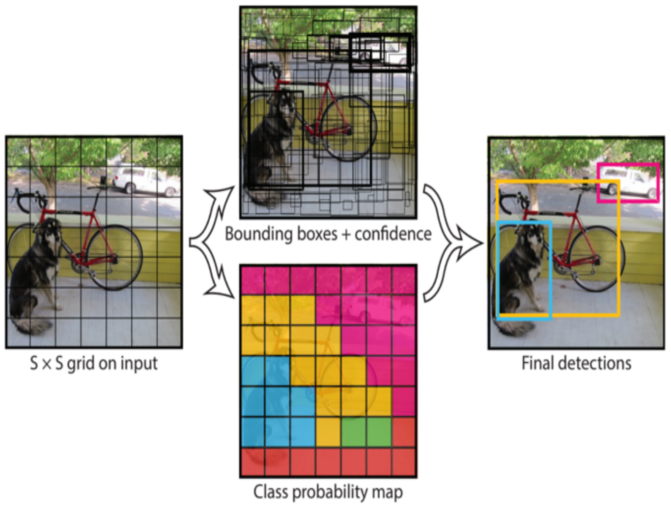


Fig. 2. Finding the objective and bounding boxes process

B. *CRNN(Covolutional Recurrent Neural Network)*

In image processing, there are cases where it is necessary to handle not only individual objects but also sequences of images, such as continuous text or musical scores. These sequence-like objects require predictions based on a series of object labels rather than single labels. While it is possible to process each element individually by dividing the sequence into smaller parts, this approach increases the computational cost due to additional preprocessing and postprocessing steps.

CRNN (Convolutional Recurrent Neural Network) is particularly effective in such cases, as it captures the relationship between images and text within a single framework. As an end-to-end model, CRNN integrates the entire process, from input image recognition to text prediction, into a single network for training and inference. This enables efficient text recognition within images without the need for cumbersome intermediate steps, streamlining the entire process.

The spatial features of an image are first extracted using a Convolutional Neural Network (CNN). These features are then transformed into a 1D sequence through Sequence Transformation, making them suitable as input for the Recurrent Neural Network (RNN). The RNN processes this input to learn the temporal and contextual relationships of the text. After passing through the RNN, time-step predictions are generated for each sequence position. These predictions, along with the label information, are utilized by the CTC Loss to recognize the characters. The CTC Loss algorithm learns the alignment between unsegmented sequences—where the positions of each class are unknown—and the RNN outputs.

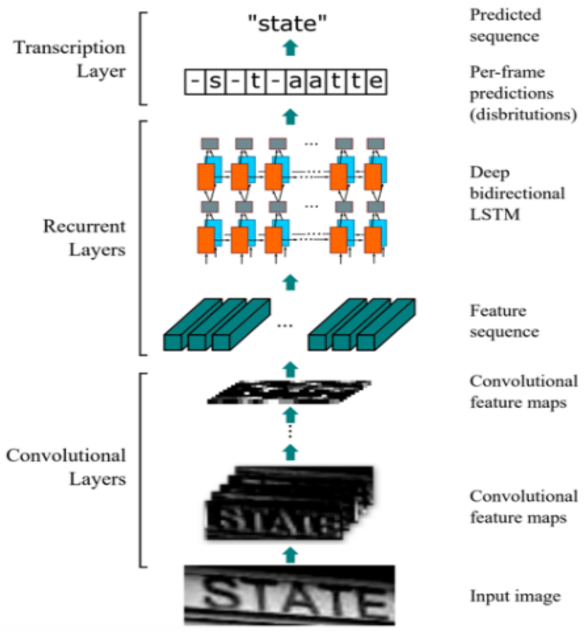


Fig. 3. The CRNN Detection System

III. Experimentation and Implementation

We determined that building a YOLO-based model would be a more effective approach than using a CRNN model, considering the characteristics of our training dataset, such as its data structure and coordinate-based data labeling. Additionally, since the data we aim to classify consists of individual characters that are independent of contextual relationships, the YOLO model demonstrated better learning capability and performance. As a result, we proceeded with the project using the YOLO-based model.

A. *Data Cleaning*

A high-level manual inspection of the data, images and labels, was carried out and mislabeled or erroneous data were removed, so as not to impair model performance. 150 such images were identified and excluded from the dataset.

B. *Data Augmentation*

To increase the amount of data available for training the model, we utilized the fact that the orientation of handwritten digits was arbitrary. For each original image, three additional copies were created by rotating the image counterclockwise in increments of 90° (90°, 180°, and 270°).

This was carried out via a custom Python script, "rotate.py", leveraging the Pillow (PIL) library [3] for image manipulation. The script also incorporated a custom formula to update the label file by calculating the new bounding box positions based on the applied rotations.

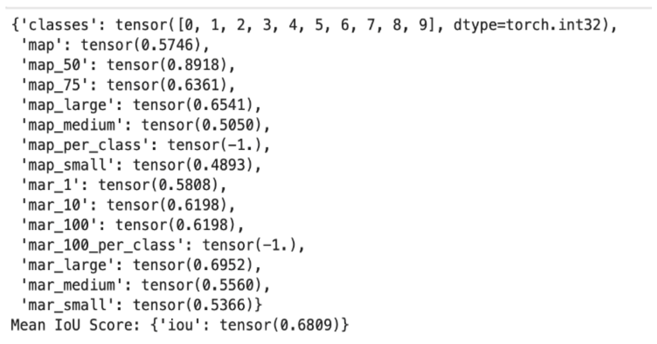
The augmented data was thoroughly inspected to ensure accuracy. Once validated, the new samples were incorporated into the training and validation sets, effectively increasing the dataset size by 300%.

C. *Implementation*

Our approach leveraged transfer learning by utilizing the YOLOv8 model pre-trained on a large-scale dataset, accessed via the *Ultralytics* library [4]. This choice was motivated by YOLOv8's demonstrated efficiency and robustness for object detection tasks. To adapt the model for our custom handwritten digit dataset, we began by studying its specific training requirements. We adhered to the directory structure guidelines outlined in the *data.yaml* configuration file, ensuring compatibility with the training pipeline. For optimization, we initially set the model's optimizer to 'auto', allowing it to determine the optimal optimization strategy dynamically. The model selected *AdamW* as the best optimizer, with a learning rate automatically tuned to 0.000714, providing a strong foundation for efficient training. Evaluation of the model's performance relied on two key accuracy metrics: Intersection Over Union (IoU) and Mean Average Precision (mAP), implemented using the *torchmetrics.detection* library [5]. For IoU calculation, ground truth annotations were prepared in the format of class label, x_{center}, y_{center}, width, height. These coordinates were then converted to x_{min}, y_{min}, x_{max}, y_{max} format, and fed into the IoU function as a list of dictionaries, where each dictionary corresponded to an image. The mAP metric additionally required the confidence scores generated by the trained model. These scores were extracted and stored in the output '.txt' files, ensuring seamless integration with the evaluation pipeline. By combining these metrics, we achieved a comprehensive assessment of the model's accuracy and robustness in detecting handwritten digits.

IV. RESULTS

The model was validated on 20% of the training data, which amounts to around 3,200 images. It was observed that the model performed well on unseen data with an average IoU score of around 0.68 and MAP50 score of around 0.89. The figure below shows the calculated metrics using the 'torchmetrics.detection' library.



```
{'classes': tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=torch.int32),
 'map': tensor(0.5746),
 'map_50': tensor(0.8918),
 'map_75': tensor(0.6361),
 'map_large': tensor(0.6541),
 'map_medium': tensor(0.5050),
 'map_per_class': tensor(-1.),
 'map_small': tensor(0.4893),
 'mar_1': tensor(0.5808),
 'mar_10': tensor(0.6198),
 'mar_100': tensor(0.6198),
 'mar_100_per_class': tensor(-1.),
 'mar_large': tensor(0.6952),
 'mar_medium': tensor(0.5560),
 'mar_small': tensor(0.5366)}
Mean IoU Score: {'iou': tensor(0.6809)}
```

Fig 4: IoU and MAP metrics

The Precision of the model increased steadily until the 15th epoch, after which it leveled off at approximately 0.82. The figure below illustrates the precision confidence curve throughout the training process.

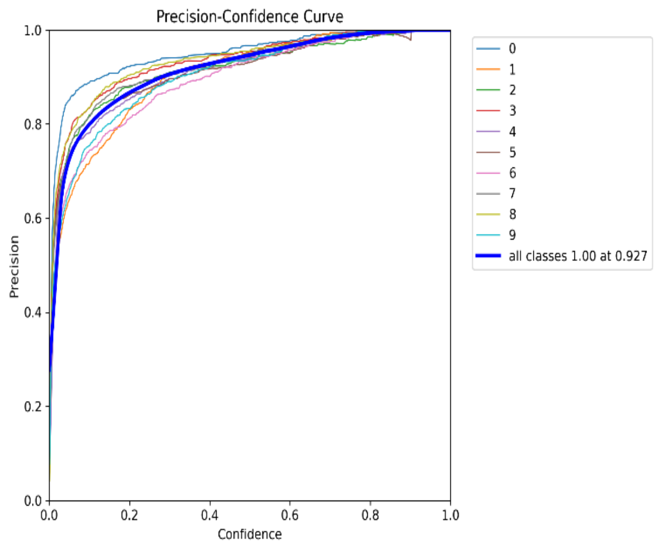


Fig 5: Precision-Confidence curve for 30 epochs

The accuracy of classification for each label can be observed in the confusion matrix shown in Fig. 6. The model performed well in many instances of detecting the correct handwritten digit class. As expected, most classification errors occurred between the digits 6 and 9, due to their similar in the context of arbitrary orientation, making it challenging to distinguish between the two numerals.

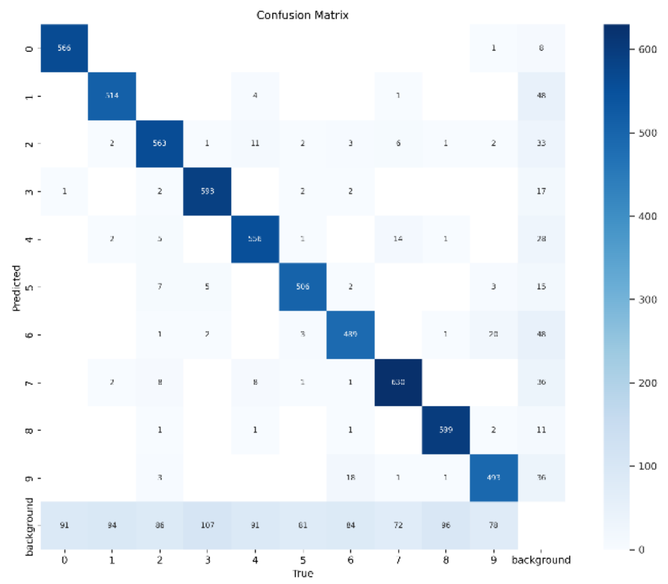


Fig 6: Confusion Matrix

The figure below presents comprehensive performance metrics, including box loss, class loss during both the training and validation stages, as well as mAP@50, mAP@50-95, recall, and precision.

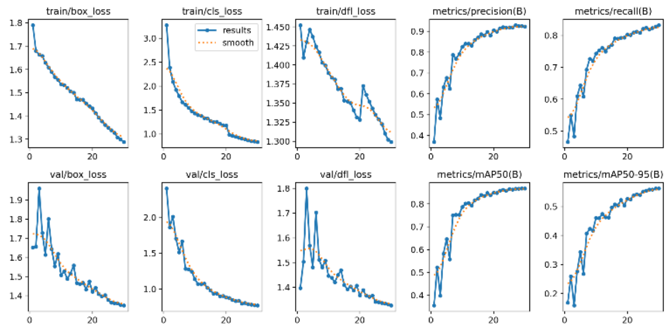


Fig 7: Several Validation Metrics for YOLOv8 over 30 epochs

V. Conclusion

This project implemented a handwritten digit recognition system using transfer learning with the YOLOv8 model. After cleaning and augmenting the dataset, the model demonstrated effective detection. The model's performance was validated through comprehensive metrics, such as Intersection Over Union (IoU) and Mean Average Precision (mAP), showcasing the effectiveness of deep learning in digit recognition. This project highlighted the importance of data preparation and evaluation for achieving high accuracy.

Several hyperparameters, such as the number of epochs, can be fine-tuned to improve accuracy. Additionally, the batch size and learning rate can be adjusted to optimize performance, with techniques like Grid Search Cross-Validation (CV) being useful, given sufficient time and computational resources. Expanding and diversifying the dataset during training may help capture finer features, potentially reducing misclassification errors, such as the confusion between the digits 6 and 9. Furthermore, removing mislabeled data can enhance model accuracy, especially considering the frequency with which these instances are observed.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Jun. 2016. doi:10.1109/cvpr.2016.91

[2] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017. doi:10.1109/tpami.2016.2646371

[3] Pillow Contributors, "Pillow: Python Imaging Library (PIL Fork)," GitHub, 2023. [Online]. Available: https://github.com/python-pillow/Pillow. [Accessed: Dec. 4, 2024].

[4] Ultralytics, "Ultralytics YOLOv8: State-of-the-art object detection," GitHub, 2023. [Online]. Available: https://github.com/ultralytics/yolov8. [Accessed: Dec. 4, 2024].

[5] torchmetrics, "torchmetrics.detection: Metrics for evaluating detection models," GitHub, 2023. [Online]. Available: https://github.com/torchmetrics/torchmetrics. [Accessed: Dec. 4, 2024].