Title: LEGO Piece Detection and Counting Using YOLO

Author: [Yuming Huang]

Methods

Dataset Preparation

The dataset used is the "Biggest LEGO Dataset - 600 Parts" from Kaggle. Images were reviewed to ensure proper annotation, and mislabeled images were removed. All labels were converted into a single class: "Lego." The dataset was split into 70% for training, 15% for validation, and 15% for testing. It was then converted into YOLO format for compatibility with the model.

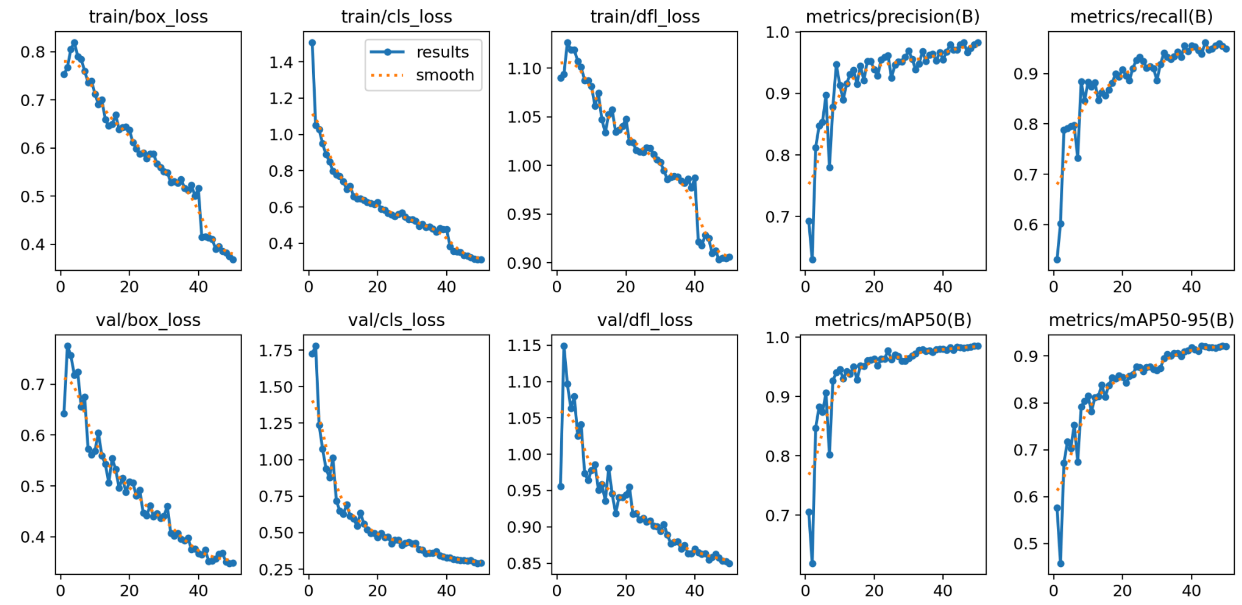
Model Training

The YOLO (You Only Look Once) model was trained using Google Colab with CUDA acceleration. The built-in YOLO loss function was used, which includes box loss for bounding box localization, classification loss for class prediction, and Distributive Focal Loss (DFL) for improved object localization. The training hyperparameters were set as follows: batch size of 16, 50 epochs, the learning rate of 0.01 with cosine decay, AdamW optimizer, and an IoU threshold of 0.5 for mean Average Precision calculation.

Results and Discussion

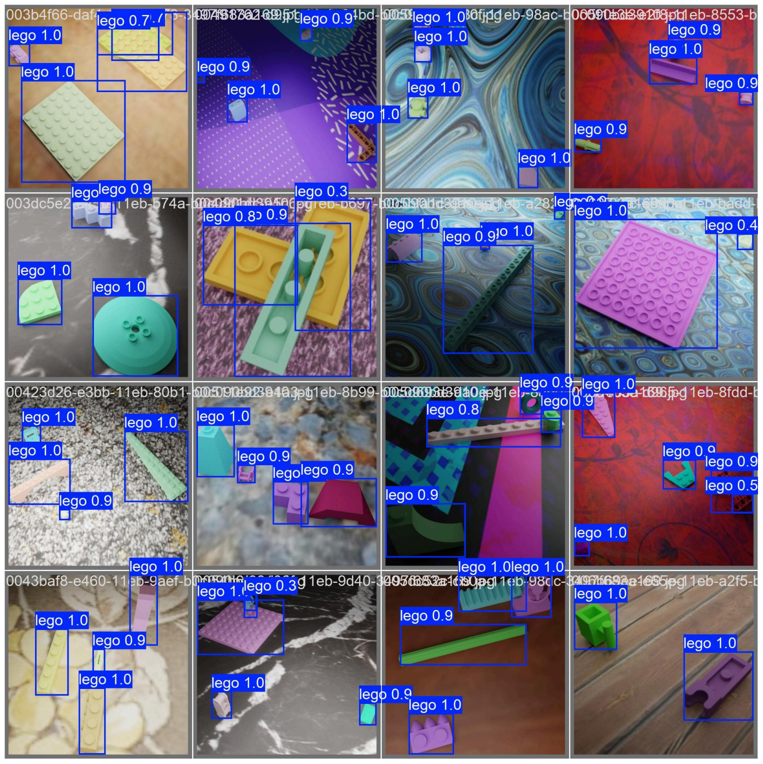
Training Performance Analysis

The training results, as shown in the attached image, indicate that box loss and classification loss consistently decreased over epochs, demonstrating effective learning and minimized errors in bounding box placement and classification. Validation loss followed a similar decreasing trend, confirming that the model generalized well without overfitting. Precision and recall improved throughout training, reaching stable high values, indicating accurate LEGO piece detection with minimal false positives and false negatives. Mean Average Precision (mAP) results showed that mAP@50 reached approximately 0.95, meaning the model correctly detected most objects at an IoU threshold of 0.5. The mAP@50-95 metric displayed a stable increasing trend, confirming the model's robustness across varying IoU thresholds.



Limitations

Despite strong performance, the dataset was limited to 1000 images, which may not fully represent all LEGO pieces. Additionally, the model does not differentiate between different LEGO piece types; it only detects and counts them. The models also cannot detect overlapping Lego pieces precisely. The second-row second-column image demonstrates that two overlapping pieces were detected as three pieces.



Conclusion

This project successfully trained a YOLO model to detect and count LEGO pieces with high accuracy (~95% mAP@50). The model demonstrated strong generalization performance with decreasing loss and increasing precision-recall metrics. Future improvements could involve increasing dataset size for better robustness, applying data augmentation techniques, and experimenting with other object detection architectures such as Faster R-CNN or YOLOv8.

References

- Kaggle Dataset: https://www.kaggle.com/datasets/dreamfactor/biggest-lego-dataset-600-parts
- YOLO Documentation: https://github.com/ultralytics/yolov5
- Google Colab for CUDA Training: https://colab.research.google.com/