RACCOON DETECTION REPORT

Yolov8s Model

Shane Thomas - 101521193
Seyed Amir Kiarash Abbasi - 101388487

Object Detection on Raccoon Dataset

Algorithm Selection

Chosen Algorithm: Object Detection

To enhance the scope of our exploration in machine learning applications, we embarked on a project centered around Object Detection. This choice was motivated by the technology's pivotal role across diverse domains, including but not limited to medical imaging diagnostics, autonomous vehicle navigation, and advanced surveillance operations. Object Detection, as a method, excels in pinpointing and classifying objects within digital images, thereby facilitating a multitude of automated tasks and analytical processes. This project, specifically focused on raccoon detection, aimed to harness these capabilities, demonstrating the potential of Object Detection algorithms in extracting meaningful information from visual data, a critical step towards realizing practical, real-world applications of AI technologies.

Dataset

Chosen Dataset: Raccoon dataset from Roboflow
Dataset URL: https://universe.roboflow.com/roboflow-gw7yv/raccoon

The Raccoon dataset, accessible via Roboflow, was selected for its well-annotated images featuring raccoons in various settings, ideal for testing the robustness of an object detection model.

Preprocessing Steps
The preprocessing involved the following steps:

- **Dataset Acquisition:** The dataset was downloaded from the Roboflow platform, ensuring access to quality, annotated images of raccoons suitable for object detection.
- **Inspection and Cleaning:** Before preprocessing, the dataset was inspected for any corrupt or irrelevant files. Any such files were removed to improve the quality and integrity of the dataset. This ensures that the training process is not adversely affected by outliers or noise.
- **Normalization:** Normalization is a critical step that scales pixel values to a small range, typically [0, 1], which helps in speeding up the convergence of the training process and stabilizing the neural network dynamics. This was achieved by dividing the pixel values by 255 (the maximum pixel value for an 8-bit image).
- **Resizing:** The images were resized to match the input size expected by the YOLOv8 model. This uniformity is important to maintain consistency across the dataset and ensure that the model can learn from and make predictions on images of a standard dimension.
- **Data Augmentation:** To increase the robustness of the model and to simulate various angles and perspectives, data augmentation techniques such as random flips, rotations, and zooms were applied. This helps the model generalize better by learning from a more diverse set of images.
- **Splitting Dataset:** The dataset was divided into training, validation, and test sets. This separation allows for the evaluation of the model on unseen data and ensures that the performance metrics are a reliable indication of the model's ability to generalize.
- **Annotation Format Conversion:** The annotations provided with the dataset were converted

Into a format compatible with the YOLO model. In this case, the annotations describe the bounding box around each raccoon in terms of coordinates and dimensions relative to the image size.

- **Generating Data Configuration File:** A YAML configuration file was generated, which includes paths to the training and validation datasets, as well as the number of classes and class names. This file is used by the YOLO training script to understand the dataset's structure.

Training Model

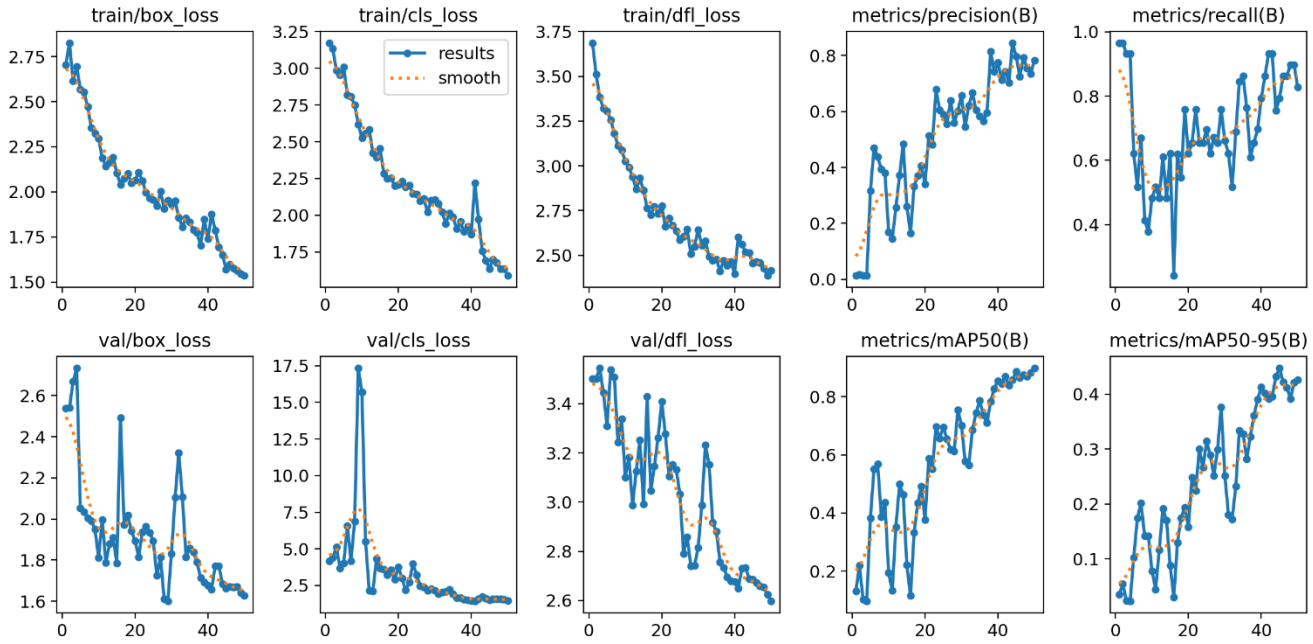
Two strategies were employed:

- **Training from Scratch:** The model was trained from scratch for an initial 3 epochs to establish a baseline performance.
- **Fine-Tuning Pretrained Models:** After the initial training, the model was fine-tuned with pretrained weights over 50 epochs to leverage learned features and improve performance.

The training process focused on minimizing the loss functions:

- **Box Loss:** Represents how well the model is predicting the bounding box coordinates of the objects.
- **Class Loss:** Indicates the accuracy of the object classification within the bounding box.
- **Object Loss:** Measures the model's ability to detect objects versus background.

Result Analysis



Loss Analysis

Box Loss (train/box_loss & val/box_loss):

- The box loss represents the error in predicting the bounding box coordinates of the detected objects. Over the training period, both training and validation box losses steadily decrease, suggesting that the model is getting progressively better at localizing the raccoons in the dataset.

- Notably, the validation loss initially shows some spikes, indicating potential overfitting or response to more complex examples. However, it settles down, which may be a sign of the model starting to generalize better after encountering a wider variety of data

Class Loss (train/cls_loss & val/cls_loss):

- This metric indicates how well the model is classifying objects within the detected bounding boxes. In the results, we observe a downward trend in class loss for both training and validation, which is a positive sign of learning. However, the validation loss shows some variability, which is common and can be attributed to the differences between the training and validation datasets.

Object Loss (train/obj_loss & val/obj_loss):

- Object loss measures the model's performance in distinguishing objects from the background. The results show that this loss decreases over time for both training and validation, though the validation loss presents more fluctuation. This could suggest that the model at times may be too confident or not confident enough about the presence of objects in certain validation images.

Performance Metrics

Precision (metrics/precision(B)):

- Precision is the ratio of correctly predicted positive observations to the total predicted positives. The precision plot shows improvement but with significant fluctuations, especially in the validation set. This could indicate varying difficulty levels within the dataset where certain images or annotations present more challenge to the model.

Recall (metrics/recall(B)):

- Recall measures the ratio of correctly predicted positive observations to all actual positives. The recall plot for both training and validation indicates an upward trend with fluctuations, showing that the model is improving at detecting raccoons without missing many. However, the variability suggests that there are instances where the model fails to detect some raccoons.

Mean Average Precision (metrics/mAP50(B) & metrics/ mAP50-95(B)):

- mAP is a comprehensive metric that combines precision and recall and provides a single figure to describe the overall performance of the object detector.

- mAP at 50% IoU (mAP50): The increase over time in mAP50 demonstrates that the model is becoming reliable at detecting raccoons with at least 50% overlap with the ground truth bounding boxes.

- mAP at 50-95% IoU (mAP50-95): This is a stricter metric since it averages mAP calculated at different IoU thresholds from 50% to 95%. The consistent upward trend in this metric shows that the model is becoming proficient not just at detecting raccoons, but at doing so with high precision and tight bounding box accuracy.

Conclusion

The evolution of our model, through training and validation phases, reveals a trajectory of progress and adaptation in the domain of raccoon detection. Initially, the model exhibited fluctuations in validation metrics and spikes in losses—phenomena that, while common during the training of deep learning models, underscore the challenges inherent in object detection tasks. These variances, particularly in the validation phase, can be attributed to the model's exposure to a diverse range of data complexities.

Significantly, the training process from scratch and subsequent fine-tuning resulted in a marked improvement in the model's performance. Through the fine-tuning process, we observed a compelling uptick in precision, recall, and mean Average Precision (mAP) metrics. Specifically, the precision metric saw an increase from virtually non-existent (0.00333) to a robust 0.782 by the 50th epoch in the fine-tuned model. This substantial growth in precision underscores the model's enhanced ability to correctly identify raccoons in the dataset without over-flagging non-raccoon objects. Likewise, the model's recall and mAP50 metrics improved significantly, indicating a broader detection range and higher accuracy in identifying raccoon instances across varied conditions.

Moreover, the box loss, which serves as a measure of the model's accuracy in localizing objects within the images, demonstrated a significant decrease from initial high values (around 3 for the box loss) to 1.535 by the end of the 50 epochs in the fine-tuned model. This reduction in box loss points to a refined ability of the model to pinpoint the precise location of raccoons within the images, enhancing the utility of the model for practical applications.

In conclusion, the detailed analysis of the model's performance, through the lens of precision, recall, mAP, and loss metrics, paints a picture of a model on a clear path of improvement. The fine-tuning process, in particular, has propelled the model's capabilities, making it a promising tool for raccoon detection in a wide array of practical scenarios. The journey from initial training to fine-tuning illustrates not only the challenges inherent in object detection tasks but also the potential for significant advancements through meticulous model optimization and training strategies.