OBJECT DETECTION PROJECT

Keese Phillips

Contents

Dataset Used:

The dataset used for this project was formed through fair use images sourced online. These images were individually annotated using the labelme software package. The dataset contains two classes a book and laptop class. The total dataset contains 109 images of laptops and 103 images of books. Each image only contains a single instance of a specific object class. The dataset originally contained utensils in place of the books class, the utensils images often contained more than a single instance of the class in one image. When training the Faster RCNN model as discussed below the model seemed to only predict the utensils class, this led to the belief that the class imbalance in the dataset could potentially be the root cause of the model's high false positives in relation to the utensils class. The Faster R-CNN model employed a custom dataset that normalized and applied transformations to the original images before passing them to the model for training. The original annotations were not in COCO format, which is accepted by the YOLO model. These annotations and images were then parsed and converted into COCO format for the YOLO model.

Related Work:

The Faster R-CNN model was originally introduced by Ren et. al. in *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* They advanced upon the concepts introduced in the Fast R-CNN through the introduction of the "Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position" (Ren et. al.). The new framework in the Faster R-CNN model achieved a new benchmark mAP when compared to the selective search and Fast R-CNN models (Ren et. al.). The YOLO model was originally introduced by Redmon et al. in *You only look once: Unified, real-time object detection*. The authors introduced the "object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance" (Redmon et al.). The original version of the YOLO model did not improve upon Faster R-CNN in terms of mAP but the YOLOv1 model achieved 21 fps in comparison to Faster R-CNN 7 fps. The original

YOLO introduced the ability to predict 3x fps in comparison to the existing frameworks leading to an ability to produce more real-time predictions. The YOLO model has greatly improved since the original model and is currently on version 8 which was used in this project.

Block Diagram:

See GitHub under Assets

Algorithms:

This project involved the use of two models, the Faster R-CNN model and Yolov8 model respectively. Faster R-CNN improved upon the Fast R-CNN model by employing a Region Proposal Network (RPN). The model first employs a convolutional layer to extract the feature maps. These feature maps are used to create the region proposal network. The feature maps and region proposal network are then passed to the detector. The region proposals are mapped onto the feature map and the extracted features are then passed to the ROI pooling layer. The pooled features are then passed to two separate models, a classifier which classifies the region proposal and the regressor which fine tunes the bounding box for the region proposal.

The YOLO model functions differently internally than the Faster R-CNN. The YOLO model provides end to end optimization of object detection. The YOLO model divides up the image into grid cells. The entire image is passed to the convolutional network to extract features. For each grid cell in the resulting feature maps, the model predicts a fixed number of bounding boxes along with their confidence scores and class probabilities. Each bounding box prediction includes offsets to predefined anchor boxes, adjusting their center coordinates, width, and height. These anchor boxes are of different scales and aspect ratios to better match various object sizes and shapes. The final detections are obtained by combining these predictions and applying non-maximum suppression to eliminate redundant detections.

Implementation:

In this project, specifically the Faster R-CNN employed ResNet50 for the feature extraction. The Faster R-CNN then used a ROI pooling layer before passing the results to the classifier and regressor for classification and bounding box fine tuning. The model was trained in 50 epochs and used a low learning rate of 1×10^{-4}. The YOLO model employed was version 8.

The YOLO model was trained on 100 epochs. Both models were trained on a low batch size of 4. Both models were then tested in terms of mean average precision, speed, and GPU memory usage during inference of an image. In terms of training, the YOLO model took about 11.4 seconds per epoch whereas the Faster R-CNN model took about 37.2 seconds per epoch. The YOLO model training and validation loss curve did not appear to overfit the dataset and continued to decrease in terms of both training and validation loss (Appendix III). The Faster R-CNN model training and validation loss curve did not significantly decrease after the 10th epoch and possibility began to overfit the dataset after the 40th epoch where the validation loss began to increase slightly (Appendix IV).

Results:

Faster R-CNN

mAP: 0.053

Speed (per image): 84.24 ms

GPU Memory (per image): 651.66 MB

YOLOv8

mAP: 0.582

Speed (per image): 47.30 ms

GPU Memory (per image): 3571.96 MB

Analysis of results:

The Faster R-CNN model performed worse than YOLOv8 in terms of speed and mean average precision. The Faster R-CNN model appears to still suffer from a classification issue. The YOLO model appears to have few false positives for the book class (Appendix I) whereas the Faster R-CNN model appears to have a significant amount of false positive for the book class (Appendix II). The YOLO model also appears to have tighter bounding boxes around the object in comparison to the Faster R-CNN model. The YOLO model appears to have more GPU memory usage in comparison to the Faster R-CNN model during inference.

Conclusion:

In comparison between the two models, the YOLO model significantly outperformed the Faster R-CNN in terms of speed and mAP. The YOLO model did not appear to suffer from the classification issue that plagued the Faster R-CNN model. This issue could possibly be explained by the small dataset sample size, the total annotated images was 212. With a greater sample size, the Faster R-CNN could possibly perform with a much higher mean average precision. The YOLO model appeared to outperform the Faster R-CNN model on both the fitting of the bounding boxes and the classifications of the objects in the image. In future work, the possibility of adding more annotated images could potentially reduce the false positive rate in the Faster R-CNN model and could overall increase the ability of both models to accurately classify and detect all the images in either class with a higher mAP.

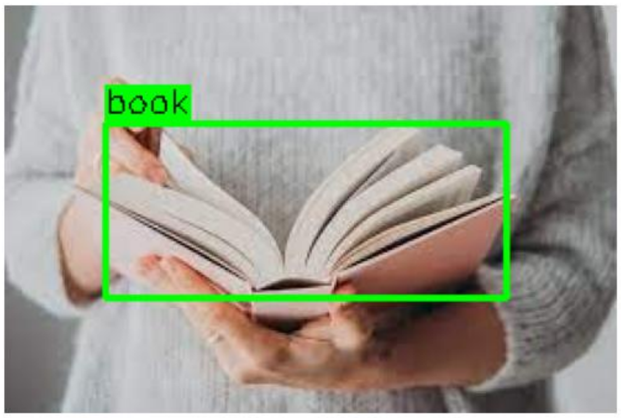
References:

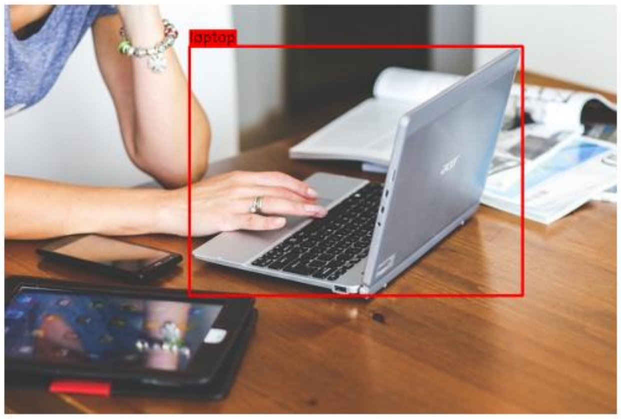
Redmon, Joseph, et al. “You only look once: Unified, real-time object detection.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, https://doi.org/10.1109/cvpr.2016.91.

Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." IEEE transactions on pattern analysis and machine intelligence 39.6 (2016): 1137-1149.

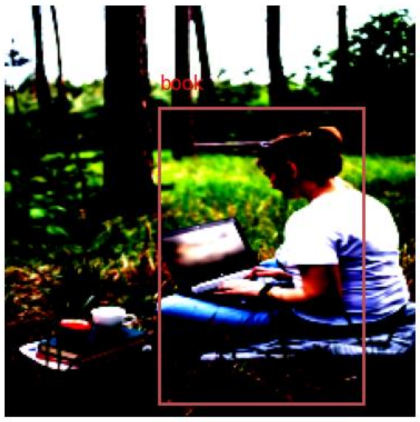
Appendices:

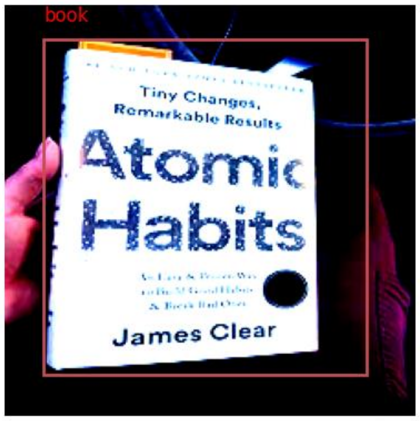
Appendix I: YOLO model inference results



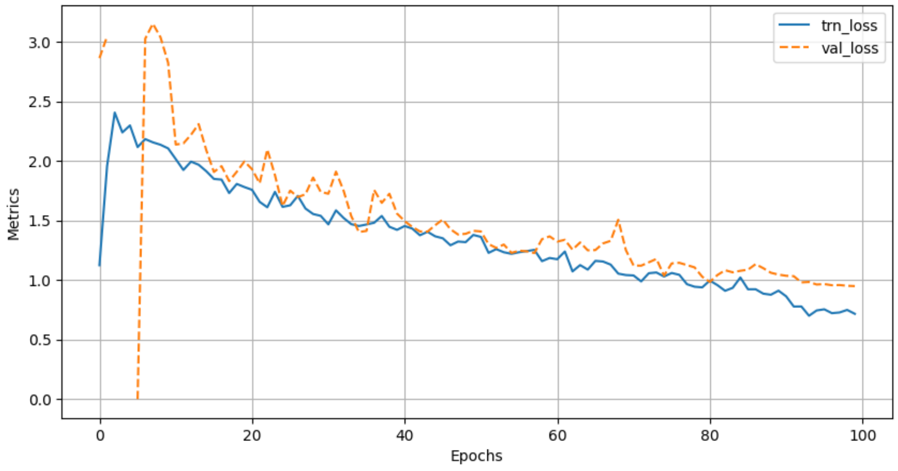


Appendix II: Faster R-CNN model inference results

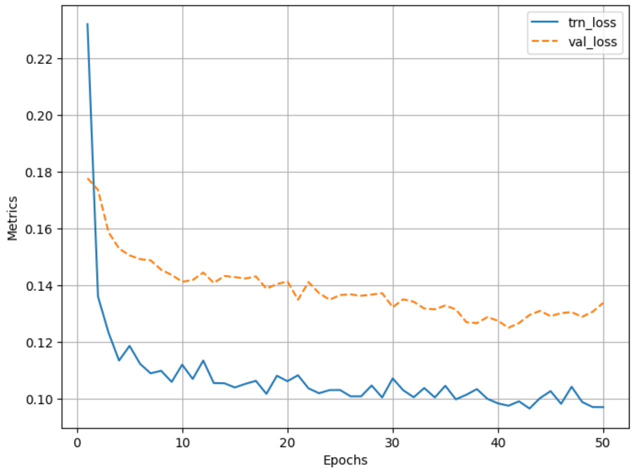




Appendix III: YOLO model training/validation loss curve



Appendix IV: Faster R-CNN model training/validation loss curve


trn_loss
val_loss
Metrics
Epochs