Computer Vision Object Detection

Using Faster R-CNN, Mask R-CNN, and YOLOv8

Project Lead: Hector R. Gavilanes
Deep Learning Architect: Jacob Knight
Data Scientist: Kyle Knuth Software Engineer: Sai Meghana

2024-02-16

Table of contents

Objectives

The initial objective of this research paper is to evaluate the performance of the Faster R-CNN methodology on a dataset provided by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun in their paper titled "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," introduced at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2015 [1]. We aim to replicate the results presented in this article, and gain a comprehensive understanding of Faster R-CNN, which will serve as the foundation for our future testing endeavors. Furthermore, we will assess the performance of Mask R-CNN, and YOLOv8 to offer a comparison of various object detection methodologies. A YOLOv8 custom label was applied to detect military tanks.

Glossary

Term	Definition
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
YOLOv8	You Only Look Once, version 8
RPN	Region Proposal Network
IoU	Intersection over Union
RoI	Region of Interest
RoIAlign	Region of Interest Align
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
mAP	mean Average Precision
COCO	Microsoft Common Objects in Context

1 Introduction

Object detection is a fundamental task in computer vision, playing a crucial role in various applications such as autonomous vehicles, surveillance systems, and medical imaging. Recent advancements in deep learning have led to the creation of sophisticated architectures such as Faster R-CNN, Mask R-CNN, and YOLO (You Only Look Once), all of which have substantially improved object detection performance. These architectures have revolutionized the field by achieving superior accuracy in detecting objects within images. However, despite significant advancements, a critical trade-off between accuracy and processing speed persists. It is essential to carefully consider this balance when selecting the most suitable model for practical applications.

The efficiency of object detection models is essential, particularly in real-time applications where rapid decision-making is vital. Currently, object detection primarily relies on region proposal methods and region CNNs. For instance, Selective Search, a widely used region proposal method, exhibits a speed of 2 seconds per image when implemented on the CPU, which is considered slow for many real-world applications. On the other hand, EdgeBoxes, another region proposal method, offers improved speed, clocking in at 0.2 seconds per image. However, a notable constraint remains— even with advancements, the region proposal step still demands substantial computational resources when executed on the CPU rather than the GPU.

Moreover, comparing the performance of Faster R-CNN with region proposal methods poses challenges. Since R-CNNs leverage GPU acceleration while region proposal methods primarily rely on CPU computation. A fair comparison between the two becomes challenging. To address this issue, Region Proposal Networks (RPNs) were introduced in 2015 as part of the Faster R-CNN framework [1]. A novel approach that integrates the region proposal step directly into the object detection algorithm itself. This architectural innovation aims to streamline the object detection process and improve overall efficiency by leveraging GPU resources effectively.

2 Literature Review

The evolution from R-CNN to faster R-CNN represents a significant advancement in object detection algorithms, especially in speed and efficiency. A brief history of the development progression of R-CNN, Fast R-CNN, and Faster R-CNN adds valuable context to our study on object detection architectures. It will help to understand the evolution of these models, the motivations behind their development, and the improvements made over time.

2.1 R-CNN (Region-Based Convolutional Network)

R-CNN was a breakthrough in object detection. It employed a multi-stage approach that involved a selective search for generating region proposals followed by a convolutional neural network for feature extraction and a support vector machine (SVM) for object classification within each region.

2.1.1 Region Proposal Generation

R-CNN began with generating region proposals using the selective search algorithm [2]. Selective search is a method for identifying potential object regions in an image based on low-level features such as color, texture, and intensity. It produces a set of bounding boxes that mostly have objects.

2.1.2 Feature Extraction

Following the generation of region proposals, R-CNN employed a pre-trained convolutional neural network to independently extract features from each region. Typically, the chosen CNN model was AlexNet, pre-trained on the ImageNet dataset specifically for image classification tasks.

2.1.3 Fine-tuning and Classification

Following feature extraction, the extracted features were input into a distinct classifier to ascertain the presence of objects within the regions. R-CNN utilized a support vector machine (SVM) [2] for this classification task. Each SVM was trained to discern whether the feature corresponded to a particular object or background.

2.1.4 Bounding Box Regression

Following classification, R-CNN conducted bounding box regression to enhance the accuracy of the detected object locations. This process adjusts the bounding boxes produced by the region proposal algorithm to align more precisely with the actual object locations within the regions.

2.1.5 Non-Maximum Suppression

Lastly, R-CNN implemented non-maximum suppression to eliminate redundant detections, ensuring that each object is detected only once.

2.1.6 Drawbacks

An initial drawback of the R-CNN architecture was its computational inefficiency during inference, primarily stemming from its sequential processing of region proposals. Processing each region proposal independently led to redundant computations and prolonged inference times.

2.2 Fast R-CNN

Fast R-CNN, a significant advancement in object detection, introduces several innovations enhancing both training and testing efficiency while improving detection accuracy. Unlike its predecessors, Fast R-CNN leverages the VGG16 network, achieving a remarkable 9x increase in training speed compared to R-CNN [3]. At test-time, Fast R-CNN demonstrates an impressive speed improvement of 213x, making it significantly faster and more practical for real-world applications. Additionally, Fast R-CNN outperforms previous methods in terms of mean Average Precision (mAP) on benchmark datasets like PASCAL VOC 2012 [3].

–Fast R-CNN represents a faster and more efficient iteration of the original R-CNN for object detection. It tackles the computational inefficiencies of R-CNN by introducing a unified

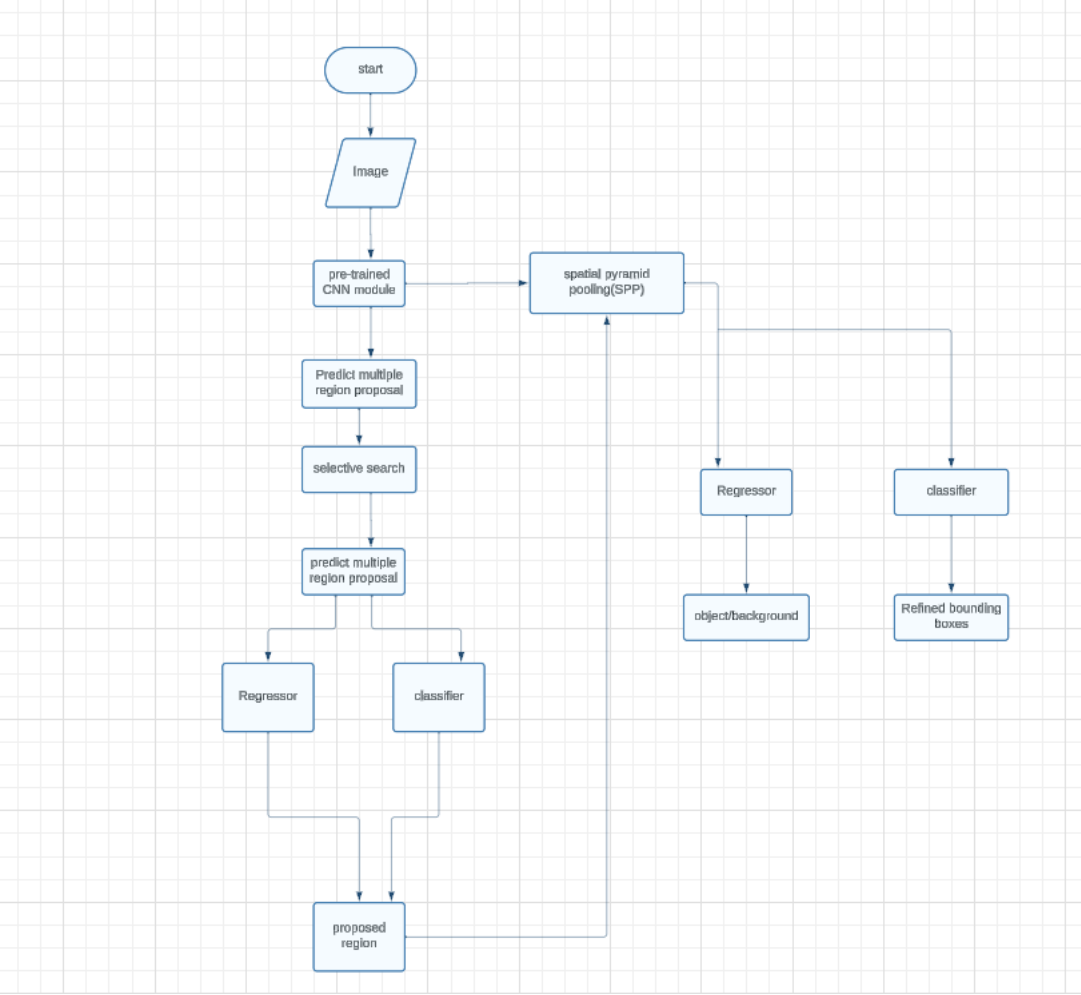

Figure 2.1: Flowchart of R-CNN

architecture that consolidates region proposal generation, feature extraction, and object classification into a single network. This approach dramatically reduces redundant computations and accelerates the inference process.–

2.2.1 Architecture

Expanding upon its architecture, Fast R-CNN incorporates several key components to achieve its performance gains. Fast R-CNN addresses the speed and efficiency limitations of R-CNN by proposing a unified architecture that integrates region proposal generation, feature extraction, and object classification into a single network.

Firstly, it utilizes a Region Proposal Network (RPN) to generate region proposals directly from the convolutional feature maps, eliminating the need for external proposal methods like selective search. The RPN functions by sliding a small grid (essentially a compact CNN) across the evolving feature map to predict the spatial dimensions (bounding boxes) and associated probability scores for objects within each sliding window. This streamlines the detection process and enhances efficiency. Furthermore, Fast R-CNN introduces the Region of Interest (RoI) pooling layer [3], allowing feature extraction from region proposals of varying sizes without the need for expensive resizing operations. This enables precise alignment of features with the corresponding regions of interest, leading to improved localization accuracy. Moreover, Fast R-CNN adopts a unified network architecture, enabling end-to-end training of both the region proposal and object detection tasks. This approach ensures better optimization, and facilitates seamless integration of different components, contributing to the overall efficiency and effectiveness of the model.

2.2.2 RoI Pooling Layer

The RoI pooling layer employs max pooling to transform features within any valid region of interest into a compact feature map with a fixed spatial extent of H $\times$ W (e.g., 7 $\times$ 7) [3], where H and W are layer hyper-parameters independent of any specific RoI. A more efficient training strategy leverages feature sharing throughout the process. During Fast R-CNN training, stochastic gradient descent (SGD) minibatches are hierarchically sampled: N images are initially sampled, followed by R/N RoIs from each image. Notably, both during forward and backward passes, ROIs from the same image share computation and memory. Fixed-size feature maps yielded from RoI pooling are input to fully connected layers for object classification and bounding-box regression. Feature classification aims to detect features and assign class probabilities for each proposed location, while bounding box regression seeks to enhance the localization accuracy of detected features. Fast R-CNN introduces multitasking loss functions that amalgamate classification and bounding box regression losses. This enables joint training of both tasks, ensuring the network learns to predict object location and precise bounding boxes concurrently.

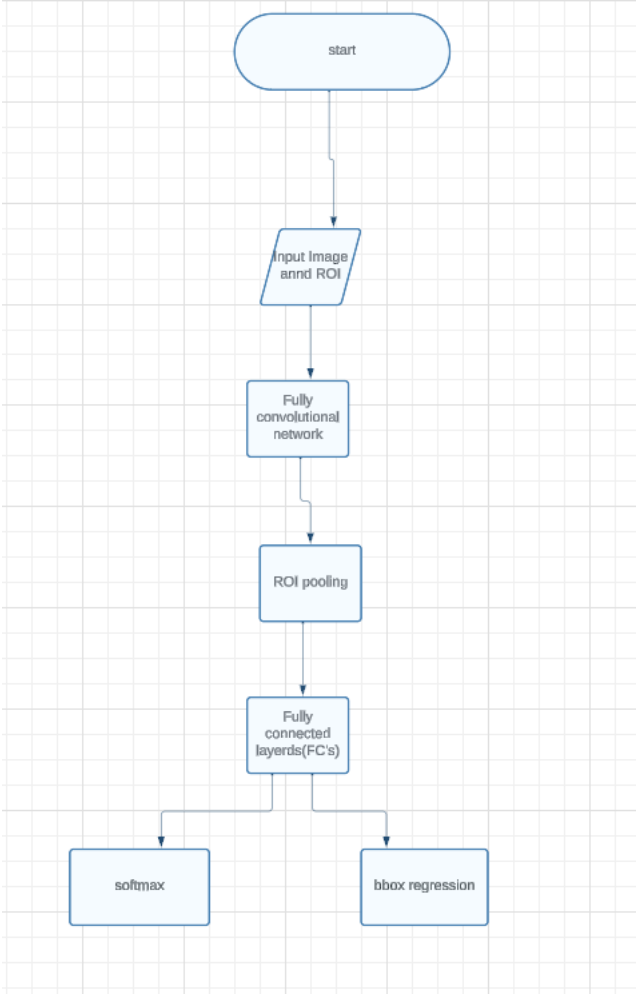

Figure 2.2: Flowchart of Fast R-CNN

2.2.3 Advantages of Fast R-CNN

In contrast to R-CNN, which comprises multiple independent steps (such as region proposal generation, feature extraction, classification, and bounding box regression), Fast R-CNN integrates these processes within a unified network architecture. This enables end-to-end training of the entire pipeline, leading to improved optimization and potentially higher accuracy. Fast R-CNN achieves variable sharing across all spatial dimensions of an image. Unlike R-CNN, which extracts features independently for each image, Fast R-CNN performs feature extraction on the entire image only once. This shared computation across convolutional features significantly minimizes redundant calculations, expediting the inference process.

2.3 Faster R-CNN

Compared to preceding methodologies, Faster R-CNN achieves more efficient object recognition by seamlessly integrating regional proposal steps directly into the network architecture. Subsequent advancements have expanded upon this framework, establishing it as a fundamental component in object recognition. The architecture commences with a feature extraction backbone, typically a convolutional neural network (CNN) pre-trained on extensive datasets such as ImageNet. This backbone extracts pertinent features from the input image.

2.3.1 Region Proposal Network (RPN)

Faster R-CNN further improves the efficiency of object detection by introducing the Region Proposal Network (RPN), which generates region proposals directly from the convolutional feature maps. It eliminates the need for external region proposal methods like Selective Search, resulting in faster and more accurate proposal generation.

The feature maps derived from the backbone network are input to a Region Proposal Network (RPN) [1]. The RPN, a compact fully convolutional network, employs a sliding window approach (typically 3x3) over the feature maps to generate region proposals. The RPN yields a collection of bounding box proposals accompanied by objectness scores, indicating the probability of containing an object. These proposals are generated by leveraging predefined anchor boxes of varying scales and aspect ratios.

2.3.2 Anchor Boxes

The RPN generates region proposals by predicting offsets and scales for a predefined set of anchor boxes at each spatial position in the feature maps. These anchor boxes, varying in size and aspect ratio, serve as reference boxes for the proposal generation process.

2.3.3 Region of Interest (RoI)

Region of Interest (RoI) refers to a specific area or region within an image that is selected for further analysis or processing. In the context of object detection and image segmentation tasks, RoIs typically represent regions where objects of interest are located.

2.3.4 Classifier and Bounding Box Regressor

The RoI-pooled or RoIAlign features are separately input into branches for classification and bounding box regression.

- **Classification:** The features traverse a classifier (e.g., fully connected layers followed by softmax) to predict the probability of each region proposal belonging to different object classes. Bounding Box Regression: Another set of fully connected layers predicts refined bounding box coordinates for each region proposal.
- **Loss Function:** The network undergoes end-to-end training using a multi-task loss function, which combines losses from the RPN (objectness score prediction and bounding box regression) with those from the classification and bounding box regression branches.

2.3.5 Non-Maximum Suppression (NMS)

Following prediction, non-maximum suppression is employed to discard redundant and overlapping detections based on their confidence scores and bounding box coordinates.

2.3.6 Output

The final output comprises a collection of object detections alongside their bounding boxes and corresponding class labels.

2.3.7 Advantages of Faster R-CNN

- **Efficient Region Proposal Generation:** The Region Proposal Network (RPN) significantly reduces the computational load of region proposal methods, enabling nearly cost-free region proposals by sharing convolutional features.
- **End-to-End Training:** RPN and Fast R-CNN can be jointly trained to share convolutional features, leading to a unified network architecture.
- **High-Quality Proposals:** RPN produces high-quality region proposals, enhancing detection accuracy compared to traditional methods like Selective Search and EdgeBoxes.

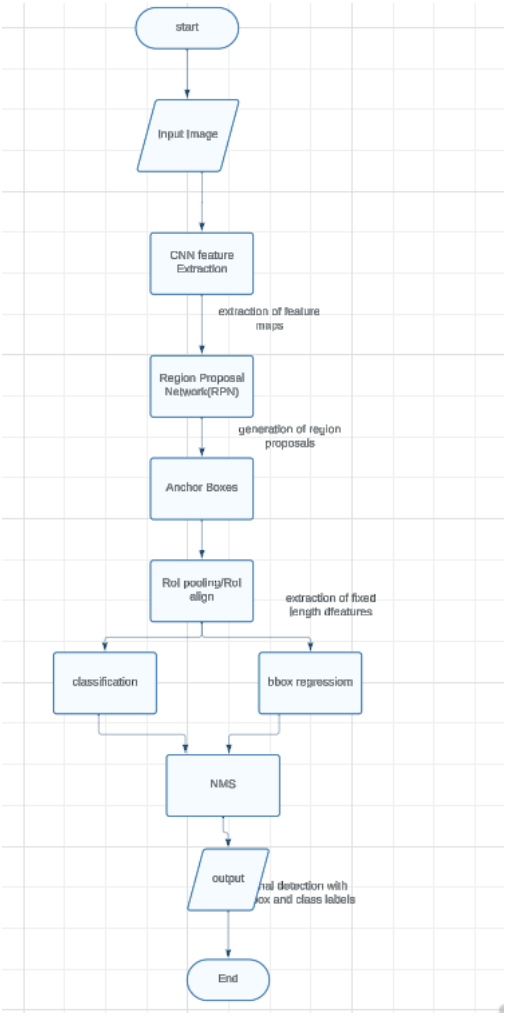

Figure 2.3: Flowchart of Faster R-CNN @ren2015fasterrcnn

- **Practical Implementation:** The proposed method achieves state-of-the-art object detection accuracy on benchmarks like PASCAL VOC [1] while maintaining a practical frame rate of 5fps on a GPU.

2.4 Mask R-CNN

Mask R-CNN extends Faster R-CNN by adding a branch to predict the segmentation mask at each Region of Interest (RoI), alongside existing branches for classification and bounding box regression. This is achieved by introducing a small, fully convolutional network on top of each RoI. Below we briefly describe the architecture components.

2.4.1 Backbone CNN

Similar to Faster R-CNN, Mask R-CNN begins with a backbone that extracts features from the input image. This backbone network is typically pre-trained on a large dataset like ImageNet to learn generic image features.

2.4.2 Region Proposal Network (RPN)

The RPN takes feature maps from the backbone network and generates region proposals using anchor boxes, refined based on their likelihood of containing objects. The region proposals generated by the RPN are subsequently used for both object detection, and instance segmentation in the Mask R-CNN framework.

2.4.3 Region of interest Align (RoIAlign)

For each region proposal generated by the RPN, features are extracted from the feature maps using RoIAlign. RoIAlign is a technique used in CNNs for object detection tasks. RoIAlign improves upon RoIPool, a previous method, by eliminating the quantization step in the pooling operation, resulting in more accurate feature extraction from regions of interest [4]. It precisely aligns features extracted from arbitrary-shaped regions with the spatial layout of the feature map, enabling more accurate object localization and segmentation.

The RoIAlign layer extracts features from each region proposal, preserving spatial information better than previous pooling methods, ensuring accurate alignment of features with the region of interest.

2.4.4 Parallel Branches

Mask R-CNN introduces parallel branches for object detection and instance segmentation.

- Object Detection Branch: Determines the class of each object within the region proposal and refines bounding box coordinates through classification and regression. – Mask Prediction Branch: Predicts segmentation masks for each object within the region proposal, generating pixel-level masks for object instances using a small fully convolutional network applied to each RoI.
- Loss Functions: Mask R-CNN employs a multi-function loss function that combines object detection (classification and bounding box regression) and instance segmentation (mask prediction) losses, weighted by hyperparameters.

$$L = L_{cls} + L_{box} + L_{mask}$$

- Training: The entire Mask R-CNN network is trained end-to-end using backpropagation with stochastic gradient descent (SGD) or other optimization algorithms, typically starting with pre-trained weights and fine-tuning for the specific task.

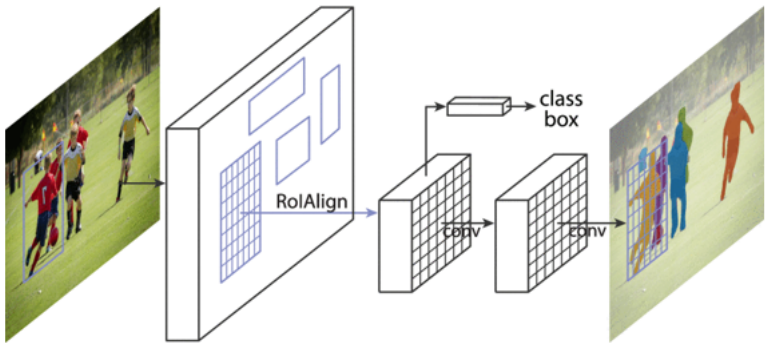

Figure 2.4: Mask R-CNN framework for Instance Segmentation @he2020maskrcnn

2.4.5 Advantages over Faster R-CNN

- Offers pixel-level segmentation alongside object detection and bounding box regression.
- Facilitates instance segmentation, segmenting each object instance in an image separately.
- Enhances understanding of object shapes and boundaries with finer detail.
- Despite increased complexity, maintains comparable speed and efficiency, particularly in scenarios necessitating instance-level segmentation.

Overall, Mask R-CNN represents a substantial advancement over previous methods by integrating object detection and instance segmentation into a unified architecture, rendering it a versatile solution for diverse computer vision applications.

3 Methodologies

3.1 Fast R-CNN Training

Fast R-CNN follows a single-stage training process [3]. It begins by processing the image through convolutional and max pooling layers, resulting in a convolutional feature map. Each object proposal then undergoes region of interest pooling, generating fixed-length feature vectors. These feature vectors are fed through fully connected layers, leading to two output layers: one for object class probabilities and the other for real-valued object coordinates. Fast R-CNN's training approach enables higher detection quality, updates all layers during training, and eliminates the need for disk storage for feature caching, contributing to its efficiency and accuracy [3].

3.2 Faster R-CNN Training

On the other hand, Faster R-CNN training is a two-stage process, jointly training a Region Proposal Network (RPN) and a Fast R-CNN detector [1] . In the first stage, the RPN generates region proposals by sliding a small network over the convolutional feature map, refining them, and assigning scores based on their likelihood of containing objects. These proposals serve as input for the second stage, where the convolutional feature map undergoes region of interest (RoI) pooling. This process results in fixed-length feature vectors for each proposal, which are then processed through fully connected layers to predict object class probabilities and refine object bounding box coordinates [3]. This joint training approach allows Faster R-CNN to produce high-quality region proposals and accurately classify and localize objects in images.

3.2.1 Region Proposal Network (RPN)

To produce the object proposals used during training, a Region Proposal Network (RPN) is used. The RPN takes in an image and produces a set of object proposals with an associating score. This region proposal is generated by sliding a small network over the convolutional feature map. At the center of each sliding window is a Translation-Invariant Anchor [1]. This anchor is a reference box that each proposal in that region is relative to. These anchors being translation-invariant means that regardless if the image is translated in any way the prediction of that region should not change.

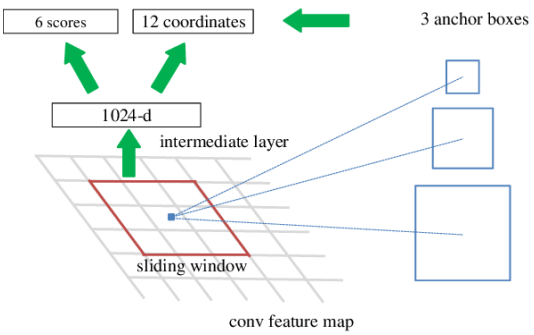

Figure 3.1: Improved anchor boxes in Faster R-CNN @zhou2022visual

Three anchor boxes are depicted within each grid, as illustrated above. The image encompasses numerous points, equivalent to a 600 x 600 image, accommodating a 38 x 38 network overlay. The blue point within the image signifies the center of the grid. Each grid center is associated with three anchor boxes and three squares of varying sizes, representing the original anchor boxes delineated in the image [5].

3.2.2 Multi-task Loss Function

RPNs use positive and negative values to assign anchors. A positive value represents a high Intersection-over-Union overlap between the anchor and the ground truth box [1]. A negative value represents the dissociation of an anchor and a certain prediction. The higher the value, the more associated the anchor is to a certain prediction. These values are used in a multi-task loss function. This function will both train for classification and perform bounding box regression. The loss function is defined as:

$$L = L_{cls} + \lambda * L_{reg}$$

Where the RPN consist of two losses.

$$L = (1/N) * \Sigma[L_{cls}(p_i, p_i*) + \lambda * L_{reg}(t_i, t_i*)]$$

The authors describe the above equation by saying [1], “Here, i is the index of an anchor in a mini-batch and pi is the predicted probability of anchor I being an object. The ground-truth label p $*$ i is 1 if the anchor is positive, and is 0 if the anchor is negative. ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t $*$ i is that of the ground-truth box associated with a positive anchor. The classification loss Lcls is log loss over two classes (object vs. not object). For the regression loss, we use Lreg (ti, t$*$ i) = R(ti $-$ t $*$ i) where R is the robust loss function (smooth L1). The term p $*$ i Lreg means the regression loss is activated only for positive anchors (p $*$ i = 1) and is disabled otherwise (p

$*$ i = 0). The outputs of the cls and reg layers consist of {pi} and {ti} respectively. The two terms are normalized with Ncls and Nreg, and a balancing weight ."

For regression, the formula for the parameterized coordinated of the predicted bounding box:

$$\Delta x = (x' - x)/w, \quad \Delta y = (y' - y)/h, \quad \Delta w = log(w'/w), \quad \Delta h = log(h'/h)$$

- (x, y, w, h) represents the coordinates of the default anchor box,
- (x', y', w', h') represents the coordinates of the predicted bounding box,
- Δx, Δy, Δw, and Δh are the parameterized adjustments to the coordinates of the default anchor box to obtain the predicted bounding box coordinates.

3.2.3 Optimization

To optimize the data, backpropagation and stochastic gradient descent are utilized. Start with the initializing from a zero-mean Gaussian distribution. Perform back-propagation through RoI pooling layers and pass partial derivatives of the loss function concerning an activation input "xi" in the RoI pooling layer.

$$\frac{\partial L}{\partial x_i} = \Sigma_r \Sigma_j [i = i * (r, j)] \frac{\partial L}{\partial y_{rj}}$$

The authors describe this backward function as [3], "In words, for each mini-batch RoI r and each pooling output unit yrj, the partial derivative L/ yrj is accumulated if i is the argmax selected for yrj by max pooling. In back-propagation, the partial derivatives L/ yrj are already computed by the backward function of the layer on top of the RoI pooling layer". As iterations continue, weights and biases will be tuned towards values that optimize predictions.

This formula represents the accumulation of partial derivatives with respect to the output units of the RoI pooling layer during backpropagation. It describes how the gradients of the loss function with respect to the outputs of the RoI pooling layer are computed and accumulated during the backward pass of the neural network training process. It helps us understand how the network learns from its mistakes and improves its predictions over time.

3.3 Metrics

One data metric utilized in the chosen paper, and that will also be used in the analysis, is the mean average precision (mAP). This evaluation metric is typically used for object detection, and consists of multiple sub-metrics such as the confusion matrix, recall, precision, and the intersection over union. mAP is mathematically defined as, n is the number of classes, and APk is the average precision of the current class (k). Mean average precision basically takes an established ground truth box around the target and compares it to the detected box from the deep learning model, yielding an accuracy score. A higher accuracy score implies that the deep learning model is accurate with its detection.

$$mAP = (1/n) * \sum_{k=1}^{k=n} (AP_k)$$

- AP_k = the Average Precision of class k
- n = the number of classes.

4 Analysis

The demonstration utilized the ResNet backbone network in conjunction with the COCO dataset. The Microsoft Common Objects in Context (COCO) dataset [6] is a large-scale dataset for object detection, segmentation, and captioning tasks. It contains over 200,000 images, each annotated with bounding boxes around objects in various categories such as people, animals, vehicles, and household items. Additionally, each image is annotated with pixel-level segmentation masks for object instances. The COCO dataset is widely used in computer vision research for benchmarking and evaluating object detection and segmentation algorithms.

Faster R-CNN, known for its proficiency in object detection tasks, has showcased remarkable performance [1] . However, challenges persist regarding its robustness to environmental variations such as shadows, occlusions, and perspective distortions. Furthermore, mislabeling of objects remains a concern, potentially leading to incorrect identifications due to similarities with trained categories.gories.

In addition to Faster R-CNN, Mask R-CNN was employed, utilizing a pretrained model from the COCO dataset. This model underwent testing on both images, and a video dataset. Transitioning to YOLOv8, preparation of the data involved annotating the training set. Twenty images of tanks were manually annotated using CVAT.ai to draw bounding boxes around the targets. Subsequently, specific bounding box coordinates for YOLOv8 were obtained. Notably, all tank images for training and testing were sourced from Flickr.

To configure the dataset appropriately for YOLOv8, a yaml file was created, specifying the dataset's structure. This file included details such as the overall path to the data, the locations of the training, validation, and test sets, and the names of the classes to be identified.

Following data preprocessing, YOLOv8 training commenced using the tank training set. Initially, twenty epochs were employed with the medium model. However, during testing, Google Colab encountered stability issues, necessitating a switch to the small model, which resulted in smooth execution.

5 Results

5.1 Faster R-CNN

We used the Faster R-CNN model from TensorFlow Hub (Inception-ResNet-V2). According to the authors [7], Inception-ResNet-V2 is a deep convolutional neural network architecture introduced as part of the Inception family.

The Faster R-CNN results can be seen in images 1-4. In the first image, the model was fairly accurate in identifying individual chairs and a couple of the tables.

Figure 5.1: Faster R-CNN result 1

In the second image, Faster R-CNN correctly identified all beetles.

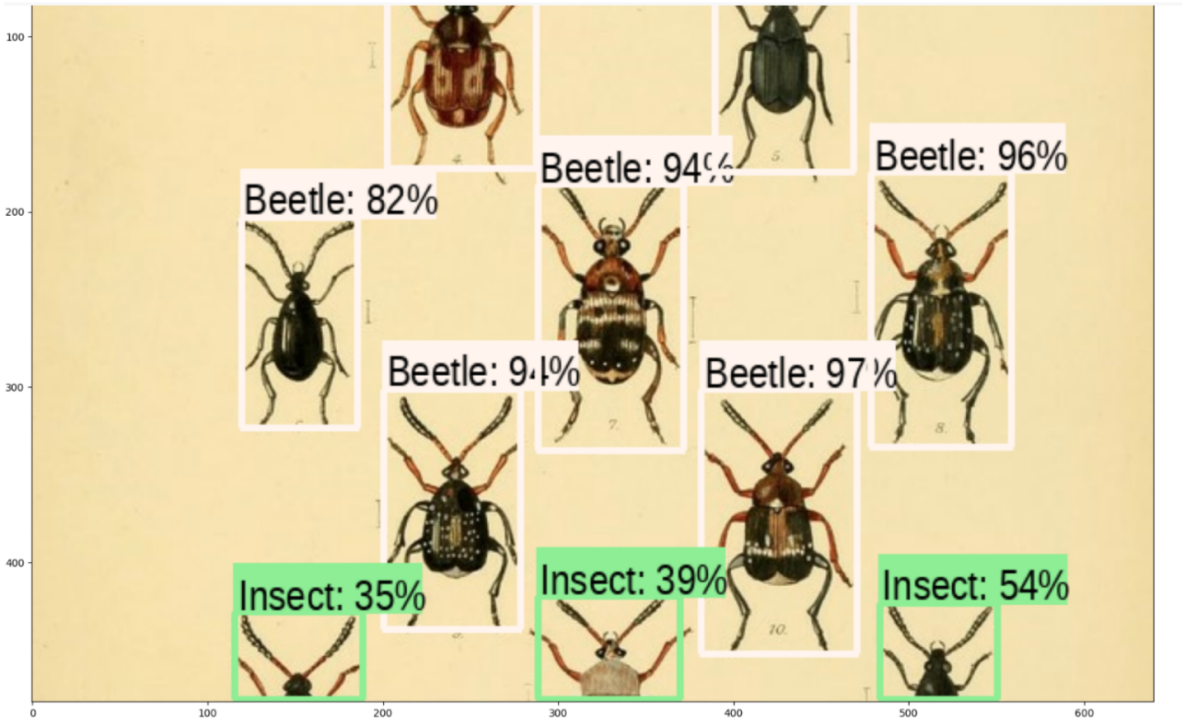

Figure 5.2: Faster R-CNN result 2

In the third image the model had some issues with misclassifications of the types of phones.

Figure 5.3: Faster R-CNN result 3

In the fourth image there seemed to be some confusion in the distinction between birds and animals in general.

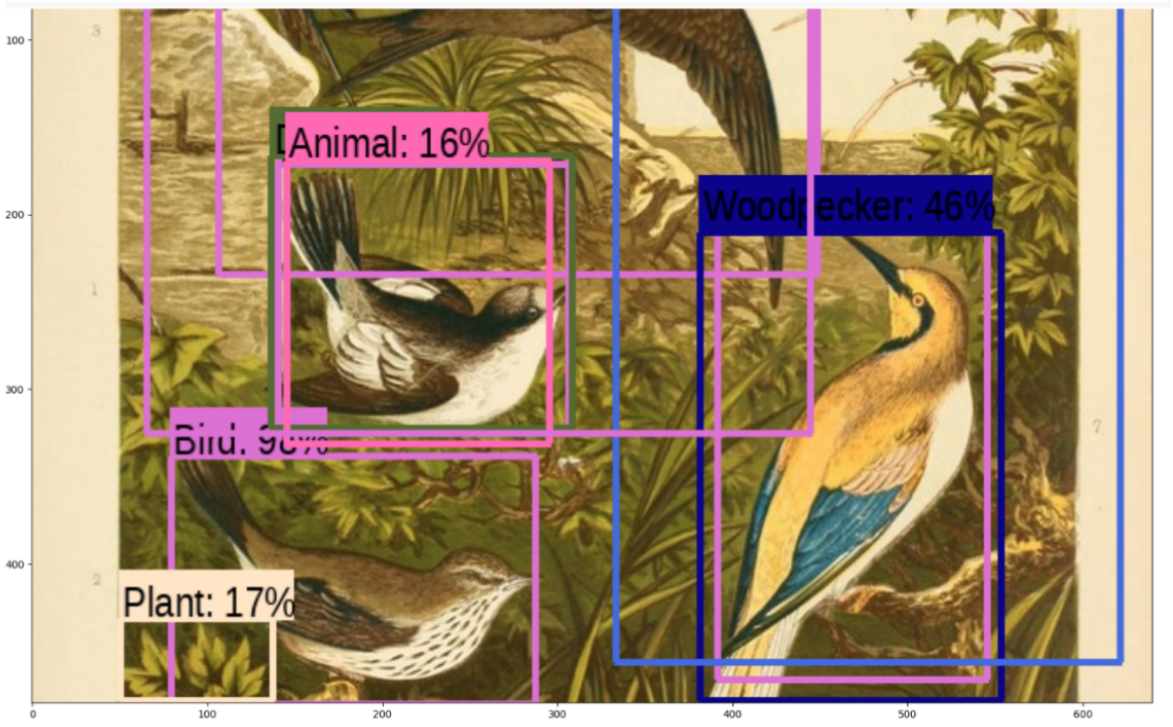

Figure 5.4: Faster R-CNN result 4

5.2 Mask R-CNN

Images 5-7 are the results from the Mask R-CNN model that was trained from the COCO dataset. Based on the results this model seems to perform much better than the Faster R-CNN model.

In image five, Mask R-CNN very accurately detected the giraffe and two zebras.

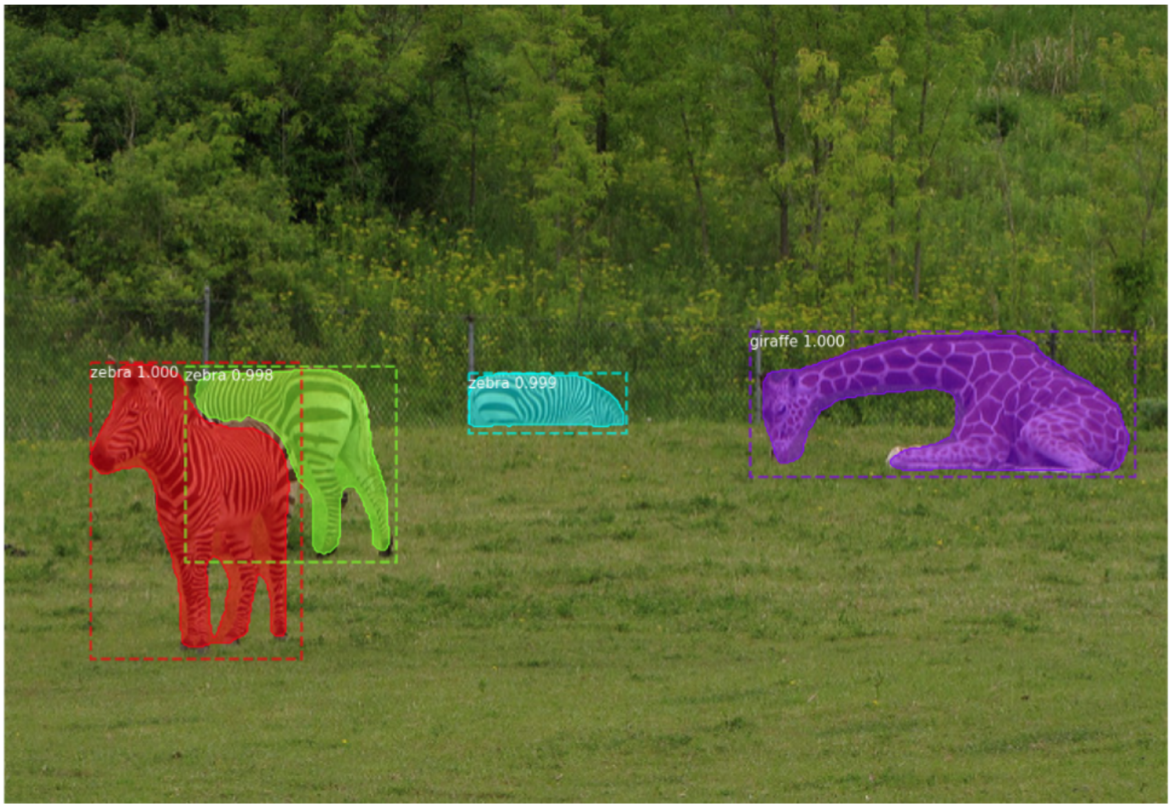

Figure 5.5: Mask R-CNN result 1

In image six, the model was able to clearly detect the objects with a high mean average precision.

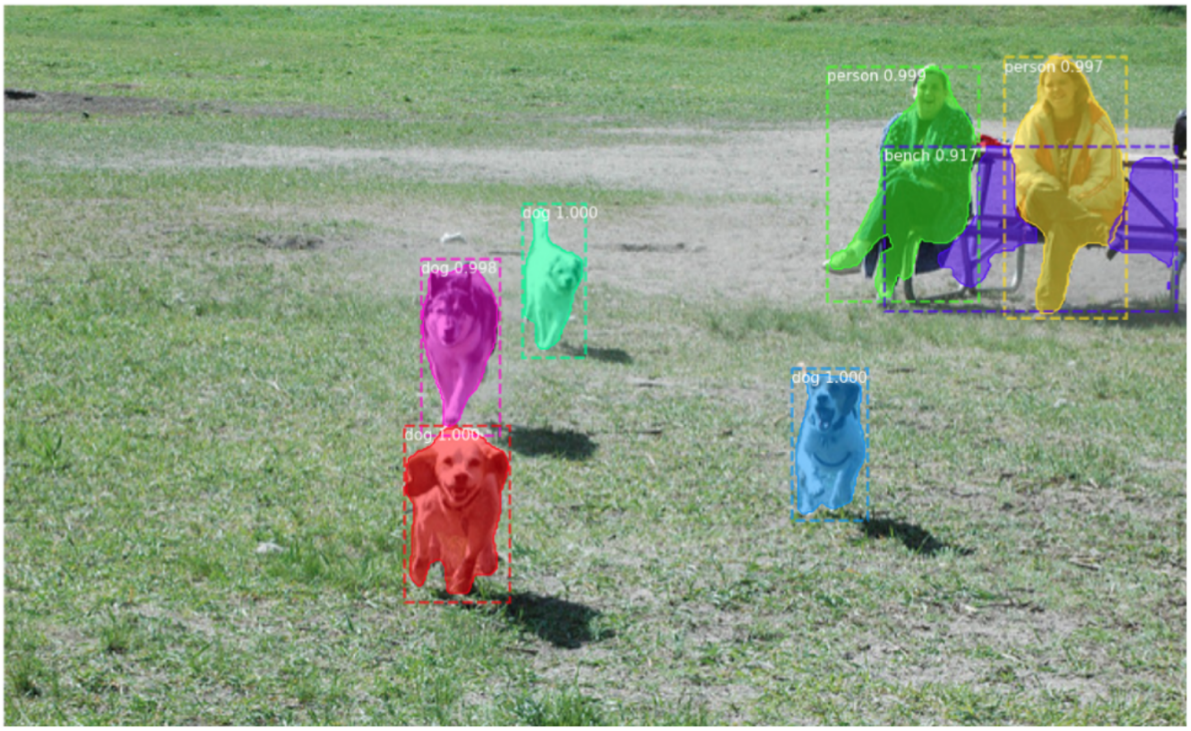

Figure 5.6: Mask R-CNN result 2

Image seven comes from a video that was fed to the Mask R-CNN model. In this frame, the model did have some trouble identifying all the objects.

Figure 5.7: Mask R-CNN result 3

YOLOv8

The last model used in our analysis was YOLOv8. This model also seemed to perform better than the Faster R-CNN model. Our YOLOv8 small (YOLOv8s) model was trained on a custom dataset that consisted of tanks and was subsequently tested on test photos of tanks. Images 8-10 show the results of the YOLOv8s model. The model accurately identified all of the tanks.

Figure 5.8: YOLOv8 result 1

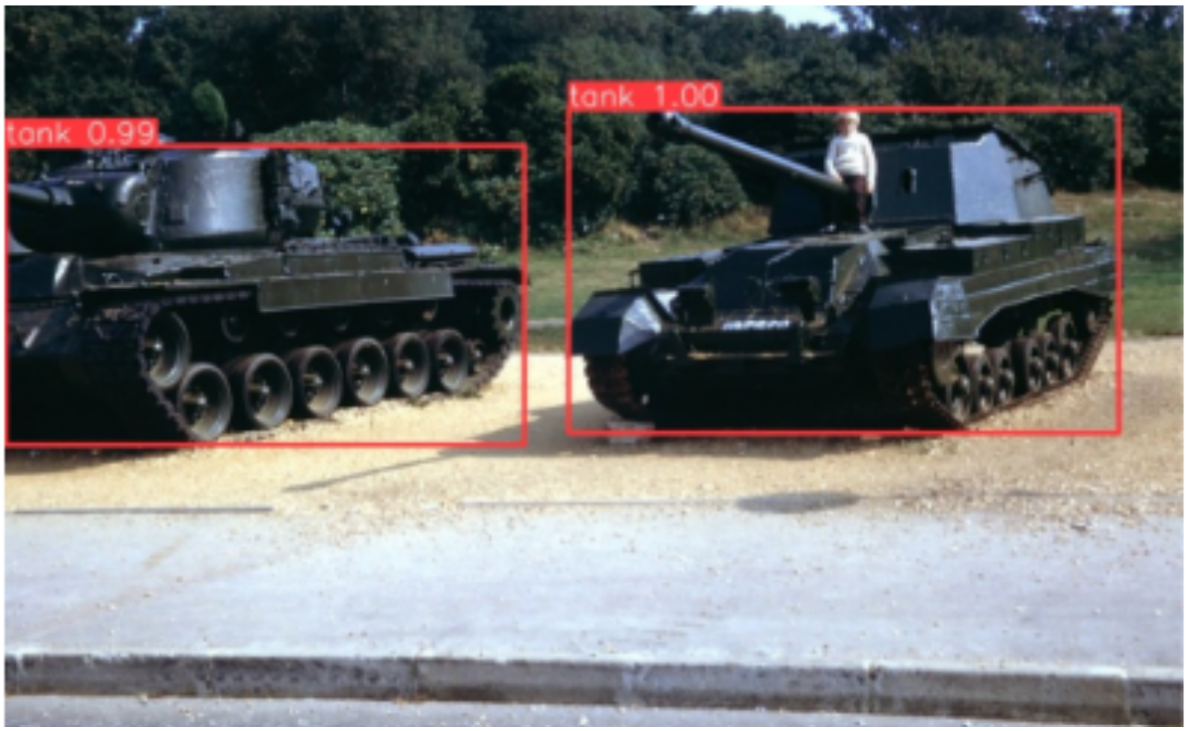

Figure 5.9: YOLOv8 result 2

Figure 5.10: YOLOv8 result 3

6 Conclusion

The goal of this project is to review and test the methodology behind Faster R-CNN, and then evaluate alternative models such as Mask R-CNN and YOLOv8. Our analysis indicates that both Mask R-CNN, and YOLOv8 demonstrate superior performance compared to Faster R-CNN in terms of object detection accuracy and instance precision. Specifically, YOLOv8 proves to be a suitable model for real-time object detection tasks, offering efficient and rapid detection capabilities. On the other hand, Mask R-CNN exhibits higher accuracy in detecting objects within photos and videos, particularly when precise segmentation and detailed understanding of object shapes and boundaries are required. However, it is important to note that Mask R-CNN's performance may be contingent upon the availability of sufficient computational resources, given its more intensive processing requirements compared to YOLOv8.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81

[3] R. Girshick, "Fast r-CNN," in *2015 IEEE international conference on computer vision (ICCV)*, 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-CNN," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017, pp. 2961–2969. doi: 10.1109/ICCV.2017.322

[5] Y. Zhou, X. Wang, and L. Zhang, "Visual identification and pose estimation algorithm of nut tightening robot system." Feb. 2022. doi: 10.21203/rs.3.rs-1391065/v1

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proceedings of the thirty-first AAAI conference on artificial intelligence (AAAI-17)*, 2017, pp. 4278–4284.

A Contributions

Group Contribution by each member.