

Computer Vision: Exploring Object Detection and Segmentation using Faster R-CNN, Mask R-CNN, and YOLOv8

A Comparative Analysis

Sai Devarashetty, Software Engineer

Hector R. Gavilanes, Project Lead

Jacob Knight, Deep Learning Architect

Kyle Knuth, Data Scientist

2024-02-16

Table of contents

Abstract	5
RCNN	6
Glossary	7
Summary	8
1 Introduction	9
2 Literature Review	10
2.1 R-CNN (Region-based Convolutional Network)	10
2.1.1 Region Proposal Generation	10
2.1.2 Feature Extraction	10
2.1.3 Selective Search	11
2.1.4 Fine-tuning and Classification	11
2.1.5 Bounding Box Regression	11
2.1.6 Non-Maximum Suppression	11
2.1.7 Drawbacks	11
2.2 Fast R-CNN	13
2.2.1 Architecture	13
2.2.2 RoI Pooling Layer	15
2.2.3 Advantages of Fast R-CNN	15
2.3 Faster R-CNN	16
2.3.1 Region Proposal Network (RPN)	16
2.3.2 Anchor Boxes	17
2.3.3 Region of Interest (RoI)	17
2.3.4 Feature Pyramid Network (FPN)	17
2.3.5 Classifier and Bounding Box Regressor	18
2.3.6 Non-Maximum Suppression (NMS)	18
2.3.7 Output	18
2.3.8 Advantages of Faster R-CNN	18
2.4 Mask R-CNN	20
2.4.1 Backbone CNN	20
2.4.2 Region Proposal Network (RPN)	20
2.4.3 Region of interest Align (RoIAlign)	20

2.4.4	Parallel Branches	20
2.4.5	Advantages over Faster R-CNN	21
2.5	YOLO (You Only Look Once)	22
2.5.1	Architecture	22
2.5.2	Anchor-Free Approach	23
2.5.3	Efficient Backbone	23
2.5.4	Improved Object Detection	23
2.5.5	Real-Time Performance	23
2.5.6	Advantages over Mask R-CNN	23
3	Methodologies	24
3.1	Faster R-CNN Training	24
3.1.1	Region Proposal Network (RPN)	24
3.1.2	Multi-task Loss Function	25
3.1.3	Optimization	26
3.2	Mask R-CNN	26
3.2.1	Two-Stage Process	26
3.2.2	Loss Function	27
3.2.3	RoI Align	27
3.3	YOLO (You Only Look Once)	27
3.4	Metrics	29
4	Analysis	30
5	Results	32
5.1	Faster R-CNN	32
5.2	Mask R-CNN	36
5.3	YOLOv8	39
6	Video	42
7	Conclusion	43
7.1	Future Studies	43
References		44
Appendices		47
A	YOLOv9	47
A.1	Information Bottleneck	47
A.1.1	Addresses Data Loss	47
A.1.2	Reversible Function	48
A.2	Programmable Gradient Information (PGI)	48

A.3	Reversible Branch	48
A.4	Generalized ELAN (GELAN)	49
B	Contributions	51
B.0.1	Sai Devarashetty, Software Engineer	51
B.0.2	Hector Gavilanes, Project Lead	51
B.0.3	Jacob Knight, Deep Learning Architect	52
B.0.4	Kyle Knuth, Data Scientist	52
C	Resources	53
C.1	Repositories	53
C.2	Web Applications	53

Abstract

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 custom class label was applied to detect military tanks using a YOLOv8 model.

RCNN



Glossary

Term	Definition
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
COCO	Microsoft Common Objects in Context
CPU	Central Processing Unit
DIoU	Distance Intersection over Union
DFL	Distribution Focal Loss
FC	Fully Connected Layer
FCN	Fully Convolutional Network
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
IoU	Intersection over Union
mAP	mean Average Precision
NMS	Non-Maximum Suppression
ReLU	Rectified Linear Unit
RoI	Region of Interest
RoIAlign	Region of Interest Align
R-CNN	Region-based Convolutional Neural Network
RPN	Region Proposal Network
SAM	Segment Anything Model
SVM	Support Vector Machine
YOLO	You Only Look Once

Summary

The faster R-CNN approach significantly enhances object recognition. It meets the requirement for high precision near real-time detection. The Regional Pattern Network (RPN), Region of Interest (RoI) Network, Backbone Convolutional Neural Network (CNN), and Training/Pattern Framework are the primary components of the rapid R-CNN technique.

Proposal Network (RPN) is a significant advancement in Faster R-CNN technology. The convolutional feature map is used to efficiently produce regional dimensions. This leads to a considerably more versatile, end-to-end trainable system that does not require external recommendation techniques like selective search.

Upon obtaining the local parameters, identically sized objects are extracted from the feature map for every local parameter using the ROI pooling layer. Then, bounding box return and classification are performed using these features. By using this method, the model may concentrate on the areas of the picture that are most important, which increases accuracy and speed.

To extract information from the visual input, the spinal cord's Convolutional Neural Network (CNN) is employed. The most widely used options for spinal networks are ResNet and VGG-16. The model gains from transfer learning, and is able to extract well-formed features from the input photos by utilizing a pre-trained CNN as its foundation.

Using a multitasking loss function that combines classification loss with bounding box regression loss, the model is trained from beginning to end. As a result, the model may learn to predict bounding boxes for objects, and classify them concurrently. The computationally trained model is appropriate for low latency applications since it may be utilized for real-time object detection.

The extensively utilized Faster R-CNN technique has served as the foundation for numerous extremely sophisticated object recognition systems. It has positively contributed to different fields including, but limited to autonomous driving, robotics, medical diagnostic, manufacturing, and video surveillance.

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) [2], 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 3.9 seconds per class per image [3] when implemented on the CPU, which is considered slow for many real-world applications. On the other hand, Edge Boxes, another region proposal method, offers improved speed, clocking in at 0.25 seconds per image [4]. 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.

Nonetheless, to mitigate this challenge, the introduction of Region Proposal Networks (RPNs) within the Faster R-CNN framework in 2015 marks a pivotal innovation. By integrating the region proposal step directly into the object detection algorithm, RPNs aim to streamline the process, optimizing resource utilization, and enhancing overall efficiency. This architectural advancement underscores the ongoing efforts to strike a balance between accuracy and speed, heralding a promising future for real-time object detection in diverse applications.

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 [5]. 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 Selective Search

Selective search employs a Hierarchical Grouping algorithm to detect objects with diverse characteristics such as scales, colors, textures, and enclosures. Initially, the method utilizes the Felzenszwalb and Huttenlocher [6] approach to segment the image into regions and obtain a set of initial regions. The grouping algorithm is detailed in the provided Algorithm.

2.1.4 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) [5] for this classification task. Each SVM was trained to discern whether the feature corresponded to a particular object or background.

2.1.5 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.

2.1.6 Non-Maximum Suppression

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

2.1.7 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.

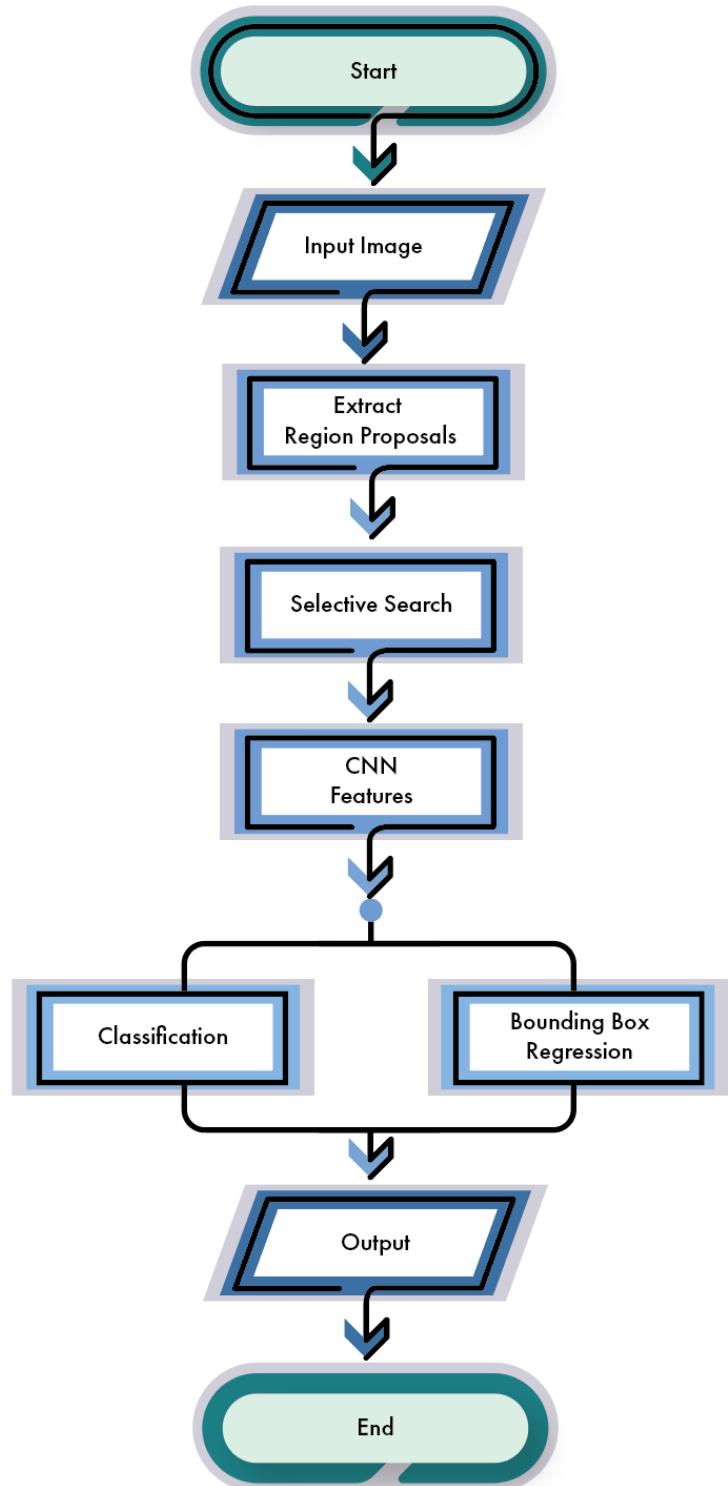


Figure 2.1: Flowchart of R-CNN

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 [7]. 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 [7].

Fast R-CNN tackles the computational inefficiencies of R-CNN by introducing a unified 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 [7], 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.

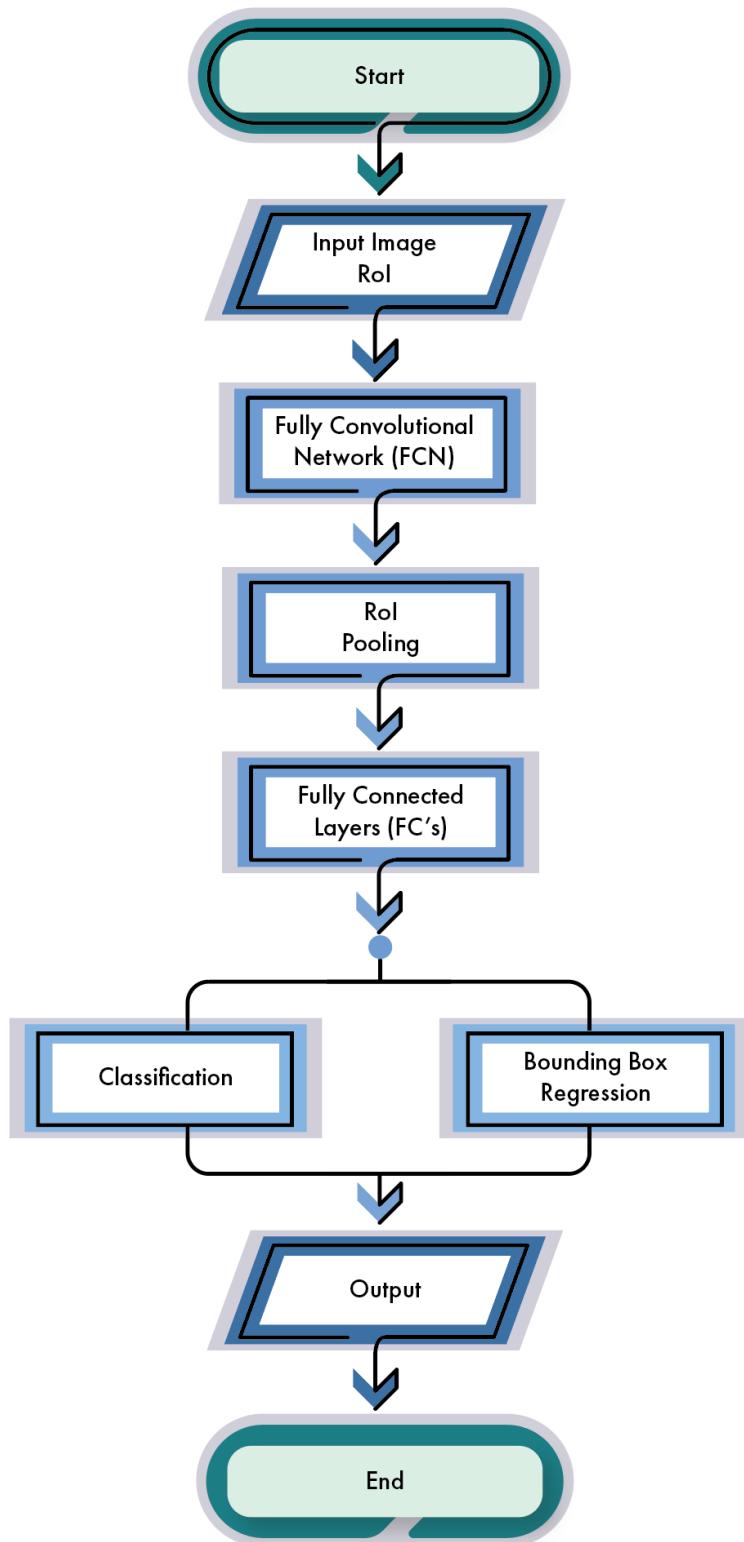


Figure 2.2: Flowchart of Fast R-CNN

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., 7×7) [7], 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.

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.

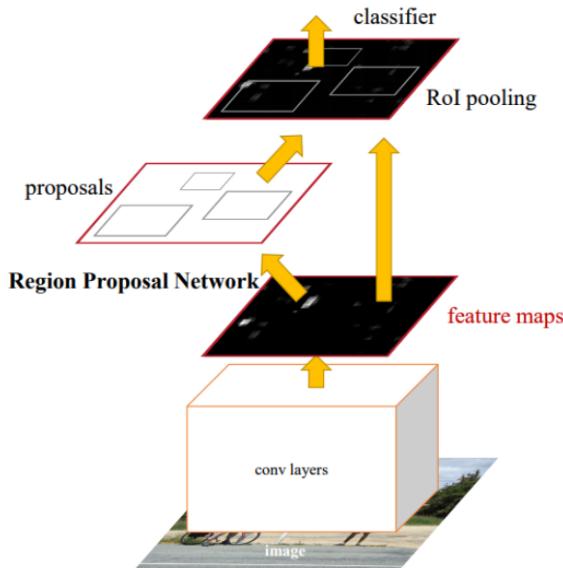


Figure 2.3: Faster R-CNN different stages [1]

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 Feature Pyramid Network (FPN)

Feature Pyramid Network (FPN) stands as a pivotal advancement in object detection methodologies, offering a robust solution for handling objects of varying scales and complexities. FPN, introduced by Facebook AI Research (FAIR), Cornell University, and Cornell Tech, integrates a pyramidal feature hierarchy that combines low-resolution, semantically strong features with high-resolution, semantically weak features through top-down pathways and lateral connections.

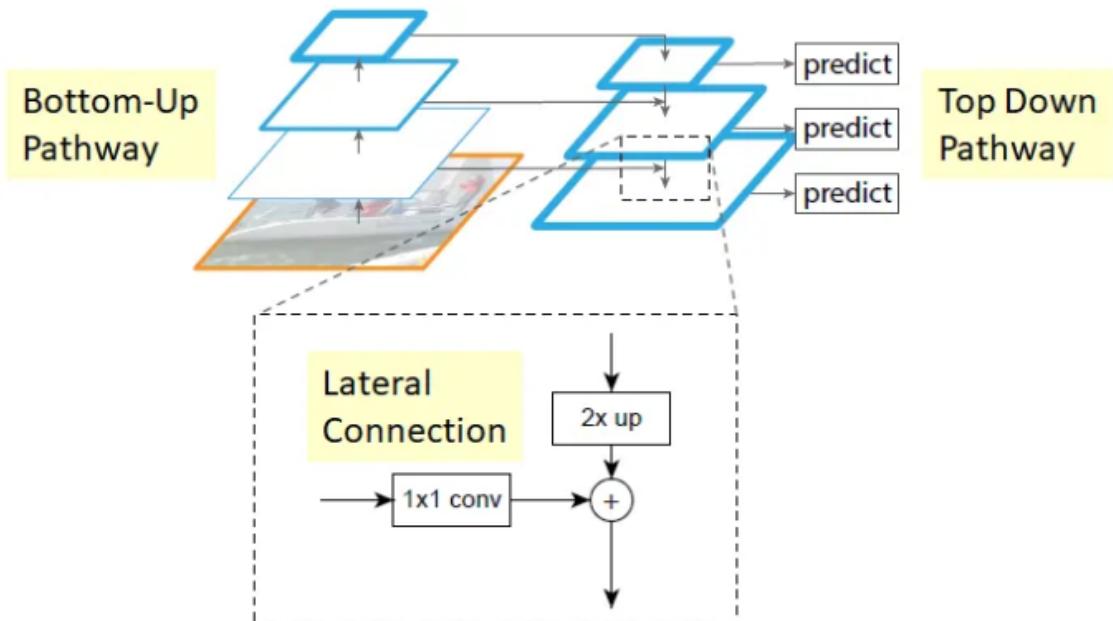


Figure 2.4: Feature Pyramid Network (FPN) [8]

This innovative approach ensures rich semantic information at all levels without compromising speed or memory efficiency. The FPN architecture has demonstrated superior performance in region proposal networks (RPN) and detection networks, showcasing remarkable improvements in detecting small objects and enhancing robustness to scale variations.

2.3.5 Classifier and Bounding Box Regressor

The RoI-pooled 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.

2.3.6 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.

2.3.7 Output

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

2.3.8 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.
- **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.

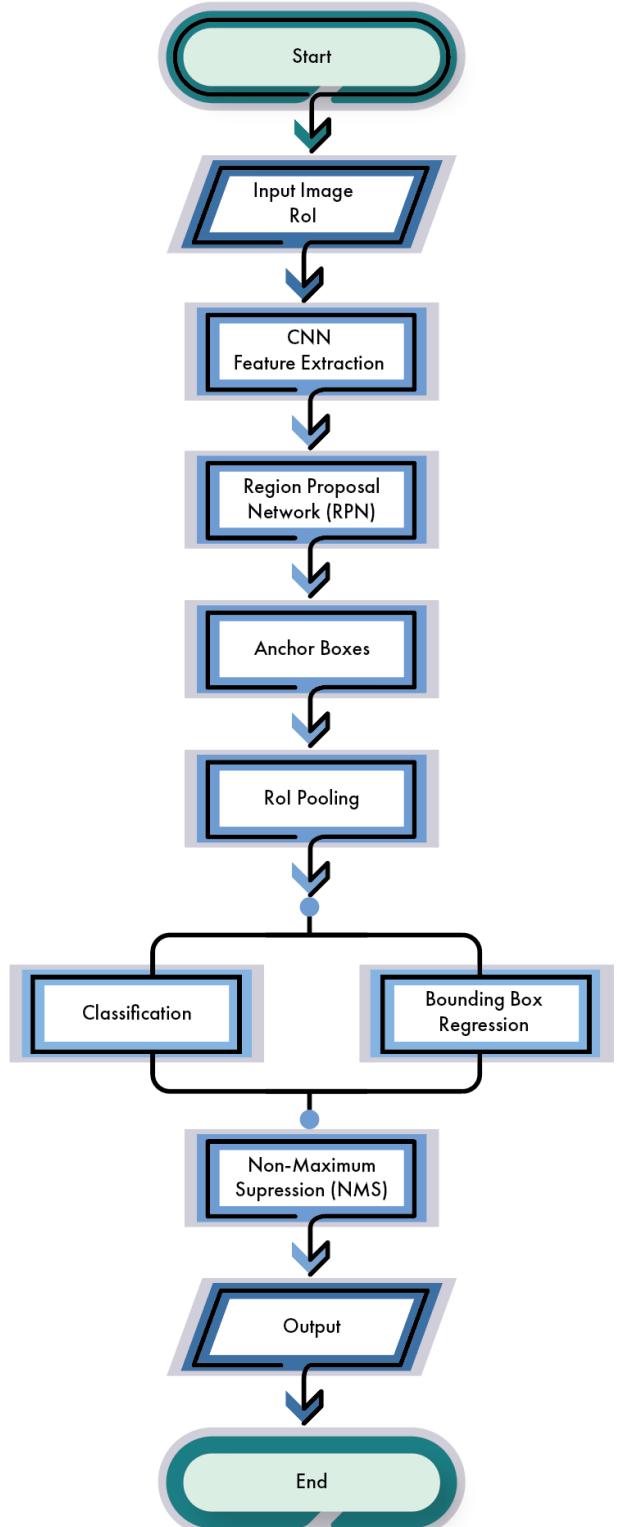


Figure 2.5: Flowchart of Faster R-CNN [1]

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 [9]. 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.

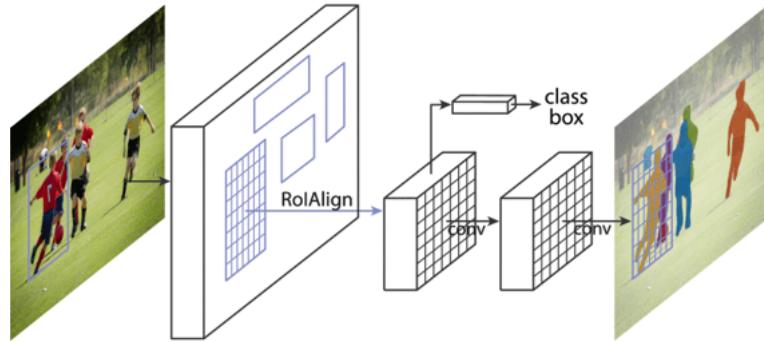


Figure 2.6: Mask R-CNN framework with RoIAlign [10]

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.

2.5 YOLO (You Only Look Once)

YOLO (You Only Look Once) is a popular object detection algorithm that revolutionized the field of computer vision due to its speed and accuracy [11]. YOLO processes images in a single pass through a neural network, enabling real-time object detection. YOLOv8 is one of the iterations of the YOLO algorithm, incorporating various improvements over its predecessors to enhance performance and efficiency.

2.5.1 Architecture

The architecture of YOLO consists of a backbone network, detection head, and output layers. The backbone network, typically based on Darknet or CSPDarknet, extracts features from the input image [12]. These features are then processed by the detection head, which predicts bounding boxes, confidence scores, and class probabilities for detected objects. Earlier YOLO models utilize anchor boxes [12] to improve localization accuracy and efficiency. Finally, the output layers produce the final detection and confidence scores.

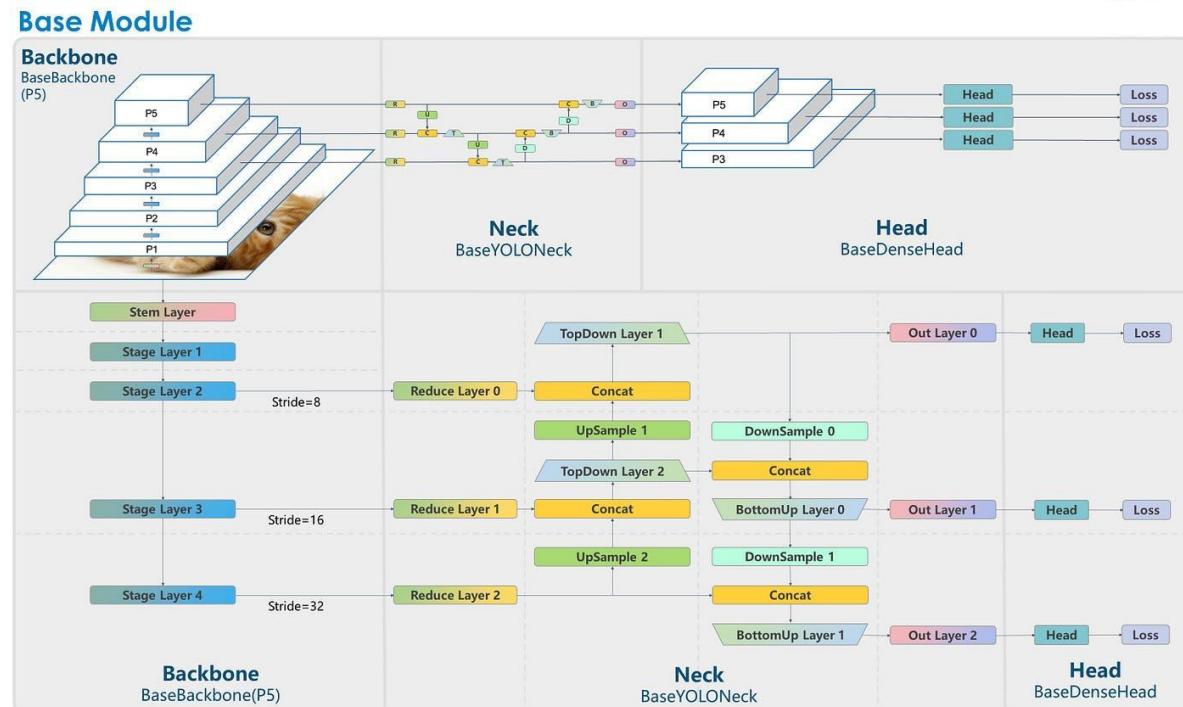


Figure 2.7: YOLOv8 architecture [13]

2.5.2 Anchor-Free Approach

YOLOv8 employs an anchor-free approach [2], eliminating the need for predefined anchor boxes. This simplifies the training process and improves model flexibility.

2.5.3 Efficient Backbone

YOLO utilizes lightweight backbone networks such as CSPDarknet53, which strikes a balance between performance and computational efficiency

2.5.4 Improved Object Detection

YOLO incorporates advanced techniques such as focal loss, which helps to address class imbalance and improve object detection accuracy [14].

2.5.5 Real-Time Performance

YOLOv8 is optimized for real-time object detection applications [2], offering fast inference speeds without compromising accuracy.

2.5.6 Advantages over Mask R-CNN

Both Mask R-CNN and YOLOv8 are instance segmentation models; however, Mask R-CNN is better suited for semantic segmentation while YOLOv8 is not initially designed for that purpose. Both models are pre-trained on the MS-COCO dataset. Mask R-CNN is an improved version of Faster R-CNN and uses ResNet-101 as its backbone and has the addition of the prediction of object masks. YOLOv8 uses CSPDarknet53 as its backbone, which has 53 convolutional layers.

The main difference between YOLOv8 and Mask R-CNN is the region proposal network. YOLOv8 is an anchor free model so it does not need a region proposal network. Another difference between the two models is that Mask R-CNN is a two stage model and YOLOv8 is a one stage model. The first stage of Mask R-CNN is the application of the region proposal network and the second stage executes “bounding box regression, classification, and mask prediction” [15].

On the other hand, YOLOv8 has one stage that consists of “directly predicting bounding boxes and class labels for objects in an image” [15]. YOLOv8 and Mask R-CNN also differ in their loss functions. YOLOv8 uses complete intersection over union loss, distribution focal loss, and binary cross entropy. Mask R-CNN uses classification loss, regression loss, and mask loss. In addition YOLOv8 is best used for real time object detection with low latency [15].

3 Methodologies

3.1 Faster R-CNN Training

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 [7]. This joint training approach allows Faster R-CNN to produce high-quality region proposals and accurately classify and localize objects in images.

3.1.1 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.

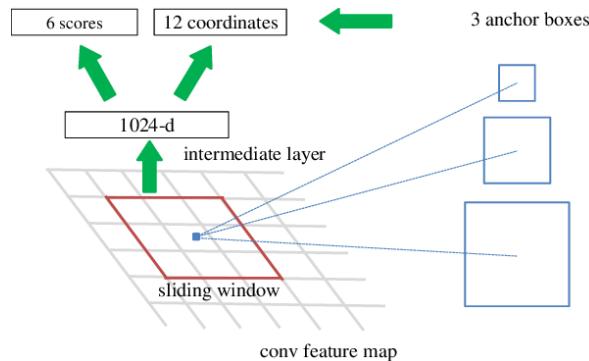


Figure 3.1: Improved anchor boxes in Faster R-CNN [16]

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 [16].

3.1.2 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. t_i 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 L_{cls} is log loss over two classes (object vs. not object). For the regression loss, we use $L_{reg}(t_i, t * i) = R(t_i - t * i)$ where R is the robust loss function (smooth L1). The term $p * i L_{reg}$ 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 $\{p_i\}$ and $\{t_i\}$ respectively. The two terms are normalized with N_{cls} and N_{reg} , 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,
- $\Delta x, \Delta y, \Delta w$, and Δh are the parameterized adjustments to the coordinates of the default anchor box to obtain the predicted bounding box coordinates.

3.1.3 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 [7], “In words, for each mini-batch RoI r and each pooling output unit y_{rj} , the partial derivative L / y_{rj} is accumulated if i is the argmax selected for y_{rj} by max pooling. In back-propagation, the partial derivatives L / y_{rj} 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.

3.2 Mask R-CNN

Mask R-CNN is a framework for object instance segmentation that efficiently detects objects in an image while generating high-quality segmentation masks for each instance. This method extends Faster R-CNN by incorporating a branch for predicting object masks alongside the existing branch for bounding box recognition [9].

3.2.1 Two-Stage Process

Mask R-CNN utilizes a two-stage target detection method. The first stage involves obtaining proposals through the Region Proposal Network (RPN), while the second stage processes features of target locations based on the proposals and regions of interest generated in the first stage, performing classification, localization, and mask prediction [17].

Feature Extraction Network: The feature extraction network of Mask R-CNN consists of bottom-up and top-down paths. The bottom-up path, composed of residual structures of different sizes like ResNet-101, is responsible for image feature extraction [17].

3.2.2 Loss Function

The loss function in Mask R-CNN includes components for class label, bounding box, and mask. L_{class} is a classification loss, while L_{mask} uses binary cross-entropy loss for mask prediction [9].

The components are as follow:

- L_{class} = Classification loss to determine if an object exists.
- L_{mask} = Binary cross-entropy loss used for mask prediction.

3.2.3 RoI Align

RoI Align, short for Region of Interest Align, is a technique used in object detection and segmentation tasks to extract a small feature map from each Region of Interest (RoI) with improved accuracy compared to RoI Pooling. RoI Align overcomes the quantization issues of RoI Pooling by employing bilinear interpolation to compute precise values at four regularly sampled locations within each RoI bin, ensuring better alignment with the input features [9], [18].

- **Bilinear Interpolation:** RoI Align utilizes bilinear interpolation to calculate exact feature values at four regularly sampled locations within each RoI bin, enhancing the alignment of extracted features with the input data [18].
- **Quantization Improvement:** Unlike RoI Pooling, which suffers from quantization errors, RoI Align avoids quantization issues by using bilinear interpolation to maintain accurate spatial information during feature extraction [18].

The RoI Align operation involves computing the exact values of input features at four regularly sampled locations within each RoI bin using bilinear interpolation [18]. This precise alignment enhances the accuracy of feature extraction for each region proposal

3.3 YOLO (You Only Look Once)

YOLOv8 employs a cross-stage partial bottleneck with two convolutions, which is called C2f module. [19] Using YOLOv7's ELAN structure, YOLOv8 uses one standard convolutional layer and maximizes the Bottleneck module to enhance the gradient branch. [20] This keeps YOLOv8 lightweight while also capturing more gradient flow information.

YOLOv8 incorporates an anchor-free model [20] with a decoupled head. This allows for independent processing of objectness, classification, and regression tasks. The anchor-free model

[19] design utilizes multiple branches, each of which focus on a single task. This is vital in improving the accuracy of the model.

YOLOv8 uses a sigmoid activation function as well as a softmax function for class probabilities. The loss functions used for bounding box loss and binary cross-entropy are the Complete IoU Loss, and the Distribution Focal Loss (DFL). Complete IoU uses Distance-IoU while taking aspect ratios into consideration [19]. Firstly, DIoU is IoU when trying to minimize the normalized distance between central points of two bounding boxes [21].

When taking aspect ratios into account, the CIoU loss function is defined as:

$$CIoU_{loss} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + av$$

Alpha here is defined as:

$$\alpha = \frac{v}{(1 - IoU) + v}$$

The optimization function for CIoU [21] is the following partial derivatives:

$$CIoU_{loss} = IoU_{loss} - v \cdot \left(\frac{d(x, y)}{c} \right) - \arctan \left(\frac{w_a}{h_a} - \frac{w_b}{h_b} \right) + v \cdot \left(\frac{C}{c^2} \right)$$

$$\frac{\partial CIoU_{loss}}{\partial x} = \frac{\partial IoU_{loss}}{\partial x} - v \cdot \frac{\partial}{\partial x} \left(\frac{d(x, y)}{c} \right) - \arctan \left(\frac{w_a}{h_a} - \frac{w_b}{h_b} \right) + v \cdot \frac{\partial}{\partial x} \left(\frac{C}{c^2} \right)$$

Distribution Focal Loss originates from Focal Loss [22], which was initially developed for one-stage object detection tasks. These tasks commonly encounter a significant class imbalance between foreground and background classes throughout the training process.

The Focal Loss function is defined as:

$$FL(p_t) = -(1 - p_t)^\gamma \cdot \log(p_t)$$

$$p_t = \begin{cases} p & \text{when } y = 1 \\ 1 - p & \text{when } y = 0 \end{cases}$$

Li [22] describes Distribution Focal Loss (DFL) as FL that “forces the network to rapidly focus on the values near label y , by explicitly enlarging the probabilities of y_i and y_{i+1} (nearest two to y , $y_i \approx y_{i+1}$).”

The DFL function between two stages is defined below:

$$\text{DFL}(S_i, S_{i+1}) = -(y_{i+1}) \log(S_i) + (y - y_i) \log(S_{i+1})$$

3.4 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 AP_k 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 [23] 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.

In addition to Faster R-CNN, Mask R-CNN was employed, utilizing a pre-trained 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, concluded in the selection of the small model, which resulted in smooth execution.

The following flowchart illustrates our process details for the custom military tank detector.

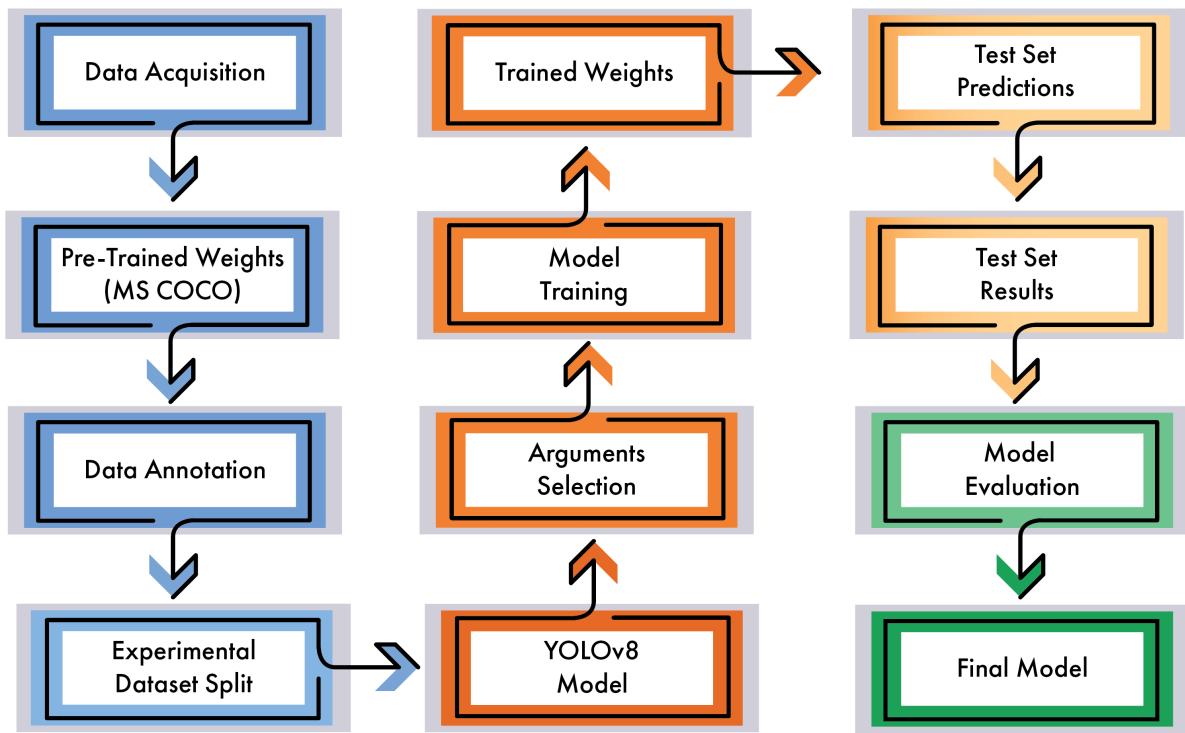


Figure 4.1: Custom Object Detector using YOLOv8

5 Results

5.1 Faster R-CNN

We used the Faster R-CNN model from TensorFlow Hub (Inception-ResNet-V2). According to the authors [24], 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.

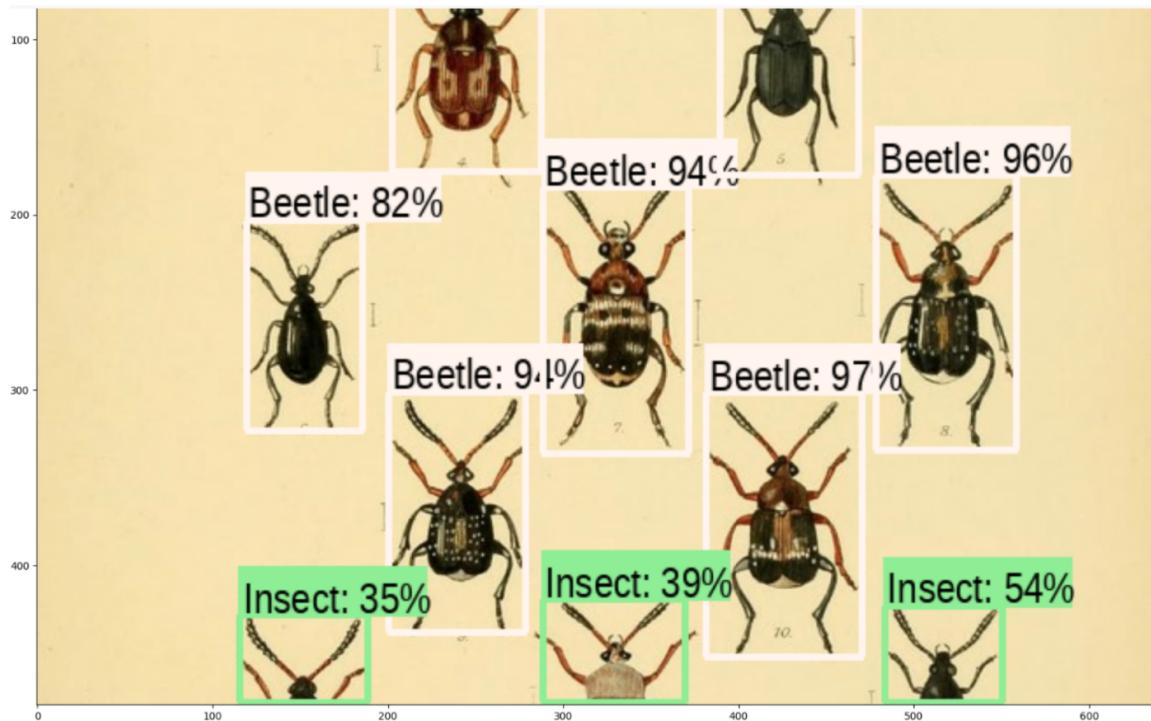


Figure 5.2: Faster R-CNN result 2

In the third image the model was able to distinguish different 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.

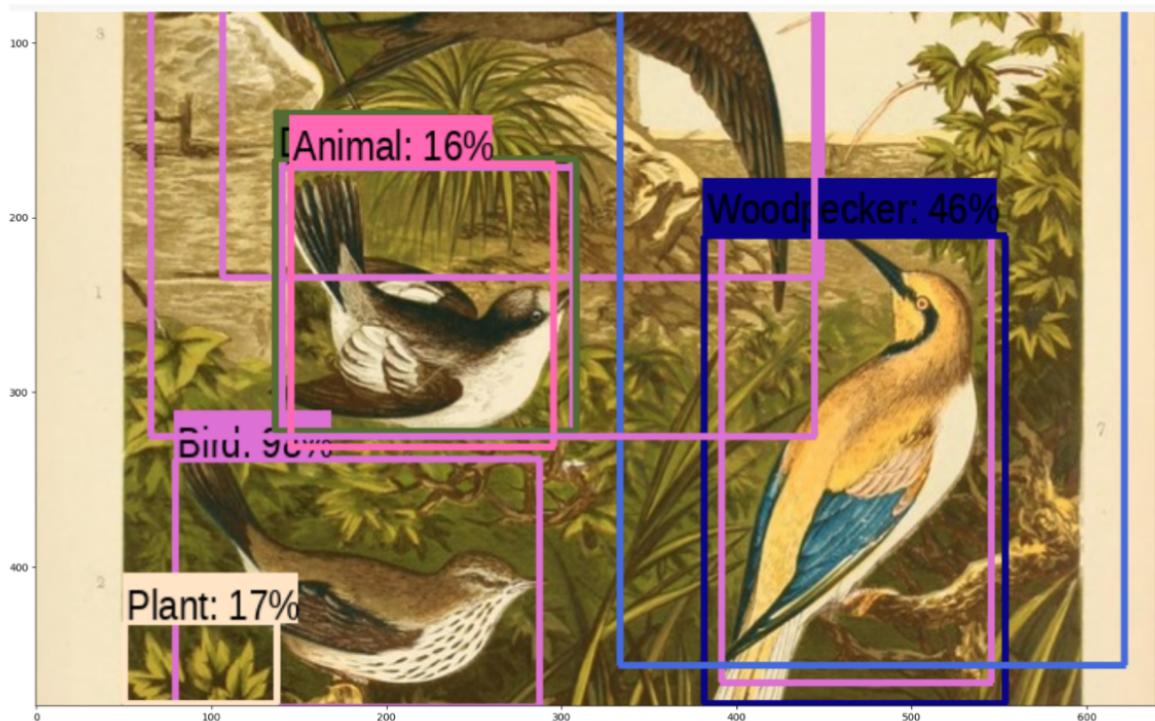


Figure 5.4: Faster R-CNN result 4

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.

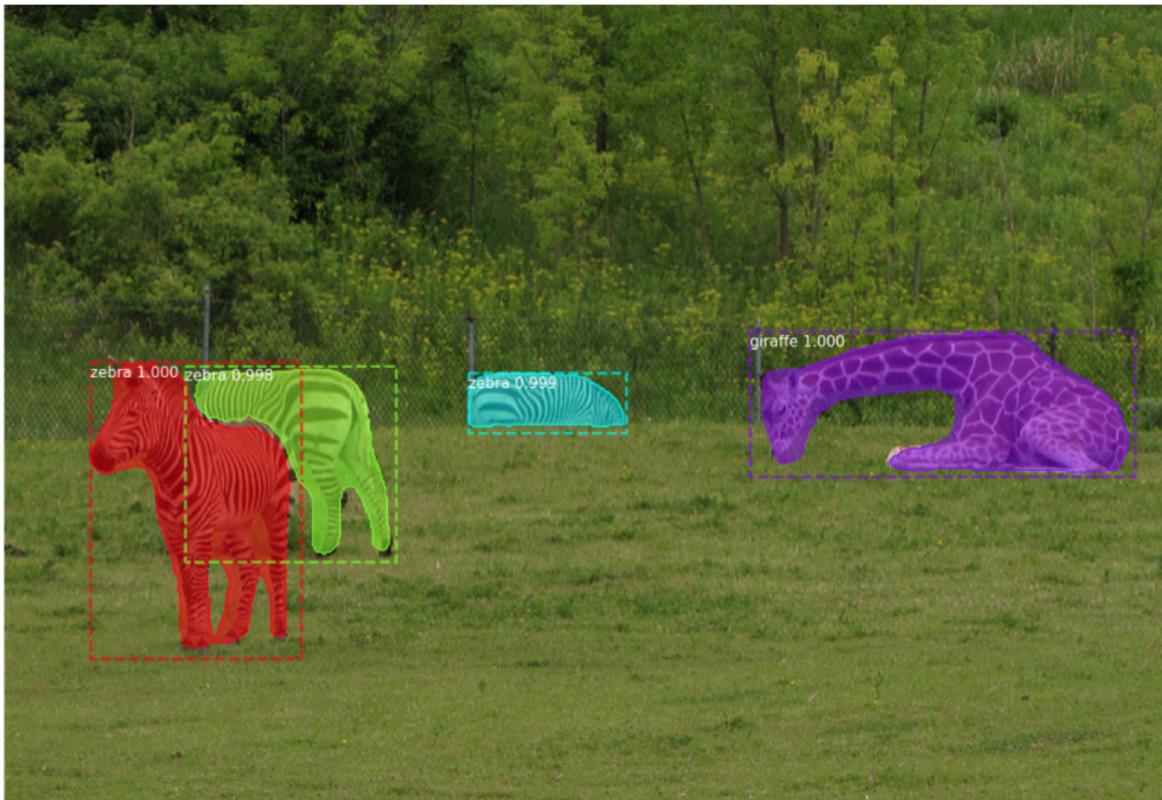


Figure 5.5: Mask R-CNN result 1

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

5.3 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.

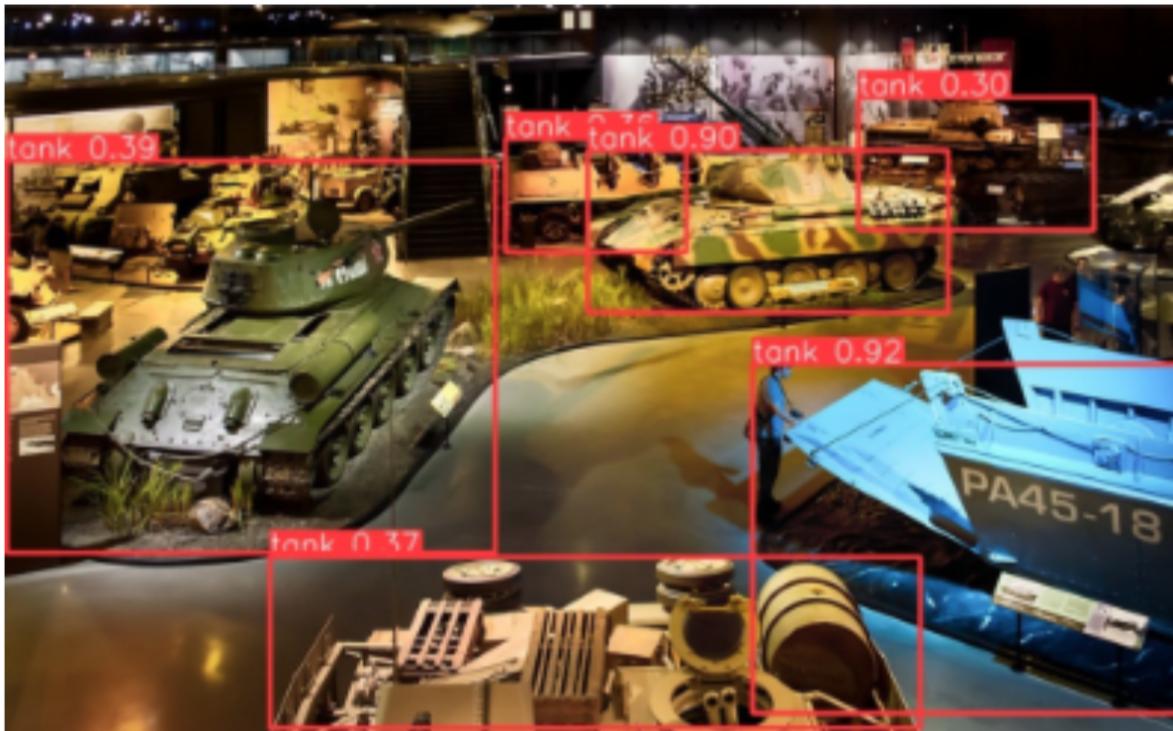


Figure 5.8: YOLOv8 result 1

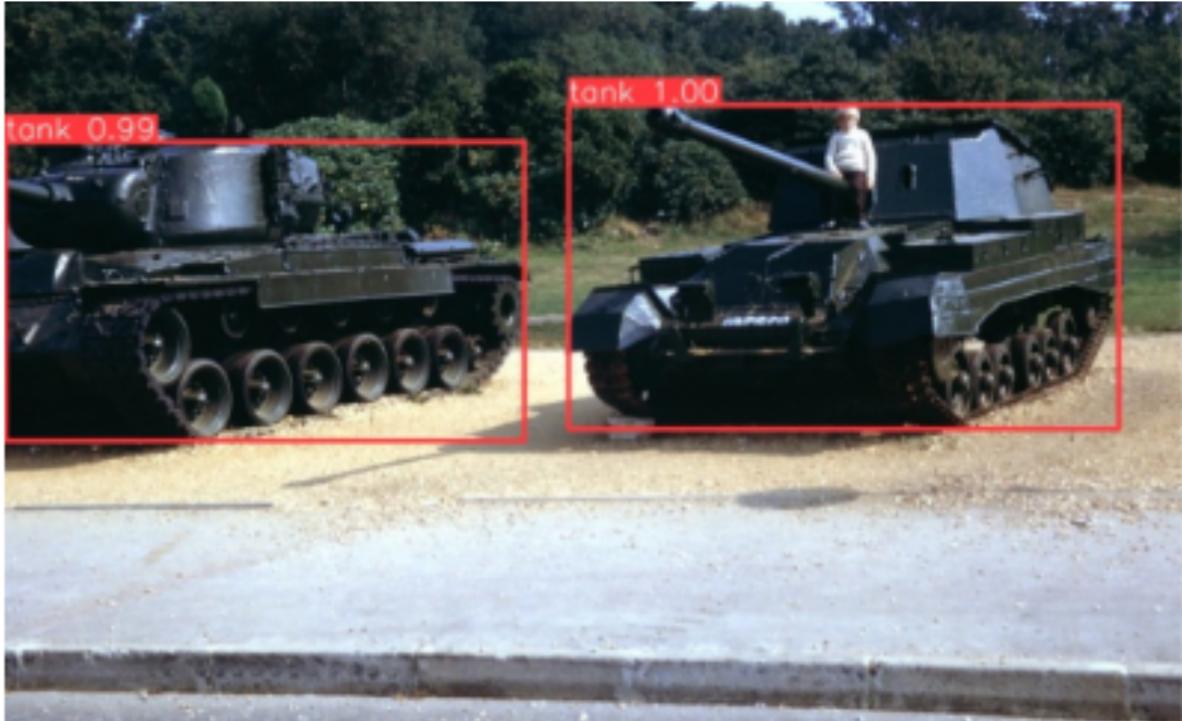


Figure 5.9: YOLOv8 result 2



Figure 5.10: YOLOv8 result 3

The last test was a real-time capture using a webcam using the original YOLOv8 Nano. The pre-trained model was able to detect and label correctly all the class objects in the room.

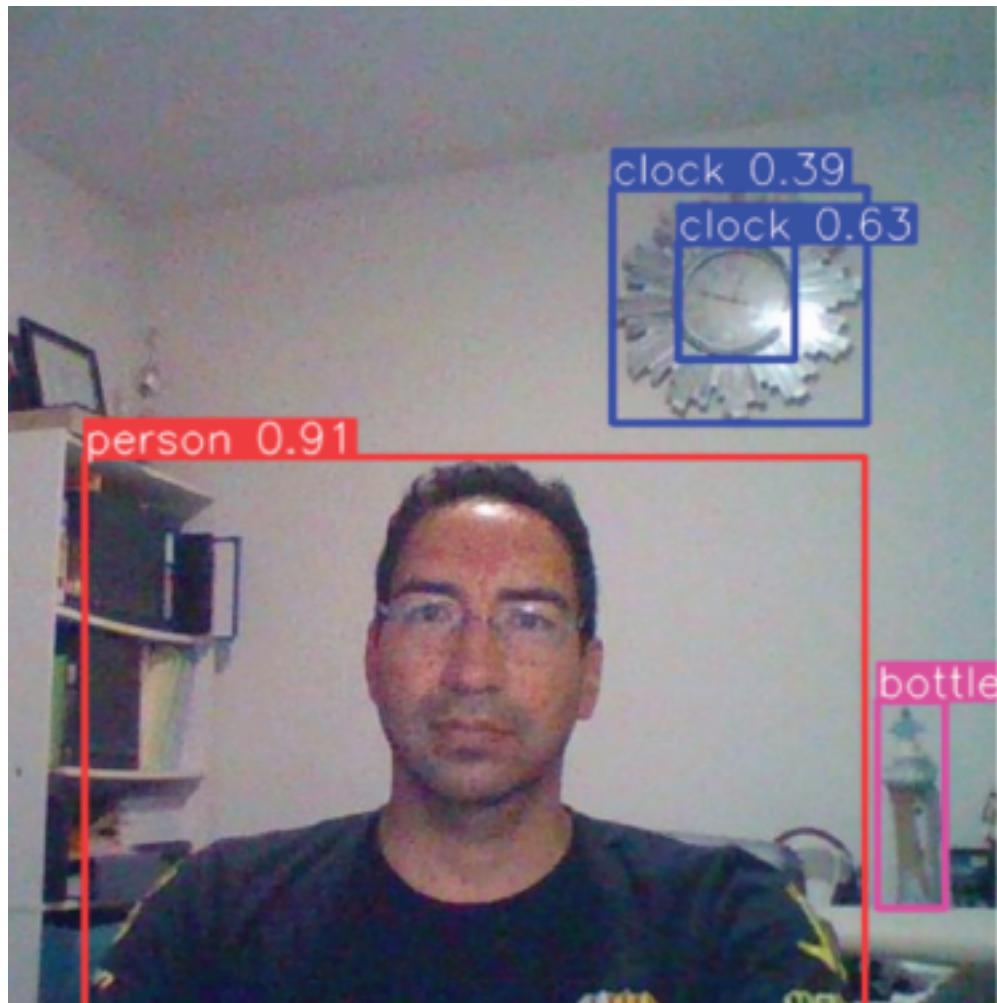


Figure 5.11: Real-Time Object Detection and Classification

6 Video

The video showcases object detection and segmentation utilizing Mask R-CNN and pre-trained YOLOv8 models. Military tanks are categorized as either trucks or cars due to the absence of a specific tank label in the COCO dataset. This initial discrepancy served as the primary motivation to train a model with custom object classes.

<https://youtu.be/-0Tg7233hDE>

7 Conclusion

In conclusion, our aim in this project was to thoroughly examine the Faster R-CNN method, alongside an exploration of alternatives like Mask R-CNN and YOLOv8. Our analysis clearly shows that both Mask R-CNN and YOLOv8 outperform Faster R-CNN in terms of spotting objects accurately, and inference time.

YOLOv8 stands out for its ability to detect objects in real-time situations. On the other hand, Mask R-CNN shines when it comes to accurately identifying objects in images and videos, especially when it requires understanding detailed object shapes and boundaries.

However, it's worth noting that Mask R-CNN's performance might rely heavily on having enough computing power because it needs more processing resources compared to YOLOv8. This highlights the importance of not only considering performance but also practical concerns and limitations when choosing a model for real-world applications.

In conclusion, our research offers insight on the strengths and weaknesses of different object detection methods. By providing practical insights, we believe that YOLOv8 algorithm offers a good trade-off between detection performance, and inference time.

7.1 Future Studies

Expanding upon the current research, future investigations could delve into the utilization of auto-annotated classes generated by SAM to enrich the COCO dataset with a diverse range of object classes. This expansion could provide valuable insights into the performance of the model when confronted with multiple classes, potentially enhancing its overall accuracy and robustness.

Moreover, to further extend the scope of this study, it would be beneficial to implement the YOLOv9 model and compare its performance against its predecessor, YOLOv8. This comparative analysis could reveal advancements and enhancements in real-time inference capabilities attained with the latest version of the YOLO model.

The dynamic field of computer vision provides diverse unexplored paths and untapped research potential. As technology progresses, it becomes increasingly important to refine these methodologies through researchers' contributions to address the evolving demands of various real-world applications.

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] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8.” 2023. Available: <https://github.com/ultralytics/ultralytics>
- [3] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [4] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *European conference on computer vision*, 2014, pp. 391–405.
- [5] 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](https://doi.org/10.1109/CVPR.2014.81)
- [6] Author(s), “Segmentation of images into regions using felzenszwalb and huttenlocher method,” *Journal/Conference*, vol. Volume, no. Number, p. Pages, Publication Year.
- [7] R. Girshick, “Fast r-CNN,” in *2015 IEEE international conference on computer vision (ICCV)*, 2015, pp. 1440–1448. doi: [10.1109/ICCV.2015.169](https://doi.org/10.1109/ICCV.2015.169)
- [8] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2016, Available: <https://api.semanticscholar.org/CorpusID:10716717>
- [9] 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](https://doi.org/10.1109/ICCV.2017.322)
- [10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, Feb. 2020, doi: [10.1109/TPAMI.2018.2844175](https://doi.org/10.1109/TPAMI.2018.2844175)
- [11] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [12] A. Bochkovskiy *et al.*, “YOLOv4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.

- [13] L. Zhang, G. Ding, C. Li, and D. Li, “DCF-Yolov8: An improved algorithm for aggregating low-level features to detect agricultural pests and diseases,” *Agronomy*, vol. 13, p. 2012, Jul. 2023, doi: [10.3390/agronomy13082012](https://doi.org/10.3390/agronomy13082012)
- [14] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: Fully convolutional one-stage object detection,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9627–9636, 2019.
- [15] Z. Ameli, S. J. Nesheli, and E. N. Landis, “Deep learning-based steel bridge corrosion segmentation and condition rating using mask RCNN and YOLOv8,” *Infrastructures*, vol. 9, no. 1, 2024, doi: [10.3390/infrastructures9010003](https://doi.org/10.3390/infrastructures9010003). Available: <https://www.mdpi.com/2412-3811/9/1/3>
- [16] 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](https://doi.org/10.21203/rs.3.rs-1391065/v1)
- [17] S. Fang, B. Zhang, and J. Hu, “Improved mask r-CNN multi-target detection and segmentation for autonomous driving in complex scenes,” *Sensors (Basel)*, vol. 23, no. 8, p. 3853, 2023, doi: [10.3390/s23083853](https://doi.org/10.3390/s23083853)
- [18] K. Erdem, “Understanding region of interest - part 2 (RoI align),” *Journal Name*, 2020.
- [19] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of YOLO architectures in computer vision: From YOLOV1 to Yolov8 and yolonas,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023, doi: [10.3390/make5040083](https://doi.org/10.3390/make5040083)
- [20] X. Wang, H. Gao, Z. Jia, and Z. Li, “BL-Yolov8: An improved road defect detection model based on yolov8,” *Sensors*, vol. 23, no. 20, p. 8361, 2023, doi: [10.3390/s23208361](https://doi.org/10.3390/s23208361)
- [21] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IOU loss: Faster and better learning for bounding box regression,” in *Proceedings of the AAAI conference on artificial intelligence*, 2020, pp. 12993–13000. doi: [10.1609/aaai.v34i07.6999](https://doi.org/10.1609/aaai.v34i07.6999)
- [22] X. Li, W. Wang, X. Hu, J. Li, J. Tang, and J. Yang, “Generalized focal loss V2: Learning reliable localization quality estimation for dense object detection,” in *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2021. doi: [10.1109/cvpr46437.2021.01146](https://doi.org/10.1109/cvpr46437.2021.01146)
- [23] 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.
- [24] 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.
- [25] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “YOLOv9: Learning what you want to learn using programmable gradient information.” 2024. Available: <https://arxiv.org/abs/2402.13616>

- [26] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSP-Net: A new backbone that can enhance learning capability of CNN,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*, 2020, pp. 390–391.
- [27] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, “Designing network design strategies through gradient path analysis,” *Journal of Information Science and Engineering (JISE)*, vol. 39, no. 4, pp. 975–995, 2023.

A YOLOv9

YOLOv9, the latest iteration in the YOLO (You Only Look Once) series of object detection models, published in February 2024, represents a significant advancement in the field of computer vision. Building upon the strengths of predecessors models, Wang and colleagues in the paper titled “*YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*” introduce novel methodologies and enhancements aimed at improving detection accuracy, speed, and efficiency. With a focus on addressing data loss through an information bottleneck, incorporating programmable gradient information for reliable weight updates, and integrating Generalized ELAN (GELAN) [25] for enhanced feature extraction. YOLOv9 stands at the forefront of cutting-edge object detection technology. This introduction sets the stage for exploring the innovations that define YOLOv9 as a state-of-the-art solution for object detection tasks.

A.1 Information Bottleneck

A.1.1 Addresses Data Loss

YOLOv9 tackles the issue of data loss during deep network transmission by introducing the concept of an information bottleneck. This bottleneck aims to minimize the loss of information as data passes through the network layers.

According to information bottleneck principle [25], we know that data X may cause information loss when going through transformation, as shown below:

$$I(X, X) \geq I(X, f_\theta(X)) \geq I(X, g_\phi(f_\theta(X)))$$

The authors stated that as the number of network layer increases, the original data have high probability to be lost.

A.1.2 Reversible Function

Implementing a reversible function helps maintain the integrity of data during processing, ensuring that information is not lost in the network's computations.

When the network utilizes a composition of reversible functions in its transformation process, it leads to the acquisition of more dependable gradients for model updates. The majority of contemporary deep learning techniques adhere to the reversible property in their architectures, ensuring robust gradient information for effective model optimization, such as:

$$X^{l+1} = X^l + f_\theta^{l+1}(X^l)$$

A.2 Programmable Gradient Information (PGI)

Reliable Gradient Information: PGI is introduced to ensure reliable gradient information is available for updating network weights. By providing complete input information for target tasks, PGI facilitates accurate gradient calculations essential for weight updates. **Update Network Weights:** The availability of reliable gradient information through PGI enables effective updating of network weights, enhancing the learning process and model performance. Auxiliary

A.3 Reversible Branch

YOLOv9 incorporates an auxiliary reversible branch to further optimize gradient information flow and improve the overall training process.

This branch introduces a novel approach that allows for reversible network operations, enabling efficient training and inference processes. By incorporating reversible layers, YOLOv9 enhances model interpretability, reduces memory consumption, and improves training speed without compromising accuracy.

A.3.0.1 Key Features of YOLOv9 Reversible Branch:

1. **Reversible Layers:** YOLOv9 integrates reversible layers that enable the network to store intermediate activations in a reversible manner, reducing memory overhead during training and inference.
2. **Improved Efficiency:** The reversible branch architecture enhances computational efficiency by eliminating the need to store intermediate activations, leading to faster training and inference times.

3. **Enhanced Interpretability:** The reversible operations in YOLOv9 facilitate better interpretability of the model by allowing the reconstruction of intermediate activations during inference.
4. **Memory Optimization:** By utilizing reversible layers, YOLOv9 optimizes memory consumption during training, making it more feasible to train deep neural networks on limited hardware resources.
5. **Maintained Accuracy:** Despite the efficiency improvements, YOLOv9’s reversible branch maintains high accuracy levels comparable to traditional YOLO architectures.

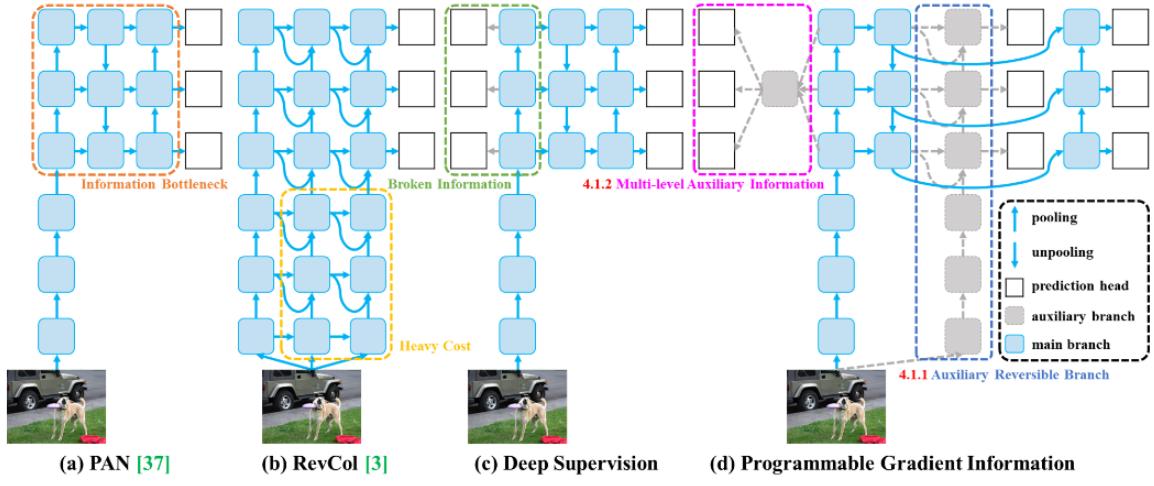


Figure A.1: Multi-Level Auxiliary Information

A.4 Generalized ELAN (GELAN)

CSPNet + ELAN: GELAN combines CSPNet with ELAN to enhance feature extraction and improve model efficiency. This fusion allows for more effective utilization of parameters and computational resources within the network architecture. Gradient Path Planning: GELAN’s architecture focuses on gradient path planning to optimize information flow and ensure that gradient updates are efficiently utilized throughout the network, leading to improved performance in object detection tasks.

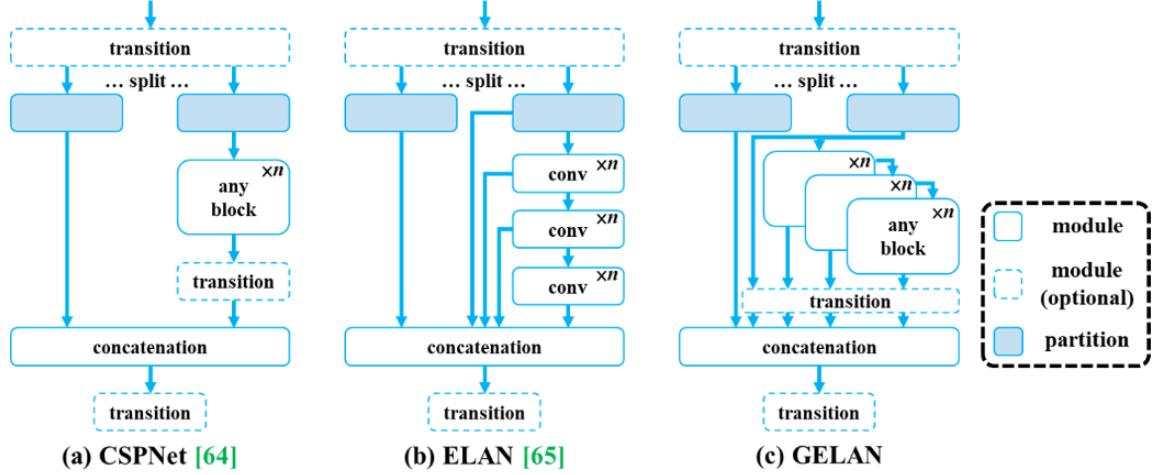


Figure A.2: The architecture of GELAN: (a) CSPNet [26], (b) ELAN [27] , and (c) proposed GELAN.

Utilizing Progressive Group Integration (PGI) to address the information bottleneck challenge and mitigate the incompatibility of deep supervision mechanisms with lightweight neural networks. Wang and colleagues developed GELAN, a notably efficient and lightweight neural network. The integration of PGI and GELAN in the creation of YOLOv9 has demonstrated competitive effectiveness. Noteworthy for its superior design, YOLOv9 achieves a reduction of 49% in parameter count and 43% in computational workload compared to YOLOv8, while concurrently exhibiting a 0.6% Average Precision (AP) [25] enhancement on the MS COCO dataset.

B Contributions

B.0.1 Sai Devarashetty, Software Engineer

- Conducted research and composed the literature review chapter.
- Conducted comparative analyses of different methodologies.
- Created explanatory flowcharts illustrating each model's architecture.
- Assisted in drafting the methodologies chapter.

B.0.2 Hector Gavilanes, Project Lead

- Conducted comprehensive proofreading and editing of all paper chapters for the final version.
- Researched and implemented code for Faster R-CNN, Mask R-CNN, and YOLOv8 methodologies.
- Analyzed, experimented, and drew conclusions throughout training, validation, and testing phases.
- Developed a specialized class label for military tank detection using YOLOv8.
- Evaluated performance across various object detection and segmentation models.
- Maintained consistent communication with the professor and peers.
- Prepared the GitHub repository and associated pages for online hosting of the research paper.
- Deployed two web applications: one for live-feed capture from a webcam using a pre-trained model, and another enabling end-users to upload photos for object detection and segmentation inference.

B.0.3 Jacob Knight, Deep Learning Architect

- Authored sections detailing methodologies such as Region Proposal Network, Translation-Invariant Anchor, and back-propagation.
- Conducted peer reviews of colleagues' work.

B.0.4 Kyle Knuth, Data Scientist

- Wrote sections covering objectives, introduction, metrics, analysis, and results.
- Assisted in collecting training and testing images for the YOLOv8 model.
- Aided in manual and auto-annotations for training data.
- Contributed to the training and implementation of the SAM (Segment Anything model).

C Resources

For your convenience, we have provided the following links accompanying the book report referencing the code notebooks used for the projects, and web applications.

C.1 Repositories

- [Online Report - GitHub repository](#)
- [Code Notebooks - GitHub repository](#)

C.2 Web Applications

- [R-CNN - Webcam App](#)
- [R-CNN - YOLOv8 App](#)