Mask R-CNN for Object Detection and Instance Segmentation

Aishwarya Joshi Keerthana Gopalakrishnan Sanika Prashant Deshmukh
Oregon State University
Corvallis, OR
joshiai@oregonstate.edu gopalake@oregonstate.edu deshmusa@oregonstate.edu

Abstract

*In this project, we implement **Mask R-CNN**, a deep learning model for object instance segmentation, which extends Faster R-CNN by incorporating a parallel branch for precise pixel-wise mask prediction. One of its key advances is the **RoIAlign** technique, which improves spatial precision and enhances segmentation accuracy. Our implementation follows the original framework, covering the entire pipeline from model training to evaluation using the **COCO dataset**. We analyze its performance in terms of detection accuracy and segmentation quality, highlighting its effectiveness in complex visual scenes. This project serves as a foundation for understanding and improving instance segmentation models, with applications in autonomous systems, medical imaging, and computer vision.Code available at: https://github.com/sanikadeshmukh/DL_MaskRCNN*

1. Introduction

Understanding the content of an image—particularly the objects' locations, their identities, and the nature of their interactions—is a core problem in the field of computer vision. Over the years, deep learning has dramatically improved object detection and segmentation, allowing machines to not only identify objects but also predict their precise geometries. Classic pipelines, including R-CNN [2] and its optimization extensions, Fast R-CNN [3] and Faster R-CNN [4], have been very successful for object detection and bounding box localization around the object. Nevertheless, simply encapsulating an object in a rectangle can be inadequate—numerous applications, including medical imaging, autonomous driving, and robotics, need a richer level of definition so as to differentiate between overlaid or clustered objects.

Mask R-CNN [1] enhances object detection by adding a segmentation module that generates pixel-level masks for each object detected. It builds on the Faster R-CNN model by introducing an additional branch to predict masks while still being capable of object classification and bounding box generation. One of the key innovations in Mask R-CNN is RoIAlign, which addresses a feature misalignment problem experienced by earlier models such as Faster R-CNN. This minor but vital modification greatly enhances segmentation accuracy, particularly for small or intricate objects.

Previous segmentation approaches tried to address the issue with various strategies. Some methods such as DeepMask [5] and SharpMask [6] tried to predict object masks prior to the classification phase; however, they had difficulty handling occlusions and overlapping objects. Conversely, Fully Convolutional Networks (FCNs) [7] tried another path by segmenting whole images at the pixel level.

However, FCNs treated all instances of the same class together, which cannot be used for tasks requiring individual object segmentation. Subsequent models such as FCIS [8] remedied this by introducing position-sensitive score maps but continued to suffer from variability in dealing with densely packed instances. Mask R-CNN alleviates the problem by extending instance segmentation as part of an object detection pipeline and is therefore more accurate and efficient.

In this project, we implemented Mask R-CNN to explore its practical implications, training the model from scratch and evaluating its performance on real-world datasets. We analyzed its efficiency, assessed its strengths and limitations, and identified challenges encountered during implementation. This report details our approach, the obstacles we faced, and potential areas for optimization to enhance Mask R-CNN's applicability in real-world scenarios. Additionally, we investigate the impact of hyperparameter tuning and dataset variations on model performance. By understanding these factors, we aim to provide insights that can guide further improvements in instance segmentation techniques.

2. Methodology

Conceptually, Mask R-CNN builds upon Faster R-CNN by adding an additional branch for object mask prediction alongside the existing class label and bounding-box offset outputs. While the idea is intuitive, generating precise

object masks requires a much finer spatial understanding compared to classification and bounding box regression. To achieve this, Mask R-CNN incorporates pixel-to-pixel alignment, a crucial enhancement missing in Fast R-CNN and Faster R-CNN, ensuring more accurate segmentation. In the following sections, we discuss its key components and how they improve instance segmentation performance. The framework for Mask R-CNN is as illustrated in Fig. 1.

2.1. Backbone Architecture

The backbone network in Mask R-CNN is responsible for extracting meaningful features from input images. In this implementation, we used pre-trained **ResNet50** and **ResNet101** with a Feature Pyramid Network **FPN** as the backbone to enhance multi-scale feature representation. ResNet architectures are widely chosen due to their efficient residual learning framework, which helps mitigate the vanishing gradient problem in deep networks. ResNet50, with 50 layers, provides a balance between computational efficiency and accuracy, making it well-suited for real-time applications. In contrast, ResNet101, with 101 layers, enables richer feature extraction and performs better on challenging datasets at the cost of increased computation.

To further optimize the backbone, we fine-tuned the pre-trained model rather than freezing its initial layers, allowing it to adapt to the instance segmentation task. Our implementation leverages the multi-scale feature extraction capability of FPN, ensuring better detection of objects at different scales. Instead of directly using the raw ResNet features, we processed the feature maps dynamically, converting them into a structured feature dictionary based on the backbone's output type (single tensor, list, or dictionary). This ensures seamless compatibility with the RoIAlign operation and enhances feature reuse. Additionally, we carefully handled input transformations and normalization to ensure stable training dynamics. By fine-tuning the backbone on our dataset and integrating it efficiently into the Mask R-CNN pipeline, we achieved improved feature extraction tailored to our segmentation task, demonstrating a well-optimized approach beyond standard pre-trained models.

2.2. RPN

In the Mask R-CNN implementation, proposals are initially generated using the existing bounding boxes from the dataset. This approach leverages the high accuracy of ground truth annotations, ensuring that the model starts with well-defined regions of interest for object detection. When ground truth proposals are unavailable or incomplete, the Region Proposal Network (RPN) steps in to generate proposals directly from the feature maps. This ensures that the model remains capable of detecting objects even in scenarios where labeled bounding boxes are missing. By using existing annotations whenever possible, the model operates more efficiently, reducing the need for computationally expensive proposal generation. At the same time, RPN provides a flexible fallback, enhancing the model's robustness and allowing it to adapt to new or incomplete data. This strategy improves both the accuracy and speed of the detection and segmentation process, ensuring reliable performance across various use cases.

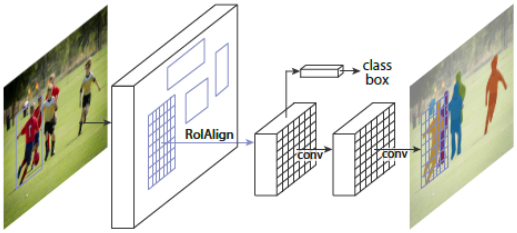

Figure 1: Framework for Instance Segmentation

2.3. RoIAlign

RoIPool, a standard method for extracting small feature maps from each Region of Interest (RoI), quantizes the floating-point RoI coordinates to fit the discrete grid of the feature map. This quantization introduces misalignments between the RoI and the extracted features, which negatively impacts tasks like **pixel-accurate** mask prediction. To overcome this limitation, we introduced an additional layer, RoIAlign, in our Mask R-CNN implementation. RoIAlign eliminates the harsh quantization of RoIPool by using bilinear interpolation to ensure precise alignment of the extracted features with the RoI boundaries. In our implementation, **MultiScaleRoIAlign** is applied, leveraging multiple feature map levels **(featmap_names=["0", "1", "2", "3"])** to better handle objects at various scales. By pooling regions to a consistent output size **(output_size=7)** with a sampling ratio of 2, RoIAlign enhances the model's ability to maintain spatial resolution, leading to more accurate predictions and improved performance in both object detection and segmentation tasks. These features are then aligned and pooled into a fixed-size output **(in this case, 7x7)**. The **sampling_ratio** parameter controls how finely the pooling operation samples the feature map to ensure high-quality feature extraction. This is depicted in Fig. 2.

2.4. BoxHead

In the Mask R-CNN architecture, the BoxHead plays a crucial role in predicting both the object class scores and the bounding box regression values. Its primary function is to process the features extracted from the Region of Interest (RoI) pooling layer and produce two key outputs: **class logits** and **bounding box predictions**. The class logits represent the likelihood of each object class for the given RoI,

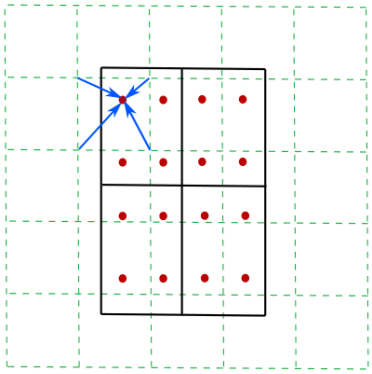

Figure 2: RoIAlign

while the bounding box predictions are used to refine the locations and sizes of the detected objects through bounding box regression.

In our implementation, after RoIAlign extracts fixed-size feature maps from the pooled regions, these feature maps are passed to the BoxHead. The input to the BoxHead is a flattened version of the pooled regions, which is processed through two fully connected layers (fc1 and fc2). These layers learn complex representations of the extracted features, ultimately outputting the class logits and bounding box predictions. The class logits are used for object classification, while the bounding box predictions are used to adjust the positions and dimensions of the proposed regions, enhancing the accuracy of object detection.

By incorporating the BoxHead, Mask R-CNN effectively bridges the gap between feature extraction and the final prediction, enabling precise classification and localization of objects. This improves the model's performance by allowing it to simultaneously predict both the object class and its spatial location, which are critical for tasks such as instance segmentation. The integration of the BoxHead thus contributes significantly to the overall performance of Mask R-CNN, enhancing both detection accuracy and mask prediction quality.

2.5. MaskHead

The MaskHead in Mask R-CNN is responsible for **generating pixel-wise segmentation masks** for each object instance detected in the image. It takes the pooled feature maps from the RoIAlign layer as input and applies a series of fully convolutional layers to produce the final segmentation mask for each object class. The network architecture of the MaskHead consists of several convolutional layers, each with ReLU activations, that progressively refine the feature maps and capture higher-level spatial information. The final convolutional layer outputs the mask predictions, where each output channel corresponds to a mask for one of the object classes. In our implementation, the MaskHead receives the feature maps from RoIAlign and applies five convolutional layers to progressively refine the features. The first four convolutional layers (with kernel size $3x3$ and hidden dimension of 256) learn increasingly complex representations of the object instance's spatial structure. The final convolutional layer reduces the number of channels to match the number of object classes, producing a mask for each class. This process allows the network to generate a binary mask for each object in the proposed regions of interest, corresponding to the class of that object. The role of the MaskHead in the overall Mask R-CNN architecture is critical for instance segmentation tasks. It enables the model to predict high-resolution segmentation masks for individual objects in the image, complementing the object classification and bounding box prediction tasks. The inclusion of the MaskHead significantly improves the model's ability to **segment** and **distinguish** objects at the pixel level, which is essential for tasks that require precise localization and delineation of object boundaries.

2.6. Network Architecture

The Mask R-CNN architecture is designed to perform instance segmentation by simultaneously predicting object bounding boxes, object classes, and pixel-wise segmentation masks. In our implementation, we explore the use of two different ResNet architectures as backbones—ResNet-50 and ResNet-101—along with the integration of a Feature Pyramid Network (FPN) for more effective feature extraction. The FPN enhances the model's performance by incorporating a top-down architecture with lateral connections, which helps in creating multi-scale feature maps. This allows the network to more accurately detect objects at different scales, making it particularly useful for object detection tasks. When using the ResNet-FPN backbone, Mask R-CNN achieves improved accuracy and robustness.

The architecture consists of two main branches under the network head:

1. **Object Classification and Bounding Box Regression Branch:**

    - This branch is responsible for classifying objects and predicting the bounding boxes. It receives feature maps from the backbone (either ResNet-50 or ResNet-101, potentially enhanced with FPN), which are then processed through the Region Proposal Network (RPN) to generate candidate object locations. The RPN is based on a convolutional layer followed by two separate

convolutional outputs: one for object classification (logits) and another for bounding box predictions. The RPN uses 512 filters in a $3x3$ convolution with ReLU activation and produces outputs for a fixed number of anchors (9 anchors per location in our case).

- After generating proposals, RoIAlign is applied to the feature maps to ensure that the regions of interest are correctly aligned spatially. This operation prevents the quantization errors typically seen in RoIPool and provides high-quality features for bounding box regression and classification. These aligned features are then passed through a fully connected head (BoxHead) consisting of two fully connected layers with ReLU activation, each with 1024 hidden units, followed by separate fully connected layers for class logits and bounding box regression outputs.

2. **Segmentation Mask Branch:**

   - The second branch is responsible for generating pixel-wise segmentation masks for each detected object. After the RoIAlign operation, the pooled feature maps are passed through a MaskHead network, which consists of five convolutional layers. The first four layers use a kernel size of 3x3 with 256 filters and ReLU activations, progressively refining the features, while the final convolutional layer reduces the number of channels to match the number of object classes (91 classes in the case of the COCO dataset). This output represents the segmentation masks for each object class in the proposed region.
   - The MaskHead is a fully convolutional network (FCN) that directly outputs a binary mask for each class.

In our implementation, the RPN is used to generate object proposals, and the RoIAlign layer ensures precise alignment of these proposals with the feature maps. The BoxHead takes the aligned features and produces class scores and bounding box predictions. Finally, the MaskHead produces the segmentation masks for each object instance. This multi-task architecture allows the model to not only classify and localize objects but also to segment them at the pixel level.

Overall, the architecture is designed for high performance, with the backbone **(ResNet-50/101+FPN)** followed by the **RPN**, **RoIAlign**, **BoxHead**, and **MaskHead**. The use of ResNet-50 or ResNet-101 as the backbone, combined with FPN, significantly improves the model's ability to detect and segment objects across various scales, while the combination of RoIAlign and the MaskHead ensures accurate segmentation and high-quality predictions.

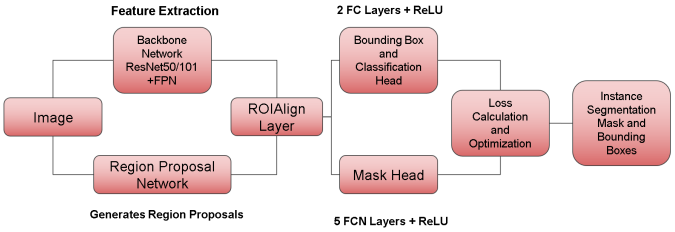

Figure 3: Model Architecture

3. Implementation Details

This section outlines the key aspects of our implementation, including dataset pre-processing, model architecture, and training procedures. Several hyperparameters were set based on the methodology presented in [1], while others were fine-tuned by us through experimentation to achieve optimal accuracy. We also discuss loss function modifications, optimization strategies, and the hardware setup, including the use of mixed-precision training and gradient clipping for improved efficiency.

3.1. Dataset and Pre-processing

The dataset used for training and evaluation was the COCO Dataset (2017), a widely used benchmark in the field of object detection and instance segmentation. It contains a diverse set of images annotated with bounding boxes, pixel-wise masks, and class labels, enabling the model to learn both object segmentation and localization.

To standardize the input data, several pre-processing steps were performed on the images, including resizing, normalization to standardize pixel values, and conversion to tensor format for compatibility with PyTorch-based deep learning models. To improve generalization and prevent overfitting, data augmentation techniques such as **horizontal flipping**, **color jitter**, and **normalization** were also applied, making the model more robust.

3.2. Training

The training process follows an image-centric approach, where input images are augmented and resized to improve generalization. We trained our model using the **AdamW optimizer**, which provided improved stability and ensured smoother convergence. The learning rate was dynamically adjusted using a **cosine annealing scheduler**, while a weight decay of 0.0001 was applied to prevent overfitting. The training was conducted for two different settings: (1) 50 epochs with a dataset of 50k images and (2) 100 epochs

with 80k images using a ResNet-50 backbone with Feature Pyramid Network (FPN) and an initial learning rate of 0.0005. The validation set consisted of 5k images. A batch size of 16 was chosen to balance efficiency and memory constraints.

To further enhance performance, we explored hyperparameter tuning by updating the backbone to ResNet-101 with FPN, allowing the model to capture more complex details. We also experimented with different training durations (20, 30, 50, and 100 epochs) and dataset sizes (20k, 30k, 50k, and 80k images). Additionally, we evaluated the impact of varying the learning rate, increasing it to 0.0007 and decreasing it to 0.0001. We also experimented with changing the minimum number of proposals generated from 15 to 30, which showed a positive impact on accuracy.

For loss optimization, we initially used a multi-task loss function:

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

where L_{cls} (classification loss) and L_{box} (bounding box regression loss) were implemented using cross-entropy and SmoothL1 loss, respectively. The mask loss L_{mask} was computed using binary cross-entropy. To further improve performance, we replaced these loss functions with Sigmoid Focal Loss for L_{cls} , IoU Loss for L_{box} , and Dice Loss for L_{mask} . These changes improved performance for smaller datasets (5k, 10k images) by helping the model avoid class imbalance and learn more precise segmentation masks. However, as dataset sizes increased, training accuracy stagnated, likely due to the model getting stuck in a local minimum rather than achieving the global optimum.

The training was conducted using high-performance hardware, specifically **A100** and **Tesla V100-SXM3-32GB** GPUs, enabling efficient processing of large-scale datasets. Mixed-precision training with gradient scaling was utilized to enhance computational efficiency. We also applied gradient clipping to stabilize training and avoid exploding gradients.

Overall, while our tuning strategies improved performance in certain settings, the challenges associated with larger datasets indicated that further architectural and optimization refinements might be required to achieve consistently better results.

4. Results and Performance

In this section, we present the performance evaluation of our trained Mask R-CNN model on the COCO 2017 dataset, along with a comparison of different backbone networks. We analyze how variations in architecture, training epochs, dataset size, and learning rates impact model performance. Additionally, we provide qualitative results by showcasing images with predicted bounding boxes and segmentation masks, highlighting the effectiveness of our trained models.

4.1. Performance Comparison

In this subsection, we compare the performance of different model architectures we experimented with. The first model utilizes Mask R-CNN with a ResNet50+FPN backbone, while the second model employs a deeper ResNet101+FPN architecture. Table 1 and Table 2 present a comparison of ResNet50+FPN and ResNet101+FPN across different training epochs and dataset sizes.

For the ResNet50+FPN model, we observed a steady improvement in validation accuracy as the number of training epochs increased from 30 to 100. The validation accuracy improved from 38% at 30 epochs to 46% at 100 epochs when trained on 50K images and tested on 5K images. However, further increasing the dataset size to 80K did not yield significant improvements, suggesting a potential saturation point where adding more data does not drastically enhance performance. This indicates that a more complex model might be required to achieve further accuracy gains.

The ResNet101+FPN model was trained with a slightly higher learning rate (0.0007 vs. 0.0005 in ResNet50+FPN) while maintaining the same dataset splits, as shown in Table 2. Compared to ResNet50+FPN, ResNet101+FPN demonstrated better performance at 30 and 50 epochs, reaching a validation accuracy of 46% at 50 epochs—the best performance achieved by ResNet50+FPN. Notably, training the ResNet101+FPN model for 100 epochs with a reduced learning rate of 0.0001 resulted in a further validation accuracy improvement to 47%, showing that deeper architectures can provide incremental benefits when trained for a longer duration with proper learning rate adjustments.

It can be observed that for ResNet101+FPN, increasing both the dataset size and the number of training epochs has the potential to yield even better accuracy. Furthermore, ResNet101+FPN with a learning rate of 0.0001 provides slightly better accuracy than ResNet50+FPN with a learning rate of 0.0005, while keeping the dataset size and the number of epochs constant. This suggests that a combination of deeper architectures, prolonged training, and careful learning rate scheduling can enhance model generalization and accuracy.

The training loss and accuracy curves in Fig. 4 illustrate the learning dynamics for the ResNet101+FPN model trained for 100 epochs with a learning rate of 0.0001 on a 50K dataset. The loss continuously decreases, while the accuracy steadily increases, peaking at around 53% for training accuracy and 47% for validation accuracy before slightly plateauing.This further supports the idea that deeper networks, when properly tuned, can outperform shallower architectures in segmentation tasks.

4.2. Predictions by Mask R-CNN

This subsection evaluates the predictions generated by our Mask R-CNN model on the COCO 2017 dataset. As

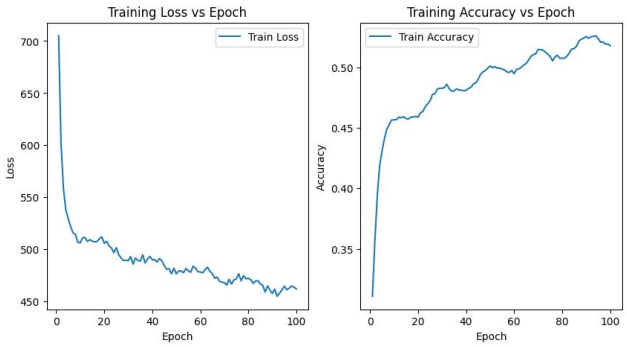

Figure 4: ResNet101+FPN: Training Loss and Accuracy

illustrated in Fig. 5 and Fig. 6, the model excels in both object detection and instance segmentation, accurately delineating multiple object classes with distinct bounding boxes and pixel-level masks. The integration of RoIAlign enhances spatial precision, contributing to sharper segmentation and improved alignment with object boundaries.

However, the model struggles with occluded or overlapping objects, where segmentation masks become less defined, leading to potential misclassification or missed detections. Another challenge was the absence of certain object labels in the training dataset. Some detected objects could not be assigned proper labels, which impacted classification accuracy. The model's reliance on COCO labels also introduced complications, as not all objects in the test images conformed strictly to the COCO annotation guidelines. This discrepancy made it difficult for the model to classify certain objects correctly, particularly when their labels were missing or ambiguous. Fine-tuning the model on a more diverse dataset could help mitigate this issue.

Despite these limitations, the overall performance of the Mask R-CNN implementation remains strong. The model demonstrates impressive segmentation capabilities and is able to generalize well across various object categories. Future improvements, such as refining the training dataset, enhancing post-processing techniques, and incorporating advanced occlusion-handling strategies, could further improve segmentation accuracy and robustness in real-world scenarios.

Epoch	Train Size	Val Size	LR	Train Acc	Val Acc
30	20K	5K	0.0005	40	38
50	50K	5K	0.0005	48	45
100	50K	5K	0.0005	54	46
100	80K	5K	0.0005	55	46

Table 1: Performance results for ResNet50+FPN

Figure 5: Mask R-CNN predictions on sample images from the COCO dataset.

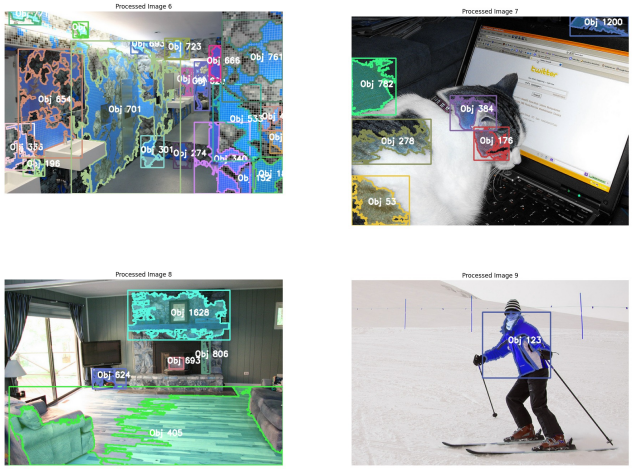

Figure 6: Additional Mask R-CNN predictions on sample images from the COCO dataset.

Epoch	Train Size	Val Size	LR	Train Acc	Val Acc
30	20K	5K	0.0007	43	40
50	50K	5K	0.0007	49	46
100	50K	5K	0.0001	53	47

Table 2: Performance results for ResNet101+FPN

5. Limitations and Challenges

Our implementation faced several challenges that impacted the overall performance and efficiency of the model. One significant issue was the difficulty in computing the loss when labels were missing, which posed a challenge

during training and evaluation. Additionally, the COCO Test Dataset annotations are not accessible to the public, limiting the ability to fully validate our model on unseen data. The COCO Validation Dataset, consisting of only 5K images, further restricted the scope of evaluation, as it did not provide sufficient variety for thorough testing. Another complication arose from the fact that not all objects strictly follow the COCO label guidelines, which made it harder to classify certain objects, especially when their labels were unknown or ambiguous.

Overfitting was a prevalent issue, particularly when training on smaller datasets, where the model could memorize the data rather than generalizing well to unseen examples. The high computational demands of training, especially with large-scale datasets, led to slow training times despite using powerful hardware like the A100 and Tesla V100 GPUs. Furthermore, the model struggled with detecting partially hidden or overlapping objects, which is a common challenge in instance segmentation tasks, and the accuracy was impacted by the difficulty in segmenting these occluded objects. These limitations highlight areas where further improvements are needed, especially in handling missing annotations, classifying ambiguous objects, and optimizing for better generalization on smaller datasets.

6. Conclusion and Future Work

In this project, we implemented Mask R-CNN for object detection and instance segmentation using ResNet-50 and ResNet-101 with FPN, using the COCO 2017 dataset as the benchmark. Our results showed that increasing the number of training epochs and dataset size improved accuracy up to a certain limit, with ResNet-101 providing better accuracy early in the training process. However, some limitations were observed, such as overfitting on small datasets, high computational requirements, and challenges in handling occlusions.

To enhance model performance, future work could focus on optimizing inference time for real-time applications and refining mask predictions for improved segmentation accuracy. To reduce the computational cost of training, methods such as pruning, quantization, or using lightweight architectures could make the model more feasible for use on edge devices. Additionally, incorporating LiDAR or depth data (e.g., point cloud data) may further enhance the model's robustness in complex environments.

References

[1] He, K., Gkioxari, G., Dollár, P., Girshick, R. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, 2017.

[2] Girshick, R. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, 2015.

[3] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In NIPS, 2016.

[4] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[5] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In NIPS, 2015.

[6] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Doll´ar. Learning to refine object segments. In ECCV, 2016.

[7] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[8] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In CVPR, 2017.