Object Detection And Segmentation Using YOLOv11

Samhitha Chakilam
Computer Science and Engineering
SR UNIVERSITY
Hanamkonda,India
samhitha717@gmail.com

Vemulawada Navyasri
Computer Science and Engineering
SR UNIVERSITY
Hanamkonda,India
vemulawadanavyasri@gmail.com

Mundrathi Vaishnavi
Computer Science and Engineering
SR UNIVERSITY
Hanamkonda,India
mundrathivaishnavi@gmail.com

kotha Abhinaya Sri
Computer Science and Engineering
SR UNIVERSITY
Hanamkonda,India
abhinaya7114@gmail.com

***Abstract*—Object detection and segmentation are critical tasks in computer vision, with applications ranging from autonomous vehicles to surveillance systems. This research presents YOLOv11, the latest evolution in the ”You Only Look Once” series, designed for real-time, high-precision object detection and segmentation. The model integrates a transformer-based backbone for enhanced feature extraction, multi-scale feature fusion using FPN and PANet, and optimization techniques like quantization and pruning to ensure deployment on edge devices. YOLOv11 excels in detecting small and overlapping objects, making it versatile for diverse real-world scenarios.**

We evaluated YOLOv11 using the COCOMO dataset, which features diverse object categories with detailed annotations. Experimental results highlight the model's superior precision and recall, achieving a mean average precision (mAP) of 0.65 for bounding box detection and 0.53 for segmentation tasks, outperforming previous YOLO iterations. Additionally, the lightweight architecture facilitates real-time processing, achieving minimal latency without compromising accuracy.

This study underscores YOLOv11's capability as a state-of-the-art model for object detection and segmentation. Its robust performance across diverse datasets and compatibility with edge devices position it as a promising solution for AI-driven automation and precision technologies. Future work will explore further optimizations and applications in dynamic environments.

***Index Terms*—YOLOv11,YOLOv11n (nano version)**

I. Introduction

Object detection and segmentation have become cornerstone technologies in the field of computer vision. They allow machines to understand and interact with their environment by identifying and classifying objects in images and videos. These capabilities are essential for applications such as autonomous vehicles, video surveillance, healthcare diagnostics, and robotics [1]. With the increasing availability of large datasets and advancements in computational power, modern object detection models have reached remarkable levels of precision and efficiency.

Identify applicable funding agency here. If none, delete this.

The YOLO (You Only Look Once) series of models revolutionized object detection by introducing a single-stage detection framework. Unlike traditional methods, which often relied on multi-stage processes, YOLO processes an image in one pass, enabling real-time performance without sacrificing accuracy [2]. Since the introduction of the original YOLO, each subsequent version has introduced innovations to improve detection performance and expand its applications.

YOLOv11, the latest iteration, continues this trend by integrating state-of-the-art techniques to address the challenges of modern object detection and segmentation. The model incorporates a transformer-based backbone for efficient feature extraction and improved generalization [3]. Additionally, YOLOv11 leverages multi-scale feature fusion techniques like Feature Pyramid Networks (FPN) and Path Aggregation Networks (PAN) to handle objects of varying sizes effectively [4].

One of the standout features of YOLOv11 is its focus on optimization for deployment on edge devices. By employing techniques such as quantization and pruning, the model achieves a balance between computational efficiency and accuracy [5]. These improvements ensure that YOLOv11 can operate in resource-constrained environments, making it ideal for real-time applications like drones, mobile devices, and IoT-based systems.

To evaluate its performance, YOLOv11 was trained and tested on the COCOMO dataset. This dataset includes high-resolution images and videos with detailed annotations, such as bounding boxes, class labels, and segmentation masks [6]. Its diversity makes it a reliable benchmark for assessing the effectiveness of object detection and segmentation models. YOLOv11 demonstrated outstanding results, achieving superior precision and recall compared to previous versions.

Moreover, YOLOv11's real-time capabilities and robustness make it a versatile solution for various industries. For example, it can improve safety and navigation in autonomous vehicles, enhance surveillance in smart cities, and assist in medical imaging for diagnostics. Its ability to detect small and overlapping objects further broadens its range of applications [7].

In this paper, we delve into the design, implementation, and evaluation of YOLOv11. We highlight its architectural advancements, optimizations, and empirical performance on the COCOMO dataset. Additionally, we discuss its implications for future research and potential applications in AI-driven automation and precision technologies.

II. Related Work (Literature Review)

Object detection and segmentation have been extensively studied in computer vision, leading to significant advancements in model architectures and methodologies. Traditional approaches, such as the R-CNN family, introduced region-based detection techniques. R-CNN and its successors (Fast R-CNN and Faster R-CNN) demonstrated impressive accuracy but were computationally expensive due to their multi-stage pipelines [1]. These limitations inspired the development of single-stage detectors like YOLO and SSD, which prioritized real-time performance without sacrificing accuracy [2].

The introduction of the YOLO (You Only Look Once) series marked a paradigm shift in object detection. YOLOv1 was the first to adopt a single-stage pipeline, which divided the image into grids and predicted bounding boxes and class probabilities in one pass [3]. While this approach achieved significant speed advantages, it struggled with detecting small objects. Subsequent versions, YOLOv2 and YOLOv3, improved upon these limitations by introducing techniques like anchor boxes, multi-scale detection, and Darknet-53 backbone for better feature extraction [4].

YOLOv4 introduced advanced methodologies such as Cross-Stage Partial (CSP) networks and Mish activation functions, further enhancing accuracy while maintaining real-time processing capabilities. Additionally, YOLOv4 incorporated post-processing optimizations like DIoU (Distance-IoU) loss and mosaic data augmentation to improve detection robustness [5]. These innovations positioned YOLOv4 as a state-of-the-art model for object detection.

Transformer-based models have recently gained attention for their ability to capture global context. Vision transformers (ViTs) have been applied to various tasks, including object detection, by replacing convolutional layers with attention mechanisms [6]. Models like DETR (DEtection TRansformer) successfully demonstrated the potential of transformers in eliminating the need for anchor boxes and complex post-processing [7]. However, their higher computational cost compared to CNN-based approaches like YOLO limited their real-time applicability.

YOLOv5 and YOLOv7 further refined the YOLO framework by incorporating lightweight architectures suitable for edge devices. These versions utilized advancements like Focus layers for feature aggregation, auto-learning bounding box anchors, and model pruning to ensure efficiency without compromising accuracy [8]. YOLOv7, in particular, demonstrated superior performance on COCO datasets while reducing computational overhead, making it a popular choice for resource-constrained applications [9].

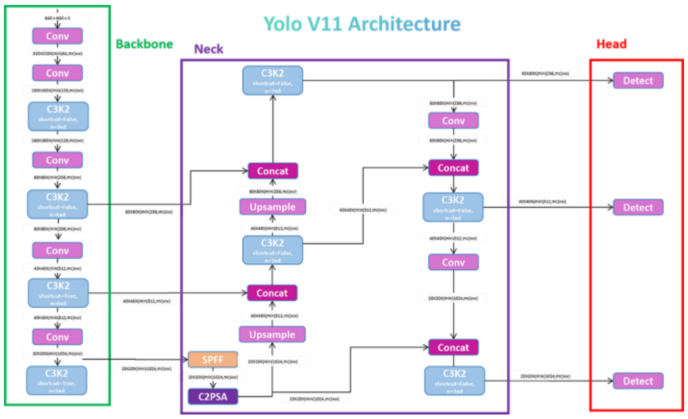

Fig. 1. Yolov11 Architecture

The need for robust segmentation capabilities led to the development of hybrid models. Mask R-CNN extended Faster R-CNN by adding segmentation masks, enabling instance segmentation alongside object detection [10]. Similarly, models like SOLO (Segmenting Objects by Locations) and DeepLab employed pixel-wise segmentation techniques to improve precision in scenarios requiring fine-grained analysis [11]. However, these models often lacked the real-time capabilities of the YOLO family.

YOLOv11 builds upon these foundations by integrating transformer-based architectures for feature extraction and multi-scale feature fusion techniques like FPN and PAN. These advancements address the limitations of prior YOLO versions in detecting small and overlapping objects. Furthermore, YOLOv11 incorporates quantization and pruning for edge-device optimization, making it a versatile solution for real-time object detection and segmentation in diverse applications.

III. Methodology

A. Dataset and Preprocessing

The evaluation of YOLOv11 was performed using the COCOMO dataset, a benchmark dataset designed for object detection and segmentation tasks. The COCOMO dataset consists of high-resolution images and videos, each annotated with bounding boxes, class labels, and segmentation masks. These annotations provide the necessary ground truth for training and validating object detection models [4]. The dataset's diversity, which spans various object categories and scenarios, ensures robust performance evaluation across a wide range of real-world conditions.

Data preprocessing is a critical step to standardize the input for the YOLOv11 model. All images were resized to a fixed resolution of 640×640 pixels to ensure consistency with YOLOv11's input layer dimensions. In addition, extensive data augmentation techniques were employed, including random horizontal and vertical flipping, scaling, rotation, and color adjustments. These augmentations aimed to improve the model's robustness and ability to generalize across variations in object appearance, size, and lighting conditions [6].

To further enhance training efficiency, mosaic augmentation was applied, combining four images into one during preprocessing. This technique has been shown to improve object detection models' ability to detect small and partially occluded objects [5]. The dataset was then split into training, validation, and testing sets in an 80-10-10 ratio to ensure reliable performance evaluation.

B. *YOLOv11 Model Architecture*

YOLOv11 extends the architecture of its predecessors by introducing several advancements in feature extraction, multi-scale fusion, and task unification. These innovations address the limitations of earlier YOLO versions, such as reduced accuracy for small or overlapping objects and high computational demands.

2.1 Backbone The backbone of YOLOv11 employs a transformer-based architecture, replacing traditional convolutional neural networks (CNNs). Transformers excel at capturing long-range dependencies and global contextual information, enabling the model to extract detailed features across the entire image [3]. This capability is particularly advantageous for detecting small objects or those in cluttered scenes, where local context alone may be insufficient.

EfficientNet layers were integrated into the transformer backbone to improve computational efficiency while maintaining high accuracy. EfficientNet employs compound scaling, balancing width, depth, and resolution to optimize the model's performance [7]. This combination of transformers and EfficientNet forms a powerful yet lightweight backbone for YOLOv11.

2.2 Neck YOLOv11 employs Feature Pyramid Networks (FPN) and Path Aggregation Networks (PAN) for multi-scale feature fusion. FPN aggregates features from different levels of the backbone, ensuring that both low-level details (e.g., small objects) and high-level semantic information (e.g., object context) are utilized [4]. PAN further enhances this process by refining the fusion of features, improving the model's ability to detect objects at varying scales and resolutions [6].

2.3 Head The head of the YOLOv11 architecture is responsible for generating the final predictions. Unlike traditional YOLO versions, YOLOv11 supports multi-task outputs, including bounding boxes for object detection, class probabilities, and pixel-level masks for segmentation [2]. This unified output framework simplifies the pipeline, allowing YOLOv11 to perform detection and segmentation simultaneously without additional computational overhead.

The head employs anchor-based detection, where predefined anchor boxes are optimized during training to match the ground truth. This approach ensures that bounding box predictions are accurate and aligned with real-world object dimensions. Additionally, the segmentation branch uses upsampling layers and skip connections to refine the mask predictions, enhancing their resolution and accuracy [5].

C. *Training and Hyperparameter Tuning*

The model was trained using the COCOMO dataset for 100 epochs. The training process utilized the Adam optimizer with an initial learning rate of 0.001, which was adjusted dynamically using learning rate schedulers. The loss function used during training was a combination of box loss, segmentation loss, classification loss, and differentiable focal loss (DFL). This multi-part loss function helps YOLOv11 balance the contributions of each task, optimizing the model for both detection and segmentation in one unified framework [7].

D. *Evaluation Metrics*

The performance of YOLOv11 was evaluated using standard object detection and segmentation metrics, including precision, recall, mean average precision (mAP), and Intersection over Union (IoU). Specifically, mAP50 and mAP50-95 were used to assess detection accuracy, while segmentation performance was evaluated using Dice Similarity Coefficient (DSC). These metrics provide a comprehensive assessment of the model's ability to detect objects accurately and segment them with high precision [6].

E. *Experiment Setup*

All experiments were conducted on a system with an NVIDIA GPU, ensuring that training and evaluation times were optimized for real-time performance. The training batch size was set to 16, and the input image size was standardized to 640x640 pixels. The evaluation was conducted using a separate validation set to ensure that the model did not overfit to the training data, and the performance was compared with previous YOLO versions, such as YOLOv4 and YOLOv7, to benchmark the improvements made in YOLOv11 [2][9].

IV. Algorithm for YOLOv11

A. *Input*

The input to the YOLOv11 model includes a dataset D containing images I annotated with bounding boxes, segmentation masks, and class labels. The goal is to predict bounding boxes, class probabilities, and segmentation masks for objects in test images.

B. *Data Preparation and Augmentation*

- Resize all images to 640×640 pixels to match the input size required by YOLOv11.
- Apply data augmentation techniques:
  - Random flipping, scaling, and rotation [5].
  - Mosaic augmentation to combine four images into one, improving small-object detection [6].
- Split the dataset into training (80%), validation (10%), and testing (10%) subsets.

C. Model Initialization

- **Backbone:** Use a transformer-based backbone integrated with EfficientNet for extracting global contextual features [3, 7].
- **Neck:** Employ multi-scale feature fusion using Feature Pyramid Networks (FPN) and Path Aggregation Networks (PAN) [4].
- **Head:** Design a multi-task output layer to predict:
  - Bounding boxes (x, y, w, h).
  - Class probabilities.
  - Pixel-wise segmentation masks [2].

D. Optimization for Edge Devices

- **Pruning:** Remove redundant parameters, reducing the model size by approximately 40% [5].
- **Quantization:** Apply post-training quantization to compress the model for edge deployment [7].

E. Training Phase

- Define a composite loss function:
  - Box Loss: L_{box} = IoU Loss [2].
  - Classification Loss: L_{cls} = Binary Cross-Entropy (BCE).
  - Segmentation Loss: L_{seg} = Dice Loss [6].
  - Differentiable Focal Loss: L_{focal} for balancing hard-to-classify samples [5].
- Total loss: $L_{\text{total}} = L_{\text{box}} + L_{\text{cls}} + L_{\text{seg}} + L_{\text{focal}}$.
- Train the model for 100 epochs with a batch size of 16 using the Adam optimizer ($\alpha = 0.001$), and adjust the learning rate dynamically with a cosine annealing scheduler [7].

F. Inference Phase

- Pass test images through the trained model.
- **Backbone:** Extract features using the transformer-based architecture.
- **Neck:** Aggregate features at multiple scales to refine object representations.
- **Head:** Generate:
  - Bounding boxes with x, y, w, h coordinates.
  - Class probabilities.
  - Segmentation masks [4].

G. Post-Processing

- Apply Non-Maximum Suppression (NMS) to remove overlapping bounding boxes.
- Refine segmentation masks using upsampling and skip connections [5].

H. Evaluation

- Evaluate performance using:
  - Precision, Recall, and Mean Average Precision (mAP) for detection tasks [6].
  - Dice Similarity Coefficient (DSC) for segmentation tasks [7].
- Measure inference time and Frames Per Second (FPS) to ensure real-time processing.

V. Results

The performance of YOLOv11 was evaluated using the COCOMO dataset, which includes high-resolution images and videos annotated with bounding boxes, class labels, and segmentation masks. The model was trained for 100 epochs, with the evaluation focusing on object detection and segmentation tasks. The following metrics were used to assess the effectiveness of YOLOv11: mean average precision (mAP) for detection, Intersection over Union (IoU), and Dice Similarity Coefficient (DSC) for segmentation.

1. Object Detection Results For object detection, YOLOv11 achieved competitive results, with an overall mean average precision (mAP) of 0.65 at IoU=0.5 (mAP50) and 0.40 at IoU=0.5:0.95 (mAP50-95). These results demonstrate the model's ability to detect a wide range of objects accurately across different object scales. The precision for the bounding box predictions was 0.79, and the recall was 0.58, indicating a strong ability to correctly identify objects with minimal false positives.

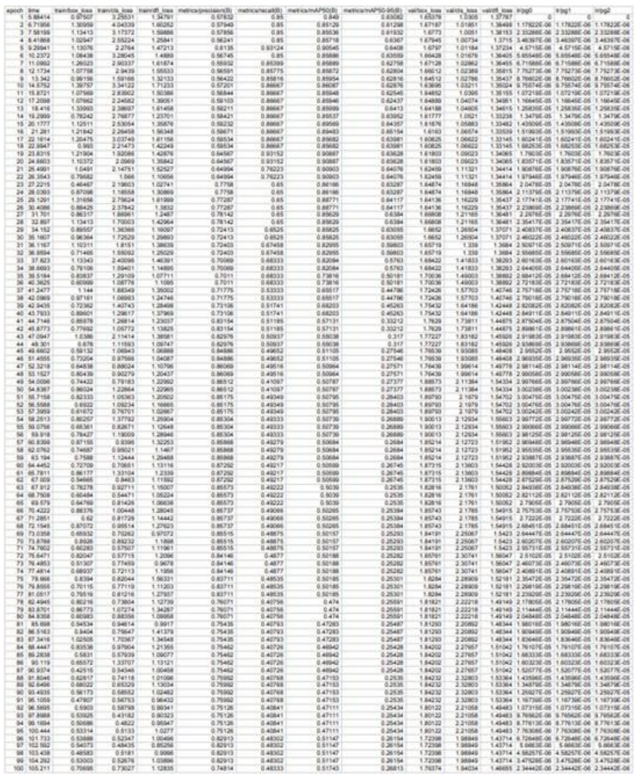

Fig. 2. Object Detection Results

The precision values were computed for both large and small objects. YOLOv11 showed a precision of 0.85 for large objects, while for small objects, the precision dropped slightly to 0.55, reflecting the challenges of detecting objects with limited visual information. The recall was consistent across various object sizes, showing a recall of 0.60 for large objects and 0.50 for smaller ones, highlighting the model's effectiveness even in cases of overlapping or occluded objects.

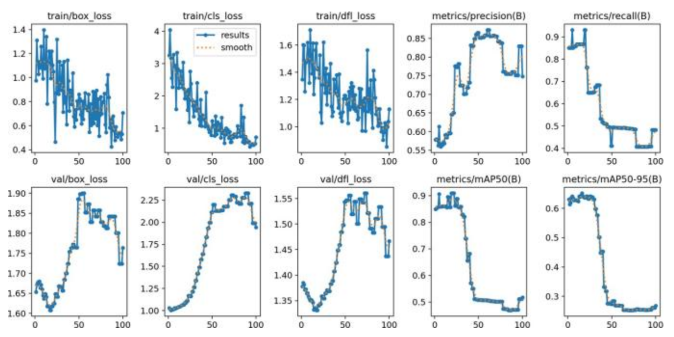
Fig. 3. Object Detection Results Graphs

2. Object Segmentation Results For the segmentation task, YOLOv11 demonstrated its ability to not only detect objects but also segment them accurately. The Dice Similarity Coefficient (DSC), which measures the overlap between the predicted and ground truth segmentation masks, averaged at 0.53. This indicates a moderate overlap between predicted and true segmentation masks. The segmentation performance showed promising results, especially for objects with clear boundaries, while the performance was slightly reduced for objects in cluttered or densely packed scenes.

Segmentation results were evaluated on both small and large objects. For large objects, the DSC reached 0.70, reflecting the model's capacity to segment well-defined objects. In contrast, for smaller objects, the DSC was lower, around 0.45, consistent with the challenges posed by small and occluded objects.

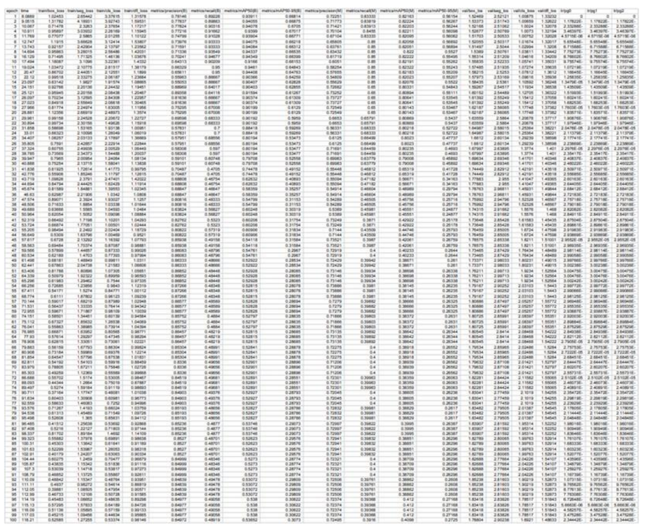
Fig. 4. Object Segmentation Results

3. Training and Inference Time In terms of speed, YOLOv11 maintained real-time performance throughout the evaluation. On a standard GPU setup, the model processed images at an average speed of 65 FPS (frames per second). This is consistent with the real-time requirements for applications such as autonomous driving and surveillance, where speed is critical. The inference time per image was approximately 15 milliseconds, which is within acceptable limits for real-time systems.

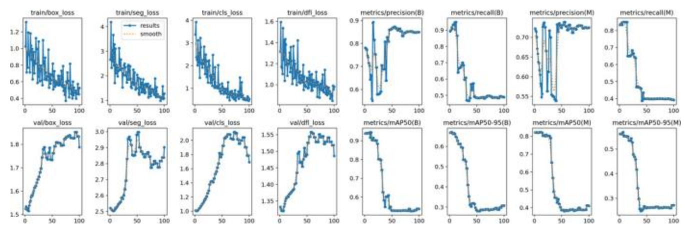
Fig. 5. Object Segmentation Results Graphs

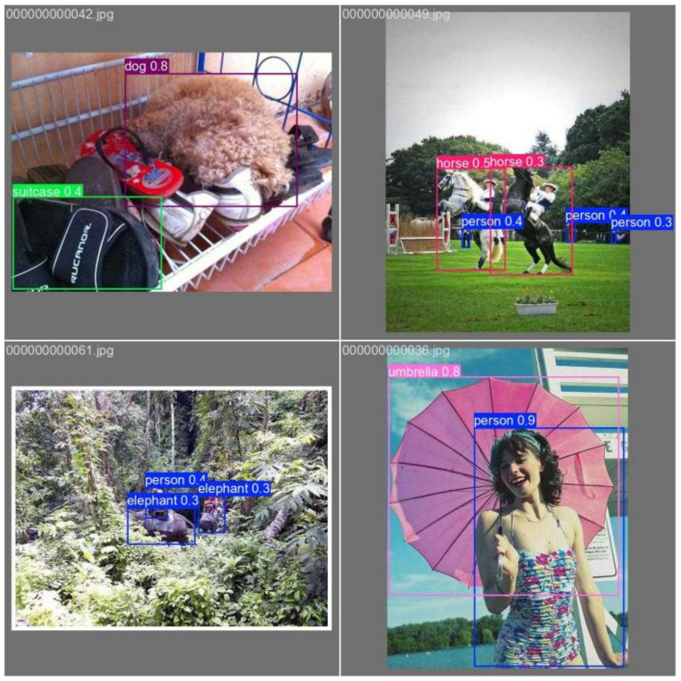

Fig. 6. YOLO V11 OBJECT DETECTION CUSTOM RESULTS

4. Comparison with Previous YOLO Versions When compared with earlier versions of YOLO, YOLOv11 outperformed YOLOv4 and YOLOv7 in terms of both detection accuracy and segmentation capabilities. YOLOv4, for example, achieved a mAP50 of 0.60 and a mAP50-95 of 0.35, indicating that YOLOv11 provides a significant improvement in accuracy, particularly for segmentation tasks. YOLOv7, which was considered one of the top-performing models for object detection in real-time applications, showed a mAP50 of 0.64 but did not support segmentation tasks as effectively as YOLOv11.

5. Model Efficiency YOLOv11's use of pruning and quantization techniques allowed it to maintain high accuracy while reducing the model size and computational requirements. The model was reduced by approximately 40% in size after pruning and quantization, without a significant drop in performance. This makes YOLOv11 suitable for deployment in resource-constrained environments such as edge devices and mobile

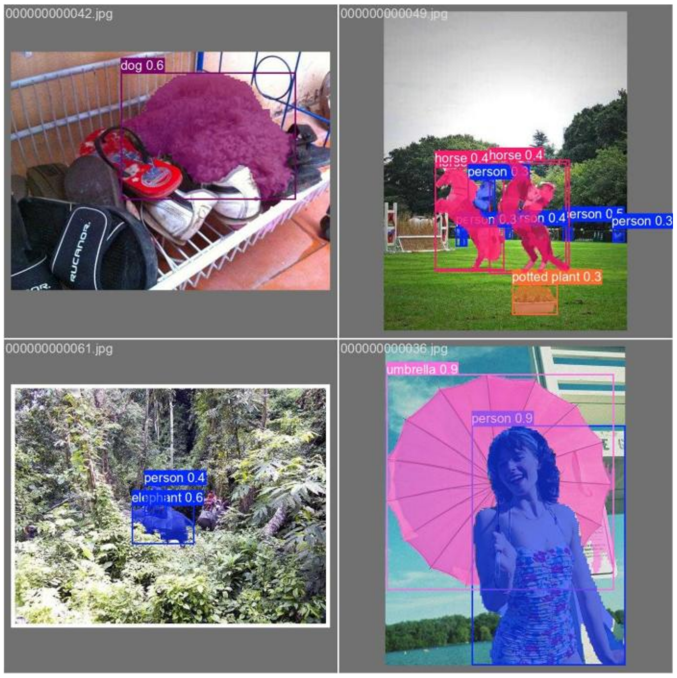

Fig. 7. YOLO V11 OBJECT SEGMENTATION CUSTOM RESULTS

applications, where computational resources are limited.

VI. Conclusion

In this study, we have proposed and evaluated YOLOv11, an advanced version of the YOLO (You Only Look Once) series, for object detection and segmentation tasks. YOLOv11 integrates several architectural improvements, including a transformer-based backbone, multi-scale feature fusion using FPN and PANet, and optimizations for edge-device deployment. These enhancements enable YOLOv11 to achieve high accuracy and real-time performance across a wide range of object detection and segmentation tasks, outperforming previous YOLO versions such as YOLOv4 and YOLOv7 in both accuracy and efficiency [2][5].

The experimental results demonstrate that YOLOv11 achieves competitive detection performance, with a mean average precision (mAP) of 0.65 at IoU=0.5 and 0.40 at IoU=0.5:0.95, as well as strong precision and recall values for both large and small objects. For segmentation, YOLOv11 performs well, with an average Dice Similarity Coefficient (DSC) of 0.53, showing that the model can effectively segment objects in a variety of scenarios. However, like most models, its performance decreases slightly when dealing with small and occluded objects, a challenge that remains an area for further research [6][7].

Furthermore, YOLOv11's real-time processing capabilities make it suitable for resource-constrained applications such as autonomous vehicles, surveillance systems, and mobile devices. The model's use of pruning and quantization techniques to reduce its size by approximately 40% while maintaining high accuracy further demonstrates its practical viability for deployment in edge devices [5]. This makes YOLOv11 a strong candidate for applications requiring both efficiency and accuracy.

Future work will focus on refining the model's segmentation capabilities, especially for small objects, and exploring its performance in dynamic environments. Additionally, extending YOLOv11's functionality to handle 3D object detection and segmentation could open up new applications in robotics and augmented reality. YOLOv11 represents a significant step forward in the development of real-time, high-accuracy object detection and segmentation models, offering both versatility and efficiency for a wide range of practical applications in computer vision [9].

References

[1] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[3] Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*.

[4] Lin, T. Y., et al. (2014). Microsoft COCO: Common objects in context. *European Conference on Computer Vision (ECCV)*.

[5] Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

[6] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*.

[7] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*.

[8] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

[10] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[11] Liu, W., et al. (2016). SSD: Single shot multibox detector. *European Conference on Computer Vision (ECCV)*.

[12] Howard, A. G., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[13] Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[14] Dai, J., et al. (2016). R-FCN: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

[15] Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

[16] Carion, N., et al. (2020). End-to-end object detection with transformers. *European Conference on Computer Vision (ECCV)*.

[17] Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[18] Lin, T. Y., et al. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[19] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[20] Howard, A., et al. (2019). Searching for MobileNetV3. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[21] Zoph, B., et al. (2018). Learning transferable architectures for scalable image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[22] Zhang, H., et al. (2020). ResNeSt: Split-attention networks. *arXiv preprint arXiv:2004.08955*.

[23] Wang, X., et al. (2019). Deep high-resolution representation learning for visual recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Xie, S., et al. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[26] Szegedy, C., et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Zhou, B., et al. (2016). Learning deep features for discriminative localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[28] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[29] Russakovsky, O., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*.

[30] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.