A Comparative Study of YOLOv8 and Faster R-CNN on Custom Object Detection Dataset

Guangzhi Su
Duke Kunshan University
gs285@duke.edu

Abstract

This paper presents a comparative analysis of two widely recognized object detection models, YOLOv8 and Faster R-CNN, using a custom dataset of 362 images annotated across four object classes: laptop, keyboard, mouse, and utensil. The dataset preparation involved unifying annotations to ensure consistency across both models. YOLOv8, known for its fast inference, and Faster R-CNN, recognized for its detection accuracy, were trained and evaluated based on detection accuracy, inference speed, and model size. Our experimental results indicate that YOLOv8 outperformed Faster R-CNN in most aspects, including inference time and overall detection accuracy, highlighting its suitability for simple object detection tasks. However, the performance gap may be influenced by the dataset's straightforward nature, which limited the display of Faster R-CNN's strengths. This study provides valuable insights into the trade-offs between speed and accuracy, aiding researchers and practitioners in selecting suitable models for specific object detection applications. My project page: https://github.com/GuangzhiSu/Object-Detection-Using-Faster_RCNN-and-YOLO

1 Introduction

Object detection is a fundamental task in computer vision, with widespread applications in areas such as autonomous driving (1), medical imaging (2), and industrial automation (3). The objective of object detection is to identify and localize multiple objects within an image, making it crucial for systems that require visual understanding. Recent advances have led to the development of highly effective object detection frameworks, among which the YOLO (You Only Look Once) series (4) and Faster R-CNN (Region-based Convolutional Neural Network) (5) are widely recognized for their performance and versatility.

The YOLO family of models is well known for its real-time speed and computational efficiency (6). YOLO treats object detection as a single-stage regression task, predicting both bounding boxes and class labels simultaneously in one pass through the network. This design enables fast inference, making YOLO ideal for applications that require quick object detection. However, YOLO models tend to struggle with small object detection and may have limited localization precision compared to more complex two-stage approaches (7). In contrast, Faster R-CNN follows a two-stage process (5). It first generates region proposals and then performs classification and bounding box refinement for each proposal. This approach offers high detection accuracy, especially in scenarios involving complex scenes with overlapping or densely packed objects. However, the added complexity results in higher computational costs, making Faster R-CNN slower than single-stage detectors like YOLO. These trade-offs highlight the importance of selecting the right model depending on the specific requirements for accuracy and inference speed.

In this project, we created a custom dataset containing 362 images, annotated across four classes: laptop, keyboard, mouse, and utensil. After annotating the dataset, we used it to train and validate both

Preprint. Under review.

YOLOv8 and Faster R-CNN models. The goal of this study was to compare the two models in terms of their accuracy and efficiency on our dataset. We evaluated the models using Average Precision (AP) across different Intersection-over-Union (IoU) thresholds in terms of different categories and recall metrics to assess how well each model detects objects consistently.

In our experiments, YOLOv8 demonstrated strong performance in terms of both detection accuracy and efficiency. It achieved higher mAP scores compared to Faster R-CNN, indicating better generalization to our dataset, which consists primarily of simple object captures with minimal clutter. In terms of inference speed, YOLOv8 outperformed Faster R-CNN, with a total processing time of approximately 3.5 milliseconds per image, compared to Faster R-CNN's 60.4 milliseconds per iteration. This makes YOLOv8 more suitable for applications requiring real-time predictions. Additionally, the model size of YOLOv8 (5.26 MB) was significantly smaller than that of Faster R-CNN (314.85 MB), reflecting the lightweight architecture of the YOLO family. These findings highlight that the simplicity of the dataset likely favored YOLOv8, minimizing its known challenges with small or overlapping objects. The comparison between the two models underscores the importance of matching model selection to the specific characteristics of the dataset and the practical requirements of the application.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work, focusing on the development and advancements of YOLO and Faster R-CNN models. Section 3 outlines the proposed methodology, detailing the data preparation, training procedures, and evaluation metrics for both models. Section 4 presents the experimental results, including comparisons of detection accuracy, inference speed, and model size. Finally, Section 5 concludes the study, summarizing the key findings and offering insights into the implications of the results for future work and practical applications.

2 Related Work

2.1 YOLO Model

YOLO (You Only Look Once) is a family of object detection models first introduced by Redmon et al. (4). Unlike traditional two-stage object detectors, such as Faster R-CNN (5), YOLO frames object detection as a single-stage task, where both object localization and classification are performed simultaneously within a single forward pass through the network. This design allows YOLO to achieve extremely fast inference speeds, making it suitable for real-time applications, such as video surveillance, autonomous driving, and robotics.

Over the years, successive versions of YOLO have introduced various improvements. YOLOv3 (8) incorporated a Feature Pyramid Network (FPN) to enhance detection at multiple scales and introduced residual connections for better feature propagation. YOLOv4 (6) further improved the architecture by integrating spatial pyramid pooling (SPP) and employing mosaic data augmentation, which increases robustness against variations in object appearances. These advancements led to significant improvements in both speed and accuracy, maintaining YOLO's relevance for real-time applications.

The latest version, YOLOv8, continues to refine the architecture by introducing a more efficient backbone network and adopting advanced optimization techniques, such as decoupled head architectures and adaptive anchor-free detection strategies. These innovations enable YOLOv8 to achieve state-of-the-art performance while preserving its low latency. However, despite these advances, YOLO models face challenges in detecting small, densely packed, or overlapping objects (9). This limitation makes them less effective in complex scenes compared to multi-stage detectors like Faster R-CNN, which rely on region proposals for precise localization (7). Nevertheless, YOLO remains one of the most popular choices for time-sensitive tasks, where fast inference is crucial.

2.2 Faster R-CNN

Faster R-CNN, introduced by Ren et al. (5), represents a significant advancement in region-based detection models. It builds on the earlier R-CNN (10) and Fast R-CNN (11) architectures by integrating a Region Proposal Network (RPN), which generates object proposals during training. This two-stage design enables Faster R-CNN to first identify Regions of Interest (RoIs) and then perform object classification and bounding box regression within each region. As a result, Faster R-CNN

achieves higher accuracy, especially when handling small or overlapping objects and cluttered scenes, although at the cost of longer inference times.

Faster R-CNN is widely used in fields where detection accuracy is prioritized over inference speed. Its applications span across various domains, including medical imaging (12), satellite imagery analysis (13), and document processing (14). The model's two-stage design allows for more detailed and precise object detection, making it particularly robust in environments with overlapping objects and complex backgrounds. However, this robustness comes with trade-offs; the increased computational complexity and need for more training iterations make Faster R-CNN less suitable for real-time applications.

Despite these limitations, Faster R-CNN remains one of the most accurate and reliable object detection models. It continues to be a favored choice in research and industrial applications where performance and precision are paramount, such as autonomous systems, visual inspection, and scene understanding tasks (7). The ongoing evolution of Faster R-CNN and its variants ensures its relevance in addressing modern challenges in computer vision.

3 Proposed Method

This section outlines the methods employed to implement and evaluate two object detection models, YOLOv8 and Faster R-CNN, using a custom dataset. The dataset consists of 362 images with annotations created through CVAT, covering four classes: laptop, mouse, keyboard, and utensils. The data preparation involved unifying annotations since three individuals independently labeled different object categories. In YOLOv8, the first column in each annotation file was adjusted to reflect unified class identifiers. For Faster R-CNN, the `category-id` field in the JSON files was updated to ensure consistency. Both models followed the same data split strategy, with 80% allocated for training and 20% for validation, ensuring balanced model development and evaluation. The following subsections describe the specific methodologies employed for each model.

3.1 YOLOv8

In this project, the lightweight YOLOv8n model was chosen for its balance of speed and accuracy. The training was conducted incrementally over 30 epochs, with progress monitored every 5 epochs to allow for timely adjustments if necessary. A batch size of 8 and an input image size of 640×640 pixels were used, optimizing memory usage while maintaining robust model performance.

Throughout the training process, key metrics such as mean average precision (mAP) at different IoU thresholds—mAP@0.5:0.95 and mAP@0.5—were monitored. These metrics provided insight into the model's detection accuracy across varying levels of overlap. Additionally, the training and validation losses were tracked to ensure convergence and detect potential overfitting or underfitting early on. Visual tools, including confusion matrices and bounding box predictions on validation images, were employed to further assess the model's performance, offering detailed insights into class-wise performance and misclassifications. These visualizations allowed for early identification of issues, ensuring that predictions aligned with the ground truth labels.

The inference speed, measured in milliseconds per image, was recorded to evaluate the model's efficiency. The trained model was temporarily saved to assess its size in megabytes (MB), confirming that it met deployment requirements. Finally, the model was exported to ONNX format to enhance portability and ensure compatibility with diverse deployment platforms.

3.2 Faster R-CNN

Faster R-CNN, a two-stage object detection framework, was also implemented for this project. This model, known for balancing accuracy and speed, first generates region proposals and then refines these proposals for final object detection. The implementation leveraged the Detectron2 library, utilizing a Faster R-CNN architecture with a ResNet-50 backbone and Feature Pyramid Network (FPN). Pre-trained weights from the COCO dataset were used to fine-tune the model for the custom dataset. To accommodate the four object classes in the dataset, the box predictor was reinitialized to output logits for five categories, including one for the background.

The training was carried out over 3000 iterations with a batch size of 2 and a learning rate of 0.001. Gradient clipping with a maximum value of 1.0 was applied to stabilize the training process and prevent exploding gradients. To monitor progress, a visualization hook was implemented, which randomly selected five validation images every 500 iterations and displayed their predictions. These visualizations provided real-time feedback, enabling adjustments during training as necessary.

Faster R-CNN was evaluated using the same mAP metrics as YOLOv8—mAP@0.5:0.95 and mAP@0.5—to ensure consistent comparison. Visual assessments, including confusion matrices and bounding box predictions, were used to further analyze the model's performance. The final trained model was saved to facilitate future inference or additional fine-tuning, emphasizing the importance of precise region proposals and accurate bounding box regression for complex object detection tasks.

4 Experiments

In this section, we present a comprehensive evaluation of the two object detection models used in this project: YOLOv8 and Faster R-CNN. The experiments are designed to assess the models' effectiveness across three key aspects: detection accuracy, inference speed, and model size. Each of these metrics plays a vital role in determining the applicability of the models in real-world scenarios. Detection accuracy, quantified through the mean average precision (mAP) metric, provides insight into the models' ability to correctly localize and classify objects within the dataset. Inference speed, measured as the time taken to process each image, reflects the models' suitability for time-sensitive applications, while model size gives an indication of the storage and memory requirements, which are crucial for deployment on resource-constrained devices. Through this evaluation, we aim to identify the strengths and limitations of YOLOv8 and Faster R-CNN, offering insights into their trade-offs and providing guidelines for their appropriate use in different applications.

4.1 Detection Accuracy

This subsection evaluates and compares the detection accuracy of Faster R-CNN and YOLOv8 on the custom dataset. The primary metric used for comparison is the mean Average Precision (mAP), which measures the model's ability to correctly detect objects across different Intersection over Union (IoU) thresholds.

Table 1: Comparison of mAP for YOLOv8 and Faster R-CNN.

Metric	**YOLOv8**	**Faster R-CNN**
mAP@0.5:0.95	0.641	0.521
mAP@0.5	0.858	0.709

Table 1 summarizes the key mAP results from both models. Faster R-CNN achieves an mAP@0.5:0.95 of 0.521, which indicates robust detection performance across multiple IoU thresholds. On the other hand, YOLOv8 surpasses Faster R-CNN with an mAP@0.5:0.95 of 0.641, demonstrating better detection capabilities across a range of IoU levels. Additionally, YOLOv8 achieves superior mAP@0.5 performance at 0.858 compared to 0.709 in Faster R-CNN, reflecting YOLOv8's strength in high IoU scenarios.

Table 2: Per-category AP Comparison between YOLOv8 and Faster R-CNN.

Category	**YOLOv8 AP**	**Faster R-CNN AP**
Laptop	0.79	0.682
Mouse	0.783	0.597
Keyboard	0.703	0.640
Utensil	0.290	0.165

Table 2 provides a per-category breakdown of AP values. YOLOv8 performs particularly well for laptops and mice, achieving APs above 75%, whereas Faster R-CNN exhibits lower AP values for the utensil class, demonstrating a potential weakness in detecting objects with high variability or complex backgrounds.

In addition to the quantitative metrics, the normalized confusion matrix in Figure 1 and the sample visualizations from the validation set in Figure 2 provide further insights into the challenges faced by the models. Specifically, the confusion matrix reveals several key patterns:

- **High Precision for Laptop and Mouse**: Both models perform exceptionally well for these classes, with YOLOv8 achieving AP values over 75% for laptops and 78% for mice.
- **Challenges in Keyboard Detection**: A portion of keyboards is misclassified as laptops or background. This likely arises from visual similarities in shape and texture between keyboards and laptops.
- **Significant Misclassifications in Utensils**: Many utensils are incorrectly identified as background or other objects, likely due to their small size and the presence of cluttered or complex backgrounds. The visualizations in Figure 2 highlight this issue, showing several instances where utensils were either missed or detected with low confidence.

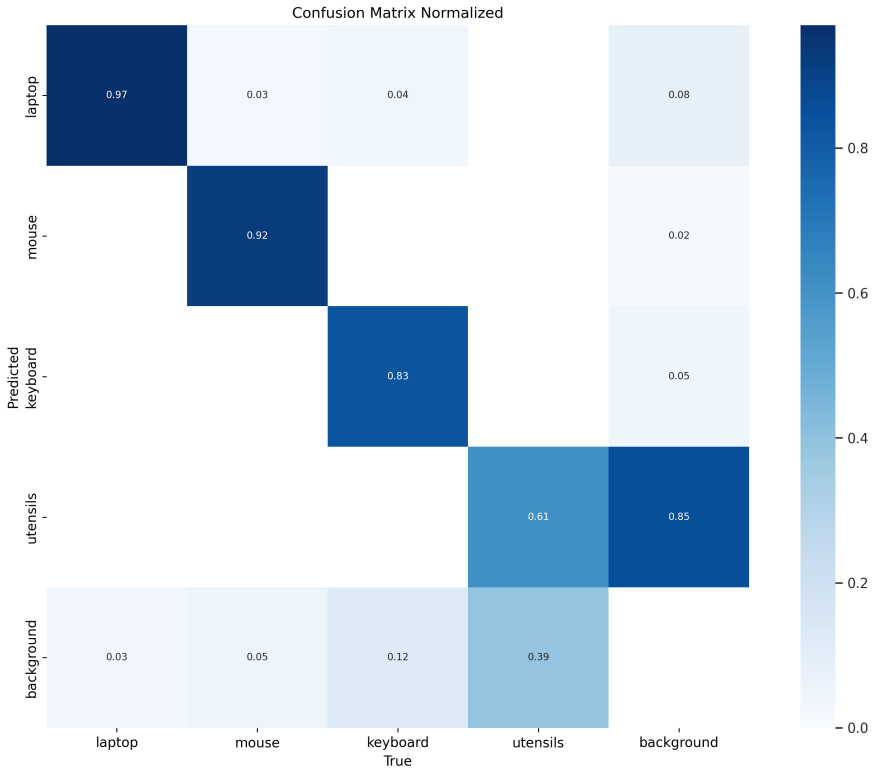

Figure 1: Normalized confusion matrix for the custom dataset. The matrix shows the model's classification performance, with diagonal values representing correct predictions and off-diagonal values indicating misclassifications.

In summary, while YOLOv8 demonstrates superior detection accuracy overall, the confusion matrix and validation set visualizations reveal that both models struggle with detecting small objects such as utensils in cluttered environments. These insights underline the importance of further optimization and dataset refinement for improving the detection performance on challenging object classes.

4.2 Inference Time

This subsection compares the inference time of YOLOv8 and Faster R-CNN, two models with distinct architectural designs optimized for different priorities. YOLO models are known for their efficiency and speed, making them highly suitable for real-time applications, while Faster R-CNN generally prioritizes accuracy over inference speed. Table 3 provides a comparison of the inference-related metrics for both models.

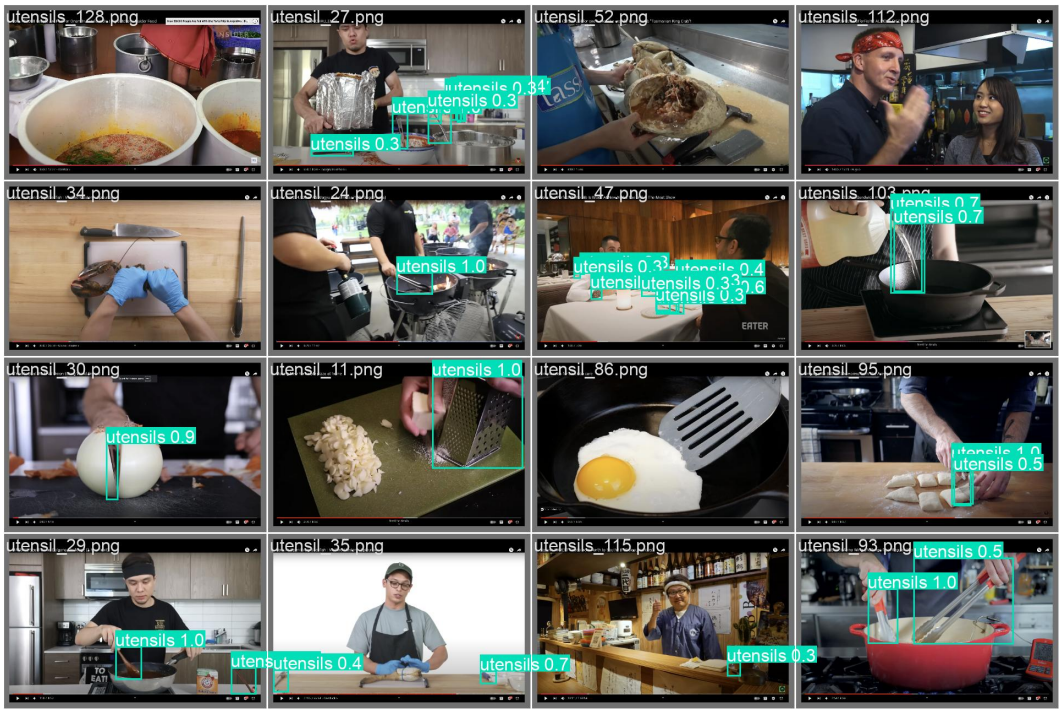

Figure 2: Visualization of predictions on the validation set. The images show that while utensils are correctly detected in some cases, others are missed or assigned low confidence scores due to their small size and complex backgrounds.

Table 3: Comparison of Inference Time for YOLOv8 and Faster R-CNN.

Metric	**YOLOv8 (ms/img)**	**Faster R-CNN (ms/img)**
Preprocessing Time	0.2	0.5
Inference Time	2.2	29.55
Loss Computation	0.0	–
Postprocessing Time	1.1	0.15
Total Inference Time	3.5	30.20

The results in Table 3 align with conventional knowledge regarding the performance characteristics of YOLO and Faster R-CNN models. As expected, YOLOv8 exhibits significantly lower inference time per image, taking approximately 3.5 milliseconds in total, including preprocessing, inference, and postprocessing. This highlights YOLOv8's suitability for time-sensitive applications, such as real-time object detection in video streams, where rapid predictions are crucial.

In contrast, Faster R-CNN requires 30.2 milliseconds per image, which includes data loading, inference, and evaluation steps. The longer inference time is attributed to Faster R-CNN's two-stage detection framework, where the region proposal network (RPN) adds computational overhead by first generating candidate regions of interest before performing classification and bounding box refinement. Additionally, the ResNet-50 backbone with a Feature Pyramid Network (FPN) contributes to the higher computational complexity of Faster R-CNN, offering more detailed feature extraction at the expense of latency.

This comparison illustrates the trade-offs between the two architectures. While Faster R-CNN may be more suited for tasks requiring higher detection accuracy and precision, YOLOv8's lightweight architecture makes it more appropriate for real-time applications, where speed is prioritized over marginal gains in accuracy.

4.3 Model Size

This subsection provides a comparison of the model sizes for YOLOv8 and Faster R-CNN. Table 4 summarizes the key statistics for each model, including the total number of parameters, FLOPs, and the final model size in megabytes.

Table 4: Comparison of Model Size between YOLOv8 and Faster R-CNN.

Model	Parameters	FLOPs	Model Size (MB)
YOLOv8	2,685,148	6.8 GFLOPs	5.26 MB
Faster R-CNN	-	-	314.85 MB

Table 4 highlights a significant difference in the size of the two models. As expected, Faster R-CNN has a considerably larger model size, occupying 314.85 MB, while YOLOv8's size is only 5.26 MB. This difference aligns with general expectations for the following reasons:

- **Model Architecture:** Faster R-CNN employs a two-stage architecture, including both a Region Proposal Network (RPN) and classification heads, which introduces additional layers and parameters. In contrast, YOLOv8 follows a single-stage architecture, which is more lightweight by design.
- **Backbone Complexity:** Faster R-CNN typically uses deep backbones like ResNet-50 with Feature Pyramid Networks (FPN), contributing to the high parameter count and model size. YOLOv8, on the other hand, uses an optimized backbone with fewer parameters, ensuring computational efficiency.
- **Parameters and FLOPs:** YOLOv8 has 2.68 million parameters and requires 6.8 GFLOPs, reflecting its design for speed and real-time performance. Faster R-CNN, with significantly more layers and complex processing, results in a model that is both larger and computationally heavier.
- **Intended Use Cases:** The smaller size of YOLOv8 makes it suitable for real-time applications such as video surveillance or mobile deployment. Conversely, Faster R-CNN's larger size aligns with applications requiring high detection precision, such as medical imaging or satellite imagery analysis, where speed is less critical.

This comparison demonstrates the trade-offs between the two models. YOLOv8 excels in scenarios where speed and storage are crucial, while Faster R-CNN provides superior accuracy at the cost of higher computational requirements and memory usage.

5 Conclusion

This project implemented and evaluated YOLOv8 and Faster R-CNN on a custom dataset consisting of four object categories: laptop, mouse, keyboard, and utensils. The experiments revealed that YOLOv8 outperformed Faster R-CNN in most aspects, achieving higher mAP and faster inference speed, with a notably smaller model size. These results diverge from typical expectations, as YOLO models are often known to struggle with detecting small or overlapping objects. However, the nature of the dataset—comprising simple, straightforward captures of objects with minimal overlap or clutter—minimized the complexity that usually challenges YOLO models. As a result, the anticipated strengths of Faster R-CNN, such as its precision with complex scenes, were not fully reflected. This outcome highlights the importance of dataset characteristics in model performance and suggests that YOLOv8, with its lightweight and efficient design, can be a more effective choice in scenarios with clear object boundaries and uncomplicated backgrounds.

References

[1] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[2] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[3] C. Liu, Y. Ying, J. Pan, and Y. Li, "Machine vision technologies for autonomous agricultural operations: A review," *Computers and electronics in agriculture*, vol. 151, pp. 226–241, 2018.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[5] 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 (NeurIPS)*, pp. 91–99, 2015.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," in *arXiv preprint arXiv:2004.10934*, 2020.

[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7310–7311, 2017.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," in *arXiv preprint arXiv:1804.02767*, 2018.

[9] Z. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[12] N. Tajbakhsh, L. Jeyaseelan, Y. Li, J. N. Chiang, Z. Wu, and X. Ding, "Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation," *Medical Image Analysis*, vol. 63, p. 101693, 2020.

[13] G. Cheng, J. Han, and X. Lu, "A survey of recent advances in object detection with deep learning," *Frontiers of Computer Science*, vol. 10, no. 2, pp. 167–176, 2016.

[14] S. Appalaraju and A. Kundu, "Docformer: End-to-end transformer for document understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 993–1003, 2021.