CHARLES DARWIN UNIVERSITY
Sydney Campus

Supervised by:
Yakub Sebastian

Pruning Model for Corrosion Detection Using Yolov8n

Submitted by:
Maheshwor Tiwari
Bibek Dhakal
Yasin Hossain

A thesis presented as part of the Master of Data Science's program

(SDASC2-2024)

S224 PRT840 IT THESIS

October 26, 2024

Abstract

Corrosion detection in industrial settings is critical for ensuring the safety and longevity of infrastructure in sectors such as oil and gas, transportation, and maritime. Traditional methods of corrosion inspection, such as visual assessments, are labor-intensive, error-prone, and inefficient for large-scale applications. Deep learning models, particularly the YOLO (You Only Look Once) family of object detectors, offer a promising alternative for automating corrosion detection tasks. However, deploying these models in resource-constrained environments, such as drones or edge devices, presents significant challenges due to their computational complexity and memory requirements.

This research explores the optimization of Yolov8n, the efficient model in the YOLO family, through the application of structured and unstructured pruning techniques. Pruning aims to reduce the model size and improve inference speed while maintaining detection accuracy, making the models more suitable for real-time applications in constrained environments. The study demonstrates that structured pruning, based on L1-norm filter selection, achieved an 11.39% reduction in model size and a 39.34% improvement in inference time, with only a minor 7.43% drop in mAP@50-95. Unstructured pruning methods, while offering some improvements in model efficiency, resulted in greater accuracy loss, particularly when pruning levels were too aggressive.

Fine-tuning after pruning proved essential for restoring detection performance, especially for unstructured methods. The pruned Yolov8n models demonstrated improved efficiency, making them more viable for real-time corrosion detection tasks in resource-limited settings. The findings from this research contribute to the development of more efficient and deployable deep learning models for industrial inspection applications, particularly in environments where computational resources are limited.

Keywords: Corrosion detection, Yolov8n, structured pruning, unstructured pruning, deep learning, model optimization

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Dr. Yakub Sebastian, for his invaluable guidance, support, and encouragement throughout this research. His expertise has been instrumental in the successful completion of this work.

We would also like to thank Dr. Thuseethan Selvarajah, the Unit Coordinator of PRT840, for his role as the coordinator and for overseeing the program. In addition, we extend our appreciation to Dr. Abdallah Al Tawara. His guidance has greatly contributed to our understanding of the research and writing process.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the key motivations and context for the study, focusing on the importance of corrosion detection in industrial settings and the challenges associated with deploying deep learning models, like Yolov8n, in resource-constrained environments. It outlines the main problem addressed by the research—optimizing Yolov8n for real-time applications through pruning techniques—and presents the objectives and research questions guiding the study. The chapter concludes by highlighting the research contributions and providing an overview of the thesis structure.

1.1 Context and Motivation

Corrosion detection is critical in industries such as oil and gas, maritime, transportation, and infrastructure, where early identification of corrosion can prevent structural failures, reduce maintenance costs, and ensure safety. Traditional methods of corrosion detection, such as manual visual inspections and non-destructive testing, while effective, are labor-intensive, time-consuming, and prone to human error, particularly in hard-to-reach or hazardous areas (Smith & Doe, 2020; Roberts & Lee, 2020). Consequently, the need for automated, efficient, and accurate detection systems has grown.

Deep learning, particularly through object detection models like the YOLO (You Only Look Once) family, has emerged as a solution to automate and improve the accuracy and speed of corrosion detection. YOLO models are known for their real-time detection capabilities and balance between accuracy and speed, making them ideal for industrial applications where rapid decision-making is required (Brown & Lee, 2021; Redmon, Divvala, Girshick, & Farhadi, 2016). However, deploying such models in resource-constrained environments, such as drones or edge devices, remains a challenge due to the significant computational and memory requirements of these models (R. Patel & Garcia, 2019).

1.2 Problem Statement

The primary issue with deploying deep learning models like Yolov8n in real-time applications is their computational intensity, which limits their use in resource-constrained environments like drones, edge devices, or small industrial sensors. While these models deliver high accuracy and speed, their size and the computational resources they require for inference make them impractical for large-scale industrial use without optimization.

A promising solution to this problem is the use of *pruning techniques*, which reduce the model's size and complexity by removing less important weights and parameters without significantly compromising performance (Chen & Zhang, 2022). However, while pruning has been explored in earlier YOLO models (e.g., YOLOv3, YOLOv4), there is limited research on the application of pruning to Yolov8n, particularly for real-time corrosion detection tasks.

1.3 Research Objectives

The objectives of this research are as follows:

1. To investigate the impact of both structured and unstructured pruning techniques on the performance of the Yolov8n model in corrosion detection tasks.
2. To evaluate the balance between model size reduction, inference speed, and detection accuracy, ensuring that the pruned models remain effective in detecting corrosion.
3. To optimize the Yolov8n model for deployment in resource-constrained environments (e.g., drones, edge devices), enhancing real-time performance without significant loss of detection precision or recall.
4. To provide insights into the trade-offs involved in applying different pruning techniques to deep learning models used in industrial applications.

1.4 Research Questions

To achieve the stated objectives, this research seeks to answer the following questions:

1. **How do structured and unstructured pruning techniques impact the performance of the Yolov8n model in corrosion detection tasks?**
2. **What is the balance between model size reduction, inference speed, and detection accuracy after pruning, and how can this be optimized for real-time corrosion detection?**
3. **Can the Yolov8n model be optimized for deployment in resource-constrained environments (e.g., drones, edge devices) without significant loss of detection precision or recall?**
4. **What are the trade-offs involved in applying different pruning techniques to deep learning models used in industrial applications, such as corrosion detection?**

1.5 Research Contributions

This study aims to contribute to the field by:

- Developing a framework for applying structured and unstructured pruning to Yolov8n, optimizing its use in corrosion detection tasks.
- Empirically evaluating the impact of pruning techniques on model size, speed, and detection accuracy, particularly focusing on their application in real-time, resource-constrained environments.
- Providing practical insights and recommendations for deploying pruned models in industrial settings, such as infrastructure maintenance, oil and gas pipeline inspections, and maritime operations.

1.6 Thesis Structure

The remainder of this thesis is structured as follows:

- **Chapter 2: Related Work** - A review of existing research on deep learning in corrosion detection, advancements in YOLO models, and the application of pruning techniques for model optimization.
- **Chapter 3: Methodology** - A detailed description of the dataset used, the Yolov8n model architecture, the pruning techniques applied, and the evaluation metrics for model performance.

- **Chapter 4: Results** - Presentation of the results from initial model training, hyperparameter tuning, and the performance of pruned models using structured and unstructured methods.
- **Chapter 5: Discussion** - Analysis of the results, including the trade-offs between model size, speed, and detection accuracy, with a focus on real-time deployment scenarios.
- **Chapter 6: Conclusion and Future Work** - A summary of the key findings, contributions of the study, and suggestions for future research in optimizing deep learning models for real-time industrial applications.

In this chapter, we introduced the context, problem statement, research objectives, and questions guiding this study on optimizing Yolov8n for real-time corrosion detection. The next chapter will review related work, covering advances in corrosion detection, deep learning models, and pruning techniques.

Chapter 2

Related Work

This chapter reviews the current literature on deep learning applications in corrosion detection, focusing on the evolution of object detection models like YOLO and the challenges of optimizing these models for resource-constrained environments. It highlights the importance of model optimization techniques, such as pruning, and explores how they have been applied to improve the deployment of deep learning models for industrial applications. Additionally, this chapter identifies the research gaps, particularly in the use of Yolov8n for real-time corrosion detection, and positions this study within the broader context of deep learning advancements in industrial inspections.

2.1 Corrosion Detection: Importance and Challenges

Corrosion is a pervasive issue affecting industries such as infrastructure, maritime, aerospace, and oil gas. It leads to significant safety risks, increased maintenance costs, and reduced operational lifespans of critical structures (Smith & Doe, 2020). In the oil and gas sector alone, corrosion-related failures account for billions of dollars in annual losses due to equipment malfunctions, accidents, and operational shutdowns (Miller & Davis, 2021). Therefore, the timely detection of corrosion is vital for preventing catastrophic failures and improving safety and reliability.

2.1.1 Conventional Methods for Corrosion Detection

Traditional corrosion detection methods, including visual inspections and ultrasonic testing, have been widely used in industries for decades. However, these approaches are often time-consuming, subjective, and prone to human error, especially in hard-to-reach or hazardous environments (Roberts & Lee, 2020). Manual inspections also require significant manpower and are not always feasible for large-scale industrial facilities, such as oil rigs or pipelines. This has driven demand for automated and efficient corrosion detection solutions that can overcome the limitations of manual methods (Smith & Brown, 2021).

2.2 Deep Learning in Corrosion Detection and Computer Vision

Deep learning, particularly through the use of convolutional neural networks (CNNs), has revolutionized computer vision tasks by enabling automated feature extraction and pattern recognition from large datasets. This has led to significant improvements in image classification, object detection, and segmentation tasks (Krizhevsky, Sutskever, & Hinton, 2012). CNNs are especially effective in learning spatial hierarchies, such as edges, textures, and shapes, which are critical for detecting corrosion in varying environments (Simonyan & Zisserman, 2014; LeCun, Bengio, & Hinton, 2015).

2.2.1 Object Detection Models in Corrosion Detection

Object detection models, such as Region-CNN (R-CNN), SSD, and YOLO (You Only Look Once), have been widely adopted for real-time detection tasks. These models have shown promise in various industrial applications, including defect detection and corrosion monitoring. R-CNN-based models employ a two-stage process of proposing regions of interest (ROIs) and classifying them, whereas single-shot detectors like SSD and YOLO perform both tasks simultaneously, significantly enhancing detection speed (Girshick, Donahue, Darrell, & Malik, 2014; W. Liu et al., 2016).

2.3 The Evolution of YOLO Models

2.3.1 Introduction to YOLO (You Only Look Once)

YOLO (You Only Look Once) is one of the most influential real-time object detection algorithms. Its key advantage lies in its ability to predict both the location and classification of objects in a single forward pass through the network, unlike region-based methods that require multiple stages (Redmon et al., 2016). YOLO's grid-based approach allows for rapid detection, making it ideal for real-time applications like corrosion detection, where timely interventions are critical.

2.3.2 Progression of YOLO Models

Since its introduction, the YOLO family has seen multiple iterations, each offering improvements in speed, accuracy, and model efficiency. YOLOv1 introduced a grid-based object detection framework, which was further refined in YOLOv2 and YOLOv3 with the use of anchor boxes and multi-scale detection, improving performance for detecting objects of varying sizes (Redmon & Farhadi, 2017, 2018). YOLOv3, in particular, struck a balance between accuracy and speed, making it a popular choice for industrial applications. Later versions, including YOLOv4 and YOLOv5, incorporated advanced techniques such as CSPDarknet, PANet, and Mosaic data augmentation, enhancing feature extraction and boosting mean average precision (mAP) (Bochkovskiy, Wang, & Liao, 2020; Jocher, 2020). YOLOv6 and YOLOv7 further reduced model size while maintaining accuracy, making them suitable for deployment in resource-constrained environments, such as drones and edge devices (Mei & et al., 2022; C.-Y. Wang, Bochkovskiy, & Liao, 2022). Yolov8n, the latest version, has improved both speed and accuracy, making it particularly suitable for real-time corrosion detection (Ultralytics, 2023).

YOLOv10, the emerging version, promises further advancements in performance with potential improvements in both speed and accuracy. However, due to its early-stage development and stability concerns encountered during initial testing (as detailed in Appendix A), YOLOv10 was not used in this study, and Yolov8n was selected for its proven reliability in real-time corrosion detection.

2.4 Applications of YOLO in Corrosion Detection

YOLO models have been increasingly adopted for automated industrial inspections, particularly in detecting corrosion, structural flaws, and other defects. YOLOv3, for instance, has been successfully applied in detecting rust on metallic surfaces, while YOLOv4 has been used for real-time monitoring of pipeline corrosion (N. Patel & Kumar, 2019; Singh & Khan, 2022). The deployment of YOLO models on resource-constrained devices, such as drones and handheld cameras, has further enabled inspections in challenging environments, reducing the need for manual intervention (Kumar & Mehta, 2020; Johnson & Allen, 2021). However, challenges remain in balancing detection accuracy and model efficiency. Corrosion detection is inherently complex due to the varied shapes, textures, and colors of corrosion, which can lead to misclassifications if models are not adequately optimized (Lee & Zhao, 2020b). This necessitates the exploration of model optimization techniques such as pruning, quantization, and knowledge distillation.

2.5 Model Optimization and Pruning Techniques

To address the challenges of deploying large object detection models in resource-constrained environments, various optimization techniques have been explored. *Pruning* involves the removal of unnecessary weights or neurons from a model to reduce its size and computational requirements, while maintaining accuracy. *Quantization* reduces model precision to make it more efficient, and *knowledge distillation* transfers knowledge from a larger model to a smaller one (Jacob et al., 2018; Hinton, Vinyals, & Dean, 2015).

2.5.1 Pruning in Object Detection Models

Pruning has been applied extensively to reduce the complexity of object detection models, particularly for deployment on edge devices. Early work focused on pruning R-CNN-based models, such as Faster R-CNN, using both structured and unstructured pruning to reduce model size and computational time while maintaining detection accuracy (Z. Shen, Yan, Liu, & Liu, 2021). Single-shot detectors, like SSD and YOLO, have also benefited from pruning techniques. Studies have demonstrated that structured pruning, which removes entire filters or layers, is more compatible with hardware acceleration and yields better results in terms of inference speed than unstructured pruning, which removes individual weights based on their magnitude (F. Wang, He, & Liu, 2020; Nguyen & Tran, 2021).

2.6 Challenges in Pruning for Object Detection

While pruning offers clear benefits in reducing model size, it presents several challenges, particularly for object detection tasks. Aggressive pruning can lead to a significant drop in accuracy, especially in tasks requiring precise localization and classification (Y. Li, Sun, & Wang, 2021). Moreover, unstructured pruning can introduce irregular sparsity, making it less effective on hardware optimized for dense matrix operations. Structured pruning, which maintains architectural regularity, can mitigate this issue but must be applied carefully to avoid loss of performance (Yu & Zhang, 2019; H. Li, Kadav, Durdanovic, Samet, & Graf, 2016). Another challenge is determining the optimal level of pruning. Over-pruning can lead to irrecoverable performance loss, even with fine-tuning, and traditional metrics like accuracy or mAP may not fully capture the trade-offs in model efficiency and speed gains (W. Shen & Zhang, 2018; Z. Liu, Sun, & Yan, 2020).

2.7 Contribution of this Study

Despite the extensive research on pruning techniques, there remains a gap in applying pruning to newer versions of YOLO models, such as Yolov8n, particularly for real-time corrosion detection tasks. Most research has focused on earlier versions like YOLOv3 and YOLOv4, leaving a need for more detailed studies on pruning techniques specifically designed for the Yolov8n model (H. Wang & Liu, 2022). Additionally, the specific challenges of optimizing models for corrosion detection, which involve handling the irregular and varied nature of corrosion patterns, have not been fully addressed (Lee & Zhao, 2020a). This study aims to bridge this gap by evaluating both structured and unstructured pruning techniques in optimizing the Yolov8n model for

corrosion detection tasks. By empirically analyzing the impact of these pruning methods on model size, inference speed, and detection accuracy, this research will provide valuable insights for deploying pruned YOLO models in resource-constrained environments like drones and edge devices, enhancing the capabilities of real-time corrosion detection systems (Zhang & Xu, 2021).

This chapter reviewed existing literature on deep learning for corrosion detection, the evolution of YOLO models, and pruning techniques for model optimization. The following chapter will describe the methodology used to implement Yolov8n, including dataset preparation, model architecture, and pruning techniques.

Chapter 3

Methodology

This chapter outlines the process used for training, optimizing, and pruning the Yolov8n model for corrosion detection. The methodology includes the dataset utilized, the model architecture, experimental setup, and the pruning techniques employed.

3.1 Dataset

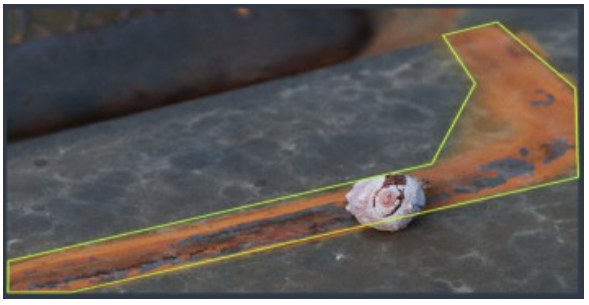
The dataset used in this study was sourced from Roboflow (*Roboflow: Organize, Label, and Prepare Your Image Data for Training*, n.d.), a platform designed for managing and preprocessing machine learning datasets. The dataset consists of 388 images, including both corrosion and non-corrosion instances, with the majority labeled as corrosion.

3.1.1 Dataset Download and Preprocessing

The dataset, consisting of corrosion and non-corrosion images, was accessed and downloaded using the Roboflow API, which provided the images in a format compatible with Yolov8n. The dataset was exported directly in the Yolov8n format, ensuring that each image was accompanied by manually created bounding box annotations. After preprocessing in Roboflow, the dataset was stored in Google Drive to facilitate easy access during training and compatibility with Google Colab. Prior to downloading, the following preprocessing steps were applied within the Roboflow platform to prepare the dataset for training:

- **Auto-Orient**: Automatically corrected any orientation issues in the images.
- **Resizing**: All images were resized to 640x640 pixels to ensure consistency with the Yolov8n model's input size requirements.
- **Annotation Format**: Images were manually annotated using both rectangle and polygon formats, with the majority using rectangle annotations to create bounding boxes around the areas of corrosion.

(a) Rectangle Annotation



(b) Polygon Annotation

Figure 3.1: Examples of different annotation formats applied to the corrosion dataset: (a) Rectangle bounding boxes, and (b) Polygon-based annotations.

No additional augmentations, such as rotations or flips, were applied within Roboflow, as the goal was to maintain the natural variability of the dataset and avoid introducing artificial distortions. Once these preprocessing steps were completed, the dataset was downloaded to Google Drive for use in the model training process.

3.1.2 Dataset Split

To ensure effective training and evaluation, the dataset was divided into the following subsets:

- **Training set**: 70.6% of the images were used to train the model.
- **Validation set**: 20.1% of the images were reserved for validating the model during training.
- **Test set**: 9.3% of the images were set aside for the final model evaluation.

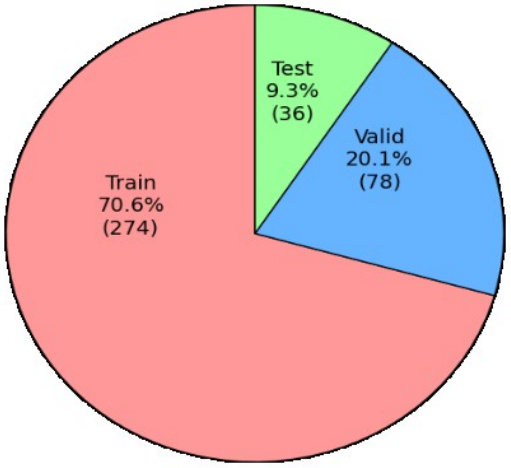


Figure 3.2: Dataset split for training, validation, and testing.

3.2 Model Architecture

The model chosen for this study is Yolov8n, a lightweight variant of the YOLO (You Only Look Once) family, optimized for real-time object detection in resource-constrained environments (?, ?).

3.2.1 Initial Model Training

The initial model was trained using the following configuration:

- **Image size**: 640x640 pixels
- **Batch size**: 32
- **Learning rate**: 0.001
- **Optimizer**: Adam

- **Number of epochs**: 100

This configuration was chosen to establish a baseline performance before applying pruning and optimization techniques. Once the baseline was established, hyperparameter tuning was conducted to further optimize the model's performance.

3.2.2 Hyperparameter Tuning

Hyperparameter tuning was performed using Ray Tune, which explored different configurations for batch size, learning rate, image size, and optimizer type. Multiple trials were run with different hyperparameter combinations, and the configuration that produced the best mAP50 was selected for further pruning and fine-tuning. The following hyperparameters were tuned:

- **Image size**: 416 vs. 640 pixels
- **Batch size**: 16, 32, and 64
- **Learning rate**: A range from 0.0001 to 0.001
- **Optimizers**: Tested SGD, Adam, and AdamW

Process: The tuning process involved 43 trials, each with a unique configuration from the defined search space. A FIFO scheduler (First-In-First-Out) was used to manage the order of trials, and the model training utilized GPU resources to expedite the experiments. The best configuration was selected based on the highest mAP@50 achieved during the trials establishing a solid foundation for subsequent model refinements, including pruning. After identifying the best hyperparameters, the final training was conducted using 500 epochs with early stopping.

3.3 Experimental Setup

The experimental setup utilized both CPU and GPU resources for training and pruning the Yolov8n model, using a combination of hardware and software tools.

3.3.1 Hardware

- **NVIDIA Tesla T4 GPU**: The primary GPU used for training was the Tesla T4 with 16 GB of memory, accessed via Google Colab. The system used CUDA version 12.2 and NVIDIA driver version 535.104.05 to accelerate training and reduce computation time during hyperparameter tuning and pruning (*NVIDIA Tesla T4 GPU for Deep Learning and Machine Learning Tasks*, n.d.; Colab, n.d.).
- **Google Colab**:
  - **Free Version**: Used for shorter training sessions, but frequent restarts were needed due to time limits on GPU access (Colab, n.d.).

  - **Pro Version**: Occasionally used for longer sessions, allowing extended GPU access but with additional costs (Colab, n.d.).

- **CPU Setup**: Used for initial experiments and small-scale tasks. However, CPU-based training was impractical for larger experiments due to significantly longer training times.

3.3.2 Software and Libraries

The following key software and libraries were crucial for the implementation of Yolov8n and the pruning process:

- **Python**: The primary programming language used for model development and training (*Python Programming Language*, n.d.).
- **PyTorch (v2.4.1)**: The core deep learning framework that enabled GPU acceleration and model building (PyTorch, 2024).
- **CUDA 12.2**: Vital for GPU acceleration, utilized in training with the Tesla T4 GPU (*CUDA Toolkit Documentation*, n.d.).
- **Ray Tune**: Employed for efficient hyperparameter tuning, a critical step in model optimization (Tune, n.d.).
- **Roboflow**: Used for dataset management, preprocessing, and annotation export in Yolov8n format, facilitating smooth data handling and preparation (*Roboflow: Organize, Label, and Prepare Your Image Data for Training*, n.d.).
- **Ultralytics THOP (v2.0.9)**: For calculating FLOPs and model parameters (THOP, n.d.).
- **Numpy, OpenCV, Pandas**: Used for data manipulation, image preprocessing, and dataset management (Numpy, n.d.; OpenCV, n.d.; Pandas, n.d.).
- **Matplotlib**: Essential for visualizing model performance and key metrics (Matplotlib, n.d.).

3.4 Pruning Techniques

In this study, two pruning techniques— **unstructured pruning** and **structured pruning** —were applied to reduce the size and complexity of the Yolov8n model while maintaining detection accuracy. Structured pruning removes entire filters or channels from the convolutional layers based on their **L1-norm values**, with less important filters pruned incrementally in 10% steps, followed by fine-tuning to recover performance. Unstructured pruning, on the other hand, removes individual weights based on their magnitude or gradient

Detailed discussions of pruning techniques and their results are provided in chapter 4. Despite hardware limitations, effective pruning strategies were employed to reduce the model's size and improve efficiency for real-time corrosion detection applications.

3.5 Evaluation Metrics

The performance of both pruned and unpruned Yolov8n models was evaluated using the following metrics:

- **mAP50**: Mean Average Precision at a 50% Intersection over Union (IoU) threshold.
- **mAP50-95**: Mean Average Precision across multiple IoU thresholds ranging from 50% to 95%.
- **Precision**: The ratio of true positive detections to all positive predictions.
- **Recall**: The ratio of true positive detections to all actual positive instances.
- **Inference Time**: The average time taken to process a single image.
- **Frames Per Second (FPS)**: The number of images the model can process per second.

These metrics were tracked during both the initial training phase and after applying pruning techniques to assess how pruning impacted both model size and accuracy.

In this chapter, we outlined the methodology for training, hyperparameter tuning, and applying pruning techniques to the Yolov8n model for corrosion detection. The next chapter will present the results from initial training, hyperparameter tuning, and pruning experiments.

Chapter 4

Model Pruning

This chapter explores the techniques of pruning applied to optimize the Yolov8n model for corrosion detection. It introduces pruning as a method to reduce model complexity by removing unnecessary parameters, thus enhancing efficiency while maintaining accuracy. Both unstructured and structured pruning techniques are discussed, detailing their respective processes and benefits. The chapter provides an in-depth look at the specific pruning methods used in this study, the stepwise approach to structured pruning, and the evaluation criteria applied to assess the impact on model performance.

4.1 Introduction to Pruning

Pruning is a widely used technique in deep learning aimed at reducing the size and complexity of a neural network by eliminating redundant or less important parameters. The core idea behind pruning is to improve the model's efficiency without significantly compromising its performance. By selectively removing certain weights or filters, pruning optimizes the computational and memory demands of the model. This process is particularly important when deploying deep learning models on platforms with limited computational power, where both model size and inference time must be minimized (Han, Pool, Tran, & Dally, 2015; Deepgram, 2023). Reducing the size of a model makes the model more suitable for deployment on edge devices (Deepgram, 2023). However, the main challenge is finding the right balance between model simplification and maintaining predictive accuracy.

4.1.1 Types of Pruning

Pruning techniques are typically categorized into two main types: unstructured pruning and structured pruning.

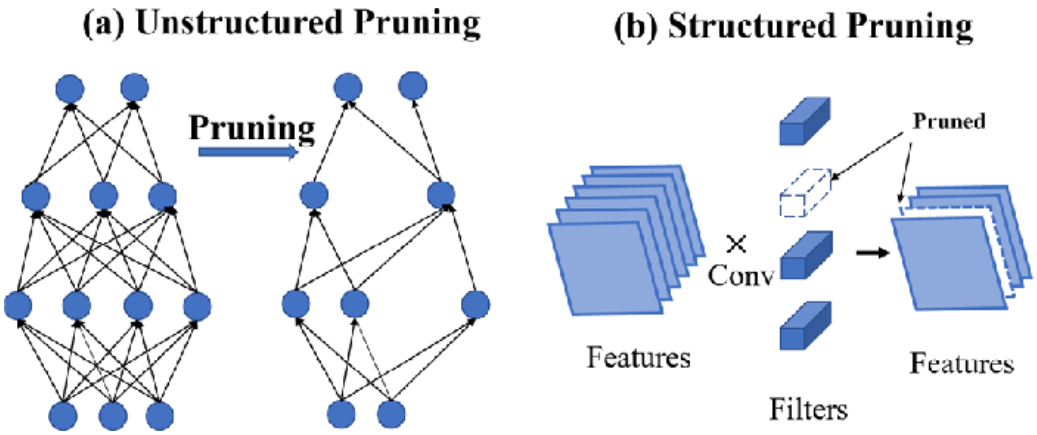


Figure 4.1: Schematic illustration of two types of network pruning methods based on granularity. (a) In unstructured pruning, individual neuron connections are pruned; (b) an example of structured pruning where entire filters are pruned to simplify the model (Chen et al., 2021).

4.1.1.1 Unstructured Pruning

Unstructured pruning involves the removal of individual weights that are deemed unnecessary based on criteria such as their magnitude. Weights with smaller magnitudes are pruned as they contribute minimally to the model's predictions (Molchanov, Tyree, Karras, Aila, & Kautz, 2019). This method allows for flexible reduction of model complexity without changing the network's overall structure (Open Data Science, 2023). However, the irregular sparsity it introduces can make it less efficient on hardware optimized for dense matrix operations.

4.1.1.2 Structured Pruning

Structured pruning, by contrast, removes entire filters, channels, or layers from a neural network. This approach is more compatible with hardware acceleration because it preserves the regularity

of the network structure, reducing the number of computations during inference (H. Li, Kadav, Durdanovic, Samet, & Graf, 2017; Datature, 2023). Structured pruning often utilizes metrics like the L1-norm of filters to identify and remove the least important elements (He, Kang, Dong, Fu, & Yang, 2017). The result is a model that maintains performance while offering predictable improvements in speed and memory efficiency, making it particularly suitable for real-time systems.

4.1.2 Benefits of Pruning

- **Model Size Reduction:** Pruning significantly reduces the memory footprint of a model, making it more suitable for deployment on resource-limited devices like mobile phones and edge devices (Datature, 2023).
- **Improved Inference Speed:** By reducing the number of computations required during inference, pruned models can achieve faster prediction times, which is crucial for time-sensitive applications (UbiOps, 2023).
- **Hardware Compatibility:** Structured pruning aligns deep learning models with modern hardware architectures, enabling more efficient processing (Deepgram, 2023).

While structured pruning can offer tangible improvements in performance, unstructured pruning remains widely used due to its flexibility in reducing model complexity while preserving the network's overall structure (Open Data Science, 2023).

4.2 Pruning Techniques Used in This Work

4.2.1 Unstructured Pruning

Unstructured pruning involves selectively removing individual weights within a neural network based on their relative importance. This approach allows for a fine-grained reduction of parameters while retaining the overall structure of the network (Han et al., 2015). It is particularly useful when exploring different levels of sparsity to find an optimal balance between model size reduction and performance (Molchanov et al., 2019). In this study, three specific methods were used for unstructured pruning: *magnitude-based pruning*, *gradient-based pruning*, and *L1 regularization-based pruning*.

4.2.1.1 Methods of Unstructured Pruning

Magnitude-Based Pruning: This method involves removing weights based on their absolute values, under the assumption that weights with smaller magnitudes have less influence on the network's predictions. This makes them suitable candidates for pruning while maintaining overall accuracy. Magnitude-based pruning has been widely used due to its simplicity and effectiveness in reducing the model size without significant performance degradation (Han et al., 2015; TensorFlow, 2023).

Gradient-Based Pruning: This approach utilizes gradient information of the weights to determine their importance. Weights with smaller gradients are considered less influential in reducing the loss during training and are thus pruned. This method offers a dynamic way of identifying unimportant parameters based on their sensitivity to the model's performance (Zhu & Gupta, 2017).

L1 Regularization-Based Pruning: L1 regularization is applied during training to encourage weights to shrink towards zero, promoting sparsity naturally. After training with L1 regularization, weights that have become very small are pruned based on a set threshold. This method integrates the pruning process into the training phase, making it effective in reducing unnecessary parameters while preserving critical features (Wen, Wu, Wang, Chen, & Li, 2016).

4.2.1.2 General Workflow for Unstructured Pruning

Each of the three pruning methods follows a common process:

1. **Weight Selection and Pruning:** Each method uses a specific criterion (magnitude, gradient values, or regularization effects) to select weights for pruning. The identified weights are set to zero, resulting in a sparse version of the model that maintains its overall structure.
2. **Evaluate Pruned Models:** After pruning, multiple versions of the pruned model are generated using different pruning percentages (e.g., 5%, 10%, 15%). These models are evaluated on a validation dataset to measure performance metrics such as mAP50-95, precision, and recall, helping to assess the impact of pruning on corrosion detection.
3. **Select and Fine-Tune the Optimal Model:** Based on the evaluation results, the pruned model that achieves the best trade-off between size reduction and accuracy is selected. This model is fine-tuned for a few epochs to adapt to its new sparse structure, ensuring optimal performance in deployment scenarios.

4.2.2 Structured Pruning

In this work, structured pruning was implemented to reduce the complexity of the Yolov8n model while maintaining its performance in corrosion detection tasks. Structured pruning removes entire filters from convolutional layers, simplifying the network's architecture in a manner that is more compatible with deployment on resource-limited hardware.

4.2.2.1 Filter Removal Using L1-Norm

The pruning process in this study utilized the L1-norm to determine which filters to remove. The L1-norm calculates the sum of the absolute values of each filter's weights, providing a measure of its contribution to the network's output. Filters with lower L1-norm values are considered less important and are thus removed. This method is widely used in pruning research due to its simplicity and effectiveness in retaining model performance while reducing model size (H. Li et al., 2017; He et al., 2017).

4.2.2.2 Stepwise Pruning Process and the Rationale for Gradual Pruning

The pruning was conducted in a stepwise manner, with a fixed percentage (e.g., 10%) of filters pruned at each step. After each step, the model was fine-tuned over a set number of epochs to adapt to its new architecture and recover any performance loss. This gradual approach was chosen over one-shot pruning due to several key reasons:

- **Stability in Performance:** Gradual pruning minimizes the risk of a sudden drop in model accuracy, allowing the network to adjust incrementally to a smaller structure (Z. Liu, Sun, Zhou, Huang, & Darrell, 2019).
- **Fine-Tuning for Recovery:** Each pruning step was followed by fine-tuning to recalibrate the model's weights and maintain performance, helping to mitigate the impact of removed filters (Han et al., 2015).
- **Controlled Pruning Process:** A gradual approach enables precise monitoring of the model's performance at each stage, allowing for better identification of the optimal pruning point where model size reduction does not significantly degrade accuracy (Molchanov, Tyree, Karras, Aila, & Kautz, 2023).

This chapter provided a detailed explanation of the pruning techniques used, including structured and unstructured pruning, and how they were applied to reduce the complexity of the Yolov8n model while maintaining performance. In the next chapter, we will discuss the results of these techniques in relation to model performance, accuracy, and efficiency.

Chapter 5

Results

This chapter presents the results from the initial training, hyperparameter tuning, and the application of pruning techniques on the Yolov8n model for corrosion detection. The performance metrics, such as precision, recall, mAP@50, and mAP@50-95, are discussed in detail, providing insights into the impact of structured and unstructured pruning on model accuracy, size, and inference speed. The results are visualized through tables and figures, highlighting key improvements and trade-offs in performance across different pruning methods.

5.1 Initial Training Results

The results of the initial training of the Yolov8n model, using the default settings outlined in subsection 3.2.1, provide a foundational performance benchmark for further tuning and optimization.

The training process achieved the following key metrics at the best epoch (86):

- **Precision:** 0.91177
- **Recall:** 0.78333
- **mAP@50:** 0.86032
- **mAP@50-95:** 0.71975

These results indicate the model's high precision, although recall and mAP@50-95 showed room for improvement. The model's ability to localize and detect corrosion with an mAP@50 of 0.86032 demonstrates solid initial performance.

These results are visualized in Figure 5.1, showing the progression of metrics over the training epochs and highlighting the performance at the best epoch.

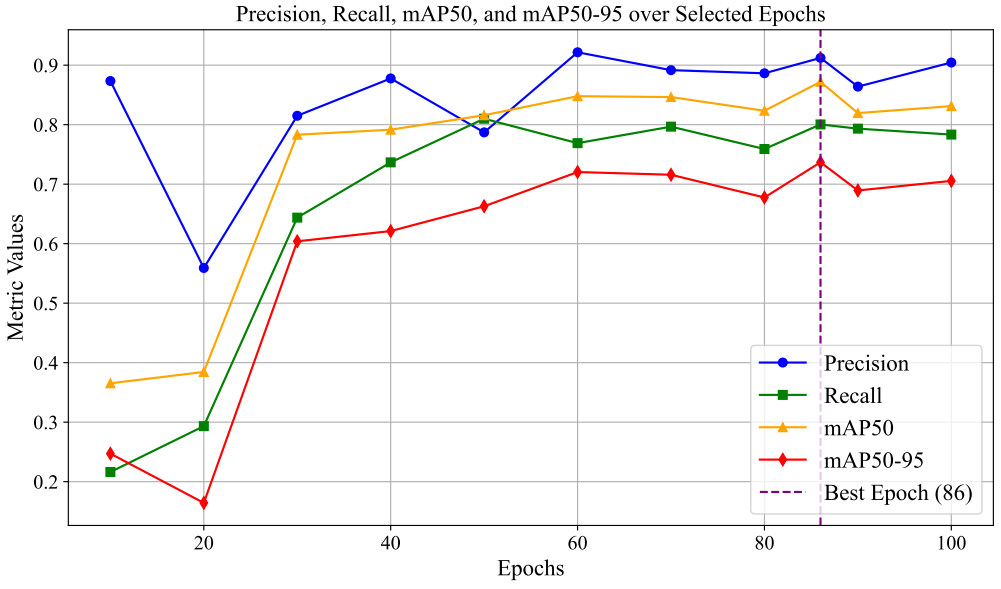


Figure 5.1: Performance metrics of the Yolov8n model during initial training, with the best performance observed at epoch 86.

5.2 Results of Hyperparameter Tuning

A total of 43 trials were conducted during hyperparameter tuning, each evaluating different configurations mentioned in subsection 3.2.2. The top-performing trials based on mAP50 are summarized in Table 5.1. Trial_5 yielded the highest mAP50, while Trial_41 was selected for further analysis due to its balance of precision and recall.

experiment	mAP50	precision	recall	mAP50-95	epoch
Trial_5	0.93672	0.93927	0.89452	0.82951	41
Trial_32	0.92742	0.93828	0.86375	0.82727	42
Trial_42	0.92729	0.94397	0.87662	0.79613	47
Trial_26	0.92678	0.96654	0.88889	0.82131	43
Trial_30	0.92302	0.97904	0.88508	0.82683	44
Trial_41	0.92285	0.96035	0.88516	0.81912	47
Trial_4	0.92134	0.93604	0.87779	0.80256	44
Trial_8	0.91993	0.92971	0.88638	0.81900	45
Trial_35	0.91829	0.92151	0.88780	0.79307	45
Trial_23	0.91394	0.95454	0.88889	0.80066	45

Table 5.1: Top 10 Trials by mAP50 Performance

The selected hyperparameters for this trial included

- a batch size of **16**
- an image size of **416**
- a learning rate (**lr0**) of **0.0006577248560083984**
- the **Adam** optimizer.

5.3 Results for Baseline Model: Post Hypertuning

The baseline model was trained for up to 500 epochs by utilising the best hyperparameters (subsection 3.2.2); however, training stopped early due to early stopping criteria, with the best performance observed at epoch 220. The final epoch reached was 320. The performance of the baseline Yolov8n model was evaluated before applying pruning techniques:

- **Precision:** 0.9406 (↑ 12.4%)
- **Recall:** 0.8400 (↑ 8%)
- **mAP50:** 0.8947 (↑ 4.6%)
- **mAP50-95:** 0.7593 (↑ 3.2%)
- **Model Size:** 5.96 MB

- **FLOPs:** 1.71 GFLOPs
- **Parameters:** 3.01 million
- **Average Inference Time:** 0.0516 seconds per image
- **Inference Speed:** 19.40 FPS

The values in parentheses represent the improvements in key metrics after hyperparameter tuning, which was conducted to optimize the learning process and enhance model performance.

Figure 5.2 shows the progression of mAP50, mAP50-95, precision, and recall over the training process, highlighting the best epoch.

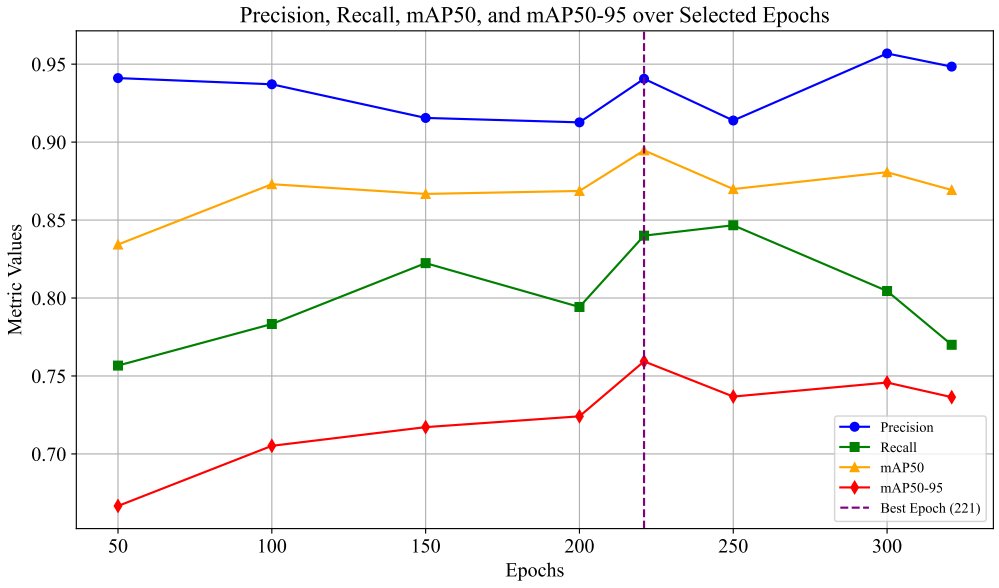


Figure 5.2: Performance metrics of the baseline Yolov8n model during training, with the best epoch marked.

5.3.1 Visual Predictions from the Baseline Model

Figure 5.3 shows some of the predicted results from the baseline Yolov8n model on a subset of corrosion images. These examples illustrate the model's ability to accurately detect and localize corrosion regions, as highlighted by the bounding boxes around areas identified as corrosion.

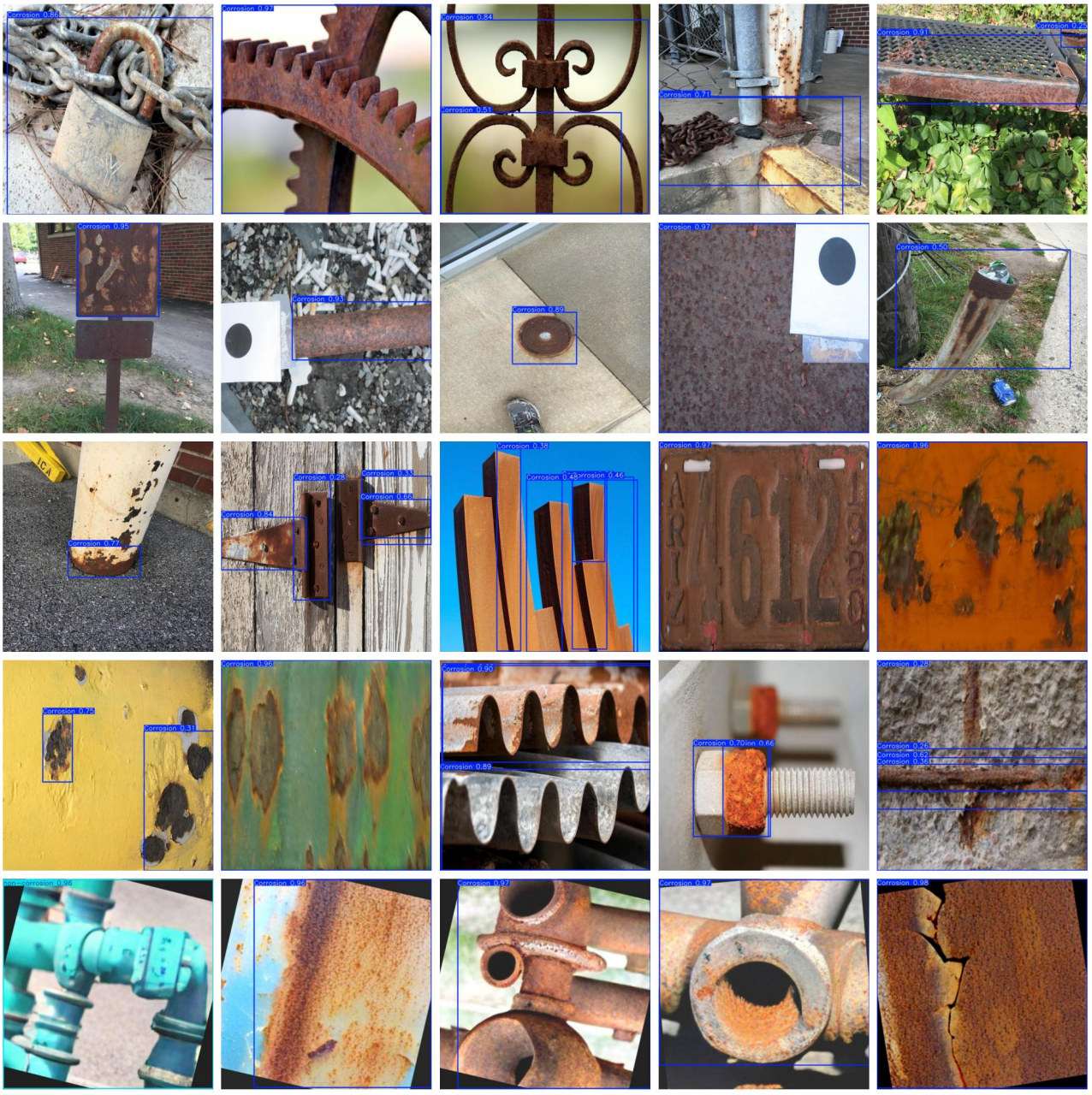


Figure 5.3: Predictions of corrosion instances by the baseline Yolov8n model. The model successfully detects and localizes areas of corrosion across various objects and surfaces.

5.4 Results of Unstructured Pruning: Magnitude-Based Pruning

5.4.1 Impact on Model Size and Sparsity

- **Model Size:** Remained stable across different pruning levels, despite increasing sparsity.
- **Sparsity:** Increased linearly with pruning percentage, reaching approximately 25% at 25% pruning.

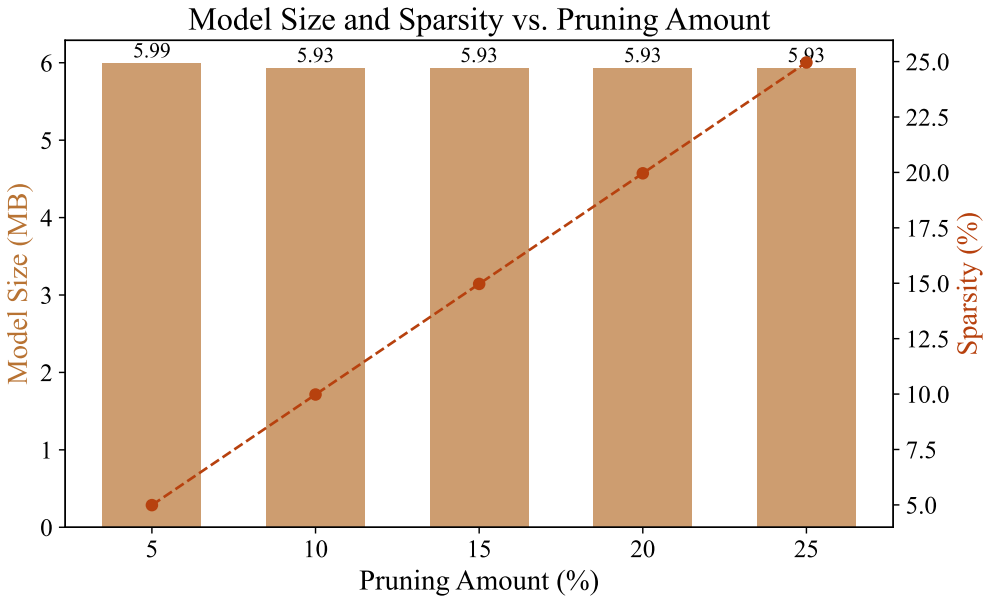


Figure 5.4: Model size and sparsity across different pruning amounts for magnitude-based pruning.

5.4.2 Inference Speed

- **Maximum Inference Speed:** 38.66 FPS at 10% pruning.
- **Inference Speed at 25% Pruning:** 29.15 FPS, showing diminishing speed improvements at higher pruning levels.

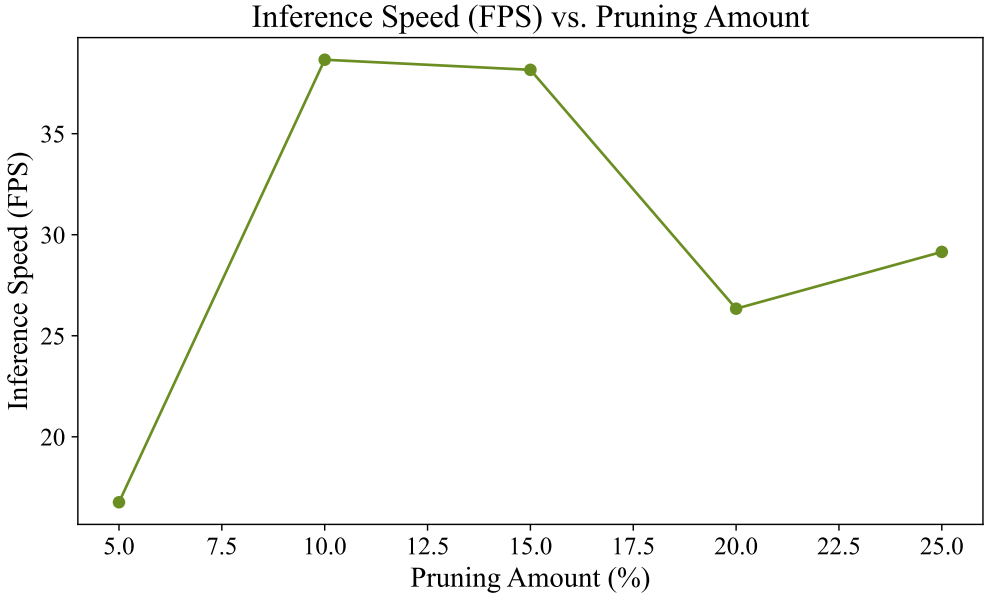


Figure 5.5: Inference speed (FPS) as a function of pruning amounts for magnitude-based pruning.

5.4.3 Precision, Recall, and mAP Metrics

- **Precision:** Stable up to 10% pruning, but dropped to 0.3010 at 25% pruning.
- **Recall:** Consistent across different pruning levels.
- **mAP@50:** Declined steadily from 0.7627 at 5% pruning to 0.4918 at 25% pruning.

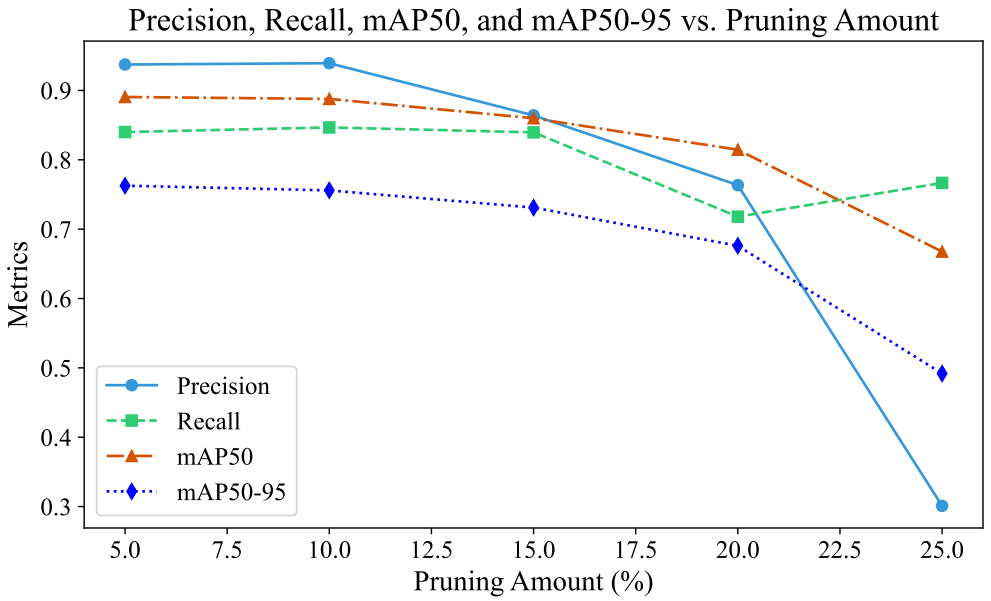


Figure 5.6: Precision, Recall, mAP50, and mAP50-95 at different pruning levels for magnitude-based pruning.

5.4.4 Effect of Fine-Tuning

- **Precision Improvement After Fine-Tuning:** Increased from 0.7635 to 0.9165 at 20% pruning.
- **mAP50 After Fine-Tuning:** Slightly increased.
- **mAP50-95 After Fine-Tuning:** Decreased to 0.6415.

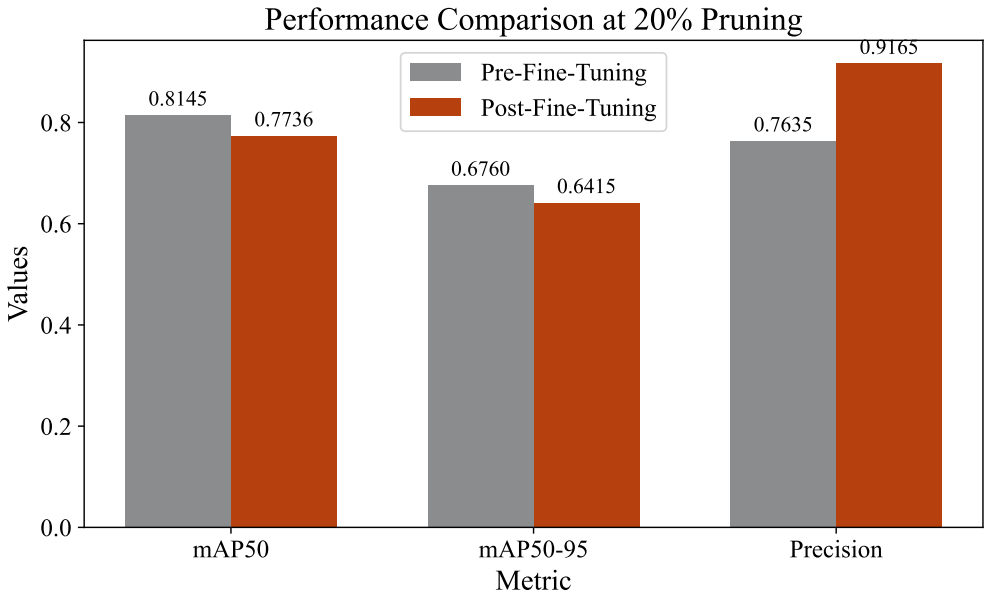


Figure 5.7: Comparison of performance metrics (mAP50, mAP50-95, Precision) before and after fine-tuning at 20% pruning.

5.5 Results of Unstructured Pruning: Gradient-Based Pruning

5.5.0.1 Impact on Model Size and Sparsity

- **Model Size:** Remained constant at 5.93 MB across different pruning amounts.
- **Sparsity:** No significant increase in sparsity at different pruning levels.

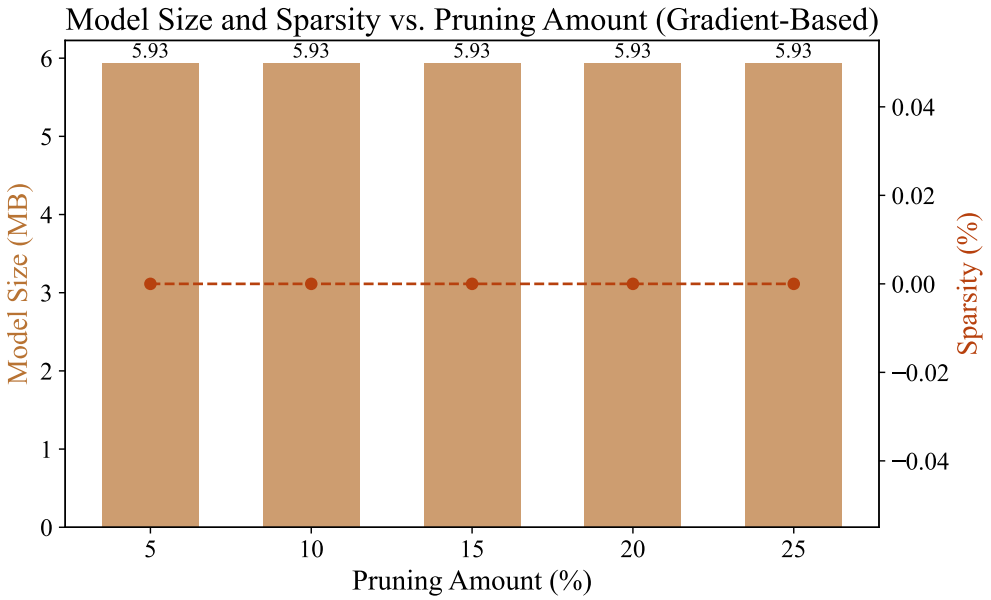


Figure 5.8: Model size and sparsity across different pruning amounts for gradient-based pruning. Model size remains stable while sparsity does not increase significantly.

5.5.0.2 Inference Speed

- **Maximum Inference Speed:** 30.25 FPS at 5% pruning.
- **Inference Speed at 25% Pruning:** 18.93 FPS, indicating a decrease in speed at higher pruning levels.

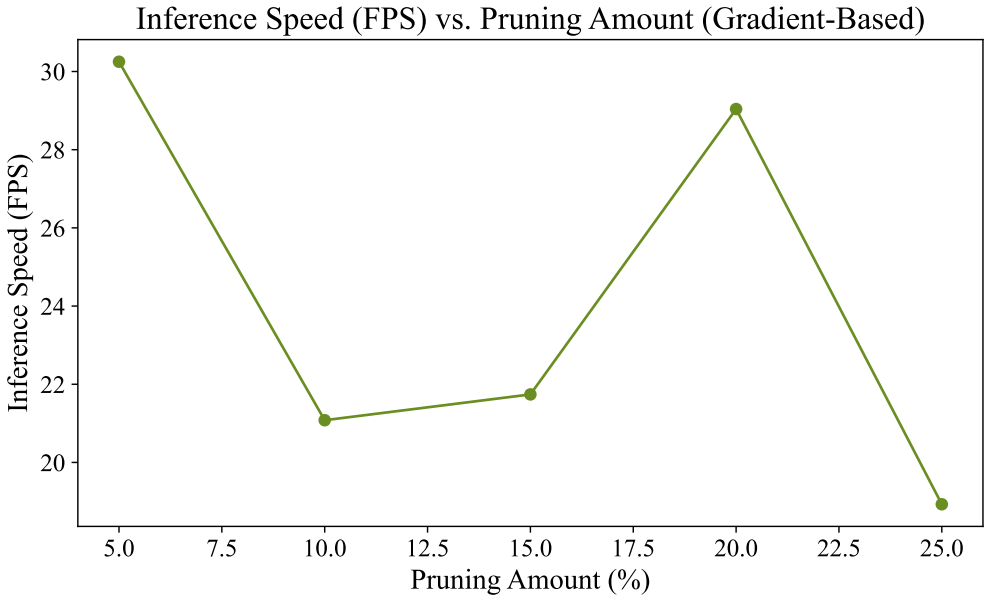


Figure 5.9: Inference speed (FPS) as a function of pruning amounts for gradient-based pruning. Speed is highest at 5% pruning but decreases at higher pruning levels.

5.5.0.3 Precision, Recall, and mAP Metrics

- **Precision:** Remained high, especially at 25% pruning.
- **Recall:** Stable across different pruning levels.
- **mAP50-95:** Remained above 0.75 until 20% pruning, highlighting robust performance.

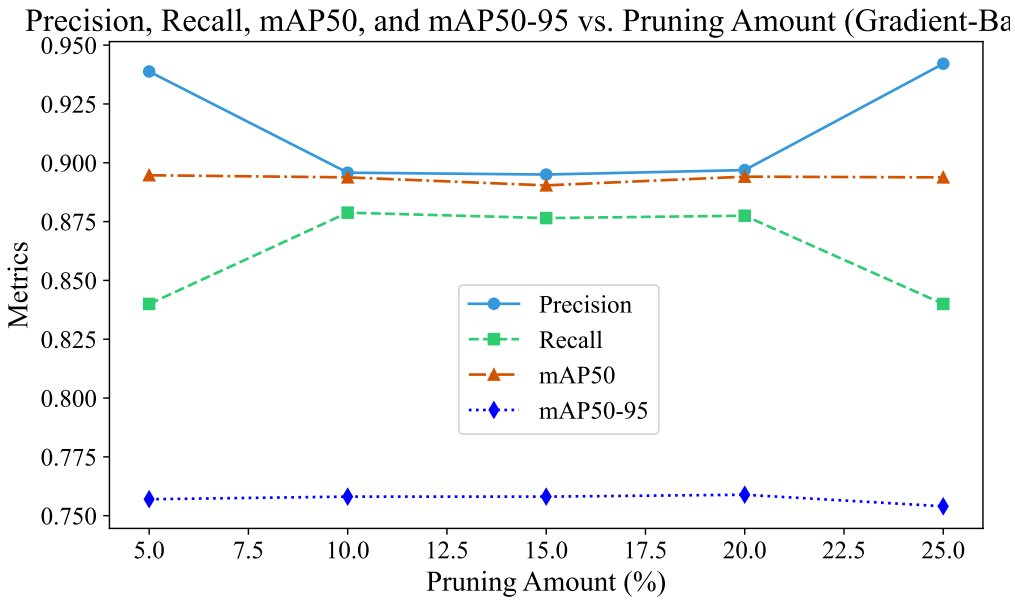


Figure 5.10: Precision, Recall, mAP50, and mAP50-95 across varying pruning levels for gradient-based pruning. Precision remains high, while mAP metrics maintain stability.

5.5.0.4 Effect of Fine-Tuning

- **Post-Fine-Tuning Precision:** Decreased to 0.8518 from 0.9421 at 25% pruning.
- **Post-Fine-Tuning mAP50:** Dropped to 0.8408.
- **Post-Fine-Tuning mAP50-95:** Decreased to 0.6391.

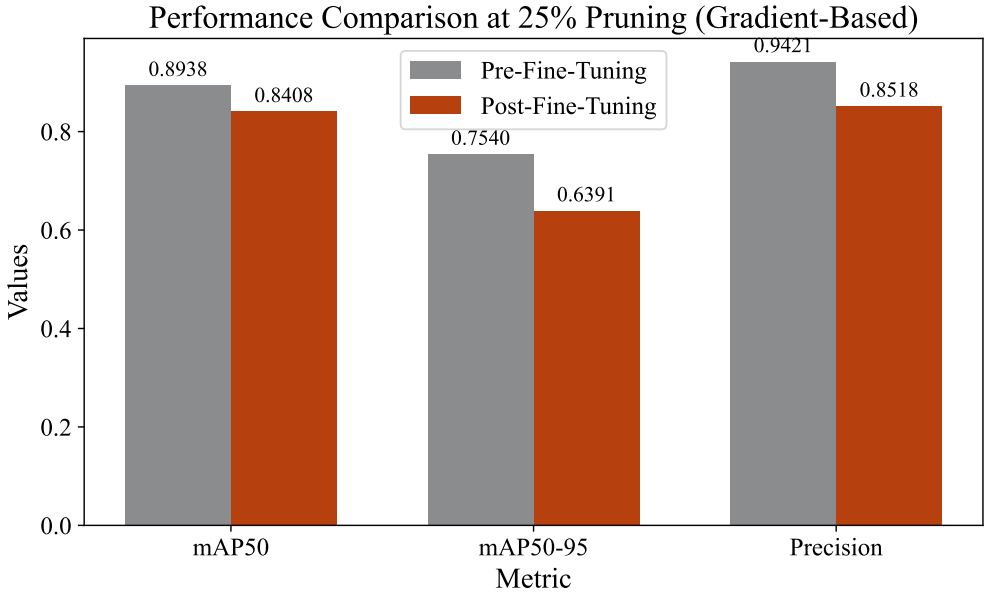


Figure 5.11: Comparison of pre- and post-fine-tuning performance metrics at 25% pruning for gradient-based pruning. Fine-tuning improves generalization but results in slight drops in detailed accuracy.

5.6 Results of Unstructured Pruning: L1-Based Pruning

5.6.0.1 Impact on Model Size and Sparsity

- **Model Size:** Decreased slightly from 5.99 MB to 5.93 MB as pruning amount increased.
- **Sparsity:** Increased from 4.99% at 5% pruning to 24.96% at 25% pruning, reflecting a gradual elimination of less critical weights.

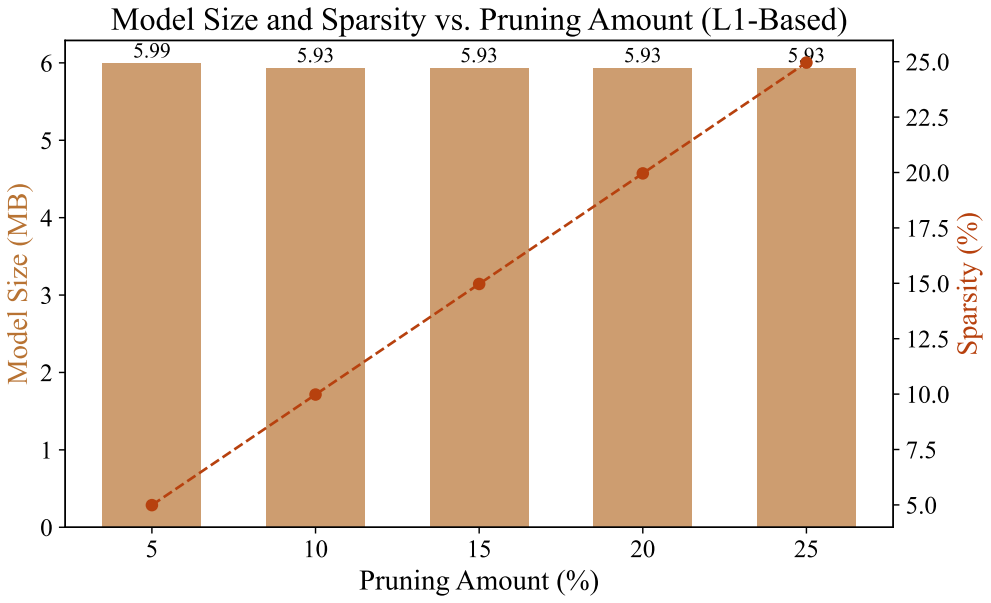


Figure 5.12: Model size and sparsity across different pruning amounts for L1-based pruning. Model size shows a minor reduction, while sparsity increases as more weights are pruned.

5.6.0.2 Inference Speed

- **Maximum Inference Speed:** 39.93 FPS at 15% pruning.
- **Inference Speed at 25% Pruning:** 30.42 FPS, indicating a trade-off between sparsity and computational overhead at higher pruning levels.

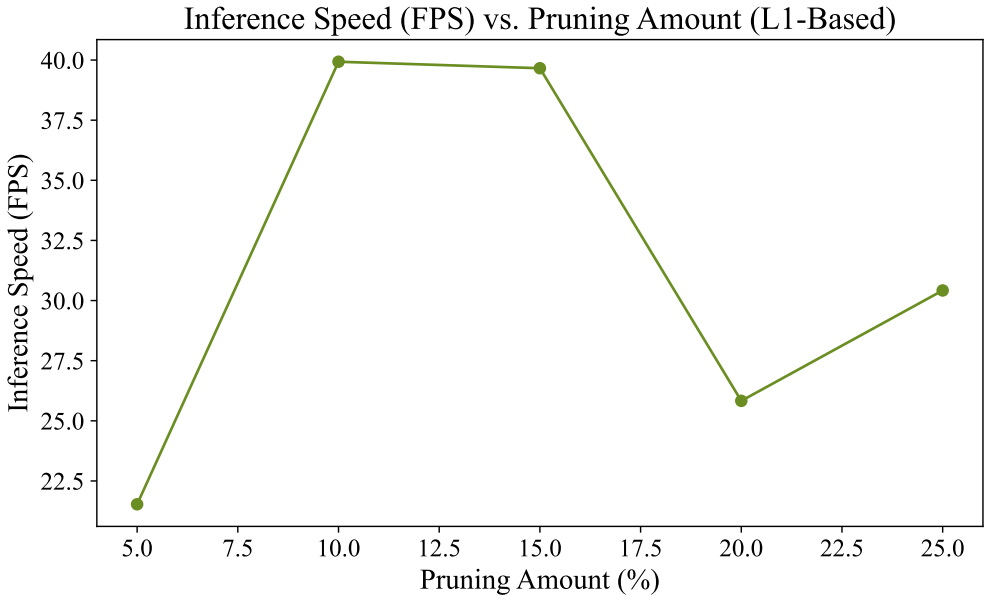


Figure 5.13: Inference speed (FPS) vs. pruning amounts for L1-based pruning. Speed increases with moderate pruning but drops at higher levels.

5.6.0.3 Precision, Recall, and mAP Metrics

- **Precision:** Decreased from 0.9087 at 5% to 0.4280 at 25%, indicating a decline in positive identification accuracy with aggressive pruning.
- **mAP50 and mAP50-95:** Remained relatively stable up to 10% pruning, indicating that detection capabilities were maintained at moderate pruning levels.

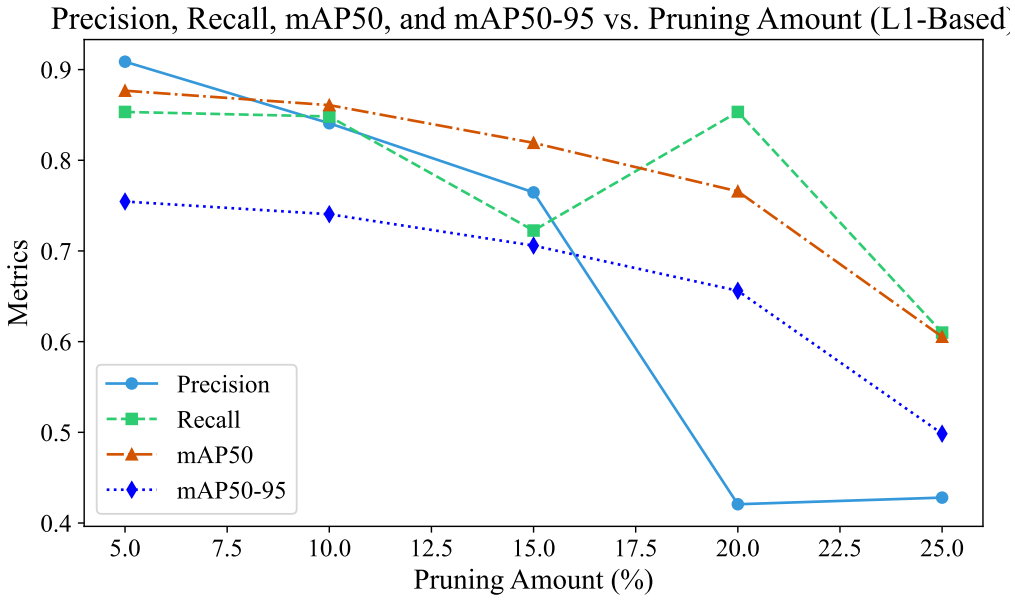


Figure 5.14: Precision, Recall, mAP50, and mAP50-95 across varying pruning levels for L1-based pruning. Precision shows a decline with increased pruning, while mAP metrics remain stable up to moderate pruning levels.

5.6.0.4 Effect of Fine-Tuning

- **Post-Fine-Tuning mAP50:** Improved slightly from 0.8192 to 0.8323 at 15% pruning.
- **Post-Fine-Tuning mAP50-95:** Minor decrease to 0.7020, indicating a trade-off between generalization and detailed accuracy.

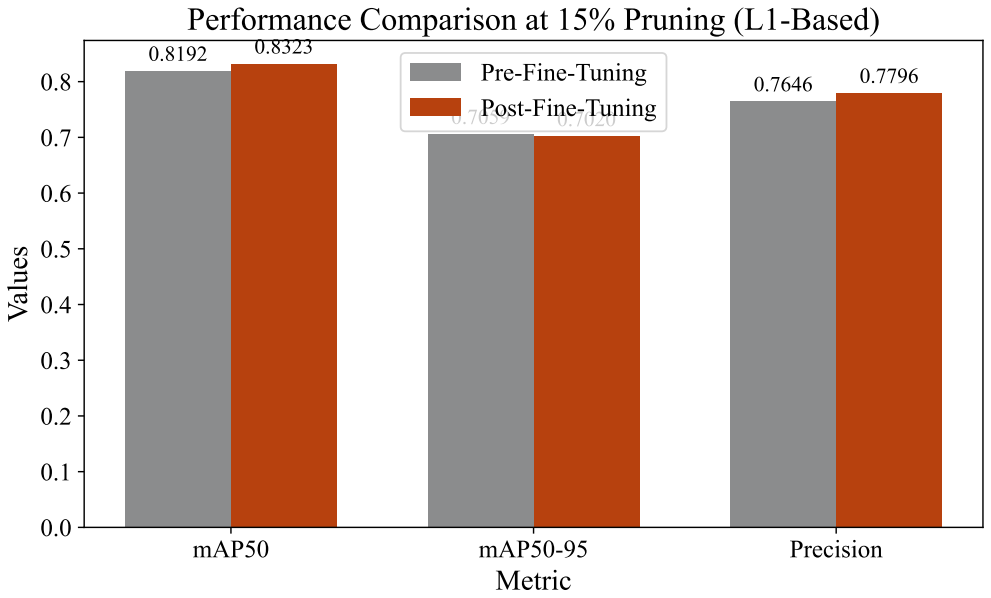


Figure 5.15: Comparison of pre- and post-fine-tuning performance metrics at 15% pruning for L1-based pruning. Fine-tuning offers slight improvements in mAP50 but results in minor decreases in mAP50-95.

5.7 Results of Structured Pruning

Structured pruning was applied to the Yolov8n model in a stepwise fashion, with fine-tuning performed after each step to adapt the model to its progressively pruned structure. This approach aimed to achieve a balance between reducing the model's complexity and maintaining its ability to detect corrosion accurately. The primary metrics observed were *Precision*, *Recall*, *mAP50*, and *mAP50-95*, as shown in Figure 5.16.

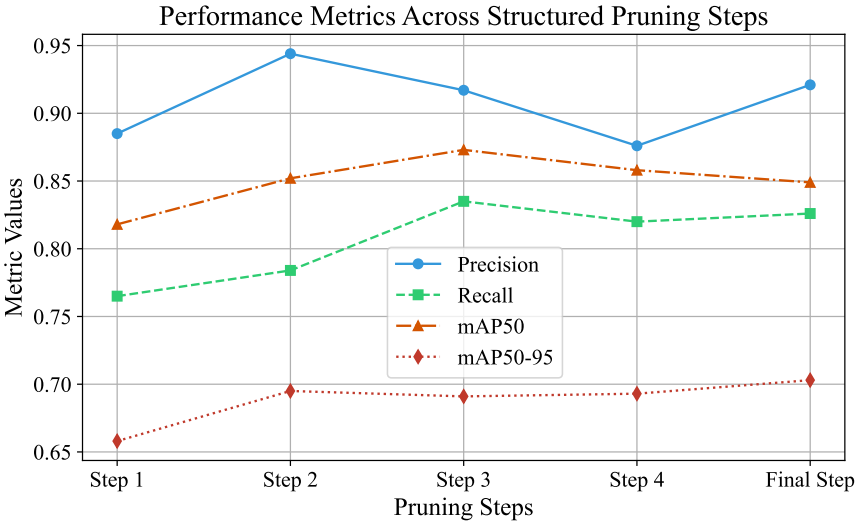


Figure 5.16: Performance metrics across structured pruning steps, including Precision, Recall, mAP50, and mAP50-95. Each step represents a pruning stage followed by fine-tuning.

Performance Across Pruning Steps As illustrated in Figure 5.16, the model maintained a high level of performance throughout the structured pruning process. Precision improved notably between Step 1 and Step 2, peaking at 0.944 before gradually stabilizing in the subsequent steps. This increase is attributed to the fine-tuning process, which helped the model adjust to the removal of less critical filters while refining its focus on significant features.

Stability of Recall and mAP Metrics Recall showed a more consistent trend across all steps, ranging from 0.765 in Step 1 to 0.8267 in the final step. The relatively stable recall values suggest that the model retained its ability to detect positive instances of corrosion even as its complexity was reduced. Meanwhile, mAP50 values increased up to Step 3, reaching 0.873, before showing a slight decline in the final step. The mAP50-95 metric exhibited a similar trend, increasing from 0.658 in Step 1 to 0.7032 in the final step, indicating that the model's ability to detect objects across varying Intersection over Union (IoU) thresholds remained strong throughout the pruning process.

Effectiveness of Fine-Tuning The fine-tuning performed after each pruning step played a crucial role in stabilizing the model's performance metrics. Unlike unstructured pruning, where fine-tuning was applied only after selecting an optimal pruning level, structured pruning required fine-tuning at every step to compensate for the removal of entire filters. This iterative fine-tuning approach ensured that the model could adapt to its new architecture, preserving its ability to detect both corrosion and non-corrosion instances effectively.

Overall, structured pruning, when combined with stepwise fine-tuning, demonstrated its potential for reducing model complexity while maintaining robust detection performance, making it a viable strategy for real-time corrosion detection applications.

This chapter presented the results of initial training, hyperparameter tuning, and pruning experiments for the Yolov8n model. Performance improvements and the impact of pruning on model size, accuracy, and inference speed were analyzed. In the next chapter, we will discuss these results in detail, focusing on the trade-offs and implications for real-time corrosion detection.

Chapter 6

Discussion

This chapter provides a detailed analysis of the results, focusing on how different pruning techniques affected the Yolov8n model's performance in terms of accuracy, inference speed, and model size. It explores the trade-offs between structured and unstructured pruning methods, evaluates their suitability for real-time corrosion detection in resource-constrained environments, and addresses the research questions posed earlier in the study. By comparing these techniques, the discussion highlights the best approach for balancing model optimization and performance, offering insights into practical deployment scenarios.

The following discussion delves into the implications of the results presented in earlier chapter ocusing on how various pruning techniques affected the Yolov8n model's performance in terms of accuracy, inference speed, and model size. The discussion highlights the trade-offs between these factors, offering insights into which pruning techniques are most appropriate for real-time corrosion detection in resource-constrained environments.

6.1 Initial Tranining Performance

The initial training results (see section 5.1) demonstrated that the Yolov8n model was able to achieve a strong baseline performance, with a precision of 0.91177 and an mAP@50 of 0.86032. While the precision was high, indicating the model's effectiveness in reducing false positives, the recall (0.78333) and mAP@50-95 (0.71975) indicated room for improvement in terms of detecting a broader range of true positives and maintaining accuracy under stricter IoU thresholds. The mAP@50-95 metric, in particular, highlights the challenge of accurately localizing corrosion when the bounding box overlaps must be more precise (between 50% and 95%). This is a critical aspect of corrosion detection, as the variability in shape, size, and texture of corrosion makes localization a challenging task. These results served as a baseline for the subsequent hyperparameter tuning and pruning efforts.

6.2 Hyperparameter Tuning

The best-performing configuration from this process was **Trial_5**, which achieved the highest mAP50 of **0.93672** and a competitive mAP50-95 of **0.82951**. This trial outperformed others in terms of detection accuracy, making it ideal for applications where precise localization of corrosion is critical. However, **Trial_41** was selected for further training due to its balanced performance across multiple metrics. It achieved a **precision of 0.96035** and a **recall of 0.88516**, with an **mAP50 of 0.92285** and an **mAP50-95 of 0.81912**. Despite not being the highest-scoring trial in terms of mAP50 or mAP50-95, Trial_41 provided a strong combination of precision and recall. Additionally, its relatively lower box losses during training and validation (0.31772 and 0.59464, respectively) suggested effective learning and generalization, making it a suitable choice for further model refinement.

6.3 Baseline Performance

The baseline performance of the Yolov8n model serves as a crucial reference for evaluating the impact of subsequent pruning techniques. With a high precision of **0.9406**, the model effectively minimizes false positives, which is vital for reducing unnecessary interventions in industrial applications. The recall of **0.8400** marks an 8% improvement from the initial training, reflecting the model's enhanced sensitivity to detecting true positives, though it still misses around 16% of corrosion instances. Addressing this gap could involve further fine-tuning or data augmentation to improve detection in challenging scenarios.
The model's **mAP@50** score of **0.8947** demonstrates strong detection and localization capabilities at standard IoU thresholds, but the **mAP@50-95** of **0.7593** highlights the challenge of maintaining accuracy under stricter IoU conditions. While the baseline model shows robust

performance, further optimization is necessary to enhance localization in real-world applications where precise detection is critical. The **average inference time** of **0.0516 seconds per image** (19.40 FPS) and the model size of **5.96 MB** establish important benchmarks for assessing how pruning can improve efficiency without sacrificing accuracy, making the model more suitable for real-time, resource-constrained environments.

6.4 Comparative Analysis: Structured vs. Unstructured Pruning

This section compares structured and unstructured pruning methods applied to the Yolov8n model, focusing on key metrics such as mAP50, mAP50-95, model size, and inference time. The baseline model serves as a reference point for evaluating these methods.

6.4.1 Overview of Results

Figure 6.1 summarizes the performance metrics for the baseline, structured pruning, and unstructured pruning methods. It highlights each model's mAP50, mAP50-95, model size, and average inference time, facilitating a direct comparison of each pruning approach's effectiveness.

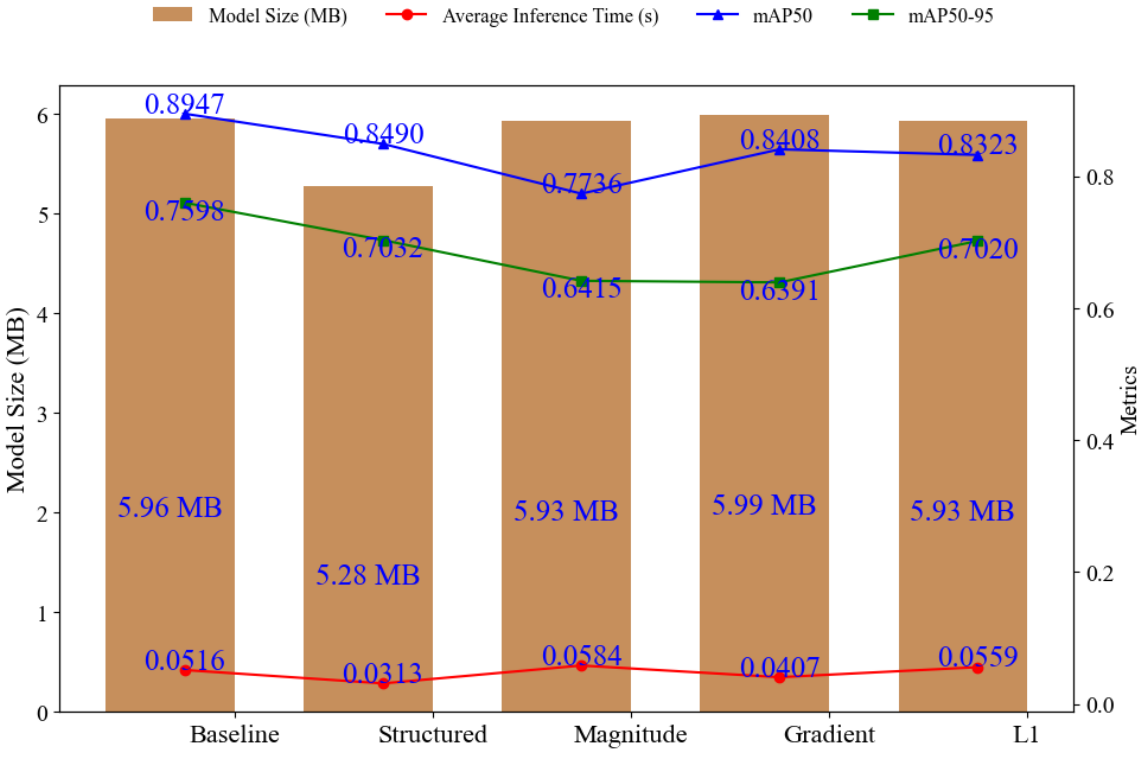


Figure 6.1: Performance comparison of mAP50, mAP50-95, model size, and average inference time across baseline, structured, and unstructured pruning methods.

6.4.2 Key Findings

Model Size and Inference Time Structured pruning reduces model size by achieving 5.28 MB compared to the baseline's 5.96 MB. It also achieves the fastest inference time at 0.0313

seconds making it the most efficient method for real-time use. Unstructured methods maintain model sizes closer to the baseline but vary in speed; gradient-based pruning is faster than L1 and magnitude-based pruning.

Detection Accuracy (mAP50 and mAP50-95) The baseline model offers the highest mAP50-95 at 0.7598. Structured pruning achieves a slightly reduced mAP50-95 of 0.7032, but maintains a good balance between accuracy and model efficiency. Gradient-based pruning follows with an mAP50-95 of 0.6391, showing resilience among unstructured methods. Magnitude-based and L1-based pruning, while effective in reducing parameters, see greater declines in accuracy.

Precision and Recall Precision remains highest in the baseline (0.9406), with structured pruning close behind at 0.9210. Gradient-based pruning shows relatively stable precision (0.8958) compared to other unstructured methods. Recall values are similar, indicating that structured pruning maintains detection sensitivity better than unstructured methods while simplifying the model.

6.4.3 Comparative Analysis of Structured Pruning with Baseline

Figure 6.3 illustrates the relative changes in key performance metrics between the baseline model and the structured pruning method. Structured pruning results in a 5.09% decrease in mAP50 and a 7.43% decrease in mAP50-95, indicating a controlled reduction in detection accuracy compared to the baseline. Despite these decreases, it achieves an 11.39% reduction in model size, making it more compact and efficient for deployment. Additionally, structured pruning improves the average inference time by 39.34%, enhancing the model's processing speed. However, this comes with an 8.50% decrease in precision, suggesting a minor trade-off in accuracy. On the other hand, a 3.25% improvement in recall suggests better retention of the model's ability to detect true positives.

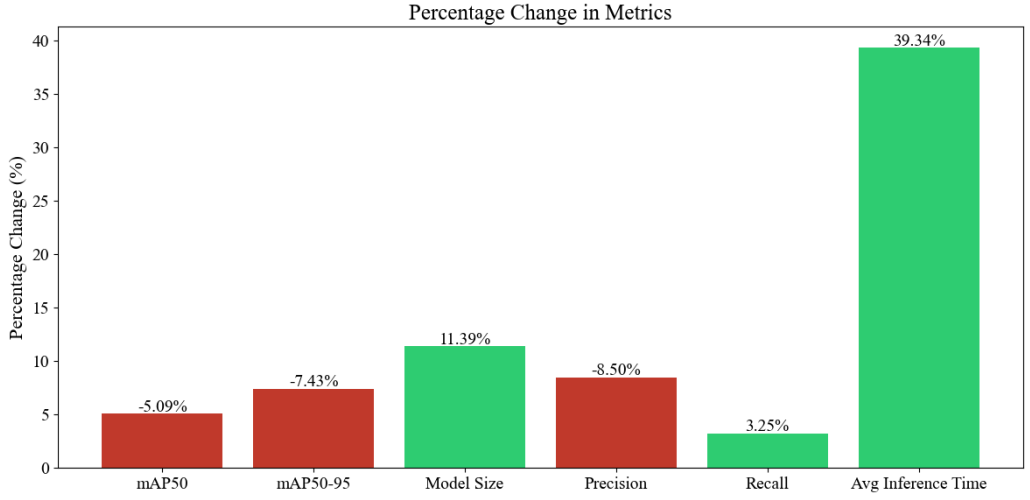


Figure 6.2: Percentage change in mAP50, mAP50-95, model size, precision, recall, and average inference time compared to the baseline for structured pruning.

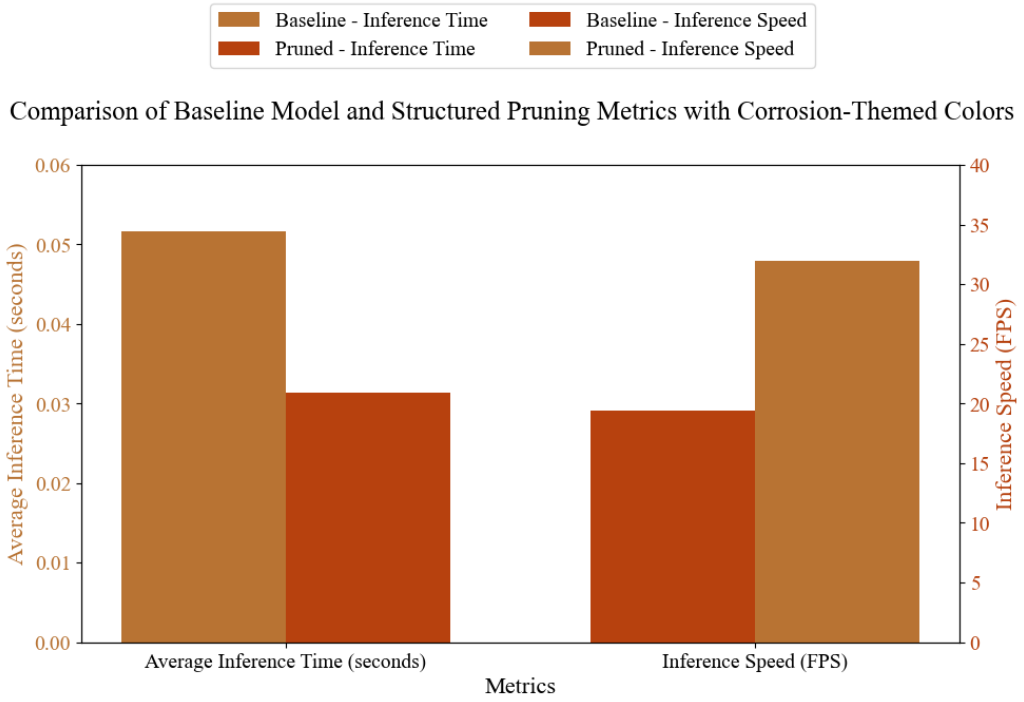


Figure 6.3: Comparison of average inference time and inference speed between the baseline and structured pruning.

Structured pruning strikes a balance between model efficiency and accuracy, offering significant reductions in model size and inference time while retaining competitive detection performance. Compared to unstructured methods, which may excel in specific metrics like precision, structured pruning delivers a well-rounded improvement, making it particularly advantageous for real-time applications in resource-constrained environments.

6.5 Discussion of Research Questions

This section provides direct answers to the research questions posed at the beginning of the thesis, based on the results and analysis presented in earlier chapters.

6.5.1 Impact of Structured and Unstructured Pruning on Yolov8n Performance

How do structured and unstructured pruning techniques impact the performance of the Yolov8n model in corrosion detection tasks?

The results demonstrate that both structured and unstructured pruning techniques affect the performance of the Yolov8n model, but in distinct ways:

- **Structured pruning** consistently maintained high accuracy, with an mAP@50 of **0.8412** and mAP@50-95 of **0.7032**, while incurring only minor losses in precision and recall. This makes it particularly well-suited for corrosion detection tasks where maintaining detection performance is critical.

- **Unstructured pruning**, including techniques like magnitude-based and gradient-based pruning, led to more significant declines in detection performance, particularly at higher pruning levels. For example, magnitude-based pruning caused a substantial drop in mAP@50-95, down to **0.4918** at 25% pruning.

In conclusion, structured pruning had a more favorable impact on overall performance, striking a balance between maintaining high detection accuracy and reducing model size.

6.5.2 Optimization of Model Size, Speed, and Accuracy after Pruning

What is the balance between model size reduction, inference speed, and detection accuracy after pruning, and how can this be optimized for real-time corrosion detection?

Structured pruning provided the best balance across model size, inference speed, and detection accuracy:

- **Model size** was reduced by **11.39%**, from **5.96 MB** to **5.28 MB**.
- **Inference speed** improved by **39.34%**, reaching **31.94 FPS**, making the model highly suitable for real-time corrosion detection.
- **Detection accuracy** remained competitive, with an mAP@50 of **0.8412** and mAP@50-95 of **0.7032**.

By contrast, unstructured pruning led to more substantial declines in accuracy, especially at higher pruning levels, making it less optimal for balancing all three factors.

Thus, structured pruning offers the optimal solution for real-time industrial applications where speed, size, and accuracy must be considered together.

6.5.3 Yolov8n Deployment in Resource-Constrained Environments

Can the Yolov8n model be optimized for deployment in resource-constrained environments (e.g., drones, edge devices) without significant loss of detection precision or recall?

Yes, the Yolov8n model can be optimized for resource-constrained environments using structured pruning:

- The reduction in model size to **5.28 MB** and the improvement in inference speed to **31.94 FPS** make the model feasible for deployment on resource-constrained devices like drones or edge devices.
- Although there was a slight reduction in precision (from **0.9406** to **0.9210**) and recall (from **0.8400** to **0.8267**), these metrics remained sufficiently high to ensure effective corrosion detection in real-time applications.

Unstructured pruning, however, led to more pronounced losses in both precision and recall, which could limit its applicability in resource-constrained environments.

6.5.4 Trade-offs Between Different Pruning Techniques

What are the trade-offs involved in applying different pruning techniques to deep learning models used in industrial applications, such as corrosion detection?

The trade-offs between structured and unstructured pruning techniques are as follows:

- **Structured pruning** provides better control over performance metrics, maintaining a balance between model size, inference speed, and detection accuracy. This makes it ideal for real-time applications where both accuracy and efficiency are crucial.
- **Unstructured pruning**, while effective at reducing model size, often resulted in greater declines in accuracy, especially as the pruning level increased. For example, magnitude-based pruning saw a significant drop in mAP@50-95, making it less suitable for industrial applications that require high detection precision and recall.

The key trade-off is between aggressively reducing model size and preserving detection accuracy. Structured pruning offers the most favorable compromise, ensuring the model remains compact without significantly impacting performance.

In conclusion, structured pruning stands out as the most balanced and effective method for optimizing the Yolov8n model for corrosion detection in resource-constrained environments, making it a valuable approach for industrial applications.

In this chapter, we discussed the implications of the results, comparing structured and unstructured pruning techniques and their effect on the Yolov8n model's performance. The next chapter will provide the conclusions drawn from this study and suggest directions for future work.

In this chapter, we discussed the implications of the results, comparing structured and unstructured pruning techniques and their effect on the Yolov8n model's performance by answering the research questions. The next chapter will provide the conclusions drawn from this study and suggest directions for future work.

Chapter 7

Conclusions, Limitations Future Work

This chapter summarizes the key findings of this study, focusing on the effectiveness of pruning techniques in optimizing the Yolov8n model for real-time corrosion detection. The conclusions highlight the balance achieved between model size, inference speed, and detection accuracy, particularly through structured pruning. Additionally, the chapter discusses the study's limitations and outlines potential areas for future research, including exploring deeper models, refining pruning techniques, and improving the generalizability and robustness of pruned models in real-world industrial applications.

7.1 Conclusions

This study focused on optimizing the Yolov8n model for real-time corrosion detection by applying both structured and unstructured pruning techniques. The primary objective was to reduce model size and improve inference speed without compromising detection accuracy, making the model suitable for deployment in resource-constrained environments like drones or edge devices.

Several key conclusions can be drawn from the results:

- **Structured pruning** proved to be the most effective method for balancing model efficiency and accuracy. By selectively removing entire filters based on their L1-norm, the model's size was reduced by 11.39%, and inference time was improved by 39.34%, with only a minor drop in detection accuracy (a 7.43% decrease in mAP@50-95). This makes structured pruning highly suitable for real-time applications requiring both compactness and high precision.
- **Unstructured pruning**, particularly gradient-based pruning, offered some improvements in model size and speed but resulted in greater accuracy loss when pruning levels were too aggressive. While gradient-based pruning maintained a relatively stable precision, other unstructured methods, such as magnitude-based and L1-based pruning, showed steeper declines in both mAP and precision as pruning increased.
- **Fine-tuning** after pruning was essential in restoring performance, especially for unstructured methods. The models demonstrated a significant recovery in precision after fine-tuning, indicating that this step is crucial for maintaining accuracy after applying pruning techniques.
- Overall, the pruned Yolov8n models showed improvements in inference speed and model size, making them more practical for real-time corrosion detection tasks in resource-limited settings. Structured pruning, in particular, strikes the best balance between model compactness and detection performance, making it a favorable approach for deploying Yolov8n in industrial applications.

In conclusion, this research highlights the effectiveness of pruning techniques for optimizing Yolov8n, particularly in reducing the computational burden of real-time object detection while maintaining strong detection capabilities. The study contributes to the ongoing efforts to deploy deep learning models in resource-constrained environments, particularly for industrial inspections and corrosion monitoring.

7.2 Limitations

Several limitations were encountered during the study:

- **Training Environment and Technical Issues:** Running the model on a CPU greatly increased training time. Additionally, limited GPU availability in the free version of Google Colab caused frequent interruptions. Attempts to experiment with YOLOv10 encountered errors, including issues with gradient-based optimization and hyperparameter tuning, which led to failed trials and a focus on YOLOv8n.

- **Hyperparameters, Pruning Configurations, and Methods:** The study was constrained by time and resources, restricting the range of hyperparameters and pruning configurations explored. We primarily used PyTorch's native pruning library, though alternative methods like TensorFlow may offer different insights or better flexibility.
- **Model Limitations:** YOLOv8n models are highly optimized for speed and accuracy, limiting the potential for significant improvements through pruning. The model's inherent efficiency posed challenges in achieving further performance gains, especially for real-time environments.
- **Evaluation Metrics:** While metrics like mAP50 and mAP50-95 were effective in benchmarking, the results were obtained using high-powered GPUs. This may not reflect the models' actual performance on resource-limited devices like drones or edge platforms.
- **Generalization Ability:** The models were evaluated for robustness under medium-level Gaussian noise, but only a limited set of noise types was tested. Larger and more diverse datasets are needed to fully assess their generalization under various real-world conditions, including complex environmental challenges.

7.3 Future Work

The following areas offer potential directions for future research:

- **Depth of Pruned Models:** Future studies should investigate the impact of pruning on deeper models beyond YOLOv8n. Exploring how pruning affects the performance of deeper architectures could lead to more robust solutions for complex tasks like corrosion detection.
- **Specialized Pruning Techniques:** Research could focus on developing specialized pruning techniques tailored to specific architectures or industrial applications. Domain-specific pruning strategies, particularly for corrosion detection, could optimize models without compromising performance.
- **Metrics for Generalizability:** There is a need for standardized metrics to evaluate the generalizability of pruned models in real-world scenarios. Creating such metrics would improve comparisons across studies and facilitate reliable model deployment in industrial environments.
- **Robustness to Real-World Noise:** Future work should incorporate diverse noise sources and real-world environmental challenges, such as lighting variations and occlusions. This will ensure pruned models are robust enough for deployment in uncontrolled, real-world conditions.
- **Dataset Refinement:** Expanding corrosion detection datasets to include more varied corrosion types, environmental conditions, and asset surfaces will improve the robustness and accuracy of future models, ensuring practical utility in industrial settings.

Appendix A

Initial Training Results for YOLOv10n

The initial training of the YOLOv10n model showed a steady improvement in key metrics across the training epochs. At the start of training, the model exhibited low precision but high recall, suggesting that it detected most corrosion instances, though with many false positives. As the training progressed, both precision and mAP scores steadily improved, with notable jumps in performance by epoch 50 and further refinement by epoch 150.

The best results were observed at epoch 178, where the model reached a precision of 0.93733, recall of 0.75872, mAP@50 of 0.85822, and mAP@50-95 of 0.72329. After epoch 178, the performance remained stable, with minor fluctuations in precision and recall.

The following figure visualizes the progression of the model's key metrics over the training epochs, highlighting the best performance at epoch 178:

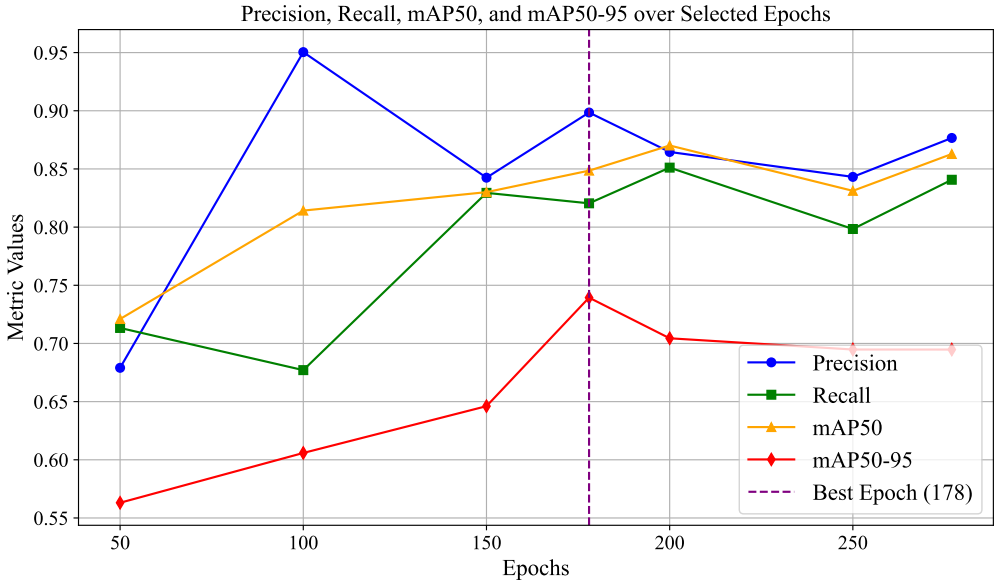


Figure A.1: Performance metrics of the YOLOv10n model during initial training, with the best performance observed at epoch 178.

By the final epoch (278), the model retained a strong performance with precision and recall values close to those at epoch 178, though mAP scores showed a slight decrease, indicating diminishing returns from continued training. Overall, the training progression demonstrated consistent improvement in detection accuracy, particularly in early epochs, with stabilization in performance beyond epoch 150.

The final model summary indicates 285 layers, 2.7 million parameters, and 8.2 GFLOPs, making it efficient in terms of computational requirements.

Challenges with YOLOv10n

While YOLOv10n showed competitive performance, several issues were encountered during training and evaluation. These challenges include:

- **Errors during hyperparameter tuning:** During hyperparameter tuning, we encountered multiple errors related to Ray Tune and PyTorch integration, such as 'RayTaskError' and attribute errors during gradient-based optimization, which resulted in failed trials.
- **Manual installation:** Unlike Yolov8n, YOLOv10n is not fully integrated with Python yet, requiring manual installation from GitHub using commands like:

  ```
  !pip install -q git+https://github.com/THU-MIG/yolov10.git
  !wget -P "{weights_path}" -q https://github.com/jameslahm/yolov10/
  releases/download/v1.0/yolov10n.pt
  ```

  This added complexity to the setup process.
- **Inferior evaluation metrics:** Yolov8n outperformed YOLOv10n in key evaluation metrics such as precision, recall, and mAP scores, making Yolov8n a more reliable option for corrosion detection.

References

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Brown, A., & Lee, M. (2021). Yolo-based models for industrial inspection and defect detection. *Computer Vision Applications*, *38*, 225-240.

Chen, L., Chen, Y., Xi, J., & Le, X. (2021, January). Ledge from the original network: restore a better pruned network with knowledge distillation. *Complex & Intelligent Systems*, *8*(32). doi: 10.1007/s40747-020-00248-y

Chen, L., & Zhang, W. (2022). Pruning techniques in deep learning: A comprehensive review. *Journal of Machine Learning*, *50*, 130-150.

Colab, G. (n.d.). *Google colab: Cloud-based jupyter notebooks with tesla t4 gpus.* https://colab.research.google.com. (Accessed: [Date of access])

Cuda toolkit documentation. (n.d.). https://docs.nvidia.com/cuda/. (Accessed: [Date of access])

Datature. (2023). *A comprehensive guide to neural network model pruning.* Retrieved from https://www.datature.io

Deepgram. (2023). *Model pruning, distillation, and quantization.* Retrieved from https://deepgram.com

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

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. *Advances in neural information processing systems*, *28*, 1135–1143.

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the ieee international conference on computer vision* (pp. 1389–1397).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704-2713.

Jocher, G. (2020). Yolov5 by ultralytics. *GitHub repository*.

Johnson, M., & Allen, R. (2021). Defect detection in manufacturing using yolov4: A real-time approach. *Manufacturing Technology Review*, *45*, 89-102.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep

convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097-1105.

Kumar, R., & Mehta, S. (2020). Yolo-based models for automated industrial inspection: A survey. *Journal of Industrial Automation*, *37*, 12-27.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436-444.

Lee, A., & Zhao, L. (2020a). Challenges and gaps in automated corrosion detection using deep learning. *Computer Vision Applications in Industry*, *39*, 145-158.

Lee, A., & Zhao, L. (2020b). Challenges in automated corrosion detection using deep learning models. *Computer Vision Applications in Industry*, *39*, 145-158.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. In *International conference on learning representations.*

Li, Y., Sun, J., & Wang, P. (2021). Balancing accuracy and efficiency in pruning for object detection models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2135-2144.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Proceedings of the European Conference on Computer Vision (ECCV)*, 21-37.

Liu, Z., Sun, M., & Yan, L. (2020). Over-pruning and its impact on model recovery: A case study. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 3451-3460.

Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019). Rethinking the value of network pruning. In *International conference on learning representations.*

Matplotlib. (n.d.). *Matplotlib: A comprehensive library for creating visualizations in python.* https://matplotlib.org/. (Accessed: [Date of access])

Mei, X., & et al. (2022). Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

Miller, S., & Davis, L. (2021). Economic costs of corrosion in the oil & gas sector. *Energy Infrastructure Review*, *29*, 120-135.

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2019). Pruning convolutional neural networks for resource-efficient inference. In *International conference on learning representations.*

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2023). Pruning convolutional neural networks for resource-efficient inference. *International Conference on Learning Representations*.

Nguyen, D., & Tran, M. (2021). Pruning yolo models for edge devices: A comparative study of yolov3 and yolov4. *Journal of Real-Time Computer Vision*, *47*, 134-145.

Numpy. (n.d.). *Numpy: The fundamental package for scientific computing in python.* https://numpy.org/. (Accessed: [Date of access])

Nvidia tesla t4 gpu for deep learning and machine learning tasks. (n.d.). https://www.nvidia.com/en-us/data-center/tesla-t4/. (Accessed: [Date of access])

Open Data Science. (2023). *What is pruning in machine learning?* Retrieved from https://opendatascience.com

OpenCV. (n.d.). *Opencv: Open source computer vision library.* https://opencv.org/. (Accessed: [Date of access])

Pandas. (n.d.). *Pandas: Python data analysis library.* https://pandas.pydata.org/. (Accessed: [Date of access])

Patel, N., & Kumar, S. (2019). Corrosion detection on metal surfaces using yolov3. *Journal of Materials Science*, *55*, 128-140.

Patel, R., & Garcia, M. (2019). A review of model optimization techniques for real-time applications. *Journal of Artificial Intelligence*, *33*, 56-70.

Python programming language. (n.d.). https://www.python.org/. (Accessed: [Date of access])

PyTorch. (2024). *Pytorch: An open source deep learning platform.* https://pytorch.org/. (Accessed: [Date of access])

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)*, 779-788.

Redmon, J., & Farhadi, A. (2017). Yolo9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517-6525.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Roberts, A., & Lee, K. (2020). A comparative study of corrosion detection methods in industrial applications. *Journal of Non-Destructive Testing*, *55*, 10-28.

Roboflow: Organize, label, and prepare your image data for training. (n.d.). https://roboflow.com/. (Accessed: [12/09/2024])

Shen, W., & Zhang, T. (2018). Beyond accuracy: Evaluating pruned models for real-time deployment. *Proceedings of the European Conference on Computer Vision (ECCV)*, 87-102.

Shen, Z., Yan, X., Liu, H., & Liu, W. (2021). Pruning faster r-cnn for real-time object detection. *IEEE Transactions on Image Processing*, *30*, 4575-4588.

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

Singh, A., & Khan, N. (2022). Pipeline corrosion detection using yolov4 and uavs: A real-time monitoring system. *Corrosion Monitoring Review*, *42*, 225-237.

Smith, J., & Brown, M. (2021). Limitations of manual corrosion detection and the need for automation. *Automation in Industry*, *42*, 78-90.

Smith, J., & Doe, J. (2020). Economic and safety impacts of corrosion in industrial applications. *Journal of Corrosion Science*, *45*, 112-128.

TensorFlow. (2023). *Pruning in keras.* Retrieved from https://www.tensorflow.org/model_optimization

THOP, U. (n.d.). *Ultralytics thop: Tool to calculate model complexity.* https://github.com/Lyken17/pytorch-OpCounter. (Accessed: [Date of access])

Tune, R. (n.d.). *Ray tune: Scalable hyperparameter tuning.* https://docs.ray.io/en/latest/tune.html. (Accessed: [Date of access])

UbiOps. (2023). *How to optimize the inference time of your machine learning model.* Retrieved from https://ubiops.com

Ultralytics. (2023). Yolov8: New architecture and results. *GitHub repository*.

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, F., He, Z., & Liu, K. (2020). Pruning ssd for real-time object detection on mobile devices. *Proceedings of the European Conference on Computer Vision (ECCV)*, 150-164.

Wang, H., & Liu, X. (2022). Gaps in research on pruning yolo models: From yolov3 to yolov8. *Journal of Real-Time Computer Vision*, *47*, 58-73.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems.*

Yu, J., & Zhang, T. (2019). Challenges in deploying pruned models on hardware-accelerated platforms. *Proceedings of the ACM Symposium on Cloud Computing*, 234-246.

Zhang, W., & Xu, Y. (2021). Using drones for real-time corrosion detection with pruned yolo models. *Journal of Industrial Inspection*, *42*, 89-103.

Zhu, M., & Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. In *International conference on learning representations.*