Final Project

Lokesh Konjeti
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
lokeshko@buffalo.edu

Sai Lakshmi Narasimha Rao Edara
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
sedara@buffalo.edu

Aryan Rachala
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
aryanrac@buffalo.edu

Abstract

This is a project proposal to conduct a detailed study by comparing three state-of-the-art object detection models, namely YOLOv9, DETR (DEtection TRansformer) for the task of steel defect detection using the Severstal: Steel Defect Detection dataset. In the steel manufacturing industry, proper and efficient quality control methods are in demand. Fully automated defect detection with the inclusion of deep learning has great scope for improvement over tradi- tional manual inspection. Here, we put into practice the testing of these models, each with a different architectural approach: YOLOv9, which is a single-stage CNN, DETR, which is a transformer-based modelwhich is based on a feature pyramid network with focal loss. As part of this project, we would train the above-mentioned models on the Severstal dataset comprising over 12,000 images of steel surfaces classified into four kinds of defects: crazing, inclusion, patches, and pitted surfaces. This shall be done through rigorous pre-processing of the data, implementation of the models by transfer learning, and thorough eval- uations. We will compare our models using the standard metrics: MAP (Mean Average Precision), precision, recall, and F1-score. The inference speed, resource requirements, and performance of all the models with respect to the defect type and size are also analyzed. The expected outcome of this work is an insightful performance comparison between the three models, insights on strengths and weaknesses related to steel defect detection, and recommendations for industrial applications. This research study seeks to advance the field of computer vision in manufacturing by creating a benchmark for advanced object detection models used in quality control processes. The comparison among the diverse architectural approaches we will present in the future will form the basis for guiding manufacturers in selecting the best model relative to their specific needs, as ac- curacy, speed, and resource constraints are generally a trade-off. The outcomes of this project will be very important to enhance efficiency and reliability in quality control at steel production.

1 Introduction

1.1 Background and Motivation

Many steel manufacturing and associated industries depend on quality control measures to make sure a defect-free steel production. The manual inspection method takes a great deal of time and is subjective, relying completely on the person doing it. This makes it prone to human error. An automated defect-detection system using deep learning has shown promising results in this field over the years.

1.2 Project Summary

The current implementations of this particular use case are using YOLOv8 and Mask R-CNN Models to detect these defects. We plan to implement the latest object detection models such as YOLOv9, DETR object detection models using different hyperparameter tuning methods and compare these end results. In contrast, DETR uses a transformer encoder-decoder architecture as its core component for object detection.

1.3 Related Work

1.3.1 Deep Learning in Steel Defect Detection

Recent studies have proved the effectiveness of convolutional neural networks (CNNs) in detecting surface defects on steel. For example, a modified Faster R-CNN was used for real-time surface defect detection in hot-rolled steel strips.

1.3.2 YOLOv8 in Industrial Applications

YOLOv8, the latest iteration of the YOLO family, has demonstrated improved accuracy and speed in various object detection tasks. Wang et al. [1] (2023) applied YOLOv8 for defect detection in solar panels with promising results.

1.3.3 DETR for Complex Detection Tasks

Carion et al. [2] (2020) introduced DETR, which uses a transformer-based architecture for object detection. It has shown competitive performance on the COCO dataset and has potential for industrial applications due to its ability to handle complex scenes.

1.3.4 Comparative Studies

Several studies have compared different object detection models for industrial applications. For instance, Liu et al. (2022) compared YOLO, Faster R-CNN, and SSD for defect detection in textile fabrics.

2 Understanding the Data

The dataset has a total of 12598 images, out of which 6666 images have defects and 5902 images do not have defects. The figure below gives the distribution of the defects in the images having defects. We needed to convert the original format to Yolo Format and COCO format for the sake for training the models.

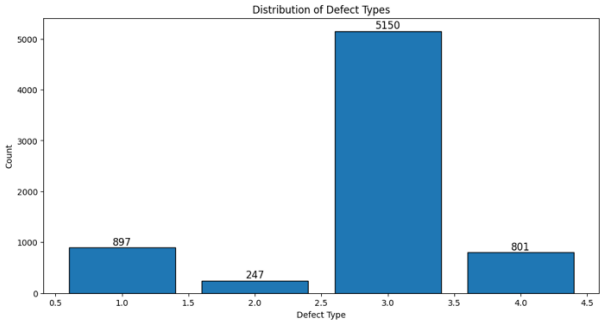

Figure 1: Distribution of Defects

3 Introduction to Models

We have used Advanced Object models such as Yolo and DERT (Transformers-based architectures) to build the project. We formally introduce these models in this section of the project report.

3.1 YOLOv9

The architecture of YOLOv9 is composed of four primary components: Backbone, Neck, Head, and a novel Auxiliary Branch. These components work together to enhance feature extraction, gradient flow, and computational efficiency. The figure below shows the architecture for YOLOv9.

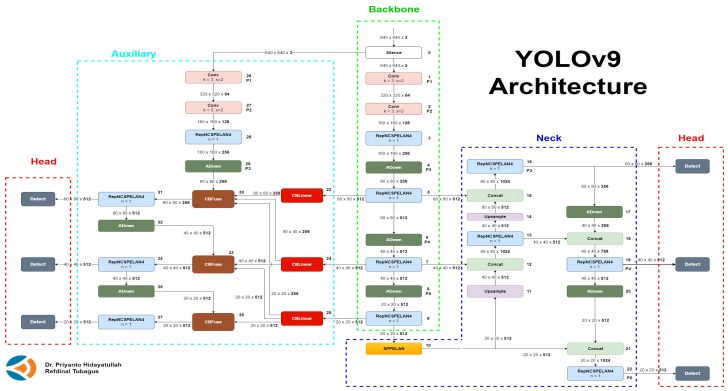

Figure 2: https://article.stunningvisionai.com/yolov9-architecture

3.2 DETR

The DEtection TRansformer (DETR) is a novel deep learning architecture for object detection that reframes the task as a direct set prediction problem. This approach eliminates the need for traditional components like region proposal networks (RPNs), anchor boxes, or non-maximum suppression (NMS), which are common in earlier object detection models.

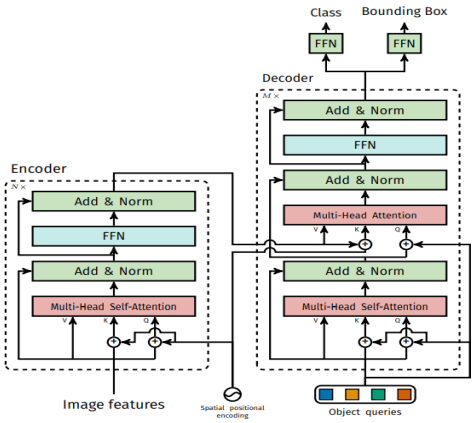

Figure 3: https://blog.roboflow.com/what-is-detr/

4 Analysis and Results

4.1 Model Results

We tried building all the models and fine-tuning them by manually changing a few parameters and using grid search methods. Below are the results for each of the models:

The images below show the F1-Confidence Curve for the YOLOv9 model and the corresponding predictions from the trained model.

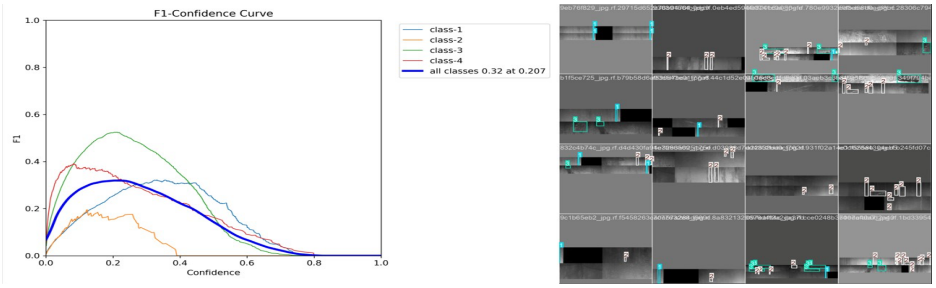

Figure 4: F1-Confidence Curve and Predictions

The overall model performance, represented by the thick blue curve, shows a maximum F1 score of 0.32 across all classes, achieved at a confidence threshold of 0.207, indicating the model performs best when considering predictions with confidence scores above this value. In terms of class-specific performance, Class 3 (green curve) exhibits the highest peak F1 score, suggesting it is the class the model performs best on. Conversely, Class 2 (orange curve) has the lowest F1 scores, indicating poor performance. Class 1 (blue curve) and Class 4 (red curve) show moderate performance, with their F1 scores peaking at different confidence thresholds.

The image to the right shows some of the predictions on the steel bars to detect the defects on the bars. The best model had a precision of 0.66, which is very suboptimal.

Similarly, the YOLOv9 and DETR models also showed low precision scores at 0.54. These results are printed as part of the code while running the test images on the trained model. More details about Precision, Recall, and mAP values can be found in the respective folder in the project repository.

4.2 Challenges

Even after using custom training on these pretrained models, the object detection seemed to be suboptimal. One major issue observed was that the models seemed to confuse horizontal lines in the steel bar images as defects, even when no defects were present. This confusion decreases significantly when using a confidence interval of 0.45 or above with the YOLOv9 model.

4.3 CNN Model Solution

To tackle this problem, we built a Convolutional Neural Network (CNN) on top of the existing models. The pipeline involves running the test images through the CNN model, which predicts if the given image has defects. If the CNN model predicts defects, we then use the YOLOv9 model weights to detect these defects on the image. This approach helps in reducing False Positives in predictions from the trained models.

We used a complex architecture similar to the VGG architecture to build the CNN model. The figures below show the Accuracy and Loss of the CNN Model trained over 10 epochs (due to computational limits). The final accuracy and loss values were accuracy: 0.9935, loss: 0.0272.

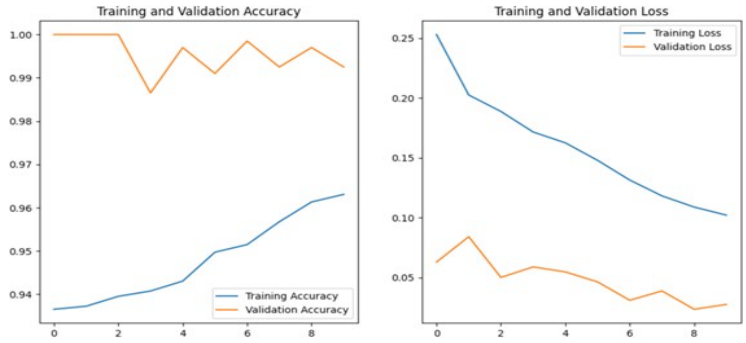

Figure 5: Accuracy and Loss of CNN model

4.4 Model Hyperparameter Tuning

While building the YOLO model, we experimented by changing various hyperparameters to fine-tune the training process. We adjusted parameters such as learning rate, weight decay, batch size, and different optimizers. After multiple iterations and optimization, we trained the model using the best hyperparameters to improve the evaluation metrics. Similarly, we also experimented with different learning rates, weight decays, and LR backbones for the DETR model.

5 Conclusion

We can conclude from comparing these models that straightforward object detection, although promising for many use cases, may not be the ideal solution for detecting defects in steel bar images. Building an ensemble of different models, such as the CNN model and implementing post-processing logic, may help make the solution more robust. Grower. (n.d.). YOLOv8 Segmentation Test. Kaggle. Retrieved from https://www.kaggle.com/code/grower/yolov8-seg-test

6. References

[1] Cao, Y., Pang, D., Zhao, Q., Yan, Y., Jiang, Y., Tian, C., & Wang, F. (2023). Improved YOLOv8-GD deep learning model for defect detection in electroluminescence images of solar photovoltaic modules. *Engineering Applications of Artificial Intelligence*, 131, 107866. Elsevier.

[2] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020) End-to-End Object Detection with Transformers. In Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I.

[3] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dolla´r, P. (2017) Focal Loss for Dense Object Detection. *IEEE Explore*

[4] *Kaggle. Retrieved from https://www.kaggle.com/code/grower/yolov8-seg-test*