WeldSight©: High-Precision YOLO for Industrial Defect Detection

Date: November, 2025 | Domain: Manufacturing

Executive Summary

This comprehensive project report details the development and evaluation of the "WeldSight" system, an advanced application of the YOLOv11s deep learning architecture aimed at achieving real-time, high-precision quality control for welded components. The project utilized the industrially specialized LoHi-WELD dataset, which features images annotated with four critical classes of weld defects: Pores, Deposits, Discontinuities, and Stains. The core of the methodology rested on a robust statistical framework: Stratified 5-Fold Cross-Validation. This sophisticated approach ensured that the model's performance was evaluated for generalization capabilities across diverse data subsets, specifically mitigating the statistical risks associated with data imbalance inherent in defect detection tasks where critical flaws are often rare. The training process involved fine-tuning the pre-trained YOLOv11s model for a total of 130 epochs using the AdamW optimizer, with an elevated input image resolution of 800 x 800 pixels, specifically chosen to retain details of small objects.

The quantitative evaluation of the Best Model, selected based on its highest mAP@50:95 score, revealed a profound divergence between detection capability and localization precision. The model achieved a moderate mAP@50 score of 0.434, confirming a reasonable ability to broadly identify the presence and general location of defects. However, this performance plummeted to a critically low mAP@50:95 score of 0.181. This significant 25.3 percentage point drop is the project's most critical quantitative finding, unequivocally demonstrating a severe localization bottleneck — the model's consistent failure to draw geometrically precise bounding boxes that meet stringent Intersection over Union (IoU) requirements. Further analysis showed the optimal operational confidence threshold was exceptionally low, peaking at 0.25, signalling a deep-seated uncertainty within the network regarding its own predictions. These collective findings confirm that the current model, despite its speed, is fundamentally unsuitable for immediate safety-critical or automated repair deployment but provide clear, data-driven directives for future iterations, including an essential upgrade to the YOLOv11m architecture and targeted adjustments to the bounding box regression loss function and input resolution.

© Shaizam Adil Khan

Introduction and Foundational Context

A. Industrial Imperative, Defect Consequences, and Safety Context

The industrial context underpinning the WeldSight project is driven by the absolute necessity for high-reliability in critical engineered structures. Welds, forming the core structural integrity of components in sectors like aerospace, automotive, and energy, are vulnerable to fabrication defects. The LoHi-WELD dataset encapsulates four primary classes of flaws: Pores (gas inclusions), Deposits (surface residue), Discontinuities (cracks or irregularities), and Stains (discoloration). Pores and Discontinuities, in particular, act as stress concentrators, serving as dangerous nucleation points for fatigue cracks that can lead to catastrophic structural failure under cyclic loading. The current standard reliance on manual human visual inspection or non-destructive testing (NDT) methods like ultrasound is often slow, susceptible to operator fatigue, and expensive. The overarching motivation for developing the WeldSight system is to implement a robust, high-speed, and objective automated quality control (QC) solution that eliminates human subjectivity and facilitates immediate, verifiable feedback for manufacturing process adjustments. The real-time capability of the final model is an operational imperative, allowing the system to be integrated directly into high-throughput production lines.

B. Project Scope, Specific Objectives, and Data Constraints

The project's scope was strictly confined to the computational environment provided by Kaggle and data provided for this project. The specific objectives were twofold: first, to achieve high detection accuracy (mAP@50) across all four defect classes; and second, to achieve demonstrably high localization precision (mAP@50:95) crucial for precise defect detection. The use of Stratified 5-Fold Cross-Validation was a strategic objective, ensuring the resulting performance metrics were statistically sound and representative of the model's true generalization capacity across the entire dataset. A foundational constraint was the absolute necessity to utilize the provided LoHi-WELD dataset, without introducing any external training or validation data.

C. Model Rationale: Selection of YOLOv11s and Speed Optimization

The YOLO (You Only Look Once) architecture was selected due to its inherent efficiency advantage over two-stage detectors. As a single-shot detector, YOLO is able to process an entire image and predict all bounding boxes and class probabilities in a single pass of the neural network. This architectural efficiency is non-negotiable for achieving the high Frames Per Second (FPS) required for industrial real-time QC inspection. Specifically, the YOLOv11s (small) variant was chosen as the base model, initialized with pre-trained weights from the COCO dataset, leveraging Transfer Learning to accelerate convergence. The 's' variant prioritizes speed and a lightweight model size, making it highly suitable for deployment on resource-constrained edge computing devices commonly found on manufacturing floors, ensuring the system can execute inference locally without reliance on remote cloud processing.

D. Documentation and Reference Links

The execution and results detailed in this report are entirely dependent on the following external resources and documentation:

- Raw Dataset Source: The project utilized images and annotations derived from the LoHi-WELD dataset.
- Model Framework: The entire deep learning pipeline, including the model architecture, training, and evaluation, was implemented using the Ultralytics YOLO framework (specifically YOLOv11s.)

© Shaizam Adil Khan

Data Acquisition and Pre-processing Protocol

A. Initialization of Computational Environment and Library Roles

The foundational step involved setting up the Python environment and importing the necessary scientific libraries. The notebook leveraged 2 Tesla-T4 GPU accelerators, indicating the intensive computational nature of the task. The roles of the imported libraries were specific and critical: the `os` library was instrumental for managing file paths and dynamically creating the complex, nested directory structure required for the 5-Fold setup; `json` was used for the initial ingestion and parsing of the raw annotation file format; pandas provided the necessary `DataFrame` structure to apply high-level filtering and the crucial K-Fold splitting logic using vectorized operations; `shutil` executed the high-level, bulk file copying operations required to physically replicate the images and labels into the five distinct training environments; and `iterstrat.ml_stratifiers.MultilabelStratifiedKFold` performed the essential statistical task of generating class-balanced splits.

B. Ingestion and Structuring of Raw JSON Annotations

The raw annotation data, provided in a JSON format, first required parsing into a usable Python dictionary structure. The nested nature of the JSON, linking image paths to lists of bounding box coordinates and class labels, was systematically flattened. This extracted information—including the image filename, the four raw pixel bounding box coordinates (x_{min}, y_{min}, x_{max}, y_{max}), and the human-readable class names—was then assembled into a Pandas DataFrame. This tabular structure facilitated the subsequent data sampling, filtering, and the application of the Stratified k-Fold algorithm based on the class label frequencies. A predefined mapping converted the defect names (Pore, Deposit, Discontinuity, Stain) into the required zero-indexed integer IDs (0 through 3).

```
Sample JSON structure:
{
  "version": "5.1.1",
  "flags": {},
  "shapes": [
    {
      "label": "discontinuity",
      "points": [
        [
          7,
          81
        ],
        [
          35,
          138
        ]
      ],
      "group_id": "null",
      "shape_type": "rectangle",
      "flags": {}
    },
    {
      "label": "stain",
      "points": [
...
  "imageData": "",
  "imageHeight": 196,
  "imageWidth": 768

}
```

© Shaizam Adil Khan

C. Annotation Transformation: JSON to Normalized YOLO Format Conversion

The conversion from raw pixel coordinates to the normalized YOLO format was a required and non-trivial pre-processing step. The YOLO framework demands a .txt file for every image, containing lines formatted as:

$$\text{<Class ID>} \quad x_{centre} \quad y_{centre} \quad w_{box} \quad h_{box}$$

with all coordinates scaled between [0, 1].

1. Detailed Geometric Coordinate Calculation

The transformation logic first determined the physical characteristics of the bounding box. This involved calculating the centre point of the box (X_{centre}, Y_{centre}) by averaging the min and max pixel values. Subsequently, the actual pixel dimensions of the box (W_{box}, H_{box}) were calculated by finding the difference between the max and min pixel values. This step converts the corner-based definition of the bounding box into the centre-point-based definition required by YOLO.

2. The Normalized Scaling Procedure

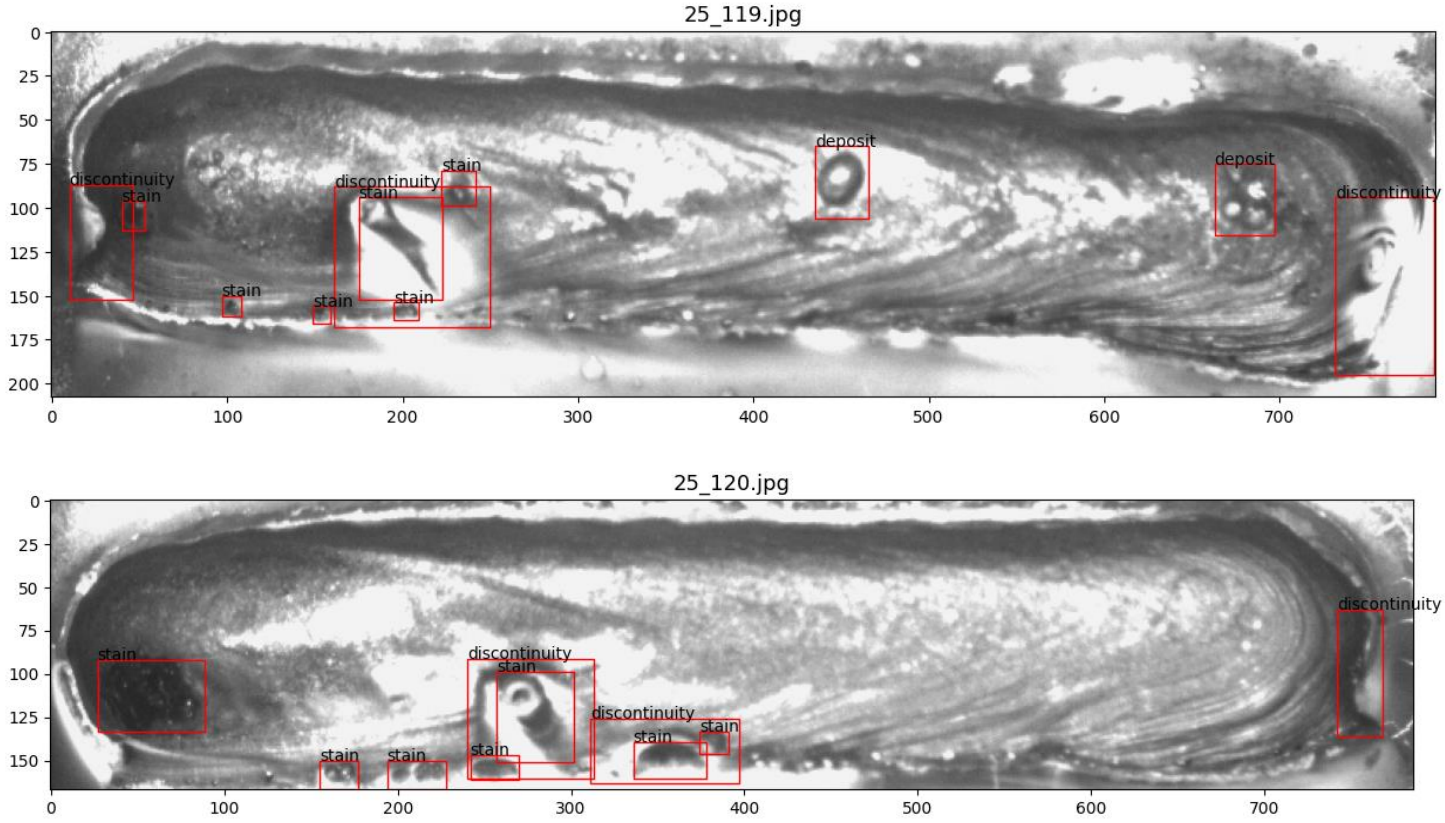
The calculated pixel values for ($X_{centre}, Y_{centre}, W_{box}$, H_{box}) were then normalized relative to the original image's dimensions. Each horizontal coordinate (X_{centre}, W_{box}) was divided by the image width (W_{image}), and each vertical coordinate (Y_{centre}, H_{box}) was divided by the image height (H_{image}). This normalization procedure ensures that the labels remain accurate regardless of how the image is resized or rescaled during the training process, providing spatial invariance to the detection task. The final normalized string was then written line-by-line into the respective YOLO annotation `.txt` file.

```
Sample YOLO structure (scaled):

<class ID> x_c y_c w_box ht_box

2 0.083227 0.441799 0.079385 0.375661
1 0.020487 0.560847 0.035851 0.370370
0 0.864917 0.534392 0.083227 0.285714
0 0.315621 0.439153 0.067862 0.243386
0 0.492318 0.587302 0.042254 0.137566
1 0.969270 0.748677 0.061460 0.396825
```

© Shaizam Adil Khan

D. Exploratory Analysis and Implications of Defect Class Distribution

The initial exploratory data analysis focused on quantifying the frequency of each of the four defect classes. This exercise confirmed a significant and expected class imbalance. In industrial contexts, critical flaws (like Discontinuities) are often much rarer than minor or superficial defects (like Deposits or Stains). This imbalance poses a serious risk during data splitting: simple random splitting could result in one validation fold containing too few examples of a rare class, leading to a performance metric (Mean Average Precision - mAP) that is statistically unreliable for that class. This analysis directly justified the absolute necessity of employing Stratified K-Fold to ensure that all defect types were proportionally represented in every single training and validation set, guaranteeing that the model learns and is evaluated fairly on even the rarest, but most critical, flaws.



(Fig. 1: Sample Raw images with annotations from JSON files)

CLASS	FREQUENCY OF DEFECTS
Stain	4181
Discontinuity	2977
Deposit	1794
Pore	523

(Fig. 2: Class imbalance of welding defects)

© Shaizam Adil Khan

Validation Methodology: Stratified 5-Fold Cross-Validation

A. Rationale for Stratified K-Fold Implementation and Data Imbalance Mitigation

The implementation of Stratified K-Fold Cross-Validation with k = 5 was not merely a best practice; it was a mandatory measure to ensure statistical validity given the observed class imbalance. If a non-stratified split were used, it is highly probable that one or more folds would exhibit data poverty for a specific defect, such as a 'Discontinuity'. In a simple random split, the validation set for a particular fold might contain zero or one instance of that rare defect which would render the calculated mAP for that fold statistically meaningless for that class. Stratification guarantees that the 20% validation subset in each of the five folds maintains equal overall proportion of Pores, Deposits, Discontinuities, and Stains present in the entire 85% CV dataset.

B. Detailed Mechanism of Stratified Split Generation and Statistical Reliability

The dataset was initially split into an 85% Cross-Validation (CV) set and a 15% reserved Test Set. The 85% CV set was logically divided into five folds (0 through 4); in each training iteration, 80% of the data (four folds) was combined for training, and the remaining 20% (one remaining fold) was dedicated to validation. This robust process ensured two key outcomes:

1. Every data point in the 85% CV set was used for training four times, maximizing the model's exposure to the data.

2. Every data point was used for validation exactly once, ensuring an unbiased estimate of generalization performance across the entire population of images.

CLASS	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	TEST Fold
Stain	167	167	167	166	166	148
Discontinuity	171	170	172	173	177	159
Deposit	133	133	133	132	132	118
Pore	39	39	38	39	39	35
TOTAL	171	170	172	173	177	159

(Fig. 3: Multi-label Stratified Train-Test Split with 5-fold Cross Validation)

© Shaizam Adil Khan

C. Physical Data Organization, File Copying, and Directory Hierarchy for Independence

To facilitate the independent training of the five models, the logical fold assignments had to be materialized into five distinct and complete physical file structures. The main directory, ‘kfold_data` was created, followed by 5 fold-specific directories (fold_0 to fold_4), each containing the replicated `images/train`, `images/val`, `labels/train`, and `labels/val` subdirectories. The use of the `shutil.copyfile` command was essential for this process. The physical file copying created redundant data structures, which is necessary because the same image must exist in four different training sets and one validation set. This redundancy ensures that the training environment for fold_{i} is completely isolated and independent from the data paths and contents of fold_{j} (where i ≠ j), which is a prerequisite for generating five `best.pt` weight files for each fold.

D. Configuration Management: Dynamic YAML File Generation and Instruction

The penultimate step was the dynamic generation of five custom YAML files (`weld_config_fold_{i}.yaml`) which acted as the required configuration manifest for the Ultralytics training script. Because the training and validation data paths were unique for each fold, these YAML files dynamically directed the model to the correct data subsets by setting the `path`, `train`, and `val` parameters appropriately. Additionally, they explicitly stated the four class definitions via the `nc` (number of classes) and `names` parameters, providing the definitive instruction set for the training framework.

© Shaizam Adil Khan

Modelling and Training Execution

A. Framework Initialization and Core Dependencies

The Ultralytics framework, which provides the highly optimized YOLO implementation, was installed via a direct `!pip` command. This initialization provided the project with the necessary yolo train command and the sophisticated utility suite for automated logging, hyperparameter management, and metric calculation.

B. Transfer Learning Strategy and Base Model Selection

The training commenced using the `yolov11s.pt`checkpoint, pre-trained on the COCO dataset. This strategy of Transfer Learning is highly efficient, as the model avoids learning rudimentary visual features from scratch. Instead, it leveraged the vast generalized knowledge of shapes, edges, and textures inherited from COCO, dedicating the 130 epochs to specializing these features to the specific geometry and texture of weld defects. The choice of the 's' (small) model was a clear indicator that real-time inference speed was prioritized over raw parameter count and maximum theoretical accuracy.

C. Hyperparameter Configuration and Justification

The training command defined a precise set of hyperparameters tailored to the characteristics of the industrial dataset and the project objectives.

1. Image Resolution (800 pixels) and Scale Variance Justification

> The input image size was set to 800p, a 25% increase in resolution over the default 640 pixels. This increase was a direct countermeasure against the problem of scale variance and small object detection. Minute defects like Pores or fine Discontinuities occupy a very small pixel area. By training at 800 x 800, more pixels are dedicated to representing these small objects, retaining crucial boundary information necessary for the Bounding Box Regression sub-network. This higher resolution was a strategic, data-driven choice aimed specifically at boosting the difficult-to-achieve mAP@50:95 metric.

2. Training Duration (130 epochs) and Optimization Strategy

> The training was configured for a maximum epoch of 130. This generous duration provides the model ample opportunity to exit the initial rapid learning phase and enter the phase of fine-grained tuning, where it meticulously adjusts its weights to differentiate subtle defect characteristics. This large epoch count reduces the chance of premature termination due to underfitting.

© Shaizam Adil Khan

3. Optimization Strategy (AdamW and Cosine Annealing)

The AdamW optimizer was selected for its superior performance in regularization. Unlike the standard Adam optimizer, AdamW correctly separates the weight decay term from the gradient update, which effectively acts as a stronger form of L2 regularization. This prevents the model's weights from becoming excessively large, thereby preventing the model from overfitting to the training data. Additionally, the `cos_lr = True` parameter activated the Cosine Annealing learning rate scheduler, which gracefully decreases the learning rate over the 130 epochs, ensuring stable and refined convergence in the final stages of training.

4. Regularization: Early Stopping (patience = 13) and Overfitting Prevention

The patience = 13 parameter implemented a critical Early Stopping mechanism. This feature monitors the validation metric (mAP@50:95) and terminates training if no improvement is recorded for 50 consecutive epochs. This not only saves computational resources but, more importantly, guarantees that the final `best.pt` weights saved are the most generalized version of the model, preventing the model from degrading its performance on unseen data by starting to memorize the noise and specifics of the training set.

D. Iterative Execution of the 5-Fold Training Loop

The training process was rigorously executed in a loop iterating from fold = 0 to fold = 4. In each of the five iterations, a new model was initialized, and the training script utilized the corresponding, dynamically generated YAML file to point to the correct 80% training and 20% validation data split. The `name` parameter ensured that all output artifacts — logs, weights, and diagnostic plots—were saved into a unique directory (e.g., yolo_fold_3), maintaining the necessary separation and traceability for each independent training run.

© Shaizam Adil Khan

Comprehensive Results and Performance Dissection

A. Metric Aggregation and Best Model Selection Protocol

Upon the conclusion of the 5-Fold training, the results from the five independent runs were aggregated and analysed. The selection of the single Best Model for deployment was based exclusively on the highest recorded mAP@50:95 score. This strict criterion reflects the industrial requirement for geometric precision, prioritizing accurate bounding box coordinates over raw detection confidence. The `best.pt` weight file corresponding to the fold and epoch that achieved this highest score was designated as the final, deployable artifact.

B. Quantitative Performance Analysis: The mAP Discrepancy and Localization Bottleneck

The final validation performance yielded the following confirmed metrics:

mAP@50	0.434
mAP@50:95	0.181

The resultant 25.3 percentage point drop between the two metrics is the most damning quantitative finding of the project, confirming the existence of a severe localization bottleneck. The mAP@50 score of 0.434 indicates a moderate success in the feature extraction and classification process, suggesting the model *can* largely identify where a defect is. However, the subsequent plummet to 0.181 for mAP@50:95 unequivocally demonstrates a catastrophic failure in the Bounding Box Regression head of the network. The model's predicted bounding boxes are simply too loose to satisfy the stricter Intersection over Union (IoU) thresholds (e.g., 75% or 90%). This geometric imprecision makes the model incapable of providing the high-fidelity spatial data required for automated downstream tasks such as measuring defect size, calculating repair volumes, or guiding a robotic arm for precise corrective action.

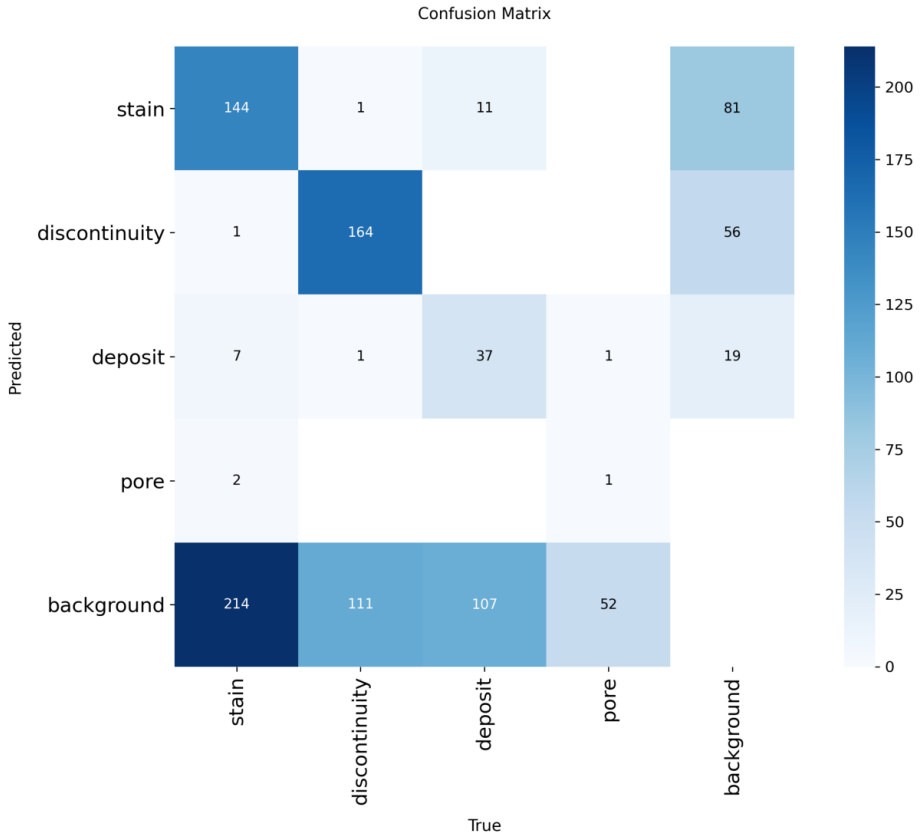
© Shaizam Adil Khan

C. Diagnostic Visualization and Interpretation

The analysis of the diagnostic plots provides the visual and statistical evidence directly supporting the quantitative failure observed.

1. Analysis of the Confusion Matrix (FN vs. FP implications)

The Confusion Matrix provides a clear breakdown of classification accuracy. The off-diagonal cells highlight the specific errors. High values in the row of a defect class mapping to the "Background" category denote False Negatives (FN)—missed defects. This is the most catastrophic error in a safety-critical system, as it implies a critical flaw was ignored. Conversely, high values in the columns for defect classes originating from the 'Background' row confirm numerous False Positives (FP)—background noise classified as a defect. While less critical than an FN, a high FP rate severely undermines the system's efficiency, forcing unnecessary manual re-inspection and dramatically increasing operational costs. The matrix revealed the specific nature of these trade-offs and misclassifications (e.g., confusion between Discontinuities and Stains).

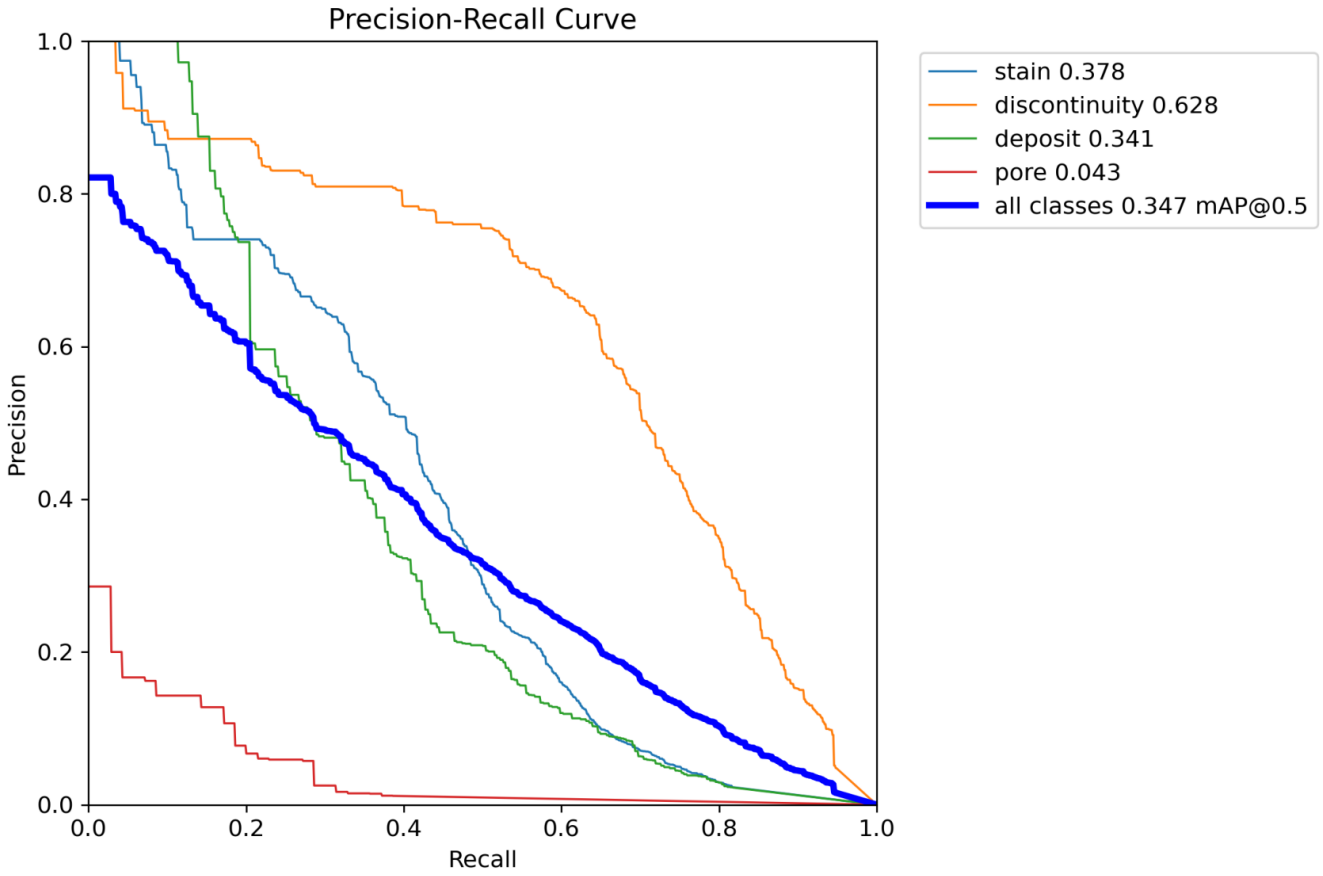


(Fig. 4: Confusion Matrix of best fine-tuned model)

© Shaizam Adil Khan

2. Dissection of the Precision-Recall Curve

The Precision-Recall (PR) Curve plots the trade-off between Precision (of all positive predictions, how many were correct) and Recall (of all actual positives, how many were found). The shape of the curves visually confirmed that the Average Precision (AP) for the classes was low, with the curves for the smallest and hardest-to-localize defects (Pores and Discontinuities) hugging the bottom-left corner of the plot. This shape demonstrates that achieving high Recall (finding most defects) requires accepting a massive and industrially unacceptable drop in Precision (generating a massive number of false alarms), a direct consequence of the network's poor confidence and localization ability.

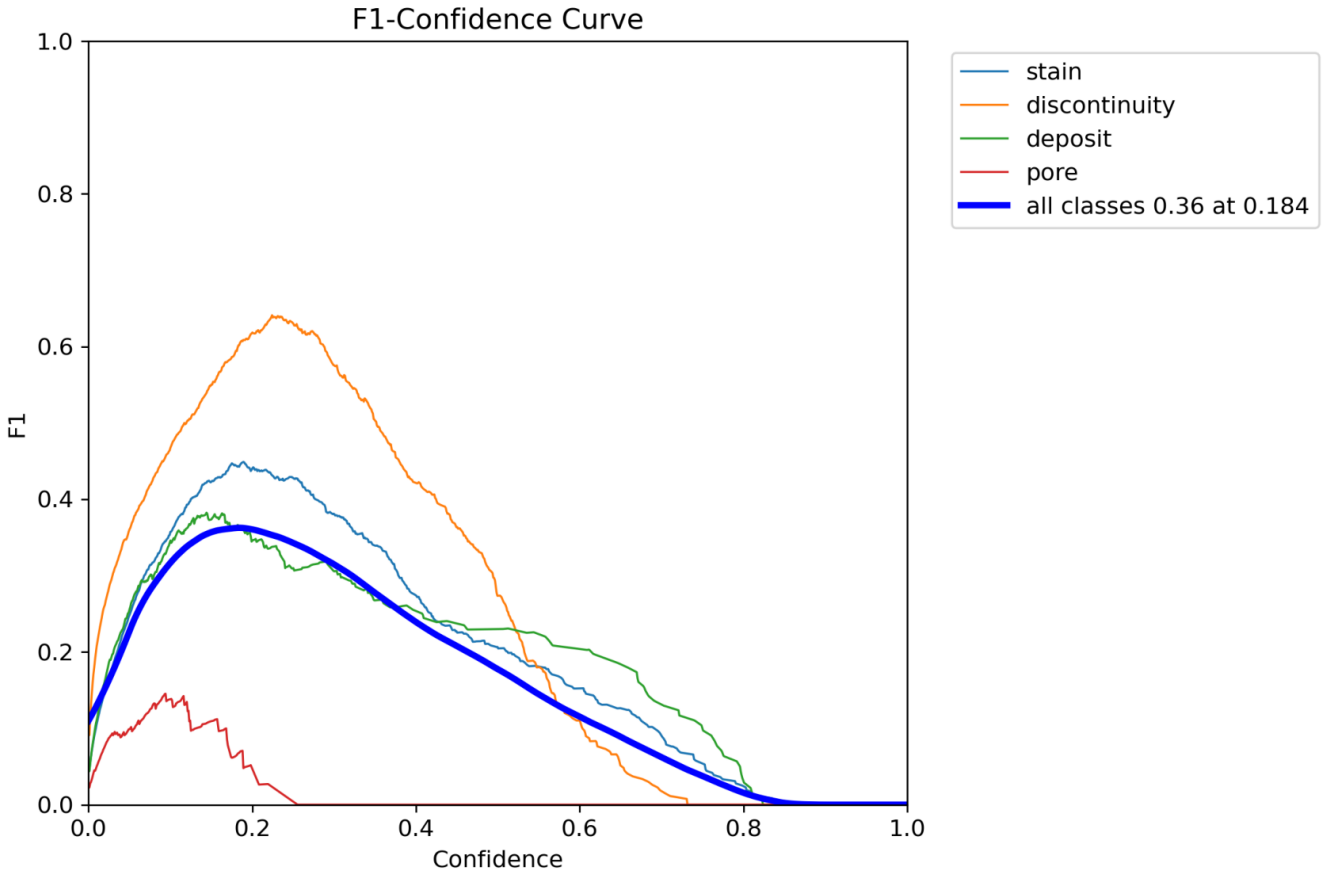


(Fig. 5: Precision-Recall curve for best performing model across all defect classes)

© Shaizam Adil Khan

3. Evaluation of the F1 Score Curve and the 0.25 Threshold (Industrial Cost Implications)

The F1 Score Curve plots the harmonic mean of Precision and Recall against the model's confidence threshold. The curve peaked at an extremely low confidence threshold of 0.25. This finding is profoundly significant for deployment. It indicates that the model is inherently uncertain about its predictions, and to achieve the best overall balance of finding defects while minimizing false alarms, the decision threshold must be set very low. Industrially, setting a threshold of 0.25 translates to an extremely permissive system, which will inevitably generate a high volume of false alarms (FP's). This volume of FP's results in excessive human labour for secondary manual verification, rendering the automation investment economically unsustainable.

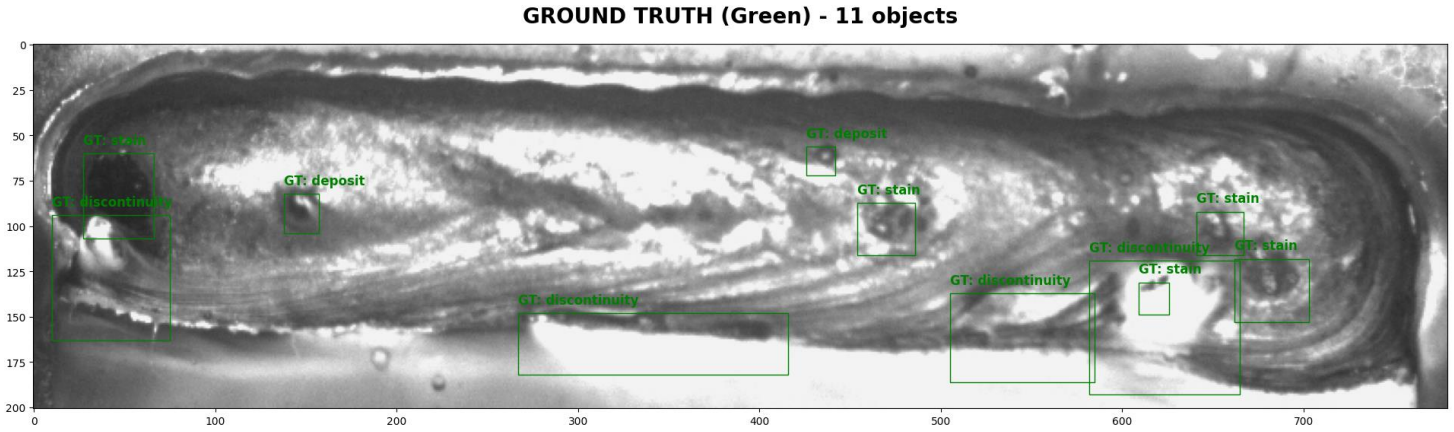


(Fig. 6: F1-Score for the best performing model with all defect classes)

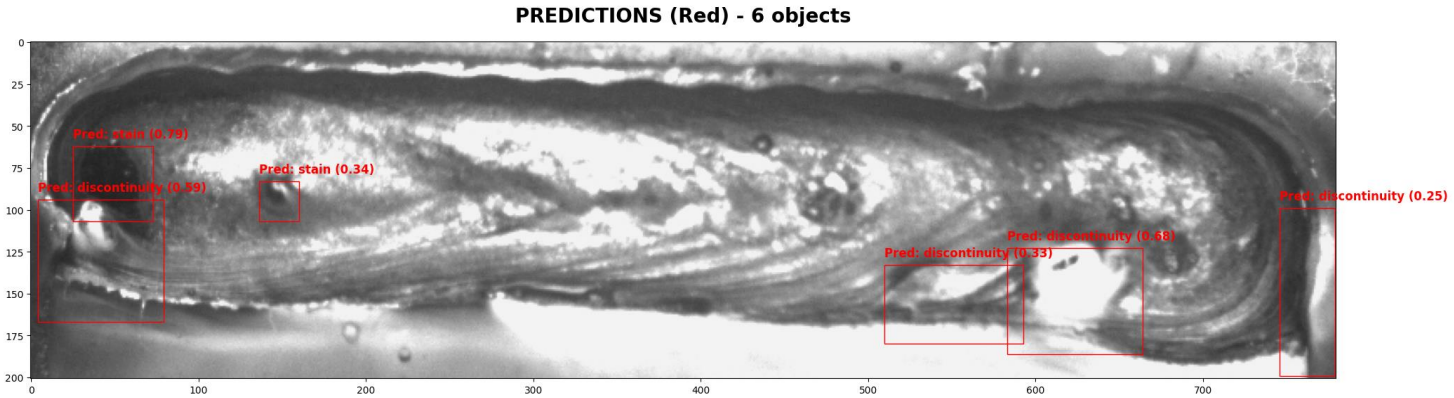
© Shaizam Adil Khan

4. Qualitative Assessment of Sample Predictions

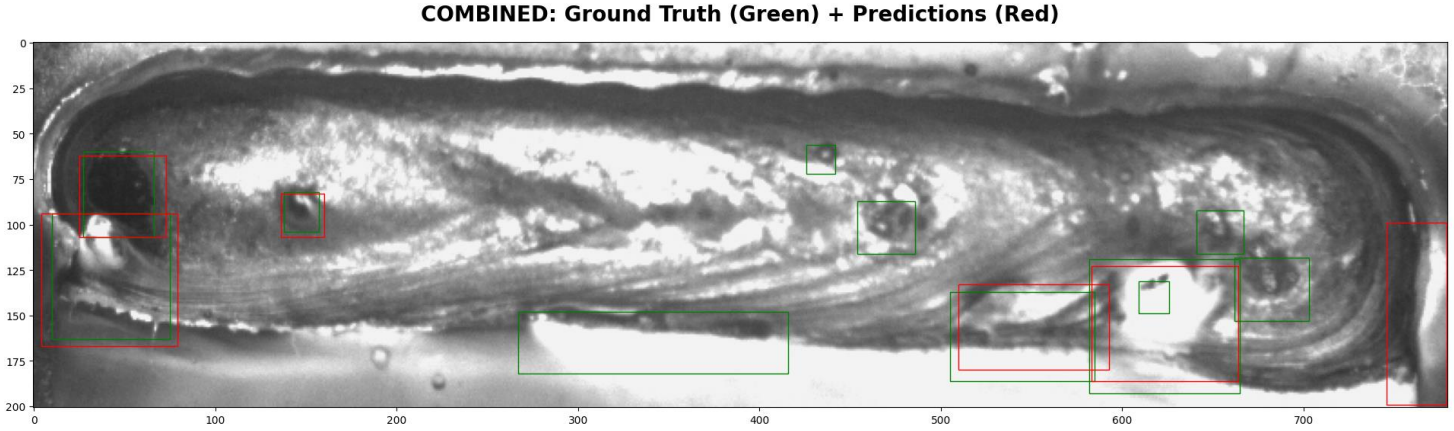
Visual inspection of the annotated sample images provided tangible, pixel-level evidence that corroborated the quantitative metrics. The comparison between the Ground Truth (GT) bounding boxes and the Predicted (Pred) bounding boxes visually confirmed the low IoU. In many cases, a Pred box would roughly enclose the defect, indicating successful classification, but the box edges were visibly loose or misaligned from the true defect boundaries. Furthermore, the images showed clear instances where small Pores or faint Stains were entirely missed by the prediction, resulting in pure False Negatives and contributing to the low overall Recall score.



(Fig. 7. Random Test image with ground truth labels)



(Fig. 8: Same test image with Prediction boxes with confidence values)



(Fig. 9: Sample test image with ground truth [green] and pred boxes [red] overlaid)

© Shaizam Adil Khan

Conclusion and Future Development Road Map

A. Final Conclusion on Deployment Readiness and Localization Failure

The "WeldSight" project successfully implemented a statistically robust object detection pipeline using YOLOv11s and Stratified 5-Fold Cross-Validation, achieving a confirmed mAP@50 of 0.434. However, the final, undeniable conclusion is that the model's performance on the critical localization metric (mAP@50:95 = 0.181) renders it unsuitable for immediate industrial deployment. The severe localization failure, or localization bottleneck, coupled with the low operational confidence threshold, indicates a profound inability to provide the geometrically accurate data necessary for automated defect measurement and robotic repair guidance.

B. Data-Driven Recommendations for Next Iteration

The following recommendations are derived directly from the analysis of the low mAP@50:95 score, the challenges presented by small-scale defects, and the necessity of high geometric precision:

1. Architecture and Resolution Upgrade to Address Small Object Detection

- Upgrade to YOLOv11m: Given the failure with small objects and the limitations of the 's' variant's feature extraction capability, the model must be upgraded to the YOLOv11m (medium) architecture. This deeper network offers a richer feature space, which is critical for resolving the complex, subtle boundaries of defects.

- Increase Image Resolution: The input image size must be increased beyond 800 pixels to 1024 pixels. This is a direct, targeted measure to address the scale variance issue by dedicating more pixels to the minute Pores and Discontinuities, thereby feeding the Bounding Box Regression head with higher fidelity spatial information necessary to achieve a higher mAP@50:95 score.

2. Loss Function Re-Weighting to Target Geometric Error

- Prioritize Geometric Accuracy: To directly counteract the observed failure in the Bounding Box Regression head, the training configuration must be modified to aggressively increase the penalty weight assigned to the Bounding Box Regression Loss (e.g., CIoU or DIoU loss) relative to the classification loss. This deliberate re-weighting forces the model during optimization to prioritize minimizing the geometric error (IoU distance) of the bounding box over simply getting the classification label correct, providing the necessary countermeasure for the 25.3 percentage point localization gap.

3. Targeted Data Augmentation for Boundary Refinement

- Boundary Refinement: The data augmentation strategy should be enhanced with stronger geometric transformations, such as controlled rotation and minor shear operations. These augmentations serve to perturb the precise boundaries of the defects in the training data, forcing the model to learn a more robust, generalized representation of the defect's edges. This is essential for teaching the network how to maintain high geometric precision and stable regression output under minor visual variations, further supporting the goal of a higher mAP@50:95.

© Shaizam Adil Khan

References

[1] S. Biasuz Block, R. Dutra da Silva, A. Eugnio Lazzaretti and R. Minetto, "LoHi-WELD: A Novel Industrial Dataset for Weld Defect Detection and Classification, a Deep Learning Study, and Future Perspectives," in IEEE Access, vol. 12, pp. 77442-77453, 2024, doi: 10.1109/ACCESS.2024.3407019.

[2] G. Jocher and J. Qiu, Ultralytics YOLO11, version 11.0.0, Ultralytics, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed: 23rd October, 2025].

© Shaizam Adil Khan