BACHELOR THESIS PROJECT

CONCRETE CRACK DETECTION AND QUANTIFICATION USING COMPUTER VISION

Under the supervision of
Prof. Damodar Maity

Kolathuru Vishnu Vardhan
21CE3AI13

OBJECTIVE

- **Automate** crack detection to eliminate **time-consuming** and **error-prone** manual inspections
- Provide a scalable solution for **large-scale infrastructure** by analyzing datasets quickly
- Quantify cracks by measuring dimensions to **prioritize repairs** effectively
- Deliver **real-time detection** for faster responses and proactive maintenance

SCOPE OF WORK

- Develop an automated system using YOLO for accurate **crack detection** and **localization** in concrete structures
- **Quantify** detected cracks by measuring their **dimensions** (length, width, area) to assess severity
- Ensure the system works effectively, and consistently for monitoring large structures

LITERATURE REVIEW & INSIGHTS

Traditional Methods:

- **Manual inspections** are subjective, slow, and error-prone
- Basic **image processing** techniques like edge detection struggle under complex conditions like poor lighting or rough surfaces

Machine Learning (ML):

- **SVM** models improved crack detection by automating classification
- However, **manual feature extraction** reduced flexibility, and these models struggled with diverse environments and complex backgrounds

Deep Learning (DL):

- DL models like ResNet, R-FPANet, and **YOLOv5** automate feature extraction, achieving high accuracy and precise crack measurements
- YOLOv5 excels in real-time crack detection and localization, making it ideal for large-scale monitoring

RESEARCH GAPS

- Struggles with **irregular cracks** compared to linear ones
- Limited ability to handle **large datasets** and changing **environmental conditions**
- Resource-intensive models like **YOLOv5** are less efficient for extended training or deployment

OUR APPROACH

- Chose **YOLO** for simultaneous classification and localization, unlike traditional methods that separate these tasks
- Offers **fast** and **accurate** real-time crack detection with minimal processing delay
- More effective than both **traditional CNNs** and rule-based **OpenCV** techniques
- Capable of handling cracks of different sizes and orientations under real-world conditions

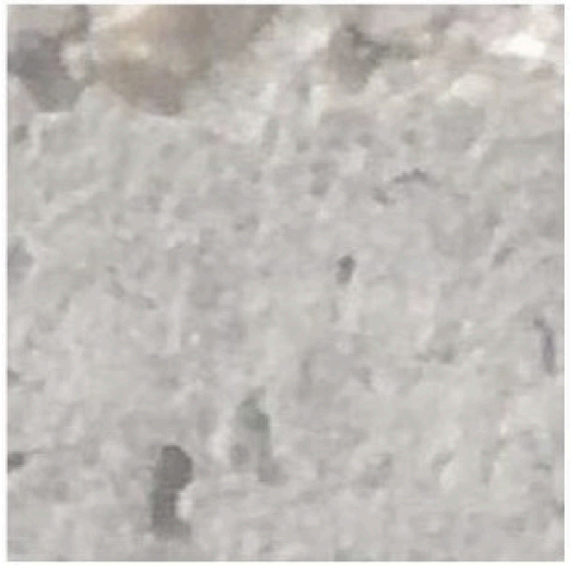
WORKFLOW

1.Data Collection

- Downloaded from the public **METU image database** on Kaggle
- **Two classes:** cracked and uncracked areas
- **Dataset size:** 40,000 images (227 x 227 pixels), 20,000 images per class
- For training the YOLO model, a subset of 1,900 images was selected

Cracked areas

Uncracked areas



2.Data Preprocessing

- **Resize (Stretch to 256x256):** YOLO models work best with images sized at **256x256, 640x640**, etc., as these dimensions are optimized for their architecture
- **RGB to Black and White Conversion:** Reduces computational complexity and lets the model focus on essential features like edges and patterns

3.Data Annotation

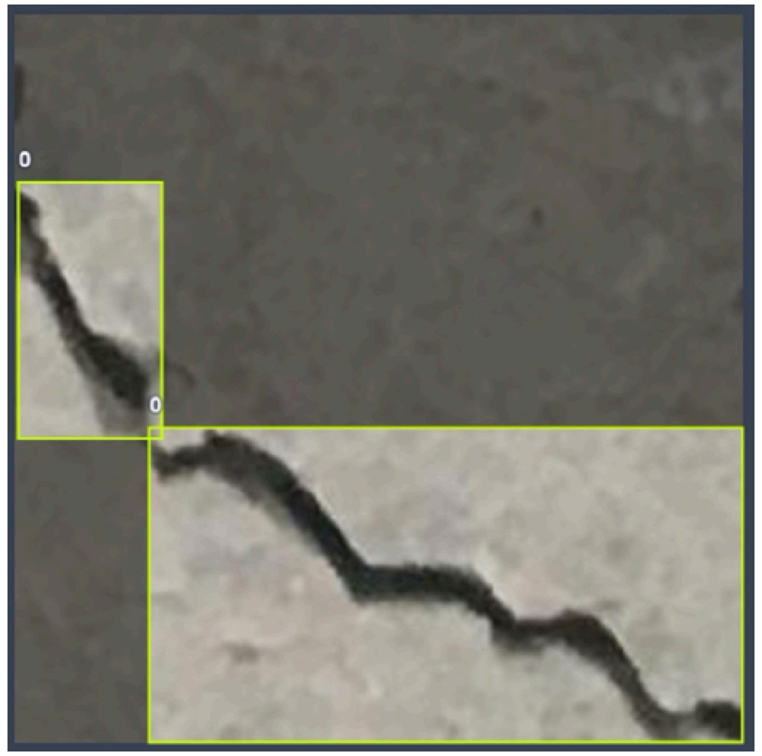
Tool Used: Roboflow, which generated labels in the **.txt** format:

0 0.502203 0.621145 0.995595 0.220264

Explanation:

- **0:** Class label (0 indicates "cracked" in this case)
- **0.502203** and **0.621145:** Coordinates of the object's center (X, Y), normalized between 0 and 1
- **0.995595** and **0.220264:** Width and height of the bounding box, respectively, normalized between 0 and 1

Annotated Image



4.Data Augmentation

- Used flipping, rotations (-10° to +10°), and cropping with 10% zoom to diverse crack orientations and perspectives
- Increased dataset from around **1,900** to **3,300** images, improving model accuracy and robustness

Methodology

Determine Optimal Background Image %:

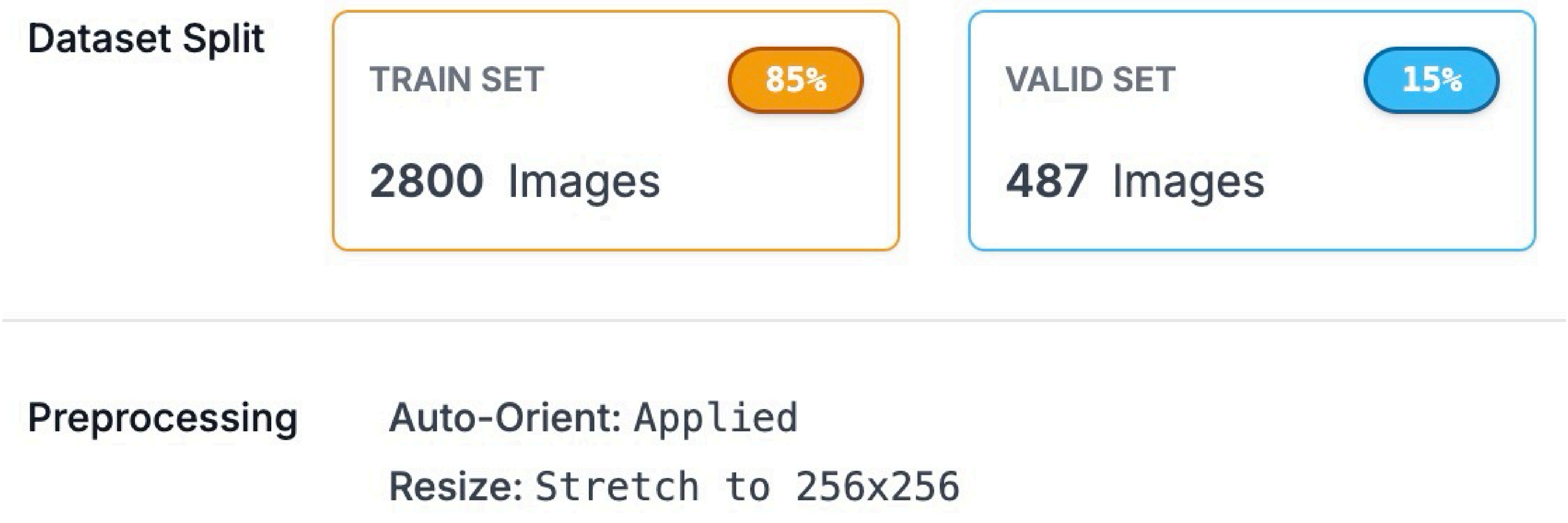
- Used **YOLOv10n** to find the best **BG% (1-10%)** by training for **50** epochs with batch size **64**

Find Optimal Batch Size:

- After fixing the optimal BG%, tested batch sizes **64, 32, 16, 8** on YOLOv10n for **30** epochs

Experiment with YOLOv10 Variants:

- With optimal BG% and batch size, trained **YOLOv10n, s, m, l** models for **100** epochs



Find Optimal Model:

- Based on a **trade-off** between **Precision, Recall, mAP@50, mAP@50-95,** and **training time**, we determined the best model, BG%, and batch size for crack detection

Object Detection Metrics

Precision:

- Measures how many of the detected cracks are actual cracks

Recall:

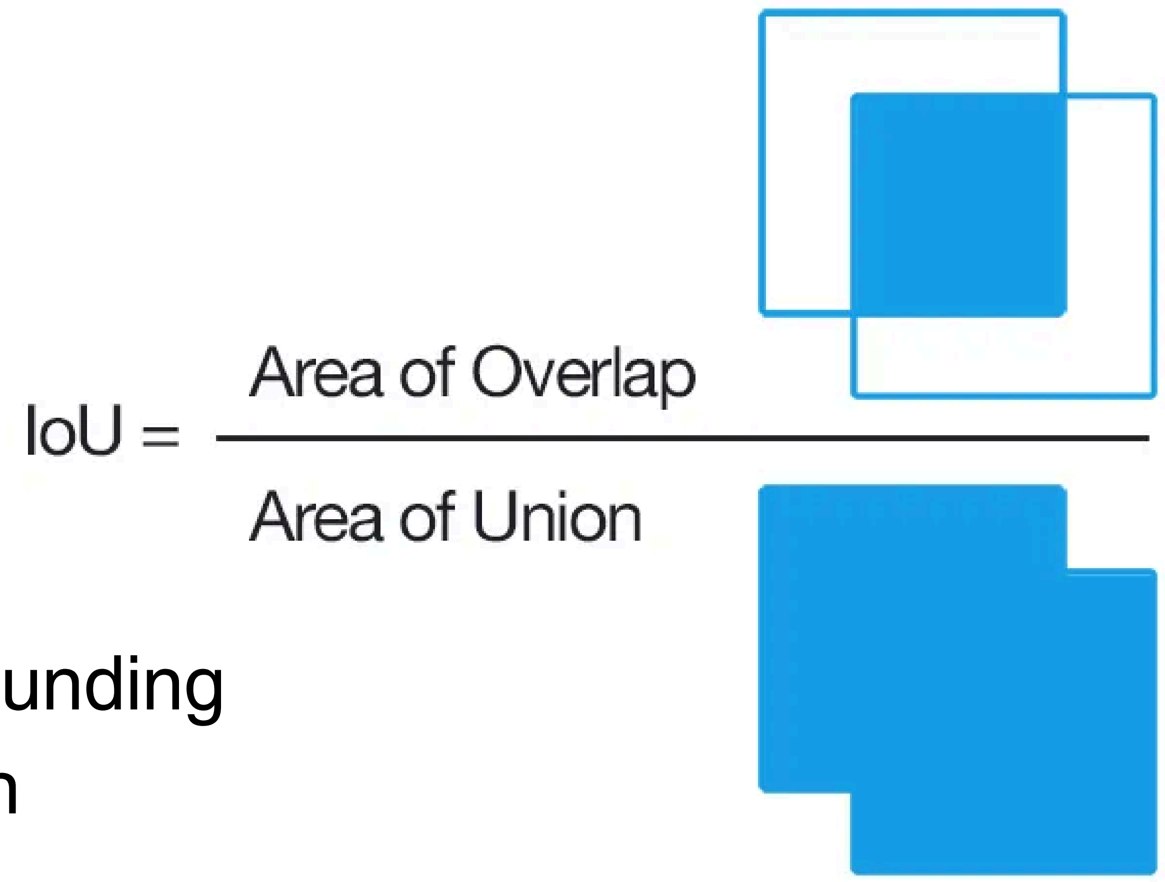
- Measures how many of the actual cracks are detected

mAP@50:

- Evaluate the model's ability to detect cracks with at least **50% Intersection over Union (IoU)** between predicted and actual bounding boxes. A high mAP@50 indicates accurate and reliable detection

mAP@50-95:

- A stricter metric that averages mAP across IoU thresholds from **0.50** to **0.95**, testing both detection and localization precision



Most Important Metric for Crack Detection?

- Focus on **mAP@50** as it directly measures detection accuracy, which is critical for ensuring cracks are reliably identified in practical scenarios.

Model Insights

Table 1 (50 epochs)

BG Image %	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
1	0.266	0.72	0.704	0.767	0.583
2	**0.29**	**0.81**	**0.702**	**0.818**	**0.632**
3	0.3	0.798	0.706	0.812	0.632
4	0.302	0.76	0.729	0.8	0.622
5	0.306	0.774	0.688	0.789	0.612
6	0.311	0.741	0.691	0.783	0.598
7	0.315	0.81	0.703	0.809	0.618
8	0.307	0.758	0.743	0.814	0.621
9	0.303	0.796	0.686	0.799	0.61
10	0.299	0.802	0.7	0.814	0.64

From the insights of Table 1:

- **YOLOv10n** trained with various Background Image % at **batch size 64** for **50 epochs**
- **Optimal background** percentage set at **2%** for best performance and training time
- Added **56 uncracked** images to training set, totaling **2,856** images

Table 2 (30 epochs)

Batch Size	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
64	**0.171**	**0.788**	**0.651**	**0.788**	**0.632**
32	0.229	0.732	0.699	0.782	0.604
16	0.346	0.703	0.722	0.769	0.596
8	0.519	0.774	0.692	0.786	0.602

Table 3 (100 epochs)

Model	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
YOLOv10n	0.532	0.805	0.701	0.806	0.629
YOLOv10s	0.583	0.784	0.742	0.822	0.631
YOLOv10m	**0.77**	**0.799**	**0.763**	**0.832**	**0.685**
YOLOv10l	1.003	0.812	0.759	0.839	0.69

- Batch size of **64** identified as optimal from insights in **Table 2**

From the insights of Table 3:

- **YOLOv10l:** Best metrics but longer training time
- **YOLOv10m:** Strong performance and 30% faster than YOLOv10l

Need for Improvement:

- Metrics are not up to the mark, prompting the need to modify the dataset

Dataset Enhancement Techniques

Contrast Improvement:

- **CLAHE** enhances crack visibility and robustness to lighting changes

Thresholding:

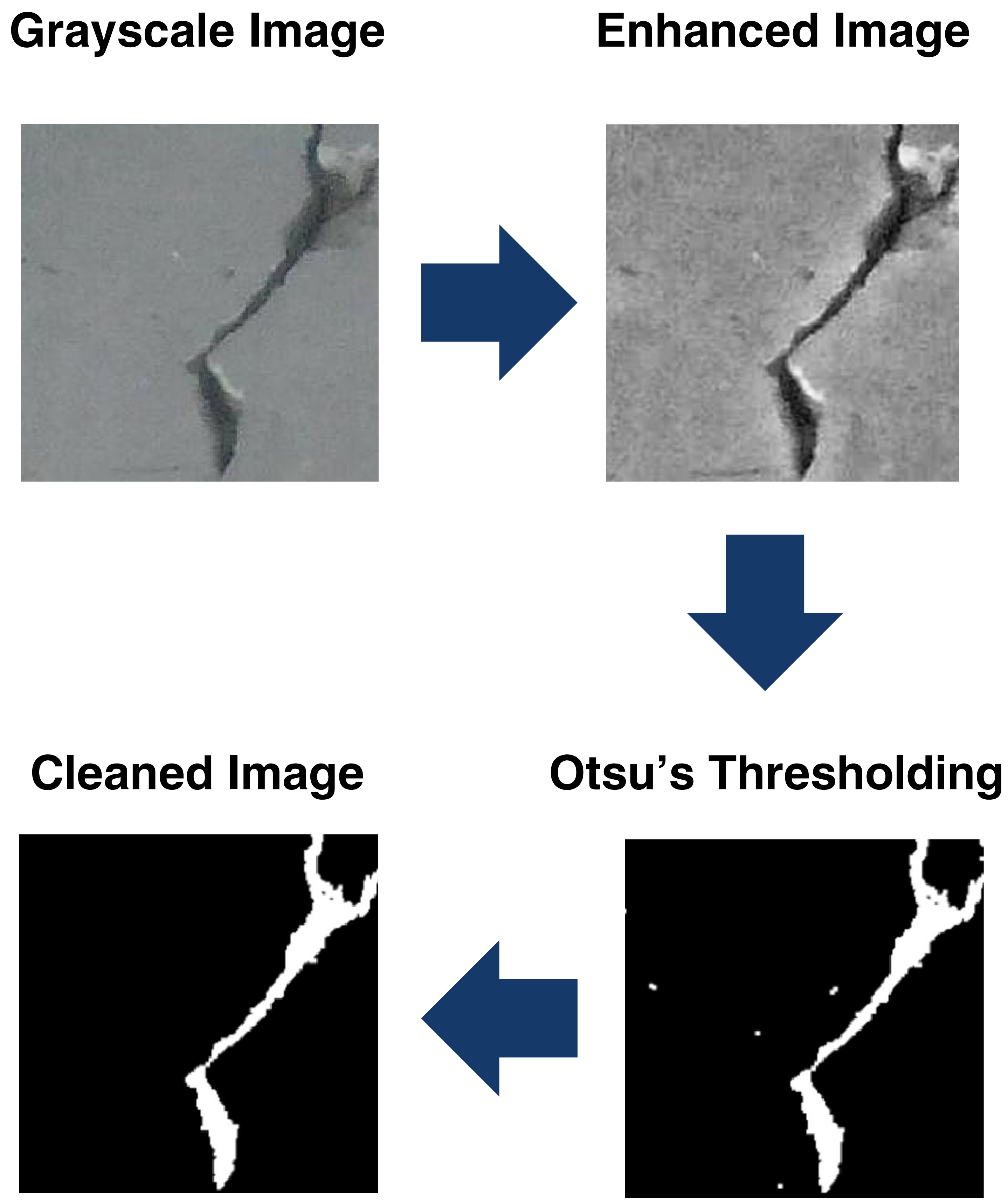
- **Otsu's Thresholding** clearly segments cracks from the background

Image Cleaning:

- **Removes noise**, focusing on significant cracks for better detection

Benefits:

- Cracks are **more visible**
- Improved performance under **varying lighting**
- Cleaner data may **reduce training time**
- Enhancements **boost** crack detection **performance**



- Next, we'll find the best YOLOv10 detection model by optimizing Background Image % and Batch Size, using the same approach as with unenhanced images

Table 4 (50 epochs)

BG Image %	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
1	0.259	0.769	0.718	0.837	0.652
2	0.26	0.792	0.756	0.856	0.659
3	0.269	0.779	0.728	0.837	0.661
4	**0.28**	**0.78**	**0.775**	**0.864**	**0.676**
5	0.282	0.755	0.755	0.841	0.655
6	0.286	0.792	0.785	0.863	0.648
7	0.284	0.767	0.755	0.837	0.619
8	0.288	0.852	0.727	0.869	0.678
9	0.288	0.659	0.728	0.755	0.562
10	0.311	0.782	0.74	0.84	0.645

Table 5 (50 epochs)

Batch Size	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
64	**0.28**	**0.78**	**0.775**	**0.864**	**0.676**
32	0.34	0.766	0.777	0.865	0.666
16	0.495	0.723	0.775	0.834	0.641

Table 6 (100 epochs)

Model	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
YOLOv10n	0.618	0.822	0.741	0.861	0.669
YOLOv10s	0.613	0.798	0.764	0.847	0.664
YOLOv10m	0.753	0.84	0.779	0.883	0.731
YOLOv10l	**1.014**	**0.821**	**0.808**	**0.891**	**0.737**
YOLOv10x	1.356	0.844	0.8	0.883	0.74

Insights Summary

From Table 4:

- **YOLOv10n** trained with various Background Image % at **batch size 64** for **50 epochs**
- Set at **4%** for best performance and training time
- Added **112 uncracked** images, totaling **2,912** images

From Table 5:

- **Batch size** of **64** identified as optimal (50 epochs)

From Table 6:

- The **best model** obtained is **YOLOv10l**, achieving strong performance metrics: Precision: 82.1% Recall: 80.8%, **mAP@50: 89.1%,** and mAP@50-95: 73.7%

Strategies to Enhance Model Performance

1. **Train the current best model for additional epochs**
2. **Increase the dataset size**
3. **Consider using the newly released YOLOv11** models for improved speed and performance, featuring better architecture and training optimizations that enhance feature extraction and detection accuracy

STRATEGY EVALUATION

1. Train the current best model for additional epochs

Table 7

Epochs	Time (Hrs)	Precision	Recall	mAP@50	mAP@50-95
50	0.28	0.78	0.775	0.864	0.676
100	0.618	0.822	0.741	0.861	0.669
200	1.205	0.783	0.775	0.853	0.661

From the insights of Table 7:

- **YOLOv10n** trained with **4%** Background Image at **batch size 64** for **various epochs**
- **Recall** improved with more epochs, but **precision** and **mAP** declined, likely due to **overfitting**

2.Increase the dataset size

Table 8

Metric	2800 Images	3600 Images	Change
Time (Hrs)	0.281	0.345	+23%
Precision	0.792	0.771	-2.7%
Recall	0.756	0.778	+2.9%
mAP@50	0.856	0.846	-1.2%
mAP@50-95	0.659	0.656	-0.5%

From the insights of Table 8:

- **Time** increased by **23%** with more data
- **Recall** improved by **2.9%,** showing higher sensitivity to cracked regions
- **Precision** and **mAP** dropped slightly (around **2-3%**), likely due to noise introduced by augmented images affecting clarity in crack detection

3.Try YOLOv11 Model

Table 9 (100 epochs)

Model	Time (Hrs)	Precision	Recall	mAP@50	mAP50-95
M	0.778	0.851	0.781	0.901	0.747
L	0.827	0.824	0.835	0.909	0.754

Table 10 (100 epochs**)**

Model	Time (hrs)	Precision	Recall	mAP50	mAP50-95
YOLOv10L	1.014	0.821	0.808	0.891	0.737
YOLOv11L	0.827	0.824	0.835	0.909	0.754

- **YOLOv11L** is chosen for its optimal time-performance tradeoff, as indicated in **Table 9**

From the insights of Table 10:

- **YOLOv11L** shows improvements in **recall (+2.7%)** and **mAP@50 (+1.8%)** over **YOLOv10L**
- It maintains **similar precision** and **mAP@50-95** while offering a slightly **faster training time**

Table 11

Epochs	Time (hrs)	Precision	Recall	mAP@50	mAP@50-95
120	1.016	0.829	0.828	0.915	0.764
130	1.107	0.803	0.841	0.909	0.762
150	**1.248**	**0.842**	**0.825**	**0.916**	**0.766**

Plan Summary:

- Out of the **three strategies** training the current best model for additional epochs, increasing the data size, and trying the YOLOv11 model
- **Strategy 3 (YOLOv11) yielded the best results**

Best Model:

YOLOv11L (150 Epochs)

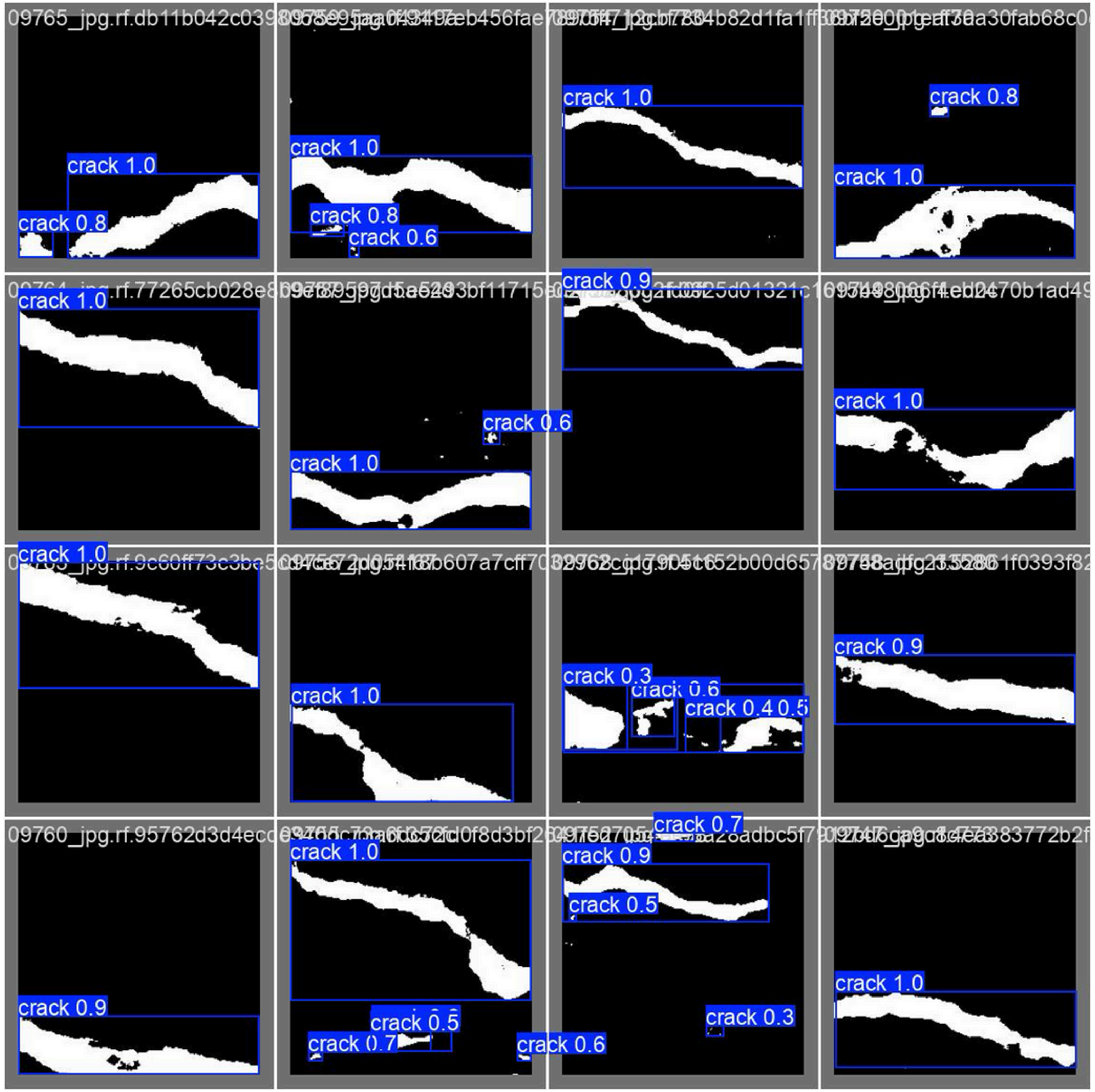
Precision: 84%

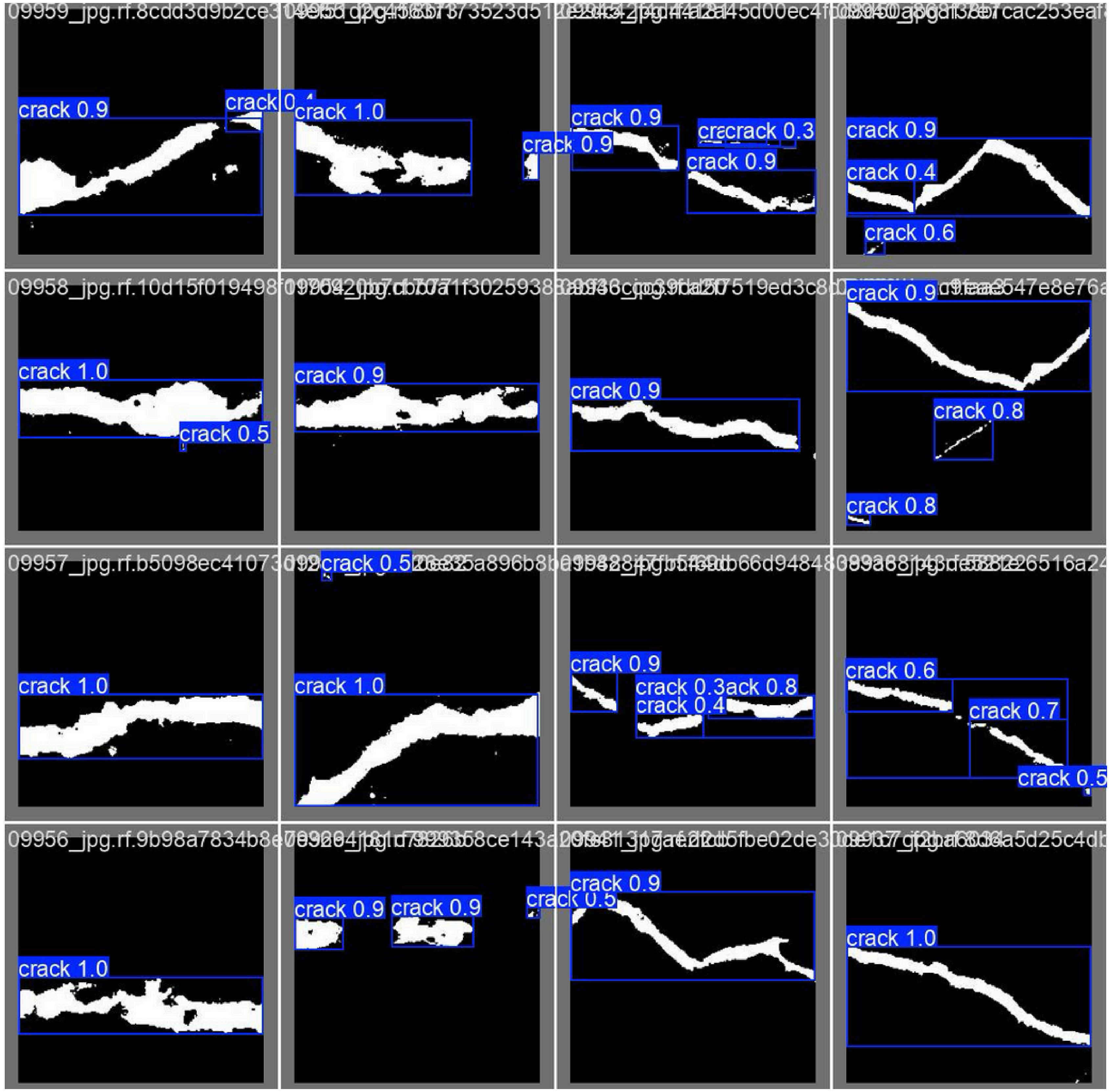
Recall: 82%

mAP@50: 92%

mAP@50-95: 77%

CRACK DETECTION RESULTS





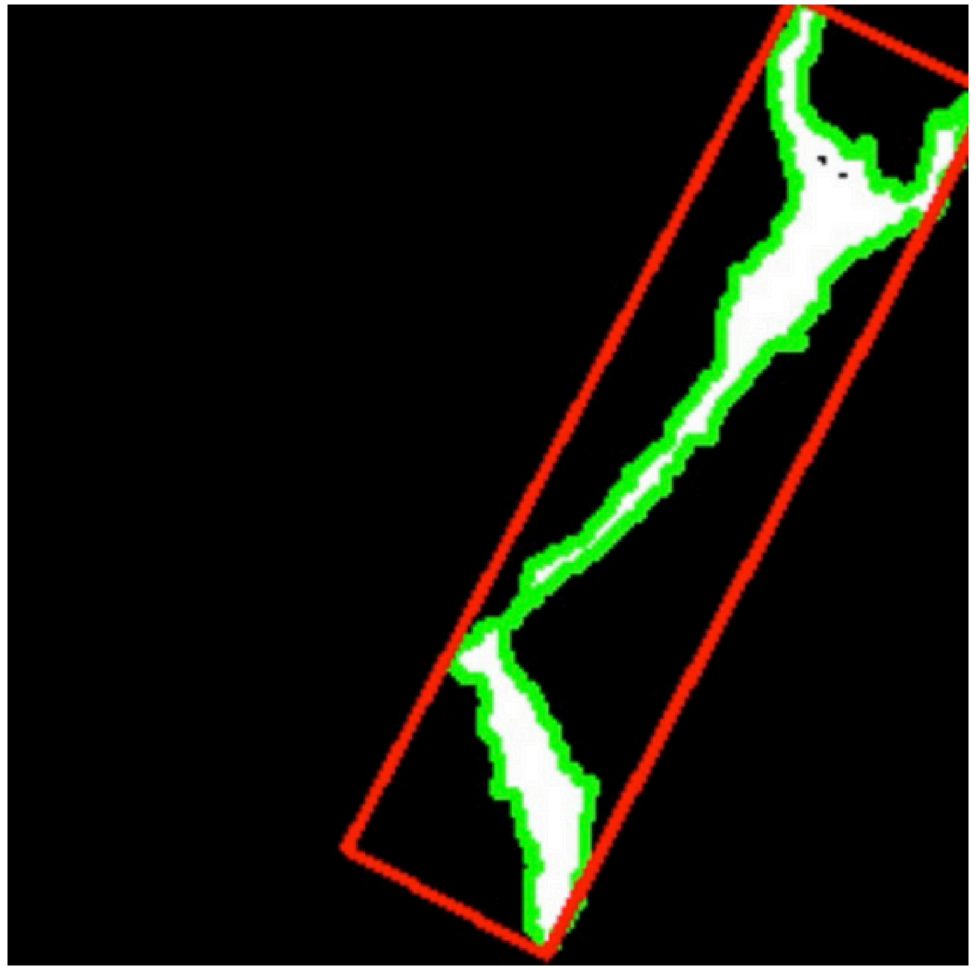
CRACK QUANTIFICATION

Metrics: Area, perimeter, length, and width are derived from analyzing cleaned binary crack images.

- **Area:** No. of white pixels within the crack, representing its total size
- **Perimeter:** Boundary pixels outlining the crack.
- **Length & Width:** Major and minor axes of the bounding rectangle around the crack

In the figure

- **White Pixels:** Represent the crack region
- **Green Boundary:** Outlines the crack's contour
- **Red Bounding Box:** Indicates the crack's orientation & dimensions



Quantified Measurements:

- Crack Area: **2948.50 px²**
- Crack Perimeter: **707.03 px**
- Crack Length: **225.89 px**
- Crack Width: **53.33 px**

DISCUSSION AND CONCLUSION

- The project automates crack detection and measurement, cutting down the time spent on manual inspections and making monitoring more efficient
- YOLOv11 provides precise crack detection and measurement, helping to assess crack severity and prioritize repairs
- The solution can be scaled for large infrastructure projects, offering real-time crack detection over large areas

FUTURE WORK

- Future work includes using **drones** to capture images of the entire structure for automated crack detection and quantification
- The software will **sort cracks** based on **severity** to **prioritize repairs** and **minimize costs**
- A key challenge is ensuring **consistent image capture distance** for accurate real-world measurements, preventing false analysis

THANK YOU