Container Damage Detection

- Zeliha Erim
- Fatih Sukas
- Osama Al Moselli
- Niyazi Mehmet Cansever

Content

- Project Description
- Literature Review
- Dataset
- Edge Detection
- **FFT Edge Detection()**
- **Method 1 (Machine Learning Models** ( ))
- Method 2 (Neural Network, Random Forest and Feature Selection)
- Conclusion

Project Description

Maritime transportation is the backbone of global trade, with most international cargo carried by sea. In this context, the physical condition of containers plays a crucial role in ensuring cargo safety, operational efficiency, and the proper handling of commercial and insurance responsibilities.

At port terminals, container damage inspections are traditionally performed during gate-in and gate-out operations through manual visual checks. While widely adopted, this approach is highly dependent on human judgment and is vulnerable to limitations such as high traffic volume, time constraints, and environmental conditions. These factors increase the risk of inconsistent evaluations and undetected damage in large-scale terminals.

Recent advances in artificial intelligence, particularly in computer vision and deep learning, offer significant potential to automate container damage detection. AI-based systems can provide faster, more objective, and consistent inspections, improving operational reliability and supporting data-driven decision-making in modern port operations.

Literature Review

1. Manual Inspection (The Current Industry Standard)

- Method: Visual assessment by human operators.
- Performance: Research indicates human accuracy typically fluctuates between 70-80%.
- Drawbacks: It is slow, costly, and highly dependent on operator fatigue and environmental lighting, leading to inconsistent grading.

2. Deep Learning Approaches

- Li et al. (2018): Utilized Faster R-CNN for metallic surface defect detection.
  - Result: Achieved 92.3% accuracy.
  - Constraint: Requires massive, pixel-perfect labeled datasets.
- Chen et al. (2020): Applied Multi-Layer Perceptrons (MLP) on extracted feature vectors.
  - Result: Reached ~85-88% accuracy on texture classification.
  - Constraint: These models often struggle to differentiate between deep shadows (ribs) and actual dark defects without extensive training data.

Literature Review

3. Spectral & Classical ML Approaches (The "Filtering" Path)

- Tsai et al. (2012): Proposed using Fast Fourier Transform (FFT) for defect detection in fabrics with periodic patterns.
- Result: Achieved 95% defect detection rates by mathematically suppressing the repeating pattern frequencies.
- Texture Analysis Studies: Research combining Gabor Filters with Random Forest or SVM classifierstypically reports 90-94% accuracy in texture-heavy industrial environments.

4. Framework of This Study

- The literature presents two valid but distinct methodologies for solving the "ribs" problem. Ourproject conducts a comparative analysis of these two schools of thought:
  - Path A : Implementing FFT Spectral Filtering combined with Random Forest to mathematicallyisolate defects.
  - Path B : Implementing Neural Networks (MLP) to learn defects from spatial features.

Dataset

- Dataset is taken from this link: https://universe.roboflow.com/thanh-fscay/container-damage-hmvl7/dataset/1 and taken from Arkas Denizcilik.
- It is binary classification dataset with damaged and not damaged images.

Mean — `mean(x)`

- What does it measure?
  - Average intensity / response strength of the filtered image
- Why is it important?
  - Indicates **overall defect activity**
  - Higher mean → larger or more widespread defect regions
- In container defect context:
  - **Corrosion / paint peeling** → elevated mean
  - **Clean or lightly damaged surfaces** → low mean
- Captures **global defect presence**.

Standard Deviation — `std(x)`

- What does it measure?
  - Variability of pixel responses
- Why is it important?
  - Defects create **non-uniform intensity patterns**
  - High std → irregular, textured, or fragmented defects
- In container defect context:
  - **Corrosion** → high std (rough texture)
  - **Scratches / cracks** → moderate std
  - **Smooth surfaces** → low std
- Measures **surface irregularity**.

95th Percentile — `percentile(x, 95)`

- What does it measure?
  - Strong response level ignoring extreme outliers
- Why is it important?
  - More stable than `max`
  - Robust to noise and single-pixel artifacts
- In container defect context:
  - **Cracks / scratches** → high p95
  - **Peeling edges** → elevated p95
  - Noise spikes → mostly ignored
- Captures **typical strong defect responses**.

Maximum — `max(x)`

- What does it measure?
  - Strongest pixel response in the image
- Why is it important?
  - Detects **presence of severe local defects**
  - Very sensitive to sharp anomalies
- In container defect context:
  - **Deep cracks**
  - **Holes**
  - **Sharp peeling boundaries**
- Signals **worst-case defect severity**.

Why these four work best together

Statistic	What it captures
Mean	Overall defect extent
Std	Texture & irregularity
95th percentile	Robust strong responses
Max	Extreme anomalies

Together they describe:

- **How much defect exists**
- **How uneven it is**
- **How strong it typically gets**
- **How severe it can be**

Spatial Edge Detection

Objective:

To detect sudden changes in image intensity that indicate physical boundaries, such as cracks, holes, or the edges of a dent.

Techniques Implemented:

- Sobel & Prewitt Filters (X & Y):
  - Calculated first-order derivatives to detect horizontal and vertical edges.
  - Role: capturing the general structure and orientation of surface anomalies.
- Laplacian Filter:
  - Calculated the second-order derivative.
  - Role: optimized for detecting "blobs" and rapid intensity peaks, making it effective for identifying distinct dents or holes.
- Canny Edge Detector:
  - A multi-stage algorithm using noise reduction and hysteresis thresholding.
  - Role: providing binary (black/white) maps of the sharpest, most significant edges.

Observation:

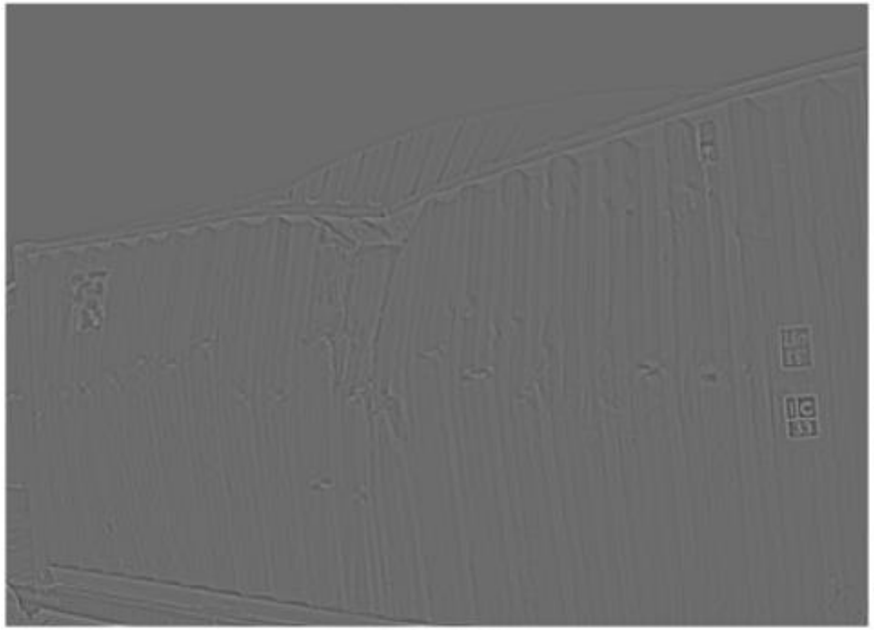
While effective at finding all edges, these filters struggled to distinguish between damage edges and the natural edges of the container ribs, creating high noise levels in the feature data.

Edge Detection

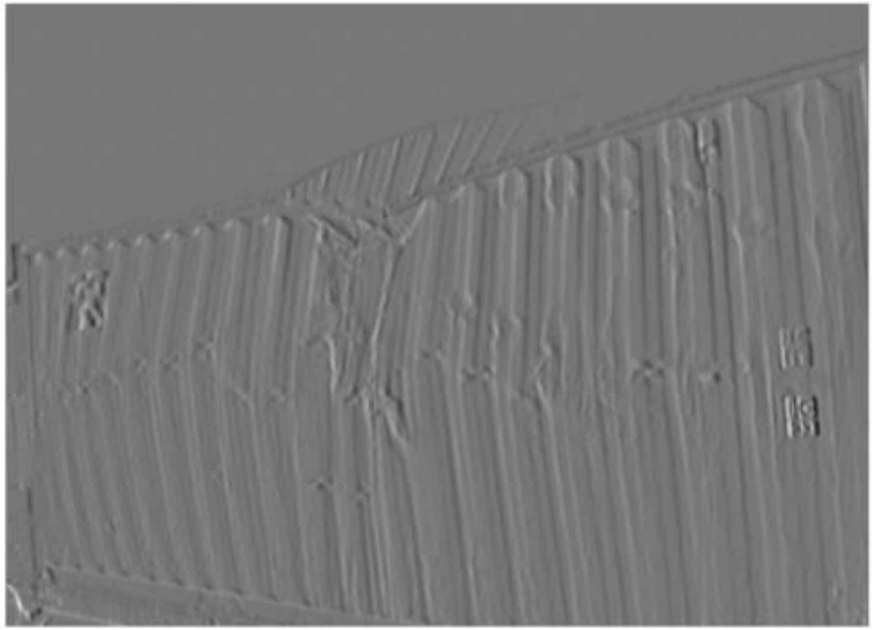
- 16 fundamental Gabor filters are employed to extract multi-orientation and multi-frequency texture features from the image.
- **Sinusoidal FFT Filter:**
  This filter operates in the frequency domain to enhance or suppress periodic and sinusoidal texture components across the entire image.
- **Local Sinusoidal FFT Filter:**
  This filter performs localized frequency analysis to capture regional texture variations and spatially varying patterns.
- **Sinusoidal FFT Filter + Sobel + Laplacian:**
  This combined approach integrates frequency-based texture information with spatial-domain edge detection to enhance both structural and boundary features.
- **Local Sinusoidal FFT Filter + Sobel + Laplacian (with Thresholding and Mask Operations):**
  This method applies localized frequency and edge analysis with thresholding and masking to isolate relevant features while reducing noise and irrelevant details.

Edge Detection

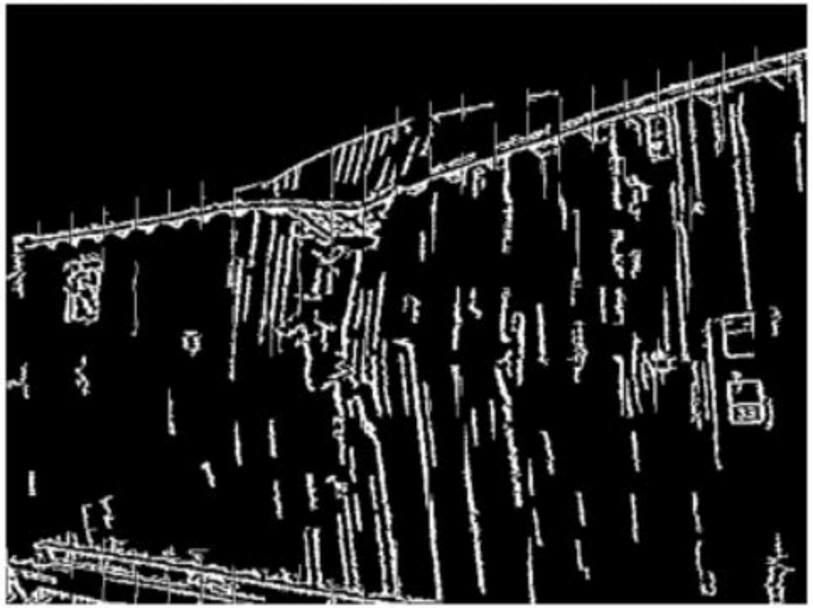
Laplacian



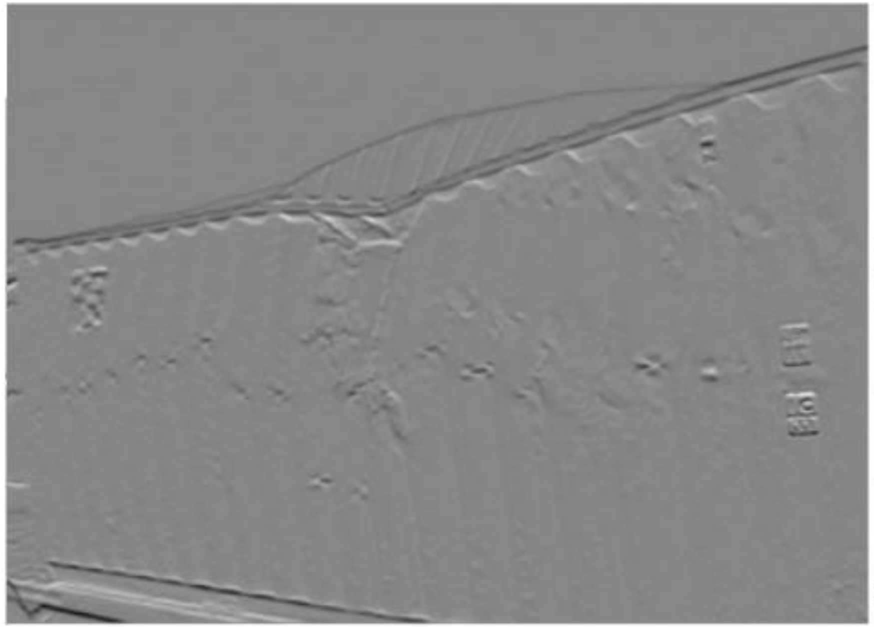
Sobel X



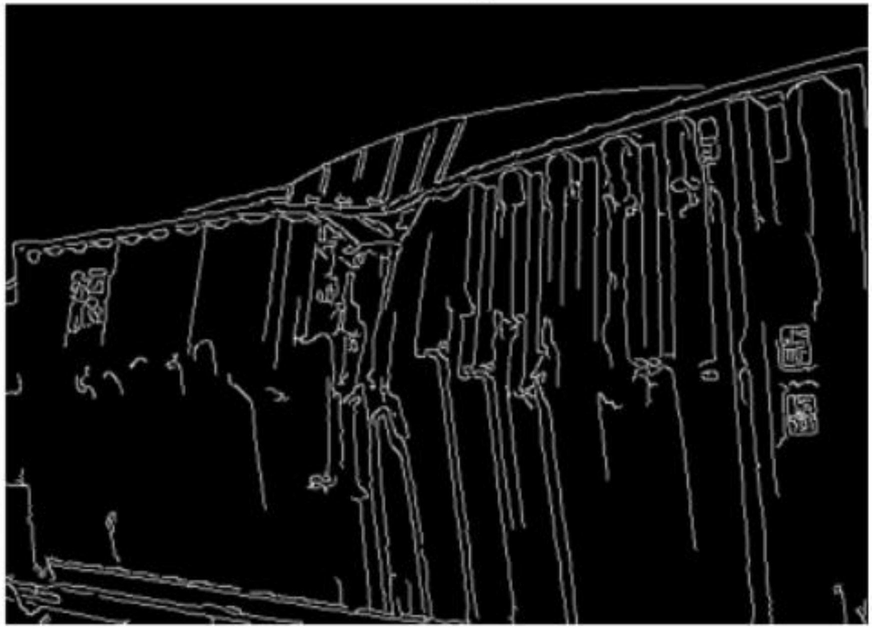
Residual Laplacian with Harmonics Masked



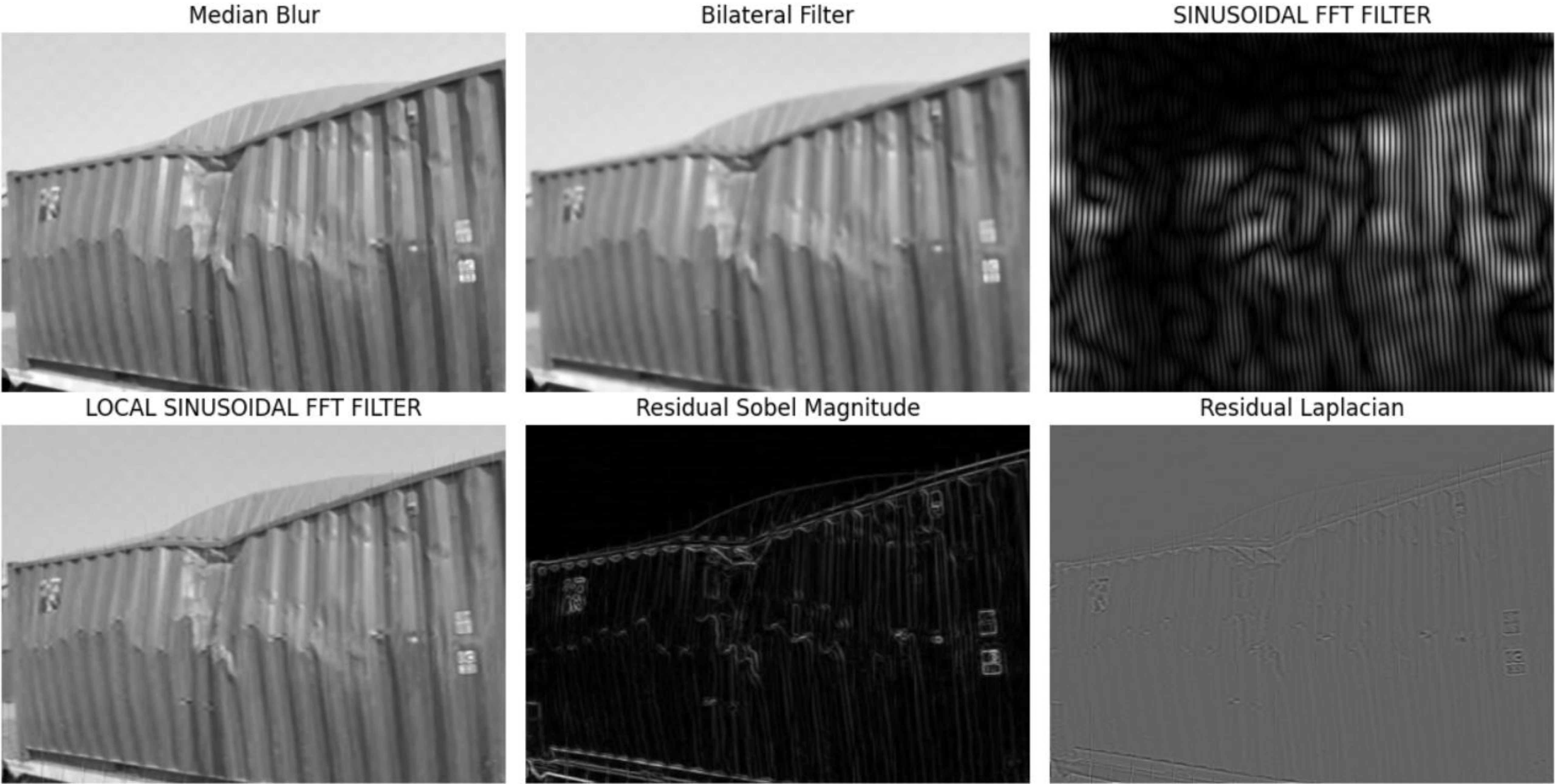
Sobel Y



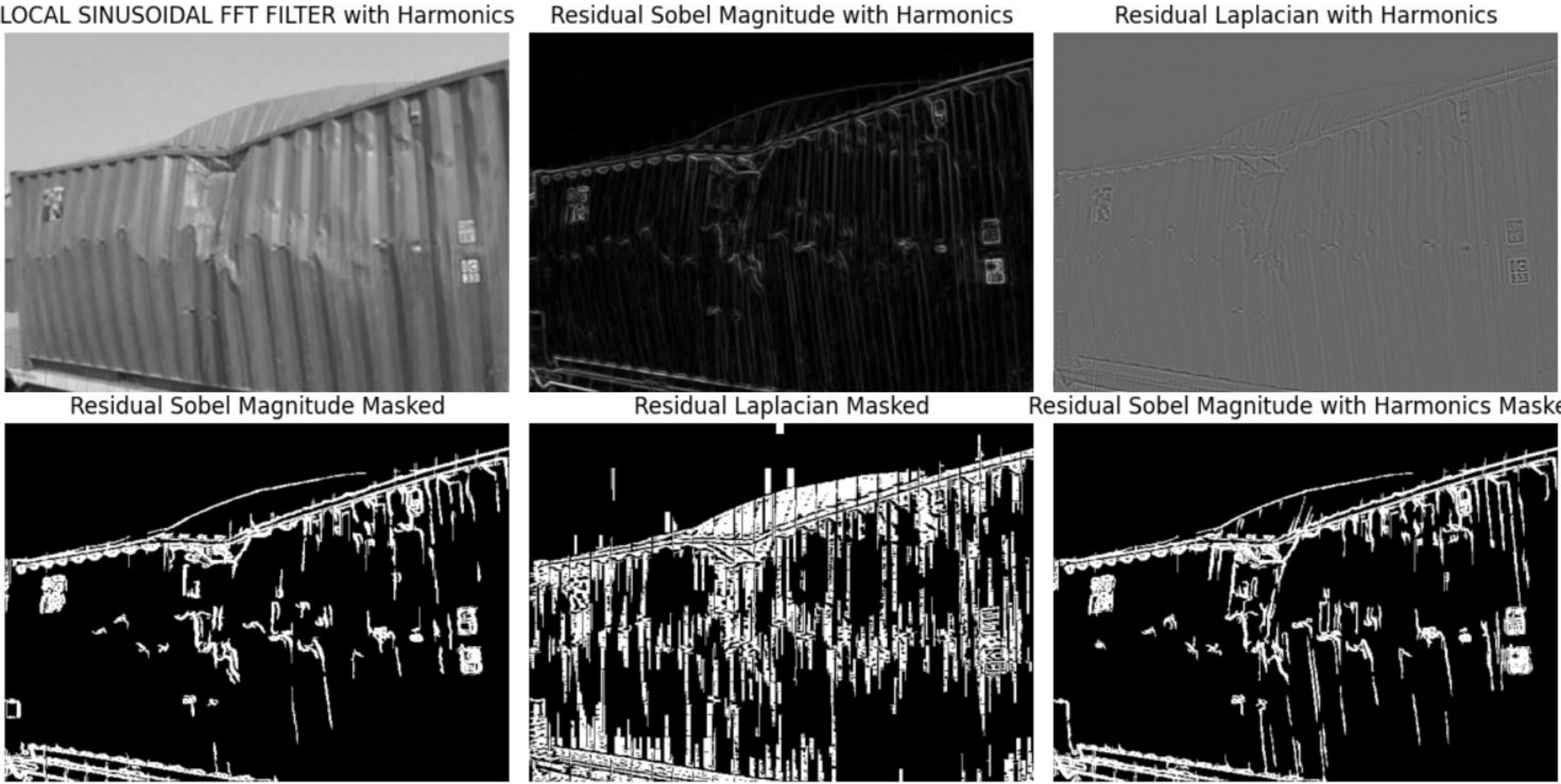
Canny



Edge Detection



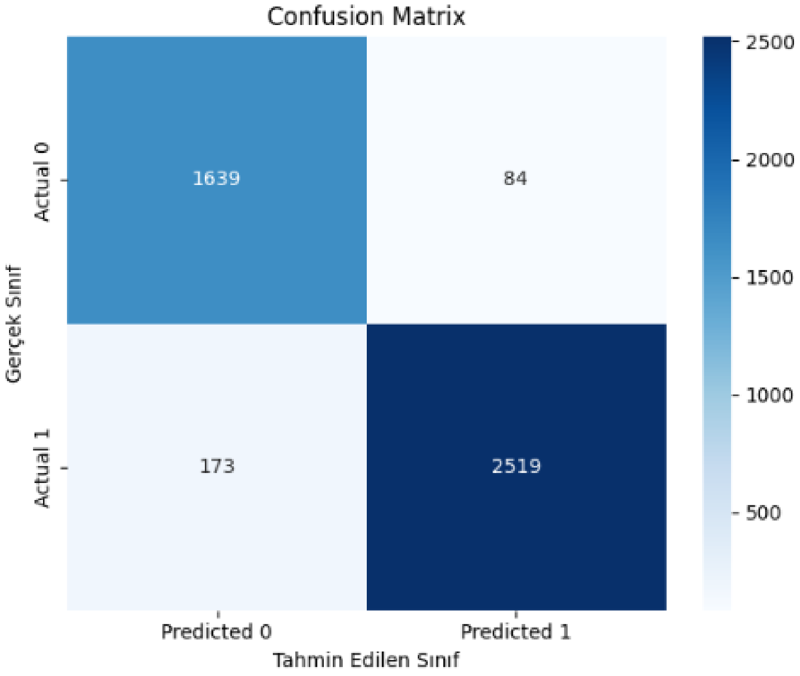
Edge Detection



Method 2

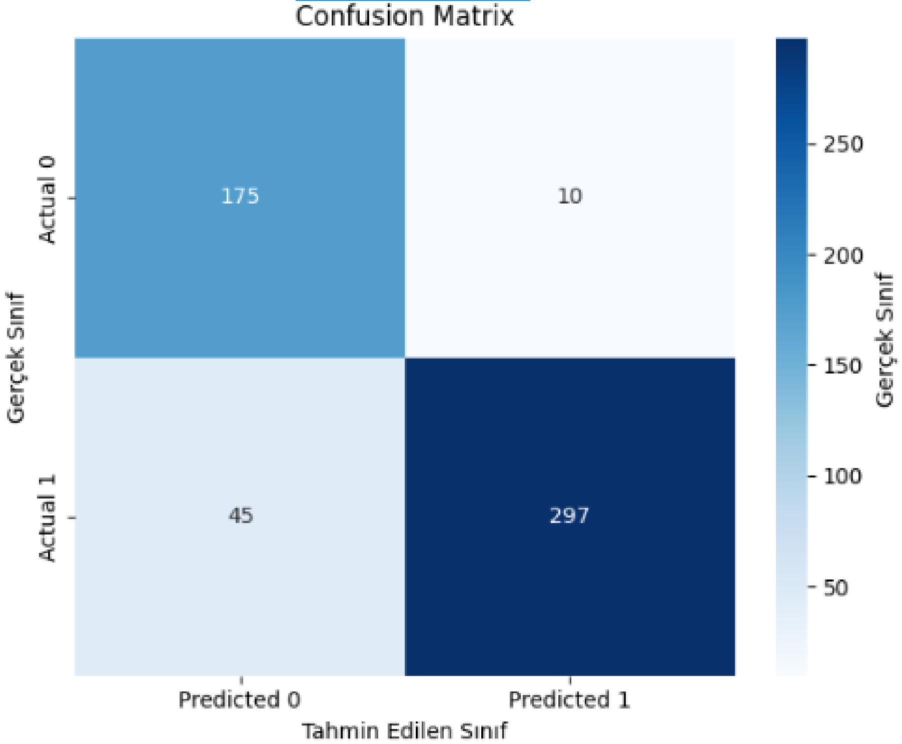
- Neural Network: The model is a Sequential fully connected neural network with 64, 32, and 16 neurons in the hidden layers, incorporating dropout for regularization, optimized using the Adam optimizer, and trained with binary cross-entropy loss.

Train Result



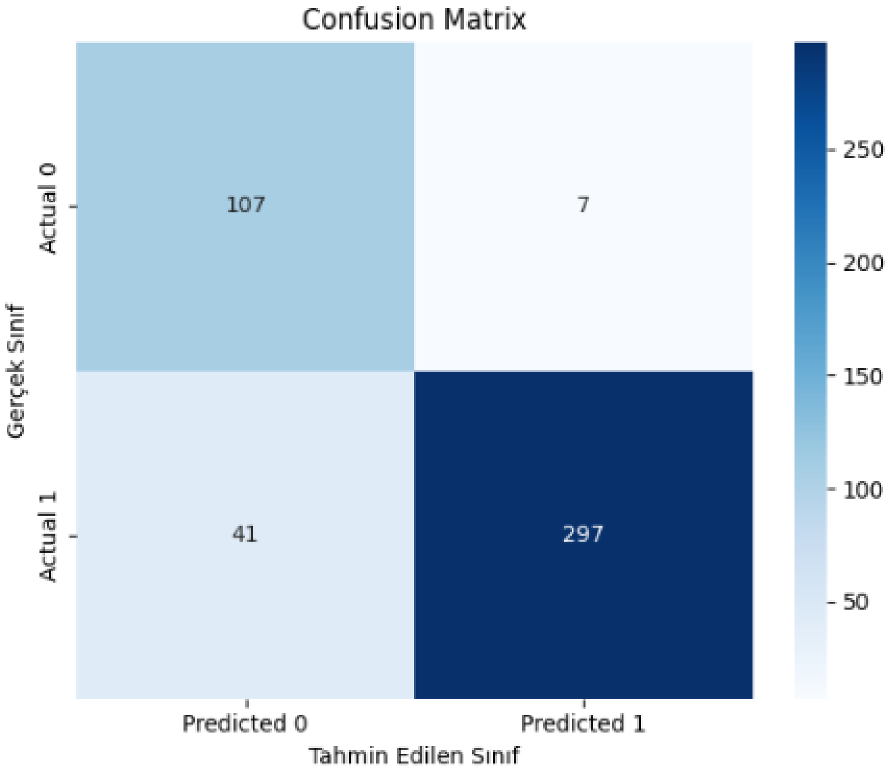
	precision	recall	f1-score	support
0	0.90	0.95	0.93	1723
1	0.97	0.94	0.95	2692
accuracy			0.94	4415
macro avg	0.94	0.94	0.94	4415
weighted avg	0.94	0.94	0.94	4415

Validation Result



	precision	recall	f1-score	support
0	0.80	0.95	0.86	185
1	0.97	0.87	0.92	342
accuracy			0.90	527
macro avg	0.88	0.91	0.89	527
weighted avg	0.91	0.90	0.90	527

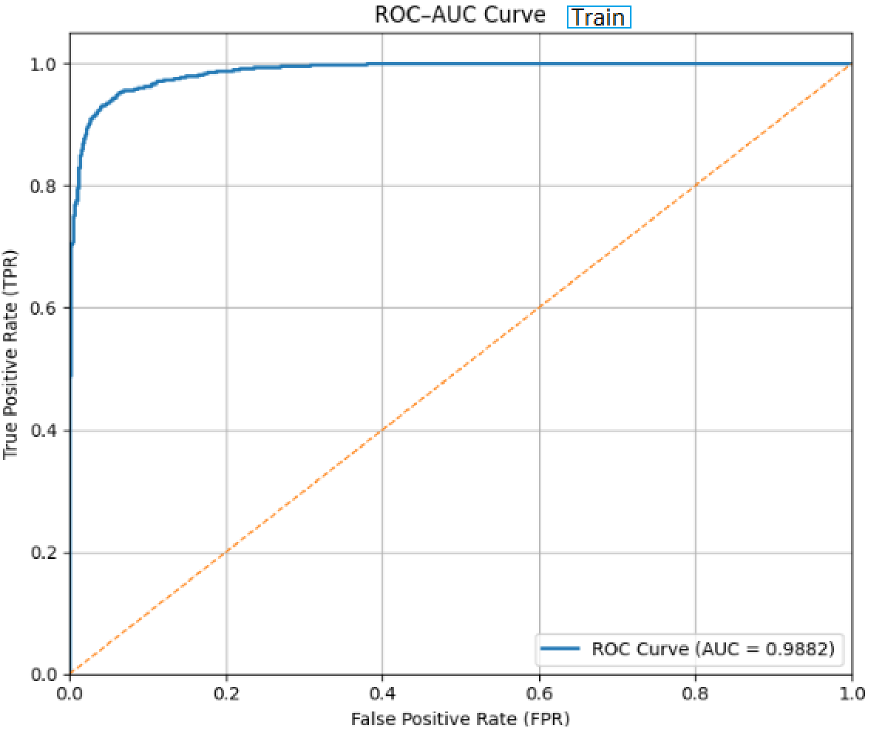
Test Result

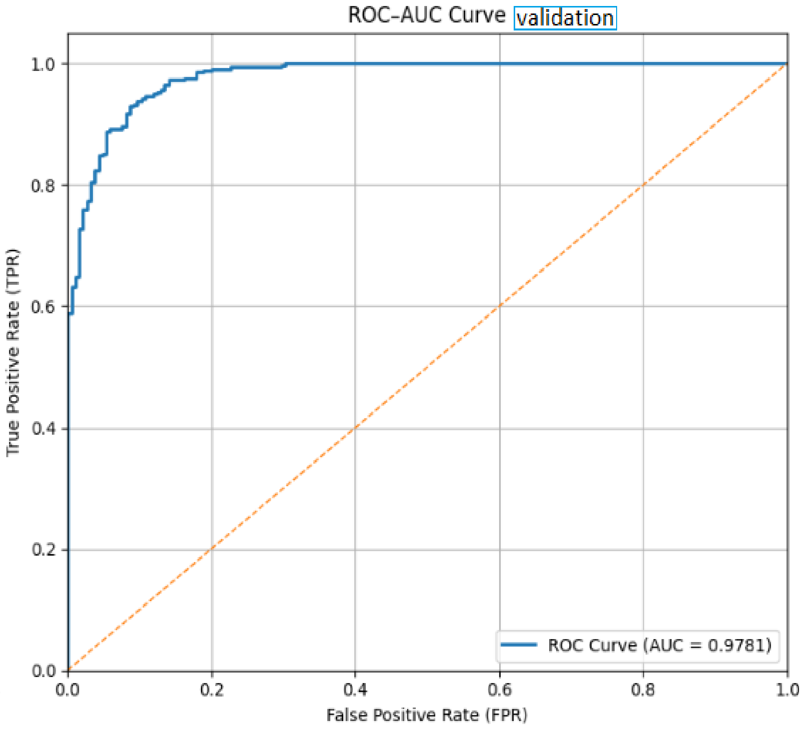


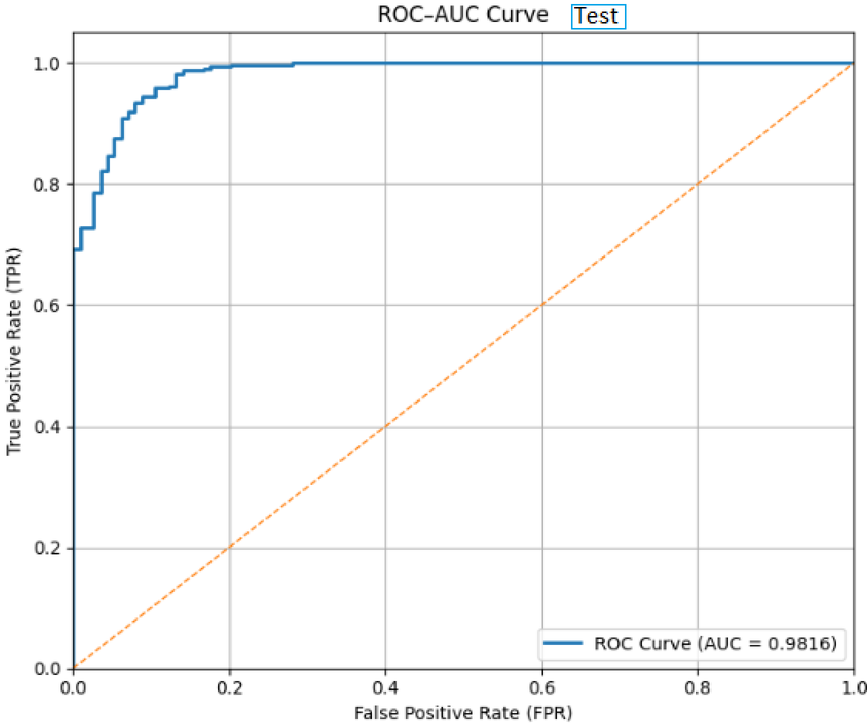
	precision	recall	f1-score	support
0	0.72	0.94	0.82	114
1	0.98	0.88	0.93	338
accuracy			0.89	452
macro avg	0.85	0.91	0.87	452
weighted avg	0.91	0.89	0.90	452

Method 2

```
138/138 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Train Accuracy: 0.9418
Validation Accuracy: 0.8845
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Test Accuracy: 0.8938

Train Accuracy      : 0.9418
Validation Accuracy : 0.8845
Test Accuracy       : 0.8938
```



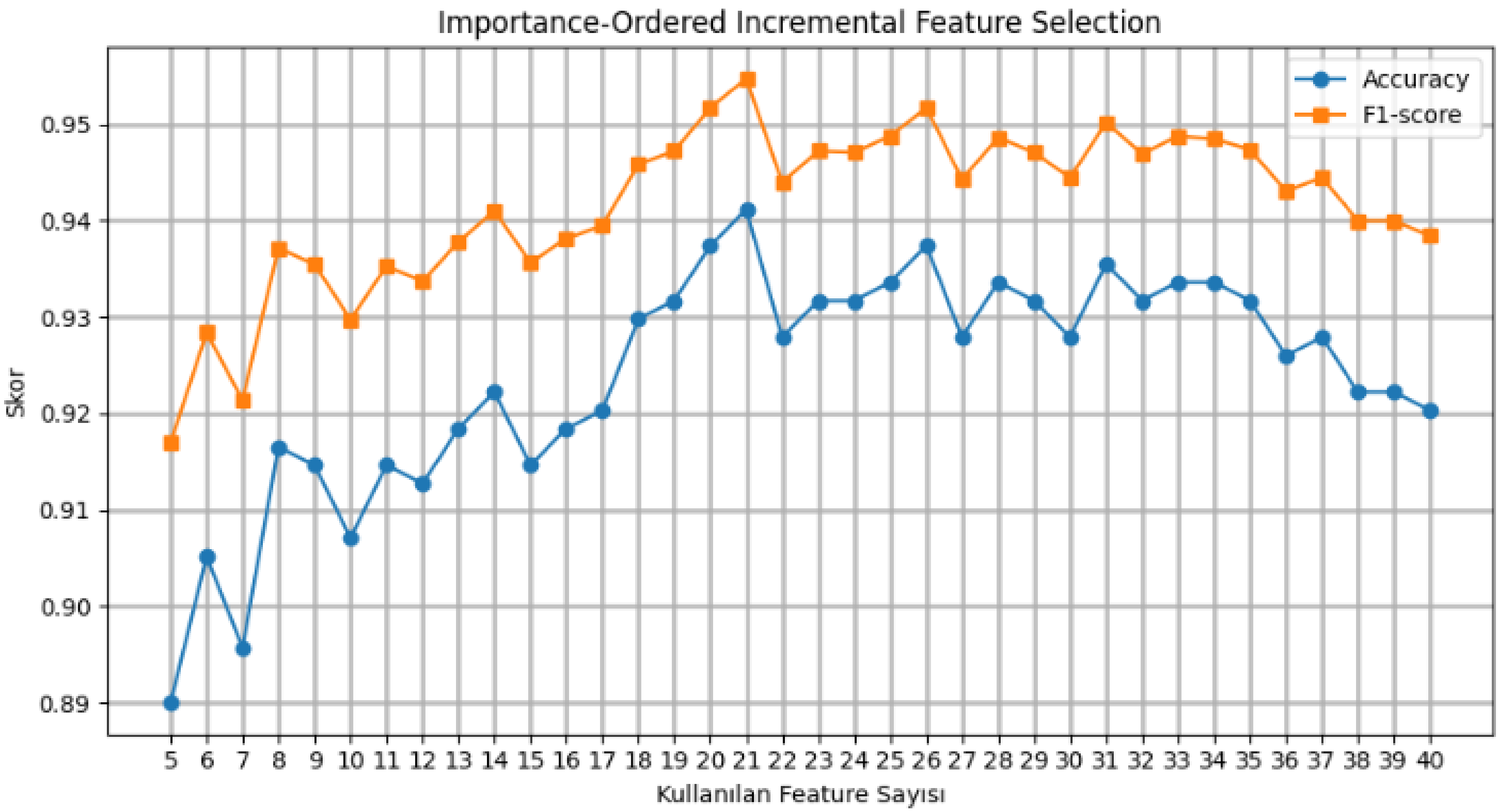




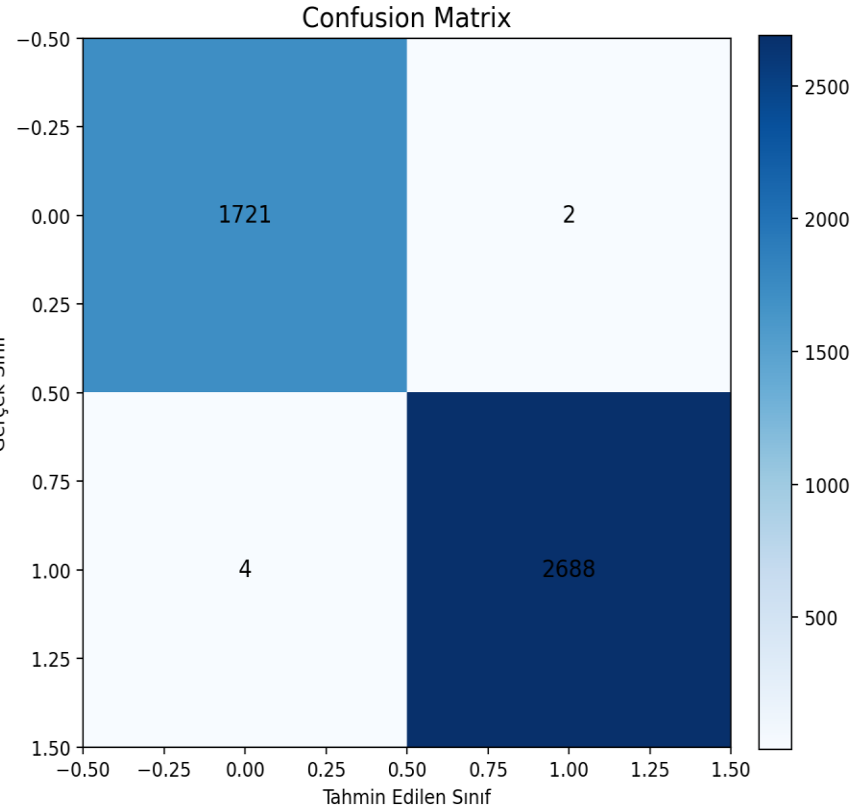
Method 2

- Feature importance and feature selection with Random-Forest algorithm.

```
EN İYİ MODEL:
Feature Sayısı : 21
Accuracy       : 0.9412
F1-score       : 0.9547

TRAIN SET PERFORMANCE:
Accuracy: 1.0
F1-score: 1.0

VALID SET PERFORMANCE:
Accuracy: 0.9411764705882353
F1-score: 0.9547445255474453

TEST SET PERFORMANCE:
Accuracy: 0.9402654867256637
F1-score: 0.9598811292719168
```

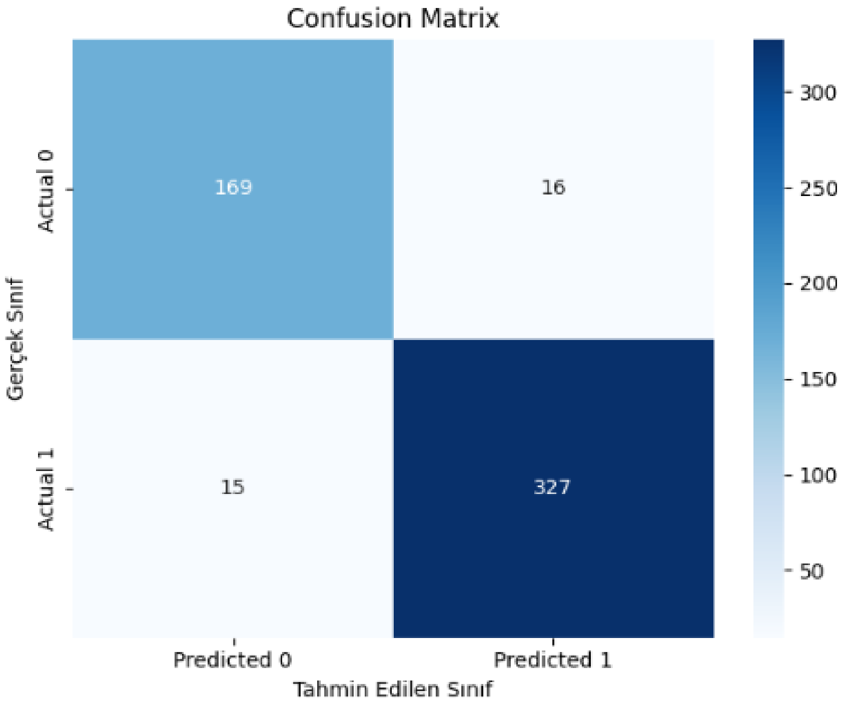


Method 2



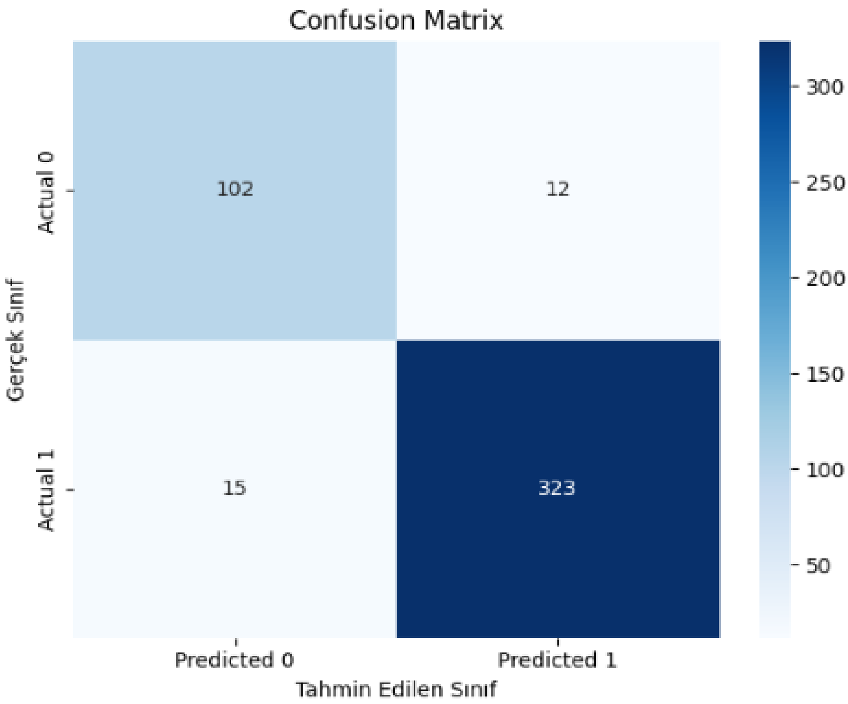
	precision	recall	f1-score	support
0	0.9977	0.9988	0.9983	1723
1	0.9993	0.9985	0.9989	2692
accuracy			0.9986	4415
macro avg	0.9985	0.9987	0.9986	4415
weighted avg	0.9986	0.9986	0.9986	4415

```
Validation Confusion Matrix
[[169  16]
 [ 15 327]]
```



	precision	recall	f1-score	support
0	0.92	0.91	0.92	185
1	0.95	0.96	0.95	342
accuracy			0.94	527
macro avg	0.94	0.93	0.94	527
weighted avg	0.94	0.94	0.94	527

```
Test Confusion Matrix
[[102  12]
 [ 15 323]]
```



	precision	recall	f1-score	support
0	0.87	0.89	0.88	114
1	0.96	0.96	0.96	338
accuracy			0.94	452
macro avg	0.92	0.93	0.92	452
weighted avg	0.94	0.94	0.94	452

Method 2

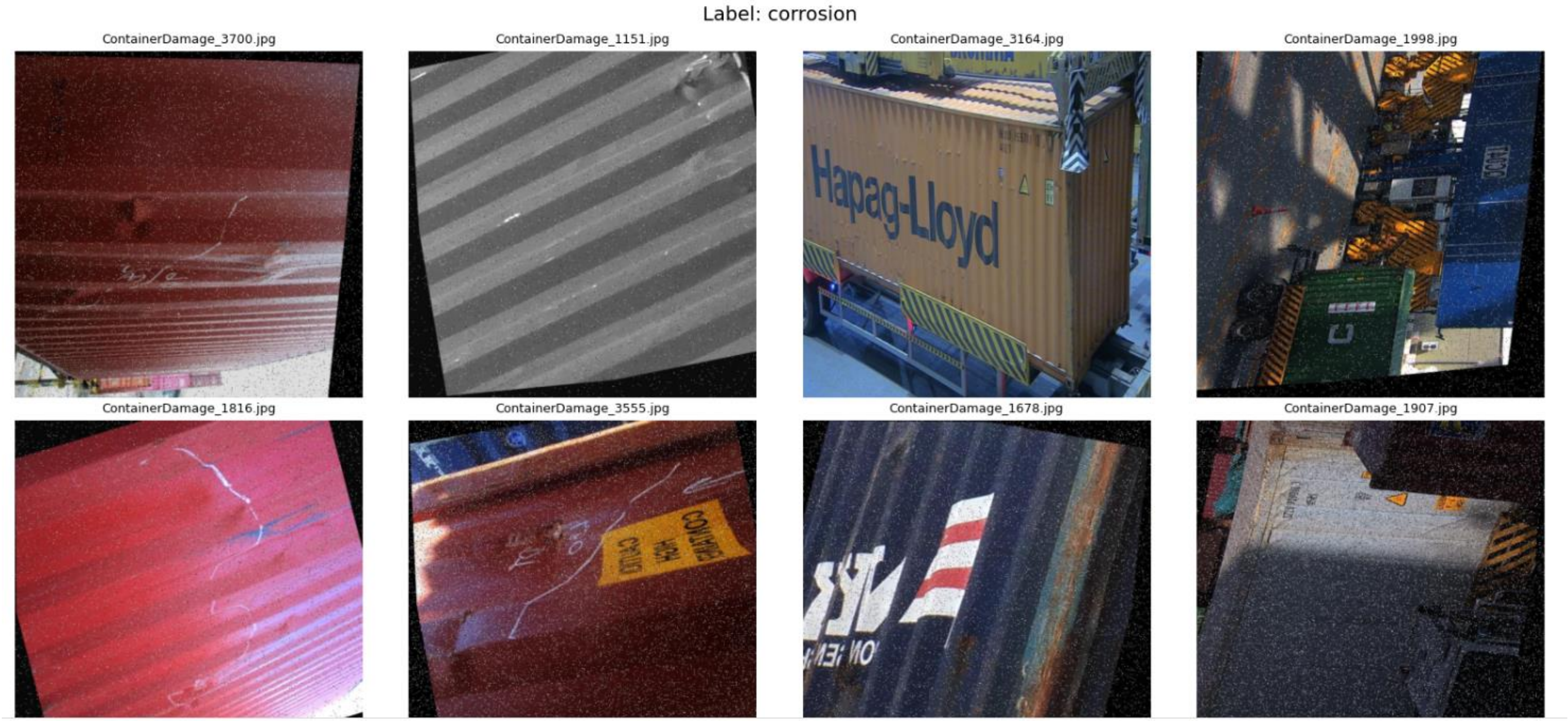
10-Fold Cross-Validation:

- **Objective** :
  - To ensure the NN model's performance is stable and not dependent on a specific, lucky train-test split.
- **Methodology**:
  - Technique: Stratified 10-Fold Cross-Validation.
  - Model Used: Multi-Layer Perceptron (Neural Network) baseline.
  - Dataset: Training set divided into 10 distinct subsets (folds).
- **Results**
  - Mean Accuracy: 89.15% (±± 0.95%)
  - Mean ROC-AUC: 0.9568 (±± 0.0082)
- **Key Takeaway**
  - The low standard deviation (<1%) proves the feature set is robust.
  - The high AUC score (>0.95) confirms excellent separability between "Damaged" and "Healthy" classes, regardless of the data subset.

Flaws - Corossion

feature	importance_mean
fft_residual_local_output_norm_with_harmonics_std	0.035253
fft_residual_local_output_norm_std	0.033040
residual_sobel_mag_masked_num_components	0.030200
residual_laplacian_with_harmonics_std	0.024474
residual_sobel_mag_with_harmonics_masked_num_c...	0.024300
residual_sobel_mag_with_harmonics_mean	0.021365
residual_laplacian_with_harmonics_masked_area_...	0.019360
residual_laplacian_with_harmonics_masked_max_area	0.018556
residual_sobel_mag_mean	0.016508
residual_sobel_mag_with_harmonics_p95	0.015751
residual_laplacian_with_harmonics_p95	0.014982
residual_laplacian_masked_max_area	0.014924
residual_laplacian_with_harmonics_mean	0.014189
residual_laplacian_std	0.013180
residual_laplacian_masked_area_ratio	0.012911

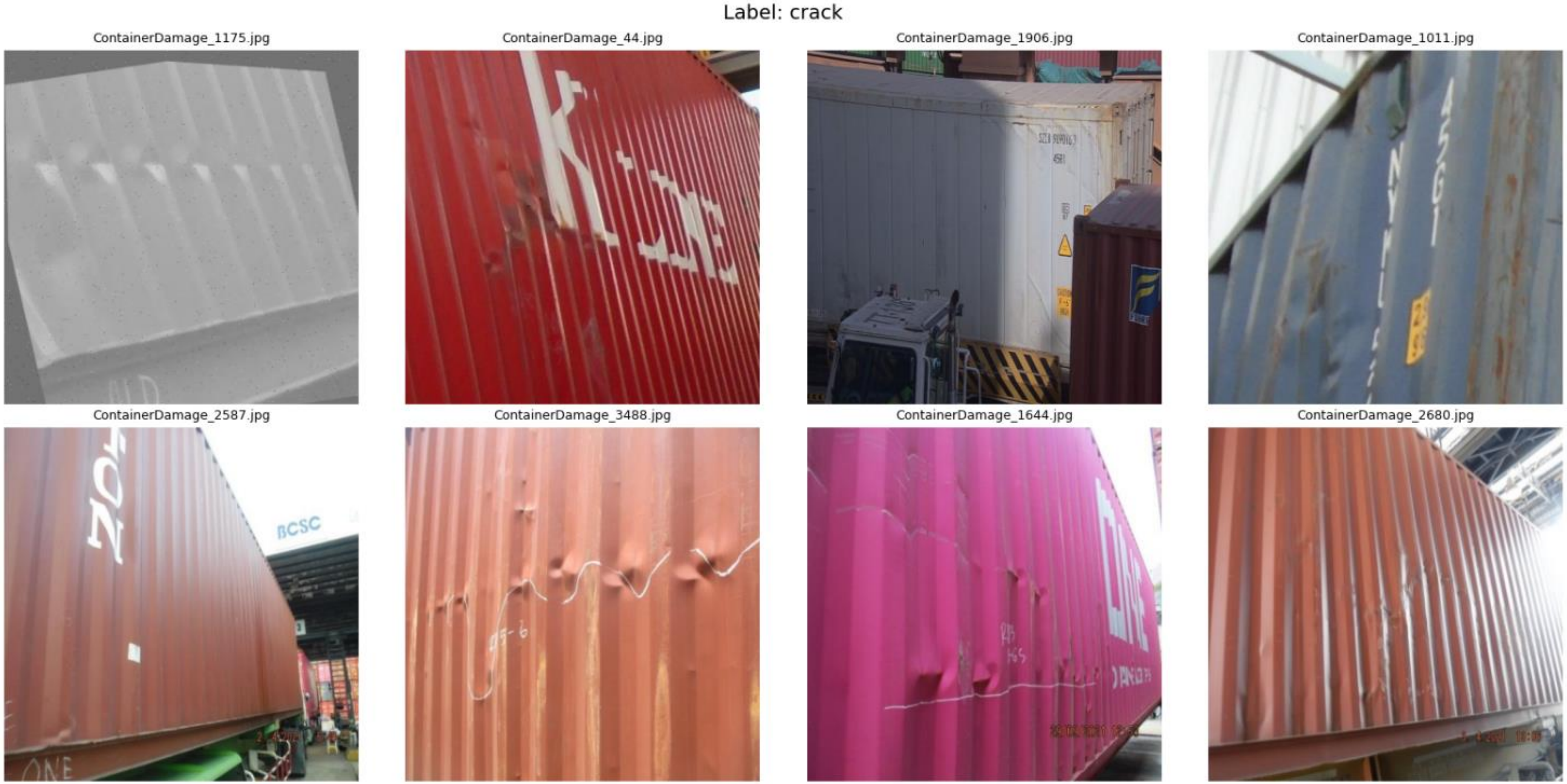
Flaws - Corossion



Flaws - Crack

	feature	importance_mean
	fft_residual_local_output_norm_with_harmonics_std	0.048897
	fft_residual_local_output_norm_std	0.046138
	residual_sobel_mag_with_harmonics_masked_area_...	0.031263
	residual_sobel_mag_masked_area_ratio	0.029139
	residual_laplacian_with_harmonics_std	0.025375
	residual_sobel_mag_with_harmonics_masked_num_c...	0.023088
	residual_laplacian_masked_mean_aspect	0.021157
	residual_laplacian_with_harmonics_masked_area_...	0.019019
	residual_laplacian_with_harmonics_mean	0.017810
	residual_sobel_mag_masked_mean_aspect	0.017227
	residual_sobel_mag_with_harmonics_std	0.016491
	residual_sobel_mag_masked_num_components	0.015970
	residual_laplacian_with_harmonics_p95	0.014687
	fft_residual_local_output_norm_with_harmonics_p95	0.014401
	residual_laplacian_with_harmonics_masked_num_c...	0.014342

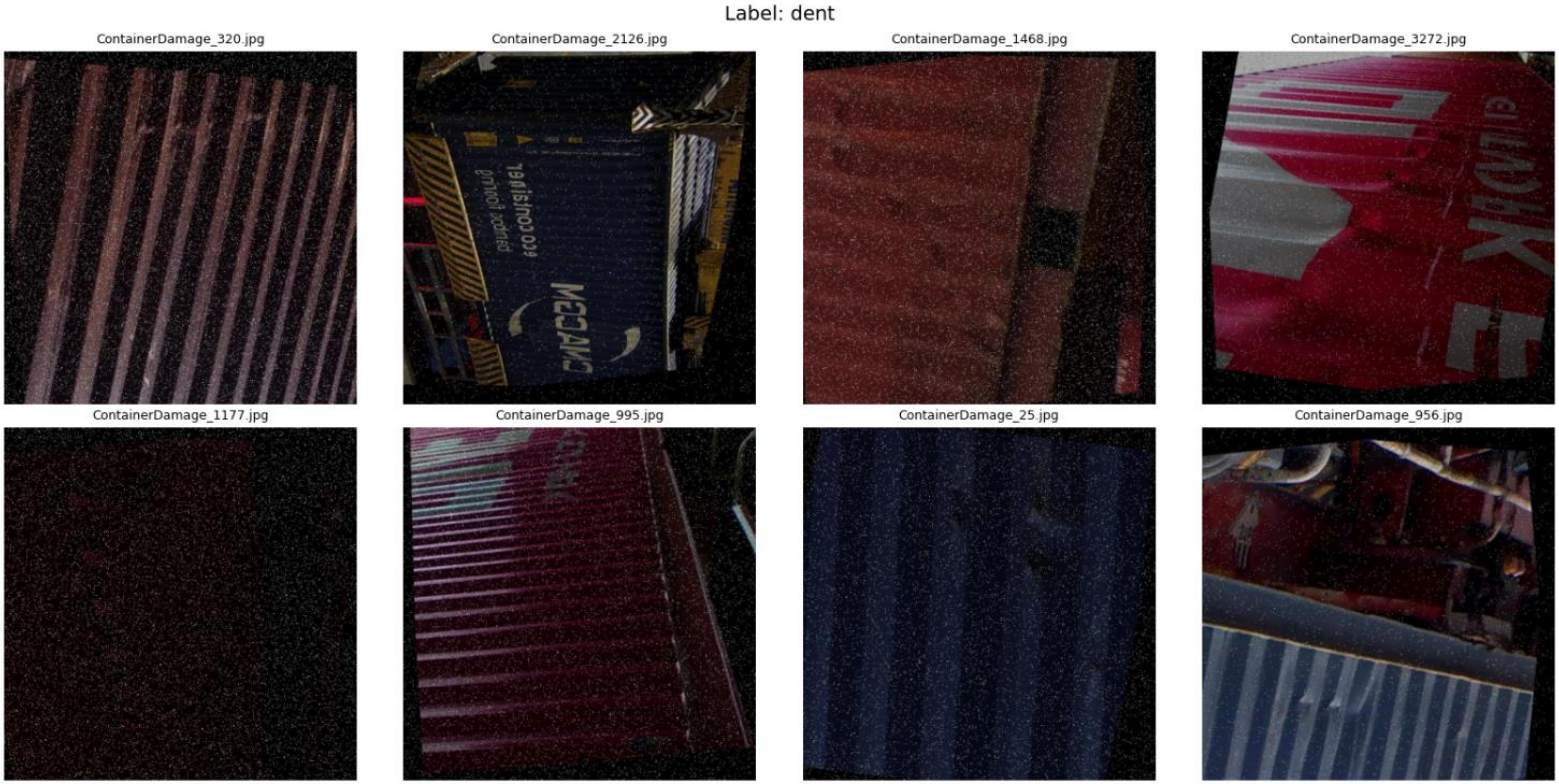
Flaws - Crack


Label: crack
ContainerDamage_1175.jpg
ContainerDamage_44.jpg
ContainerDamage_1906.jpg
ContainerDamage_1011.jpg
ContainerDamage_2587.jpg
ContainerDamage_3488.jpg
ContainerDamage_1644.jpg
ContainerDamage_2680.jpg
BCSC

Flaws - Dent

feature	importance_mean
fft_residual_local_output_norm_std	0.067314
fft_residual_local_output_norm_with_harmonics_std	0.064570
residual_laplacian_with_harmonics_std	0.039578
residual_laplacian_with_harmonics_masked_area_...	0.033291
residual_laplacian_with_harmonics_masked_max_area	0.026275
residual_laplacian_with_harmonics_p95	0.024947
residual_sobel_mag_with_harmonics_p95	0.024478
residual_laplacian_with_harmonics_mean	0.023847
residual_laplacian_masked_mean_aspect	0.023050
residual_sobel_mag_with_harmonics_mean	0.022950
residual_sobel_mag_with_harmonics_masked_area_...	0.022621
residual_sobel_mag_with_harmonics_std	0.022500
residual_laplacian_masked_max_area	0.022490
residual_laplacian_std	0.022132
residual_laplacian_p95	0.018808

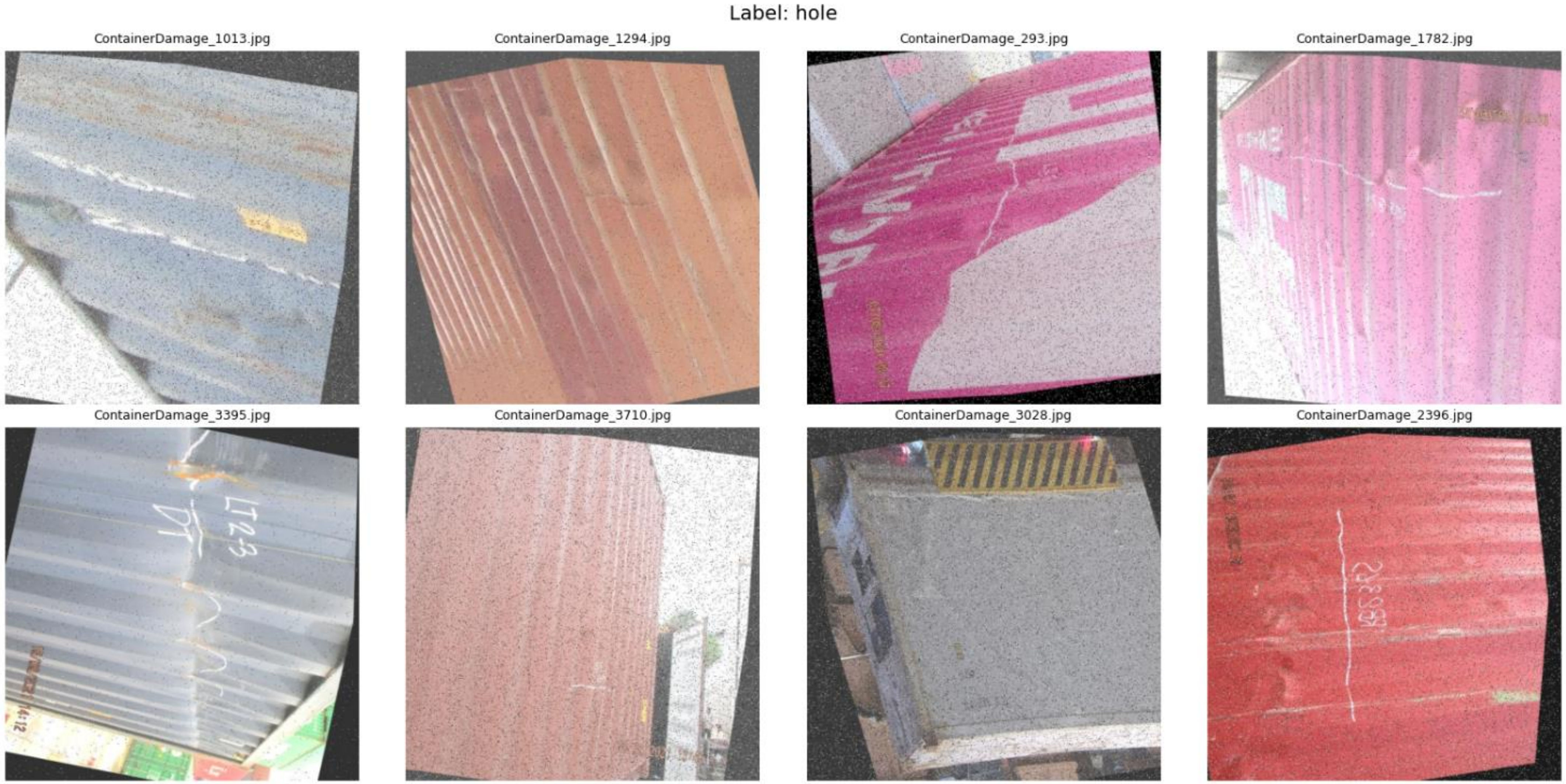
Flaws - Dent


Label: dent
ContainerDamage_320.jpg
ContainerDamage_2126.jpg
ContainerDamage_1468.jpg
ContainerDamage_3272.jpg
ContainerDamage_1177.jpg
ContainerDamage_995.jpg
ContainerDamage_25.jpg
ContainerDamage_956.jpg

Flaws - Hole

feature	importance_mean
fft_residual_local_output_norm_std	0.032805
residual_laplacian_with_harmonics_std	0.031896
fft_residual_local_output_norm_with_harmonics_std	0.028517
residual_laplacian_with_harmonics_masked_area_...	0.024824
residual_sobel_mag_with_harmonics_mean	0.022334
residual_laplacian_with_harmonics_mean	0.021448
residual_sobel_mag_with_harmonics_p95	0.020207
residual_sobel_mag_with_harmonics_std	0.020185
residual_laplacian_masked_mean_aspect	0.018951
residual_laplacian_masked_max_area	0.018294
residual_laplacian_with_harmonics_masked_max_area	0.017458
residual_sobel_mag_with_harmonics_masked_area_...	0.016688
residual_sobel_mag_with_harmonics_masked_num_c...	0.016152
residual_sobel_mag_masked_area_ratio	0.015536
fft_residual_local_output_norm_with_harmonics_p95	0.013424

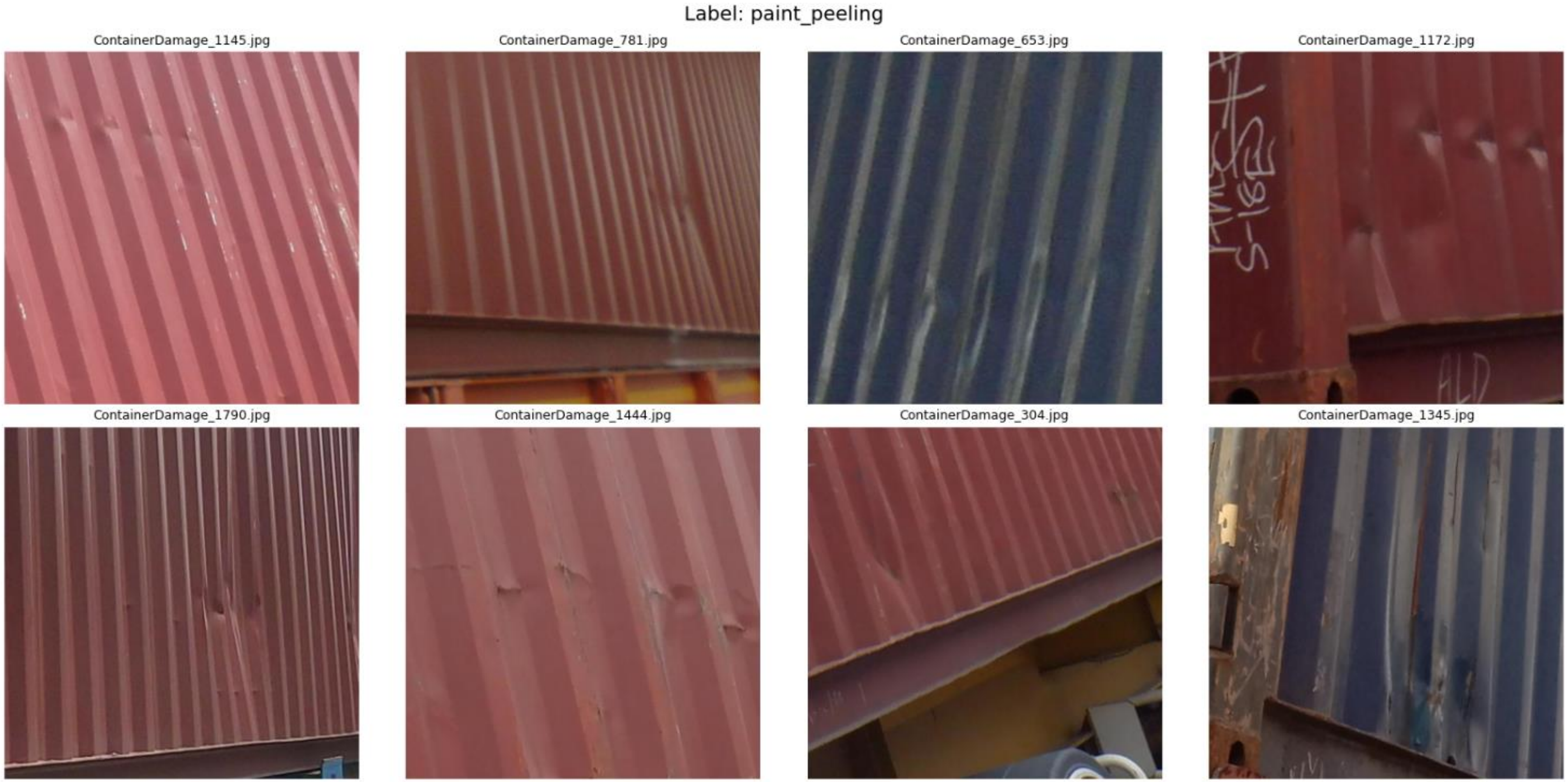
Flaws - Hole


Label: hole
ContainerDamage_1013.jpg
ContainerDamage_1294.jpg
ContainerDamage_293.jpg
ContainerDamage_1782.jpg
ContainerDamage_3395.jpg
ContainerDamage_3710.jpg
ContainerDamage_3028.jpg
ContainerDamage_2396.jpg

Flaws – Paint Peeling

feature	importance_mean
fft_residual_local_output_norm_with_harmonics_std	0.090252
fft_residual_local_output_norm_std	0.071942
residual_laplacian_with_harmonics_std	0.055557
residual_laplacian_with_harmonics_masked_area_...	0.049910
residual_laplacian_with_harmonics_mean	0.039925
residual_sobel_mag_with_harmonics_masked_area_...	0.039315
residual_sobel_mag_with_harmonics_std	0.038529
residual_laplacian_with_harmonics_p95	0.036407
residual_laplacian_p95	0.032051
residual_sobel_mag_with_harmonics_p95	0.031937
residual_sobel_mag_with_harmonics_mean	0.031266
residual_laplacian_masked_mean_aspect	0.031177
residual_laplacian_masked_max_area	0.029843
fft_residual_local_output_norm_with_harmonics_p95	0.029730
residual_laplacian_std	0.026883

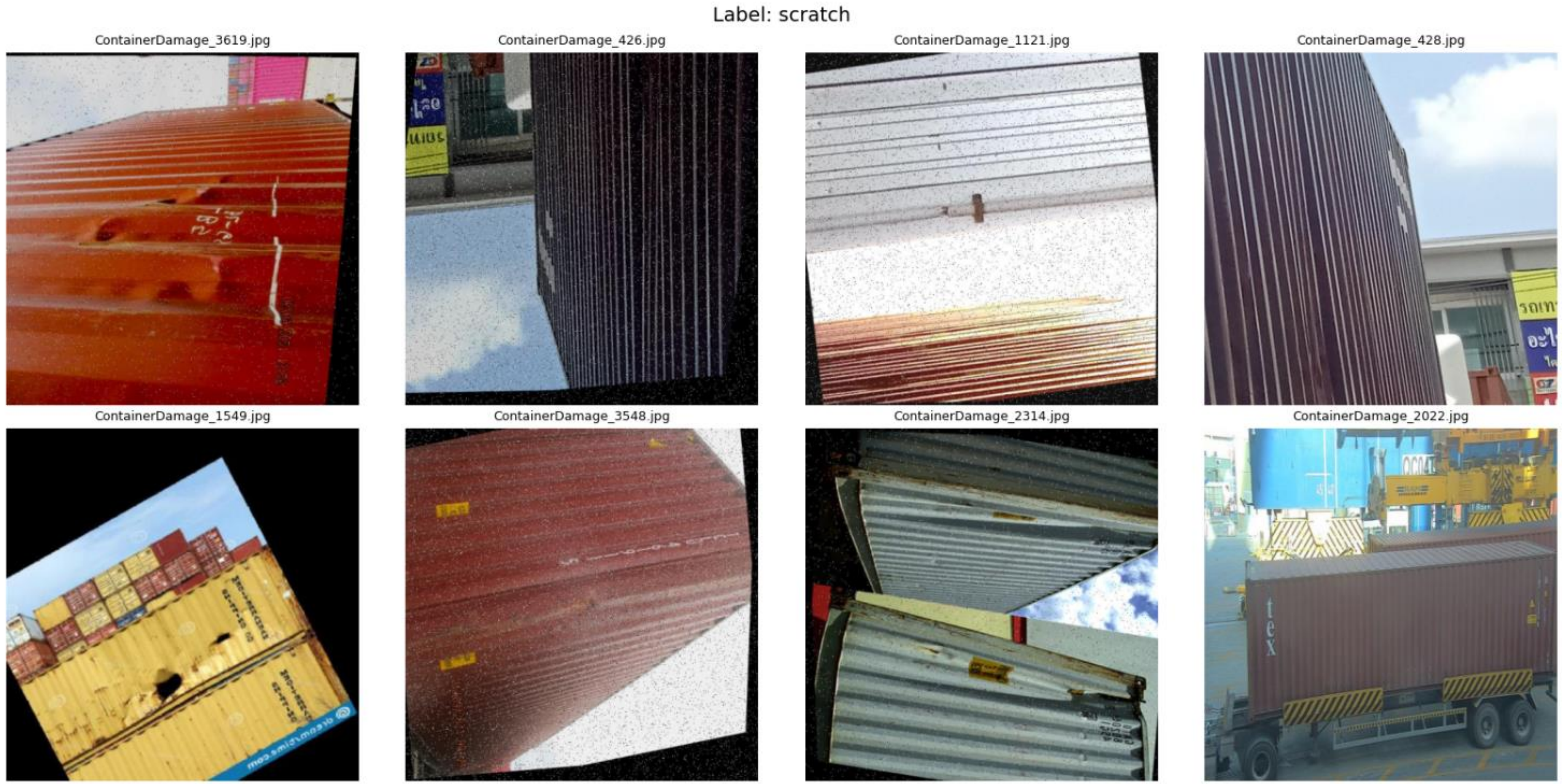
Flaws – Paint Peeling



Flaws - Scratch

feature	importance_mean
residual_laplacian_masked_max_area	0.040437
residual_laplacian_with_harmonics_masked_area_...	0.040313
residual_sobel_mag_with_harmonics_masked_num_c...	0.039356
residual_laplacian_masked_mean_aspect	0.035278
residual_laplacian_with_harmonics_mean	0.033240
residual_laplacian_with_harmonics_masked_max_area	0.032554
residual_laplacian_with_harmonics_std	0.032140
fft_residual_local_output_norm_with_harmonics_std	0.031475
fft_residual_local_output_norm_std	0.031465
residual_sobel_mag_masked_num_components	0.029552
residual_laplacian_with_harmonics_p95	0.027804
residual_laplacian_mean	0.027625
residual_sobel_mag_masked_mean_aspect	0.026297
residual_laplacian_p95	0.025377
residual_sobel_mag_with_harmonics_mean	0.024885

Flaws - Scratch



Conclusion

- According to the literature, our project is observed to be quite successful. In similar studies reported in the literature, the success rate is approximately in the range of 88%–94%. Due to the relatively easier nature of the dataset and the suitability of the methods we selected, our success rates were obtained as 99% on the training dataset, %94.1 on the validation dataset, and %94 on the test dataset. You can find the Kaggle notebook link of our project below.

- https://www.kaggle.com/code/osamamoselli/image-processing-notebook-v2