

Real-time Verification of Container Seal Integrity using Deep Learning

Sotiris Vasileiadis*, Sijun Yu*, Kyriacos Orphanides[†], Alessandro Cassera[†],
Michalis P. Michaelides* and Herodotos Herodotou*

*Cyprus University of Technology, 3036 Limassol, Cyprus

ORCID: 0009-0002-4479-1169, 0009-0005-4595-5657, 0000-0002-0549-704X, 0000-0002-8717-1691

Email: sr.vasileiadis@edu.cut.ac.cy; yy.yu@edu.cut.ac.cy; michalis.michaelides@cut.ac.cy; herodotos.herodotou@cut.ac.cy

[†]EUROGATE Container Terminal Limassol Ltd, 3045 Limassol, Cyprus

ORCID: 0009-0009-4510-3939, 0009-0005-5616-149X

Email: kyriacos.orphanides@eurogate-limassol.com; alessandro.cassera@eurogate-limassol.com

Abstract—Container seal integrity verification represents a critical yet underexplored challenge in maritime logistics security. Early theft detection is critical for minimizing operational delays, reducing costs, and ensuring safety in global maritime logistics. This work presents a deep learning-based methodology for real-time detection of container seals during crane unloading operations at container terminals. We develop and deploy two specialized YOLOv12-based object detection models: one for identifying container doors in motion and another for detecting in-place security fixtures, such as standard locks, container seals, and customs seals. Our models are trained and evaluated on a real-world dataset curated from video feeds captured at the EUROGATE Container Terminal in Limassol Port, Cyprus. The system is designed for robust performance under realistic terminal conditions, including variable lighting and motion. Our models achieve high detection accuracy across variable weather conditions, with mAP50 scores of 0.95 for container door detection and 0.74 for seal detection, substantially outperforming existing benchmarks. These findings underscore the practical potential of our methodology for enhancing the efficiency and security of automated maritime logistics.

Index Terms—computer vision; deep learning; container door detection; container seal detection; maritime container terminal;

I. INTRODUCTION

CONTAINERS are fundamental to global maritime trade, enabling the transport of import, export, and transshipment cargo. As international seaborne container traffic continues to grow rapidly, port operators are under increasing pressure to enhance and automate terminal processes in order to handle demand efficiently while minimizing safety risks associated with human-machine interaction [1]. Currently, there are around 20 million active containers worldwide, collectively making more than 200 million journeys each year [2]. Ports serve as critical nodes where goods from across the globe are consolidated, transferred, and dispatched, with terminals responsible for the secure loading, unloading, and handling of these containers.

However, the sheer volume of container traffic makes monitoring and detecting theft or illegal activities particularly difficult. In practice, most inspection procedures still rely heavily on manual checks [3], [4], which are often inefficient,

time-consuming, costly, error-prone, and pose safety risks. According to the International Maritime Organization (IMO), losses due to compromised container seals can reach up to \$1.7 billion annually, with approximately 34% of incidents attributed to failures in manual inspection processes [5], [6].

For this reason, effective inspection of container integrity is crucial in preventing disputes between port operators and transportation companies, especially in cases involving suspected theft [7], [8]. Containers are typically secured using various types of seals, such as standard key locks, customs seals, and company-specific security seals. Damage to these seals may result from issues like improper handling during transit or inadequate storage conditions at the terminal, but it can also indicate potential security breaches. If improperly sealed containers are not identified upon arrival, shipping companies may hold ports liable and impose penalties, assuming tampering or theft occurred within the terminal.

In contemporary research, computer vision and deep learning frameworks are increasingly utilized to monitor logistical integrity at seaports [3], [9]. Machine learning techniques have become the gold standard for object detection across diverse sectors, including security, military, and transportation [10]. To modernize container security verification at maritime terminals, industry stakeholders and researchers are prioritizing automated systems for inspecting container seals [11], [12].

For example, early approaches explored image processing methods, such as analyzing hue, saturation, and intensity, via fixed sensors and cameras [13]. Subsequent studies [14] applied phase correlation techniques to identify seal placement and status. However, many existing systems rely on cloud-based processing and storage, which requires transmitting large volumes of data from the terminal to remote servers. This architectural bottleneck consumes significant time and bandwidth, making it impractical for the high-velocity requirements of real-time port operations. To circumvent these limitations, the edge computing paradigm [15] has been introduced, enabling data processing at the source to meet strict real-time performance standards.

To enhance the reliability and speed of security inspections, this study introduces a deep learning-based framework

designed for the real-time detection of container doors and security seals during crane unloading operations. Utilizing the YOLOv12 architecture, initially pre-trained on the COCO128 dataset and subsequently fine-tuned via transfer learning, we optimize for low-latency performance in maritime environments. The primary contributions of this work are threefold:

- **A Specialized Multi-Stage Detection Pipeline:** We propose a dual-model approach that sequentially identifies container doors and security fixtures, ensuring robust verification even in motion.
- **A Real-World Terminal Dataset:** We utilize a manually-labeled dataset curated from live video feeds at the EUROGATE Container Terminal, capturing the environmental variability of a real port.
- **Edge-Optimized Deployment:** We demonstrate a successful transfer learning methodology based on YOLOv12 that enables high-precision inference on edge devices, achieving mAP50 scores of 0.95 for door detection and 0.74 for seal identification.

The rest of this paper is structured as follows. Section II reviews the related work. Section III describes the proposed methodology in detail and outlines the datasets used. Section IV presents the experimental setup, the obtained results, and their discussion. Finally, Section V summarizes the study and provides concluding remarks.

II. RELATED WORK

Only a few studies have explored the development of automated approaches for seal detection. In the domain of automated maritime security, Bahrami et al. [7] proposed a specialized deep learning framework, High-Resolution with Context R-CNN (HRCR-CNN), specifically designed to detect security seals on shipping containers. Recognizing the limitations of manual visual inspection at terminal gates, the authors introduced a Multi-Depth Super-Resolved Features Generation (MDSRFG) module. This module enhances detection by processing high-resolution images with shallow networks to preserve spatial positional information, while simultaneously using deep networks on low-resolution images to extract complex semantic information. Furthermore, the framework incorporates an attention mechanism combined with long-term memory to leverage contextual information from unlabeled data, thereby improving robustness across diverse environmental conditions at seaports. Their experimental results demonstrate that specialized architectures can accelerate container logistics by automating seal verification and provide a technical foundation for security monitoring in supply chain management. Unlike the HRCR-CNN framework [7], which relies on complex multi-depth super-resolution modules that increase computational overhead, our use of the YOLOv12 architecture strikes a superior balance between high-speed inference and detection accuracy, without requiring additional feature-generation modules.

To provide a more comprehensive view of the field, Basaras et al. [8] explored the deployment of a computer vision-based seal-detection service at the far edge of a 5G network. The

authors highlight that container seals are typically located in the lower-right quadrant of the container door but pose a significant detection challenge due to their small size relative to the container and visual noise from rust, stickers, and shadows. This work proposes a template-matching-based algorithm optimized for low-latency execution on 5G-connected IoT devices. Their research underscores the importance of local data processing to meet the 30-second-per-container inspection window required at modern maritime terminals, aligning with the industry's growing shift toward decentralized, real-time, automated inspection systems. While [8] uses template matching to overcome data scarcity at the edge, our framework leverages a manually labeled, real-world dataset, enabling our deep learning models to generalize much more effectively across variable lighting and visual noise than rigid template-based algorithms.

The work proposed by Delgado et al. [16] is an automated pipeline to streamline the visual inspection of shipping containers during port operations. Using high-resolution video streams from cameras mounted on Ship-to-Shore (STS) cranes, the system employs multiple deep learning models to concurrently detect container IDs, identify structural damage such as dents or holes, recognize hazardous material labels, and determine container orientation, specifically the Door/No Door orientation. To address the lack of real-world data, the researchers used the synthetic dataset SeaFront for training and validation. A heuristic algorithm then fuses the information from different camera angles over time to generate a single, comprehensive report for each container. While the system performs well at identifying IDs and container positions, the authors noted that further work is needed to improve the detection of smaller structural defects. Whereas [16] relies on synthetic data for crane-based inspections, our models are trained and validated on actual terminal video feeds, ensuring that our high mAP50 scores translate directly to robust performance in live maritime environments.

III. DATASET DESCRIPTION AND METHODOLOGY

The current study develops two models for container door detection and seal detection for the Port of Limassol, Cyprus. Both models utilize a pre-trained YOLOv12 model [17], [18], which we fine-tuned on custom datasets created specifically for this research. Separating the tasks into two models provides key advantages: seal detection is only performed when a container door is first identified, reducing unnecessary computation and enabling the second model to be more specialized and focused. Further details regarding the datasets and the chosen methodology are presented next.

A. Dataset for Container Door and Seal Detection

This study uses a real-world dataset from the EUROGATE Container Terminal Limassol in Cyprus. The dataset consists of video feeds captured by IPTV cameras installed on STS cranes that load and unload containers onto and from vessels. These videos were converted into image frames, which were

manually labeled whenever they contained container doors and door seals.

To prepare the data for the models, we collected videos recorded throughout 2024 and 2025 at various times of day and night and under different weather conditions (e.g., sunny, cloudy, rainy). These videos were converted into frames, after which each frame was manually reviewed, and bounding boxes were defined around the container doors using an open-source tool called LabelImg (also known as Label Studio) [19]. Over 1000 images containing containers were labeled, including containers of varying sizes (e.g., 20 ft, 40 ft), types (e.g., regular, reefer, open top), and colors (e.g., yellow, blue, red).

For the seal detection model, we utilized images from the door dataset. First, we cropped the images so that only the door was visible, and then we followed the same labeling procedure as for the first model, but applied it to seals instead, as shown in Figure 1. There are a few types of seals available. For instance, Figure 1(b) contains two different types. Overall, we tried to include as many as we could from our video footage, but we do not distinguish among the various seal types.

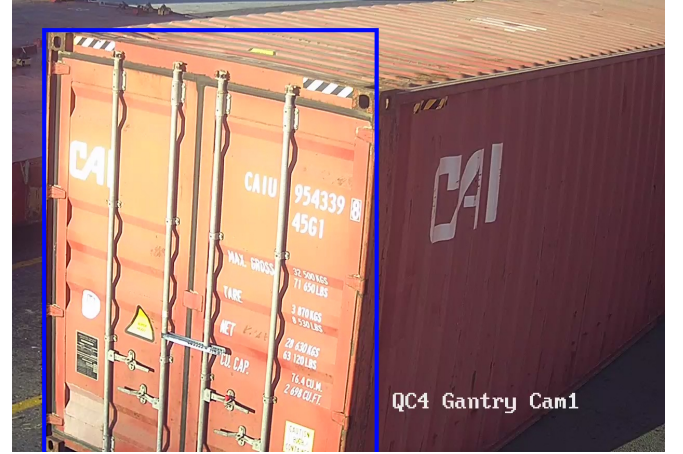
In an effort to increase our dataset, we attempted to employ data augmentation techniques, including horizontal flips and brightness adjustments. However, after extensive experimentation, we concluded that, due to the small and sensitive nature of seals, these augmentations hindered rather than improved the model's performance. These findings are further discussed in Section III-C.

Both datasets were then randomly split into training, validation, and testing. The container doors dataset contains 513, 152, and 500 images for training, validation, and testing, respectively, while the seals dataset includes 602, 129, and 176 images for training, validation, and testing, respectively. To prevent identity-based data leakage, all subsets were manually verified to be mutually exclusive; frames featuring a specific container were strictly confined to a single split.

B. Model Development

YOLOv12 is a state-of-the-art object detection model that achieves a strong balance between speed and accuracy, making it well-suited for real-time applications [20]. The model is trained on the COCO128 dataset for 100 epochs, with parameters optimized using techniques such as stochastic gradient descent (SGD) and its variants. Architecturally, YOLOv12 is based on a deep Convolutional Neural Network (CNN) that partitions an input image into a grid and predicts object bounding boxes and class probabilities for each grid cell. In contrast to traditional two-stage detectors such as R-CNN, YOLOv12 uses a single-stage approach. This design eliminates the need for a separate region-proposal step, significantly improving inference speed while maintaining competitive accuracy, making it particularly suitable for applications such as video surveillance, autonomous driving, and robotics [21].

For container door detection, a customized YOLOv12 model is trained on a dedicated dataset of container images. The training process involves careful tuning of key parameters to



(a) Labeled image from the container door dataset



(b) Labeled image from the seal dataset

Fig. 1: Example of labeled images from (a) the container door and (b) the seal datasets (note the small red seal on the handle)

maximize performance, including 100 training epochs, an input image resolution of 1024 pixels to preserve fine details, and a batch size of 24 to ensure efficient computation. The model is initialized from the pre-trained YOLOv12n variant, which offers a favorable trade-off between computational efficiency and detection capability. This configuration enables accurate and reliable detection of container doors in real-world con-

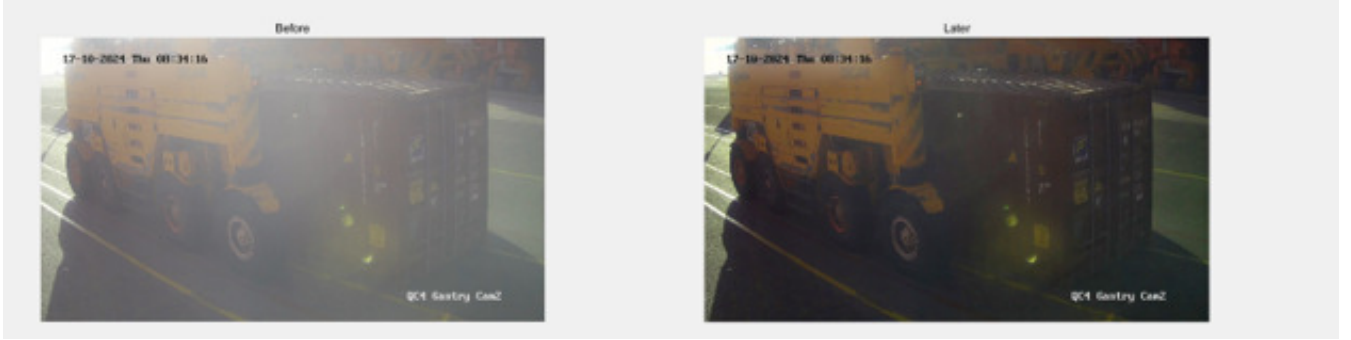


Fig. 2: Comparison of before and after image enhancement

TABLE I: Parameter ranges and optimal values for hyperparameter tuning for both models

Parameter	Tested Range	Optimal Value
IoU Threshold	0.3 – 0.7	0.50
HSV Saturation	0.0 – 0.9	0.30
HSV Value	0.0 – 0.9	0.30
HSV Hue	0.0 – 0.5	0.05

ditions. For seal detection, the same pre-trained YOLOv12n model is further trained on a dedicated seal dataset. A similar optimization strategy is applied as in the container detection stage to maintain consistency and performance. To further enhance model performance, various configuration settings available in YOLOv12 are explored through hyperparameter tuning. This includes augmenting input images by adjusting properties such as hue and saturation, thereby improving robustness in detecting container doors and seals under varying visual conditions. The explored training configuration parameter ranges and the selected optimal values are summarized in Table I and are applied consistently across both the door and seal detection models.

C. Image Enhancements

While this study initially sought to address illumination challenges such as haze, low-light conditions, and overexposure through a tailored image enhancement pipeline, empirical testing indicated that these pre-processing steps did not yield the expected performance gains. We developed and implemented an adaptive defogging algorithm based on the Dark Channel Prior (DCP), optimized using CUDA kernels, achieving a processing time of ≤ 8 ms per frame at 1080p resolution. This was followed by an adaptive gamma correction stage to dynamically adjust luminance. Although these techniques successfully enhanced the visual clarity of containers and reduced glare, as shown in Figure 2, they introduced unexpected noise and artifacts that ultimately confused the YOLO model's feature extraction. Comparative testing showed that baseline models trained on unprocessed, default images achieved higher mAP scores and better localization of small objects, such as seals. Consequently, the enhancement pipeline

TABLE II: Comparison of model performance across image enhancement techniques

Method	Precision (P)	Recall (R)	mAP50
No Enhancement (Baseline)	0.814	0.787	0.820
Dehazing	0.756	0.714	0.726
Gamma Adjustment	0.830	0.751	0.788

was excluded from the final architecture to maintain higher detection accuracy and lower computational overhead.

The comparative analysis in Table II reveals that the baseline model, using unprocessed, default images, outperformed both the Dehazing and Gamma Adjustment interventions on the primary metric, mAP50 (0.82). While the Gamma Adjustment method achieved slightly higher precision (0.83), it suffered a notable drop in recall, suggesting that the luminance corrections may have inadvertently suppressed relevant features or introduced noise, hindering the model's ability to identify all ground-truth objects. The Dehazing approach was the least effective, resulting in a significant degradation across all metrics. Consequently, the results indicate that the YOLO model is more resilient and accurate when trained on the original dataset without additional pre-processing enhancements.

D. Pipeline Deployment on the Edge

The developed computer vision models were integrated into a complete end-to-end deployment pipeline. Figure 3 illustrates the full workflow of the system in a step-by-step manner. Initially, video footage is captured in real time using CCTV cameras (Hikvision DS-2XM6756G0-IS/ND) mounted on ship-to-shore (STS) cranes, providing a dynamic, elevated viewpoint of container-handling operations. These cameras continuously stream data over IP-based Ethernet networks, ensuring low-latency and reliable transmission.

The video streams are routed through network switches and directed toward edge computing devices, specifically NVIDIA Jetson AGX Orin units. At this stage, the incoming frames are processed locally using the deployed deep learning models for tasks such as container detection, damage identification, door detection, and seal inspection. Performing inference at the edge significantly reduces bandwidth usage and minimizes response time, enabling near real-time analysis without requiring

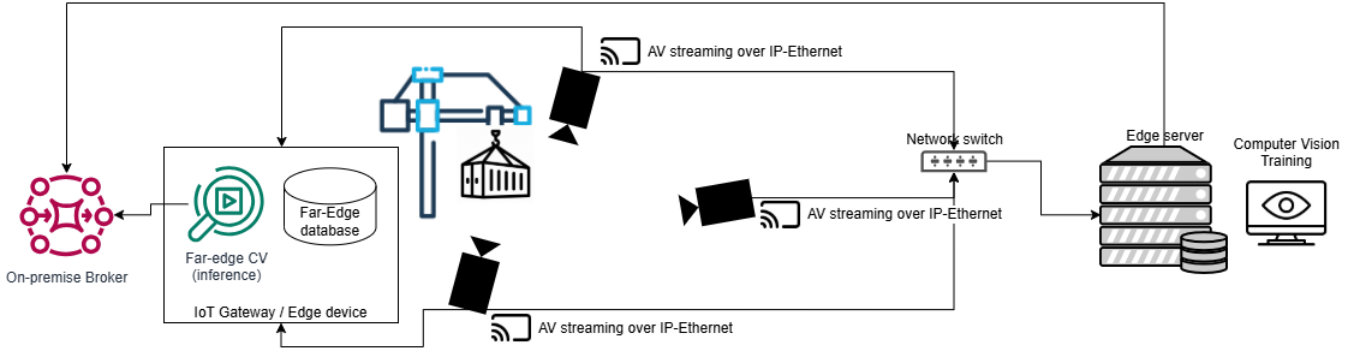


Fig. 3: Visualization of the edge-based deployment scheme

constant communication with centralized infrastructure. The results of the inference process, along with relevant metadata, are stored temporarily in a far-edge database for quick access and buffering.

After the inference, the processed data is transmitted to an on-premises EMQX MQTT broker, which serves as the system's central messaging hub. This broker enables efficient, scalable data distribution to downstream components, allowing multiple services to subscribe to relevant topics. From there, the data can be forwarded to higher-level systems, such as edge servers used for aggregation, long-term storage, or further analytics, as well as computer vision training environments where new models can be refined using collected data. This architecture ensures a modular, scalable, and efficient deployment pipeline that supports real-time monitoring and continuous model improvement within port operations.

IV. RESULTS AND DISCUSSION

This section presents the experimental evaluation of the container door and seal detection YOLOv12 models and compares their performance with other state-of-the-art approaches. As described in Section III-A, the study uses real-world datasets collected at the EUROGATE Container Terminal Limassol, Cyprus.

All model training was conducted on a server equipped with two AMD EPYC 7352 24-core processors, 128 GB of RAM, and an NVIDIA Tesla T4 GPU, running Ubuntu 22.04.4 LTS. Model testing was conducted on both the training server and on an NVIDIA Jetson AGX Orin 64GB edge device.

A. Container Door Detection Model Evaluation

Figure 4 shows validation results for the door detection model, demonstrating its ability to detect container doors across frames, including scenarios with multiple containers per image or partially viewed containers. To quantitatively evaluate the performance of the developed models, we adopt the mAP50 metric, which is widely used in object detection tasks [22]. The mAP50 metric measures the mean average precision at an intersection-over-union (IoU) threshold of 0.5, indicating how well the model detects objects with at least 50% overlap between predicted and ground-truth bounding

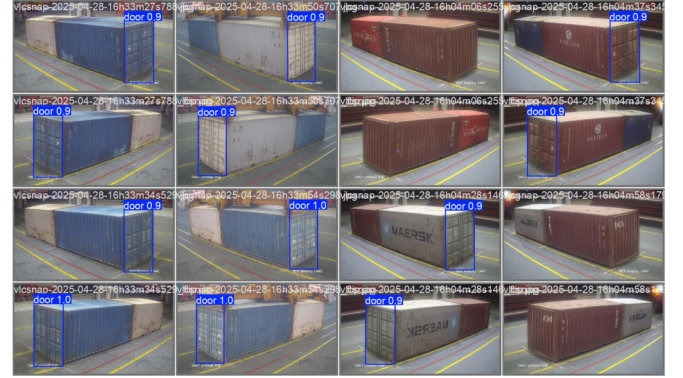


Fig. 4: Validation results for container door detection

boxes. In addition, the F1 score is considered, as it represents the harmonic mean of precision and recall, offering a balanced measure of the model's detection performance. Together, these metrics provide a thorough and reliable evaluation of the model's overall effectiveness.

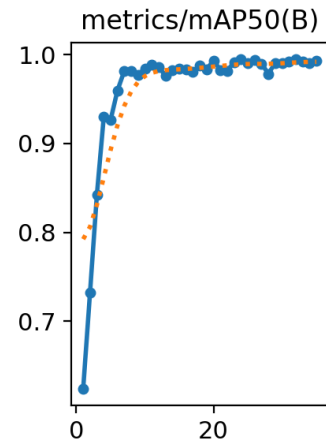


Fig. 5: mAP50 for container door detection during training (the x-axis denotes the number of epochs)

The container door detection model was evaluated using mAP50, achieving strong results. As shown in Figure 5, the

model achieved mAP50 values between 0.96 and 0.99 within the first 10 epochs. The best performance was observed at epoch 24, when the model reached mAP50 of 0.999 on the validation dataset and 0.95 on the test dataset. These results indicate high accuracy and consistency, reflecting the model's robustness and precise detection capabilities. Furthermore, despite being trained on a relatively limited dataset, the model maintains strong performance, underscoring its effectiveness for door detection tasks.

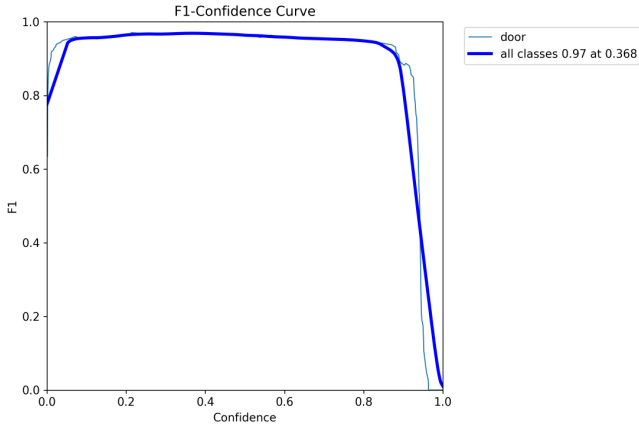


Fig. 6: F1 score for container door detection across varying confidence thresholds

The results shown in Figure 6 present the F1 scores of the door detection model across different confidence thresholds, offering insight into the model's balance between false positives and false negatives. The model demonstrates strong performance, achieving a near-perfect F1 score across confidence thresholds from 0.2 to 0.8. This reflects a solid balance between precision and recall, confirming the model's reliability in accurately detecting container doors. The consistently high F1 scores across thresholds also highlight the robustness of the detection system.

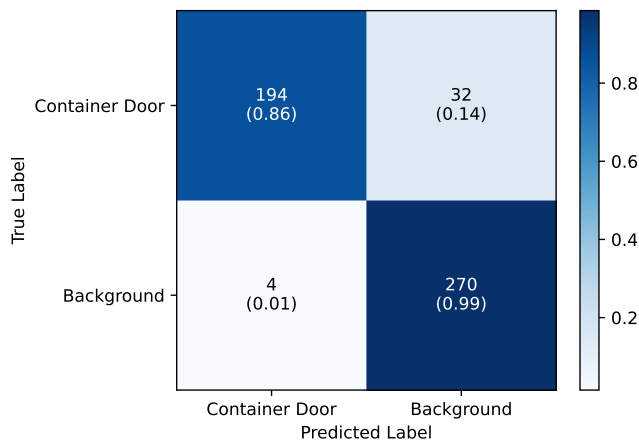


Fig. 7: Confusion matrix for container door detection using the test dataset



Fig. 8: Test results for seal detection

This study also develops a confusion matrix to validate the door detection model's performance, as shown in Figure 7. A confusion matrix is used to evaluate classification performance by comparing predicted labels with ground-truth labels for each class. It provides a structured overview of correct and incorrect predictions, allowing performance to be quantified through metrics such as true positives, true negatives, false positives, and false negatives. This makes it particularly useful for identifying specific types of errors and gaining deeper insight into model behavior beyond overall accuracy. As illustrated in Figure 7, only 36 out of the 500 test images were misclassified (32 as false negatives and 4 as false positives), corresponding to a relatively small error rate (7.2%). This outcome highlights the model's strong ability to correctly identify container doors while maintaining a low number of incorrect detections, further supporting its reliability and effectiveness in practical deployment scenarios. A significant number of background-only images were intentionally included in the test set to account for diverse operational scenarios, such as empty frames, container sides without a door, and open-top containers. Including these variations ensures a robust evaluation of the model's resistance to false positives.

B. Seal Model Evaluation

Figure 8 shows results from the seal detection model on the test dataset. The model performs very well at distinguishing the seals when present, with varying levels of confidence. Even when the image is grainy and unclear, it typically does not pick up any noise as false positives.

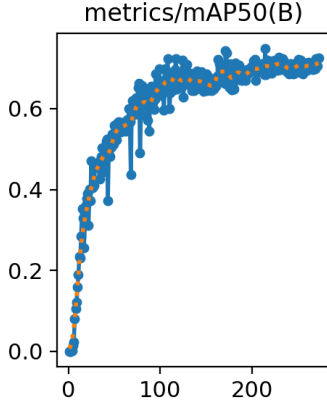


Fig. 9: mAP50 for seal detection during training (the x-axis denotes the number of epochs)

Figure 9 illustrates the mAP50 performance of the seal detection model. The model reached a peak mAP50 of 0.743 after approximately 200 training epochs and achieved 0.737 on the test dataset, indicating moderate detection accuracy. The consistent improvement of the metric throughout training suggests stable convergence and an effective learning process. The moderate localization precision is likely limited by the difficulty of precisely bounding small, moving objects.

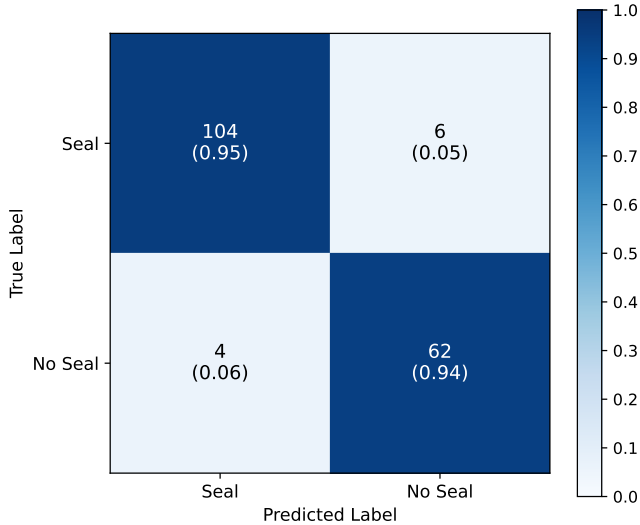


Fig. 10: Confusion matrix for seal detection using the test dataset

The performance of the seal detection model on the test dataset is presented using a confusion matrix in Figure 10. The primary objective of this model is to correctly identify the presence of at least one seal per image. As shown in the confusion matrix, the model produces very few false positives (i.e., incorrectly detecting seals where none exist) and also rarely misses actual seals. Overall, the model achieves a detection rate of 95% for images containing at least one seal, while correctly identifying 94% of images without seals.

These results demonstrate that the model is both reliable and well-suited for practical deployment in seal verification tasks.

While the mAP50 of 0.737 reflects the inherent difficulty of precise localization for small-scale objects in high-motion environments, the high True Positive (95%) and True Negative (94%) rates demonstrate the framework's exceptional utility for binary integrity verification. In an operational context, the model's ability to reliably detect the presence or absence of a seal is more critical than sub-pixel bounding-box accuracy. Furthermore, its performance is competitive with, and in many cases surpasses, that of other state-of-the-art approaches, as will be discussed in Section IV-D.

C. Weather Condition Performance Analysis

TABLE III: Model performance on weather variation.

Weather	Correct Predictions	Incorrect Predictions	Accuracy (%)
Clear Day	60	5	92.3
Night (Low Light)	63	2	96.9
Hazy / Blurry	43	3	93.4

Note: Correct prediction of at least one seal or absence of a seal

Table III presents the model's performance under different weather and visibility conditions. The evaluation was performed by comparing the model's predictions with the ground-truth annotations for the presence or absence of container seals. A prediction was considered correct when the model accurately identified at least one seal that was present (true positive) or correctly determined that no seal was present (true negative). Conversely, a prediction was considered incorrect when the model detected a seal where none existed (false positive) or failed to detect at least one seal that was present (false negative). The results show that the model maintained high accuracy across all tested conditions, achieving 92% accuracy in clear daytime, 97% in low-light nighttime, and 93% in hazy/blurry environments. The slightly lower performance observed in clear daytime conditions is primarily attributed to strong sunlight conditions, which can reduce the visual contrast between seals and their surrounding container surfaces. For example, a white seal on a yellow container may become harder to distinguish in intense sunlight due to color washout and reflections. Nevertheless, the results indicate that the model is robust to variations in lighting and visibility, with only a small number of misclassifications observed across the evaluation dataset.

D. Comparison with State-of-the-Art Work

This section evaluates the proposed approach against widely adopted baseline models presented in [12], including MobileNetV2 [23], InceptionV3 [24], and Faster R-CNN with a ResNet50 backbone [25]. These are well-established benchmarks in the literature and provide a solid basis for comparative analysis. Table IV presents a comparative evaluation of MobileNetV2, InceptionV3, Faster R-CNN with a ResNet50 backbone, and YOLOv12 (our proposed approach), using

TABLE IV: Seal detection performance comparison across models

Class	MobileNetV2			InceptionV3			Faster R-CNN (ResNet50)			YOLOv12		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Seal	0.78	0.87	0.82	0.78	0.92	0.85	0.95	0.96	0.96	0.96	0.95	0.95
No Seals	0.70	0.57	0.63	0.79	0.56	0.65	0.94	0.91	0.92	0.91	0.93	0.92
Macro Avg	0.74	0.72	0.73	0.79	0.74	0.75	0.95	0.94	0.94	0.94	0.94	0.94
Weighted Avg	0.75	0.77	0.75	0.78	0.80	0.78	0.95	0.94	0.95	0.94	0.94	0.94

precision, recall, and F1-score as performance metrics. The evaluation is conducted on a dedicated seal test set consisting of 115 seal-positive and 61 seal-negative images.

To ensure a fair comparison across different model types, the object detection-based approaches (Faster R-CNN and YOLOv12) are evaluated in a binary manner. Specifically, an image is considered a True Positive if at least one seal is correctly localized. This formulation aligns the multi-object detection outputs with the binary predictions of the classification models, enabling a direct and consistent comparison.

The results show that InceptionV3 slightly outperforms MobileNetV2 among the classification models, achieving a macro-average F1-score of 0.75 and a weighted average of 0.78, compared to 0.73 and 0.75 for MobileNetV2, respectively. In contrast, YOLOv12 demonstrates strong, well-balanced performance, achieving an F1-score of approximately 0.94, with both precision and recall around 0.94. Faster R-CNN with a ResNet50 backbone yields comparable results, though with a slight reduction in mAP50.

A more detailed analysis reveals that both MobileNetV2 and InceptionV3 struggle to correctly identify images without seals, as reflected by their relatively low recall values of 0.57 and 0.56, respectively. This suggests a tendency to misclassify no-seal cases. By comparison, YOLOv12 achieves a more balanced precision-recall trade-off by explicitly localizing seal instances, thereby reducing false positives and improving robustness in practical deployment scenarios.

An important factor contributing to the observed performance differences is the input resolution used by each model. MobileNetV2 operates on 224×224-pixel inputs, InceptionV3 on 299×299, while YOLOv12 and ResNet50 process significantly higher-resolution images up to 1024×1024 pixels. The results indicate a clear relationship between input resolution and detection performance, with higher resolutions preserving finer spatial details that are essential for accurately identifying small, visually subtle objects such as container seals. Consequently, models operating on higher-resolution inputs are better able to capture these fine-grained features, leading to improved overall effectiveness.

However, it is important to note that the performance gains observed in YOLOv12 and ResNet50 cannot be attributed solely to input resolution. The models' object detection architecture enables them to learn both spatial localization and classification jointly, providing a significant advantage over conventional classification networks that operate on entire images without explicit spatial awareness.

In summary, YOLOv12 and ResNet50 achieve strong, balanced performance not only because they can process higher-resolution inputs, but also because of their detection-based formulation. Unlike MobileNetV2 and InceptionV3, which are limited to only predicting the presence or absence of a seal at the image level, YOLOv12 and ResNet50 are capable of: (i) detecting multiple seals within the same image, and (ii) localizing each detected seal by producing precise bounding boxes.

This consistency in performance extends to the mAP50 metric within the object detection framework. In this context, YOLOv12 achieves a score of 0.737, closely followed by ResNet50 at 0.730. By integrating localization with classification, the proposed approach is better suited for real-world deployment, particularly in scenarios involving multiple subjects where precise spatial data is critical. Consequently, the detection-based model offers not only competitive quantitative results but also practical utility that standard classification methods cannot provide.

E. Model Inference Speed

Reducing latency was a primary objective in the design of these models, due to their deployment in real-time applications. To assess performance, we benchmarked the developed models across two platforms: (i) the server used during model training, and (ii) a Jetson AGX Orin 64GB device that has been deployed on an STS crane, where it processes IPTV camera streams in real time, as discussed in Section III-D.

TABLE V: Inference speed and latency breakdown on two hardware setups

Metric	Jetson AGX Orin	Server Machine
GPU	Integrated	NVIDIA T4
Container Door Detection (ms)	17.5	9.8
Seal Detection (ms)	16.2	8.9
Other Processing (ms)	8.0	6.2
Total Frame Time (ms)	41.7	24.9
Achieved FPS	24	40

Note: Time metrics taken from the test dataset for seal detection.

Table V presents the average latency (in ms) for detecting a container door, identifying any seals, and performing additional processing steps (e.g., frame splitting, image enhancement, and result generation). The reported latency values are approximate and may be influenced by external factors; they are estimated based on the relative contribution of each

component to the overall frame processing time at an input resolution of 1024×1024.

Overall, even the more resource-constrained device, namely the Jetson AGX Orin, can process a full frame in 41.7 ms, highlighting the efficiency and lightweight design of the proposed models. It is also worth noting that the input video streams are recorded at six frames per second (FPS). As indicated in Table V, both the Jetson AGX Orin and the server equipped with an NVIDIA T4 GPU achieve inference speeds well above this rate, at approximately 24 FPS and 40 FPS, respectively. This performance headroom allows the system to handle multiple video streams simultaneously while preserving real-time operation.

TABLE VI: Performance comparison of inference between Faster R-CNN (ResNet50) and YOLOv12

Metric	Faster R-CNN (ResNet50)	YOLOv12
Total Inference Time	18.261 s	1.566 s
Avg. Time per Image	103.8 ms	8.9 ms
Throughput (FPS)	9.64	112.36
Relative Speed	1x (Baseline)	11.66x Faster

In our comparative inference speed evaluation (Table VI), the YOLOv12 architecture demonstrated a significant computational advantage over the ResNet50-based Faster R-CNN, achieving a throughput of 112.36 FPS compared to 9.64 FPS for the latter. While Faster R-CNN's two-stage approach, using a Region Proposal Network followed by a ResNet50 classifier, yielded an average latency of 103.8 ms per image, YOLO's single-stage regression-based approach reduced this to a mere 8.9 ms. This represents an 11.66x increase in inference speed, reducing the entire 176-image test set to only 1.57 seconds, whereas the ResNet50 model required 18.26 seconds. These results indicate that while Faster R-CNN remains a robust baseline, YOLO is substantially better suited for high-throughput or real-time edge-based applications where low latency is a critical operational requirement.

F. Memory Usage

Memory consumption is a critical factor when evaluating deep learning models for deployment, particularly in resource-limited or edge computing environments. As such, we analyzed the models' memory usage from two perspectives: model storage size and peak GPU memory consumption during inference. Peak GPU memory usage was measured during validation using TensorFlow's GPU memory information API for MobileNetV2 and InceptionV3, and PyTorch's peak CUDA memory statistics functions for YOLO and Faster R-CNN. The reported values correspond to the maximum GPU memory allocated during inference and provide an estimate of each model's runtime memory requirements.

Table VII summarizes the storage requirements and peak GPU memory usage of the four evaluated models: YOLOv12, Faster R-CNN (ResNet50), MobileNetV2, and InceptionV3. The results show clear differences in memory efficiency across the evaluated models. All models were evaluated with batch

TABLE VII: Model storage size and peak GPU memory usage during inference for each tested model

Model	File Size (MB)	Peak GPU Mem (MB)
YOLOv12	5.1	146.0
Faster R-CNN (ResNet50)	151.8	604.9
MobileNetV2	9.2	208.6
InceptionV3	84.2	2149.8

size 1 to reflect realistic deployment conditions (frame by frame analysis). YOLOv12 is the most efficient, requiring only 5.1 MB of storage and 146 MB of peak GPU memory. MobileNetV2 also demonstrates strong efficiency, with a model size of 9.2 MB and a peak GPU memory usage of 208.6 MB. The Faster R-CNN model requires substantially more resources, using 151.8 MB of storage and 604.9 MB of GPU memory. InceptionV3 is the most demanding model, reaching a peak GPU memory usage of 2149.8 MB.

Overall, YOLOv12 provides the best balance between storage and runtime memory requirements, making it the most suitable option for deployment in resource-constrained and real-time environments. MobileNetV2 offers a lightweight alternative, while Faster R-CNN and InceptionV3 achieve their performance at the cost of significantly higher resource consumption.

V. CONCLUSION

In this study, YOLOv12-based models for container door and seal detection are proposed to enhance operational efficiency at container terminals. The primary objective is to reduce reliance on manual inspection in hazardous environments by introducing automated systems that perform the same tasks with greater consistency and safety. The proposed models are trained on real-world data collected at the EURO-GATE Container Terminal Limassol, Cyprus. The models are subsequently validated on the test set, demonstrating strong performance with high detection accuracy, low error rates, and high efficiency.

The proposed approach offers significant advantages over existing state-of-the-art methods, both in terms of improved detection performance and enhanced functionality. In particular, the system can track multiple container doors within a single video frame and detect multiple seals on a single container door. This enables more comprehensive and scalable monitoring in real-world port operations, where multiple objects and instances must be analyzed simultaneously.

The development of the proposed system was subject to a few key constraints. The primary challenge in identifying specific seal types stemmed from the limited spatial resolution and detail of the terminal's existing IP cameras, as well as the scarcity of public data on modern seal designs. Furthermore, the moderate accuracy in seal verification reflects the limited scale of our custom dataset and the operational challenges posed by variable weather and lighting conditions at the container terminal.

Moving forward, we plan to implement a self-improving feedback loop that autonomously identifies and integrates new door and seal samples into our training pipeline, iteratively enhancing model robustness. Furthermore, we aim to transition this framework into a live operational environment by interfacing our detection models with existing port IP camera networks and the broader terminal management system (TMS).

ACKNOWLEDGMENT

This research was funded by the European Union's Horizon Europe program for Research and Innovation through the aerOS project under Grant No. 101069732.

GENERATIVE AI DISCLOSURE STATEMENT

During the preparation of this work, the authors used Gemini 3.5 Flash in order to improve the language and clarity of the manuscript. After using this tool, the authors reviewed and edited the content as needed, and take full responsibility for the publication's content.

REFERENCES

- [1] S. Aslam, M. P. Michaelides, and H. Herodotou, "A Survey on Computational Intelligence Approaches for Intelligent Marine Terminal Operations," *IET Intelligent Transport Systems*, vol. 18, no. 5, pp. 755–793, 2024. doi: 10.1049/itr2.12469
- [2] Universal Containers, "13 Facts About Shipping Containers," <https://universal-containers.com/news/13-facts-about-shipping-containers/>, 2024, last accessed: April 15, 2026.
- [3] H. Wang, Q. Liu, and G. Zhang, "Container Damage Detection Algorithm Based on Fast-Solo," in *Chinese Intelligent Systems Conference*. Springer, 2022. doi: 10.1007/978-981-19-6226-4_13 pp. 119–131.
- [4] S. Aslam, H. Herodotou, E. Garro, A. Martínez-Romero, M. A. Burgos, A. Cassera, G. Papas, P. Dias, and M. P. Michaelides, "IoT for the Maritime Industry: Challenges and Emerging Applications," in *18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023. doi: 10.15439/2023F3625 pp. 855–858.
- [5] S. Jakovlev, T. Eglynas, V. Jankunas, M. Jusis, and M. Voznak, "Analysis of Damage to Shipping Container Sides during Port Handling Operations," *Journal of Marine Science and Engineering*, vol. 13, no. 5, p. 982, 2025. doi: 10.3390/jmse13050982
- [6] E. Grey, "Cargo Theft: A Billion Dollar Problem in Shipping," <https://finance.yahoo.com/news/cargo-theft-billion-dollar-problem-121530759.html>, 2024, last accessed: April 15, 2026.
- [7] Z. Bahrami, R. Zhang, R. Rayhana, and Z. Liu, "An HRCR-CNN Framework for Automated Security Seal Detection on the Shipping Container," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021. doi: 10.1109/TIM.2021.3117079
- [8] P. Basaras, M. Antonopoulos, K. V. Katsaros, G. Kanellopoulos, S. Tsagalis, and A. J. Amditis, "5G Enabled Video Analytics for Detecting Container Seals in Port Operations," in *Proceedings of the 8th International Physical Internet Conference (IPIC)*, 2021. [Online]. Available: <https://repository.gatech.edu/bitstreams/574f27f0-5e8a-4075-b13d-ef40836592a6/download>
- [9] S. Vasileiadis, S. Aslam, K. Orphanides, A. Cassera, E. G. Crevillen, A. Martínez-Romero, M. P. Michaelides, and H. Herodotou, "Real-time Container Tracking and Damage Detection at Seaports Using Deep Learning," in *2025 20th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2025. doi: 10.15439/2025F5518 pp. 277–285.
- [10] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of Deep Learning for Object Detection," *Procedia Computer Science*, vol. 132, pp. 1706–1717, 2018. doi: 10.1016/j.procs.2018.05.144
- [11] X. Li, X. Huang, and Q. Liu, "Container Damage Identification Based on RP-FCN," in *39th Chinese Control Conference (CCC)*. IEEE, 2020. doi: 10.23919/CCC50068.2020.9189392 pp. 7031–7034.
- [12] Z. Wang, J. Gao, Q. Zeng, and Y. Sun, "Multitype Damage Detection of Container using CNN based on Transfer Learning," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–12, 2021. doi: 10.1155/2021/5395494
- [13] B.-S. Jeng, Q.-Z. Wu, Y.-P. Chen, and W.-Y. Cheng, "Image-based Container Defects Detector," Oct. 3 2006, US Patent 7,116,814.
- [14] C. Tang, P. Chen, and Y. Li, "Automatic Damage-Detecting System for Port Container Gate Based on AI," in *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, 2020. doi: 10.1145/3436369.3436480 pp. 146–151.
- [15] W. Shi, X. Zhang, Y. Wang, and Q. Zhang, "Edge Computing: State-of-the-art and Future Directions," *Journal of Computer Research and Development*, vol. 56, no. 1, pp. 69–89, 2019. doi: 10.7544/jssn1000-1239.2019.20180760
- [16] G. Delgado, A. Cortés, and E. Loyo, "Pipeline for Visual Container Inspection Application using Deep Learning," in *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*. SCITEPRESS, 2022. doi: 10.5220/0011590900003332 pp. 404–411.
- [17] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-centric Real-time Object Detectors," *arXiv preprint arXiv:2502.12524*, 2025.
- [18] M. A. R. Alif and M. Hussain, "YOLOv12: A Breakdown of the Key Architectural Features," *arXiv preprint arXiv:2502.14740*, 2025.
- [19] Tzatalin, "LabelImg," <https://github.com/HumanSignal/labelImg>, 2022, last accessed: April 15, 2026.
- [20] H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi, and H. Chen, "DC-YOLOv8: Small-size Object Detection Algorithm based on Camera Sensor," *Electronics*, vol. 12, no. 10, p. 2323, 2023. doi: 10.3390/electronics12102323
- [21] Ultralytics, "Ultralytics - Revolutionizing the World of Vision AI," <https://www.ultralytics.com/>, May 2024, last accessed: April 15, 2026.
- [22] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time Flying Object Detection with YOLOv8," *arXiv preprint arXiv:2305.09972*, 2023. doi: 10.48550/arXiv.2305.09972
- [23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.48550/arXiv.1801.04381 pp. 4510–4520.
- [24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.48550/arXiv.1512.00567 pp. 2818–2826.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2016.90 pp. 770–778.