Balancing Accuracy and Efficiency: Navigating The Trade-off Between Machine-Readable Code Detection and Data Size Reduction

Imen Jegham, Ons Loukil, Besma Guesmi, David Moloney
Ubotica Technologies, Dublin, Ireland;
Email: {imen.jegham, ons.loukil, besma.guesmi, david.moloney}@ubotica.com

***Abstract*—Detecting machine-readable codes in industrial manufacturing is a critical yet challenging task, as it directly impacts both defect detection and broader visual anomaly detection. The complexity of visual data, coupled with high-speed production processes generating vast volumes of information, makes accurate and efficient detection essential for ensuring product quality and operational efficiency. This paper takes significant steps toward addressing these challenges by identifying and categorizing key issues related to machine-readable code defect detection, introducing the first publicly available and challenging dataset as a benchmark for future studies, and exploring techniques to enhance both detection performance and data storage efficiency. Experimental results demonstrate that leveraging the YUV color space, combined with data compression, significantly improves detection accuracy while minimizing storage requirements. This work highlights the importance of balancing data complexity, storage optimization, and detection reliability, laying a strong foundation for future advancements in defect detection, anomaly identification, and cost-efficient industrial automation.**

***Index Terms*—Visual Inspection, Machine Readable code Detection, Barcode, QRCode, Industrial Manufacturing, Uncontrolled Environments, Visual Anomaly Detection**

I. Introduction

Machine-readable codes have become widely adopted across various domains due to their numerous advantages, such as high-density encoding, the ability to store large amounts of information, low production costs, and ease of manufacturing [Nguyen, 2024]. Machine-readable codes refer to patterns or symbols that can be automatically recognized and interpreted by machines, facilitating rapid data retrieval and processing. The most common types are barcodes and QR codes. In industrial manufacturing systems, machine-readable codes are instrumental in enhancing traceability, streamlining inventory management, and improving quality control [Gazeau et al., 2024]. By embedding essential product information directly onto components or packaging, machine-readable codes enable seamless integration with automated systems, facilitating real-time tracking and ensuring operational efficiency in complex production environments. In industrial manufacturing, various defects can occur, such as blurred printing, stains, smudges, and misalignments [Zhao et al., 2024]. These degrade machine-readable code quality, reducing recognition rates and potentially causing misreading or failure. Such issues undermine functionality, especially in critical applications requiring accurate and timely data retrieval. Ensuring production quality requires detecting both the codes and their defects early, preventing failures before they impact critical applications. However, detecting machine-readable codes and their defects presents several challenges, primarily due to the limitations of datasets used for this task.

While high real-time detection performance is crucial, optimizing data size is equally essential for efficiency and reliability in industrial manufacturing systems. The vast amount of visual data generated in these environments, if not properly managed, increases storage demands, transmission latency, and computational overhead. These challenges hinder real-time processing and complicate deployment on resource-constrained hardware, such as embedded systems and edge devices. Reducing data size while maintaining real-time detection accuracy enables faster inference, lowers operational costs, and facilitates seamless integration into automated production workflows.

In this paper, we take the first step toward balancing real-time detection of machine-readable codes and their anomalies with data size reduction in industrial manufacturing.

The main contributions of this paper can be summarized as follows:

- We provide a comprehensive analysis of the challenges in detecting machine-readable codes and their anomalies in industrial environments, highlighting key limitations in existing approaches.
- We construct a challenging and diverse machine-readable codes dataset. This dataset will be publicly available.
- We conduct an in-depth study on data size reduction, applying various optimization strategies to minimize storage and computational costs while analyzing the trade-off between real-time detection performance and data size optimization.

The rest of this paper is organized as follows. Section 2 details the challenges and issues related to machine-readable code detection and classification. In Section 3, we review related work. Section 4 introduces our strategy for enhancing machine-readable code detection performance while optimizing data storage. Section 5 presents the experimental results, followed by a discussion in Section 6. Finally, Section 7 concludes the paper and outlines directions for future work.

II. RELATED WORK

Machine-readable code detection and analysis is a critical visual identification task within industrial surface inspection. Early detection algorithms generally relied on traditional machine learning techniques for region proposal [Chen et al., 2021]. These methods are effective for identifying machine-readable codes with distinct and easily recognizable characteristics. However, they are often slow and computationally expensive, particularly when dealing with complex backgrounds. With the recent advancement of deep learning, significant improvements have been achieved in various visual tasks, including object detection, and industrial defect segmentation [Bhatt et al., 2021]. Among various research fields, machine-readable code detection and classification is a key domain that has widely adopted deep learning, offering significant advantages over traditional approaches [Jia et al., 2020], [Yuan et al., 2019]. In complex scenes, machine-readable codes may appear blurred due to pixel-level distortions, distance variations, or other factors, and they may also undergo rotations and deformations. In such cases, deep learning-based detection methods have proven their effectiveness in addressing these critical challenges [Kamnardsiri et al., 2022]. In this matter, to face multiscale and multiview issues, Zhang *et al.* [Zhang et al., 2019] proposed a region-based end-to-end network to accurately localize and classify 1D barcodes and QR codes in complex environments. Their network includes two specialized layers: a quadrilateral regression layer for detecting arbitrary quadrilateral bounding boxes and a Multi-scale Spatial Pyramid Pooling (MSPP) layer to enhance the detection accuracy of small-scale barcodes. Peng et al. [Peng et al., 2020] integrated a Feature Pyramid Network module into Faster R-CNN to enhance the detection performance of small and multi-scale QR codes. Recently, given the growing importance of real-time machine-readable code detection in industrial applications and its increasing criticality in production lines, Chen *et al.* [Chen et al., 2023] proposed a rapid QR code detection approach based on multistage stepwise discrimination and a compressed MobileNet. Zhao *et al.* [Zhao et al., 2024] constructed their own QR code defect dataset and proposed an enhanced QR code detection and surface defect classification algorithm based on YOLOv8, referred to as YOLOv8-QR. Although these methods have demonstrated strong real-time QR code detection and classification performance, they have not focused on optimizing data size, making them less suitable for industrial real-time applications. However, data size reduction is crucial in such applications, as it optimizes storage, minimizes transmission latency, and enables efficient processing on resource-constrained hardware. Several methods for data size reduction have been proposed in the literature, but none have been applied to machine-readable code images. Several methods for data size reduction have been proposed in the literature [Yang and Mandt, 2023], [Chan et al., 2022]. However, none of these methods have been specifically applied to machine-readable code images.

In this paper, we study and select an efficient real-time detector to, unprecedentedly, analyse the impact of data size reduction on detection performance and explore the trade-off between detection accuracy and data size optimization.

III. CHALLENGES AND ISSUES

In industrial manufacturing systems, detecting machine-readable codes and classifying their defects present significant challenges that can be broadly categorized into two main groups: hardware constraints and data-related issues. Despite their importance, these challenges remain underexplored, contributing to the scarcity of public datasets that accurately capture these complexities.

A. Hardware constraints

One of the most pressing hardware-related challenges in industrial manufacturing systems is the issue of storage. The production process generates massive volumes of data, requiring advanced storage solutions capable of efficiently managing the size and complexity of this data. Additionally, the need for real-time detection imposes stringent requirements on data processing speed to ensure seamless integration with high-speed production lines.

B. Data-related issues

Data-related issues in machine-readable code detection and classification are various and complex. Among these, the most prominent is the notable lack of sufficient and diverse training data, which greatly hinders the development and performance of robust machine learning models. Moreover, data complexity presents a major challenge. Machine-readable codes are frequently embedded in visually noisy or complex environments. This clutter makes it challenging to isolate the machine-readable code from its surroundings, particularly in dynamic settings such as production lines. Furthermore, the diversity of machine-readable codes in terms of type, size, orientation, printing quality, and placement further complicates detection. In complex real-world industrial settings, environmental conditions introduce additional challenges. Variations in lighting can affect the visibility of machine-readable codes, while transparent backgrounds make detection more difficult when codes are printed on translucent or see-through materials. Shadows can obscure critical portions of the machine-readable code, and small defects, which, though seemingly minor, can still disrupt functionality. Such variability underscores the need for robust detection systems capable of ensuring reliable performance in industrial inspection applications. Moreover, machine-readable defect code detection and classification task faces additional challenges due to defect variability and scale. Common defects include blur, where the machine-readable code loses sharpness; offset, caused by the misalignment of machine-readable code elements; stains in the form of marks or spots; and peripheral stains, which are smudges or blemishes along the edges of the machine-readable code [Zhao et al., 2024]. However, unexpected defects can also occur, further complicating detection. Additionally, in industrial environments, anomalies are rare, resulting in highly imbalanced

datasets where the majority of instances are normal (negative) samples, while abnormal (positive) samples are either scarce or entirely absent. This imbalance complicates models training, as the models may struggle to effectively detect rare defects amidst the overwhelming number of normal instances. Moreover, in industrial visual inspection, high-resolution data capture is often used to detect even the smallest defects, which can be challenging to discern. These factors, combined with the rapid pace of industrial manufacturing processes, significantly increase the difficulty of accurately identifying and isolating machine-readable codes in real time.

IV. APPROACH DESCRIPTION

In this work, we study and select an efficient real-time detector to reliably detect machine-readable codes while optimizing data size. Our goal is to reduce storage and computational costs without compromising detection performance, which is crucial for industrial applications.

A. Convolution-based machine-readable code detection

Deep learning-based object detectors can be broadly classified into two main categories: multi-stage detectors and single-stage detectors. Single-stage detectors offer reduced computational time by directly mapping image pixels to bounding box coordinates and class probabilities, bypassing the intermediate stages used in multi-stage approaches. Among single-stage detectors, YOLO stands out as one of the most widely used due to its ability to achieve an excellent balance between speed and accuracy. Its robustness in detecting a diverse range of objects, including small ones, makes it particularly effective for various applications. Since its inception, the YOLO family has undergone multiple iterations, with each version addressing previous limitations and enhancing performance. YOLOv8 has demonstrated exceptional effectiveness in detecting small objects [Terven et al., 2023]. It employs an anchor-free architecture with a decoupled head, enabling the independent processing of objectness, classification, and regression tasks. This design allows each branch to specialize in its specific task, thereby improving the model's overall accuracy. The YOLOv8 architecture incorporates several key components:a modified CSPDarknet53 backbone, a cross-stage partial bottleneck with two convolutions (C2f module) that combines high-level features with contextual information to enhance detection accuracy, a spatial pyramid pooling fast (SPPF) layer to expedite computation by pooling features into a fixed-size map, and batch normalization with SiLU activation for each convolution operation.

B. Data size optimisation

In industrial applications, optimizing data size while preserving performance is crucial to address storage limitations and enhance processing efficiency. For machine-readable code detection, we applied three key techniques to reduce data size without compromising model accuracy: color space transformation, data type analysis, and data compression.

1) Color space transformation: Transforming the color space of images is a crucial step in optimizing dataset size and improving model efficiency [Yang et al., 2010], [Starosolski, 2014]. By eliminating redundant or unnecessary information, these transformations allow the focus to shift to the most critical features for machine-readable code detection. This approach was particularly beneficial in addressing storage constraints within industrial environments and adapting datasets for use in resource-limited systems. Images in the Red-Green-Blue (RGB) color space can be transformed into other color spaces to improve image processing, analysis, and reduce data size. In this paper, we focus on three main color spaces: YUV, grayscale, and binary.

- YUV color space: The YUV color space separates luminance (Y) from chrominance (U and V). Since machine-readable code detection relies more heavily on contrast and structure rather than color, the transformation from RGB to YUV reduces file size while retaining essential details. The conversion from RGB value to YUV is given by Equation 1.

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U &= -0.14713 \times R - 0.28886 \times G + 0.436 \times B \\ V &= 0.615 \times R - 0.51499 \times G - 0.10001 \times B \end{aligned} \quad (1)$$

- Grayscale color space: Transforming images to grayscale effectively reduces storage requirements by converting the three-channel RGB information into a single intensity channel. This simplified representation preserves the essential contrast patterns crucial for machine-readable code detection while eliminating irrelevant color data, thereby optimizing both storage efficiency and computational demands. The conversion from RGB value to grayscale is given by Equation 2, a weighted sum of the three color channels.

$$Grayscale = 0.2989 \times R + 0.587 \times G + 0.114 \times B \quad (2)$$

- Binary color space: In this transformation, grayscale images are thresholded to create binary images, where each pixel is represented as either black or white.

The data type of stored images was adjusted to refine the balance between storage efficiency and computational accuracy [Jayasankar et al., 2021]. The pixel data type was converted from the default uint8 (8-bit unsigned integer) to float16 (16-bit floating point) and float32 (32-bit floating point) formats, allowing for an enhanced representation of pixel intensity values. This conversion proved particularly useful in improving numerical precision during preprocessing and model training, especially for complex operations requiring higher accuracy.

2) Data compression: Data compression is essential for minimizing storage requirements while preserving the critical features necessary for accurate machine-readable code detection [Kaur and Choudhary, 2016]. Different compression

rates were applied to optimize storage while maintaining the quality required for detection tasks. The compression rate is calculated using the Equation 3. Thus, The more the image is compressed, the lower the compression rate becomes.

$$\text{Compression Rate (CR)} = \frac{\text{Size of Compressed Data}}{\text{Size of Original Data}} \quad (3)$$

V. EXPERIMENTS

To highlight the importance of the strategy outlined, this section offers a comprehensive overview of the results obtained during the experimentation phase.

A. Experimental methodology

1) Experimental setup: All experiments are conducted on a 64-bit computer with intel (R) CPU core (TM) 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 24 GB of RAM, and an NVIDIA GTX 1650 GPU with 6GB of VRAM. We use the Adam optimizer with a learning rate of 0.01, a batch size of 16, 400 epochs, and an image size of 320.

2) Data preparation: Due to the lack of a public challenging machine-readable code detection and classification dataset that captures real-world challenges, we had to collect our own dataset. On the other hand, the availability of public datasets designed for barcode and QR code detection [Souchet, 2023], as well as product recognition, facilitated our task. We selected two datasets [Kamnardsiri et al., 2022] from different domains (images captured from daily life consumer goods in supermarkets and images of parcels shot at post offices) to gather our high-resolution images. A total of 101 machine-readable code images with complex backgrounds were randomly selected from the InventBar dataset. Using the Roboflow platform, we manually annotated each image to create a labeled dataset suitable for training and evaluation. To further enhance the diversity of our dataset, we applied various data augmentation techniques, including transformations such as rotations, adding blur, and adjusting saturation and brightness. This process significantly increased the variability of the data, enabling the model to generalize better to unseen data. As a result, we obtained a dataset consisting of 243 images, which was split into two subsets: a training set and a validation set. For the testing set, we manually selected and annotated 25 images from the InventBar and ParcelBar datasets. Moreover, a total of 207 high-resolution images (1478x1108) with complex backgrounds were randomly selected from the two datasets to be pseudo-labeled.

In fact, pseudo-labeling is a semi-supervised learning technique that reduces the need for manual annotation while increasing the amount of labeled data available for training, ultimately enhancing model performance. By leveraging the model's own predictions on unlabeled data, this approach expands the training dataset, improving generalization to real-world variations. YOLOv8 offers five versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large). Among them, YOLOv8n is the most suitable for production lines due to its speed, with a minimal number of parameters (3.2 million) and FLOPs (8.7). Although its detection performance is lower (achieving the smallest mAP on the COCO dataset), its computational efficiency makes it ideal for real-time applications in industrial settings, where speed and low resource consumption are critical. Manually annotated images are used to train the YOLOv8n model. Once trained using the annotated data, the unlabeled images are fed into the trained model for testing, generating pseudo-labels. The significant difference between the training and unlabeled data, as well as the complexity of the unlabeled images, led to many false and missed detections. As a result, a manual verification step was required to review the detections, and images with false detections were removed. This process resulted in a refined subset of 130 retained pseudo-labeled images. Finally, these selected images, along with the initially annotated ones, were used to retrain the YOLOv8n model. Using the test set, we evaluate the effectiveness of the model before and after pseudo-labeling. The obtained results are reported in Table I. When comparing the results before and after pseudo-labeling, we observe an improvement in terms of mAP 50-95, precision and Average F1 score, highlighting the importance of retraining to address data fracture and to enhance the model's ability to generalize.

TABLE I
YOLOV8N PERFORMANCES BEFORE AND AFTER PSEUDO-LABELING PROCESS.

Metric	Before pseudo-labeling	After pseudo-labeling
mAP 50-95	73.87%	**83.35%**
Precision	96.98%	97.49%
Recall	100%	100%
F1 score	0.98	**0.99**

B. Data size optimisation

To evaluate the efficiency of machine-readable code detection while minimizing storage requirements, we conducted a comprehensive study on data size optimization. The goal was to achieve a balance between detection performance and data storage needs. Our investigation focused on three key aspects: color space transformation, data type modification, and data compression.

1) Color space transformation: We examined the impact of converting RGB images into various color spaces, including YUV, grayscale, and binary formats, to determine the most storage-efficient representation while preserving detection accuracy. Unexpectedly, converting our data from RGB to grayscale and binary while retaining the same data type led to an increase in dataset size than a reduction, which was not the case for data containing fewer details. On the other hand, a significant reduction in size of around 15.7% was recorded for the YUV color space. Additionally, machine-readable code detection performance in terms of precision improved, while the mAP slightly decreased for the YUV and binary color spaces. Among the tested color spaces, the YUV format achieved the best balance between detection accuracy and storage reduction. This result underscores the importance of

preserving essential color information for effective machine-readable code detection.

2) Data type modification: Since the YUV color space yielded the best results, we conducted further analysis on the impact of data types using this color space. Initially, the data was encoded in the uint8 format, resulting in a dataset size of 64.7 MB. We then experimented with float16 and float32 encoding, which drastically increased the dataset size to 2.84 GB and 5.68 GB, respectively, due to their higher memory requirements per pixel. Consequently, the uint8 data type proved to be the optimal choice.

3) Data compression: We began by compressing the raw RGB data using various compression rates (85, 70, 50, 25, 15, and 10). Next, since the YUV color space demonstrated the best balance between detection accuracy and storage efficiency, we aimed to further reduce the size of the YUV data. To achieve this, we applied chroma subsampling along with compression techniques using the same rates previously applied to the RGB data. Table II and Table III present an evaluation of the image quality after compression in terms of SSIM (Structural Similarity Index), PSNR (Peak Signal-to-Noise Ratio), and MSE (Mean Squared Error) for RGB and YUV images, respectively. According to these tables, high image quality is maintained after compression, especially with a significant reduction in size. The compression of YUV images achieved excellent quality, while the compression of RGB images resulted in good quality, though slightly lower than that of YUV images.

- RGB data compression: As depicted in Figure 1, a significant reduction in data size was achieved, decreasing from 76.7 MB to 15.9 MB at a compression rate of 10. Moreover, an overall improvement in detection performance is noticed. A slight degradation in precision was observed for low compression rates as excessive compression introduces visual artifacts, such as the loss of essential details or distortions, which interfere with key visual features. These artifacts adversely affect the model's detection performance by degrading the quality of critical information. For RGB images, although most codes are detected, the model produces a significant number of false positives.

CR	**85**	**70**	**0**	**25**	**15**	**10**
SSIM ↑	0.996	0.994	0.95	0.909	0.875	0.846
PSNR ↑	45.9	43.73	34.18	31.69	29.84	18.08
MSE ↓	1.84	2.92	27.31	47.58	71.67	105.7

TABLE II
EVALUATION OF RGB IMAGE QUALITY AFTER COMPRESSION.

- YUV data compression: In addition to compression, we employed chroma subsampling, commonly used in image compression and processing, involves formats like YUV 4:2:2 (where the U and V components are horizontally subsampled, halving the chrominance data) and YUV 4:2:0 (where the U and V components are subsampled both horizontally and vertically, further reducing chrominance data). This process involved converting RGB images to the YUV color space, followed by transformations to YUV 4:2:2 and YUV 4:2:0 formats. As shown in Figure 2, while chroma subsampling effectively reduces data size, it results in an overall degradation in detection performance. In addition, a significant reduction in data size is observed, along with an overall improvement in detection performance. Using the same high compression rates, a greater reduction in size and a more substantial enhancement in detection performance are achieved compared to the RGB modality. However, for very low compression rates (e.g., 15 and 10), a significant drop in performance is recorded due to the poor image quality after compression. Moreover, many codes are missed.

CR	**85**	**70**	**0**	**25**	**15**	**10**	**YUV 4:2:2**	**YUV 4:2:0**
SSIM ↑	0.986	0.981	0.975	0.96	0.947	0.933	0.984	0.977
PSNR ↑	47.9	46.2	44.76	42.1	39.58	37.23	47.24	45.61
MSE ↓	1.132	1.655	2.278	4.12	7.28	12.7	1.346	1.93

TABLE III
EVALUATION OF YUV IMAGE QUALITY AFTER COMPRESSION.

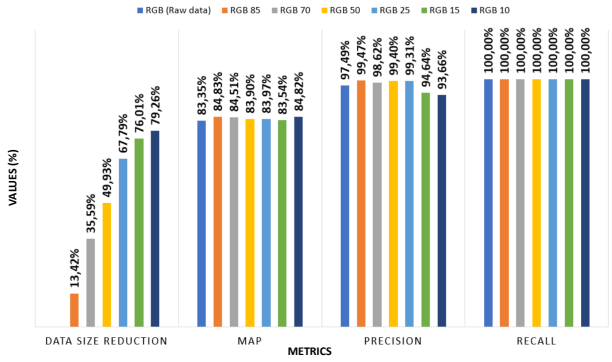

Fig. 1. Machine-readable code detection performance and data size analysis for RGB data types at varying compression rates.

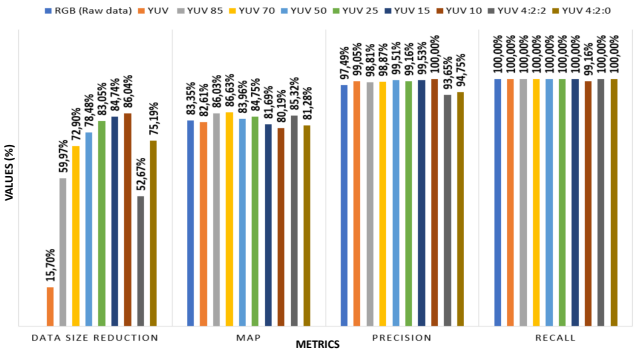

Fig. 2. Machine-readable code detection performance and data size analysis for YUV data types at varying compression rates.

VI. DISCUSSION AND FUTURE DIRECTIONS

Detection performance using YUV data after compression consistently surpasses the baseline achieved with RGB data or even uncompressed YUV data. This is because the Y component (luminance) in the YUV format contains the critical structural information, such as edges and contours, which are key for machine-readable code detection. Even after compression, these essential features are preserved in the YUV space. The separation of luminance and chrominance information in YUV allows it to better retain critical details under compression compared to RGB, where all information is mixed. As a result, detection performance with YUV remains relatively stable, or even improves, at fairly low compression rates. Despite using the same compression rates, YUV datasets consistently have smaller sizes compared to RGB datasets. However, while RGB compressed data ensures that all machine-readable codes are detected, YUV compressed data sometimes leads to missed codes and false detections. After applying chroma subsampling, while the luminance (Y) component remains unchanged, the reduction in resolution of the U and V components decrease color accuracy. Since color information is particularly important for machine-readable code detection, this results in degraded detection performance. In summary, the experiments demonstrated effective control over file size while maintaining high detection performance, particularly with YUV data under compression. It is noteworthy that the test images used in this study were particularly challenging. These images included cluttered backgrounds and text in the Thai language that visually resembled barcodes. Additionally, the test images contain multiple machine-readable codes per image. This paper presents a foundational step toward detecting and classifying machine-readable code in industrial manufacturing systems, addressing several real-world challenges. However, significant efforts are still needed to overcome the lack of training data for defective cases. One key area for improvement is data augmentation, where generating synthetic defect patterns could help diversify the dataset and improve the model's robustness to various defect scenarios. Such data is rare and difficult to collect. Therefore, it often needs to be synthesized. Additionally, the development of lightweight and novel frameworks for defect detection and classification could significantly enhance the system's performance and reliability, making it more applicable in real-world industrial settings.

VII. CONCLUSION

In conclusion, this paper provides a critical first step toward detecting and classifying machine-readable code in industrial manufacturing systems by addressing key challenges. Our experimentation demonstrated that the YUV color space, when combined with data compression techniques, outperformed traditional RGB representation by achieving an optimal balance between storage efficiency and detection accuracy. Despite these advancements, further research is needed to tackle the complexities of real-world industrial environments fully. The insights and methods presented in this work lay a strong foundation for future developments, advancing the capabilities of automated visual inspection systems in industrial manufacturing.

REFERENCES

[Bhatt et al., 2021] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., and Gupta, S. K. (2021). Image-based surface defect detection using deep learning: A review. *Journal of Computing and Information Science in Engineering*, 21(4):040801.

[Chan et al., 2022] Chan, K. H. R., Yu, Y., You, C., Qi, H., Wright, J., and Ma, Y. (2022). Redunet: A white-box deep network from the principle of maximizing rate reduction. *Journal of machine learning research*, 23(114):1–103.

[Chen et al., 2023] Chen, R., Huang, H., Yu, Y., Ren, J., Wang, P., Zhao, H., and Lu, X. (2023). Rapid detection of multi-qr codes based on multistage stepwise discrimination and a compressed mobilenet. *IEEE Internet of Things Journal*, 10(18):15966–15979.

[Chen et al., 2021] Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z., and Shao, L. (2021). Surface defect detection methods for industrial products: A review. *Applied Sciences*, 11(16):7657.

[Gazeau et al., 2024] Gazeau, B., Zaman, A., Minunno, R., and Shaikh, F. (2024). Developing traceability systems for effective circular economy of plastic: A systematic review and meta-analysis. *Sustainability*, 16(22):9973.

[Jayasankar et al., 2021] Jayasankar, U., Thirumal, V., and Ponnurangam, D. (2021). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University - Computer and Information Sciences*, 33(2):119–140.

[Jia et al., 2020] Jia, J., Zhai, G., Ren, P., Zhang, J., Gao, Z., Min, X., and Yang, X. (2020). Tiny-bdn: An efficient and compact barcode detection network. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):688–699.

[Kamnardsiri et al., 2022] Kamnardsiri, T., Charoenkwan, P., Malang, C., and Wudhikarn, R. (2022). 1d barcode detection: Novel benchmark datasets and comprehensive comparison of deep convolutional neural network approaches. *Sensors*, 22(22):8788.

[Kaur and Choudhary, 2016] Kaur, R. and Choudhary, P. (2016). A review of image compression techniques. *Int. J. Comput. Appl*, 142(1):8–11.

[Nguyen, 2024] Nguyen, D. (2024). Convenient efficiency: A media genealogy of qr codes. *New Media & Society*, 26(10):5742–5762.

[Peng et al., 2020] Peng, J., Yuan, S., and Yuan, X. (2020). Qr code detection with faster-rcnn based on fpn. In Sun, X., Wang, J., and Bertino, E., editors, *Artificial Intelligence and Security*, pages 434–443, Cham. Springer International Publishing.

[Souchet, 2023] Souchet, B. (2023). List of barcode qr code datasets. Last accessed 15/12/2024.

[Starosolski, 2014] Starosolski, R. (2014). New simple and efficient color space transformations for lossless image compression. *Journal of Visual Communication and Image Representation*, 25(5):1056–1063.

[Terven et al., 2023] Terven, J., Córdova-Esparza, D.-M., and Romero-González, J.-A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716.

[Yang et al., 2010] Yang, J., Liu, C., and Zhang, L. (2010). Color space normalization: Enhancing the discriminating power of color spaces for face recognition. *Pattern Recognition*, 43(4):1454–1466.

[Yang and Mandt, 2023] Yang, Y. and Mandt, S. (2023). Computationally-efficient neural image compression with shallow decoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 530–540.

[Yuan et al., 2019] Yuan, B., Li, Y., Jiang, F., Xu, X., Guo, Y., Zhao, J., Zhang, D., Guo, J., and Shen, X. (2019). Mu r-cnn: A two-dimensional code instance segmentation network based on deep learning. *Future Internet*, 11(9):197.

[Zhang et al., 2019] Zhang, J., Jia, J., Zhu, Z., Min, X., Zhai, G., and Zhang, X.-P. (2019). Fine detection and classification of multi-class barcode in complex environments. In *2019 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 306–311.

[Zhao et al., 2024] Zhao, L., Liu, J., Ren, Y., Lin, C., Liu, J., Abbas, Z., Islam, M. S., and Xiao, G. (2024). Yolov8-qr: An improved yolov8 model via attention mechanism for object detection of qr code defects. *Computers and Electrical Engineering*, 118:109376.