Synthetic Data Generation for Robust Optical Character Verification in Industrial Laser Marking

Júlia Garcia Torné

June 30, 2025

Abstract

Optical Character Verification (OCV) is critical for ensuring traceability in industrial laser marking, yet real-world datasets often suffer from limited diversity, imbalanced character distributions, and insufficient defect examples. This work proposes a synthetic data generation pipeline leveraging diffusion models to overcome these limitations. A ControlNet architecture, fine-tuned on Canny edge maps from real industrial packaging images, generates high-fidelity synthetic samples with varied characters, scales, backgrounds, and simulated defects (e.g., misplacements, beam distortions). The synthetic dataset addresses key gaps in the original data, including underrepresented characters and missing error types. To demonstrate feasibility, a preliminary YOLOv11-based approach is implemented, confirming the utility of synthetic data for training robust OCV systems. The results highlight the potential of generative models to bypass costly real-world data collection while maintaining industrial applicability.

Keywords: Synthetic Data Generation, Optical Character Verification (OCV), Industrial Defect Detection, Stable Diffusion, ControlNet, YOLOv11, Laser Marking, Dataset Augmentation

1 Introduction

In modern industrial environments, especially in the food and pharmaceutical sectors, product traceability is essential for ensuring safety, compliance, and quality control. This traceability relies on the accurate and legible printing of critical information such as expiration dates, batch numbers, and unique identifiers. These codes are typically applied using high-speed laser printing systems directly onto packaging materials during production.

To verify these markings, automated Optical Character Verification (OCV) systems are employed. However, these systems often struggle with real-world variability in lighting, surface textures, scale, and positioning. Such inconsistencies can compromise the reliability of printed code recognition, leading to potential misreads or failures in the verification pipeline.

Developing robust machine learning models capable of accurately reading and validating printed codes in challenging industrial environments is of critical importance. However, achieving high performance in such tasks necessitates access to large, diverse, and thoroughly annotated datasets. The process of collecting and annotating real-world industrial data poses several challenges, including being time-consuming, labor-intensive, and constrained by various limitations. These include imbalanced character distributions due to repetitive industrial code patterns, limited diversity in product backgrounds and printing conditions, and the absence of fully automated annotation tools. Consequently, the creation of a high-quality dataset is a fundamental prerequisite for training reliable and generalizable models.

2 Problem Statement

It is essential to identify possible errors in the laser marking process and study the available dataset to uncover its key limitations. This understanding helps guide the creation of more diverse and balanced data, which is crucial for training reliable Optical Character Verification models.

Error Types

Several types of errors may occur during the laser marking process. The following have been identified:

- **Invalid Placement:** The mark is located outside the designated clear zone reserved for code placement.

• Contact E-mail: david.castells@uab.cat
• Supervised by: David Castells Rufas
• Academic Year 2024/25

- **Bad Surface:** Wrinkles, reflections, or unexpected foreign elements on paper or plastic packaging surfaces interfere with the laser marking process, causing degradation in mark quality.
- **Laser Beam Defect:** Insufficient or excessive laser power, faulty optics, or focus issues lead to blurred or underexposed marks due to the laser beam's Gaussian distribution.
- **Laser Deflection Defect:** Errors in the galvanometers or their controllers result in incorrect beam positioning.
- **Unmatching Code:** The printed code does not match the expected code in the image, indicating logical inconsistencies in the data transfer process.
- **OCV System Malfunction:** The Optical Character Verification system fails to validate properly marked codes due to internal errors, representing a failure of the inspection system outside the scope of this work.

Dataset Description

The dataset consists of 1,696 annotated real-world images, obtained in 2023 from multiple production lines in the food and pharmaceutical industries. These samples were captured using inline imaging systems within actual packaging environments.

To facilitate efficient annotation, a semiautomatic tool was developed. This tool uses template matching based on the known font used by the printers to detect character locations but requires manual correction. It proposes annotations for each image, which are then validated and corrected by human annotators. The annotations include the marking region (bounding box), code scale (font size), error type label for each image, and the content and location of each character within the image. Overall, the dataset contains 51,189 annotated characters. However, the character distribution is highly imbalanced, with numeric digits—especially '0' and '2'—being overrepresented, while many letters and symbols are completely absent.

In addition to character imbalance, the dataset suffers from limited diversity. It includes only four distinct background types. Although font scales vary, their distribution is narrow. Moreover, not all relevant error types are represented; only "Invalid Placement" has labeled examples, as shown in Table 1.

TABLE 1: Frequency of errors on the real dataset.

Type	Number of Samples	Frequency
Valid	1645	96.99%
Invalid Placement	51	3.01%
Bad Surface	0	0%
Laser Beam Defect	0	0%
Laser Deflection Defect	0	0%
Unmatching Code	0	0%
TOTAL	1696	100%

Objective

The primary objective of this work is to develop a method for generating a variable and diverse dataset that addresses the limitations of the current real-world dataset, enabling effective training of an Optical Character Verification (OCV) model capable of recognizing every character defined by the provided font, NewPal, and detecting defects in printed codes. Given the scarcity and imbalance of error types and character representation in the existing data, the focus is on creating a comprehensive synthetic data generation pipeline to ensure diversity in printed codes, image backgrounds, and error cases. This will support robust code localization and recognition, including error detection, as preliminary steps toward a complete OCV solution. Deployment to edge devices and runtime optimization are beyond the scope of this work.

3 State of the Art

In contemporary industrial environments, Optical Character Verification (OCV) has evolved beyond traditional, rule-based Optical Character Recognition (OCR) systems [1]. The adoption of deep learning methodologies—particularly those based on Transformer architectures—has significantly enhanced the robustness of OCV, enabling reliable interpretation of faint, distorted, or misaligned characters under challenging conditions [2]. Modern solutions increasingly integrate convolutional and recurrent neural network hybrids, thereby improving spatio-temporal reasoning and adaptability to complex packaging geometries [3].

In the domain of character detection, real-time object detection models—most notably the successive iterations of the YOLO (You Only Look Once) family—have emerged as pivotal tools in high-speed, inline quality assurance pipelines [4]. YOLOv7 and its more recent variants offer exceptional frame rates and fine-grained handling of multi-character regions, making them particularly well-suited for high-throughput sectors such as pharmaceuticals and food logistics [5].

A critical shift in the advancement of these systems lies in the approach to data acquisition and training. Industrial scenarios often present chaotic backgrounds, limited annotated data, and infrequent defects—challenges that render traditional data collection both expensive and insufficient [6]. In response, generative techniques have gained prominence, not merely as augmentation tools but as foundational components of training pipelines. Early methods, such as Defect-GAN, enabled the simulation of anomalies [7]; however, the introduction of diffusion models has significantly elevated synthetic data quality [8]. These models learn the underlying distributions of defect-free samples, enabling both the inpainting of missing elements and the insertion of realistic defects with high visual fidelity [9].

Tools such as Stable Diffusion—augmented by frameworks like ControlNet and IP-Adapter—facilitate precise control over synthetic text placement and surface curvature, thereby replicating real-world conditions with greater

accuracy [10, 11]. This includes the simulation of reflective surfaces, irregular geometries, and printer-induced distortions across a wide range of substrates, from aluminum canisters to blister packaging [12].

This data-centric shift marks a new frontier in OCV development: systems are now trained not only on empirical data but also on richly imagined, highly controlled synthetic scenarios—narrowing the domain gap between laboratory research and real-world deployment [13].

4 METHODOLOGY

The project comprises two phases: the first involves generating a balanced dataset, while the second develops a preliminary solution to demonstrate the effectiveness of data augmentation in enhancing model generalization.

4.1 Data Generation

To mitigate the dataset imbalance, a diffusion-based image generation strategy was adopted. Stable Diffusion XL (SDXL) was chosen due to its effective balance between high-quality image output and diversity [14]. Since generation speed was not a major constraint, SDXL's slower inference time was acceptable in favor of better visual results.

To condition the generation on structural information, ControlNet was integrated with SDXL. This architecture is well-suited for scenarios with limited training data. Specifically, ControlNet-Canny was used to guide the generative model based on edge features.

Training Dataset

ControlNet was fine-tuned exclusively using real-world data. As illustrated in Figure 1, a custom dataset was constructed by pairing each real image sample with its corresponding Canny edge map [15], which served as the conditioning input. The original images were designated as target outputs during training. This structured input-output pairing facilitated the model's effective adaptation to the specific task. The dataset comprised 1,696 samples in total — of which 2 were reserved for validation, with the remaining used for training.

Training Setup

Fine-tuning was conducted using the pre-trained SDXL base model (*stabilityai/stable-diffusion-xl-base-1.0*) [14] alongside the ControlNet Canny variant (*diffusers/controlnet-canny-sdxl-1.0*) [16]. A pre-trained VAE (*madebyollin/sdxl-vae-fp16-fix*) was also employed [17].

Training was performed on an NVIDIA GeForce RTX 3090 GPU (24 GB VRAM) using the Accelerate library [18]. Mixed-precision (fp16) training, memory-efficient attention (via xFormers), and gradient checkpointing were enabled to optimize memory usage and maintain output quality. The input consisted of 800×800 resolution image pairs (edge map + grayscale image). Training lasted for 10,000 steps with a batch size of 1 and gradient accumulation over 4 steps. A low learning rate of 5×10^{-7} was used, scheduled with cosine learning rate restarts.

Validation was carried out every 100 steps using two fixed samples and consistent text prompts describing the scenes. Training progress and model performance were monitored using Weights & Biases (wandb) [19].

Fig. 1: Samples used for fine-tuning ControlNet. Left: Canny edge maps (conditioning inputs). Right: corresponding original images (targets).

Sampling Pipeline

With the trained model, new data can be generated through a structured sampling process that incorporates multiple inputs to guide the creation of synthetic images. First, a reference image—sourced either from the real dataset or external sources such as the internet—is used to extract the background structure via edge detection. The resulting edge map, denoted as e_1, enables the model to preserve and reconstruct the layout of the background.

Next, the code to be printed is specified, along with its desired position and scale within the image. This information is rendered visually and processed to generate a second edge map, e_2, which represents the structure and placement of the text.

The two edge maps, e_1 and e_2, are then merged using a bitwise OR operation to produce a single conditioning image c_1, which is passed to the ControlNet module. This allows the model to follow both the background structure defined by e_1 and the text layout provided by e_2. An optional text prompt can also be supplied to further guide the image generation. The full process is visually represented in Figure 2.

4.1.1 Dataset Augmentation Strategies

This section presents methods for using the trained model to generate synthetic samples that enhance dataset balance

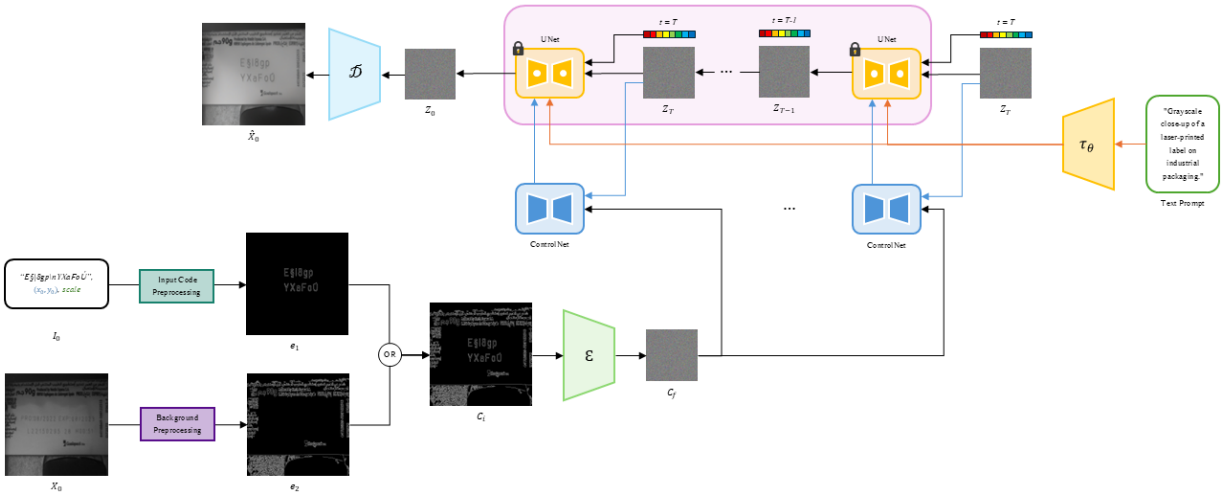

Fig. 2: Pipeline for generating synthetic images using Stable Diffusion XL with ControlNet. The background edge map e_1 is extracted from a reference image, and the code to be printed is rendered and processed into e_2. The combination of both, $c_i = e_1 \vee e_2$, is used as the conditioning input for ControlNet, optionally accompanied by a text prompt.

and diversity. In each scenario, the conditioning edge map is carefully adapted to guide the model toward producing the desired visual outcome.

Underrepresented Characters

A major limitation of the original dataset is the absence or underrepresentation of many font characters. This limits the system's ability to generalize across the full character set. To address this, edge maps can be created using the font's vector representations to include missing characters.

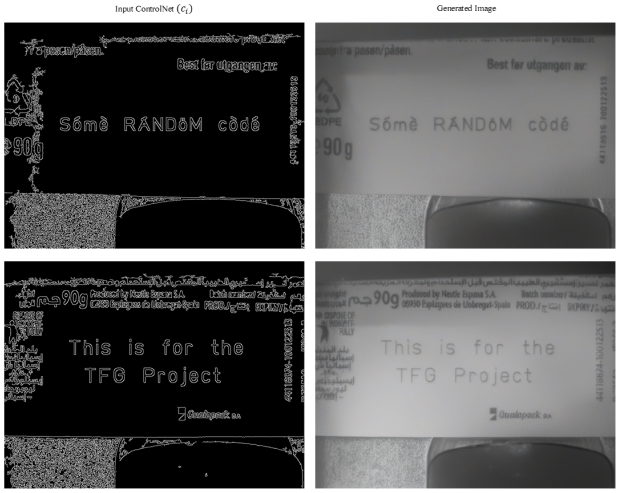

Fig. 3: Examples of paired edge maps and generated images containing characters absent from the original dataset.

These maps guide the model to generate new samples that maintain the original font style while enriching the dataset with a complete and diverse set of characters.

Scale Diversity

The dataset features only a limited variety of text sizes, restricting the system's ability to adapt to scale variations. This can hinder performance in practical settings where text size changes. To improve this, edge maps can be resized to show the code at various scales. The generative model then generates outputs that reflect these scale changes while preserving the original background structure and style.

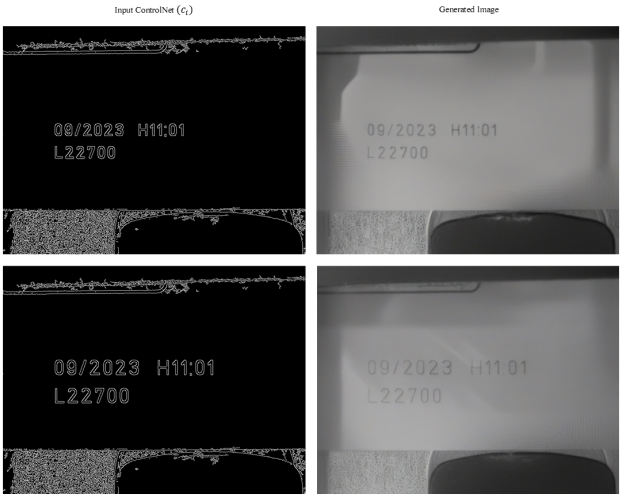

Fig. 4: Examples of paired edge maps and generated images showing the same background structure and code, but with different code scales: 0.8 (top) and 1.0 (bottom).

Background Diversity

The dataset includes only four types of backgrounds, which limits the system's exposure to layout variation. Although the generative model can remix and reuse learned background features, it tends to replicate familiar patterns. To address this, edge maps can be creatively modified or extracted from real-world packaging images to introduce new structural backgrounds. This encourages the model to produce samples with visually distinct layouts while remaining stylistically coherent.

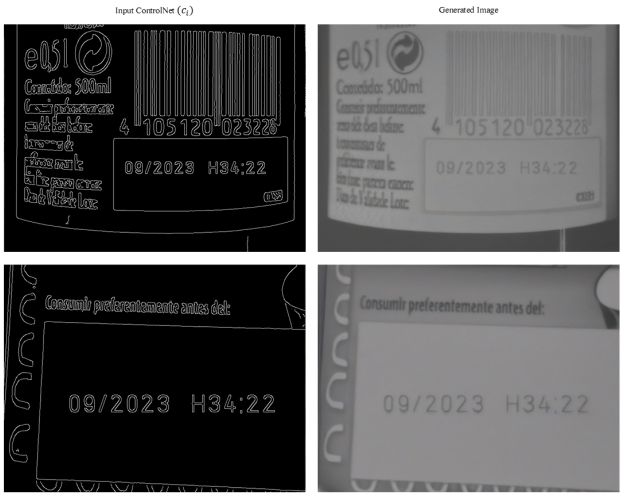

Fig. 5: Examples of paired edge maps and generated images showing completely new backgrounds with a consistent visual style.

Error: Invalid Placement

Only a few samples in the original dataset show character codes placed incorrectly outside the marking zone. This limits the system's understanding of placement errors. To simulate these cases, the code can be deliberately placed outside the valid area within the edge maps. The generated images are guided by these modified maps and accurately reflect the incorrect placement.

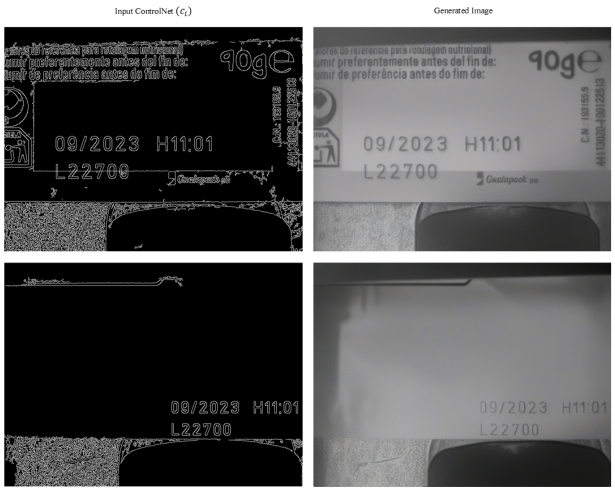

Fig. 6: Examples of paired edge maps and generated image where the code is placed outside the marking region.

Error: Bad Surface

Defective surfaces are not naturally present in the dataset, yet are important for robust system training. To simulate these, visual imperfections such as cracks, noise, or scratches can be added directly to the edge maps. The model can then generate outputs that realistically reflect damaged or degraded surfaces.

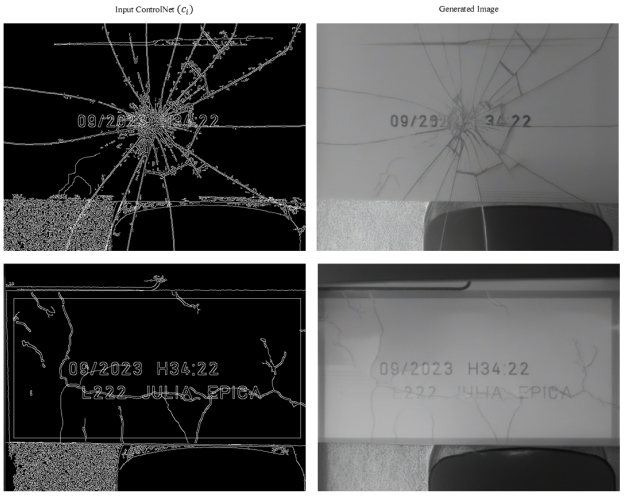

Fig. 7: Examples of paired edge maps and generated images showing different possible surface defects.

Error: Laser Beam Defect

This type of defect is not represented in the original dataset. It can be simulated by manipulating the contour width of the characters in the edge maps — making them slightly thicker or thinner. This mimics the visual effect of insufficient or excessive laser power, faulty optics, or focus issues, which typically result in blurred, overexposed, or underexposed marks due to the laser beam's Gaussian distribution.

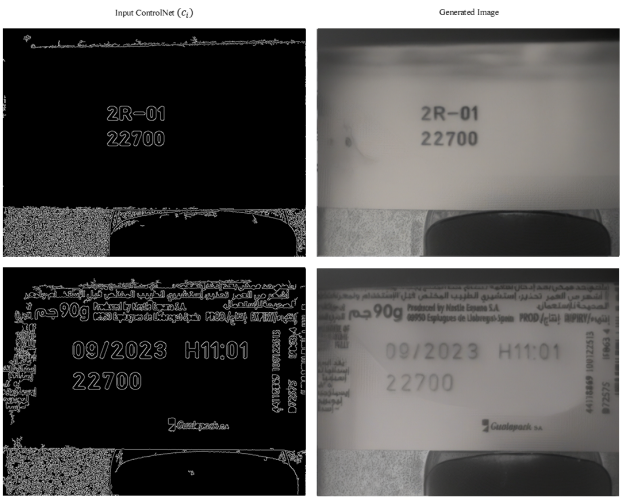

Fig. 8: Examples of paired edge maps and generated images showing beam-related defects through varying contour thickness.

Error: Laser Deflection Defect

This error is also not found in the dataset but can be simulated by duplicating the character code in the edge map — one in the correct position and a slightly shifted version nearby. This reflects errors caused by faulty galvanometers or control systems, which lead to incorrect laser beam positioning.

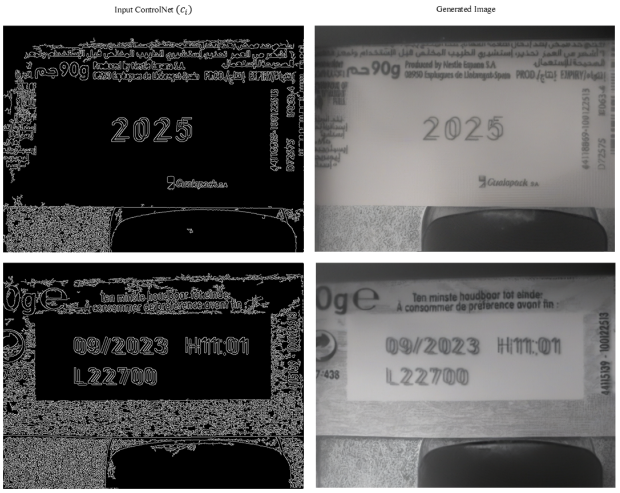

Fig. 9: Examples of paired edge maps and generated images showing displaced character codes simulating deflection errors.

Error: Unmatched Code

The original dataset does not visually reflect cases where printed codes and their corresponding annotations do not match, even though such errors can occur in real production environments. To simulate this scenario, the image remains unchanged while the annotation is deliberately altered to represent an incorrect character. This introduces a logical inconsistency between the visual content and its metadata.

4.1.2 Synthetic Dataset

To support the initial stage of the second part of the project — training the recognizer and locator model — the dataset was augmented using the previously described strategies. A total of 5,000 synthetic samples were generated to ensure comprehensive character coverage across the entire font set. These samples were produced by randomly composing codes containing 1–3 lines and 4–10 characters per line, with varied text scales ranging from 0.5 to 1.5. Out of these, 2,500 samples were intentionally created with errors involving code placement outside the valid marking region. The remaining 2,500 were valid samples, designed to improve general model robustness and diversity.

In addition, 100 samples for each of the other simulated error types — including bad surface, laser beam defect, laser deflection, and unmatched code — were generated using their respective augmentation strategies. These error-specific samples were used to validate the error detection pipeline, which is detailed in the following sections.

4.2 Code Location and Recognition for Error Detection

Once data generation is complete, the pipeline moves to the second stage, which focuses on detection, recognition, and error identification — ultimately the core objective of the quality control system.

4.2.1 Code Location and Recognition

YOLOv11 was selected for this task due to its high accuracy, rapid inference speed, and strong performance in dense object detection scenarios [4, 5]. Given the specificity of the task — detecting fine-grained character details and structured regions — fine-tuning the model was necessary to adapt it effectively to this application.

Dataset Preparation

The dataset used in this work consisted of both synthetic and real-world images. All data was formatted according to the YOLO object detection framework. In each image, every character within the printed code was annotated individually using bounding boxes, with each character treated as a separate class. Altogether, the model was trained on 174 unique classes: 173 representing individual characters from the target font set, and one additional class for the laser marking region. This setup allows the model to precisely detect both the characters and their specific location within the image. For training purposes, 85% of the data was used for training, while the remaining 15% was allocated for validation — ensuring that both training and validation sets included a representative mix of data.

Training Setup

The YOLOv11 model was trained using a combination of real and synthetic data. Training was conducted over 500 epochs with a batch size of 16. Early stopping was configured with a patience of 30 epochs to prevent overfitting and reduce unnecessary computation.

The learning rate was initialized at 0.003 and decayed to 0.001 throughout training. To improve generalization, a dropout rate of 0.3 was applied. A warm-up phase of 5 epochs was included to stabilize the model at the beginning of training.

Extensive data augmentation techniques were utilized to improve robustness, including hue-saturation-value (HSV) transformations, translation, scaling, horizontal flipping, and mixup [20]. Additional augmentation strategies included randaugment to introduce diverse perturbations across training samples.

Advanced training techniques were also employed. Multi-scale training was activated to enhance detection across varying image resolutions. Mosaic augmentation was used to increase context variation in each batch, and

data caching was enabled to optimize loading performance. Training progress was monitored and visualized through real-time plotting.

4.2.2 Error detection

Error detection remains an open challenge within this project and is currently the focus of ongoing research. However, considering the constraints of deployment on edge devices — where speed and efficiency are essential — a streamlined and lightweight pipeline is proposed. This approach uses the fine-tuned YOLOv11 model, which is capable of detecting both individual characters and the laser marking region in a single inference pass. This unified detection process offers a simple yet effective solution for real-time error detection in production environments.

The error detection logic is defined by the following criteria:

- **Valid Case:** The detected character sequence perfectly matches the expected code, and all characters lie within the bounding box of the marking region.
- **Misplaced Code:** One or more detected characters fall outside the boundaries of the designated marking region.
- **Incorrect Characters:** The detected character sequence does not match the expected code, with high confidence scores from the model.
- **Surface Defects:** Any discrepancy between the detected and expected codes, accompanied by low confidence scores, may indicate potential surface damage or degradation.

This approach represents an initial step, effectively handling the most common errors. While it is functional for current needs, it does not yet cover all possible error types and will be refined further as research progresses.

5 Experiments and Results

5.1 Diffusion-Based Approaches Comparison

A series of diffusion-based methods were explored to determine the most effective strategy for image generation under the constraints of limited data availability and a need for high efficiency.

Initially, the pretrained Stable Diffusion XL (SDXL) model was employed in a text-to-image setting. Although the generated outputs did not closely resemble the dataset used in this work, this step provided useful insight into the model's default interpretation of the input prompt: "grayscale close-up of a laser-printed label on industrial packaging." This baseline highlighted the limitations of relying solely on generic pretrained models when targeting highly specific visual styles or structural content.

The second approach introduced the use of ControlNet with edge maps as structural guidance. In this setup, synthetic edge maps were created to mimic the layout of actual labels. The pretrained ControlNet was applied to enforce this structure while generating images. Although this approach resulted in a notable improvement in structural consistency, the overall visual fidelity and style alignment with real data remained suboptimal.

To address style similarity, a LoRA (Low-Rank Adaptation) module was trained using the real images from the original dataset [21]. This module was then integrated with the text-to-image pipeline, in combination with the ControlNet. This configuration achieved a better balance between structure and visual style. However, the results were still not fully satisfactory.

Subsequently, the LoRA component was replaced with a pretrained IP-Adapter, which enables image-based style conditioning. The IP-Adapter received a reference image from the dataset, while the ControlNet continued to guide the structural layout using edge maps. This combination yielded further improvements, particularly in terms of style reproduction. Nevertheless, some discrepancies persisted between the generated and target images.

To further enhance the quality, the ControlNet component was fine-tuned on the specific dataset described in previous sections. This adjustment significantly improved both the structural accuracy and visual resemblance of the generated outputs to the real images. The fine-tuned model exhibited the most promising results, balancing both the semantic content and stylistic fidelity.

A visual comparison of the different diffusion-based methods is provided in Figure 10, illustrating the progression in quality across the various experimental configurations.

5.2 Generation Results

This section presents both qualitative and quantitative results obtained from the image generation methodology. The experiments demonstrate the effectiveness of the generative approach in producing visually compelling and stylistically coherent outputs.

5.2.1 Qualitative Results

Figure 11 illustrates a selection of images generated using the generative method. The generated outputs exhibit high fidelity, stylistic consistency, and semantic coherence. These qualitative examples underscore the strength of the ControlNet-based pipeline, which provides substantial control over the generation process. Notably, the architecture allows precise conditioning while preserving visual quality across varied prompts.

5.2.2 Quantitative Results

Image generation was executed on an NVIDIA GeForce RTX 3090 GPU, yielding an average generation time of approximately 5.06 seconds per image. Although this duration is relatively high, it represents a trade-off that yields notably superior image quality.

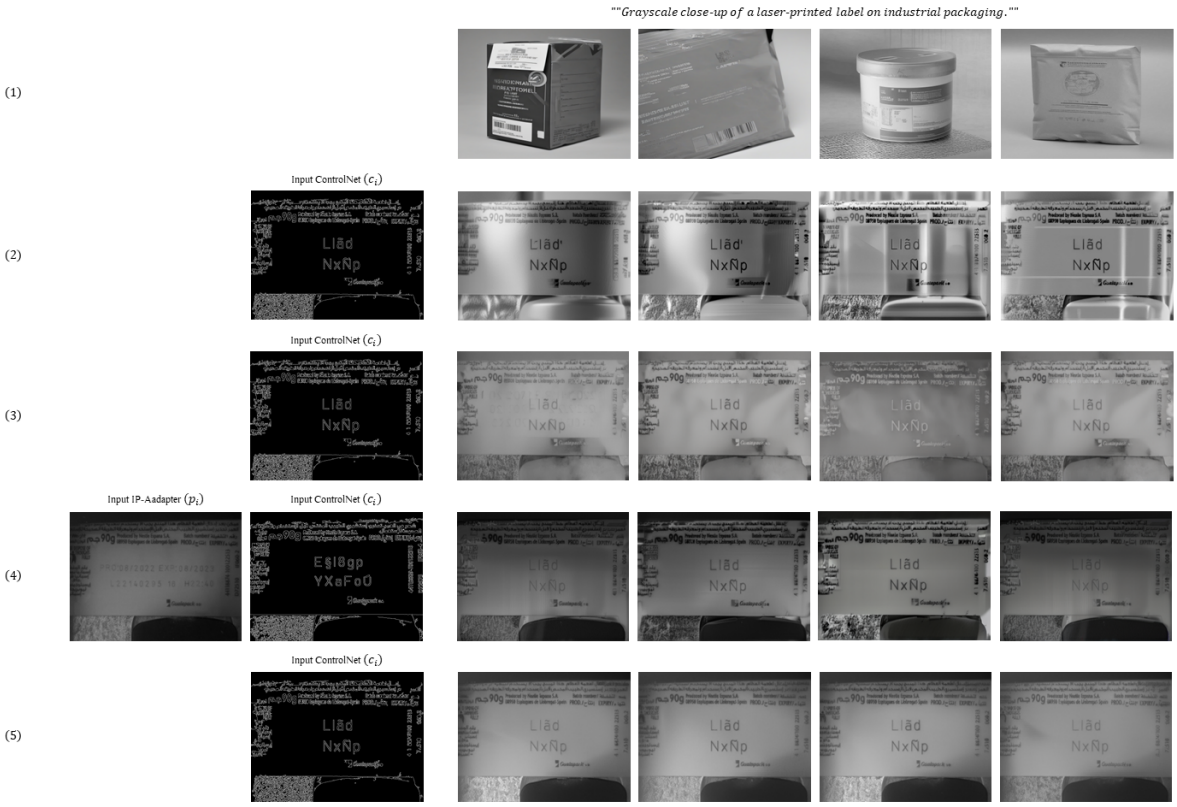

Fig. 10: Generated images using different methods: (1) Pretrained SDXL for text-to-image generation without additional conditions. (2) SDXL with ControlNet. (3) SDXL with ControlNet and the trained LoRA. (4) SDXL with ControlNet and an IP-Adapter. (5) SDXL with the fine-tuned ControlNet.

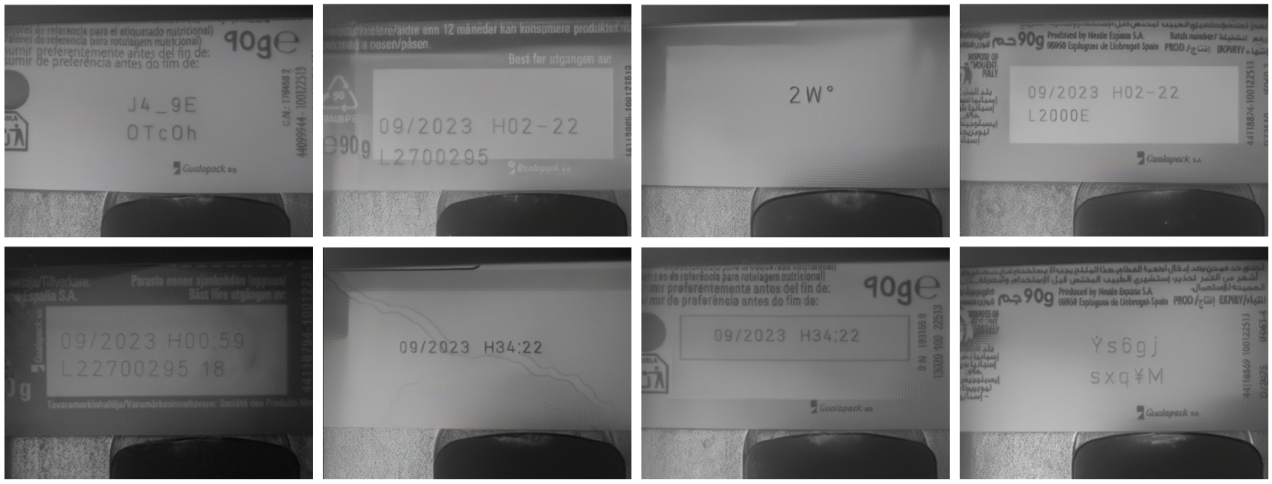

Fig. 11: Qualitative results showcasing outputs generated using our ControlNet-based method.

Due to the limited dataset size, conventional evaluation metrics such as Fréchet Inception Distance (FID) were deemed unsuitable [22], as they tend to be unstable and overly sensitive to outliers in small-scale settings. Instead, CLIP-based similarity was used to quantitatively assess the alignment between real and generated images. Specifically, CLIP embeddings were extracted for both real and generated image pairs, and cosine similarity was computed [23].

The average CLIP score achieved was 0.7544, which indicates a strong perceptual similarity between the generated outputs and their real counterparts. This is particularly notable given that the generated images often depict entirely novel characters and scenes not present in the training set, emphasizing the model's ability to generalize while preserving semantic and stylistic alignment.

5.2.3 Comparison with Classic Pixel Manipulation

To further assess the advantages of our approach, we compared it against traditional image manipulation techniques. Classic pixel-level editing was performed by applying blurs and manually inserting synthetic code regions using pixel color interpolation (see Appendix A.3). While this method is computationally efficient and simple to execute, it lacks the semantic depth and realism offered by generative approaches. Moreover, classic techniques often result in visually inconsistent artifacts that are easily detectable by human observers.

User Study

A user study was conducted via a custom-built web interface. Participants were first shown a series of real images to calibrate their expectations. Subsequently, they were presented with 10 pairs of images: one created with Stable Diffusion XL (SDXL) and one with the classic pixel manipulation technique. Importantly, participants were told that one image in each pair was real and one was synthetic, though in reality, both were generated.

The study had 65 participants, who were asked to identify which image in each pair they believed to be real. The results demonstrated a clear preference for the SDXL-based generation approach:

- **84.1%** of selections favored the SDXL-generated images as appearing more realistic.
- Only **9.5%** of responses identified the classic method as the real one.
- **6.3%** of participants found both images equally realistic.

These results highlight the superior perceptual realism of deep learning-based image synthesis compared to traditional pixel editing techniques. The study supports the conclusion that generative models not only produce visually coherent results but also better align with human perceptions of realism.

For a more detailed breakdown of the study configuration, please refer to Appendix B.

5.3 Extended Background Generation

One of the primary limitations of the initial real dataset was the lack of background variety. This issue was partially addressed using the generative approach: the fine-tuned model was capable of producing diverse backgrounds while adding realistic details such as shadows and textures that resemble those in the original dataset. Furthermore, new structural backgrounds were introduced by creatively manipulating the edge maps—while still preserving the visual style of the primitive dataset. Beyond these efforts, this section explores a different approach to generating printed codes in entirely new visual styles, such as those found on bottles, egg packages, and other packaging types.

While background style diversification is not the central focus of this project, it presents a promising direction for future work. The ability to synthesize printed codes on unfamiliar surfaces could significantly increase dataset diversity and improve model robustness in real-world scenarios.

To explore this idea, a small experiment was conducted using IP-Adapter in combination with ControlNet. The process involved:

- Using a real image of a labeled object as a reference input for the IP-Adapter.
- Creating a Canny edge map of a similar object with the printed code, which served as input for ControlNet.

This setup enabled the generation of new images, as illustrated in Figure 12, where printed codes appeared over surfaces that closely resemble the reference objects. Although the results are still preliminary and require further refinement, they demonstrate the potential of this strategy for future dataset versions.

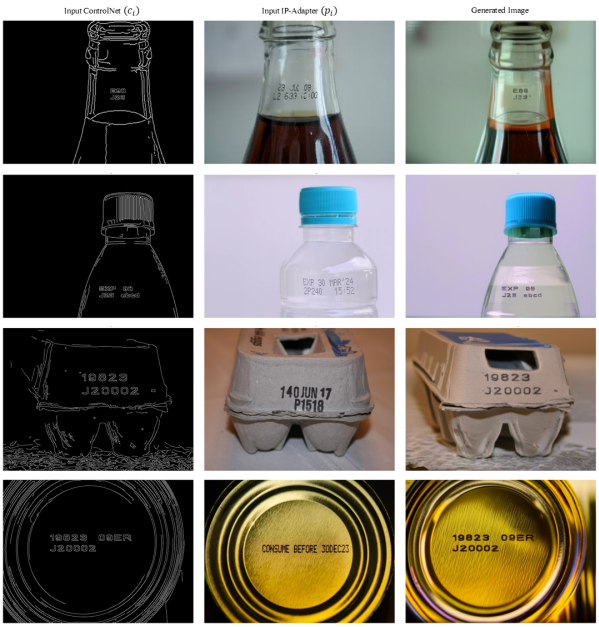

Fig. 12: Example results from background simulation using IP-Adapter and ControlNet.

5.4 Recognition Results

5.4.1 Performance Evaluation of OCR Techniques

The initial approach employed a two-step pipeline. First, the marking region was localized and then cropped to minimize background noise and enhance the accuracy of character recognition. Subsequently, an OCR technique was applied to the processed images. A total of 350 samples from the original dataset were used in this evaluation.

RoiNet

A neural network was implemented to predict the bounding

box coordinates of the marking region. Its primary purpose was to isolate the relevant area of the image, ensuring a cleaner input for subsequent OCR models. This helped reduce background interference, improving recognition accuracy.

Classical OCR Models

Three widely used OCR systems were evaluated on the cropped images produced by the marking region detector. These models—PaddleOCR [24], Tesseract [25], and EasyOCR [26]—were tested without any fine-tuning. All experiments were conducted on a laptop equipped with an NVIDIA GeForce RTX 4060 GPU.

Method	Exact Match Rate (%)	Avg. Levenshtein Distance	Time per Image (s)
RoiNet + PaddleOCR	38.24	0.91	0.0983
RoiNet + Tesseract	0.59	6.76	0.2447
RoiNet + EasyOCR	0.00	8.10	0.1465

TABLE 2: Comparison of OCR techniques.

The mentioned methodologies demonstrated limited effectiveness, even when combined with region localization. Due to the poor performance and the availability of more advanced alternatives, these models were not considered for further refinement or fine-tuning.

5.4.2 YOLOv11-Based End-to-End Approach

The end-to-end YOLOv11 model was selected for its ability to streamline the detection pipeline. Unlike previous multi-stage methods, this approach integrates region localization and classification within a single architecture, enabling a more efficient and scalable solution. Its support for rapid fine-tuning and suitability for real-time processing make it a strong candidate for practical deployment scenarios that require low latency and high accuracy.

Qualitative Results

Visual inspection of the model's predictions revealed consistently accurate detection of the target regions across diverse backgrounds and marking styles. As illustrated in Figure 15, the model successfully identified most of the regions of interest with minimal errors. When misclassifications occurred, they were typically associated with heavily degraded or ambiguous input images.

Quantitative Results

The evaluation metrics summarized in Table 3 highlight the strong performance of the model. High values for precision, recall, and mean Average Precision (mAP) confirm its reliability and consistency in the task of marking region detection.

5.5 Error Detection Preliminary Results

Following the described methodology, the approach was evaluated using 100 samples from each subset: valid cases, misplaced codes, incorrect characters, and surface defects. The pipeline demonstrated strong overall performance,

Metric	Value
Precision (mean)	0.9624
Recall (mean)	0.9835
mAP@0.5	0.9947
mAP@0.5:0.95	0.9299

TABLE 3: Performance metrics of the YOLOv11 end-to-end model on the test set.

with only minor misclassifications observed.

Specifically, the model achieved perfect classification for both valid cases and surface defects, attaining 100% accuracy. Misplaced codes were correctly identified in 99% of instances, with a single false positive mistakenly categorized as valid—likely due to overly broad marking region identification. Within the incorrect characters subset, 97 out of 100 samples were accurately classified. The remaining three were misclassified as surface defects, which can be attributed to lower confidence scores hovering near the established discrimination threshold.

The optimal discrimination threshold between incorrect characters and surface defects was set at a confidence score of 0.85. Samples with confidence scores above this threshold were classified as incorrect codes, reflecting high model certainty, whereas scores below this threshold indicated surface defects, reflecting model uncertainty.

Test Set	**Valid**	**Misplaced**	**Incorrect**	**Surface Defect**
Valid	100	0	0	0
Misplaced	1	99	0	0
Incorrect	0	0	97	3
Surface Defect	0	0	0	50

TABLE 4: Error detection performance across test categories.

While these preliminary results are promising, further refinement is necessary to improve classification accuracy and reduce the remaining errors.

6 Limitations and Future Work

6.1 Sensitivity to Edge Map Quality

The synthetic data pipeline employed in this study utilizes a pretrained ControlNet architecture conditioned on Canny edge maps. While this allows for scalable and automated generation of structurally coherent images, it introduces a significant reliance on the integrity of the edge input.

Degraded edge maps—resulting from suboptimal lighting, motion-induced blur, or background clutter—contribute to distorted generation outputs. These low-fidelity edges impede the preservation of geometric structure during diffusion-based synthesis, thereby deteriorating the realism of synthetic samples. Such limitations propagate into downstream tasks, weakening training efficacy and system generalizability.

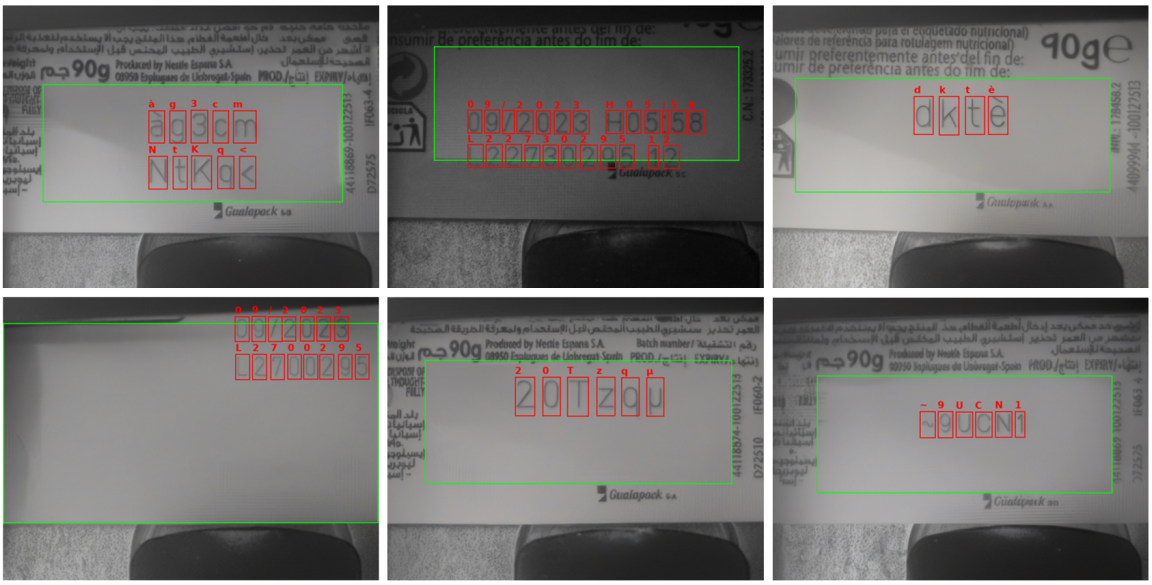

Fig. 13: Sample results from the YOLOv11 model on the validation set.

To mitigate this, preprocessing steps such as Gaussian denoising, local histogram equalization, or adaptive thresholding may be integrated. Nevertheless, these add to pipeline complexity and raise maintenance overhead.

Future research could prioritize robust conditioning strategies, such as noise-aware diffusion priors or adaptive edge weighting mechanisms, to suppress the impact of imperfect inputs without sacrificing automation. Figure 14 presents a comparative analysis of generated outputs under varying edge degradation levels.

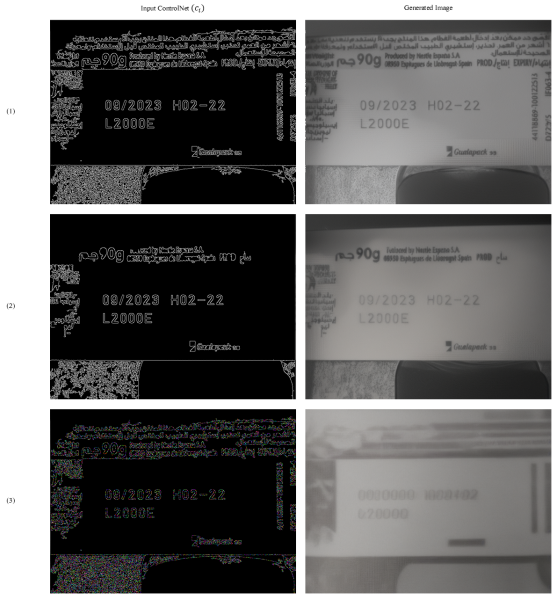

Fig. 14: Examples of synthetic image quality from: (1) Clean edges (2) Incomplete edges (3) Corrupted edges

6.2 High Computational Costs

Diffusion-based generation—particularly at high resolution—remains computationally intensive. Fine-tuning ControlNet for domain-specific data and sampling realistic outputs entail considerable GPU memory consumption, prolonged training cycles, and limited parallelizability.

This restricts accessibility for low-resource institutions and hinders deployment in real-time or embedded systems. Despite recent advances in acceleration (e.g., FlashAttention, LoRA), a cost-performance bottleneck persists.

Future directions may investigate model distillation, pruning, and quantization to compress large models without degrading output fidelity. Additionally, tailored architectural modifications—such as sparse convolutions or early-exit layers—can contribute to throughput gains while minimizing hardware demand.

6.3 Reducing Edge Dependency

Although edge-conditioned synthesis provides a structurally grounded framework, its rigidity may inhibit adaptability across datasets with varying visual characteristics.

Alternative control mechanisms, such as semantic layout maps, keypoint-conditioned generation, or other multimodal prompts (e.g., language + sketch input), could enhance flexibility. Such paradigms enable abstraction from low-level features, reduce preprocessing steps, and improve domain transfer.

Latent diffusion approaches that leverage internal representations (rather than explicit pixel-space edges) may further attenuate the reliance on high-contrast boundary cues. Exploring hybrid controls that combine soft and hard structural priors could offer a trade-off between flexibility and fidelity.

6.4 Multi-Defect Simulation

The current generation paradigm primarily synthesizes single-defect samples, which may not sufficiently emulate the heterogeneity of real-world print or rendering anomalies.

In operational environments, compound defects—such as kerning errors co-occurring with noise, occlusions paired with misalignments—are frequently observed and may require specialized detection strategies.

Dynamic synthesis of multi-defect scenarios using conditional logic or adversarial perturbations could train detection models to better manage ambiguity and variability. Incorporating context-aware generation mechanisms would further align synthetic samples with real-world complexity.

6.5 Model Generalization Limitations

While the fine-tuned YOLOv11 model demonstrates strong performance on in-domain datasets, its generalization capacity diminishes under cross-domain shifts—particularly involving novel fonts, textures, or lighting conditions.

This brittleness under distribution shift reveals the necessity for more diverse and representative data augmentation strategies. Future work could adopt curriculum learning, domain adaptation frameworks (e.g., adversarial alignment, feature norm regularization), or meta-learning techniques to enhance cross-domain robustness.

Additionally, integrating synthetic data into semi-supervised learning pipelines may bridge the gap between labeled source domains and unlabeled target domains, improving adaptability without incurring extensive annotation costs.

6.6 Incomplete Error Detection Coverage

The present defect detection framework does not fully encompass all anomaly types observed across industrial datasets. Rare or subtle errors—such as halo artifacts, ink bleeding, or spatial jitter—remain underrepresented, leading to blind spots in the current pipeline.

Augmenting model capacity through transformer-based detectors or feature-ensemble fusion may bolster sensitivity to such anomalies. Moreover, targeted dataset curation emphasizing hard-negative mining and rare-class amplification can expand coverage.

Future research should also investigate few-shot detection methods to enable adaptability in data-scarce classes, and assess reliability in scenarios exhibiting annotation uncertainty or perceptual ambiguity.correction.

7 Conclusions

This work presents a comprehensive, data-centric framework for enhancing Optical Character Verification (OCV) in industrial laser marking environments. By combining state-of-the-art generative modeling with targeted training strategies, the proposed approach addresses major challenges inherent in real-world industrial datasets: imbalanced character distributions, limited defect representation, and constrained background variability.

A key innovation lies in the use of fine-tuned diffusion models—specifically Stable Diffusion XL with ControlNet—to generate high-fidelity synthetic images that replicate the structure and style of real-world packaging environments. Through strategically crafted edge maps and augmentation strategies, the system effectively simulates critical defect types, underrepresented characters, varied font scales, and realistic backgrounds, enabling the creation of a richly diverse training dataset.

Experimental validation demonstrates that this synthetic data significantly improves downstream model performance. The YOLOv11-based recognition pipeline achieves high precision and recall in character localization and classification tasks, while an integrated error detection module effectively identifies misplaced codes, incorrect characters, and surface degradation. A user study further confirms the perceptual realism of the generated images, with 84.1% of participants favoring the diffusion-generated outputs over traditional pixel manipulation methods.

However, certain limitations persist. The reliance on high-quality Canny edge maps introduces a dependency on input preprocessing. The YOLOv11 model, while robust, is narrowly tuned to a specific font and visual domain, limiting generalization to unseen contexts. Furthermore, the current error detection logic, though functional, lacks complete coverage for more complex or combined defects.

Looking ahead, future research can focus on minimizing preprocessing requirements through edge-free or semantic-guided generation, expanding model adaptability across fonts and surfaces, and enriching the error taxonomy with uncertainty estimation and multi-stage reasoning modules. Additionally, investigating lightweight architectures and model distillation could make the system more viable for deployment on edge devices.

In summary, this work demonstrates how principled synthetic data generation and real-time learning architectures can be harmonized to build scalable, high-precision OCV systems for industrial quality assurance. The modular design ensures adaptability to evolving production demands, while the reliance on synthetic data minimizes the burden of manual annotation and real-world data acquisition.

The full source code, training scripts, and additional materials are available at the GitHub repository: https://github.com/juliagartor/Synthetic-Data-Generation-for-Robust-Optical-Character-Verification-in-Industrial-Laser-Marking.git.

REFERENCES

[1] R. Smith, "Optical character recognition: An illustrated guide to the frontier," *Document Analysis and Recognition*, vol. 2, pp. 111–118, 2007.

[2] Y. Baek, G. Kim, J. Lee, *et al.*, "Scene text recognition with transformer-based sequence modeling," *Proceedings of CVPR*, 2021.

[3] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE TPAMI*, vol. 39, no. 11, pp. 2298–2304, 2017.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CVPR*, pp. 779–788, 2016.

[5] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[6] J. Zhou, F. Wang, and L. Zhang, "Challenges in industrial computer vision systems: A review," *Journal of Manufacturing Systems*, vol. 56, pp. 343–356, 2020.

[7] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "Defect-gan: Generative adversarial networks for defect synthesis in industrial inspection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 196–16 205.

[8] S. Yang, C. Meng, and J. Ho, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.

[9] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image inpainting with latent diffusion models," *arXiv preprint arXiv:2112.10752*, 2021.

[10] L. Zhang and M. Agrawala, "Controlnet: Adding conditional control to text-to-image diffusion models," 2023, available at https://github.com/lllyasviel/ControlNet.

[11] R. Zhang *et al.*, "Ip-adapter: Text-to-image diffusion models with visual condition injection," 2023, available at https://github.com/tencent-ailab/IP-Adapter.

[12] X. Yao, L. Wang, *et al.*, "Synthetic data generation for industrial surface inspection using physics-based rendering and gans," *Computers in Industry*, vol. 125, p. 103376, 2021.

[13] K. Lis, P. Schönberger, A. Dittadi, *et al.*, "Bridging the domain gap in industrial anomaly detection with synthetic data," *NeurIPS*, 2022.

[14] S. AI, "Stable diffusion xl," https://github.com/Stability-AI/generative-models, 2023.

[15] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[16] H. Face, "Controlnet for sdxl," https://huggingface.co/xinsir/controlnet-union-sdxl-1.0, 2023.

[17] madebyollin, "Sdxl vae fp16 fix," https://huggingface.co/madebyollin/sdxl-vae-fp16-fix, 2023.

[18] H. Face, "Accelerate: A simple way to train and deploy pytorch models with mixed precision and multi-gpu support," https://github.com/huggingface/accelerate, 2021.

[19] L. Biewald, "Weights and biases," https://www.wandb.com, 2020.

[20] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," 2020, available at https://arxiv.org/abs/1909.13719.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," https://arxiv.org/abs/2106.09685, 2021.

[22] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6626–6637.

[23] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021.

[24] B. Du *et al.*, "Paddleocr: An open-source ocr toolkit based on paddlepaddle," https://github.com/PaddlePaddle/PaddleOCR, 2020.

[25] R. Smith, "An overview of the tesseract ocr engine," in *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2007, pp. 629–633.

[26] JaidedAI, "Easyocr: Ready-to-use ocr with 80+ languages supported out of the box," https://github.com/JaidedAI/EasyOCR, 2020.

[27] N. Ruiz, Y. Li, V. Jampani, *et al.*, "Dreambooth: Fine-tuning text-to-image diffusion models for subject-driven generation," https://dreambooth.github.io, 2023.

[28] Anonymous, "Realisticvision: High-fidelity photorealistic diffusion model," https://civitai.com/models/4201/realistic-vision-v50, 2023.

[29] S. Inc., "Streamlit: Turn data scripts into shareable web apps," https://streamlit.io, 2021.

[30] G. LLC, "Google forms: Online surveys made easy," https://www.google.com/forms/about/, 2023.

APPENDIX

A INITIAL DATA AUGMENTATION EXPERIMENTS

A.1 DreamBooth Inpainting for Character Substitution

DreamBooth is a fine-tuning technique designed to personalize text-to-image diffusion models by learning unique visual traits from a limited set of reference images [27]. This technique was adapted to perform digit augmentation through an inpainting approach.

Methodology

Real dataset images are used during training, where a digit is masked and the model is prompted to reconstruct it (e.g., *'Generate the digit [[5]]'*). The target output is the original unmasked image, enabling the model to learn digit-specific inpainting within natural contexts. Training mirrors DreamBooth's approach, combining visual and textual cues to guide reconstruction. Once fine-tuned, the model can synthesize new digit variations by masking regions and specifying target characters.

Results and Limitations

While effective for augmenting datasets with controlled variations (see Figure 15), the approach has key constraints:

- **Limited Generalization**: The model only regenerates digits seen during training and fails to reproduce unseen characters.
- **Static Backgrounds**: Backgrounds remain largely unchanged, as only the masked region is modified.
- **Error Generation**: The method is unsuitable for synthesizing targeted errors.

A.2 Inpainting for Surface Defect Generation

To avoid additional training, an inpainting-based approach was explored for generating realistic surface defects. This method leverages a pre-trained diffusion model (RealisticVision) [28] to synthesize defects on existing images without fine-tuning.

Methodology

The input consists of:

- An image with a randomly masked region (defining the defect location)
- A text prompt describing the desired defect characteristics

The model then generates the defective area while preserving the surrounding context. This zero-shot approach eliminates the need for defect-specific training data.

Results and Limitations

While the method produces visually convincing defects (see Figure 16), several limitations were observed:

- **Shape Control**: The random mask shapes don't always match real defect geometries
- **Texture Consistency**: Generated defects may lack material-specific textures
- **Position Sensitivity**: Results vary significantly based on mask location
- **Scale Limitations**: Small defects often lack detail

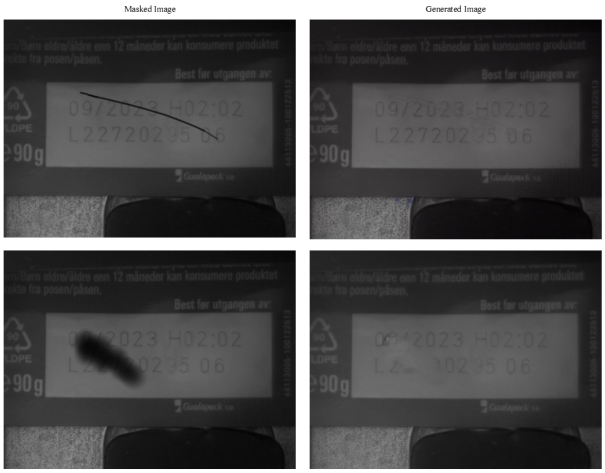

Fig. 16: Defect generation examples: (Left) Input images with random masks, (Right) Output images with generated defects. While visually plausible, the defects may lack physical accuracy.

A.3 Classic Pixel Manipulation

To avoid data-hungry and computationally intensive generative approaches, a simpler method was explored for surface code manipulation. This pixel-based technique enables the removal and replacement of printed codes without the need for model training or large datasets.

Methodology

This method consists of two key stages: Background Reconstruction and Code Synthesis.

In the Background Reconstruction phase, a binary mask identifying the original code is created and expanded using morphological dilation. The masked region is then filled by sampling pixel values based on the statistical properties (mean and standard deviation) of the surrounding area. This step ensures that the background maintains a realistic appearance after code removal.

Fig. 15: Inpainting results: (Left) Source image with masked digit, (Center) Generated segmentation mask, (Right) Reconstructed output using DreamBooth. The method effectively regenerates digits such as [[a]] and [[e]], preserving visual consistency.

Following this, in the Code Synthesis phase, new synthetic codes are digitally rendered over the reconstructed background using vector fonts. This allows precise control over code attributes such as position, size, and content.

Results and Limitations

This approach delivers visually clean and customizable outputs, especially in scenarios with uniform or textured surfaces. It allows for fine-grained control over the placement and appearance of synthetic elements, making it ideal for structured datasets and controlled experiments.

However, several limitations were observed:

- **Background Assumptions**: Works best on regular or moderately textured surfaces; performance declines on complex or highly varied backgrounds.
- **Manual Effort**: May require some human supervision or fine-tuning to ensure seamless blending in challenging cases.
- **Limited Creative Flexibility**: Compared to generative approaches, this method lacks the capability to introduce spontaneous or diverse visual variations.

B User Study

A user study was conducted using a custom web application deployed on Streamlit Cloud [29]. Streamlit is a Python framework that enables rapid development of interactive web apps for data science and machine learning projects, with simple code and live updates, making it ideal for quick user studies and experiments.

The app presented participants with pairs of synthetic images generated by two different methods: Stable Diffusion XL (an advanced AI image generation model) and a simulated pixel manipulation technique. In each round, participants viewed two images side-by-side and were asked to select which image looked more like a real photograph. To reduce bias, the position of images and their corresponding method labels were randomized.

Before the test rounds, participants were shown examples of real package code images to familiarize them with authentic visuals. Throughout the test, participant choices were tracked. After completing all rounds, participants were shown a summary revealing that all images were synthetic and received feedback on their preferences between the two methods.

The anonymized results — including the number of times each method was selected — were automatically submitted to a Google Form for collection and further analysis [30]. The study included 65 participants, whose responses helped identify which synthetic image generation method was perceived as more realistic by human observers.

B.1 Results

The user study results demonstrate a clear preference for Stable Diffusion XL, with participants overwhelmingly selecting it as the more realistic image generation method. In most cases, Stable Diffusion XL won by wide margins (e.g., 8-2, 9-1, or even 10-0), while the simulated method was preferred in only a few instances (e.g., 7-3, 6-4). A small number of responses resulted in ties (5-5), indicating no strong preference.

The data confirms that Stable Diffusion XL consistently outperforms the simulated method in perceived realism, aligning with its advanced AI capabilities. The rare exceptions suggest that the simulated approach may occasionally produce convincing results, but overall, Stable Diffusion XL remains the superior choice for generating realistic synthetic images.