Tap, Scan, Exploit: The Hidden Vulnerabilities of Everyday QR Codes

Supplementary Material

Table 1. Performance of two GAN-based restoration models evaluated on the full 18-corruption dataset, with each model specifically trained on either motion blur or zoom blur distortions. Bold values indicate the best performance among all methods.

Model	Trained On	RR	PSNR	SSIM
No Defense	–	64.90	18.04	0.61
Dong et al. [2]	Motion Blur	**89.84**	18.76	0.79
Dong et al. [2]	Zoom Blur	84.66	24.54	0.89
Gu et al. [5]	Motion Blur	81.81	16.31	0.78
Gu et al. [5]	Zoom Blur	87.76	**28.81**	**0.92**

1. Performance of SOTA Defense Models over the Complete Corruption Spectrum

In the main paper, our analysis focused on a restricted subset of five corruption classes. In this comprehensive evaluation, we include all 18 corruption types to capture the full range of distortions encountered in real-world scenarios. Across these corruption categories, a total of 5,040 images ($56 \times 5 \times 18$) are evaluated. As summarized in Table 1, both GAN-based defense models substantially outperform the no-defense baseline across all metrics. Among the two architectures, the method proposed by Dong et al. demonstrates stronger robustness under motion blur, while the model by Gu et al. remains competitive and achieves its best performance under zoom blur. These results suggest that corruption-specific training improves generalization across the diverse set of 18 corruptions, reinforcing the usefulness of corruption-aware learning strategies for enhancing QR code robustness. Nevertheless, the diversity and complexity of these distortions suggest that further investigation is needed to understand model behavior fully and to develop more generalizable defense strategies.

2. Extended Aesthetic QR Code Analysis

In the main paper, we demonstrate that aesthetic QR codes are more vulnerable to image corruptions. Here, we extend this analysis by considering different publicly available variants. We have now generated a large-scale aesthetic code dataset comprising 4000 samples (1,000 per dataset) using four methods, ranging from web-based to prompt-based generation. We evaluate the best-performing scanner from the main paper (QReader) in Table 2 with the image size 512x512. While most corruption has a limited impact, possibly due to high-quality aesthetic codes, MFI corruption significantly degrades decoding performance.

Table 2. Accuracy (%) of scanners under various real-life corruptions on different dataset types of aesthetic QR codes. EE: Environmental Effects, MFI: Motion and Focus Issues, PDI: Printing and Display Imperfections, LCV: Lighting and Color Variations, PDO: Physical Damage and Obstruction. Halftone:[6], QArt Codes:[1], Web-based:[4], Prompt-Based:[3].

Dataset	EE	MFI	PDI	LCV	PDO
Halftone QR [6]	**86.67**	**33.20**	**67.50**	100	**76.00**
QR Code AI Art [1]	98.67	70.80	86.50	100	87.34
QR Planet GmbH [4]	96.67	46.40	76.00	98.67	88.67
QArt Coder [3]	96.67	81.20	99.50	**98.00**	96.00

3. Data visualization

Figure 1 provides visual examples showcasing both standard and aesthetically enhanced QR codes subjected to diverse distortion conditions. Figure 2 shows contrast and zoom blur corruption types across severity levels. These illustrations highlight the visual degradation patterns across multiple perturbation scenarios. Complementing this, Table 3 systematically outlines 18 distinct corruption methodologies, complete with their mathematical formalisms and parameter configurations spanning five progressive severity levels.

4. Implementation Details

The framework is run on the Ubuntu operating system with an Intel Core i9-12900K CPU, 64GB of RAM, and an NVIDIA GeForce RTX 3050 GPU (8GB VRAM). The Python-based scanners utilize built-in Python libraries, while preliminary studies are developed with PyTorch. For both the [2] and [5] architectures, we use Adam ($\beta_1 = 0.5$, $\beta_2 = 0.999$) with a constant 1×10^{-4} learning rate, training each generator–discriminator pair for 300 epochs at batch size 1. Following [2], we apply a LinearLR scheduler that linearly decays the learning rate to zero over the first 150 epochs; for [5], we employ a custom LambdaLR schedule that linearly reduces the rate from 1×10^{-4} to 1×10^{-7} beginning at epoch 150. To ensure reproducibility and to maintain consistency with commercial scanning environments, all experiments are conducted systematically. A Gorilla tripod is used to securely mount the mobile device, and all scanning tests are performed in the morning to minimize variations in ambient lighting. For each test scenario, the scanner is given a fixed 10-second window to decode the corrupted QR code. The screen-to-scanner distance is standardized according to the recommended scanning distances (RSD) as shown in Table 4, following the QR sizing guide-

Figure 1. Normal and aesthetic QR codes under various real-world distortions.

Table 3. Image corruption types with mathematical formulations and severity levels. Adapted from [7].

Corruption Type	Mathematical Formulation	Severity Parameters
Shot Noise	$I' = \frac{\text{Poisson}(I\lambda)}{\lambda}$	$\lambda \in \{60, 25, 12, 5, 3\}$
Impulse Noise	$I'(x) = \begin{cases} 0 & \text{w.p. } p/2 \\ 255 & \text{w.p. } p/2 \\ I(x) & \text{otherwise} \end{cases}$	$p \in \{0.03, 0.06, 0.09, 0.17, 0.27\}$
Speckle Noise	$I' = I + I \cdot \mathcal{N}(0, \sigma^2)$	$\sigma \in \{0.15, 0.2, 0.35, 0.45, 0.6\}$
Gaussian Blur	$I' = I * G_\sigma,\ G_\sigma(r) = \frac{1}{2\pi\sigma^2} e^{-r^2/(2\sigma^2)}$	$\sigma \in \{1, 2, 3, 4, 6\}$
Glass Blur	$I' = I * G(\sigma)$, then pixel swap with offset δ	$(\sigma, \delta, n) \in \{(0.7, 1, 2), (0.9, 2, 1), (1, 2, 3), (1.1, 3, 2), (1.5, 4, 2)\}$
Defocus Blur	$I'(x, y) = \frac{1}{\lvert B_r \rvert} \sum_{(dx,dy) \in B_r} I(x-dx,\ y-dy)$	$(r, \sigma) \in \{(3, 0.1), (4, 0.5), (6, 0.5), (8, 0.5), (10, 0.5)\}$
Motion Blur	$I'(x, y) = \frac{1}{L} \sum_{k=0}^{L-1} I(x-kv_x,\ y-kv_y)$	$(L, \theta) \in \{(10, 3), (15, 5), (15, 8), (15, 12), (20, 15)\}$
Zoom Blur	$I' = \frac{1}{n+1}\left(I + \sum_{i=1}^{n} \text{Zoom}(I, z_i)\right)$	z ranges: $[1.00, 1.10]$ (Δ=0.01) to $[1.00, 1.30]$ (Δ=0.03)
Fog	$I' = I + \alpha \cdot \text{Plasma}(\beta)$	$(\alpha, \beta) \in \{(1.5, 2), (2, 2), (2.5, 1.7), (2.5, 1.5), (3, 1.4)\}$
Frost	$I' = \alpha I + \beta F,\ F \sim \text{FrostPattern}$	$(\alpha, \beta) \in \{(1, 0.4), (0.8, 0.6), (0.7, 0.7), (0.65, 0.7), (0.6, 0.75)\}$
Snow	$I' = \alpha I + (1-\alpha)\max(I, G) + S + \text{rot}(S)$	Severity 1–5 with varying α, snow layer parameters, and brightness
Spatter	$I' = I(1 - M) + MC + N$	Mud/water spatter with varying mask M, color C, intensity
Contrast	$I' = (I - \mu_I) \cdot c + \mu_I$	$c \in \{0.4, 0.3, 0.2, 0.1, 0.05\}$
Brightness	$I' = \text{clip}(I + c,\ 0,\ 255)$	$c \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
Saturate	$I'_{\text{HSV}} = (H,\ S \cdot c_1 + c_2,\ V)$	$(c_1, c_2) \in \{(0.3, 0), (0.1, 0), (2, 0), (5, 0.1), (20, 0.2)\}$
JPEG Compression	$I' = \text{JPEG}(I, q)$	$q \in \{25, 18, 15, 10, 7\}$
Pixelate	$I' = \text{Upsample}(\text{Downsample}(I, r))$	$r \in \{0.6, 0.5, 0.4, 0.3, 0.25\}$
Elastic Transform	$I'(x, y) = I(x+\alpha\eta_x,\ y+\alpha\eta_y),\ \eta \sim G_\sigma$	$\alpha \in \{12.5, 16.25, 21.25, 25, 30\},\ \sigma = 4$

lines provided by Scanova [8]. For prompt-based generation using QArt Coder [3], the prompts used in our experiments are provided in the project page linked in the main paper.

Table 4. Recommended scanning distances (RSD) based on the displayed QR code size on the screen.

Image	Physical Width	RSD
300×300	2.54 cm	∼20–30 cm
600×600	5.08 cm	∼40–60 cm
1200×1200	10.16 cm	∼80–120 cm
512×512	4.34 cm	∼40–50 cm

5. Evaluation Metrics

To assess the robustness of models used in the mitigation analysis, we use three main metrics for comparison. The **Recognition Rate (%)** metrics evaluate the scanning performance of the distorted or restored QR code. For fair comparison, we use the pyzbar Python library [9] to benchmark it with [2].

$$\text{Recognition Rate (\%)} = \frac{\text{No. of Decodable QR Codes}}{\text{Total Samples}} \times 100 \tag{1}$$

Peak Signal-to-Noise Ratio (PSNR) to measure image quality and **Structural Similarity Index Measure (SSIM)** for image similarity or structural information. Both implemented via `skimage.metrics` [10].

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right), \tag{2}$$

where MAX is the maximum possible pixel value of the image, 255 in our case for no defense and 1 for models which we have trained, and MSE denotes the mean squared error between the deblurred and ground truth images.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{3}$$

where μ_x and μ_y are the local means, σ_x^2 and σ_y^2 are the local variances, and σ_{xy} is the local covariance of image patches x and y. For calculating SSIM, we use a window size of

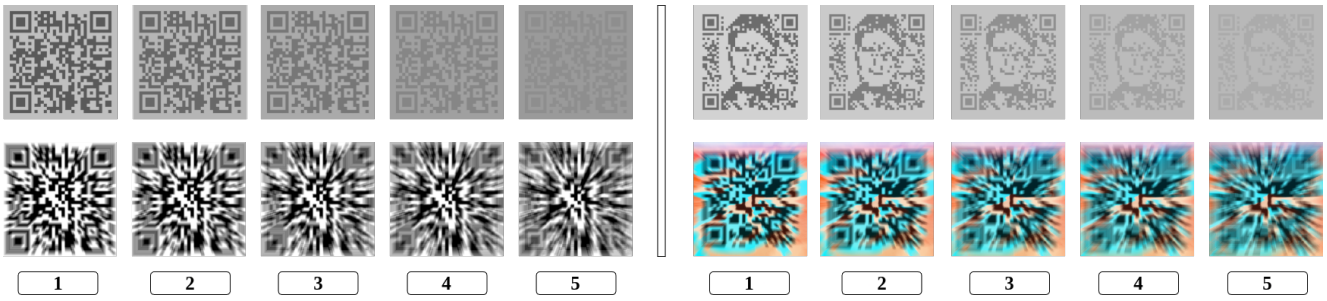

Figure 2. Examples of global corrupted QR code images across severity levels (1–5). Standard QR codes are shown on the left and aesthetic QR codes on the right. The top row corresponds to contrast corruption, and the bottom row to zoom blur.

5 and specify the color channel using `channel_axis=2`. The parameter `data_range=255` for no defense, and `data_range=1` for models that we have trained.

6. Future Work

In this work, we evaluate a diverse set of QR code scanners under controlled settings, including fixed versions, masking patterns, error-correction levels, and encoded data lengths, key factors influencing QR code robustness. A natural extension of this study is to conduct more fine-grained analyses by systematically varying individual parameters to better understand the impact of different corruption types on decoding performance. Additionally, while we provide a preliminary benchmarking of a generative model, further exploration of advanced generative architectures could yield deeper insights. Investigating novel model architectures and designing dedicated defense mechanisms remains a promising direction for improving robustness against unseen corruptions.

References

[1] QArt Coder, 2012. Accessed: 2026-25-01. 1

[2] Hao Dong, Haibin Liu, Mingfei Li, Fujie Ren, and Feng Xie. An algorithm for the recognition of motion-blurred qr codes based on generative adversarial networks and attention mechanisms. *International Journal of Computational Intelligence Systems*, 17(1):83, 2024. 1, 3

[3] QR Code AI Art Generator, 2023. Accessed: 2026-25-01. 1, 3

[4] QR Planet GmbH, 2023. Accessed: 2026-25-01. 1

[5] Wencheng Gu, Kexue Sun, Zhipeng Jiang, and Li Sun. Gs-deblurganv2: a qr code deblurring algorithm based on lightweight network structure. *Multimedia Systems*, 30(2): 87, 2024. 1

[6] Halftone QR Codes, 2013. Accessed: 2026-25-01. 1

[7] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations, 2019. 3

[8] Minimum QR Code Size: Find The Perfect Size For Your Use Case. Ashish chandra, 2025. Accessed: 2025-02-08. 3

[9] Pyzbar Developers. Pyzbar: A python library for qr code and barcode decoding, 2024. Accessed: 2025-02-08. 3

[10] Scikit-image team. Skimage.metrics, 2025. Accessed: 2025-02-06. 3