Digisuraksha Parhari Foundation

Cybersecurity Internship Research Paper

Quick Response (QR) CODE Detector

By

Atharva Keluskar

Soham Kale

Abstract

In the era of rapid technological advancement driven by artificial intelligence (AI), QR code detection systems have evolved from simple image recognition tools to intelligent, context-aware components integrated across diverse industries. Traditional QR code scanners relied on classical computer vision techniques, but modern AI-enhanced detectors leverage deep learning models such as convolutional neural networks (CNNs) for improved accuracy, speed, and robustness in varied lighting, angles, and noise conditions. These detectors now feature in applications ranging from secure digital payments and smart retail systems to augmented reality and industrial automation. Moreover, with the integration of edge AI and federated learning, QR code detection is becoming more efficient and privacy-preserving. This shift exemplifies how AI is not only augmenting the capabilities of existing technologies but also reshaping their roles in smart, interconnected ecosystems. As we progress further into the AI revolution, QR code detectors stand as a microcosm of broader innovations—smarter, faster, and more adaptable tools bridging the physical and digital worlds.

Problem Statements & Objectives

- <u>**Problem Statements**</u>

1. Limited Performance of Traditional Methods:

Existing QR code detection methods based on classical image processing techniques, such as edge detection and pattern recognition, struggle under real-world conditions like poor lighting, varying angles, motion blur, and background clutter.

2. Inability to Handle Real-World Challenges:

Traditional QR detection systems fail to detect QR codes reliably when they are partially occluded, rotated, or distorted due to real-time environmental factors.

3. Low Robustness in Complex Environments:

QR codes in real-world scenarios often face background noise, image distortions, or damaged codes that affect traditional detection methods' performance, resulting in failed or delayed scans.

4. Inefficient Mobile and Embedded Deployment:

Current QR detection solutions are not optimized for mobile or embedded devices, making it difficult to achieve real-time scanning and processing, especially in low-resource environments.

- **<u>Project Objectives</u>**

1. Analyze Limitations of Conventional Detection Methods:

Identify and assess the challenges posed by traditional QR code detection systems under real-world conditions, such as lighting variations, occlusion, and motion blur.

2. Develop an AI-based QR Code Detection System:

Design and implement a QR code detection model using deep learning architectures like Convolutional Neural Networks (CNNs) or YOLO (You Only Look Once) to improve accuracy and robustness.

3. Create and Curate a Diverse QR Code Dataset:

Build a custom dataset featuring QR codes captured in various conditions (different lighting, backgrounds, occlusions, rotations, etc.) to ensure the model's generalizability and robustness.

4. Benchmark Against Traditional Detection Techniques:

Compare the AI-based model's performance with conventional QR code detection techniques (e.g., OpenCV's QRCodeDetector) in terms of accuracy, speed, and robustness under varied conditions.

5. Optimize the Model for Mobile and Embedded Devices:

Implement model optimization techniques such as quantization and pruning to reduce the model's size and ensure fast inference on mobile and embedded platforms (e.g., smartphones, Raspberry Pi).

6. Develop a Real-Time Application Prototype:

Create a real-time QR code detection application for mobile or embedded devices, demonstrating the practical use of the AI model in real-world scenarios (e.g., inventory management, ticket scanning, etc.).

7. Evaluate System Performance in Real-World Environments*:

Evaluate the prototype application in practical use cases and environments to assess the model’s real-time performance, speed, and reliability in real-world conditions.

Literature Review

The task of QR code detection has evolved significantly since the inception of the QR code by Denso Wave in 1994. Over the years, researchers and developers have explored various methods for reliable and efficient detection of QR codes, ranging from classical image processing to modern deep learning approaches. This section summarizes notable works and trends in QR code detection technology.

A. Traditional QR Code Detection Methods

Early QR code detection techniques were predominantly based on classical computer vision algorithms. OpenCV, an open-source library for image processing, offers a built-in QRCodeDetector that uses techniques like edge detection, contour analysis, and geometric transformation to locate QR codes. These methods are computationally efficient and work well under controlled conditions; however, they exhibit poor performance in the presence of:

- **Image distortion (e.g., blur, low contrast)**
- **Rotated or partially occluded QR codes**
- **Variations in lighting and background clutter**

Studies such as Zhang et al. (2016) demonstrated the use of morphological operations and Hough transforms for QR code localization, achieving acceptable results under optimal conditions. However, the reliance on handcrafted features and threshold-based decision-making limits adaptability in diverse environments.

B. Applications in Real-Time and Edge Environments

As the need for mobile and IoT-based scanning grows, recent studies have investigated model optimization for low-power devices. Techniques such as model quantization, pruning, and TensorRT acceleration have enabled deployment of deep learning models on embedded systems like Raspberry Pi and Jetson Nano.

Chen et al. (2022) demonstrated a QR code scanner based on Tiny-YOLOv4 optimized for Android devices, highlighting significant improvements in scan speed and energy efficiency. Their work underscored the feasibility of integrating deep learning detectors into real-world mobile applications.

C. Gaps in the Literature

Despite these advancements, several research gaps remain:

- Limited availability of large, diverse QR code datasets with real-world variations.
- Lack of comparative evaluations between AI-based and traditional methods under standardized conditions.
- Underutilization of edge AI techniques for offline, low-latency QR code recognition.
- Few end-to-end solutions that combine detection, decoding, and user interaction in a single framework.

Research Methodology

1. Problem Definition:

Analyze limitations of traditional QR code detection methods in real-world conditions (lighting, occlusion, etc.).

2. Dataset Creation:

Collect diverse QR code images under varying conditions and apply data augmentation (rotation, noise, etc.) to expand the dataset.

3. Model Selection:

Choose an efficient deep learning model (e.g., YOLOv5) for real-time QR code detection.

4. Model Training:

Fine-tune a pretrained model using the custom dataset with optimized hyperparameters.

5. Model Evaluation:

Measure model performance using metrics like accuracy, precision, recall, and FPS for real-time testing.

6. Optimization for Edge Devices:

Apply quantization and pruning techniques to optimize the model for mobile and embedded devices.

7. Prototype Development:

Develop a mobile or embedded system application that integrates the trained QR code detection model for real-time use.

8. Real-World Testing:

Test the prototype in real-world environments to assess its accuracy, speed, and robustness.

9. Conclusion:

Summarize findings, evaluate feasibility for deployment, and suggest future improvements.

Tool Implementation

The QR Code Risk Scanner is implemented using basic web technologies:

- **Frontend: Built using HTML, CSS, and JavaScript to create a simple user interface that allows image uploads.**
- **QR Code Scanning: JavaScript (with libraries like jsQR) scans the uploaded QR image and decodes the URL.**
- **URL Safety Check: Once the URL is decoded, JavaScript sends a request to the Google Safe Browsing API to check if the URL is malicious.**
- **Results Display: The result (Safe or Malicious) is shown directly on the webpage without using a backend server.**

1. Data Collection and Annotation Tools

- **LabelImg: A graphical image annotation tool to label the QR code positions in the training dataset with bounding boxes. This tool helps in generating ground-truth annotations for training.**
- **Roboflow: An alternative tool to LabelImg, it allows for easier data augmentation and dataset management, improving model generalization.**

2. Version Control

Git/GitHub: For tracking changes, collaborating with team members, and versioning code throughout the project's development lifecycle.

Ethical Impact & Market Relevance

Ethical Impact:

1. **Privacy and Data Security:**

   Ensure user data is not stored or shared without consent. Implement encryption to protect sensitive information.

2. **Bias and Fairness:**

   Use a diverse dataset for model training to avoid biases and ensure fair detection across various environments and QR code types.

3. **Dependence on Technology:**

   Mitigate over-reliance on QR codes by educating users on potential system failures and encouraging responsible use.

4. **Misuse of QR Codes:**

   Implement security measures to detect and warn users about potentially malicious QR codes and phishing attempts.

5. **Environmental Impact:**

   Optimize the model to reduce energy consumption on edge devices and promote sustainable practices in deployment.

Future Scope

- **Integration with Augmented Reality (AR):**
  Enhance QR code detection by integrating AR features, allowing users to interact with real-world objects and information by scanning QR codes.

- **Multi-Format Code Detection:**

  Expand detection capabilities to include other types of codes (e.g., Data Matrix, Aztec, and PDF417), making the system more versatile across industries.

- **Real-Time Analytics and Reporting:**

  Develop real-time analytics tools to gather and analyze data from QR code scans for businesses to track customer behavior and product interactions.

- **Improved Security Features:**

  Implement advanced security protocols, such as secure QR code validation, to protect users from phishing and malicious activities.

- **Enhanced OCR Integration:**

  Integrate Optical Character Recognition (OCR) to extract and process data from QR codes that contain text, offering more functionality in various use cases.

- **Cross-Platform Support:**

Extend the project to support various platforms (iOS, Android, Web, and Desktop), ensuring widespread usability across different devices.

- **Edge Device Optimization:**

  Focus on optimizing the model further for resource-constrained devices like low-power embedded systems or IoT devices, improving performance and energy efficiency.

- **AI-based QR Code Generation:**

  Develop a system that uses AI to generate unique, secure, and custom QR codes, which can be applied in various business scenarios (e.g., secure ticketing, personalized promotions).

- **Integration with Blockchain for Secure Transactions:**

  Implement blockchain technology to ensure the authenticity and security of QR codes used for transactions, especially in fields like finance and healthcare.

- **User Experience Enhancements:**

  Focus on improving the system’s user interface and experience, such as adding voice feedback or making the scanning process faster and more intuitive.

References

1. Gupta, N., & Gupta, M. (2018). QR Code Detection Using Deep Learning for Real-time Applications. International Journal of Computer Applications, 179(23), 12-18. https://doi.org/10.5120/ijca2018917727

2. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2016.91

3. Huang, L., & Zhang, Z. (2017). Efficient QR Code Detection Using Convolutional Neural Networks. Proceedings of the IEEE Conference on Computer Vision. https://doi.org/10.1109/ICCV.2017.00023

4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://mitpress.mit.edu/books/deep-learning

5. OpenCV. (2020). OpenCV Documentation. Retrieved from https://docs.opencv.org/

6. TensorFlow. (2020). TensorFlow Lite Documentation. Retrieved from [https://www.tensorflow.org/lite](https://www.tensorflow.or

g/lite)

7. Ultralytics. (2020). YOLOv5: State-of-the-art Object Detection. GitHub Repository. Retrieved from https://github.com/ultralytics/yolov5

8. Statista Research Department. (2023). QR Code Usage Trends and Market Growth. Statista. Retrieved from https://www.statista.com/statistics/1181999/qr-code-use-worldwide/

9. MarketsandMarkets. (2022). Global QR Code Payment Market Report. MarketsandMarkets. Retrieved from https://www.marketsandmarkets.com/

10. Techwithtim. (2020). How to Build a QR Code Scanner with Python and OpenCV. Techwithtim. Retrieved from https://techwithtim.net/tutorials/qr-code-scanner-python-opencv/