# QR Code Authentication: Detecting Original vs. Counterfeit Prints

## Report

## 1. Introduction

Counterfeit QR codes pose a significant threat in security-sensitive applications. This project aims to classify QR codes as first prints (original) or second prints (counterfeit) using machine learning and deep learning techniques. The dataset consists of 100 original and 100 counterfeit QR codes, with subtle differences in print quality and microscopic patterns.

### Key Objectives:

• Analyze visual differences between original and counterfeit QR codes.  
• Develop and compare two classification models:  
 - Traditional ML (Random Forest) with handcrafted features.  
 - Deep Learning (CNN) for end-to-end classification.  
• Evaluate model performance and discuss deployment considerations.

## 2. Methodology

### 2.1 Data Exploration & Preprocessing

• Dataset Composition:  
 - 100 first prints (original).  
 - 100 second prints (counterfeit).  
• Preprocessing Steps:  
 - Converted images to grayscale.  
 - Resized to 128×128 pixels for uniformity.  
 - Flattened into 1D feature vectors (16,384 features) for Random Forest.  
 - Reshaped into (128, 128, 1) for CNN input.

### 2.2 Feature Engineering

• Random Forest: Used raw pixel intensities as features.  
• CNN: Automatically extracted hierarchical features via convolutional layers.

### 2.3 Model Architectures

#### A. Random Forest

• Parameters:  
 - n\_estimators=100, random\_state=42.  
• Advantages:  
 - Handles high-dimensional features well.  
 - Interpretable feature importance.

#### B. Convolutional Neural Network (CNN)

Architecture:  
```python  
Sequential([  
 Conv2D(32, (3,3), activation='relu', input\_shape=(128,128,1)),  
 MaxPooling2D((2,2)),  
 Conv2D(64, (3,3), activation='relu'),  
 MaxPooling2D((2,2)),  
 Flatten(),  
 Dense(64, activation='relu'),  
 Dense(1, activation='sigmoid')  
])  
```

Training:  
• Optimizer: Adam.  
• Loss: Binary cross-entropy.  
• Epochs: 10.

## 3. Results & Evaluation

### 3.1 Performance Metrics

Model Performance:  
| Model | Accuracy | Precision | Recall | F1-Score |  
|------------------|----------|-----------|--------|----------|  
| Random Forest | 95% | 0.95 | 0.95 | 0.95 |  
| CNN | 82% | 0.83 | 0.82 | 0.82 |

### 3.2 Misclassification Analysis

• Random Forest: Misclassified 2/40 test samples.  
• CNN: Misclassified 19/40 test samples.  
• Key Insight:  
 - Random Forest outperformed CNN, possibly due to:  
 - Limited training data for deep learning.  
 - CNN may need more layers/data augmentation.

### 3.3 Training Curves (CNN)

Observation:  
• Training accuracy (~94%) >> Validation accuracy (~82%), suggesting overfitting.

## 4. Discussion

### 4.1 Key Findings

• Random Forest achieved 95% accuracy, making it more reliable for this task.  
• CNN underperformed (82%) but could improve with:  
 - More training data.

### 4.2 Deployment Considerations

• Edge Deployment:  
 - Random Forest is lightweight and suitable for mobile/embedded systems.  
• Server-Based API:  
 - CNN can be deployed via cloud APIs if computational resources are available.  
• Robustness:  
 - Test under varying lighting/scanning conditions.

## 5. Conclusion & Future Work

• Best Model: Random Forest (95% accuracy).  
• Improvements:  
 - Experiment with advanced feature extraction .  
 - Try transfer learning (e.g., EfficientNet) for CNN.  
 - Expand dataset with more diverse counterfeit samples.

\*\*GitHub Repository: https://github.com/Shivesh0911/QR\_Code\_Authentication