**QR Code Authentication: Detecting Original vs. Counterfeit Prints**

**1. Introduction**

Counterfeiting is a significant challenge in digital authentication, particularly with QR codes. This assignment focuses on detecting counterfeit QR codes using machine learning and deep learning techniques. The dataset comprises first prints (original QR codes) and second prints (counterfeit versions created by scanning and reprinting first prints).

**2. Data Exploration and Analysis**

**Dataset Overview**

* The dataset consists of QR codes categorized into:
  + **First Prints:** Original QR codes with embedded copy detection patterns (CDPs).
  + **Second Prints:** Scanned and reprinted counterfeit QR codes.

**Feature Analysis**

* **Visual Differences:** Second prints often exhibit degradation in resolution, loss of fine details in CDPs, and variations in contrast.
* **Statistical Summary:**
  + Number of images per class
  + Distribution of pixel intensities
  + Edge detection differences using Sobel filters

**Feature Visualization**

* Histogram of pixel intensities
* Edge detection comparison
* Sample first-print vs. second-print images

**3. Feature Engineering**

To distinguish between first and second prints, we extract features such as:

* **Global Image Properties:**
  + Mean and variance of pixel intensities
  + Histogram of gradients (HOG features)
  + Edge density and contrast levels
* **Local Patterns within QR Codes:**
  + Gabor filters for texture analysis
  + Structural similarity index (SSI) comparison
  + Fourier transform for frequency domain differences

**4. Model Development**

**Approach 1: Traditional Machine Learning (SVM)**

**Preprocessing**

* Convert images to grayscale
* Resize images to a standard size (e.g., 128x128)
* Extract HOG and Gabor filter features

**Model Training**

* Train an **SVM classifier** using the extracted features
* Perform hyperparameter tuning using Grid Search
* Use **5-fold cross-validation** to validate performance

**Approach 2: Deep Learning (CNN)**

**CNN Architecture**

* **Conv2D layers** with ReLU activation
* **MaxPooling layers** to reduce dimensionality
* **Flatten and Dense layers** for classification
* **Softmax activation** for final class prediction

**Training Strategy**

* Use **data augmentation** to improve generalization
* Optimize with **Adam optimizer** and categorical cross-entropy loss
* Train for **20 epochs with early stopping**

**5. Evaluation and Results**

**Performance Metrics**

* **Accuracy:** Measures overall correctness
* **Precision & Recall:** Evaluates class-specific performance
* **F1-Score:** Harmonic mean of precision and recall
* **Confusion Matrix:** Analyzes misclassifications
* SVM Accuracy: 1.0

precision recall f1-score support

0 1.00 1.00 1.00 20

1 1.00 1.00 1.00 20

accuracy 1.00 40

macro avg 1.00 1.00 1.00 40

weighted avg 1.00 1.00 1.00 4

* CNN Accuracy: 0.95

precision recall f1-score support

0 1.00 0.90 0.95 20

1 0.91 1.00 0.95 20

accuracy 0.95 40

macro avg 0.95 0.95 0.95 40

weighted avg 0.95 0.95 0.95 40

* **Misclassification Analysis**
* Identify images where models fail (e.g., low-contrast second prints)
* Use Grad-CAM to visualize CNN decision regions

**6. Deployment Considerations**

**Computational Efficiency**

* **SVM:** Lightweight, suitable for edge devices.
* **CNN:** Requires GPU for fast inference, optimized using TensorFlow Lite.

**Robustness to Scanning Conditions**

* Use adaptive histogram equalization for varying lighting.
* Train with different resolutions to improve generalization.

**Security Implications**

* **Data Privacy:** Encrypt stored QR codes using AES.
* **Model Protection:** Use adversarial training to prevent spoofing attacks.
* **Ethical Considerations:** Ensure unbiased training with diverse QR code samples.
* **Deployment Considerations**
* Deploying the QR code authentication system in a real-world setting requires careful consideration of computational efficiency, robustness under various conditions, and security measures. Below are the key aspects to ensure a successful implementation:
* **1. Computational Efficiency**
* **Optimized Model Deployment:** The deep learning model (CNN) can be deployed using **TensorFlow Lite**, **ONNX**, or **OpenVINO** to optimize it for edge devices like smartphones and barcode scanners.
* **Cloud vs. Edge Processing:** The system can be deployed on a **cloud server** for high computational power or **edge devices** for low-latency, real-time processing in industries like retail and supply chain.
* **Efficient Preprocessing:** Using **grayscale conversion** and **image resizing** before feeding images into the model can reduce computational load while maintaining accuracy.
* **2. Robustness to Different Scanning Conditions**
* **Handling Variability in Lighting & Noise:** Models should be trained with diverse datasets containing QR codes captured under different lighting conditions, angles, and backgrounds.
* **Generalization Across Devices:** The dataset should include scans from multiple smartphones, industrial scanners, and printers to improve model adaptability.
* **Data Augmentation Techniques:** Using **rotation, scaling, and Gaussian noise augmentation** ensures the model remains robust to variations in scanning conditions.
* **3. Security Implications**
* **Preventing Adversarial Attacks:** Attackers may attempt to create modified counterfeit QR codes. To mitigate this:
* Implement **adversarial training** by including slightly altered but real QR codes in the dataset.
* Use **hashing techniques** to compare the embedded copy detection patterns (CDPs) for verification.
* **Encryption & Digital Signatures:**
* Each QR code can have an encrypted signature that gets **verified on the server** before authentication.
* Public-key cryptography can be used to ensure only authorized QR codes pass the authentication check.
* **Secure API Deployment:** If a cloud-based solution is used, implementing **HTTPS, authentication tokens, and rate limiting** will prevent unauthorized access.
* **Final Deployment Strategy**
* **Mobile & Web Integration:** The model can be deployed in **mobile apps (Android/iOS)** or as an API for easy scanning and verification.
* **Real-Time Validation System:** Large-scale deployments (e.g., retail, pharmaceuticals) can use **real-time scanning stations** equipped with AI-powered cameras.
* **Continuous Model Updates:** A feedback loop should be established where the system collects misclassified cases to **retrain and improve accuracy** over time.
* By addressing these factors, the QR code authentication system can be effectively deployed to prevent counterfeiting while maintaining high accuracy and efficiency.

**7. Conclusion**

This study demonstrates that both traditional and deep learning models effectively detect counterfeit QR codes. By CNN model accuracy can be increase but requires higher computational resources. Future work includes improving robustness to noisy QR code prints and optimizing real-time deployment.