QR Code Authentication: Detecting Original vs. Counterfeit Prints

Jaideep Anandkumar Jaiswal

1. Background

- **Task:** Develop a machine learning model to classify QR code images as either "first print" (original) or "second print" (counterfeit).
- **Importance:** Essential for anti-counterfeiting systems to verify the authenticity of printed QR codes.
- **Dataset Description:**
  - **First prints:** 100 images (original QR codes with embedded Copy Detection Patterns – CDPs)
  - **Second prints:** 100 images (counterfeit versions created by scanning and reprinting first prints)
  - **Total images:** 200

2. Data Exploration and Analysis

- **Visual Observations:**
  - The middle part of the second print is noticeably darker and denser compared to the first print.
  - The first print exhibits higher overall brightness.
- **Computed Statistics:**
  - **Overall Black Ratio (Ink Coverage):**
    * First Print: 0.5534 (55.34%)
    * Second Print: 0.5957 (59.57%)
    * **Conclusion:** The second print has ∼4% more ink coverage, indicating a darker appearance due to scanning and reprinting.
  - **Center Black Ratio:**
    * First Print: 0.4753 (47.53%)
    * Second Print: 0.5819 (58.19%)
    * **Conclusion:** The center of the second print is about 10.66% darker, highlighting pronounced central darkening.
  - **GLCM Contrast:**
    * First Print: 280.2138

    - Second Print: 142.0851
    - **Conclusion:** Higher contrast in the first print indicates greater local intensity variation; lower contrast in the second print suggests a smoother or more uniform intensity.
  - **GLCM Homogeneity:**
    - First Print: 0.4399
    - Second Print: 0.2971
    - **Conclusion:** The first print has a more uniform texture, while the second print shows more irregularity.
  - **Texture and Wavelet Analysis:**
    - **LBP & GLCM Patterns:** Second print shows a higher uniform bin in LBP and lower homogeneity/ASM in GLCM, indicating altered local texture due to reprinting.
    - **Wavelet Analysis:** The second print may lose some high-frequency details (observed as a lower HL ratio), leading to reduced sharpness.

3. Feature Engineering and Data Preprocessing

- **Extracted Features:**
  - **Brightness Features:** Overall and center brightness (average pixel values).
  - **Dark Pixel Ratios:** Overall black ratio and center black ratio computed after thresholding.
  - **Texture Features:** GLCM-based features (contrast, homogeneity, energy, correlation) and Local Binary Patterns (LBP).
  - **Wavelet Features:** Sub-band energy calculations to capture high-frequency details.
- **Data Preprocessing for Traditional ML Approach:**
  - **Image Resizing:** All images were resized to **224x224 pixels** to ensure uniformity.
  - **Grayscale Conversion:** Images were converted to grayscale using OpenCV to simplify texture analysis.
  - **Histogram Equalization:** Applied (if necessary) to enhance image contrast before feature extraction.
  - **Feature Extraction:** Numerical features such as GLCM, LBP, and color statistics were computed for each image.
  - **Normalization:** Features were normalized using Min-Max scaling to maintain a consistent range across features.
  - **Train-Test Split:** Data was split into training (80%) and testing (20%) sets using stratified sampling to preserve the class distribution.
- **Approach:** Both global (entire image) and local (center region) features were extracted to capture subtle differences between original and counterfeit prints.

4. Model Development

A. Traditional ML Pipeline

- **Methodology:** Handcrafted features extracted from images used for classification.
- **Models Implemented:**
  - Support Vector Machine (SVM)
  - Logistic Regression
  - Naïve Bayes
  - Decision Tree
  - K-Nearest Neighbors (KNN)
  - Random Forest
  - XGBoost (Achieved 100% accuracy)
  - AdaBoost
  - Gradient Boosting
- **Key Result:** XGBoost achieved 100% accuracy with an inference time of approximately 0.007 seconds per sample.

B. Deep Learning Approach – Custom CNN

- **Architecture:** Multiple convolutional layers (with ReLU activations) followed by MaxPooling, a Flatten layer, and Dense layers with Dropout.
- **Performance:** Validation accuracy of approximately 95.2%.
- **Computational Details:**
  - Inference Time: ∼0.035 seconds per sample.
  - Memory Usage: ∼52 MB during inference.
  - Predictions per Second: ∼28 per second.

C. Deep Learning Approach – InceptionV3-Based Model (Transfer Learning)

- **Architecture:** Pre-trained InceptionV3 (using ImageNet weights; base frozen) followed by GlobalAveragePooling2D and a Dense classification layer.
- **Performance:** Validation accuracy of approximately 97.8%.
- **Computational Details:**
  - Inference Time: ∼0.024 seconds per sample.
  - Memory Usage: ∼68 MB during inference.
  - Predictions per Second: ∼42 per second.

5. Evaluation and Results

- **Metrics Reported:** Accuracy, Precision, Recall, F1-Score, and Confusion Matrices.
- **Key Findings:**
  - **Traditional ML (XGBoost):** 100% accuracy; inference time of ~7.05 ms per sample.
  - **Custom CNN:** Approximately 95.2% accuracy; inference time of ~35 ms per sample.
  - **InceptionV3-Based Model:** Approximately 97.8% accuracy; inference time of ~24 ms per sample.
- **Visualizations:** Training curves, confusion matrices, and sample predictions were used to evaluate and compare model performance.

6. Deployment Considerations

A. Traditional ML Model (XGBoost)

- Inference Time: ~0.007 seconds per sample.
- Memory Usage: Minimal (compact feature vectors, approx. 0.5–1 MB for test features).
- Predictions per Second: ~141.
- **Implication:** Highly efficient and ideal for real-time applications on edge devices.

B. Custom CNN Model

- Inference Time: ~0.035 seconds per sample.
- Memory Usage: ~52 MB during inference.
- Predictions per Second: ~28.
- **Implication:** Suitable for environments with moderate hardware resources.

C. InceptionV3-Based Model

- Inference Time: ~0.024 seconds per sample.
- Memory Usage: ~68 MB during inference.
- Predictions per Second: ~42.
- **Implication:** Balances high accuracy with computational efficiency. Further optimization via quantization or pruning is recommended for deployment on edge devices.
- **Additional Considerations:**
  - Robustness: Test under varying scanning conditions (lighting, angles).
  - Security: Implement model encryption and secure input handling to mitigate adversarial risks.

7. Conclusion and Future Work

- **Summary:**
  - Both traditional ML and deep learning approaches effectively classify QR codes.
  - The InceptionV3-based model provided the best overall performance with high accuracy and reasonable computational efficiency.
  - Handcrafted features (brightness, GLCM, LBP, wavelet) were crucial for the traditional ML models.
- **Deployment Readiness:**
  - All models demonstrate low inference times and acceptable memory usage, making them viable for real-time applications and edge deployment.
- **Future Work:**
  - Validate the models on a more diverse dataset to further improve robustness.
  - Explore additional model optimization techniques (quantization, pruning) for enhanced deployment efficiency.
  - Enhance security measures to protect against adversarial attacks and ensure model integrity.