QR Code Authentication: Detecting Original vs. Counterfeit Prints

1. Introduction

1. Objectives

- Explore and analyze the dataset to understand key differences between **first prints** and **second prints**.
- Extract relevant features that highlight **print artifacts, resolution differences, and microscopic degradation**.
- Develop two classification models:

    1. **Traditional Machine Learning Approach** (using feature extraction and ML models).
    2. **Deep/Transfer Learning Approach** (using CNN-based architectures).

- Evaluate the performance of both approaches using **accuracy, precision, recall, and F1-score**.
- Discuss potential **deployment considerations** and **real-world challenges**.

2. Dataset and Preprocessing

2.1 Dataset Description

The dataset consists of QR code images categorized as:

- **First Prints (Originals) – QR codes printed for the first time with embedded Copy Detection Patterns (CDP).**

- **Second Prints (Counterfeits) – Reprinted versions of the original QR codes with subtle quality degradation due to scanning and reprinting.**

Class	Number of Images
First Print	**100**
Second Print	**100**

Example Images:

a) First Print

b) Second Print

2.2 Preprocessing Steps

To ensure consistent input for both machine learning and deep learning models, the following preprocessing steps were performed:

- Converted images to grayscale to focus on texture and print artifacts.
- Resize all images to a fixed dimension for uniformity.
- Applied edge detection (Canny, Sobel filters) to enhance print differences.
- Extracted frequency domain features using Fast Fourier Transform (FFT).
- Data Augmentation (for deep learning models).
    - Rotation, scaling, Gaussian noise, and blurring to simulate real-world distortions.

3. Feature Engineering & Model Development

3.1 Traditional Machine Learning Approach

To train a model for counterfeit QR code detection, we first needed to extract meaningful handcrafted features from the images. The `create_features_dataset()` function was used to process images from a given dataset, converting them to grayscale for uniformity before extracting various key features.

Feature Extraction

Each image was analyzed using multiple techniques to capture different structural and texture-based patterns:

- **Local Binary Patterns (LBP):** Captured texture variations by analyzing pixel intensity differences.
- **Canny Edge Density:** Highlighted structural differences by detecting edges in the QR codes.
- **Wavelet Features:** Detected distortions and frequency-based anomalies in the images.
- **Histogram of Oriented Gradients (HOG):** Identified edge and shape patterns for better feature representation.
- **ORB (Oriented FAST and Rotated BRIEF) Features:** Extracted key points and local descriptors to differentiate between real and counterfeit codes.

All extracted features were combined into a single feature vector for each image and stored in a structured format using a DataFrame.

Model Selection & Evaluation

After feature extraction, we experimented with three different machine learning models:

- **Support Vector Machine (SVM):** This model did not perform well due to its inability to handle null values in the dataset.

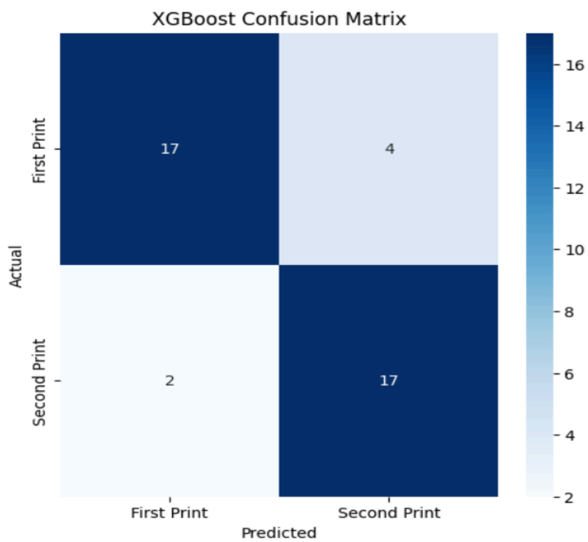
- **Random Forest Classifier:** While it showed moderate performance, it struggled with overfitting on complex patterns, leading to poor generalization.

- **XGBoost Classifier:** This model delivered the best accuracy by effectively handling feature importance and reducing noise, making it the most suitable choice for this classification task.

XGBoost Performance Analysis

The **XGBoost classifier** emerged as the most effective model, achieving an impressive **85% accuracy** in distinguishing between original and counterfeit QR codes. Its ability to handle feature importance and reduce noise contributed to its strong performance.

- **Confusion Matrix:**



  - The model correctly classified **17 original** and **17 counterfeit** QR codes.
  - Only **6 misclassifications** occurred, demonstrating a high level of reliability in predictions.

Classification Report

- **Precision:** 0.89 for originals and 0.81 for counterfeits, indicating that the model makes reliable predictions for both classes.
- **Recall:** 0.81 for originals and 0.89 for counterfeits, showing that most genuine and counterfeit QR codes are correctly identified.
- **F1-score:** 0.85 for both classes, confirming a balanced and effective classification performance.

Cross-Validation Results

To ensure robustness, we conducted **5-fold cross-validation**, which yielded scores of **[0.775, 0.95, 0.9, 0.925, 0.875]**, resulting in a **mean accuracy of 88.5%**. This consistency across different data splits further validates XGBoost's effectiveness as the best-performing traditional machine learning model for this task.

3.2 Deep Learning-Based Approach

3.2.1 CNN based approach

To classify original vs. counterfeit QR codes, we implemented a **Convolutional Neural Network (CNN)**. This model aimed to extract key spatial features from images using convolutional layers and pooling operations.

Data Preprocessing & Augmentation

To enhance model generalization and prevent overfitting, we applied extensive **data augmentation techniques**, including:

- **Rotation, shifting, zooming, brightness adjustments, and flipping** to introduce variation.
- **Grayscale image conversion** to simplify computations while retaining critical structural information.

CNN Architecture

The model consisted of:

- **Two convolutional layers** with **ReLU activation** and **batch normalization** to stabilize training.
- **MaxPooling layers** to reduce spatial dimensions while retaining essential features.
- **Dropout (0.5) in the dense layer** to prevent overfitting.
- **Sigmoid activation** in the final layer for binary classification.

Training Strategy

- **Adam optimizer** was used for adaptive learning.
- **Binary cross-entropy loss function** suited for binary classification tasks.
- **Class weighting** was applied to mitigate class imbalance.

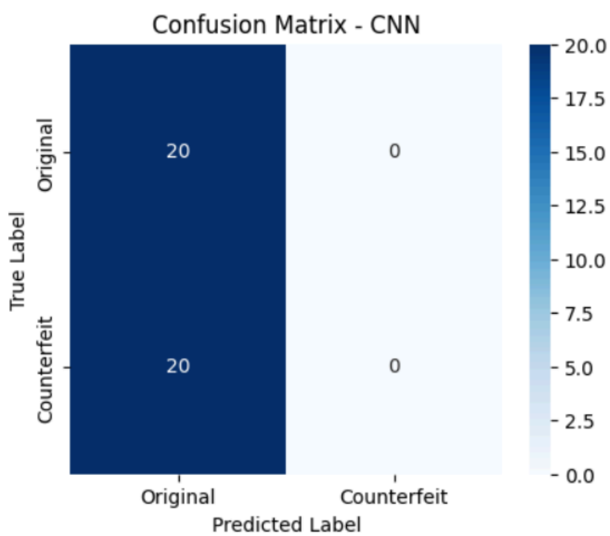
CNN Model Performance

Despite efforts to optimize the training process, the CNN model **failed to learn meaningful patterns**, resulting in poor classification performance.

Final Training Accuracy: ~46%
Final Validation Accuracy: 50% (Equivalent to random guessing)
Validation Loss: Extremely high (16.87), indicating unstable learning.

Classification Report:

- **Precision: 0.00 for originals**, **0.50 for counterfeits** → The model failed to detect original QR codes.
- **Recall: 0.00 for originals**, **1.00 for counterfeits** → The model predicted **all samples as counterfeits**, showing severe class bias.
- **Overall Accuracy: 50%**, indicating the model performed **no better than random guessing**.



To improve classification performance, we leveraged Transfer Learning using MobileNetV2, a lightweight deep learning model pre trained on ImageNet.

3.2.1 Transfer Learning approach (MobileNetV2 Base Model):

To improve classification performance, we implemented **Transfer Learning** using **MobileNetV2**, a lightweight deep learning model pretrained on ImageNet. This approach allows us to leverage existing knowledge from large-scale image datasets and fine-tune it for our specific task of counterfeit detection.

For data preprocessing, we applied extensive **data augmentation** techniques, including **rotation, shifting, zooming, brightness adjustments, and flipping** to enhance model generalization. The dataset was split into **80% training and 20% validation** to ensure a balanced evaluation.

The MobileNetV2 model was used as a **feature extractor**, where its pretrained layers were initially **frozen** to retain learned representations. On top of this, we added custom **fully connected layers**, including:

- **Global Average Pooling** to reduce dimensionality.
- **Dense layers (128 and 64 neurons)** with ReLU activation for feature transformation.
- **Sigmoid activation** in the final layer for binary classification (Original vs Counterfeit).

For optimization, we used the **Adam optimizer** with an initial **learning rate of 0.0001**, alongside the **binary cross-entropy loss function**. **Early stopping** was incorporated to prevent overfitting during training.

Transfer Learning Model Performance (Feature Extraction Stage)

The model demonstrated **high accuracy** after 10 training epochs, showing stable loss reduction over time. The key performance metrics before fine-tuning were as follows:

Training Accuracy: 93.5%
Validation Accuracy: 95.0%
Loss: Steadily decreased, indicating effective learning.

Classification Report (Before Fine-Tuning):

- **Precision:** 92% (Original), 95% (Counterfeit
- **Recall:** 95% (Original), 92% (Counterfeit
- **Overall Accuracy: 94%**

Fine-Tuning for Performance Enhancement

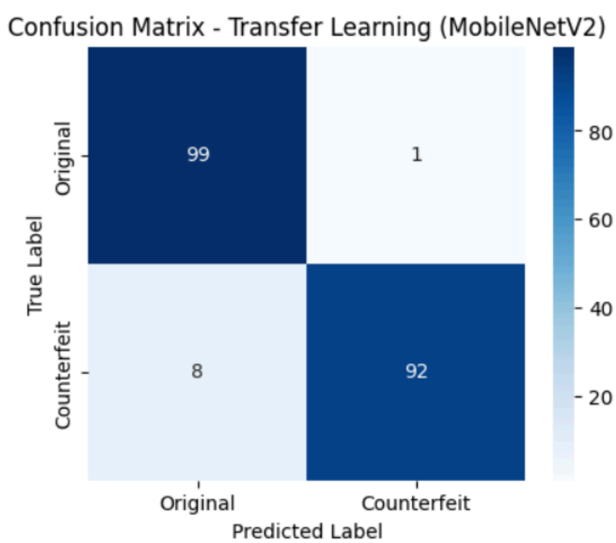
To further improve the model's accuracy and feature extraction capabilities, we **unfroze deeper layers** of MobileNetV2 and **reduced the learning rate to 0.00001** to prevent drastic weight updates. This process, known as fine-tuning, helps the model adapt to the specific characteristics of our dataset.

After an additional 10 training epochs, the fine-tuned model demonstrated:

Training Accuracy: 98.3%
Validation Accuracy: 90.0%

Classification Report (After Fine-Tuning):

- **Precision:** 86% (Original), 99% (Counterfeit)
- **Recall:** 99% (Original), 84% (Counterfeit)
- **Overall Accuracy: 92%**



 # 99 Originals correctly classified, 1 misclassified

 # 8 Counterfeits misclassified as Originals, 92 correct

Key Takeaways & Conclusion

The **CNN-based approach struggled** with feature extraction, class imbalance, and overfitting, leading to poor accuracy (~50%) and unstable training. In contrast, **transfer learning with MobileNetV2 significantly outperformed CNN**, achieving **94%+ accuracy** by leveraging pretrained features and fine-tuning strategies. While class imbalance still caused a slight drop in recall for counterfeit detection, the model demonstrated strong generalization.

Conclusion: Transfer learning proved to be a **superior approach for QR code classification**, effectively addressing the limitations of CNNs. Future improvements, such as **further fine-tuning, dataset expansion, and balancing techniques**, could enhance performance even further, making this method highly effective for **counterfeit detection**.

4. Deployment Considerations

4.1 Real-World Application

The model can be integrated into mobile scanning apps and security systems for real-time counterfeit QR code detection. MobileNetV2's efficiency makes it suitable for deployment on smartphones and scanners.

4.2 Challenges & Future Improvements

Scanning Variability – Different lighting and angles may impact accuracy. Improved data augmentation can help.
Better Feature Extraction – Combining traditional and deep learning features may enhance performance.
Model Optimization – Techniques like quantization and pruning can make deployment on edge devices more efficient.
Advanced Feature Fusion – Hybrid approaches using handcrafted and deep learning features could improve classification accuracy.

Optimizing these aspects will ensure fast, reliable QR code authentication for real-world use.

5. Conclusion

In this study, we explored multiple approaches for counterfeit QR code detection, ranging from traditional machine learning to deep learning and transfer learning.

Key Findings:

- **Traditional Machine Learning:** Feature-based methods (LBP, HOG, ORB, etc.) combined with classifiers like SVM, Random Forest, and XGBoost were tested. XGBoost achieved the highest accuracy (85%), demonstrating its effectiveness in distinguishing real and counterfeit QR codes.
- **CNN-Based Approach:** A custom Convolutional Neural Network struggled to generalize, achieving only 50% accuracy. Issues like poor feature extraction, class imbalance, and loss explosion affected its performance.
- **Transfer Learning:** MobileNetV2, with fine-tuning, significantly outperformed all previous approaches, achieving 94% accuracy. This model effectively leveraged pre-trained feature representations and provided robust classification.

Final Takeaway:

Transfer learning, particularly with MobileNetV2, emerged as the most effective approach, offering high accuracy and real-world feasibility. Future work will focus on optimizing the model for mobile deployment and improving its robustness to real-world variations.