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

**1. Data Exploration and Analysis**

**Dataset Overview**

This assignment focuses on classifying QR code images as either original (“First Print”) or counterfeit (“Second Print”). Each image contains a QR code with subtle print quality and microscopic pattern differences introduced during scanning and reprinting. Specifically:

* **First Print**: 100 images of original QR codes with embedded copy detection patterns (CDPs).
* **Second Print**: 100 images of counterfeit QR codes created by scanning and reprinting the originals.

**Dataset Composition**

* **Total Images**: 200
* **First Print**: 100 images
* **Second Print**: 100 images

**Data Splitting Strategy**

To ensure balanced representation of both classes, a 70–30 split was used:

* **Training Data**: 70% of First Print and 70% of Second Print images
* **Testing Data**: 30% of First Print and 30% of Second Print images

This split provided a roughly equal distribution of each class in both sets.

**2. Feature Engineering**

**Extracted Features**

A combination of global and local features was extracted to capture potential differences between original and counterfeit QR codes:

1. **Brightness**
   * Average pixel intensity to detect inconsistencies in printing or scanning.
2. **Edge Density**
   * Ratio of edge pixels (detected via Canny edge detection) to total pixels, indicating variations in image sharpness or reprint artifacts.
3. **Local Binary Pattern (LBP) Mean & Std**
   * Captures local texture variations by analyzing pixel-wise intensity differences.
4. **GLCM (Gray-Level Co-occurrence Matrix) Features**
   * **Contrast**: Measures intensity variation.
   * **Correlation**: Indicates how pixel intensities correlate with neighbors.
   * **Energy**: Reflects the uniformity or smoothness of texture.
   * **Homogeneity**: Shows how similar pixel intensities are within local regions.
5. **SIFT Keypoints**
   * Counts the number of distinct local features, potentially revealing differences caused by reprinting processes.

**3. Traditional Machine Learning: Support Vector Machine (SVM)**

After extracting the features, a linear SVM was trained using a 70–30 split for each class (70% for training, 30% for testing). The SVM achieved:

* **Accuracy**: 86.67%
* **Precision**:
  + Class 0 (First Print): 0.84
  + Class 1 (Second Print): 0.89
* **Recall**:
  + Class 0 (First Print): 0.90
  + Class 1 (Second Print): 0.83
* **F1 Score**:
  + Class 0 (First Print): 0.87
  + Class 1 (Second Print): 0.86

**Confusion Matrix**

|  | **Predicted 0** | **Predicted 1** |
| --- | --- | --- |
| **Actual 0** | 27 | 3 |
| **Actual 1** | 5 | 25 |

The SVM performed well, likely due to the discriminative power of hand-engineered features and a relatively simple dataset size.

**4. Deep Learning Approach: EfficientNetB0**

To explore a deep learning alternative, images were fed into a pretrained EfficientNetB0 model (with additional fine-tuning layers). Despite experimenting with data augmentation and label smoothing, the best result was:

* **Accuracy**: 58.33%
* **Precision**:
  + Class 0 (First Print): 0.62
  + Class 1 (Second Print): 0.56
* **Recall**:
  + Class 0 (First Print): 0.43
  + Class 1 (Second Print): 0.73
* **F1 Score**:
  + Class 0 (First Print): 0.51
  + Class 1 (Second Print): 0.64

**Confusion Matrix**

|  | **Predicted 0** | **Predicted 1** |
| --- | --- | --- |
| **Actual 0** | 13 | 17 |
| **Actual 1** | 8 | 22 |

The deep learning model struggled to surpass the SVM’s performance. Pretrained networks often excel with large, varied datasets of natural images, but highly structured QR codes can be challenging unless many domain-specific samples are provided.

**5. Performance Comparison**

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| --- | --- | --- | --- | --- |
| **SVM** | 86.67% | 0.87 | 0.87 | 0.87 |
| **EfficientNetB0** | 58.33% | 0.59 | 0.58 | 0.57 |

Clearly, the SVM outperformed the deep learning approach in this setting.

**6. Misclassification Analysis**

* **SVM Misclassified Samples**: 8 out of 60
* **Neural Network Misclassified Samples**: 25 out of 60

Observations:

1. **SVM**
   * Used feature-based differences effectively.
   * Achieved high accuracy and balanced precision/recall.
2. **Neural Network**
   * Likely suffered from overfitting due to small dataset size.
   * Fine-tuning a pretrained model did not adequately capture QR-specific patterns.

**7. Additional Challenges and Experiments**

**Simple CNN with Data Augmentation**  
Attempted to train a basic CNN from scratch using rotated, flipped, and brightness-adjusted images. However, data augmentation was insufficient to overcome the limited dataset size, leading to poor generalization.

**Artificial Neural Network (ANN) with Extracted Features**  
Used the same set of engineered features as for SVM. This ANN achieved performance worst than SVM also required longer training and more careful hyperparameter tuning.

**Decision Tree Classifier**  
Explored for interpretability, but overfitted easily due to the small dataset, resulting in lower accuracy than SVM.

**Challenges Encountered**

* **Limited Dataset Size**: Deep learning models typically require large, diverse datasets to avoid overfitting, which was not feasible here.
* **Pretrained Model Issues**: Models like EfficientNetB0 are optimized for natural images and do not always adapt well to highly structured data, such as QR codes.
* **Domain-Specific Features**: Manually engineered features (like GLCM or SIFT) appear to better capture small print differences in QR codes than generalized deep features.

**8. Real-World Deployment Considerations**

**Computational Efficiency**

* The SVM model is lightweight and can run efficiently on CPUs or edge devices.
* Deep learning requires more computational resources (e.g., GPUs) and may not be practical for real-time offline scenarios.

**Robustness to Different Scanning Conditions**

* Data augmentation helps neural networks learn invariances but requires more samples.
* SVM models can be retrained or combined with domain-specific features to improve resilience against varying lighting and scanning conditions.

**Security Implications**

* Feature-based approaches often have better explainability, aiding fraud detection.
* Deep learning architectures might be susceptible to adversarial attacks, where minor input perturbations result in misclassification.

**9. Conclusion**

This assignment explored both traditional machine learning and deep learning approaches for authenticating QR codes:

1. **SVM** on handcrafted features performed best, achieving ~86.67% accuracy.
2. **EfficientNetB0** fine-tuning lagged behind at ~58.33% accuracy, possibly due to the small dataset and the model’s reliance on large-scale natural image training.
3. Additional experiments (simple CNN, ANN, decision trees) highlighted the importance of dataset size and domain-specific features.

**Key Takeaways**:

* **Feature-Based Methods**: Highly effective for structured data like QR codes, especially when the dataset is limited.
* **Deep Learning**: May require significantly larger datasets or more specialized architectures to capture nuanced print artifacts.
* **Future Work**: Could involve creating a hybrid solution that combines handcrafted features (GLCM, SIFT) with a smaller CNN for robust performance, along with increasing dataset diversity through additional data collection or more advanced augmentation techniques.