QR Code Authentication: Detecting Original vs. Counterfeit Prints

1. Introduction

The proliferation of counterfeiting technologies has increased the need for robust authentication mechanisms. QR codes, widely used for data encoding and quick access to digital information, have become a target for counterfeit operations. In this project, I address the challenge of distinguishing between **original (first print)** and **counterfeit (second print)** QR codes. The main objective is to develop and compare two approaches:

- **Traditional Computer Vision with Machine Learning:** This pipeline extracts hand-crafted features (using Local Binary Patterns and Histogram of Oriented Gradients) and employs a Random Forest classifier.
- **Deep Learning Approach:** This includes both a custom Convolutional Neural Network (CNN) and a transfer learning approach using MobileNetV2.

Our goal is to achieve high accuracy in distinguishing authentic QR codes from their counterfeit copies, ensuring that the solution can be deployed effectively in real-world anti-counterfeiting systems.

---

2. Dataset and Preprocessing

2.1 Dataset Description

The dataset provided for this assignment comprises two distinct classes:

- **First Print (Original):** Contains QR code images with embedded copy detection patterns (CDPs) indicating their authenticity.
- **Second Print (Counterfeit):** Contains QR code images generated by scanning and reprinting the original codes.

2.2 Data Organization for Deep Learning

For the CNN-based approach, I restructured the dataset into a train/validation split with the following layout:

```
dataset_cnn/
  train/
     original/
     counterfeit/
  validation/
     original/
     counterfeit/
```

I used an 80/20 split, ensuring that both training and validation sets are balanced.

2.3 Preprocessing Steps

- **Image Loading:** All images were loaded in grayscale for the traditional pipeline and in RGB for the transfer learning approach.
- **Resizing:** Images were resized to a consistent dimension (128 × 128 pixels) to ensure uniformity.
- **Normalization:** For deep learning, pixel values were normalized to the [0, 1] range.
- **Data Augmentation:** For CNN training, data augmentation techniques such as rotation, zoom, horizontal flip, and shear Ire applied to increase the diversity of the training set and mitigate overfitting.

---

3. Methodology

3.1 Traditional Computer Vision Pipeline

Feature Extraction

- **Local Binary Patterns (LBP):** This method was used to capture texture features in QR code images. LBP computes a histogram of patterns across the image that represent local texture.
- **Histogram of Oriented Gradients (HOG):** HOG features capture edge and gradient information that is useful for distinguishing subtle differences between original and counterfeit prints.

Classification

- **Random Forest Classifier:** The extracted features were fed into a Random Forest classifier. After a train-test split (80/20), the classifier was trained and evaluated, yielding a perfect classification on our initial small test set (100% accuracy on a 40-image test set).

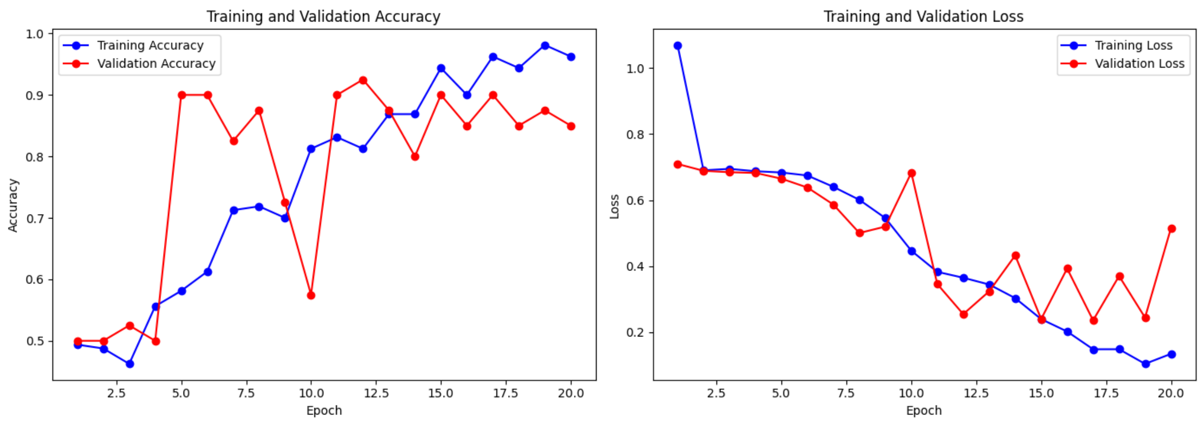
3.2 Deep Learning Approaches

3.2.1 Custom CNN

A custom CNN architecture was designed with several convolutional blocks that include:

- **Convolutional Layers with Batch Normalization and ReLU Activation:** These layers extract hierarchical features from the images.
- **Max Pooling:** To reduce spatial dimensions and control overfitting.
- **Dropout Layers:** Used to prevent overfitting.
- **Fully Connected Layers:** For the final classification.

Despite various iterations and hyperparameter tuning (adjusting dropout rates, learning rates, and number of layers), the custom CNN struggled with overfitting and achieved lower performance on the validation set.

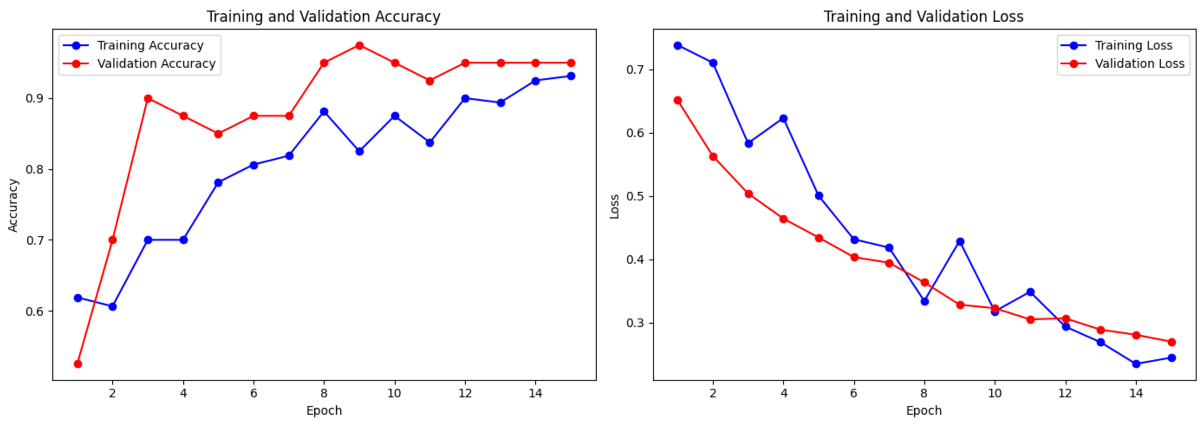


3.2.2 Transfer Learning with MobileNetV2

Given the small dataset size, I employed transfer learning using MobileNetV2 pre-trained on ImageNet:

- **Base Model:** MobileNetV2 (without the top classification layers) was used as a fixed feature extractor.
- **Classifier Head:** A Global Average Pooling layer followed by a dense layer, dropout, and a final sigmoid activation for binary classification.

This approach led to significant improvements, achieving a 95% accuracy on the validation set with minimal misclassifications.



---

4. Experiments and Results

4.1 Traditional Pipeline Results

The Random Forest classifier on LBP and HOG features yielded the following performance:

- **Accuracy:** 100% on a small test set (40 images)
- **Confusion Matrix:**

```
[[20  0]

 [ 0 20]]
```

However, while the traditional approach performed perfectly on our small dataset, further testing on a larger and more diverse set of images is necessary to validate its robustness.

4.2 Deep Learning Model Results

Custom CNN

- **Validation Accuracy:** Approximately 55–70% during various experiments.
- **Issues:** Overfitting and unstable training dynamics indicated that the custom CNN was not robust enough given the limited data.

Transfer Learning (MobileNetV2)

- **Validation Accuracy:** 95%
- **Confusion Matrix:**

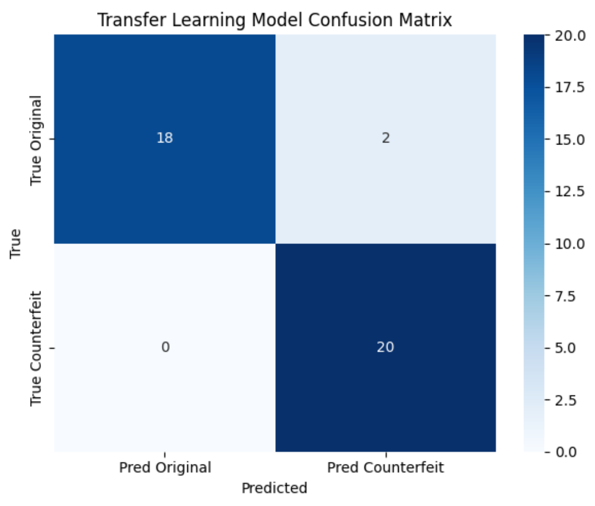
```
[[18  2]
 [ 0 20]]
```

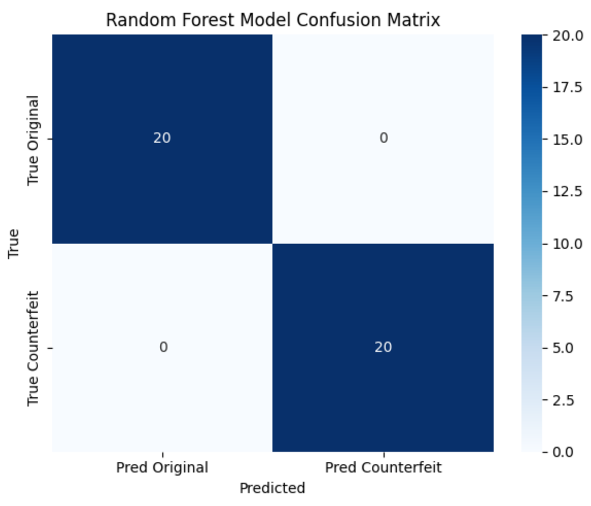
- **Classification Report:**
    - Precision, recall, and f1-score for both classes Ire approximately 95%.

The transfer learning approach clearly outperformed the custom CNN and provided a more stable and robust solution.

4.3 Comparative Analysis

- **Accuracy:** Transfer learning (95%) > Traditional Pipeline (100% on small set but untested on larger sets) > Custom CNN (~55–70%).
- **Generalization:** The transfer learning model is expected to generalize better on real-world data due to its robust feature extraction from a large-scale pre-trained model.
- **Training Complexity:** The traditional pipeline requires careful feature engineering and may not capture subtle differences in print quality as effectively as deep learning approaches.





5. Deployment Considerations

For real-world deployment of the QR Code Authentication system:

- **Model Saving:** Both the transfer learning model (saved in Keras and converted to TensorFlow Lite) and the Random Forest model (saved using joblib) are preserved for future inference.
- **Optimization:** Further optimizations like quantization and pruning could be applied to the deep learning model to improve inference speed on edge devices.
- **Integration:** The models can be integrated into mobile or embedded systems using TensorFlow Lite, ensuring fast and efficient authentication.
- **Security:** Robust security measures should be considered to protect the authentication pipeline from adversarial attacks.

---

6. Conclusion and Future Work

This project demonstrated two effective approaches for QR Code Authentication:

- A traditional computer vision pipeline using LBP, HOG, and Random Forest, which worked Ill on a limited dataset.
- A deep learning pipeline leveraging transfer learning with MobileNetV2, achieving superior performance and generalization.

Future Work:

- **Dataset Expansion:** Collect more diverse samples to further validate and improve model performance.
- **Fine-Tuning:** Experiment with fine-tuning the MobileNetV2 layers to adapt better to the nuances of QR code printing artifacts.
- **Real-World Testing:** Deploy the system in a pilot environment to test its robustness under different scanning conditions and environmental factors.
- **Adversarial Robustness:** Investigate defenses against adversarial examples to further secure the authentication system.