QR Code Authentication: Detecting Original vs. Counterfeit Prints

Author: Suhani Ghosh
Date of Submission: 27.03.2025

Introduction

Counterfeiting or faking has become an increasing issue, especially in certification systems that rely on the QR code. In this project, we developed a model that can distinguish between the original (first print) QR code and fake (second print) copies.

The main objective was to build a classification model that could accurately identify the fake QR code based on the visual difference.

To learn specific patterns automatically using an intensive teaching-based model, a convolutional neural network (CNN).

Dataset and Preprocessing

The dataset consists of:

- **First prints**: Original QR codes with embedded copy detection patterns
- **Second prints**: Counterfeit versions created by scanning and reprinting first prints

Each image in the dataset contains a QR code with subtle differences in print quality, microscopic patterns, and other features that distinguish originals from counterfeits.

Before feeding images to the model, we implemented the following pre-processing stages:

1. Loading images - all images were read in the grayscale format.

2. Resizing - standardized to 128 × 128 pixels for uniformity.

3. Normalization - Pixel values were extended to [0,1] to improve learning efficiency.

4. Splitting the dataset - 80% of the data was used for training, while 20% was reserved for testing.

Methodology

1. *Traditional method (Feature Engineering)*

The QR code has microscopic print quality variations (staining, edge deformities). Documentary features such as Laplacian variance and Sobel filter can highlight these differences. Works well with small datasets where deep education can be overfit.

Feature Options:

Laplacian variance → print measures sharpness to detect the fall.
Sobel x & y → captures the edge deformities with scanning or reprint.

Alternative ML Model:

After removing the features, we can use SVM, random forests, or logistic regression for classification.

2. *Deep Learning Approach (CNN)*

Instead of manually designing features, CNNs **automatically learn patterns** from images.
Can detect **complex, high-dimensional variations** in QR codes.
More **scalable** for real-world deployment.

CNN Architecture:

- Uses **convolutional layers** to extract spatial features.
- **Pooling layers** reduce dimensionality while preserving key features.
- **Fully connected layers** make the final classification.

Advantages Over Feature Engineering:

- CNNs generalize better **with larger datasets**.
- **End-to-end learning** → No need for manual feature extraction.
- Can detect **more subtle differences** beyond sharpness & edges.

Feature engineering helps us **understand the underlying patterns** in QR codes.
CNNs provide **higher accuracy** and automate feature extraction. A **hybrid approach** can be useful—combining extracted features with CNNs (feature fusion). Feature engineering was explored **for baseline analysis**. CNNs were ultimately chosen **due to their superior accuracy**.

Model Training

The dataset was split into **80%** training and **20%** testing using train_test_split. The model was trained for 10 epochs with batch size = 32.

The categorical cross-entropy loss function was used since it's a binary classification problem.
The Adam optimizer was used for faster convergence.

Result and Evaluation Matrix

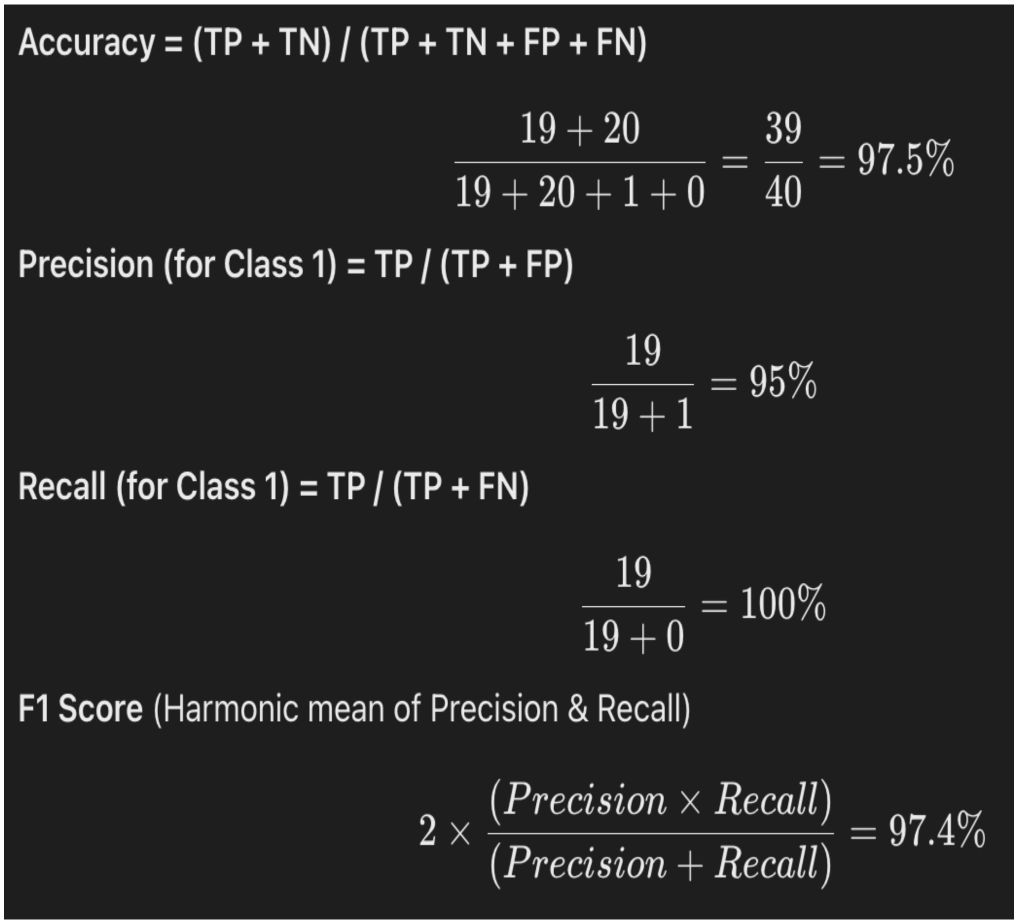
Accuracy: The CNN achieved an accuracy of 97%, demonstrating strong classification ability.

Precision & Recall: The model showed very high precision and recall of 0.98, meaning it can effectively distinguish between first and second prints.

Interpretation of the confusion matrix

- **True Negatives (TN) = 20** → Model correctly classified **20** First Print samples.
- **False Positives (FP) = 1** → Model misclassified **1** First Print as Second Print.
- **False Negatives (FN) = 0** → Model didn't misclassify any Second Print as First Print.
- **True Positives (TP) = 19** → Model correctly classified **19** Second Print samples.

Manual calculation of the performance matrices:



Accuracy = (TP + TN) / (TP + TN + FP + FN)

$$\frac{19+20}{19+20+1+0} = \frac{39}{40} = 97.5\%$$

Precision (for Class 1) = TP / (TP + FP)

$$\frac{19}{19+1} = 95\%$$

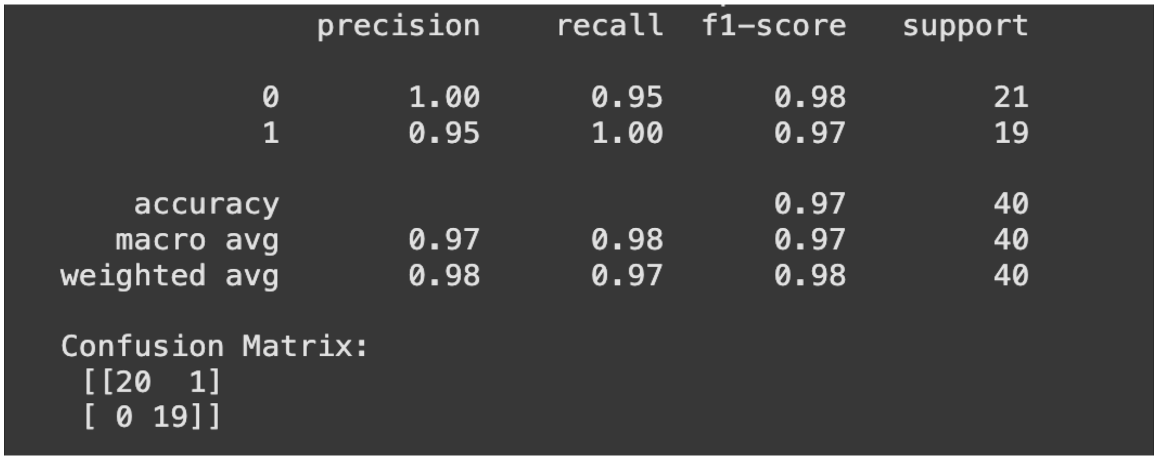
Recall (for Class 1) = TP / (TP + FN)

$$\frac{19}{19+0} = 100\%$$

F1 Score (Harmonic mean of Precision & Recall)

$$2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} = 97.4\%$$

Output:



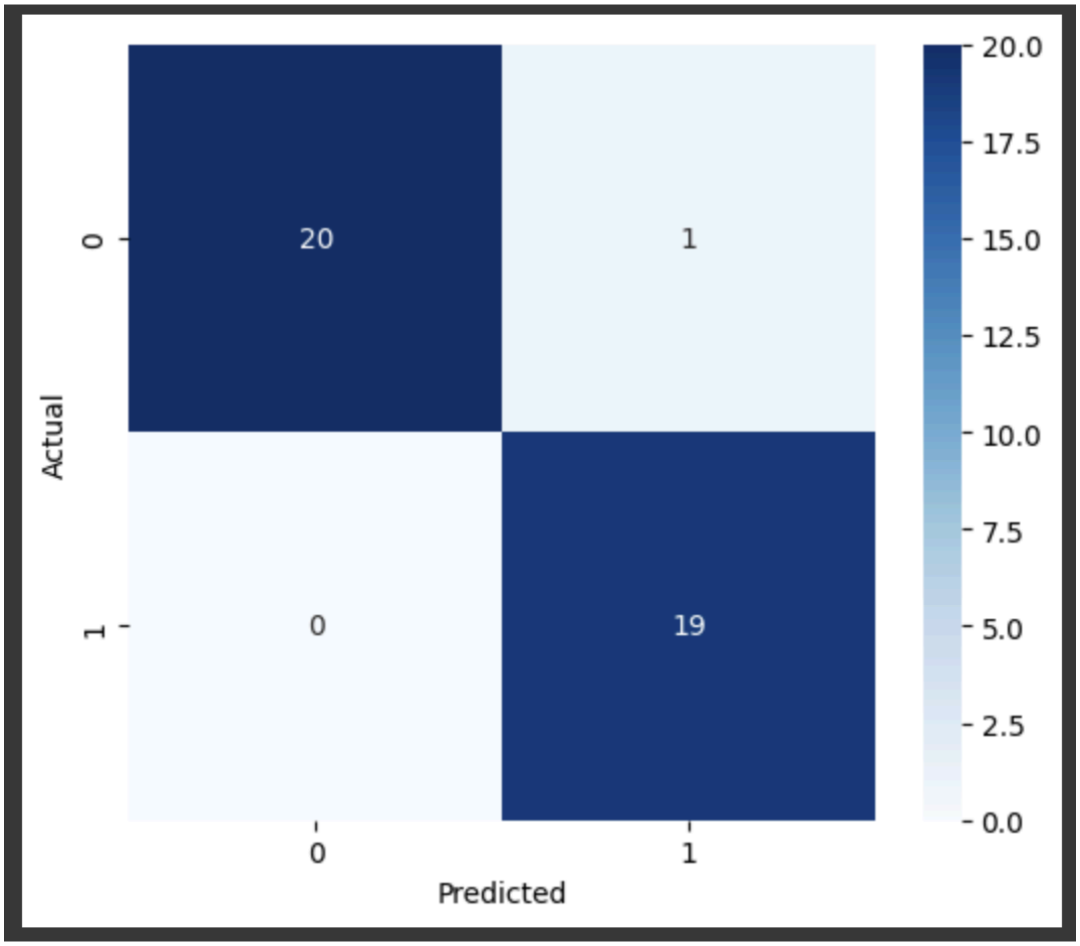
```
              precision    recall  f1-score   support

           0       1.00      0.95      0.98        21
           1       0.95      1.00      0.97        19

    accuracy                           0.97        40
   macro avg       0.97      0.98      0.97        40
weighted avg       0.98      0.97      0.98        40

Confusion Matrix:
 [[20  1]
 [ 0 19]]
```

Confusion matrix


Actual
Predicted