QR Code Authentication: Detecting Original vs. Counterfeit Prints

GitHub repo link: https://github.com/udaykirankoruvada/QR-Code-Authentication.git

Objective

The objective is to build an efficient classification system in both traditional machine learning and deep learning techniques, ensuring high accuracy and robustness in detecting counterfeit QR codes.

Dataset Description:

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.

Data Exploration and Analysis:

In the given dataset found that each class contains 100 samples with the class names:

First Print, Second Print

Classes in the dataset: ['First Print', 'Second Print']

Number of images per class: {'First Print': 100, 'Second Print': 100}

Sample images from the Dataset:

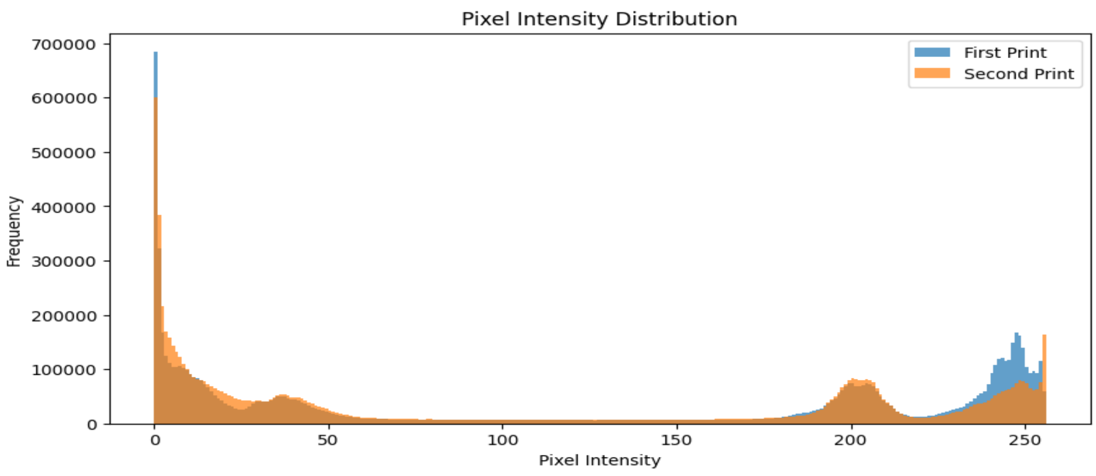
Displaying samples for class: **First Print**	Displaying samples for class: **Second Print**
First Print - input_image_active (1).png First Print - input_image_active (3).png First Print - input_image_active (4).png	Second Print - input_image_active (1).png Second Print - input_image_active (3).png Second Print - input_image_active (4).png

Data Augmentation:

Tried to perform different augmentation to increase the size of the dataset to perform the training and testing for a generalized and robust model. But unfortunately, in the process of data augmentation the data in the QR are being lost. So the model training and testing is performed on a given dataset only.
“NO DATA AUGMENTATION IS DONE – TO REMAIN THE DATA INTENT AND NOT TO LOSS DATA IN QR”

Intensity Distribution:

From the below intensity graph of both the first and second print gives the clear view on difference between the first and second prints of the QR. The bule and orange graph illustrates the differences at the both of the graph in the colour range at 0 (black) and 255(white).

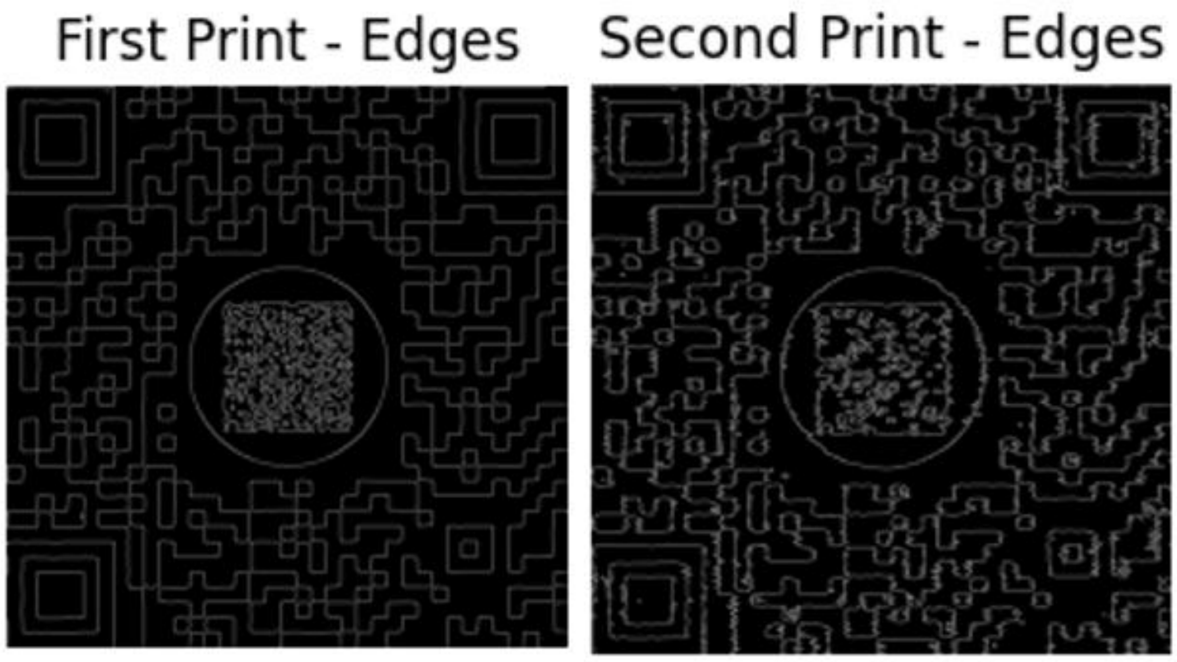


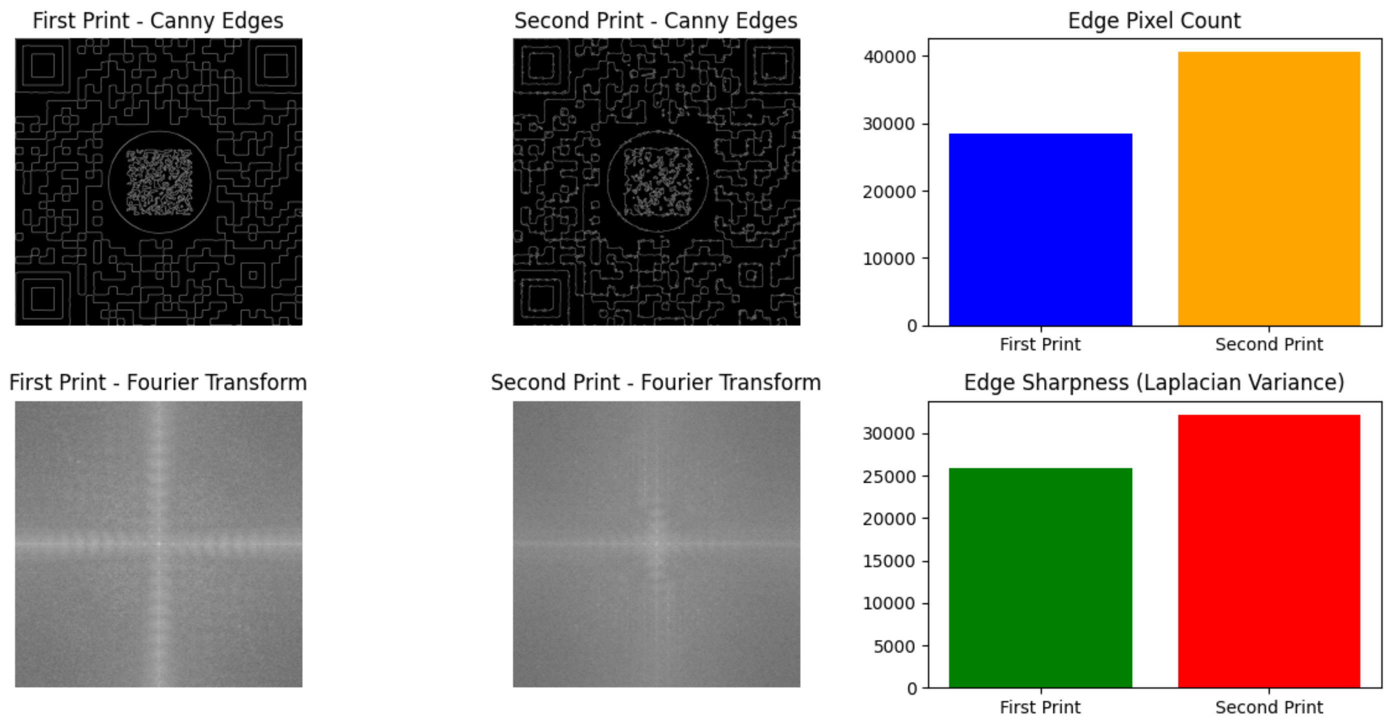
Edge Detection:

Using the canny edge detection, we can visualise the how the edges are distorted with noise and how clear the edges are in the first print. So with the help of this edge feature we can classify the first and second print.

Comparing input_image_active (1).png and input_image_active (1).png:

- Edge Pixels: 31344 vs 39443
- Sharpness: 22994.00 vs 39187.95
- Structural Similarity Index (SSIM): 0.5614





So, from the observation of the data, we can use the intensity, HOG, Edge feature of the QR of the first and second prints and to perform the classification of the prints.

Feature Engineering:

Feature extraction focused on distinguishing characteristics:

- **Traditional Features:**
    - Histogram of Oriented Gradients (HOG)
    - Edge detection using Canny filter

- **Deep Learning Features:**
    - Automatic feature extraction using Convolutional Neural Networks (CNN)

      The layers in the CNN detects the features and auto update the values of the node and layers for classification.

Model Development:

Machine Learning Approach:

As the Edges are the main characteristics feature for classification. So, SVM (support vector machine) works well for these kind of classification tasks. SVM – Support Vector Machine performs very on our dataset also.

Algorithm: SVM – support vector machine

Features used: HOG (Histogram of Oriented Gradients) + Canny Edge Detection

Performance:

Final Model Performance

Best Hyperparameters Found (via Grid Search):

C: 0.1

Kernel: Linear

Gamma: Scale

Cross-Validation Accuracy Scores: [1.00, 1.00, 0.96875, 1.00, 1.00]

Mean Cross-Validation Accuracy: 99.38% ± 1.25%

Test Accuracy: 100%

Classification Metrics

Class	**Precision**	**Recall**	**F1-Score**	**Support**
First Print	1.00	1.00	1.00	20
Second Print	1.00	1.00	1.00	20

Overall Accuracy: 100%

Macro Average: 1.00 (Precision, Recall, F1-Score)

Weighted Average: 1.00 (Precision, Recall, F1-Score)

Number of Misclassified Samples: 0

Observations

The **SVM classifier achieved perfect accuracy** on the test set, with no errors.

The **mean cross-validation accuracy (99.38%)** indicates strong generalization.

Since SVM uses extracted features instead of raw images, it proves to be a robust alternative to deep learning.

Deep Learning Approach:

The CNN model was trained for **100 epochs**, achieving high performance on both training and validation sets.

The dropout layers are added to the architecture to make the model generalized and L2 regularization is used to make the model overfitting and performs well on the unseen data.

Final Model Performance

Training Accuracy: 98.98%

Validation Accuracy: 100%

Test Accuracy: 100%

Classification Report

Class	Precision	Recall	F1-Score	Support
Class 0	1.00	1.00	1.00	21
Class 1	1.00	1.00	1.00	19

Overall Accuracy: 100%

Macro Average: 1.00 (Precision, Recall, F1-Score)

Weighted Average: 1.00 (Precision, Recall, F1-Score)

Observations

The CNN model successfully classified all test samples correctly.

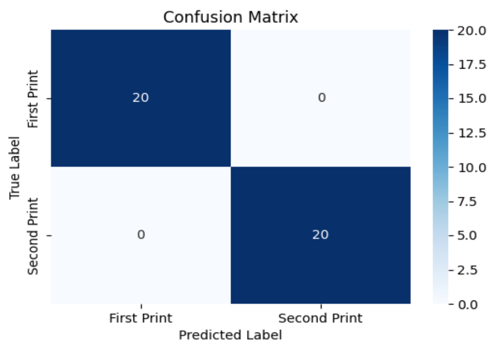
No misclassifications were observed, indicating high generalization ability.

The loss values show stable training without overfitting.

Evaluation and Results:

Both the SVM model and the CNN architecture performs and predicts very well in both the classes with an accuracy of 100 % in testing and training data which may resembles overfitting. So trained the svm model using the K-fold validation which makes the model more generealised.

Confusion matrix (svm):



Both CNN and SVM models have demonstrated **exceptional performance** in QR code authentication. The CNN model provides an **end-to-end deep learning approach**, while the SVM model leverages **feature extraction** for classification.

- **CNN** is **more flexible** as it automatically extracts features from images.
- **SVM** is **lightweight and interpretable**, making it a viable alternative for systems with computational constraints.

Deployment Considerations:

For real-world deployment, further testing with **noisy, distorted, and adversarial QR codes** is recommended to ensure **robustness and security**.

For a more generalized and robust models we need to work on with a some more large dataset then it the model cannot be overfitted.

1. Test it in different lighting conditions (because the first print might look like second in different conditions)
2. Test it in with different angle and rotation
3. Check with more quality of the second prints for correct classification

For a scalable deployment we can prefer both the web or mobile application with supports the camera module for scanning or by using the file using options.

Mail: udaykirankoruvada004@gmail.com

Koruvada Uday Kiran