QR Code Authentication: First Print vs. Second Print

Kaustubh Bhalerao

March 22, 2025

Index

1 Introduction

QR codes are widely used for product authentication and verification. In this project, we develop a machine learning pipeline to distinguish between "First Print" (original) and "Second Print" (counterfeit) QR codes. We implemented both a traditional SVM model and an enhanced Convolutional Neural Network (CNN) model to achieve high classification accuracy.
The goal is to:

- Accurately classify QR codes as either First Print or Second Print.
- Compare performance between a traditional SVM and an enhanced CNN model.
- Optimize model performance with techniques like early stopping, regularization, and dropout.

Our results show that CNN significantly outperforms SVM due to better feature extraction and training improvements.

2 Dataset and Preprocessing

2.1 Dataset

We used two ZIP files containing images of QR codes:

- **First Print**: Original QR codes
- **Second Print**: Counterfeit versions generated by reprinting original QR codes

After unzipping, the files were loaded and resized to 128×128 grayscale images.

2.2 Preprocessing

The images were normalized to pixel values between $[0, 1]$, and reshaped for CNN input:

$$\text{Normalized Image} = \frac{\text{Pixel Value}}{255.0}$$

The dataset was split into 80% training and 20% testing.

3 SVM Model

We extracted edge-based features using Canny edge detection:

$$\text{Feature} = \text{Mean}(\text{Canny Edges})$$

The features were passed to an SVM classifier with a linear kernel:

```python
from sklearn.svm import SVC

svm_model = SVC(kernel='linear')
svm_model.fit(features_train.reshape(-1, 1), y_train)
svm_predictions = svm_model.predict(features_test.reshape(-1, 1))
```

SVM Result:

```
              precision    recall  f1-score   support

           0       0.57      0.57      0.57        21
           1       0.53      0.53      0.53        19

    accuracy                           0.55        40
   macro avg       0.55      0.55      0.55        40
weighted avg       0.55      0.55      0.55        40

Confusion Matrix:
[[12  9]
 [ 9 10]]
```

The SVM model struggled to differentiate between First Print and Second Print images. The low performance suggests that handcrafted features (like Canny edges) may not be sufficient for this problem.

4 Enhanced CNN Model

We built a Convolutional Neural Network (CNN) with the following architecture:

- 3 Convolution Layers with ReLU activation and Max Pooling
- L2 Regularization ($\lambda = 0.01$) to prevent overfitting
- Dropout layers ($30\%, 40\%, 50\%, 60\%$) to improve generalization
- Sigmoid output layer for binary classification

4.1 Training Strategy

We added Early Stopping and a Learning Rate Scheduler for optimal training:

```
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=1e-5)

cnn_model.fit(X_train_cnn, y_train_cnn,
              epochs=20,
              batch_size=16,
              validation_split=0.2,
              callbacks=[early_stopping, lr_scheduler])
```

CNN Result:

```
Precision: 0.95
Recall: 0.95
F1-Score: 0.95
Confusion Matrix:
[[20  1]
 [ 1 18]]
```

5 Conclusion

The CNN outperformed the SVM model significantly due to better feature extraction, regularization, and improved training strategies.

Key takeaways:

- SVM achieved 55% accuracy with handcrafted features.
- CNN reached 95% accuracy with enhanced architecture and training.
- L2 regularization, dropout, early stopping, and learning rate scheduling improved generalization.

6 References

- TensorFlow documentation: `https://www.tensorflow.org/`
- OpenCV library: `https://docs.opencv.org/`
- Scikit-learn library: `https://scikit-learn.org/`