**Methodology**

**Dataset Description**

The dataset consists of **200 QR code images**, divided into two categories:

| **Category** | **Number of Images** |
| --- | --- |
| **Original QR Codes** | 100 |
| **Counterfeit QR Codes** | 100 |

The counterfeit QR codes are generated by scanning and reprinting the original ones. These reprinted codes often contain subtle distortions, noise, and quality loss, which can be used as features to detect them.

**Preprocessing & Feature Extraction**

Before training the models, the images go through several preprocessing steps:

1. **Grayscale Conversion**: Converts images to grayscale for better feature extraction.
2. **Resizing**: All images are resized to **128×128 pixels** to ensure consistency.
3. **Normalization**: Pixel values are scaled between **0 and 1** to enhance model training.

**Feature Extraction Techniques**

* **Edge Detection**: Captures distortions and quality loss in the QR codes.
* **Local Binary Patterns (LBP)**: Extracts texture differences between original and reprinted QR codes.
* **Histogram of Oriented Gradients (HOG)**: Helps identify microscopic changes in patterns.

**Model Development**

**Machine Learning Model – Support Vector Machine (SVM)**

The **SVM model** is trained using the extracted features from QR codes. Since SVM is effective in high-dimensional spaces, it helps in distinguishing the subtle differences between original and counterfeit QR codes.

The model is trained on a **flattened version of the grayscale images** and uses a **linear kernel** for classification.

**Deep Learning Model – Convolutional Neural Network (CNN)**

The **CNN model** is designed to learn complex features from the QR code images automatically. It consists of:

* Convolutional layers for feature extraction
* Max-pooling layers to reduce dimensionality
* Fully connected layers for classification

The CNN model is trained on the preprocessed QR codes to learn patterns that differentiate original and counterfeit versions.

**4. Experiment Results & Evaluation**

**Confusion Matrix Analysis**

The confusion matrix shows how well the models classify the QR codes.

**SVM Model:**

The SVM model achieved **100% accuracy**, correctly classifying all QR codes in the test set. However, this high accuracy might indicate **overfitting**, meaning the model may not perform as well on unseen data.

**CNN Model:**

The CNN model achieved **93% accuracy**, with minor misclassifications. It showed better generalization than the SVM model, meaning it is more likely to work well with new QR codes.

**Classification Report**

The classification report provides additional details such as **precision, recall, and F1-score**.

* **SVM Model:**
  + **Precision:** 1.00
  + **Recall:** 1.00
  + **F1-score:** 1.00
* **CNN Model:**
  + **Precision:** 1.00 for original QR codes, **0.86** for counterfeit QR codes
  + **Recall:** 0.86 for original QR codes, **1.00** for counterfeit QR codes
  + **F1-score:** Around **0.93**

These results show that while SVM gives **perfect accuracy**, it may be overfitting. The CNN model, though slightly less accurate, is better at handling **new, unseen QR codes**.

**5. GUI Implementation (Tkinter)**

To make the system user-friendly, a **Graphical User Interface (GUI)** is developed using **Tkinter**. This allows non-technical users to upload QR codes and instantly check if they are **original or counterfeit**.

**How the GUI Works**

1. The user **clicks the "Upload QR Code" button**.
2. The system **loads the selected QR image** and processes it.
3. The system **runs the QR code through both SVM and CNN models**.
4. The **result is displayed** as **"Original" or "Counterfeit"**.

The GUI makes it easy for users to verify QR codes without needing to run complex scripts.

**6. Conclusion & Future Improvements**

**Conclusion**

* The **SVM model achieved 100% accuracy**, but it may be overfitting.
* The **CNN model achieved 93% accuracy** and showed better generalization.
* The **Tkinter GUI** allows easy QR code classification in real-time.

**Final Summary**

| **Aspect** | **Details** |
| --- | --- |
| **Models Used** | SVM & CNN |
| **Dataset Size** | 200 QR Codes |
| **SVM Accuracy** | 100% (Possible Overfitting) |
| **CNN Accuracy** | 93% (Better Generalization) |
| **User Interface** | Tkinter GUI for easy testing |
| **Next Steps** | Improve dataset, optimize CNN, add real-time scanning |