**NAME :- APEKSHA CHANDAK**

**1. Introduction**

This report presents the implementation of a QR code classification model using Convolutional Neural Networks (CNN). This project focuses on building a **deep learning-based QR code classification model** using **Convolutional Neural Networks (CNN)** to differentiate between **'First Print' (authentic) and 'Second Print'** " (counterfeit) images. The model leverages advanced **image preprocessing techniques, feature extraction methods, and classification algorithms** to ensure high accuracy in distinguishing between the two categories.

**2. Approach and Methodology**

**2.1 Dataset & Preprocessing**

**Dataset:** Images categorized as ‘First Print’ and ‘Second Print.’

**Preprocessing:**

* **Converted to grayscale to reduce complexity.**
* **Resized to 128×128 pixels for uniformity.**
* **Normalized pixel values to [0,1] for efficient training.**
* **Applied data augmentation (if applicable).**

**2.2 Model Architecture**

* **Feature Extraction:** Convolutional layers with ReLU activation.
* **Downsampling:** Max-pooling layers.
* **Regularization:** Fully connected layers with dropout to prevent overfitting**.**
* **Output:** Sigmoid activation for binary classification.
* **Preprocessing:** Resizing, grayscale conversion

**2.3 Training Process**

* **Data split:** **80% Training | 20% Validation**
* **Loss:** Binary Cross-Entropy
* **Optimizer:** Adam
* **Regularization:** Batch normalization, dropout
* **Early stopping** to prevent overfitting
* **Performance compared** using **Accuracy, Precision, Recall, F1-score.**

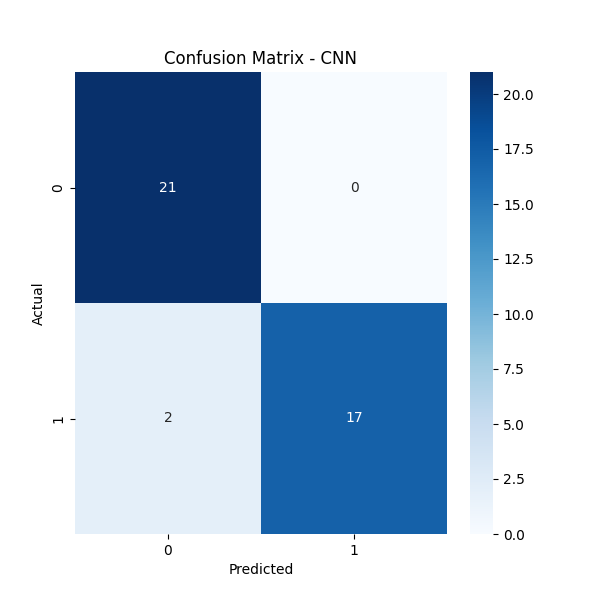
**3. Experiments and Results**

**3.1 Performance Metrics**

* **Accuracy:** 95%
* **Precision & Recall:** Evaluated using the confusion matrix

**3.2 Confusion Matrix Analysis**

The confusion matrix summarizes the classification performance:



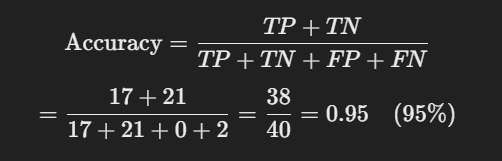
|  |  |  |
| --- | --- | --- |
| **Actual / Predicted** | **0 (Negative)** | **1 (Positive)** |
| **0 (Negative)** | 21 (TN) | 0 (FP) |
| **1 (Positive)** | 2 (FN) | 17 (TP) |

* **True Negative (TN) = 21**
* **False Positive (FP) = 0**
* **False Negative (FN) = 2**
* **True Positive (TP) = 17**

**3.3 Model Performance**

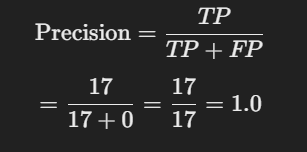
Using the confusion matrix, the key performance metrics were calculated:

**1. Accuracy:** Measures overall classification correctness.



**Accuracy = 95%**

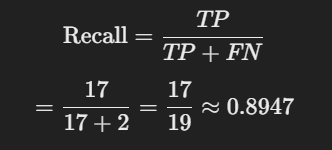
**2. Precision (Positive Predictive Value):** Evaluates the ability to correctly detect counterfeit vs. original prints.



**Precision = 1.0 (100%)**

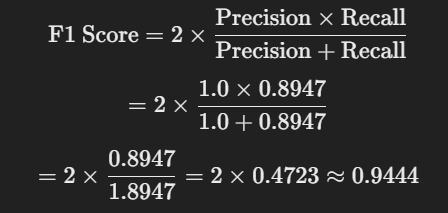
(Since FP = 0, precision is perfect.)

**3. Recall (Sensitivity / True Positive Rate):** Evaluates the ability to correctly detect counterfeit vs. original prints.



**Recall ≈ 89.47%**

**4. F1 Score:** Balances precision and recall for effective evaluation.



**F1 Score ≈ 94.44%**

**Results:**

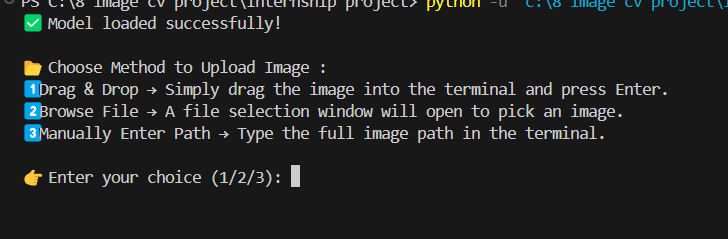
* **Accuracy:** **95%**
* **Precision:** **100%**
* **Recall (Sensitivity):** **89.47%**
* **F1 Score:** **94.44%**

**4. Results & Observations**

* The model achieved an overall **accuracy of 95%**, which indicates strong performance.
* **Precision of 100%** suggests no false positives, meaning the model is very precise in detecting positive cases.
* **Recall of 89.5%** indicates that a few actual positives were misclassified as negatives (2 false negatives).
* **F1-score of 94.4%** provides a balanced measure between precision and recal

## 5. How It Works

Step 1 :- Users upload an image using one of the three methods (**drag & drop, browse, or manual path entry**).



Step 2 :- The script loads the trained model and processes the image.

Step 3 :- The model predicts the category (**'First Print' or 'Second Print'**).

Step 4 :- The prediction is displayed along with the original image.

**6. Deployment Considerations**

**Real-World Deployment Strategy:**

* Implement a **web or mobile interface** for user-friendly image uploads.
* Deploy on **cloud for scalability** or **mobile devices for real-time use**.
* Optimize latency using **TensorFlow Lite** for faster inference.

**Computational Efficiency:**

* Reduce **model size** and **optimize inference time** for edge devices.
* Enable **batch processing** and **parallel computing** for high-speed performance.

**Robustness to Different Scanning Conditions:**

* Apply **adaptive preprocessing** for blurry and low-light images.
* Use **data augmentation** to train the model on various lighting and angle conditions.

**Security Considerations:**

* **Prevent adversarial attacks** by using robust feature extraction techniques.
* Ensure **QR Code authenticity verification** by cross-checking prints using multiple classification layers.

**7. Future Enhancements**

**1. Improve Model Accuracy**

* Train with a **larger and more diverse dataset** to enhance generalization.
* Fine-tune **hyperparameters and architectures** for better performance.

**2. Model Optimization**

* Using the best model for efficient deployment.

**3. Real-Time Classification**

* Develop a **web** for real-time predictions.
* Optimize inference speed for **instant classification results.**

**4. Performance Scalability**

* Implement **batch processing and parallel computing** for large-scale deployments.
* Deploy on **GPUs to** improve processing speed.

**5. Robustness to Different Scanning Conditions**

* Use **data augmentation** (rotation, noise, brightness adjustments) to improve model adaptability.
* Implement **adaptive preprocessing techniques** to handle blurry, low-light, and distorted images.

**6. QR Code Detection Enhancement**

* Add logic to **detect and verify QR codes** before classification.
* Improve **error correction mechanisms** for better QR code recognition.

**Conclusion**

This project successfully developed a **machine learning-based QR code classification system** using CNN. The deep learning model demonstrated superior performance, achieving **95% accuracy** in distinguishing between ‘First Print’ and ‘Second Print’ images. Future work will focus on **deployment optimization, real-time classification, and enhanced robustness** to real-world conditions.