South African Bank Notes Recognition

Mvelo Mlangeni , Simakahle Goge

1 Introduction

Bank note classification plays a significant role in financial systems, such as automated teller machines (ATMs), currency sorting machines, and counterfeit detection systems. Accurate identification of bank notes is essential to prevent fraud and ensure seamless transactions. This project aims to create a reliable bank note classification system utilizing state-of-the-art image processing and machine learning techniques.

Github: https://github.com/gogesimma/Banknote-project?tab=readme-ov-file#banknote-project

2 Data Structuring and Organization

In structuring the data for our South African banknotes recognition project, we aimed for organization and efficiency to optimize model training and evaluation processes. Here's how we structured the data:

1. **Dataset Organization**: The dataset was organized into a hierarchical structure, with each banknote denomination as the top-level category. For example, categories included the old R10, new R10, old R20, new R20, and so forth.

2. **Subcategories for Sides and Orientations**: Within each denomination category, we further categorized the images based on the side (front or back) and orientation (e.g., upright or flipped) of the banknotes. This subdivision allowed us to account for variations in how the banknotes may appear in real-world scenarios.

3. **Data Splitting**: We split the dataset into training, validation, and test sets to facilitate model development and evaluation. This splitting ensured that the models were trained on a diverse range of images while also having separate sets for validation to tune hyperparameters and test for generalization performance.

4. **Balancing Classes**: We ensured that each class (combination of denomination, side, and orientation) had a balanced representation in the dataset. This balancing act helped prevent biases during model training and ensured that the models learned to classify all classes effectively.

5. **File Naming Convention**: Each image file was named systematically to reflect its category and attributes. This naming convention facilitated easy access to specific images during data preprocessing, augmentation, and model training.

By structuring the data in this manner, we created a well-organized and balanced dataset that provided the necessary input for training our South African banknotes recognition system. This structured approach not only improved the efficiency of model development but also contributed to the overall robustness and accuracy of the system.

3 Image Preprocessing and Enhancement

Image preprocessing and enhancement are crucial steps in improving the quality and consistency of input data. These steps help enhance the performance of subsequent stages, such as segmentation, feature extraction, and classification.

3.1 Data Augmentation

To increase the dataset's diversity and improve model robustness, data augmentation techniques were applied. Transformations included rotations, width and height shifts, shear transformations, zooming, and horizontal flipping. These augmentations help the model generalize better by introducing variations in the input images.

3.2 Resizing and Normalization

All images were resized to a standard size of 256x256 pixels to maintain uniformity. Resizing ensures consistent input dimensions across the dataset, which is essential for neural network processing. Additionally, pixel values were normalized to the range [0, 1] to facilitate better convergence during model training. Normalization scales the input values, making the training process more stable and efficient.

4 Image Segmentation

Image segmentation involves partitioning an image into multiple segments to simplify its representation and make it more meaningful for analysis. Various segmentation techniques were explored in this project, including Sobel edge detection, Canny edge detection, and other traditional methods. However, these techniques did not perform well for our specific task of bank note classification.

4.1 Sobel Edge Detection

Sobel edge detection is a gradient-based method used to detect edges in an image. It computes the gradient of the image intensity at each pixel, emphasizing regions with high spatial frequency that correspond to edges. Despite its effectiveness in detecting edges, Sobel edge detection did not perform well for our bank note dataset. The detected edges were often noisy and incomplete, leading to inaccurate segmentation and poor model performance.

4.2 Canny Edge Detection

Canny edge detection is a multi-stage algorithm designed to detect a wide range of edges in images. It involves noise reduction, gradient calculation, non-maximum suppression, double thresholding, and edge tracking by hysteresis. Although more advanced than Sobel edge detection, Canny edge detection also yielded unsatisfactory results. The algorithm struggled with the complex textures and patterns of the bank note images, resulting in fragmented and inaccurate segmentation.

4.3 Other Segmentation Techniques

Other traditional segmentation techniques, such as thresholding, region-based segmentation, and morphological operations, were also explored. These methods faced similar challenges as Sobel and Canny edge detection, struggling with the variability in bank note designs and lighting conditions. Due to the limitations of these techniques, we focused on deep learning models for feature extraction.

5 Feature Extraction

Feature extraction is crucial in identifying distinguishing characteristics of bank notes. In this project, the VGG16 model was employed as a feature extractor. VGG16 is a deep convolutional neural network (CNN) architecture known for its excellent performance in image recognition tasks. By leveraging the pre-trained VGG16 model, we were able to extract high-level features from bank note images.

The VGG16 model was initialized with ImageNet weights and fine-tuned on our bank note dataset. The model was configured to exclude the fully connected layers (*include top=False*) to retain only the convolutional layers. This allowed us to utilize the learned features from the VGG16 model while adapting the final layers for our specific classification task.

The output of the last convolutional layer of the VGG16 model served as the feature representation of each bank note image. These features were then flattened into a onedimensional vector and used as input for the subsequent classification model. By leveraging the rich hierarchical representations learned by VGG16 on ImageNet, we were able to capture informative features that facilitated accurate classification of bank notes.

6 Bank Note Classification

Classification of bank notes was performed using several machine learning models trained on the extracted features. The models used included Multi-Layer Perceptrons (MLP), k-Nearest Neighbors (KNN), and Decision Trees. Each model was evaluated for its performance in accurately classifying bank notes based on the extracted features.

6.1 Precision, Recall, and Confusion Matrix Analysis

- **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives. High precision relates to the low false positive rate. The MLP classifier had the highest precision among the three models, indicating its reliability in predicting the correct class.

- **Recall:** Recall is the ratio of correctly predicted positive observations to all observations in the actual class. High recall relates to the low false negative rate. The MLP classifier also had the highest recall, showing its effectiveness in capturing the true positive rate.

- **Confusion Matrix:** The confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. The confusion matrix for each model provided insights into the classification performance, with the MLP model showing the least confusion among classes.

6.2 Multi-Layer Perceptron (MLP) Classifier

The MLP classifier achieved an accuracy of 92.31%, with a precision of 92.58% and a recall of 92.31%. The model configuration included one hidden layer with 200 neurons, ReLU activation function, and Adam optimizer. The model was trained for 2500 iterations to ensure convergence.

6.3 k-Nearest Neighbors (KNN) Classifier

The KNN classifier achieved an accuracy of 65.28%, with a precision of 70.30% and a recall of 65.28%. Despite its simplicity, the KNN model achieved commendable accuracy.

6.4 Decision Tree Classifier

The Decision Tree classifier achieved an accuracy of 32.64%, with a precision of 33.28% and a recall of 32.64%. Although interpretable, the Decision Tree model was the least effective among the three classifiers.

Accuracy of the MLP classifier: 92.31%

Accuracy of the KNN classifier: 65.28%

Accuracy of the Decision Tree classifier: 32.64%

6.5 Multi-Layer Perceptron (MLP) Classifier

The MLP classifier achieved an accuracy of 92.31%, with a precision of 92.58% and a recall of 92.31%. The confusion matrix for the MLP classifier is as follows:

$$18 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

0 0 0 0 0 0 0 0 0 0

0 13 1 1 0 3 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 2 36 0 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 1 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 0 0 0 12 2 0 0 0 0 0 0 2 0 0 0 0 0 0 0

1 1 1 0 2 17 0 0 0 0

0 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 37 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 22

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

18 0 0 0 1 0 0 0 3 0

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 21 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 25 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 0
\end{bmatrix}$$

$$
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 20
\end{array}
$$

7 Conclusion

In this project, we developed a robust bank note classification system using advanced image processing techniques and machine learning models. The project involved preprocessing and enhancing images, segmenting individual bank notes, extracting relevant features using the VGG16 model, and classifying them using various classifiers.

Traditional segmentation techniques such as Sobel and Canny edge detection did not perform well for our specific task, leading us to rely on deep learning for feature extraction. The VGG16 model was instrumental in extracting high-level features from the bank note images, which were then used for classification.

Among the classifiers tested, the MLP classifier achieved the highest accuracy, followed by the KNN and Decision Tree classifiers. The high accuracy of the MLP classifier highlights the effectiveness of using deep learning for feature extraction and the power of neural networks for classification tasks.

This project demonstrates the potential of deep learning and machine learning techniques in accurately classifying bank notes, paving the way for more advanced and reliable currency recognition systems in the future.