South African Bank Note Recognition System

Shaneer Luchman, Daniel Mundell, Siphokazi H. Nene, and Akaylan Perumal

University of KwaZulu-Natal, South Africa
{216003502,220108104,216009670,216035695}@stu.ukzn.ac.za

Abstract. In this paper, we propose a bank note recognition system by combining various feature extraction and classification techniques, including Grey-Level Co-Occurrence Matrix and Haralick texture features, K-means data clustering, a Decision Tree Classifier, two Neural Networks and a K-Nearest Neighbours classifier. The proposed system works on both the front and back sides of all new and old South African bank notes, where the bank notes can be in the correct orientation or rotated 180 degrees. An overall accuracy rate of 96.6% was achieved, along with a false reject rate of 4% and a false accept rate of 0%. The system takes an average of 0.4 seconds to classify one bank note.

Keywords: Bank Note Recognition · Haralick Features · Decision Tree Classifier · Neural Network · K-Nearest Neighbours.

1 Introduction

Machine learning paired with Computer Vision provided a massive breakthrough for computer researchers and engineers to solve various application domains, and critical real-life problems without needing the additional use of expensive hardware. Computer Vision can be regarded as the training of computer systems to understand and interpret digital images, similar to how human vision works. Systems must process and analyse digital images to extract and store features(properties), which quantifies the information stored in them. The information stored in these systems can be used to classify and predict the labels, and information provided by other input images [12]. Using the concepts of Machine learning and Computer Vision, we aimed to create a system that could successfully classify an input bank note image as genuine or counterfeit, as well as the value of the note.

Bank note recognition is used in a variety of automated systems, such as ATM machines [1], vending machines, and money counters [2], where bank notes may be deposited in exchange for money in a digital account or to purchase a product. Such a recognition system needs to detect non-genuine or counterfeit bank notes to prevent criminals from exploiting the system, while still classifying the denomination of genuine bank notes efficiently with a high enough accuracy rate so as to ensure the satisfaction of regular customers. The speed of classification required depends on the throughput of the machine in which the recognition system is being used. Another common use of bank note recognition is in assisting people who are visually impaired [3, 4, 5].

Our proposed bank note recognition system has been designed to work with bank note images which have already been segmented from their surroundings, or in which the images have been scanned from very close up, such as in an ATM machine or money counter. Therefore our goal was not to detect bank notes from broader images where the bank notes are potentially obscured, but rather to accurately classify bank notes into their respective denomination classes, and to detect when a non-genuine bank note has been entered.

The system was trained to work on both the front and back faces of new and old South African bank notes of all given denominations (R10, R20, R50, R100, R200). The system was designed to be invariant to scale and rotation, i.e. bank notes can be in the correct orientation or rotated 180 degrees, but not at obscure angles in between.

The development, training and testing stages were performed in PyCharm Community Edition 2019 with Python 3.7.6, and our source code is available on Google Drive at [1]. Various external libraries were used, including NumPy [6], Open Source Computer Vision (OpenCV) [7], scikit-image [8] and scikit-learn [9]. While there are many existing feature detection and matching algorithms available which have been shown to produce excellent performance, such as Scale-Invariant Feature Transform (SIFT) [10], Speeded Up Robust Features (SURF) [3], and Oriented fast and Rotated Brief (ORB) [5], these algorithms tend to complete multiple steps in a single function, without requiring knowledge of how the inner workings of the function. Therefore, we have decided to first implement all the stages required in a recognition system separately to gain a better understanding of these individual steps. In the future, if we required a perfectly performing recognition system, or needed to improve the classification speed, we could use one of the existing more advanced and well-tested algorithms defined above, but take with us the knowledge we gained from building this system.

The rest of the paper will discuss each of the bank note recognition processes as follows: Section 2 details the preprocessing and enhancement first applied to bank note images when they are entered, including scaling, grey scale, histogram equalization, thresholding, and Gaussian smoothing. Section 3 discusses the segmentation process used, and provides figures with various examples. Section 4 explores the three feature extraction processes used on the segmented regions of the binary image, grey-scale Gaussian smoothed image and original colour image, including a white pixel count per region, a GLCM and Haralick feature extraction, and K-means data clustering extraction. Section 5 discusses the three classification techniques used, i.e. Decision Tree Classifier (DCT), neural networks, and K-Nearest Neighbours (K-NN), where each technique applies to one of the three feature extraction processes. Section 6 shares and discusses the results and performance of the proposed system, and section 7 concludes the report.

[1] https://drive.google.com/file/d/14jMwmb1K-6GrcvVdGXohGAio9Xra2gg8/

2 Preprocessing and Enhancement

This section will detail the preprocessing and enhancement methods used to allow the segmentation to work on all bank notes, regardless of scale, to remove unwanted noise, and to enhance the performance of the feature extraction and classification processes.

2.1 Fixed Scaling

First, the original bank note image is scaled to a *width* x *height* of 400x200 using the OpenCV *resize*() function. While the different bank note denominations have different aspect ratios, this scaling process allows the segmentation process to identify the correct regions regardless of the input scale or bank note denomination, thus making the bank note recognition system invariant to scale. From our testing, the size of 400 x 200 was able to adequately capture all the detail required for an accurate bank note recognition system. The scaled image is shown in Figure 1.

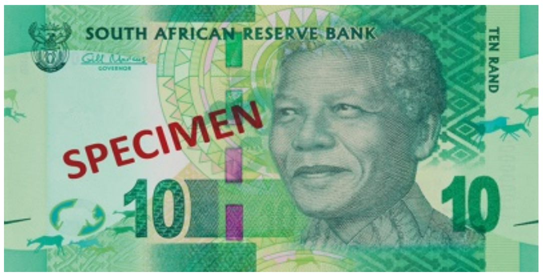


Fig. 1: The original image scaled to a fixed *width* x *height* of 400x200.

2.2 Grey Scale

The scaled image is then converted to grey scale using the OpenCV *cvtColor*() function and the COLOR_BGR2GRAY parameter. Both the thresholding technique, and the Grey-Level Co-Occurrence Matrix (GLCM) and Haralick feature extraction, which are implemented later in the pre-processing and feature extraction sections respectively, require this grey-scale image. The grey-scale image is shown in Figure 2.

2.3 Histogram Equalization

Histogram equalization is then applied to the grey-scale image using the OpenCV *equalizeHist*() function. This technique improves the global contrast in the images, allowing a more equal comparison between bank note images taken with poor and good lighting conditions. The equalized image is shown in Figure 3

Fig. 2: The scaled image has been converted to grey scale.

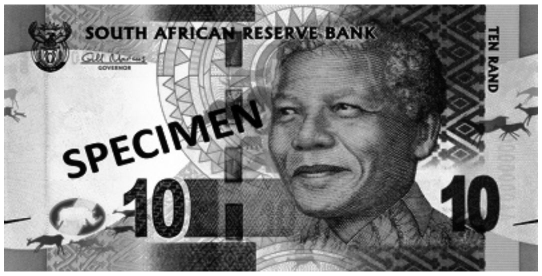


Fig. 3: Histogram equalization is applied to the grey-scale image to improve contrast.

2.4 Binary Thresholding

A binary image, shown in Figure 4, is then created from the equalized image using a thresholding technique. After some manual testing, it was determined that the best results for our use case were produced using the OpenCV *threshold*() function with a thresh value or 180 and a max value of 255. This binary image is used later in a DTC for the classification of the bank note category (front/back and new/old) and the rotation of the image.

2.5 Gaussian Smoothing

To remove any sharp imperfections in the image, and thereby greatly improving the accuracy of the overall system, we apply a Gaussian smoothing effect to the equalized image using the OpenCV *GaussianBlur*() function with a 7x7 filter size. This blurred image, shown in Figure 5, is used as the basis of calculating the GLCM and Haralick features in the feature extraction section.

3 Segmentation

The goal with segmentation is to split an image into various regions, allowing the system to gain more meaningful information when feature extraction takes

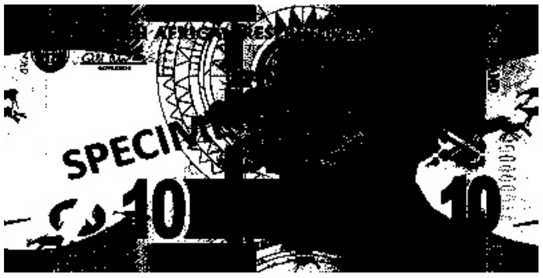


Fig. 4: A binary image created from the equalized image using a thresholding technique.

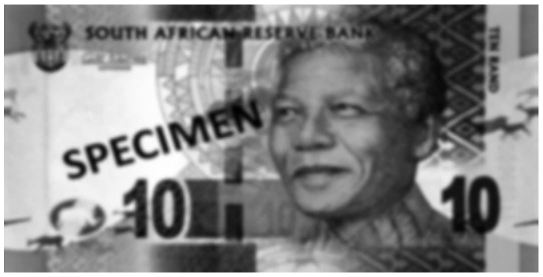


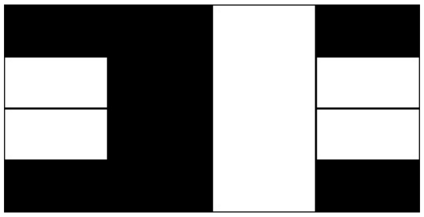
Fig. 5: A blurred image is created by applying Gaussian smoothing to the equalized image.

place with each individual region instead of the image as a whole. We visually inspected the scaled bank notes and identified regions which had unique textures or patterns across the different bank note categories (front/back and new/old) and denominations (10/20/50/100/200), as well as which allowed the system to detect whether the note had the correct rotation. The identified region boundaries were hard-coded into the program to ensure the same regions were segmented for feature extraction in both the training and testing phases. Because the bank notes are first scaled, when these regions are segmented from the blurred image, the same information is captured regardless of the initial size and scale of the bank note.

3.1 Binary Image

There were 5 regions identified from the binary images, shown in Figure 6, which allow the binary image region feature extraction system and the corresponding DTC to detect the bank note category (front/back and new/old) and whether the note had the correct rotation or required a 180 degree corrective rotation.

Once the bank note category is known, and the rotation has been corrected if necessary, the system can then segment regions unique to the bank note category, in order to detect the bank note denomination, without the worry of segmenting incorrect regions and potentially missing crucial information.


(a) Region Mask

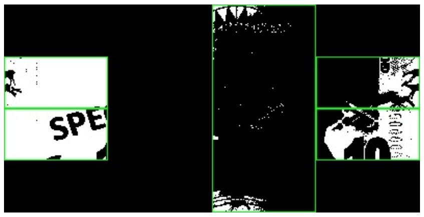
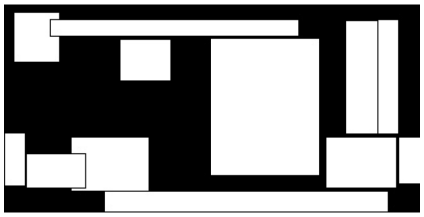
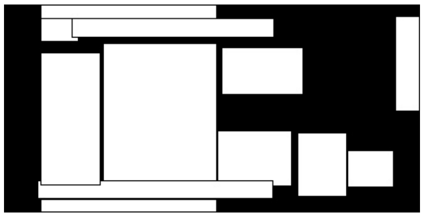

(b) Example Usage

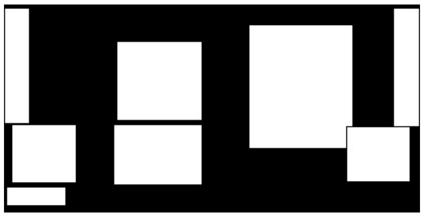
Fig. 6: Region mask used for segmentation of the binary image.

3.2 Blurred Image

There were 12 and 8 regions identified for the front and back of new bank notes, and the front and back of old bank notes, respectively, which allowed the GLCM and Haralick feature extraction system and the neural network classification system to detect the bank note denomination. The region masks identified, and examples of the masks being applied, are shown in Figure 7 and Figure 8, respectively.


(a) Front New


(b) Back New


(c) Front Old

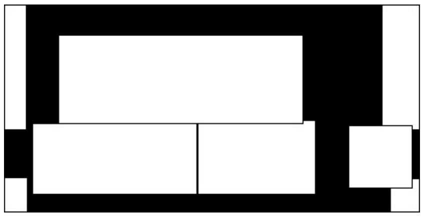
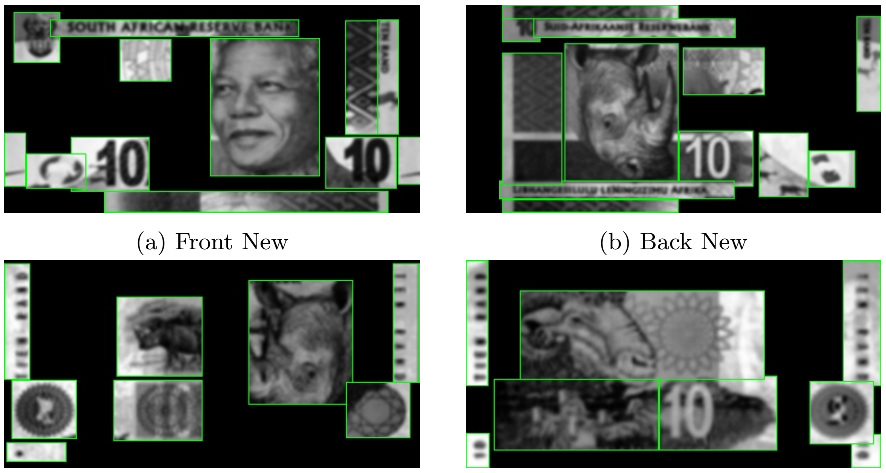

(d) Back Old

Fig. 7: Region masks used for segmentation of blurred bank note.

4 Feature Extraction

This section details the three feature extraction techniques used in various parts the bank note recognition system. The features extracted from these three techniques correspond directly to the three classification techniques discussed in the



(a) Front New (b) Back New

(c) Front Old (d) Back Old

Fig. 8: Example segmentation of blurred R10 bank note.

next section. While the ratio (width/height) is unique with each bank note denomination, and so would proof to be a reliable feature for classification, we decided not to extract this feature in an effort to make our recognition system scale invariant.

4.1 Binary Image Region Features

The number of white pixels in each of the binary image regions is counted. These features are used in the DTC to detect the bank note category and rotation. As shown in Figure 6, these are the four equally-sized regions on either side of the larger region 3. The features from the four smaller regions will be compared to each other in the DTC, while the white pixel count of Region 3 will be compared to a static value which has been determined through manual testing.

4.2 Grey-Level Co-Occurrence Matrix and Haralick Features

After initially writing our own loop-based implementations to calculate the GLCM and corresponding Haralick features, it was determined that this was too inefficient and would take far to long to calculate for one region, let alone all the regions for all the bank notes in the training set. We decided to use a python image processing library called $scikit-image$ [8], which was able to much more efficiently calculate the GLCM and Haralick features, using the $greycomatrix()$ and $greycoprops()$ functions, respectively.

The GLCM was calculated using a distance of 1, an angle of 0 degrees, and 256 intensity levels. While we could have used more complex settings, such as larger distances or multiple angles, the settings used provided adequate performance in our testing.

Five Haralick features were selected to be calculated for each segmented region: Contrast, dissimilarity, homogeneity, energy, and correlation. These features were calculated according to the Equations 1–5. While there are many more Haralick features available, the five selected provided adequate performance when used in neural network classifiers. Since we have bank notes with 12 and 8 regions, this will produce 60 and 40 features respectively. These features are normalized to between 0 and 1 using a python machine learning library called $scikit-learn$ [9], specifically the $preprocessing.normalize()$ function. The normalization maps the features into a common scale which improves the neural network performance by preventing data points with much larger ranges from contributing a disproportionately large amounts to the outcome [11].

$$\text{Contrast} = \sum_{i,j=0}^{L-1} P_{i,j}(i-j)^2 \tag{1}$$

$$\text{Dissimilarity} = \sum_{i,j=0}^{L-1} P_{i,j}|i-j| \tag{2}$$

$$\text{Homogeneity} = \sum_{i,j=0}^{L-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{3}$$

$$\text{Energy} = \sqrt{\sum_{i,j=0}^{L-1} P_{i,j}^2} \tag{4}$$

$$\text{Correlation} = \sum_{i,j=0}^{L-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right] \tag{5}$$

Where L is the number of grey levels, in our case 256, and μ and σ are the means and standard deviations, respectively.

4.3 Features From Colours

Five features are extracted from the colours of the original bank note image, and are used in two different classifiers.

K-means data clustering was used to extract the two most dominant colours in the image. This is done using the OpenCV $kmeans()$ function, with 2 center points, 10 max iterations, an epsilon value of 1.0, 10 attempts, and random initial centers. The function is terminated when either the max iterations or the required level accuracy is reached.

Three mean values are calculated, one for each of the RGB channels. The values are normalized to between 0 and 1 by diving them by the max intensity level of 255. These three features are added to the already normalized Haralick features to be used in the neural network classifiers. We use the three mean values separately instead of combining them to represent a single average colour, as that would likely result in a dull brown for all the bank notes.

5 Classification

The classification process is a combination of three different classification techniques. We will first outline the overall process before going into more detail on each technique.

First, the DTC classifies the entered bank note into categories (front/back and new/old) and fixes its orientation. Then the correct regions are extracted and used in the corresponding neural network to predict a bank note denomination. At the same time, the K-NN predicts a bank note denomination based on the dominant colour features, and these two predicted denominations are compared. If the predictions do not match, or the neural network confidence is below 70%, the bank note is classified as non-genuine, otherwise it is classified as the matching predicted denomination.

5.1 Decision Tree Classifier for Binary Image Region Features

Through manual inspection of the region features extracted from the binary image, the DTC shown in Figure 9 was designed. It is able to classify bank notes into three classes, one from each of the three bank note categories shown below.

Note: After the first test node, if the $True$ branch is followed, the binary image being used for extraction is rotated 180 degrees. Since this will affect the information which is extracted from each region, the feature extraction process shown in section 4.1 should be run at the same time as the DTC, rather than first extracting all the features and then running the classifier.

Five features extracted from the binary image:

$$x_i = \text{White pixel count of Region } i\text{, where } i = 1, 2, 3, 4, 5$$

Three bank note categories with two classes each:

$$\begin{aligned} \text{F} &= \text{Front}, \ \ \text{B} = \text{Back} \\ \text{N} &= \text{New}, \ \ \text{O} = \text{Old} \\ \text{R} &= \text{Rotation needed}, \ \ - = \text{No rotation needed} \end{aligned}$$

For an example, the above DTC is applied to the front side of a correctly orientated new R10 bank note from Figure 8. The feature values for the five regions re-shown in Figure 10 are first calculated and shown in Table 1, and then we follow the decisions in the DTC arriving at the correct classification.

Following the DTC, first we compare the mean of Region 1 and Region 2 ($4641 < 1108 =$ False), and traverse the right hand $False$ branch. Then we

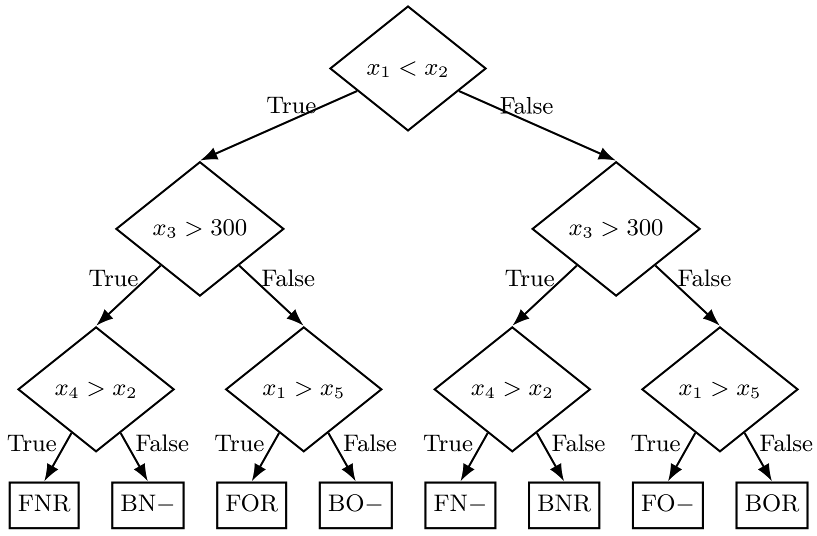


Fig. 9: Decision tree classifier for binary image region features

Table 1: White pixel features of the five regions from Figure 10.

	Region 1	Region 2	Region 3	Region 4	Region 5
White pixel count	4641	1108	840	2557	4058

compare the white pixel count of Region 3 with 300 ($840 > 300 =$ True), and traverse the left hand $True$ branch. Then we compare the mean of Region 4 and Region 2 ($2557 > 1108 =$ True), and traverse the left hand $True$ branch, ending in the correct classification of FN–, i.e. The front side of a correctly orientated new bank note.

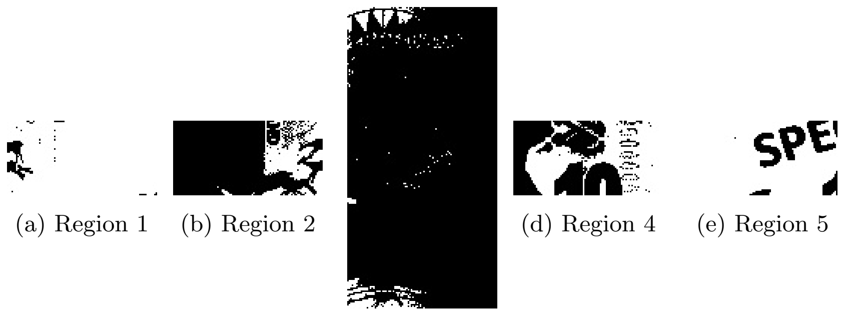
5.2 Neural Network for Haralick Features and Mean Colours

Neural Network Structure An adaption to jamesloyys' neural network implementation [13] was used to create two neural networks used to classify bank note denominations based on a combination of the Haralack features and mean colours extracted.

The first neural network was created for the front and back of new South African bank notes. Since these bank notes have 12 region masks, and 5 Haralick features were extracted from each region, this gives us a total of 60 Haralick features and 3 mean colour features as inputs.

Similarly, the second neural network was created for the front and back of old bank notes, giving us a total of 43 features (40 Haralick features and 3 mean colour features) as inputs.

The output layers of both neural networks have five nodes, one for each of the five denominations. Both neural networks have one hidden layer where the



(a) Region 1 (b) Region 2 (c) Region 3 (d) Region 4 (e) Region 5

Fig. 10: Five regions segmented from the binary image of the front side of a correctly orientated new R10 bank note.

number of nodes is determined by the sum of the number of input and output nodes divided by 2, as shown in Equation 6. This gives 34 and 24 hidden nodes for the new and old bank note neural networks, respectively. A summary of the number of nodes per layer is shown in Table 2.

$$\text{no. of hidden nodes} = (\text{no. of input nodes} + \text{no. of output nodes})/2 \qquad (6)$$

Table 2: Number of nodes per layer in the neural networks.

Neural network	Input layer	Hidden layer	Output layer
New bank notes	63	34	5
Old bank notes	43	24	5

Training After some manual testing and adjustments, it was decided to train both neural networks for 10,000 iterations. Training for more or less iterations than this resulted in overfitting and underfitting, respectively. The trained neural network weights were saved to Comma Separated Values (CSV) files and imported at the start of the prediction phase.

5.3 K-Nearest Neighbours for Dominant Colours

The K-NN classification technique was used to determine the most likely bank note denomination based on the two dominant colours extracted. There are five classes of bank note denominations (R10, R20, R50, R100, R200) which all have unique dominant colours regardless of whether they are the front or back of new or old notes. The Euclidean distances between the two dominant colours extracted from the entered bank note and the 40 dominant colours extracted from the dataset of 20 bank notes during training, are calculated, and the denomination class selected based on the smallest distance. Since the dominant colours of each bank note denomination are easily distinguishable, this K-NN with $k = 1$ performed adequately. We would have investigated using a K-NN with a larger number of nearest neighbours being considered if there was a more complex classification needed.

5.4 Dataset

Training Dataset The training dataset was made up of the best quality image we had of each bank note. Since there are five denominations, each with two sides, and all of these have a new and old variant, there were 20 images in the training dataset. For the neural network classifier, the training dataset was split between the new and old neural networks, providing 10 images for each. We did not include any images which had been rotated, as the DTC would first correct the rotation of any bank note entered. We also did not include any non-genuine South African bank note images, as we did not have a reliable source for these.

Testing Dataset The testing dataset was made up of 118 bank note images: 100 genuine South African bank note images and 18 non-genuine bank notes. This included the front and back of 4 new bank notes and 1 old bank note for each of the 5 denominations, giving $2 \times (4 + 1) \times 5 = 50$ images. The 180 degree rotation of all these 50 images was also included to ensure the system was rotation invariant, giving $50 + 50 = 100$ genuine South African bank note images. The 18 non-genuine bank notes were a combination of American Dollars, Indian Rupees, Zimbabwean Dollars, and Monopoly Dollars, in both the correct and rotated orientations. While some of these are genuine bank notes in their respective countries, we refer to them as non-genuine because our recognition system is only interested in South African bank notes. Figure 11 shows a selection of bank notes from the testing dataset, with a combination of both orientations of front and back, new and old genuine bank notes, as well as both orientations of non-genuine bank notes.

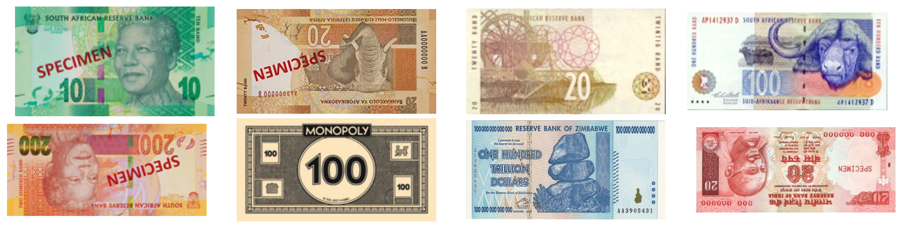


Fig. 11: A selection of eight bank notes from the testing dataset.

6 Results and Discussion

The proposed recognition system correctly classified 114 of the 118 bank notes in the testing dataset, giving an overall accuracy rate of 96.6%. Out of the 100 genuine bank notes tested, 96 were correctly classified, while all 18 of the non-genuine bank notes were correctly classified as non-genuine. This gives a False Rejection Rate (FRR) of 4% and a False Acceptance Rate (FAR) of 0%.

$$\begin{aligned}\text{FRR} &= \text{no. of false rejections/total no. of attempts}\\ &= 4/100 = 4\%\end{aligned}$$

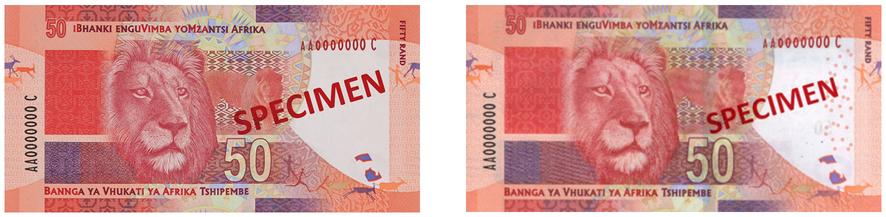
$$\begin{aligned}\text{FAR} &= \text{no. of false accepts/total no. of attempts}\\ &= 0/14 = 0\%\end{aligned}$$

Since the recognition of bank notes could be used in a high security environment such as a bank, our system considers a low FAR more important than a low FRR. If some genuine bank notes are erroneously rejected, they can be rechecked, or manually inspected, and therefore as long as the FRR is kept relatively low there will not be a huge problem. On the other hand, a high FAR would cause non-genuine bank notes to be accepted, and could allow criminals to exploit a system using this recognition system for money deposits using non-genuine bank notes. Our FAR of 0% would therefore completely remove the possibility of non-genuine bank notes being accepted. While our FRR of 4% may seem good at first glance and may perform adequately for small systems, this FRR would cause too much required human re-checking when scaled up to larger amounts of bank notes being tested. Additionally, if an ATM or other automated system intended to use the recognition system, there would be no humans available to inspect the falsely rejected bank notes.

Both the DTC and the k-NN dominant colour classifiers correctly classified every genuine bank note in the testing dataset. Looking at the 4 bank notes which were incorrectly rejected, we see that the DTC correctly classified them into the front/back, old/new and rotation, and the k-NN correctly identified the denomination based on the dominant colours. Therefore the neural network is the weak performer in the system causing the 4 incorrect rejections. This may be because of the large number of features, which could have caused an overfitting problem.

Figure 12 visually compares one of the incorrectly rejected bank notes with the training dataset bank note of the same denomination and type. There are minor differences, most noticeably in the lions mane and the dots in the white section on the right. This may show overfitting, because the notes are so similar but the slight differences caused a rejection. However, if a counterfeit South African bank note was created, we would expect it to be very similar to the genuine bank note, but it would still need to be rejected for the system to perform adequately. Since we did not have a source for any high quality images of fake South African bank notes, we could not train or test the system using them. With a larger training dataset, the system could have trained with multiple genuine versions of each bank note as well as counterfeit bank notes.

The total elapsed time to classify all 118 bank notes was 47.4 seconds, giving an average elapsed time per bank note of 0.4 seconds. The elapsed time per bank note would be suitable for applications such as an ATM or vending machine


50
IBHANKI ENGUVIMBA YOMZANTSI AFRIKA
AA0000000 C
FIFTY RAND
SPECIMEN
50
BANNGA YA VHUKATI YA AFRIKA TSHIPEMBE
AA0000000 C

(a) Training bank note (b) Incorrectly rejected bank note

Fig. 12: A comparison of the training and testing R50 bank notes.

which don't require large amounts of throughput processed at once. However, this may not be adequate for money counters, as compact money counters can process one bank note every 0.06 seconds, with the performance increasing up to one bank note every 0.02 seconds with more powerful devices [2].

7 Conclusion

The proposed bank note recognition system was able to correctly classify 96.6% of the tested bank notes, with a FRR and FAR of 4% and 0%, respectively, and an average elapsed time per bank note of 0.4 seconds. While the system successfully rejected all non-genuine bank notes, it should be tested with actual counterfeit South African bank notes before it could be considered for real world applications or be incorporated in current hardware systems used to evaluate the legitimacy of South African bank notes.

There are numerous ways in which this system could be improved. Firstly, the system could be extended to automatically detect the regions to segment for feature extraction. This would allow the program to be used in multiple countries with any currencies, without needing an operator to first visually inspect the bank notes and identify the regions to segment. Since the neural networks were the cause of the false rejections, improving these would yield better results. This could be done in various ways, such as decreasing the number of features used, training for a different number of iterations or using a max error allowance to stop the training, changing the hyperparameters of the neural network, or by using a larger and more comprehensive training dataset. Further work could involve the use of other machine learning techniques such as, support vector machines(SVM) or the naive bayes classifier and results can be compared with our proposed model.

If we needed to extend the system to recognize bank notes in an environment where they are partially hidden or at obscure angles, or improve the throughput of our system by classifying more bank notes per second, we would make use of an existing feature detection and matching algorithm, which has been extensively tested, such as SURF or ORB, rather than creating our own matching implementation based on the GLCM and Haralick features with custom neural networks as was done with this system.

Bibliography

[1] Satish, K., Viswandadham, Y.K., Leela Priya, I.: Money to ATM – Fake Currency Detection. International Journal of Computer Science and Information Technologies, vol. **3**(5), 5046–5050 (2012)

[2] Cash Counter|G+D: Cash Counters in use: Advantages and Key functions, https://www.gi-de.com/en/za/currency-technology/trends/cash-cycle-insights/cash-counter/. Last accessed 16 Aug 2020

[3] Hasanuzzaman, F.M., Yang, X., Tian, Y.: Robust and Effective Component-based Banknote Recognition for the Blind. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, **42**(6), 1021–1030 (2012)

[4] Olanrewaju, R.F., Eniola, F.F., M., S.M.M.: Automated Bank Note Identification System for Visually Impaired Subjects in Malaysia. International Conference on Computer and Communication Engineering (ICCCE), 115–120 (2016)

[5] Yousry, A., Taha, M., Selim, M.M.: Currency Recognition System for Blind people using ORB Algorithm. International Arab Journal of e-Technology, vol. **5**(1), 34–40 (2018)

[6] NumPy: The fundamental package for scientific computing with Python, https://numpy.org/. Last accessed 10 Aug 2020

[7] OpenCV: Open Source Computer Vision Library, https://opencv.org/. Last accessed 10 Aug 2020

[8] scikit-image: Image processing in python, https://scikit-image.org/. Last accessed 10 Aug 2020

[9] scikit-learn: Machine learning in python, https://scikit-learn.org/stable/. Last accessed 10 Aug 2020

[10] Costa, C.M., Veiga, G., Sousa, A.: Recognition of Banknotes in Multiple Perspectives Using Selective Feature Matching and Shape Analysis. IEEE International Conference on Autonomous Robot Systems and Competitions, 235–240 (2016)

[11] Jaitley, U.: Why Data Normalization is necessary for Machine Learning models, https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029. Last accessed 10 Aug 2020

[12] Machine Learning in Computer Vision, https://fullscale.io/machine-learning-computer-vision/. Last accessed 11 Aug 2020

[13] jamesloyys (James) · Github: neural_network_backprop.py, https://gist.github.com/jamesloyys/ff7a7bb1540384f709856f9cdcdee70d. Last accessed 02 Aug 2020