Bank Note Recognition

Nikita C. Tirumol
Sharad B. Singh
Praveer Ramphul

School of Mathematics, Statistics, and Computer Science, University of Kwazulu-Natal, Private Bag Box X54001, Durban 4000, South Africa

217014607@stu.ukzn.ac.za

217079609@stu.ukzn.ac.za

217011174@stu.ukzn.ac.za

Abstract- In this paper, an approach to South African bank note recognition is taken on. It provides a simple, straightforward method of recognising bank notes by allowing users to select images , with which it performs multiple steps of pre-processing, followed by enhancement, feature extraction and classification of bank notes. It serves as a proof of concept for future research in the area to assist the visually impaired in their day-to-day lives by helping them differentiate between South African bank notes.

Keywords- Banknote, SVM, Computer Vision, Image Processing, ORB, Haralick, Gaussian

---

I. Introduction

According to the World Health Organisation (WHO), as of 2002, approximately 2.6% of the world's entire population were visually impaired[1]. These people have a hard time with navigating environments and performing everyday tasks. Those of which include running errands and making purchases. Identifying monetary resources is a task that we take for granted, yet it provides a challenge to visually impaired people.

Since the 7th century, banknotes were being developed and in the 11th century, paper money was introduced in China[2]. Until now, bank notes have been one of the most prevalent forms of currency. South Africa's form of currency is the Rand which can be represented physically by use of a multitude of coins, ranging from 10c to R5, as well as banknotes which include the R10, R20, R50, R100 and R200 notes[3].

There are features built into South African coins for a visually impaired person to identify the coins easily, however, while South African banknotes have these features as well[4], it is still not as easy to identify them by touch. Software that can identify South African banknotes would be a useful tool for identifying and differentiating between them.

While people may have visual impairments, computer vision is a growing field[5] and has the ability to be a third eye to be people with poor or no vision. Computer vision has the tools needed, such as pattern recognition[6], to identify objects, which, in this context would be banknotes.

In this paper we see the use of computer vision, in which there is Pre-processing and Enhancement, Segmentation, Feature Extraction, as well and Classification. These concepts are used in conjunction to allow the software to read, identify and classify banknotes and output the details to the user.

II. Pre-processing and enhancement

Pre-processing:
The input image could be one of multiple image formats. For example, a .jpeg, .bmp or .png file could be used. The image is first resized[Figure 1, 2]. A resized or smaller image makes computing lighter and results in faster processing speeds. When representing colour images mathematically, a stack of 3-arrays are used. This is since, according to Bayes' theory, colours are represented digitally with varying intensities of red, green and blue light[7]. Due to the heavy computing needed to analyse 3 arrays, the image is converted to greyscale[Figure 3] and a Gaussian filter[Figure 3, 4] is applied to it. This assists in reducing the noise of the new greyscale image[8].

```
def resize_img(image, scale):
    res = cv2.resize(image, None, fx=scale, fy=scale, interpolation = cv2.INTER_AREA)
    return res
```

Figure 1: The resize_image method.

```
original = resize_img(original, 1.2)
```

Figure 2: Calling the resize_image method.

```
# convert image to grayscale
def img_to_gray(image):
    img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return img_gray


# gaussian blurred grayscale
def img_to_gaussian_gray(image):
    img_gray = cv2.GaussianBlur(img_to_gray(image), (5, 5), 0)
    return img_gray
```

Figure 3: The img_to_gray and img_to_gaussian_gray methods.

```
original1=img_to_gaussian_gray(original)
```

Figure 4: Applying img_to_gaussian_gray method to input image.

After applying these methods to the original image, the following images are the state in which the image now is, compared to the original input image[Figure 5, 6].

Figure 5: The original image, as compared to the partially grey image.

Figure 6: The image with Gaussian Blurring performed with Gaussian filter.

The Gaussian Blur was able to remove the noise in the image and it was able to maintain the sharpness in the image , this coupled together with its fast processing speed means that it is a good choice for removing noise.
While there are different methods that can be used within the pre-processing phase of object recognition, the chosen methods are those that were specified above. We could not use a standard dimension for the height and width as the images that the software will need to work with are of differing dimensions and it would lead to a distorted image

Enhancement:
Once the pre-processing is complete, it is now time to perform image enhancement. Image enhancement is done to improve picture quality. Due to the different possible environments in which images can be taken, there are sometimes bad lighting, shadows or other factors that can depreciate the quality of an image. Image enhancement is a way of curbing this issue by using software.
Histogram Equalisation is a possible method that could be used on the image. This is a method of evening out the lighting within the greyscale image. When an image's histogram is weighted towards one side or the other, it indicates that the image is overall dark or bright. Histogram Equalisation rectifies this by redistributing frequencies of different pixel intensities. A flatter Histogram represents an image with a more even light distribution[9].
Possible alternatives to Histogram Equalization are Median Filtering[10], Decorrelation Stretch[11] and Unsharp Mask Filtering[12].
Histogram Equalisation was chosen due to its robust way of improving image contrast. This makes it easier to detect smaller details in images in order to identify and classify the bank note.

```
classifyImage = cv2.equalizeHist(classifyImage)
```

Figure 7: The code snippet used to perform Histogram Equalisation

Other enhancement methods may be used in conjunction, however, for the implementation of this paper, no other methods were used as they seemed to alter results.

Figure 8: The image after performing Histogram Equalisation.

Figure 9: Denoising

The Denoising filter does a great job of removing noise whilst also retaining sharpness around the edges. When compared to the Gaussian blur it takes slightly longer but the results are better.
CODE: cv2.fastNlMeansDenoising(original3,None,3,7,21)

Figure 9: Median Blurring
CODE: cv2.medianBlur(original3,3)

The median blur does a worse job than both the Gaussian and Denoising filter. The sharpness of the image deteriorates slightly too much and as such we did not use this method, The Denoising is slightly better than the Gaussian blur however after rigorous testing with the ORB and Sobel Edge detection , the Gaussian blur appears to give better accuracy when determining notes..

III. Bank note segmentation

Image segmentation is a useful technique to detect objects[13]. The idea behind it is that it breaks up the image in question in order to use the different segments for comparison. It works by ensuring that only important bits of data are used and that unnecessary bits are left out. There are different methods that can be used to perform image segmentation.

Sobel Edge Detection

Sobel Edge Detection is a technique which performs edge detection. It performs 2-D spatial gradient measurement which emphasises areas with high spatial frequency and outputs an image with edges that are shown distinctly[14]. From comparing it to other methods, it was decided that this is the most reliable method of consistently providing accurate results, hence, sobel thresholding was the preferred method for performing edge detection..

CODE:

```
grad_x = cv2.Sobel(image, cv2.CV_16S, 1, 0, ksize=3, borderType = cv2.BORDER_DEFAULT)
```

```
grad_y = cv2.Sobel(image, cv2.CV_16S, 0, 1, ksize=3, borderType = cv2.BORDER_DEFAULT)

abs_grad_x = cv2.convertScaleAbs(grad_x)
abs_grad_y = cv2.convertScaleAbs(grad_y)

dst = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
return dst
```

Figure 9: Before and after performing sobel thresholding.

Binary thresholding

Binary thresholding is a simple method of performing image segmentation[15]. Starting with a greyscale image, when binary thresholding is performed, pixels within the image become either black (value of 0) or they become white (value of 255). The decision is done by changing pixels with values lower than a specified threshold to 0 and those that are higher to 255[Figure 10].

Figure 10: Before and after binary thresholding on an Image with not background.

CODE

```
cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)
```

While binary thresholding may serve as a good option to make features stand out, the threshold may need to be changed constantly, depending on the lighting of the image. If the threshold is set incorrectly for the image at hand, the following can occur, as shown in [Figure 11].

Figure 11: The original, compared to an image after binary thresholding with a threshold of 65.

For this reason, binary thresholding was not used in order to ensure that the software remains robust and accurate in different lighting conditions.

Adaptive thresholding

Adaptive thresholding is another method that could be used. The method that it follows is, in theory, a good option. It also makes use of a threshold, however, the threshold is not applied to the entire image, but rather to different regions within the image[Figure 12]. This means that each region could have a different threshold. While this may seem viable, it was found that the adaptive thresholding method introduces 'specks' which is interpreted as noise and ultimately alters the result of the detection of bank notes. Using adaptive thresholding on a picture that included a background like in figure 11 resulted in a similar response as figure 11.

Figure 12: The image before and after adaptive thresholding is applied.

CODE:
cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 8)

Canny edge detection was also tested but it resulted in an image that has multiple regions of lines and did not provide good results. If our program only had to detect fully cropped images of bank notes with no background as in figure 10 and figure 12 the binary threshold is shown to give the best results however as in figure 11 our program includes non cropped images and as such thresholding resulted in unusable results. Sobel edge detection , although not perfect was decided on as it gave consistent results when using ORB Detection

IV. Banknote feature extraction

Feature extraction is a useful and much needed process in object detection. This process reduces the amount of resources required as well as lowers the amount of redundant data[16]. For feature extraction in this paper, ORB was used. ORB (Oriented FAST and Rotated BRIEF) is a detector that was developed as an alternative to SURF (Speeded Up Robust Features)[17] and SIFT (Scale-Invariant Feature Transform)[18].

ORB had provided the best results, as compared to other methods such as SIFT and SURF and had been used to extract features of the bank notes. It was chosen due to its robustness as well as performance, by completing the extraction in a shorter time than the SIFT and SURF methods. Our ORB was set as follows:
orb= cv2.ORB_create(nfeatures=12000)
Extensive Testing was performed and 12000 nfeatures was to give the best results.
The image after processing was passed through a or detector as such:
(kp1, des1) = orb.detectAndCompute(original, None)
This is image was then compared to 45 “Training Images” :
These training images were the Front, Back, Obverse,Reverse, Front_Large, Upside Down,Vertical,back Upside down and back vertical for each Currency Denomination.
Increasing the amount of images in the training set increased the accuracy of the results substantially

```
(kp1, des1) = orb.detectAndCompute(original3, None)
keys=cv2.drawKeypoints(original3,kp1,None)
cv2.imshow("Keypoints",keys)
```

Figure 13: Code snippet used to call the ORB detector.

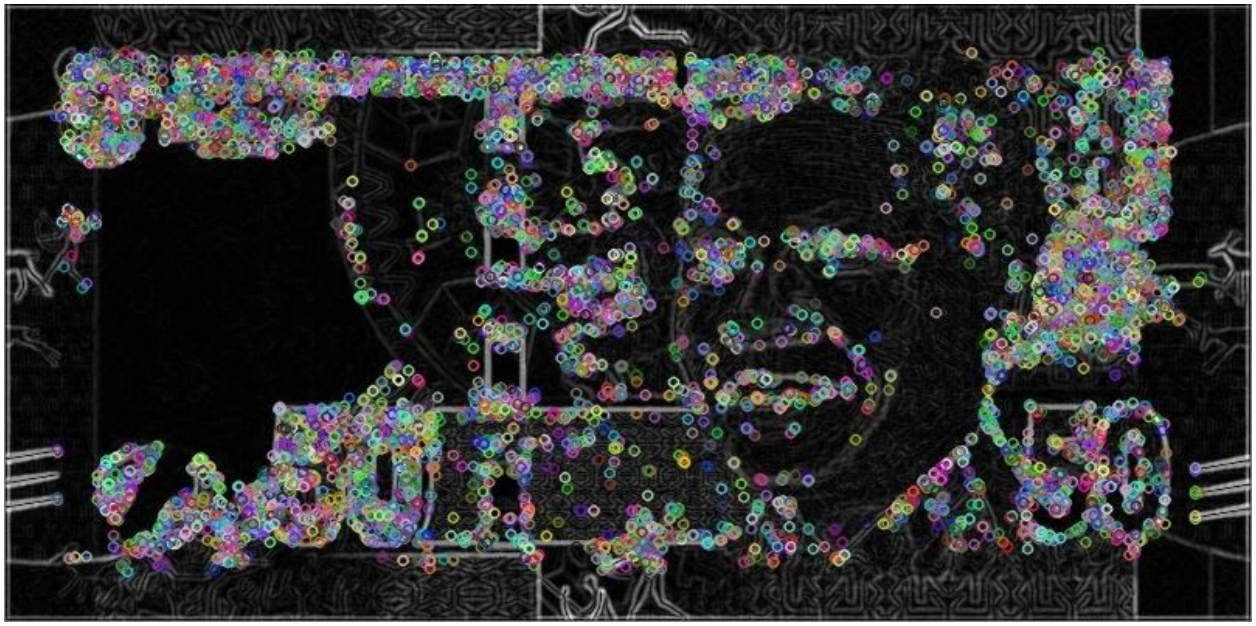

Figure 14: Before and after the ORB detector is applied to an image with no Background.

```
13   files/20front_large.jpg   267
14   files/20busd.jpg   68
15   files/20bvert.jpg   65
16   files/20fusd.jpg   56
17   files/20fvert.jpg   73
18   files/50front.jpg   8729
19   files/50back.jpg   77
20   files/050obverse.jpg   161
21   files/050reverse.jpg   58
22   files/50front_large.jpg   1288
23   files/50fusd.jpg   72
24   files/50fvert.jpg   7690
25   files/50bvert.jpg   54
26   files/50busd.jpg   70
27   files/100back.jpg   66
28   files/100front.jpg   555
29   files/100obverse.jpg   125
30   files/100reverse.jpg   72
31   files/100front_large.jpg   373
32   files/100fusd.jpg   552
```

Figure 15: Results from Cropped 50 with no background

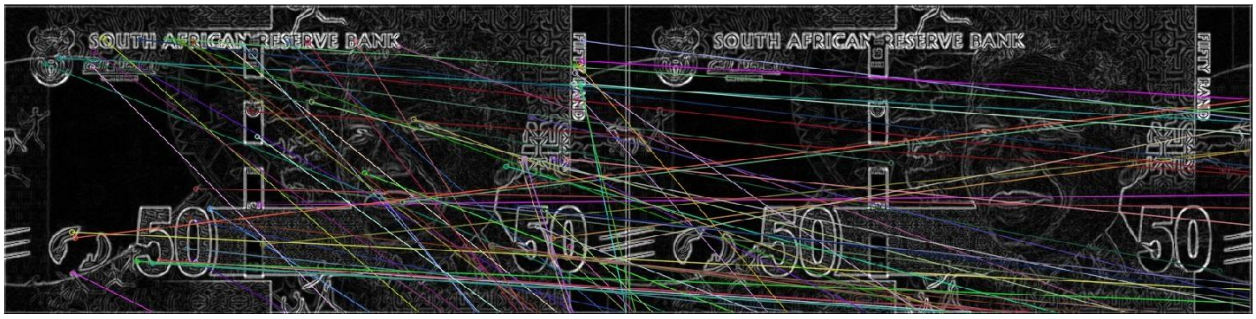

Figure 16: KeyPoint Comparison between Test and Trained Image

Figure 17: A 100 Rand Note with Background

Figure 18: A 100 Rand Note with Background After Processing

```
27   files/100back.jpg   231
28   files/100front.jpg   127
29   files/100obverse.jpg   137
30   files/100reverse.jpg   151
31   files/100front_large.jpg   138
32   files/100fusd.jpg   131
33   files/100fvert.jpg   145
34   files/100bvert.jpg   234
35   files/100busd.jpg   230
36   files/200back.jpg   157
37   files/200front.jpg   119
38   files/200obverse.jpg   138
39   files/200reverse.jpg   141
40   files/200front_large.jpg   149
41   files/200fusd.jpg   127
42   files/200fvert.jpg   111
43   files/200bvert.jpg   99
44   files/200busd.jpg   124

Detected denomination: R  100bvert.
Completed in:  48.7  seconds
```

Figure 19: A 100 Rand Note with Background results

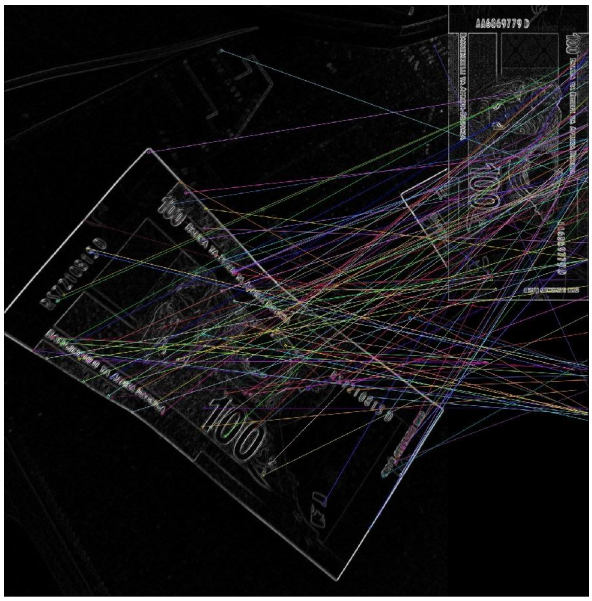

Figure 20a and 20b: A 100 Rand Note with Background results

Discussion From Figure 14-16 we see the use of the ORB detector on an Image that has no background and from Figure 17 - 20 we see the ORB detector on an image with a black background. The results for the 50 rand are exceptional with the correct note being identified and a match score of 8729 whilst the other scores were all below 1000(for non 50 rand fronts). The KeyPoints do not match up well in figure 16 but the result is still Excellent

nonetheless. The results for the 100 Rand note on a table are also accurate but the effect of having a background is clearly visible with a match score of 230 with all the other scores being below 190(other than 100 rands backs). We can see from the detection points that it matches with areas on the desk that are not the note but it still does a good enough job to accurately detect the correct note.As mentioned earlier we would normally want a light image that used thresholding but due to backgrounds being included in our scope we decided to use a darker image with its edges highlighted. When compared to a thresholded image(figure 11 and 12) our model outperformed it for nearly every test.

V. Banknote classification

Four methods were attempted for bank note classification. These methods were :
Convolutional Neural Network multi class classifier
Convolutional Neural Network binary classifier
Support Vector Machine
KNN Brute Force Matching

Support Vector Machine

For the classification of banknotes, 2 methods are used. In order to determine whether the image being tested is of a bank note or not, a Support-Vector Machine (SVM)[19] is used. SVM is an algorithm that originated in statistics and works by simply creating a line to separate data into classes[20]. This is how the software determines which note is being identified. We used Haralick feature extraction on notes that went through GreyScaling, Gaussian Blurring and Sobel Edge Detection. Harilick Feature extraction is available on the Mahotas Python library. The SVM was created using SciKit Learn. We took 678 Pictures of South African Bank Notes to train the SVM as well as 550 Negative Images. The SVM was originally trained using only the Stock bank note Images and did a decent job of determining what is and is not a bank note. This however only worked for the Stock bank Note Images and once Rotated images or images with a background were introduced and the model was retrained , The SVMs accuracy deteriorated by a fair amount. In order to mitigate the problem the SVM was used in conjunction with a Threshold on the ORB detector. This provides an excellent result and determines what is and isn't a bank note with a high accuracy. [21]

```
img1=img_to_gaussian_gray(img1)
img1= sobel_edge2(img1)
features = extract_features(img1)

label = "Note"
train_features.append(features)
train_labels.append(label)
print("Iteration Number (", (i+1),") Image ", (imageIndex+1), " of ", len(train_names), " done")
imageIndex +=1
```

Figure 21: SVM implementation code

K-Nearest Neighbours(KNN Matching)

In this implementation a brute force matcher along with KNN was utilized.The Norm_Hamming implementation of Brute force matchers use the descriptors of one feature set and match all of the features in the second set using Hamming distance calculation. The KNN is given a trained dataset for the input we want to classify and will find K instances closest to the instance in the training dataset, then will use the most frequent label(classification category). [21]

```
(kp2, des2) = orb.detectAndCompute(tr4, None)

# brute force matcher
bf = cv2.BFMatcher(cv2.NORM_HAMMING)


all_matches = bf.knnMatch(des1, des2, k=2)

good = []
# give an arbitrary number -> 0.789
# if good -> append to list of good matches
for (m, n) in all_matches:
    if m.distance < 0.789 * n.distance:
        good.append([m])

if len(good) > max_val:
    max_val = len(good)
    max_pt = i
    max_kp = kp2

print(i, ' ', training_set[i], ' ', len(good))
```

Figure 22: KNN Code Implementation

Convolution Neural Network(CNN)

A CNN is a neural network responsible for image recognition and image classification. The neural network is composed with layers, each containing a collection of neurons which takes input and produces some output. These neurons have associated weights which are manipulated during the training phase which will allow the classifier to adapt to the given problem and dataset supplied. Hence, the more data we use for CNN, the better the prediction can be made by this classifier. For this paper, a multi-classification CNN was implemented and a binary classification CNN was implemented.

Multi-classification CNN

The multi-classification CNN of this paper was trained using RGB images and no filters were applied. Both old notes and new notes of one type were classified under one category. A sequential mode of seven 7 layers were used for the CNN with activation function relu and softmax (considering as layers).

```
model = Sequential()
model.add(ZeroPadding2D(input_shape=(64, 64, 3), padding=(3, 3)))
model.add(Conv2D(32, (7, 7), strides=(1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(6))
model.add(Activation('softmax'))
```

Figure 23: layers for MultiClass CNN

The dataset used was uneven. Each image had differing sizes and hence were resized to 64 by 64. The amount of each type of rand the CNN was trained also differed. Due to their being no readily available dataset we took a large number of images by phone.

It also differed in the amount of rotations for each note. For example, ten images of the front of the R10 were and five images of the back of the R10 trained the image.
Due to the imbalance in the amount of data for each note type, we set epochs to 20 so each rand type with smaller data were not overfitted and each data with large data amounts were not under fitted. 20% of the images from the dataset were also used for testing. [22]

Binary classification CNN

In the binary CNN classification, we classified images according to whether they were a bank note or not. If it was a bank note we proceeded to implement the KNN matching to determine the type of note. For this implementation we increased our dataset utilizing several more pictures of differing rotations and brightness stimulating different lighting conditions. The dataset still had differing image sizes and hence we resized 320 by 180. There were 18 layers used to create the neural network for this model, again including activation functions as a layer [22]

```
model = Sequential()
model.add(Conv2D(input_shape=(180, 320 ,3),padding='same',kernel_size=3,filters=16))
model.add(LeakyReLU(0.1))
model.add(Conv2D(padding='same',kernel_size=3,filters=32))
model.add(LeakyReLU(0.1))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(padding='same',kernel_size=3,filters=32))
model.add(LeakyReLU(0.1))
model.add(Conv2D(padding='same',kernel_size=3,filters=64))
model.add(LeakyReLU(0.1))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256))
model.add(Dropout(0.5))
model.add(LeakyReLU(0.1))
model.add(Dense(2))
model.add(Activation('softmax'))
```

Figure 22: Layers for Binary Classification CNN

VI. Results and discussion

Numerous different methods were attempted to attain the best results. These methods include:
ORB with SVM & KNN
ORB with CNN & KNN
Multi-Class CNN

The ORB does an excellent job of determining the denomination of the Bank Note. We used pre-processing methods along with gaussian blurring and Sobel Edge Detection for the ORB method. Through rigorous testing we were able to find the perfect balance in order to obtain accurate results. That being said the ORB did return some false positives i.e. it would determine that a Beach or House was a note. In order to fix this we proposed two methods the first of which was to use a CNN to classify the notes as either a Bank Note or Not. We used the stock

Images as well a few extra stock images to train the CNN using a Binary Classifier but the results were poor and after much testing it was decided that we would try a SVM next.

The SVM did a good job of determining what was a Note and Not a note when we only used stock images to train it and when we only used stock images to test it. When we added rotated notes and notes with a background the SVM performed worse than before and sometimes for example mistook houses for notes. In order to fix this we added more images to the ORB training set which resulted in more accurate ORB detection and matching point. This means we were able to raise the threshold on the ORB detector when classifying images. We the following IF statement to determine whether an image was a note or not: prediction=="Note" and max_val >150. This performs excellently and determines notes from not notes as well as the NOTe denomination.
In Order to determine whether a note is fake or not we attempted to solve the problem using a MultiClass CNN. We took an additional 2500 images of notes to train the CNN:

The CNN trained well with epochs up to 20, giving an accuracy of approximately 92% at the final epoch-(visible in figure below). However, when we supplied other images out of the dataset to test the CNN it predicted some of them incorrectly, i.e. R20 as a R200. We suspect this is because of the same visual color that the two images share. Due to the images being different sizes, they had to be resized also making them obscure. Majority images were also taken by phone which also resulted in some unclear images as well images with bad lighting. We unfortunately had to scrap this idea however for future works we would like to implement a CNN to accurately determine fake notes from real ones. This is something we hope to expand on.

```
Epoch 14/20
9/9 [==============================] - 1s 164ms/step - loss: 0.4571 - accuracy: 0.8441
Epoch 15/20
9/9 [==============================] - 2s 171ms/step - loss: 0.3939 - accuracy: 0.8638
Epoch 16/20
9/9 [==============================] - 1s 166ms/step - loss: 0.3512 - accuracy: 0.8799
Epoch 17/20
9/9 [==============================] - 1s 157ms/step - loss: 0.2980 - accuracy: 0.9014
Epoch 18/20
9/9 [==============================] - 1s 156ms/step - loss: 0.2596 - accuracy: 0.9194
Epoch 19/20
9/9 [==============================] - 1s 157ms/step - loss: 0.2314 - accuracy: 0.9194
Epoch 20/20
9/9 [==============================] - 1s 157ms/step - loss: 0.2184 - accuracy: 0.9247
```

Figure 22: Results from Multi Class CNN

VII. References

[1] I. Kocur, R. Parajasegaram, and G. Pokharel, “Global data on visual impairment in the year 2002,” Bull. World Health Org., vol. 82, pp. 844–851, 2004.

[2] Daniel R. Headrick (1 April 2009). Technology: A World History. Oxford University Press. pp. 85–. ISBN 978-0-19-988759-0.

[3] South african rand. (n.d.). Retrieved August 15, 2020, from https://www.globalexchange.es/en/currencies-of-the-world/south-african-rand

[4] Hoosain, S. (2012, November 08). SA National Council for the Blind Commends New Mandela Series Notes. Retrieved August 15, 2020, from http://www.ngopulse.org/press-release/sa-national-council-blind-commends-new-mandela-series-notes

[5] Mihajlovic, I. (2020, February 09). Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome. Retrieved August 15, 2020, from https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

[6] Deguchi, K. (2011). Introduction to Pattern Recognition and Computer Vision. *Interdisciplinary Information Sciences, 17*(2), 49-129. doi:10.4036/iis.2011.49

[7] Kumar, T., & Verma, K. (2010). A Theory Based on Conversion of RGB image to Gray image. *International Journal of Computer Applications, 7*(2), 5-12. doi:10.5120/1140-1493

[8] Brar, K. (2020, January 14). Part 2 -Remove noise in an image using Gaussian Blur. Retrieved August 16, 2020, from https://medium.com/@kumar.brar/part-2-remove-noise-in-an-image-using-gaussian-blur-85b1b3a1cb4f

[9] Karnewar, A. (2018, October 12). Back-to-basics Part 1: Histogram Equalization in Image Processing. Retrieved August 16, 2020, from https://medium.com/@animeshsk3/back-to-basics-part-1-histogram-equalization-in-image-processing-f607f33c5d55

[10] Huang, T., Yang, G., & Tang, G. (1979). A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 27*(1), 13-18. doi:10.1109/tassp.1979.1163188

[11] Guo, L. J., & Moore, J. M. (1996). Direct decorrelation stretch technique for RGB colour composition. *International Journal of Remote Sensing, 17*(5), 1005-1018. doi:10.1080/01431169608949060

[12] Fujita, M., Kodera, Y., Ogawa, M., Tanimoto, K., Sunayashiki, T., Wada, T., & Doi, K. (1987). Digital image processing of dentomaxillofacial radiographs. *Oral Surgery, Oral Medicine, Oral Pathology, 64*(4), 485-493. doi:10.1016/0030-4220(87)90158-7

[13] Haralick, R. M., & Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing, 29*(1), 100-132. doi:10.1016/s0734-189x(85)90153-7

[14] Kazakova, N., Margala, M., & Durdle, N. (n.d.). Sobel edge detection processor for a real-time volume rendering system. *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*. doi:10.1109/iscas.2004.1329421

[15] Bai, X., & Zhou, F. (2007). Edge Detection Based on Mathematical Morphology and Iterative Thresholding. *Computational Intelligence and Security Lecture Notes in Computer Science,* 953-962. doi:10.1007/978-3-540-74377-4_100

[16] DeepAI. (2019, May 17). Feature Extraction. Retrieved August 17, 2020, from https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

[17] Pang, Y., Li, W., Yuan, Y., & Pan, J. (2012). Fully affine invariant SURF for image matching. *Neurocomputing, 85*, 6-10. doi:10.1016/j.neucom.2011.12.006

[18] Sun, Y., Zhao, L., Huang, S., Yan, L., & Dissanayake, G. (2014). -SIFT: SIFT feature extraction and matching for large images in large-scale aerial photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing, 91*, 1-16. doi:10.1016/j.isprsjprs.2014.02.001

[19] Zhang, L., Zhou, W., & Jiao, L. (2004). Wavelet Support Vector Machine. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 34*(1), 34-39. doi:10.1109/tsmcb.2003.811113

[20] Pupale, R. (2019, February 11). Support Vector Machines(SVM) - An Overview. Retrieved August 17, 2020, from https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[21] Rani, R., Kumar, R., & Singh, A. P. (2019). Implementation of ORB and Object Classification using KNN and SVM Classifiers. *International Journal of Computer Sciences and Engineering, 7*(3), 280-285. doi:10.26438/ijcse/v7i3.280285

[22] Binod Prasad Yadav. “Indian Currency Recognition and Verification System Using Image Processing”. In: International Journal of Advanced Research in Computer Science and Software Engineering 4.12 (2014)