ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.5

Forex Detection using Neural Networks in Image Processing

Aditya Shettigar[1], Priyank Singal[2]
BE Student[1, 2]
Department of Computer Engineering
K.J Somaiya College of Engineering, Vidyavihar, Mumbai, India

Abstract:
Currency Recognition is performed to diminish human energy to naturally perceive the money related estimation of cash and change. This article presents a system that could be utilized for different monetary standards. The fundamental goal of the study is to outline a simple yet proficient algorithm using Image Processing and Artificial intelligence that would be valuable for most extreme number of monetary standards, i.e to design one algorithm that could be used for recognition of all available currencies.
General Terms, Currency detection, neural networks, Image processing

Keywords: Extraction, Training, Classification

1. INTRODUCTION

There are around 200+ unique monetary standards utilized as a part of various nations around the globe. Humans can't perceive monetary standards of various nations effectively. The current technological advances make it easy to build a computerized framework to recognize and check its denomination. The money will be confirmed by using Image Processing (IP) and Aritificial Intelligence (AI) techniques. The approach to forex detection can be divided in five parts. The whole framework is outlined in MATLAB. A suitable dataset of several currency models is created. Image Processing is used to perform feature extraction on this dataset. This feature extracted data is trained to a neural network. Using a normalized Gradient Descent Algorithm the datasets in the neural network are classified based on Mean Squared error. Target image is inserted in the system to verify it's currency model and denomination. The general flowchart of the system is illustrated in Figure 1. The aim of the overall system is to provide a reliable and useful currency detection and analytical system for users both business and non-business alike.

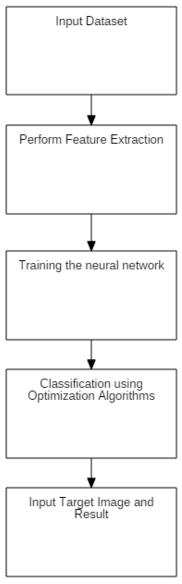

Input Dataset
Perform Feature Extraction
Training the neural network
Classification using Optimization Algorithms
Input Target Image and Result

Figure.1. Flowchart

2. EXISTING SYSTEMS

International banks as well as business users employ currency counting machines with currency detection systems. These systems operate on a hardware level. Accountants use hardware specific devices for currency marking and detection. Systems based on a software level using IP and AI is rare in the real world. Existing systems in banks use hardware systems or software recognition but public availability of such systems is limited.

3. PROPOSED MODEL

Neural Network is an efficient tool for Pattern matching and recognition. In our proposed system the image dataset is processed through different modules in the Forex Detection system. These modules are detailed as follows.

3.1 CREATE DATASET

To train a neural network, a dataset is created consisiting of several images of different currency models. For optimal training and classification using the extracted features, the images should be of resonable quality with the dimensions of atleast 1500x700 and 150 dpi.

Difference in image sizes will not affect the working since the pre-processing conforms all datasets to standard values. We create a decent quality dataset for every currency model to improve accuracy and reduce false detections.

3.2 FEATURE EXTRACTION

Every currency note in the world employs strong patterns that feature sharp edges. These edge patterns can be extracted to classify and perform reognition.

The input image from the dataset undergoes gray scale image conversion as a preprocessing step after resizing of the image as shown in figure 2. All the images from every dataset are preprocessed using a loop.

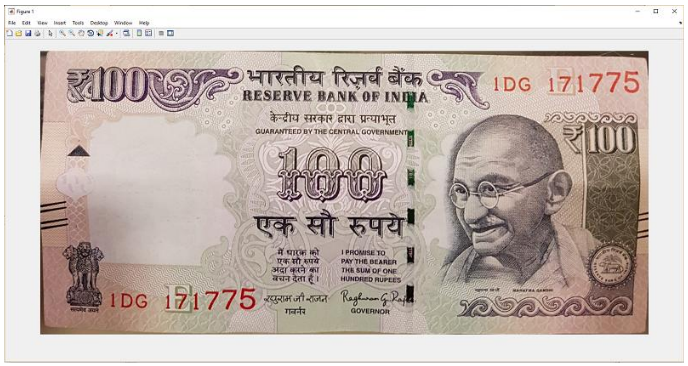


Figure.2. (a): Original Image

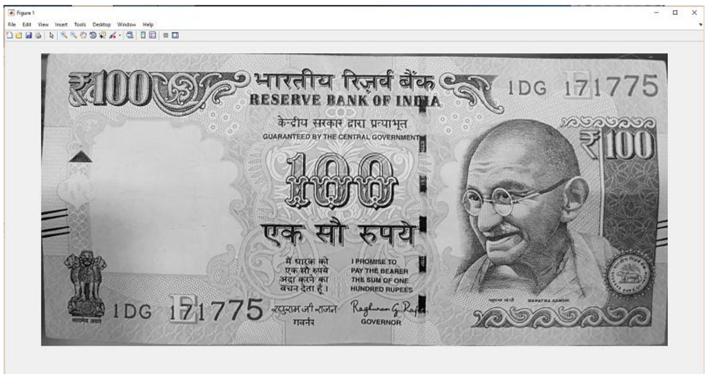


Figure.2. (b): Greyscale Image

This gray scale image is subjected to a custom Sobel edge Detection algorithm. The Sobel operator is used particularly within edge detection algorithms where it creates an image emphasising edges. The algorithm employs sobel masks in 2 directions as follows:

The magnitude of the vector Δf is denoted as,

$$\Delta f = mag(\Delta f) = [G_x^2 + G_y^2]^{1/2}$$

where Gx is for x direction and Gy for y direction.

The sobel masks (3x3):

For x-Direction:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

For Y-direction:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(1)

To avoid complex computation, the gradient can also be computed using the formula:

$$\Delta f \approx |Gx| + |Gy|$$

(2)

Sobel edge detection detects the rough edges from the greyscale image giving us a sharp pattern simplifying classification. Sobel image is illustrated in figure 3(a).

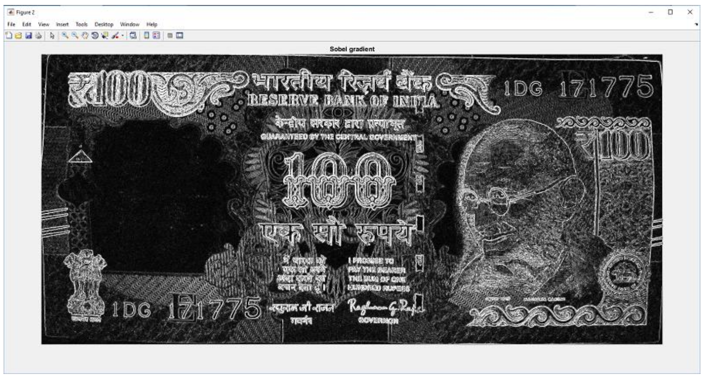


Figure.3. (a): Sobel Image

To obtain a better result, we consider the intensity values of the image which range from 0-255. Choosing an appropriate threshold, we perform thresholding over the sobel image to seperate the edges from background. This image is then inverted to generate the feature extracted image illustrated in figure 3(b).

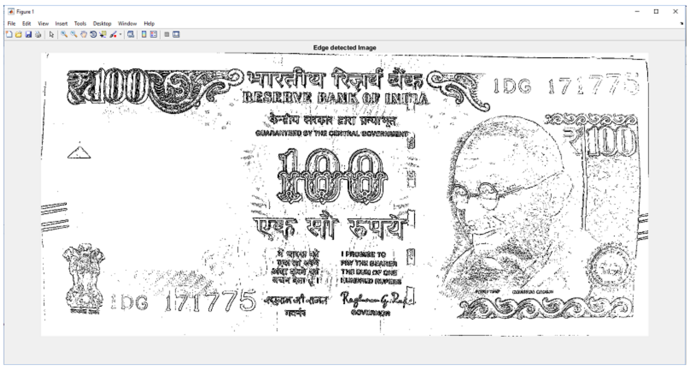


Figure.3. (b): Threshold Inverted Image

All the images in the dataset are subjected to the following steps of pre-processing and feature extraction. These images are then stored in two-dimensional arrays (matrices) for further procedures.

3.3 TRAINING THE NEURAL NETWORK

In training phase we employ machine learning using a set of neural networks. Machine Learning is a current application of AI upon which the heart of our system is based. We implement training using the Neural Network toolbox available in MATLAB. The feature extracted data that is stored in the arrays are fed to the neural network in the form of two arrays namely input layer and output layer. The Input layer contains the feature extracted data of all datasets concatenated together. The Output layer will contain the training dataset to be inserted in the neural network training tool. We set a custom number of hidden layers which feeds the data to the output layer. Training is implemented using the feed-forward propagation method as displayed in figure 4.

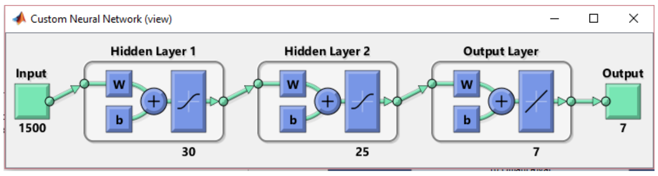
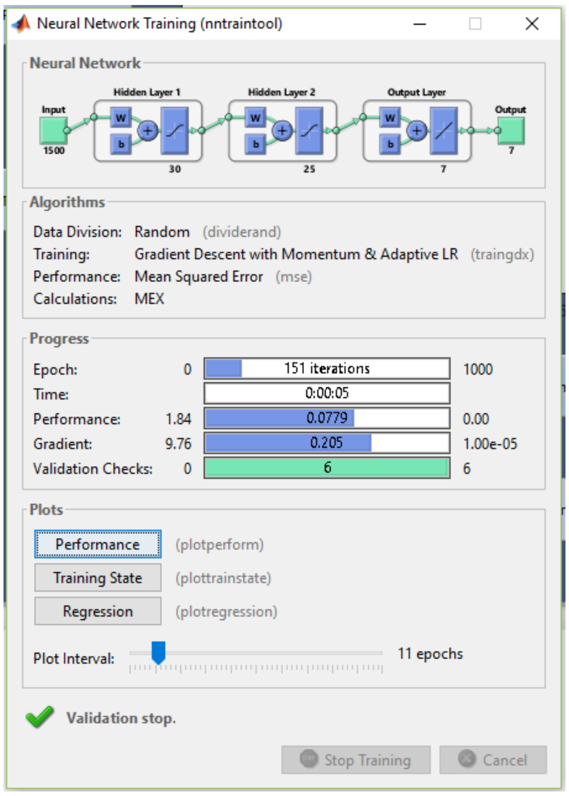


Figure.4. Network

The dataset is trained to the neural network again and again using the nntool built in the MATLAB toolbox. The nntool runs the training process for a random number of iterations every time the process is started. The iteration count is proportional to the performance of the neural network classifier. The nntool is illustrated below in figure 5.


Neural Network Training (nntraintool)

Neural Network

Input
1500
Hidden Layer 1
W
b
30
Hidden Layer 2
W
b
25
Output Layer
W
b
7
Output
7

Algorithms

Data Division: Random (dividerand)
Training: Gradient Descent with Momentum & Adaptive LR (traingdx)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	151 iterations	1000
Time:		0:00:05	
Performance:	1.84	0.0779	0.00
Gradient:	9.76	0.205	1.00e-05
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 11 epochs

✓ Validation stop.

FIGURE.5. NNTRAINTOOL

The proposed method here is using the NN for currency Recognition in a three-layer configuration as shown in Figure 6. The hidden unit number is (30, 25) which have been decided though various experiments and trial runs of the training tool to improve the error rate.

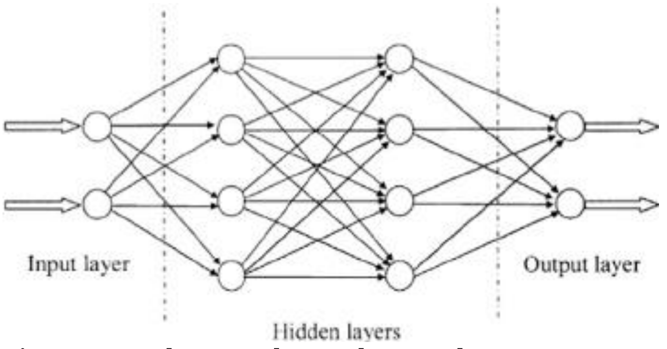


Figure. 6. Feed Forward Neural Network

3.4 CLASSIFICATION USING OPTIMIZATION ALGORITMS

Every dataset created and trained to neural network is classified using Gradient Descent Algorithm with Momentum and Adaptive Learning Rate. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. . Gradient descent is used to minimize an objective function J(θ)J(θ) parameterized by a model's parameters θ∈Rdθ∈Rd by updating the parameters in the opposite direction of the gradient of the objective function ∇θJ(θ)∇θJ(θ) w.r.t. to the parameters. Gradient Descent has difficulty navigating areas where the surface curves much more steeply in one dimension than in another, common around local optima. In these scenarios, GD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum as in Figure 7(a). Momentum is a method that helps accelerate GD in the relevant direction and dampen oscillations as seen in Figure 7(b). By adding a fraction γγ of the update vector of the past time step to the current update vector we can reduce the shortcomings of GD.

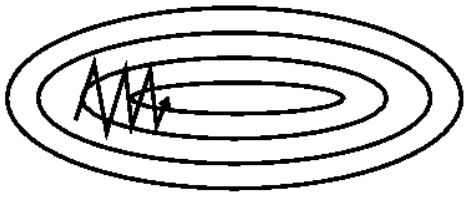


Figure.7. (a): GD without momentum

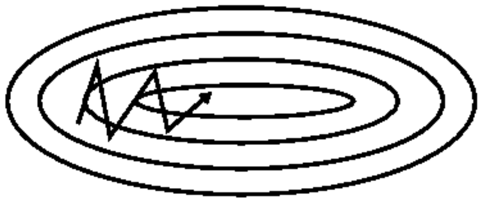


Figure.7. (a): GD with momentum

The updated equation for gradient descent is given as :

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta).$$

$$\theta = \theta - v_t.$$

(3)

To compare the values after classification using a simple if-else loop we apply Mean Squared Error in combination with Gradient Descent. The MSE is applied to the first row of every data array to calculate their final values to be inserted in the if-else loop. The cost function J for a particular choice of parameters ϴ is the mean squared error (MSE) :

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

(4)

Where the variables used are:

m	The number of training examples
$x^{(i)}$	The input vector for the ith training example
$y^{(i)}$	The class label for the ith training example
ϴ	The chosen parameter values or "weights" ($Θ_0$, $Θ_1$, $Θ_2$)
$h_\theta(x^{(i)})$	The algorithm's prediction for the ith training example using the parameters ϴ.

(5)

The MSE measures the average amount that the model's predictions vary from the correct values, so you can think of it as a measure of the model's performance on the training set. The cost is higher when the model is performing poorly on the training set. The objective of the learning algorithm, then, is to find the parameters ϴ which give the minimum possible cost J. Gradient descent is thus used in combination to find ϴ that minimizes the cost. Gradient descent is an iterative algorithm which is run many times. The learning rate gives the engineer some additional control over how large the steps are made. A plot of the function, J (Θ), and the value of Θ over ten iterations of gradient descent is given in figure 8.

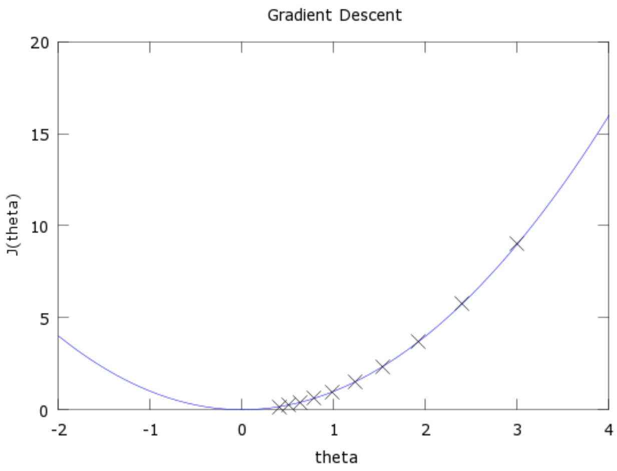


Figure.8. GD with MSE and Adaptive LR

3.5 RECOGNITION AND RESULT

This module displays the working of our proposed system. A User interface is constructed in MATLAB as shown in figure 9. The dataset first trained and then a target image is input to find the currency denomination. With proper implementation zero errors are encountered and the results are accurate. Extra functions can be added to measure the Performance of every run as well as Regression and Error histogram.

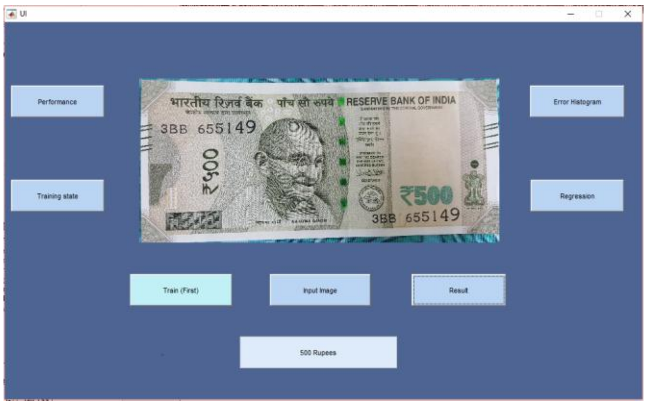


Figure. 9. System User Interface

4. ANALYSIS OF RESULTS

The analysis was carried out on a data set of 7 currency images. The statistical values obtained after applying all the said procedures and algorithms have been tabulorized below.

TABLE.1.

Serial No.	Hidden unit number	Performance
1	(10,10)	0.095959
2	(10,15,10)	0.10072
3	(20,10)	0.081129
4	(15,15)	0.095549
5	(20,20)	0.087928
6	(25,15)	0.078054
7	(30,25)	0.062747
8	(35,25)	0.080304
9	(40,30)	0.083914
10	(20,25,20)	0.095249

TABLE 2

Serial No.	Performace	Iteration Count	Regression
1	0.095959	149	0.4668
2	0.10072	147	0.4293
3	0.081129	152	0.5875
4	0.095549	148	0.4825
5	0.087928	139	0.5401
6	0.078054	156	0.6078
7	0.062747	159	0.6118
8	0.080304	150	0.5976
9	0.083914	149	0.5815
10	0.095249	140	0.4613

The following table values are noted down after several runs of testing and training. The most important point to be noted is that all the performance metrics in machine learning and neural network taining are random at best loosely depending on the quality of dataset and number of hidden layers. Only the hidden unit numbers in Table 1 are taken as constant to generate other values in both Table 1 and 2.

5. CONCLUSION

In this paper, we have proposed and devised a system to recognize various currency models using image processing and artificial intelligence. From ten currency models that we have implemented, we can implement additional models as required in the defined scope. Furthermore, the currency models can be scanned for forgery using different algorithms based on fake note detection. Additional features can be implemented as required to increase the scope of the above system. With the added flexibility of this proposed system, scope can be increased easily with time.

6. REFERENCES

[1]. https://in.mathworks.com/help/nnet/ref/traingdx.html

[2]. F. Takeda and S. Omatu, "High speed paper currency recognition by neural networks," in *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 73-77, Jan 1995.

[3]. M. Tanaka, F. Takeda, K. Ohkouchi and Y. Michiyuki, "Recognition of paper currencies by hybrid neural network," *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, Anchorage, AK, 1998, pp. 1748-1753 vol.3.

[4]. https://en.wikipedia.org/wiki/Mean_squared_error

[5]. https://in.mathworks.com/discovery/edge-detection.html

[6]. https://www.rroij.com/open-access/currency-recognition-using-image-processing.pdf

[7]. H. Hassanpour, A. Yaseri and G. Ardeshiri, "Feature extraction for paper currency recognition," *2007 9th International Symposium on Signal Processing and Its Applications*, Sharjah, 2007, pp. 1-4.

[8]. Er-Hu Zhang, Bo Jiang, Jing-Hong Duan and Zheng-Zhong Bian, "Research on paper currency recognition by neural networks," Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), 2003, pp. 2193-2197 Vol.4.

[9]. http://sebastianruder.com/optimizing-gradient-descent/

[10]. http://mccormickml.com/2014/03/04/gradient-descent-derivation/