International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Automatic Recognition of Handwritten Digits Using Multi-Layer Sigmoid Neural Network

Said Kassim Katungunya[1], Xuewen Ding[2], Juma Joram Mashenene[3]

[1, 2, 3]Tianjin University of Technology and Education, School of Electronics Engineering, Dagu Nanlu Tianjin, 300222, P.R China

Abstract: ***One of the challenges we face in human vision is recognizing handwritten digits, since digits writing by free hand may differ from one person to another, sometimes it can be difficult to identify exact type of digits due to their shape and style upon which the digit is written. Automatic recognition of handwritten digits using multi-layer sigmoid neural network provides a solution to this problem. This approach employs the use of sigmoid neuron with feed forward and back propagation technique. This tool can ensure accuracy of more than 98 percent in recognizing various handwritten digits.***

Keywords: logistic regression, sigmoid function, Feed forward, Back propagation, regularization parameter

1. Introduction

Handwritten digits plays important role in various firm such as reading payment checks, zip code, car plates, barcodes and so on. Also it plays important role to ensure no amount of figure is misread in critical systems like monetary systems where failure to recognize the correct digits could lead to huge loss of money and other inconveniences. So it becomes very much important to develop a tool that can identify handwritten digits accurately and efficiently. To ensure learning accuracy a data set containing vast amount of handwritten digits is prepared for training sigmoid neural network.

2. Dataset Preparation

The grayscale image of many digits is segmented into a sequence of separate images containing single digit.

Figure 1: shows a portion of random image extracted from dataset before segmentation

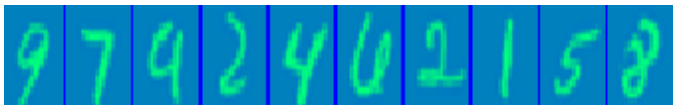

Figure 2: Shows a portion of same random image after segmentation

The grayscale intensity at a particular location of grayscale pixel in a dataset ranges between 0.0 and 1.0, where by 0.0 means white and 1.0 means black. Any value between 0.0 and 1.0 represent slightly darkening intensity. We use5000 training example from thousands of scanned images from MNIST data set.A matrix X is prepared by unrolling 28x28 grid of pixels into a 784-dimensional vector. Each of these training examples becomes a single row in a data matrix X.

If we let$\theta$to be the weight of sigmoid neural network then the product between vectors X and weight θ can be computed by using transpose matrix, $\theta^T x$.

Dimensional vector, (y) represents labels of the training set.

3. Multi-layer Sigmoid Neural Network Design for Digits Classification

To present a training data of 28x28 pixel image of digits, we design input layer of 784neurons. We chose a hidden layer of 28 neurons denoted by n and the output layer of 10 neurons. We build a multi-class classifier consisting of 10 classes to classify digits 0 through 9.The output neurons are arranged in such a way that outputs from each one of them corresponds to the value of the digit 0 through 9.

Biases are weights associated with vectors that lead from a single node whose location is outside of the main network and whose activation is always 1.The use of biases in neural network increases the capacity of the network to solve problems by allowing the hyperplanes that separate individual classes to be offset for superior positioning.

By taking x and y as the input and output vectors, our training example dimension will be 784 for x and 10 for y .Whereby for input vector x each entry represents grey scale value for a single pixel in the image. A careful selection of weight and biases is made to minimize the cost function given by gradient descent for better classification accuracy.

We are using regularized logistic regression with sigmoid function with feed forward and back propagation technique to train our neural network. Sigmoid function serves the purpose as activation function. Since the images are of size 28 x 28, this gives us 784 excluding bias unit. Dimensional weights are expressed as,

$\theta^{(1)}$ = 25 x 785 (1)

$\theta^{(2)}$=10 x 26 (2)

4. Feed Forward and Back Propagation

4.1. Feed forward

Feed forward simply means output from one layer is used as input to the next layer. In other words network has links that extend in only one direction, except during training. There

are no backward links in a feedforward network; all links proceed from input nodes toward output nodes.

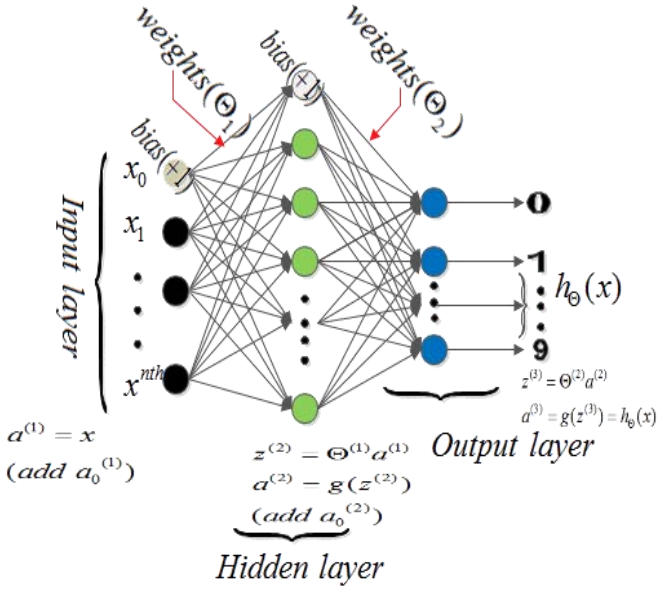

Figure 3: feedforward pass

Implementation of feedforward, Consider below equations.

$$a^{(1)} = x\ (add\ a_0^{(2)}) \quad (3)$$
$$z^{(2)} = \theta^{(1)} a^{(1)} \quad (4)$$
$$a^{(2)} = g(z^{(2)})\ (add\ a_0^{(2)}) \quad (5)$$
$$z^{(3)} = \theta^{(2)} a^{(2)} \quad (6)$$
$$a^{(3)} = g(z^{(3)}) = h_\theta(x) \quad (7)$$

Where, $a^{(0)}, a^{(1)}, a^{(2)}$ and $a^{(3)}$ are activations inputs. $h_\theta(x)$ Is the output of the 3rd layer neurons. $g(z^{(2)}), g(z^{(3)})$ are sigmoid activation functions

4.2 Backpropagation

Back Propagation (BP) is a method for training multilayer feedforward networks It works by training the output layer and then propagating the error calculated for these output neurons, back though the weights of the network, to train the neurons in the inner (hidden) layers, the backpropagation algorithm employs the Delta Rule for calculating error at output units, while error at neurons in the layer directly preceding the output layer is a function of the errors on all units that use its output. The effects of error in the output node(s) are propagated backward through the network after each training case. We use back propagation to calculate the gradient of the cost function with regularized term, then we train the neural network by minimizing the cost function $J(\theta)$.

Setting the weights based on training patterns and the desired output is the crucial problem. The backpropagation is one of the simplest and most general methods for supervised training of multilayer neural networks. If the "proper" outputs for a hidden unit were known for any pattern, the input-to-hidden weights could be adjusted to approximate it. Back-propagation allows us to calculate an effective error for each hidden unit, and thus derive a learning rule for the input-to-hidden weights. Networks have two primary modes of operation: feedforward and learning. Feed-forward operation consists of presenting a pattern to the input units and passing the signals through the network in order to yield outputs from the output units. Supervised learning consists of presenting an input pattern and changing the network parameters to bring the actual outputs closer to the desired teaching or target values.

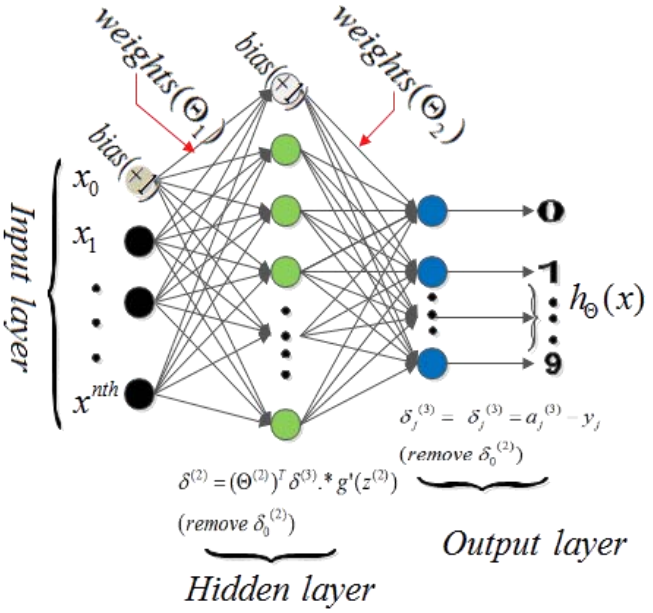

Figure 4: Backpropagation

First we run a "forward pass" to compute all the activations throughout the network, including the output value of the hypothesis $h_\theta(x)$. Then, for each node j in layer l, we compute an error term $\delta_j^{(l)}$ that measures how much that node was responsible for any errors in the output.

$\delta_j^{(3)}$ Is defined by measuring the difference between network activation and the true target.

$\delta_j^{(l)}$ Is computed for the hidden layer based on a weighted average of the error terms of the nodes in layer $(l+1)$.

Firstly $(z^{(2)}, a^{(2)}, z^{(3)}, a^{(3)})$ for layers 2 and 3. are computed, then for each output unit k in layer 3, weset $\delta_k^{(3)} = (a_k^{(3)} - y_k)$ Where $y_k \epsilon \{0,1\}$ indicates whether the current training example belongs to class k $(y_k = 1)$, or if it belongs to a different class $(y_k = 0)$, for the hidden layer $l = 2$, we

$$set \delta^2 = \left(\theta^{(2)}\right)^T \delta^3 .* g'(z^{(2)}) \quad (8)$$
$$g'(z) = \frac{d}{d_z} g(z) = g(z)(1 - g(z)) \quad (9)$$
$$\text{Where, sigmoid}(z) = g(z) = \frac{1}{1+e^{-z}} \quad (10)$$

z is a real number given by $\theta^T x$

A sigmoid function is a bounded differentiable real function that is defined for all real input values and has a positive derivative at each point.

A cumulative gradient is given by

$$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T \quad (11)$$

It follows that

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} \text{ for } j = 0 \quad (12)$$
$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} + \frac{\lambda}{m} \theta_{ij}^{(l)} \text{ for } j \geq 1 \quad (13)$$

Where,

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k\right] + \frac{\lambda}{2m}\left[\sum_{j=1}^{28}\sum_{k=1}^{784}(\theta_{j,k}^{(1)})^2 + \sum_{j=1}^{10}\sum_{k=1}^{28}(\theta_{j,k}^{(2)})^2\right] \quad (14)$$

Where, m is the number of training samples

5. Learning Curves

An under-performing algorithm can be due to two possible situations: high bias (under-fitting) and high variance (over-fitting). In order to evaluate our algorithm, we set aside a portion of our training data for cross-validation. Using the technique of learning curves, we can train on progressively larger subsets of the data, evaluating the training error and cross-validation error to determine whether our algorithm has high variance or high bias.

By observing a learning curves a high bias can be identified if regularized logistic regression to predict handwritten digits, results in to unacceptably large errors in its predictions on new set of features and performs poorly on the training set. The following measure has to be taken to mitigate high bias problem. If algorithm suffers from high bias we have to add more features, use fewer training samples and decrease regularization parameter (λ). Regularization add a penalty term that depends on the characteristics of the parameters. If a model has high bias, decreasing the effect of regularization can lead to better results. A high variance can be identified when prediction algorithm makes large errors a new set of images but has low error on the training set. To correct high variance we have to use a smaller set of features, add more training samples and increasing the regularization parameter (lambda λ).

6. Random initialization of weights

Although the ideal initial values for weights (i.e., those that will maximize the effectiveness and speed with which a neural network learns) cannot yet be determined theoretically it is general practice to assign randomly-generated positive and negative quantities as the initial weight values. Such a random distribution can help minimize the chances of the network becoming stuck in local minima. Typically, values are selected from a range $-\epsilon_{init}, \epsilon_{init}$ where 0.1 < a <2. The reason for using random initial weights is to break symmetry, while the reason for using small initial weights is to avoid immediate saturation of the activation function. One effective strategy for random initialization is to randomly select values for $\theta^{(l)}$ uniformly in the range $[-\epsilon_{init}, \epsilon_{init}]$. We chose $\epsilon_{init} = 0.12^2$. This range of values ensures that the parameters are kept small and makes the learning more efficient. Then initialize the weights for θ, one effective strategy for choosing ϵ_{init} is to base it on the number of units in the network. A good choice of ϵ_{init} is computed as depicted below.

$$\epsilon_{init} = \frac{\sqrt{6}}{\sqrt{L_{in}+L_{out}}} \quad (15)$$

Whereby $L_{in} and L_{out}$ are the number of units in the layers adjacent to $\theta^{(l)}$

We achieved a higher training accuracy by conducting more and more iterations.

7. Representations at the Hidden Layer-Weights

In addition to visualizing on the transformation of patterns in a network, we can also consider the representation of learned weights. Because the hidden-to-output weighs merely leads to a linear discriminant, it is instead the input-to-hidden weights that are most instructive. In particular, such weights at a single hidden unit describe the input pattern that leads to maximum activation of that hidden unit. It is convenient to think of the hidden units as finding feature groupings useful for the linear classifier implemented by the hidden-to-output layer weights.

8. Experiment and Results

This system is implemented using Matlab 2015a.First we have to check whether our algorithm is under-fitting or over-fitting the data by plotting cost on cross-validation set versus number of training set.

We chose different value of regularization parameter to determine the one which gives best accurate results and then we plot a graph of cost against regularization parameter (lamda). Regularized logistic regression algorithm is run for 400 iterations to minimize cost function. Results shows that as the number of iterations increase the computed cost values keeps on decreasing. Finally a prediction algorithm is executed to identify digits. Also the results shows that the confidence of neural network is for each digit prediction is more than 90 percent.

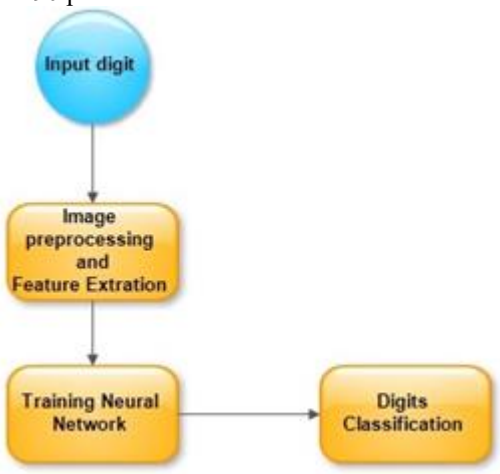

Figure 5: flow chart of the algorithm

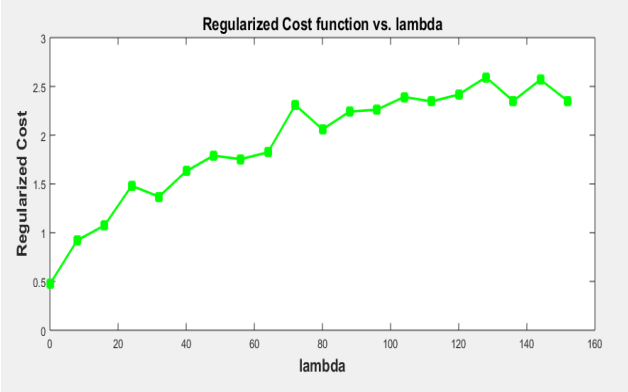

Figure 6: shows regularized cost increases with the value of regularization parameter (lambda, λ).

Figure 7: learning curve

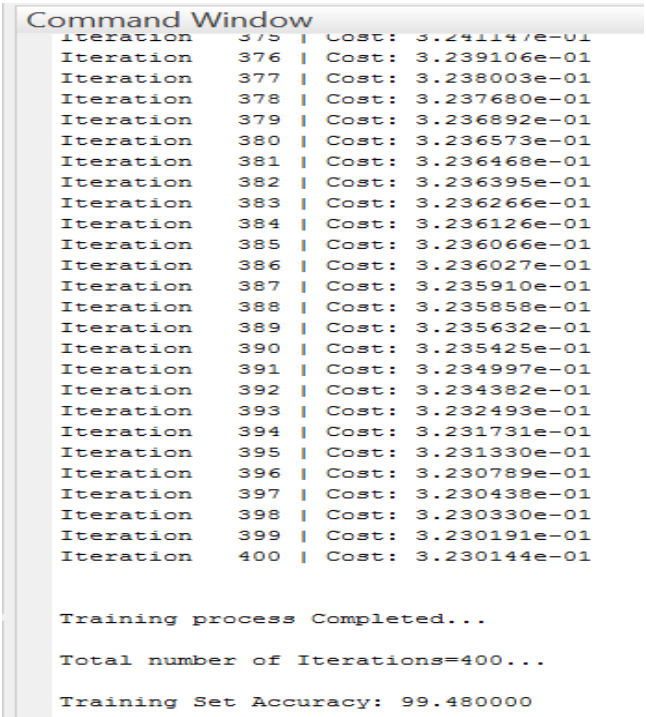

Figure 8: Training neural network for 400 iterations

Figure 9 and 10 display few examples of the digits recognized by this method

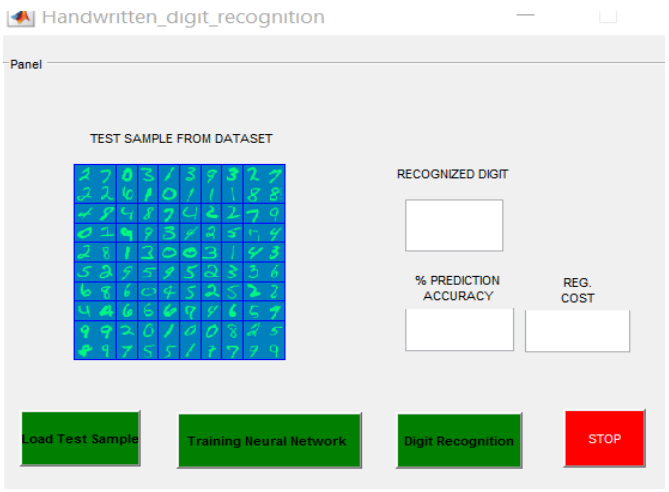

Figure 9: loading random sample of digits from the dataset

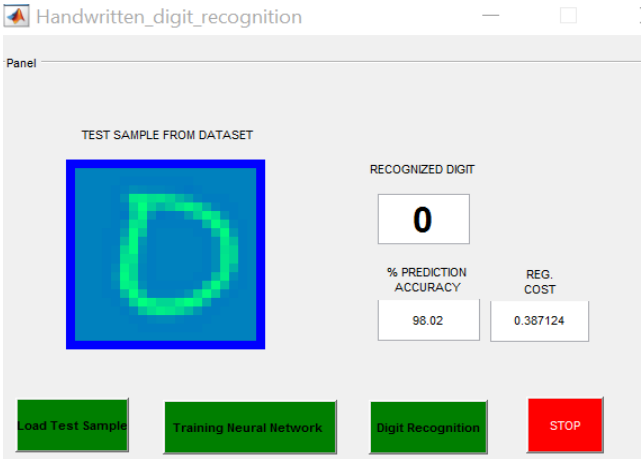

Figure 9: 0 digit is recognized

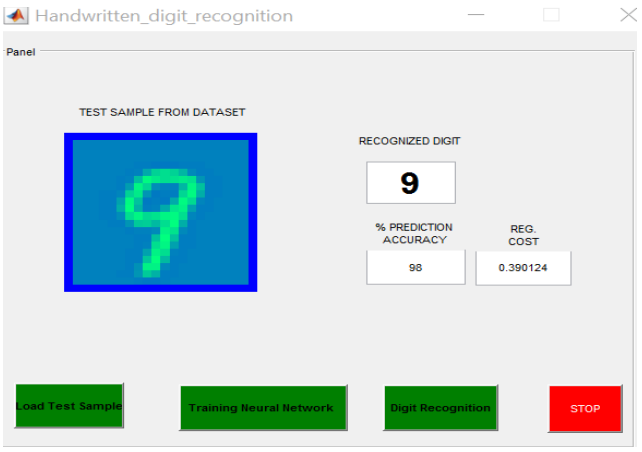

Figure 10: 9 digit is recognized

Table 1: Prediction accuracy for each digit

Digits	Average Prediction error	Average accuracy
0	0.	98.01
1	0	99.00
2	0	100
3	0	99.04
4	0	97.07
5	0	98.06
6	0	99.80
7	0	100
8	0	99.25
9	0	99.09

9. Conclusion

This paper set a method for handwritten digits recognition using multi-layer sigmoid neural network trained by thousands of image of handwritten digits from training set. This approach is fast in computation and highly accurate compared to other methods. Based on this method further research can be done on this method to extend the capability of neural network to tackle most complex problems of machine learning.

References

[1] Simon Hykin, Neural network and learning machines, third editionISBN:9787111265283
[2] Rafael Gonzalez, C., E. Richard Woods and L. Steven Eddins, 2010. Digital Image Processing using MATLAB. 3rdEdition. Prentice Hall, USA. ISBN: 9787121102073
[3] Andrew Yan-Tak Ng. Coursera Machine learning, Jan 25, 2016 - Apr 17, 2016 Stanford University.
[4] Gallant, S.I., 1993. Neural Network Learning and Expert Systems. MIT Press, Cambridge.
[5] Hansen, L.K., and Salamon, P., 1990. Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12: 993-1001.
[6] Holmström, L., and Koistinen, P., 1992. Using additive noise in back-propagation training, IEEE Transactions on Neural Networks, 3: 24-38.
[7] An Evaluation of Satellite Imagery and Classification Algorithms, PhD Thesis, U.Manitoba, Winnipeg,
[8] Rich, E., and Knight, K., 1991. Artificial Intelligence. McGraw-Hill, New York.
[9] Han, Jun; Morag, Claudio (1995). "The influence of the sigmoid function parameters on the speed of backpropagation learning". In Mira, José; Sandoval, Francisco. *From Natural to Artificial Neural Computation*. pp. 195–201.
[10] Yann LeCun, Patrick Haffner, Leon Bottou, and Yoshua Bengion. Object Recognition with Gradient-Based Learning
[11] Tzanakou, E.M., 2000. Classifiers: An Overview, from Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence. CRC Press LLC., ISBN: 0-8493-2278-2, pp: 371.
[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998
[13] O. Matan, R.K Kiang, C.E Sternard, B. Boser, J.S Denker, D. Handerson, R.E. Howard, W.Hubbard, L.D Jackel and Y.Le Cun, "Handwritten Character Recognition Using Neural Network Architecture
[14] Jyrki Selinummi, Jenni Seppälä, Olli Yli-Harja, and Jaakko A. Puhakka „Software for quantification of labeled bacteria from digital microscope images by automated image analysis Tampere University of Technology, Tampere, FinlandBio Techniques 39:859-863 (December 2005) doi 10.2144/000112018

Author Profile

Said Kassim Katungunya, was born in Tanzania. He received B.E degree in Electrical and Electronics Engineering from Acharya Institute of Technology, India in2009. Currently he Pursue Master"s degree in Signals and Information Processing at Tianjin University of Technology and Education, China. Also he hold CISCO certifications in CCNA security and CCNP routing and switching. From 2010 to 2013, He worked as a Network Engineer before joined Tianjin University of Technology and Education. His research interests include machine learning, artificial intelligence, pattern recognition and biomedical image processing.

Xuewen Ding is an Associate Professor of Electrical Engineering School at Tianjin University of Technology and Education, and Director of Electrical Engineering Department. He received an M.S. in Signal and Information Processing from Tianjin University in 2003, and a PhD. in Signal and Information Processing from the same university in 2008. He worked as a visiting scholar at Oakland University in 2015. Dr. Ding is very actively involved in Intelligent Information Processing. His research areas include Computer vision, Machine learning and Real Time Systems

Juma Joram Mashenene is a Tutor at University of Dar Es Salaam Computing Centre. He received his B.E in Computer Science and Engineering from St. Joseph University in Tanzania in 2012 and current pursuing M.E in Signal and Information Processing from Tianjin University of Technology and Education. He is interested in Computer Vision, Embedded Systems and Cognitive Radio Networks as research Areas.