# An Edge Detection Method for Grayscale Images based on BP Feedforward Neural Network

Jesal Vasavada
M.Tech (CSE)
MITS, Laxmangarh

Shamik Tiwari
Assistant Professor
MITS, Laxmangarh

## ABSTRACT

The edges provide important visual information since they correspond to major physical and geometrical variations in scene object. Edge detection is a terminology in image processing that refers to algorithms which aim at identifying edges in an image. In this paper a Feedforward Neural Network (FNN) based algorithm is proposed to detect edges in gray scale images. The backpropagation learning algorithm is used to minimize the error. Standard deviation and gradient values are used as training patterns. In the end the network is tested for a number of different kinds of grayscale images. The proposed scheme is compared with Prewitt, Roberts, Sobel, LoG and other neural network based method in which binary training patterns are used. Our method has performed significantly better as compared to other methods.

## Keywords

Edge Detection, Neural Networks, MATLAB, Backpropagation.

## 1. INTRODUCTION

Edges in images are simply the boundary between the object and the background. Edges provide important information of objects such as shape, area, perimeter etc [1]. Edges are often used to distinguish one object from the other and/or separate them from the background. In image processing, edge detection is used to filter out less relevant information and at the same time preserves the basic structural properties of an image. It significantly reduces the amount of data to be processed in the subsequent steps such as feature extraction, image segmentation. Edge detection is the basic or fundamental steps whose results are used in many applications like pattern recognition, image analysis, object recognition etc.

Since edges occur when there is sudden change in the intensity values of an image thus edge detection is the process which is used to identify and locate such sharp intensity contrasts (i.e., discontinuities) in an image. Small changes in intensity values correspond to small values of derivatives while large change in intensity values correspond to large values of derivatives. Based on this principle, a two-dimensional spatial filter called as “the gradient operator” is often used in conventional edge detection algorithms [2]. This filter yields no response to non-edge (uniform) regions in the image. Also variety of filters (or "masks") have been developed to detect various types of edge in horizontal, vertical, or diagonal directions, respectively. Once the mask is constructed, it is convolved with the entire image, pixel by pixel, to detect edges. The typical conventional edge detectors are the Roberts edge detector, the Prewitt edge detector, the Sobel detector, the Laplacian of Gaussian (LoG) detector etc [3]. The limitation of the above traditional algorithms is that multiple threshold values need to be set through a trial-and-error process; and these values can dramatically affect the performance of each algorithm. Also Traditional techniques are sensitive to noise and lead to false edge detections in case of noisy images.

Recently, artificial neural networks (ANN) [4] have been applied to edge detection. An artificial neural network is computational model that is inspired by the structure and /or functional aspects of “biological neurons of brain” .So ANN consists of an interconnected group of artificial neurons and has a natural property for storing experimental knowledge and making it available for use in order to process information.

Artificial neural network can be used as a very prevalent technology, neural network technology is better than classical edge detectors in various perspectives. First it provides less operation load and has more advantageous for reducing the effect of the noise in images. Second is its adaptive learning ability. If a neural network is trained to detect edges in grayscale images having uniform contrast, it can be easily retrained to deal images having non-uniform contrast with minor changes in lightening conditions as neural network has ability to change its synaptic weights in real time. The third advantage is its generalization ability. Generalization refers to the neural network’s ability to detect edges for those images also that are not encountered during training (learning) phase. Fourth is its non linear mapping ability. An artificial neuron can be linear or non linear. Non–linearity is highly important property, particularly if the underlying physical mechanism responsible for generation of the input image is inherently non-linear. Fifth, its parallel organization permits solutions to problems where multiple constraints must be satisfied simultaneously. Due to this property artificial neural network is more useful, because multiple inputs and multiple outputs can be used during the stage of training. For example in traditional methods one pixel is processed at a time, but in neural networks multiple pixels as inputs can be given. In paper [5], nine pixels of a 3x3 window of an image as inputs are given simultaneously. Also artificial neural networks are fault tolerant in nature as its performance degrades gracefully under adverse conditions.

When using neural networks for edge detection most researchers have proposed to use binary images or convert the grayscale image into binary image. On the other hand , a few researchers proposed to use binary patterns as training data set for edge detection in grayscale images. In this paper feedforward neural network based method has been proposed to detect edges in grayscale images using standard deviation and gradient values as training patterns.

The paper is organized as follows. Section 2 gives the related work done in this area Section 3 explains backpropagation learning algorithm. Section 4 describes the proposed model. Results and analysis are reported in section 5. Finally the section 6 is concluding the paper.

## 2. RELATED WORKS

The edges provide important visual information since they correspond to major physical, photometrical or geometrical variations in scene object. Edge detection is a terminology in image processing that refers to algorithms which aim at identifying edges in an image. Many works have been done to detect the edges using neural networks. Li and Wang in [6] proposed a parallel model of Back-Propagation (BP) neural network to detect tile defect. Firstly, the BP neural network for edge detection of binary image was designed. Secondly, the gray image was divided into 8 binary images according to the bit plane. Thirdly, a parallel model of BP neural network, which was composed by 8 sub BP neural network for edge detection of binary image, was constructed. Fourthly, the sub BP neural network was used for edge detection of every binary image. Finally, the output of every sub BP neural network was adjusted according the weight of every bit plane, and the accurate edge of gray image will be obtained. Terry and Vu in [7] investigated the application of multi-layer feedforward neural networks for the edge detection of the LADAR (laser radar) image of a bridge. Multiple neural networks are trained by synthetic edge patterns; each one of them can detect a specific edge pattern (e.g., horizontal, vertical, diagonal, etc.). If desired, one can also combine the outputs from the "group" of neural networks to detect multiple types of edges in images. Lu et al. in [5] proposed the ANN model which has one input layer, one output layer, and one hidden layer. There are 9 neurons in the input layer; in other words, the input of this network is a 9×1 vector which is converted from a 3×3 mask. There are 10 hidden neurons in the hidden layer; and one neuron in the output layer which indicates where an edge is detected. The neural network is fully trained by the 17 binary patterns. Yasar Becerikli and H. Engin Demiray in [8] proposed a method in which each image was broken into 3x3 over-lapping grids. The network was set up with 9 input. Again the binary patterns are used for training. nodes, one hidden layer with 12 nodes, and 1 output node.

## 3. NEURAL NETWORKS

### 3.1 FeedForward Neural Networks

A feedforward neural network [4] is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing *units*, organized in *layers*. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, each connection may have a different strength or *weight*. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called *nodes*. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called *feedforward* neural networks.

It is classified into single layer feedforward or multi-layer feed-forward neural network. In single layer feedforward neural network, we have an input layer of source nodes that projects into output layer of neurons. This network is strictly feed forward or acyclic. “single layer” referring to output layer of computational nodes(neurons).We do not count input layer of source nodes because no computation is performed here. In our algorithm we have used multi-layer feed-forward neural network. Multi-layer feed forward neural network distinguishes itself by the presence of one or more “hidden layers” whose computation nodes are called hidden neurons. Hidden layer has ability to extract higher order statistics particularly when size of input layer is large. The source node in the input layer constitute the input signals applied to neurons in the second layer( first hidden layer).The output signal of second layer are used as inputs to 3rd layer and so on for the rest of the network. Fig.1 shows an example of a 2-layered network: an *output* layer with 2 units, a *hidden* layer with 5 units, respectively. The input layer has 3 units.

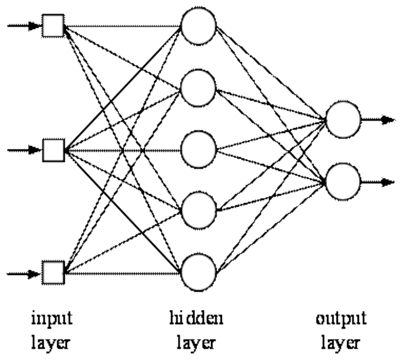


**Fig.1: 2 layer Feedforward neural network.**

### 3.2 Backpropagation Learning

During learning, the error function decreases and the weights are updated. The decrease may be accomplished with different optimization techniques such as the Delta rule, Boltzmann’s algorithm, the backpropagation learning algorithm and simulation annealing.

An edge-detection neural network can be trained with backpropagation using relatively few training patterns. The most difficult part of any neural network training problem is defining the proper training set. Backpropagation [9] is the most commonly used method for training the artificial neural network to minimize the gradient as shown in Fig.2. The steps in Backpropagation algorithm are as follows:

#### *3.2.1 Randomly initialize the weight.*

#### *3.2.2 Feed the training sample.*

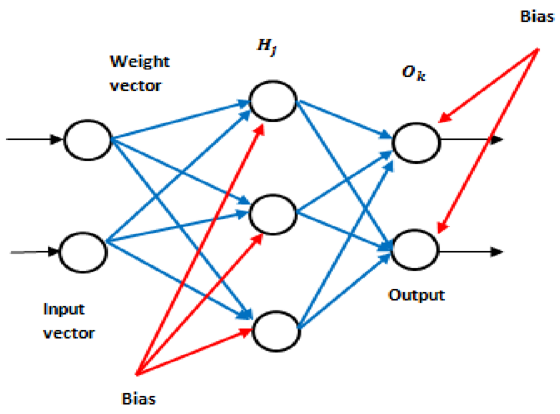


**Fig.2: Error propagation through hidden layer.**

#### *3.2.3 Propagate the inputs forward*

This step computes the net input and output of each unit in the hidden and output layers as follows:-
Each hidden unit (Hj) sums its weighted input signals.

$$H_{inj} = \sum_i w_{ij}\, x_i + w_{oj}$$

Where, $w_{ij}$ is the weight between hidden and input layer; $x_i$ is the input training vector; and $w_{oj}$ is the bias of the unit. Applying Activation function:

$$H_j = f(H_{inj})$$

And sigmoid function is calculated as:

$$f(H_{inj}) = \frac{2}{1 + e^{-2H_{inj}}} - 1$$

And this is provided as input for the output layer.

Each output unit ($O_k$, k=1….m) sums its weighted input signals.

$$O_{ink} = \sum_j w_{jk} H_j + w_{ok}$$

And output signal after applying activation function:

$$O_k = f(O_{ink})$$

### 3.2.4 Back propagate the error to the hidden layer

When reaching the Output layer, the error is computed and propagated backwards to hidden.

For a unit k in the output layer the error is computed by a formula:

$$\delta_k = (D_k - O_k) f(O_{ink})$$

$\delta_k$ is error at output unit $O_k$

Each hidden unit Hj sums its delta input from above layer inputs.

$$\delta_{inj} = \sum_k \delta_k w_{jk}$$

The error is calculated as

$$\delta_j = (\partial_{inj}) f(H_{inj})$$

$\delta_j$ is Error at hidden unit Hj

### 3.2.5 Update weights

Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases. Weights are updated by the following equations

$$\Delta w_{ij} = \alpha \delta_j x_i$$
$$\Delta w_{jk} = \alpha \delta_k H_j$$

Where, α is learning rate

Biases are updated by the following equations:

$$\Delta w_{oj} = \alpha \delta_j$$
$$\Delta w_{ok} = \alpha \delta_k$$

So new weights are:

$$w_{ij(new)} = w_{ij(old)} + \Delta w_{ij}$$
$$w_{jk(new)} = w_{jk(old)} + \Delta w_{jk}$$

And new biases are:

$$w_{oj(new)} = w_{oj(old)} + \Delta w_{oj}$$
$$w_{ok(new)} = w_{ok(old)} + \Delta w_{ok}$$

### 3.2.6 Terminating condition

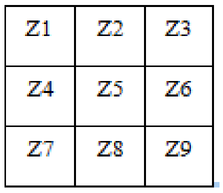
Networks run until at least one of the following Termination conditions were satisfied: Maximum Epoch as specified is reached or Given Minimum Gradient reached.

## 4. PROPOSED METHOD

The proposed algorithm is implemented in MATLAB 7.10.0. The proposed method has seven steps.

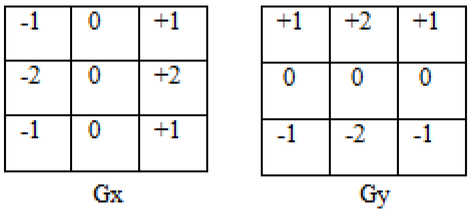
### 4.1 Calculation of input values

Calculate the gradient and standard deviation values of the image to be processed. The gradient values are calculated using the Sobel operator. Sobel operator looks for edges in horizontal and vertical directions then combine the information into a single metric. To calculate the gradient the image is divided into 3x3 windows as shown in Fig.3 where Z are the intensity values. The gradient in horizontal direction is calculated using the mask Gx and for vertical direction mask Gy as given in Fig.4 is used.



| | | |
|---|---|---|
| Z1 | Z2 | Z3 |
| Z4 | Z5 | Z6 |
| Z7 | Z8 | Z9 |

**Fig.3: 3x3 window (neighborhood).**



Gx

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

Gy

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

**Fig.4: Sobel convolution masks.**

Finally the edge magnitude is calculated using the equation given as:

$$G = (Gx^2 + Gy^2)^{1/2}$$

$$G = \{[(Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)]^2 + [(Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)]^2\}^{1/2}$$

The standard deviation of the pixels of a 3X3 neighborhood is computed as using the equation as given below:

$$\text{var}\{s\} = \text{sqrt}(\frac{\text{sum}(\text{var}\{x\} - \text{var}\{mu\}).^2}{\text{var}\{N\} - 1})$$

where var{mu} is the mean value of the pixels in the neighborhood, var{N} is the number of pixels in the neighborhood. Elements of the mask with a non-zero value are considered part of the neighborhood.

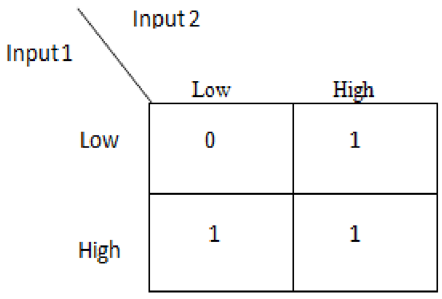
Normalize both the values in the range 0 to 1 using the equation:

$$n(x) = \frac{x}{\max(X(:))}$$

Where $n(x)$ is the normalized value of the pixel x and X is the image in matrix form.

### *4.2 Training data*

120 patterns are taken as training set out of which 25 are non-edge patterns and 96 are edge patterns. The pattern consists of two inputs. These values/inputs can range from 0 to 1 in the interval of 0.1. The training patterns and the desired output are as shown in Fig.5.



| Input1 \ Input2 | Low | High |
|---|---|---|
| Low | 0 | 1 |
| High | 1 | 1 |

**Fig.5: Training patterns and desired output.**

Thus when both the inputs are low, the desired output is low else the desired output will be 1. We have considered all the values below 0.5 as low.

### *4.3 Testing data*

The standard deviation and gradient values calculated using step 1 of the images to be tested are given as inputs to the network.

### *4.4 Network architecture*

Network has one input Layer, one hidden layer and one output layer. So it’s a 2 layer FeedForward network. Here 2 neurons at input layer corresponding to standard deviation and gradient values, 3 neurons at the hidden layer and 1 neuron at output layer. Tan-sigmoid transfer function is used. Hyperbolic tangent transfer function (TANSIG, Fig.6) in the term of neural networks, is related to a bipolar sigmoid which has an output in the range of -1 to +1.

As it can be seen that it is mathematically equivalent to tanh (n). It differs in that it runs faster than tanh, but the results can have very small numerical differences. Tansig is a neural transfer function which calculate a layer's output from its net input. This function is a good tradeoff for neural networks, where speed is more important than the exact shape of the transfer function.

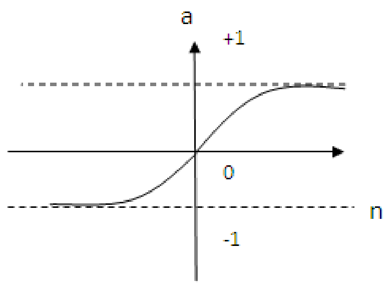


**Fig.6: Hyperbolic tangent transfer function**

$$a = \mathrm{Tansig}(n) = \frac{2}{1 + e^{-2n}} - 1$$

### *4.5 Parameter Adjustment and Weight Initialization*

The weights between the input and the hidden layer and between hidden and the output layer are initialized randomly. Learning rate is 0.5 initially. The network is trained for 1000 epochs.

### *4.6 Training*

The network training is done using backpropagation learning algorithm which minimizes the error and update the weights during learning until the calculated outputs are within the margin of the known outputs.

### *4.7 Testing*

Now after successful training of network, the network is tested for a number of different kind of images and the desired output for any kind of grayscale image is obtained. To obtain the best and accurate result threshold of 0.5 is applied during testing.

## 5. RESULTS AND COMPARISONS

As stated before our algorithm can detect all the edges of any kind of grayscale image, and to prove that we have tested our algorithm on all possible kinds of grayscale images like natural human image (Lena.jpg), beans image (beans.jpg) and flower image (flower.jpg). In this section comparison is done between the proposed method discussed above with the Roberts, Prewitt, Sobel, LoG and other neural network based method in which the training patterns are binary. In Fig.7 the comparison between all operators is done on visual perception for all three images. Study of the edge maps of all the edge detectors is done. For each edge map the number of edge pixels is count and compared. In Fig.8 comparison is done on the basis of edge pixel count in all operators for three different images. It can be seen from the figure that our method detects the highest edge pixels. Our method performs well in case of noisy images also. To prove this, two types of noise ‘salt and pepper’ and speckle noise are added to flower.jpg with different noise levels from 30db to 60db using the equations as given below:

$$\mathrm{MSE} = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N}$$

To compute PSNR value, first MSE (mean square error) is computed. In the previous equation, M and N are the number of rows and columns in the input images, respectively. Then the block computes the PSNR using the following equation:

$$\mathrm{PSNR} = 10\log_{10}\left[\frac{R^2}{\mathrm{MSE}}\right]$$

In the previous equation, *R* is the maximum fluctuation in the input image data type. For an 8-bit unsigned integer data type, *R* is 255. The MSE represents the cumulative squared error between the output and the original image, while PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

Then they are filtered with median filter for salt and pepper noise and with lee filter for speckle noise. We can see from Fig.9 and Fig.10 our method produces the best results and detect highest edge pixels in both cases. The proposed method method reports the higher edge pixels as shown in Fig.7 to Fig.10.

Sobel and Prewitt are sensitive to noise as in case of noisy images they detect very less edge pixels. LoG sometimes produces double edges. The neural network based algorithm that uses binary patterns also distort the images sometimes as it can be seen in Fig.7, the Lena image is distorted. So our method produces the best output in all cases.

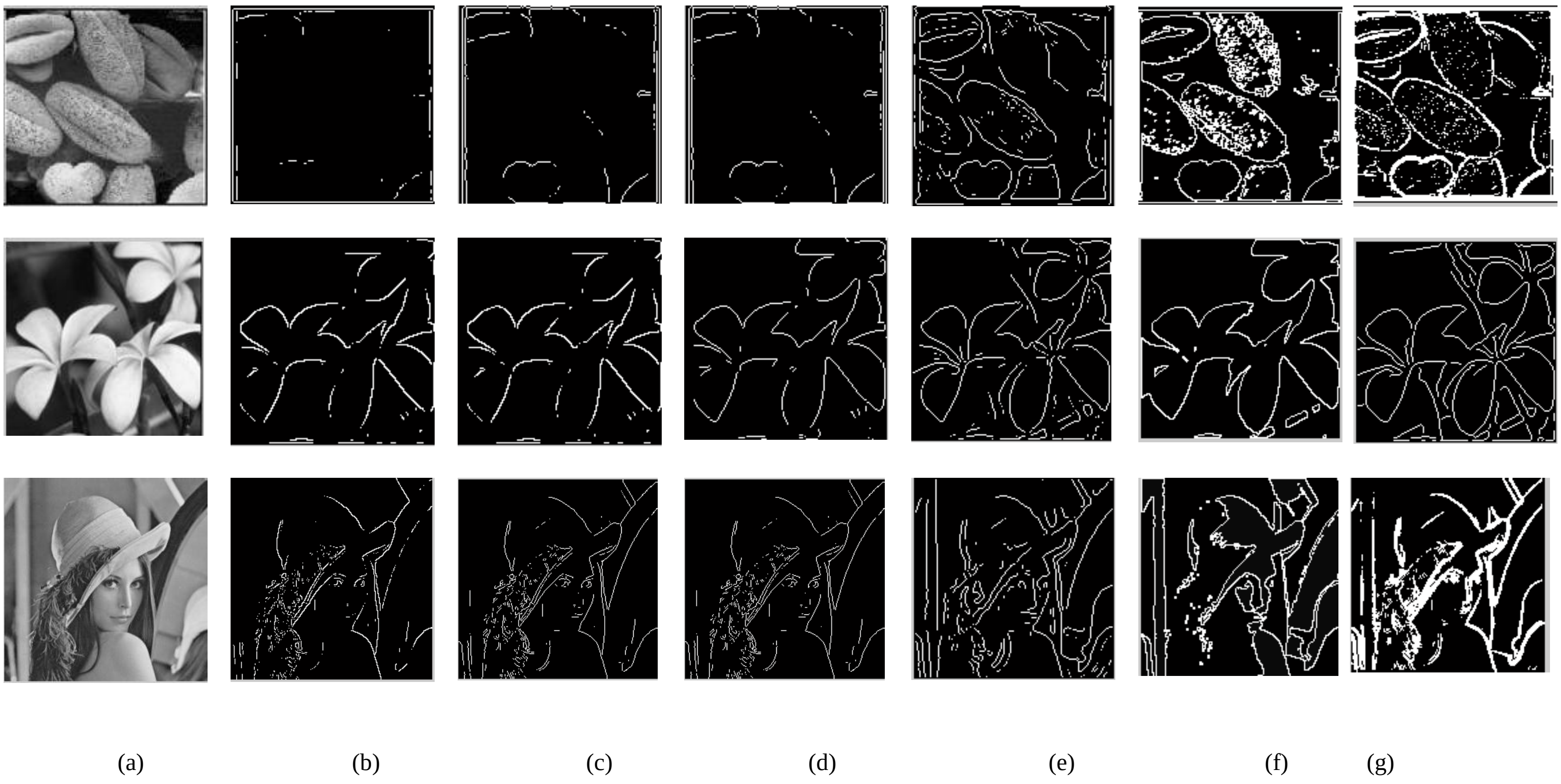


**Fig. 7: a) Main image, b) Roberts results, c) Prewitt results, d) Sobel results, e) LoG results, f) Binary training patterns using neural networks and g) proposed method using grayscale pattern.**

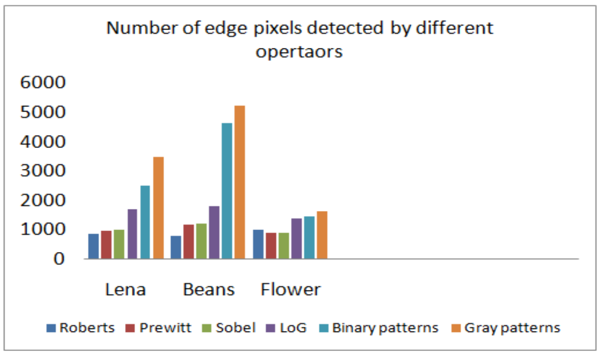


**Fig.8: Number of edge pixels detected by different operators for different images.**

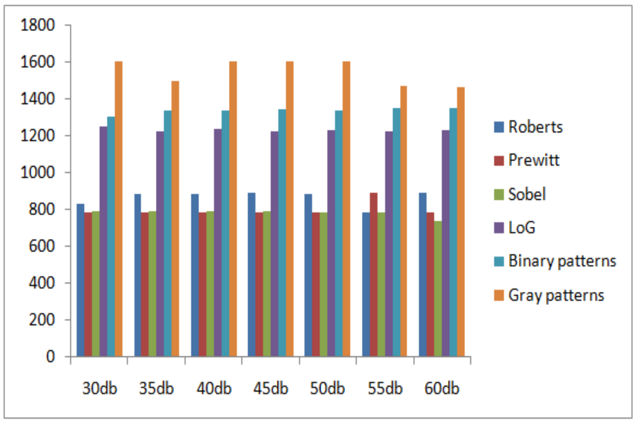


**Fig.9: Number of edge pixels detected by different operators for flower image after filtering speckle noise from 30 to 60 db.**

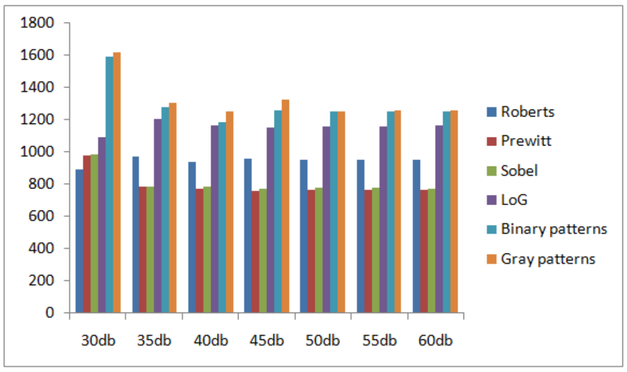


**Fig.10: Number of edge pixels detected by different operators for flower image after filtering salt and pepper noise from 30 to 60db.**

## 6. CONCLUSION

A feedforward neural network based adaptive system for detection of edges in grayscale images using standard deviation and gradient values as training data sets is presented in this paper. Backpropagation learning method is used. The proposed method is compared with traditional edge detectors as well as with the other neural network based methods that uses binary training patterns. On the basis of visual perception and edge pixels counts, the experimental results show that our algorithm is able to detect highest edge pixels in noise free images as well as in case of noisy images and perform robustly for all kind of images encountered in real world applications as it does not distort the shape and also retains the important features.

## 7. REFERENCES

[1] Ramani Maini and Dr. Himanshu Aggarwal, *Study and Comparison of Various Image Edge Detection Techniques*, International Journal of Image Processing (IJIP), Vol. 3, Issue 1,2011.

[2] S.Lakshmi and Dr. V. Sankaranarayan, *Study of Edge Detection Techniques Segmentation Computing Approaches*, IJCA Special Issue on Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications, CASCT, 2010.

[3] N. Senthilkumaran and R. Rajesh, *Edge Detection Techniques for Image Segmentation- A Survey*, International Conference on Managing Next Generation Software Applications, pp.749-760, 2008.

[4] Simon Haykin, *Neural Networks and Learning Machines*, 3rd edition.

[5] Yasar Becerikli and H. Engin Demiray, *Alternative Neural Network Based Edge Detection*, Neural Information Processing - Letters and Reviews Vol. 10, Nos. 8-9, Aug.-Sept. 2006.

[6] Li, W., Wang, C., Wang, Q., Chen, G., *An Edge Detection Method Based on Optimized BP Neural Network*, Proceedings of the International Symposium on Information Science and Engineering, pp. 40-44, Dec 2008.

[7] Jesal Vasavada and Shamik Tiwari, *Sobel-Fuzzy Technique to Enhance the Detection of Edges in Grayscale Images Using Auto-Thresholding*, International Conference on Soft Computing for Problem Solving, paper-268, Dec 2012.

[8] Terry, P., Vu, D., *Edge Detection Using Neural Networks*, Conference on Signals, Systems and Computers, Vol. 1, pp.391-395, 1993.

[9] Dingran Lu, Xiao-Hua Yu, Xiaomin Jin, Bin Li, Quan Chen, Jianhua Zhu, *Neural Network Based Edge Detection for Automated Medical Diagnosis*, Proceeding of the IEEE International Conference on Information and Automation, pp.343-348.

[10] Satish kumar, *Neural Networks -A Classroom Approach*, 2nd edition.