ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

# NOISE EFFECT ON ARTIFICAL NEURAL NETWORK BASED IMAGE COMPRESSION

**Mr. Gade M.R.*, Asst. Proff. Deshpande A.S**
Electronics and Telecommunication Department, ICOER, Wagholi,Pune, India

## ABSTRACT

This paper presents a neural networks as image processing tools for image compression, present a direct solution method based neural network for image compression. Digital images require large amounts of memory for storage. Thus, the transmission of an image from one computer to another can be very time consuming. By using data compression techniques, it is possible to remove some of the redundant information contained in images, requiring less storage space and less time to transmit. To observe effects of noisy image on decompressed image different type of noise are added in image and their result are compared.

**KEYWORDS**:Image compression, Artificial neural networks.

## INTRODUCTION

Modern application requires high volume of data to accommodate large number of images. Image compression represents the process of data reduction and at the same time retains image information. The key components to compress the data are specified channel bandwidths or storage requirements and maintaining the highest possible quality. So efficient data compression techniques save storage space and accelerate the transmission time. Compression of digital images has been the focus of much recent research interest in today's rapidly changing world. It has basically two inherent benefits. First, it makes possible to use digital images in transmission and storage environment that would not support uncompressed raw images. For example, current Internet throughput rates are insufficient to handle uncompressed image in real time (even at small image size). Second, compression enables more efficient use of transmission and storage resources. If a high bit rate transmission channel is available, then it is more attractive proposition to send high resolution compressed image or multiple compressed images than to send a single, low resolution, uncompressed image. The requirement of enormous expenditure on bandwidth and storage necessitates the use of various compression schemeseven with constant advances in storage and transmission capacity [1]. Artificial neural networks are popular and found its application in many fields, mainly in function approximation, due to their ability to approximate complicated nonlinear functions. There are mainly three types of learning used in different application i.e., supervised, unsupervised and reinforcement learning. The multilayer perceptron (MLP) along with the backpropagation (BP) learning algorithm can be categorized as a supervised neural network and most frequently used neural network in practical situations.

## NEED OF IMAGE COMRESSION

Image compression means reducing or eliminating of redundant or irrelevant information and is one of the most used techniques in the field of image processing.A digital movie (video) is a sequence of video frames, which are full-color digital still images.A two-hour Standard Definition (SD) TV movie that we need to store digitally. The standard rate for an NTSC video is close to 30 fps (frames per second), therefore, the bit rate required for SD video is30× (720 × 480) ×3 = 31,104,000 bytes / sec.Where 30 is Frame rate in(720 × 480) image resolution and 3 pixel depth. A storage requirement for a two-hour movie would be $31,104,000\times60\times60\times2 \approx 2.24\times10^{11}$bytes.Approximately 224 GB is needed to store an uncompressed two-hour SD movie. Therefore, to fit such movie on a standard DVD-9, data must be compressed by a factor of approximately 26.3. TheCompression must be higher for High Definition (HD) TV, where image resolution is up to 1920×1080 pixels. Similarly, 8-MP digital camera could store about 41 uncompressed full-color images on a 1 GB memory card. Compression can also significantly reduce

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

transmission time needed to transmit an image over the web[2].

## PROPOSED METHODOLOGY

L-M ALGORITHM

In order to make sure that the approximated Hessian matrix $J^TJ$ is invertible, Levenberg–Marquardt algorithm introduces another approximation to Hessian matrix:

$H \approx J^TJ + \mu I$ .....(1)

Where$\mu$ is always positive, called combination coefficient I is the identity matrix From Equation (1) one may notice that the elements on the main diagonal of the approximated Hessian matrix will be larger than zero. Therefore, with this approximation (Equation 1), it can be sure that matrix *H* is always invertible. By combining the equation 1 and update rule of the Gauss–Newton algorithm update rule of Levenberg–Marquardt algorithm can be presented as

$$w_{k+1} = w_k - \left(J_k^T J_K + \mu I\right)^{-1} J_K e_k$$

Table summarizes the update rules for various algorithms.

***Table 3.1 Specifications of Different Algorithms***

| Algorithms | Update Rules | Convergence | Computation Complexity |
|---|---|---|---|
| **EBP algorithm** | $w_{k+1} = w_k - \alpha g_k$ | Stable, slow | Gradient |
| **Newton algorithm** | $w_{k+1} = w_k - H_k^{-1} g_k$ | Unstable, fast | Gradient and Hessian |
| **Gauss–Newton algorithm** | $w_{k+1} = w_k - \left(J_k^T J_K\right)^{-1} J_K e_k$ | Unstable, fast | Jacobian |
| **Levenberg–Marquardt algorithm** | $w_{k+1} = w_k - \left(J_k^T J_K + \mu I\right)^{-1} J_K e_k$ | Unstable, fast | Jacobian |

## ALGORITHM STEPS

Step1: Read the input image.
Step2: Image is divided into number of blocks.
Step3: Each blocks scanning for complexity level.
Step4: Initialization of neurons.
Step5: Apply each neuron on the input layer from scan vectors.
Step 6: Execute the operation depending on the weights and the logic involve.
Step7: Passes them to the hidden layer.
Step8: Repeat the same as in step6.
Step9: Reassemble the outputs.
Step10: Neural network training and wait the weights.

## MATLAB RESULTS

In this projectMATLAB is used to implement the program. The well-known 'Rice' gray scale image (256 × 256) has been used to demonstrate the technique. Each pixel in an image can be denoted as a coefficient, which represents the intensity of the image at that point. Then, the idea of compressing an image is to encode these coefficients with reduced bits and at the same time, retain the quality of the image to satisfactory limits.The multi-layer feed-forward neural net has been used to compress images. The rice image that has been used for compression purposes is a 256 × 256 image. This image can be broken into blocks of size 32 ×32. There will then be 1024 pixels per block. Totally, there will be [32 × 32] = 64 blocks. The 1024 pixels in each block then becomes the input vector to the neural net. If each pixel is encoded using 8 bits, then the total number of bits to be transmitted without compression is [256 × 256 × 8] for a [256 × 256] pixel image. A [256 × 256] pixel image is split into [4 × 4] or [8 × 8] or [16 × 16] pixel sub-images. The normalized pixel value of the sub-image is the input to the nodes. 64 input layers (in case of [8 × 8] sub-image size) are takenwith 1 pixel input to each layer. The three-layered backpropagationlearning network has been trained with each sub image. The number of neurons in the hidden layer will be taken according to the desired compression. Here, we have taken 8 hidden layers. The number of neurons in the output layer will be the same as that in the input layer (64 in our case). The input layer and output layer are fully connected to the hidden layer. The weights of synapses connecting input neurons and hidden neurons and weights ofsynapses connecting hidden neurons and output neurons are initialized to small random values from say -1 to 1.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

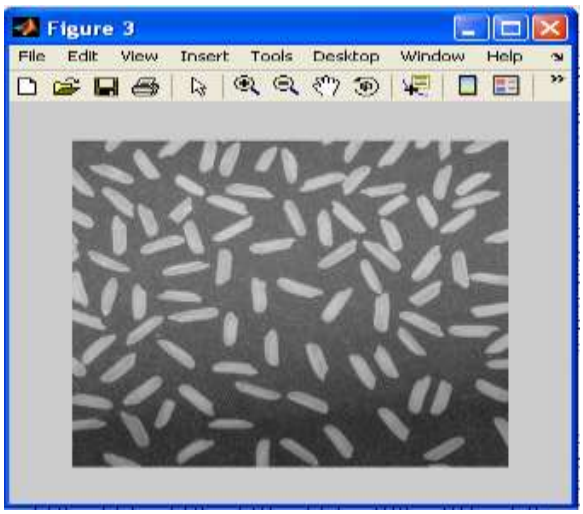



***Figure 5.1 Original 'Rice' image***

Only the weights between the hidden layer and the output layer are required for reconstruction. So, the numbers of weights are [64 × 8] and numbers of bits used to represent them are [64 × 8 × 8]. The input layer uses linear activation function. The hidden layer units evaluate the output using the sigmoid function.The output layer neuron evaluates the output using linear activation function. As the image is split into [8 × 8] pixel blocks, the total number of blocks becomes [32 × 32] and for each sub-image, the number of coefficients out of the hidden layer is 8. So, the total numbers of coefficients are [32 × 32 × 8]. Thus, total number of bits required to represent the coefficients are [32 × 32 × 8 × 8].Totalnumber of bits to be transmitted without compression is $T_b$. After training the network, number of bits to be transmitted is$C_b$ .Therefore compression achieved is, Compression= $[(1 – C_b /T_b) \times 100\%]$ Total number of bits to be transmitted without compression, $T_b$ = [32 × 32 × 8]. After training the network, number of bits to be transmitted,

$C_b = [(4 \times 32 \times 8) + (4 \times 32 \times 8)]$

Therefore compression achieved is:

$T_b = [32 \times 32 \times 8] = 8192$

$C_b = [(4 \times 32 \times 8) + (4 \times 32 \times 8)] = 2048$

Compression= $[(1 – C_b/T_b) \times 100\%] \approx 87\%$

$= [(1 – 2048/8192) \times 100\%]$

Compression = 75%

At the receiving end, the image is reconstructed by multiplying the weights between the hidden layer and the output layer to the corresponding coefficients obtained from the hidden layer.

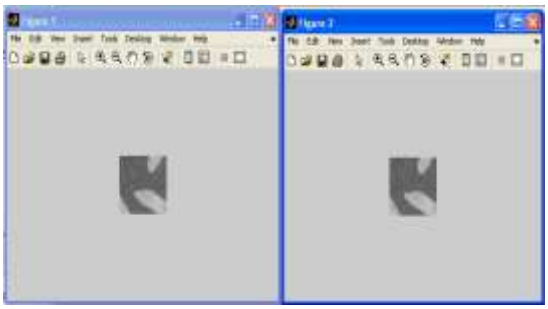


***Figure 5.2 Block of original 'Rice' image and Decompressed block***

## NOISE EFFECT ON IMAGE

To observe effects of noisy image on decompressed image different type of noise are added in image and their result are compared.

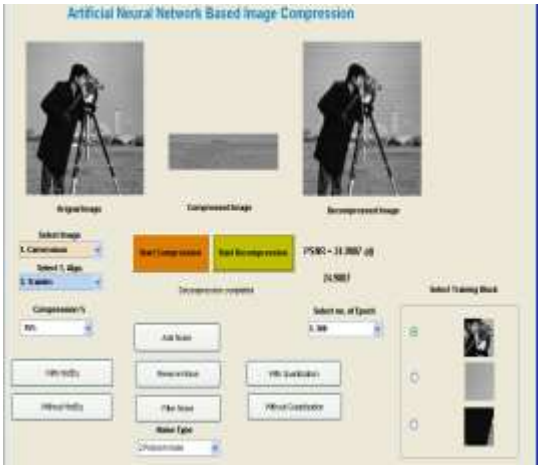
***Figure 6.1 Original Lena image and image with salt and pepper pixels noise***

Three type of noise are used, Figure 6.5 shows original image and image with on off pixel noise.Matlab function imnoise is used to add noise in image. J = imnoise(I,*type*) adds noise of a given type to the intensity image I. *type* is a string that can have one of these values. For adding on off pixel noiseJ = imnoise(I, 'salt & pepper') is used. For adding Poisson noise and Gaussian white noise value 'poisson'and 'gaussian'are used respectively.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

*Figure 6.2 Lena image with Poisson noise and Gaussian white noise*

## EFFECT OF NOISE ON CAMERAMAN IMAGE




In below table image without Histogram equalization and with noise is used. After compression, compressed image is given for decompression without quantization.

| Image | Algorithm Function | Noise | No. of Epoch | PSNR (Original Image) | Training Time |
|---|---|---|---|---|---|
| Cameraman | trainlm | Poisson | 200 | 26.0683 | 0.25 |
| Cameraman | trainlm | Poisson | 400 | 26.0683 | 0.51 |
| Cameraman | trainlm | Poisson | 600 | 26.0752 | 1.13 |
| Cameraman | trainlm | Poisson | 800 | 26.0707 | 1.47 |
| Cameraman | trainlm | Poisson | 1000 | 26.0708 | 2.06 |
| Cameraman | trainlm | on and off pixel | 200 | 19.2626 | 0.26 |
| Cameraman | trainlm | on and off pixel | 400 | 19.2646 | 0.48 |
| Cameraman | trainlm | on and off pixel | 600 | 19.2559 | 1.12 |
| Cameraman | trainlm | on and off pixel | 800 | 19.2597 | 1.48 |
| Cameraman | trainlm | on and off pixel | 1000 | 19.2595 | 2.05 |

***Table 6.1 Noise effect on Cameraman image***

From above result it can observed that for image with Poisson noise give more PSNR as compared image with on and off pixel noise. Figure 6.18 shows above result in graphical format

## CONCLUSION

In order to evaluate data compression and generalization capabilities of a neural network model as learning machines, image data compression and generalization characteristics were examined experimentally. Image datacompression using a 3-layered simple network developed using MATLAB. The image with Poisson noise gives more PSNR as compared to the image with salt and pepper noise

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

## REFERENCES

[1] B.Arunapriya, D.KavithaDevi, "Improved Digital Image Compression using Modified Single Layer Linear Neural Networks" International Journal of Computer Applications (0975 – 8887) Volume 10– No.1, November 2010

[2] R. C. Gonzales, R. E. Woods, Digital Image Processing, Second Edition, Prentice-Hall, 2002.

[3] H. NaitCharif, Fathi M. Salam, **"**Neural Networks-based Image Compression System", 43rd IEEE Mildwest Symposium on Circuits and Systems, PP. no. 846-849, vol. 2, IEEE 2000.

[4] Rudy Setiono and Guojun Lu, "Image Compression Using a Feed forward Neural Network", IEEE World Congress on Computational Intelligence, PP. No. 4761-4765, Vol. 7, IEEE 1994.

[5] S. Anna Durai, and E. Anna Saro "Image Compression with Back-Propagation Neural Network using Cumulative Distribution Function", World Academy of Science, Engineering and Technology 17, 2006.

[6] DiptaPratimDutta, Samrat Deb Choudhury, Md. Anwar Hussain, "Digital Image Compression using Neural Networks", International Conference on Advances in Computing, Control, and Telecommunication Technologies, PP. No. 116-120, IEEE 2009.

[7] PremaKarthikeyan, Narayanan Sreekumar "A Study on Image Compression with Neural Networks Using Modified Levenberg Maruard Method" Global Journal of Computer Science and Technology, Volume 11 Issue Version 1.0, March 2011.

[8] N. Sonhara, M. Kawato, S. Miyake and K. Nakane, "Image Data Compression using a Neural Network Model", Proc. Inter. Joint Conference on Neural Networks,11-35-11-41, 1989.

[9] Howard Demuth, Mark Beale,MartinHagan"Neural Network Toolbox™ 6 user's guide" by The MathWorks, Inc 2009.

[10] Hagan, M.T., H.B. Demuth, and M.H. Beale, "Neural Network Design", Boston, MA: PWS Publishing, 1996.