Implementation of Feed Forward Neural Network for Image Compression on FPGA

Yash V. Prajapati[1] Prof. Mayuri Prajapati[2]
[1]Student [2]Assistant Professor
[1,2]Department of Electronics and Communication Engineering
[1,2]Kalol Institute of Research Centre

Abstract— Compression of data in any form is a large and active field as well as a big business. Image compression is a subset of this huge field of data compression, where we undertake the compression of image data specifically. Research in this field aims at reducing the number of bits needed to represent an image. Inter-pixel relationship is highly non-linear and unpredicted in the absence of a prior knowledge of the image itself. Thus Artificial Neural Networks has been used here for image compression by training the net using the image to be compressed. This paper explores the application of artificial neural network to image compression. An image compressing take place in the hidden layer, after the pre-processing. In this paper, we are going to implement real time image compression on FPGA

Key words: Compression, Artificial Neural network, PSNR, MSE, FPGA

I. INTRODUCTION

Data compression has become a necessity for saving bandwidth, power, storage space, etc. Thus it has turned out to be a present day craze as well as source of competition in the race for technology and research with so much manpower, time and money involved for its development. Out of the image compression techniques available, transform coding is the most preferred method. Since energy distribution after transform coding varies with each image, compression in the spatial domain is not an easy task. Images do however tend to compact their energy in the frequency domain making compression in the frequency domain much more effective. Transform coding is simply the compression of the images in the frequency domain. So transform based techniques like DWT, DCT, SVD, DWT-DCT, DWT-SVD, etc. have been extensively used. It is known that compression algorithms can be classified into two types – 'lossy' and 'lossless'. If the recovered image (after decompression) does not have the same quality as the original image then there has been a loss of some image data during compression. This is called a 'lossy compression algorithm'. But some algorithms have the ability to retain the quality of the image, even after the compression, and the decompression processes. Such algorithms come under the category of 'lossless compression algorithms'. Among learning algorithms, back-propagation algorithm is a widely used learning algorithm in Artificial Neural Networks. The Feed-Forward Neural Network architecture is capable of approximating most problems with high accuracy and generalization ability. This algorithm is based on the error correction learning rule. Error propagation consists of two passes through the different layers of the network, a forward pass and a backward pass. In the forward pass the input vector is applied to the sensory nodes of the network and its effect propagates through the network layer by layer. Finally a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weight of the networks are all fixed. During the back pass the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from the desired response to produce an error signal. This error signal is then propagated backward through the network against the direction of synaptic conditions. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response.

II. IMAGE COMPRESSION PROCEDURE

In recent time, digital image is very popular. Every digital image is specified by the number of pixels associated with the image. An image with size of 256 x 256, means that there are 65536 pixels (intensity points) in the image in a matrix form with 256 rows and 256 columns.

There are two types of digital images: grayscale images and colour images. Colour images can be defined by the combination of the three primary colours – red, green and blue. A grayscale image has no colour information. Therefore, very pixel in a grayscale image has different shade of gray which is commonly represented by 8 (Unsigned integers of 8 bits). So, there is 28 =256 possible intensity values (shades of gray) for a grayscale image which ranging from 0 to 255. The depth of the image is said to be 8, since 8 bits are used to represent each pixel.

Since 8 bits are used to represent each pixel, to represent an image which is of dimension [256 x 256, 256 x 256 x 8] = 524288 bits are needed. With limited memory space, it becomes useful to compress the digital image so that it occupies less memory and also becomes easier to share over a medium such as the internet.

MATLAB has been used to implement the program. The well-known 'cameraman' (tiff format used here) grayscale image (256 x 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. Once compressed, these coded images which occupy less memory space can be transferred over the internet. At the receiver side, these compressed images need to be decoded or decompressed so that one can recover the original image. The quality of the received image can be tested by some standard error calculations. The mean of all the squared errors for all the pixels, called the MSE (Mean Square Error) can be used. The higher the value of this MSE, lower the quality of the decompressed image.

Here, the multi-layer feed-forward neural net has been used to compress image. The cameraman image that has been used for compression purposes is a 256 x 256

image. This image will be broken into blocks of size 4 x 4. So there will be 16 pixels per block. Totally, there will be [64 x 64] = 4096 blocks. Then, the 16 pixels in each block becomes the input vector to the neural network. The main idea in using a neural network to compress images is to code these 16 coefficients using a reduced number of coefficients (reduce the dimension of the data). The neural network architecture is very helpful in this context. If one can reduce the number of dimensions in the hidden layer (number of hidden neurons) to be much less than the number of dimensions in the input layer, then there will be a reduction in the number of coefficients during the transition from the input to the hidden layer. An input layer with 16 dimensions and a hidden layer with 10 (any number less than 16) dimensions, for example, means that the 16 pixels of the image (4 x 4) block which is applied to the input layer has been transformed into 10 coefficients in the hidden layer. Then, one could again use an output layer which has 16 dimensions to recover the original 16 pixels. The idea here is to learn the identity mapping or rather associative mapping which means the output of the neural net is the same as its input. Hence, input layer with a 16 dimensions, hidden layer with 10 dimensions, and output layer with 16 dimensions – the neural network has been used for image compression. The feed forward neural network and block diagram of compression and decompression is shown in below fig 1.

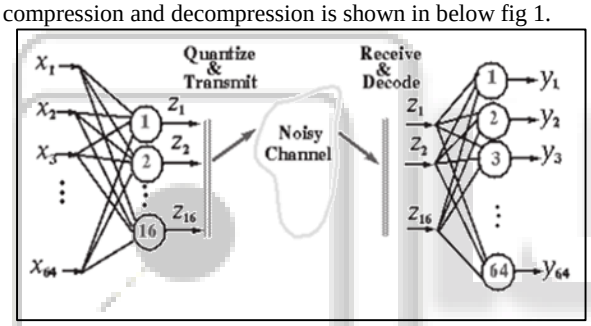


Fig. 1: The block diagram of compression and decompression of an image

In this image compression technique, the compression is achieved by training a neural network with the image and then using the weights and the bias from the hidden layer as the data to recreate the image. Here, we take 10 hidden layers. The number of neurons in the output layer will be equal to that in the input layer (16 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 of synapses connecting hidden neurons and output neurons are initialized to small random values from say –1 to +1. In our case, we have used linear activation function at input layer and output layer. So, hidden layer units evaluate the output using linear activation function. As the image is split into [4 x 4] pixel blocks, the total number of blocks becomes [64 x 64] and for each sub-image, the number of coefficients out of the hidden layer is 10. So, the total numbers of coefficients are [16x16x10]. Thus, total number of bits required to represent the coefficients are [16x16x10x10].
Therefore compression achieved is:

Compression Ratio = (n1/n2)*100=62.50 %

At the receiving side, 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.

In our case, the network is train using Scaled Conjugate gradient Algorithm which is best suitable for the pattern recognition problems. Trainscg is a network training function that updates weights and bias values according to the scaled conjugate gradient method. The compression performance is evaluated in terms of Compression ratio, PSNR and execution time. The execution time is measured from the time of start of training until the error saturates to a minimum and the trainscg function stops execution.

III. Results of ANN

The main two images, well-known cameraman (.tif format) image and rice image have been analyzed and compress using artificial neural network and reconstruct it at the receiver side. The original and reconstructed images are shown in Fig 2.

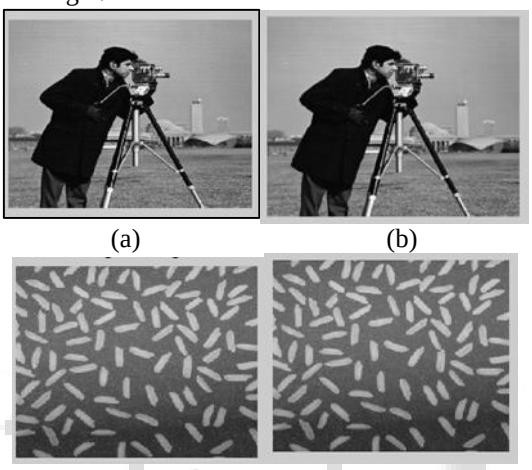


Fig. 2(a): original cameraman image (b): reconstructed cameraman image (c): Original rice image (d): reconstructed rice image

Here, we cannot see actual difference from the original and reconstructed figures. So, we derived results in terms of CR, PSNR, MSE and Execution time.

The following table 1 shows the results of ANN.

Image	Performance parameter			
	Time (sec)	PSNR (dB)	CR (%)	MSE
camer aman	0.1103	81.4419	62.5	4.5750e-04
leena	0.1164	84.4215	62.5	2.3677e-04

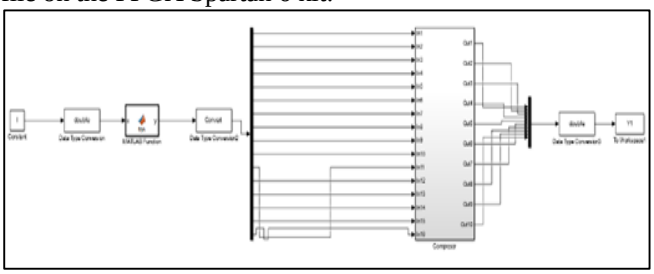
Table 1: Results of ANN

IV. Simulation Models and Hardware Results

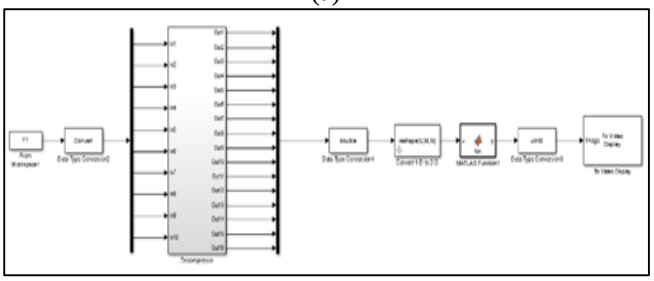
To implement this ANN on FPGA, we need to make simulation model of compression and decompression. Using HDL work flow advisor, vhdl code can be generated to load our ANN on FPGA.

Here, we have made real time compression and decompression model as shown in below fig 3 (a) & (b). I capture real time image and is given to the compressor part where image is compressed into the hidden layer. After that, we need to generate hdl codes using FPGA-In the-Loop in the HDL workflow advisor. A FIL block named compressor module can be generated in a new model. After that, this newly generated FIL compressor block need to be replaced

with original compressor block and then load programming file on the FPGA Spartan 6 kit.



(a)



(b)

Fig. 3(a): Compression model (b): Decompression Model

The output of the compression model is then given to the input layer of the decompression model. Decompression is the reverse process of the compression where compressed image is recovered. The following fig 4 shows the real time image which is recovered at the decompression side.

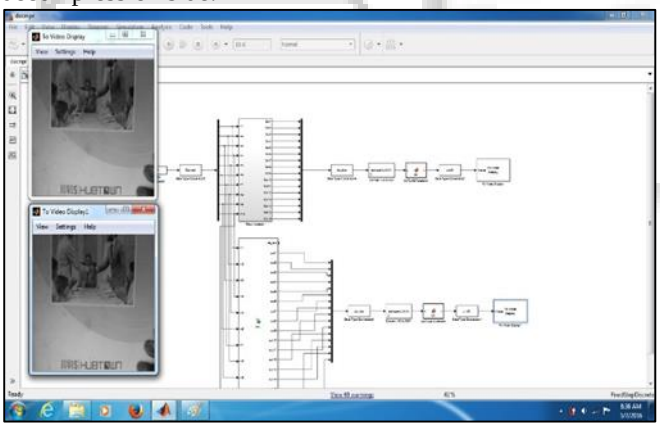


Fig. 4: hardware results which shows the reconstructed real time image

V. XILINX IMPLEMENTATION

The hardware implementation of the project can be realized on FPGAs. Here, for my project point of view, we chose Xilinx Atlys Spartan 6 development board because these FPGA board supports higher multiplications and addition functions. First of all, the design is synthesized using Xilinx ISE. By applying appropriate constraints and using various sub-tool from Xilinx, we can complete implementation. Results of multiplier program which uses large amount of adders and multipliers. Multipliers are used to multiply hidden layer matrix with weight matrix and also output layer matrix with weight matrix.

VI. CONCLUSION

Artificial Neural network based image compression produced some promising results. Neural Network is well suited to the task of processing image data. They are highly parallel structure and a high degree of inter connections. They supports higher multiplication and additions. So from the results, we can concluded that the feed forward neural network is very effective structure to compress the image. We obtained high PSNR, less execution time to compress the image and better MSE than other traditional methods. It has also observed that as the image size increases, higher compression is obtained and errors are also reduced, but training time increases significantly.

REFERENCES

[1] R.Praisline Jasmi, Mr.B.Perumal, Dr.M.Pallikonda Rajasekaran," Comparision of Image Compression Techniques using HuffmanCoding,DWT and Fractal Algorithm", International Conference on Computer Communication and Informatics 2015.

[2] Smitha Joyce Pinto, Prof.Jayanand, P.Gawande, "Performance analysis of medical image compression technique", 978-1-4673-2590-5/5/12/ ©2012 IEEE

[3] Yevgeniya Sulema (Ukraine), Samira Ebrahimi Kahou (Iran),"Image Compression: Comparative Ananlysis of Basic Algorithms", 978-1-4244-9556- 6/10 2010 IEEE.

[4] Dipta Pratim Dutta, Samrat Deb Choudhury, Md. Anwar Hussain, Swanirbhar Majumder," Digital Image Compression using Neural Networks", International Conference on Advances in Computing, Control and Telecommunication Technologies,2009 IEEE.

[5] Rafael Gadea, Joaquin Cerda, Franciso Ballester and Antonio Mocholi," Artificial Neural Network Implementation on a single FPGA of a Pipelined on-Line Backpropagation".

