International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307 (Online), Volume-9 Issue-6, March 2020

Deep Autoencoder-Based Image Compression using Multi-Layer Perceptrons

G.G.H.M.T.R. Bandara, R. Siyambalapitiya

Abstract*:The Artificial Neural Network is one of the heavily used alternatives for solving complex problems in machine learning and deep learning. In this research, a deep autoencoder-based multi-layer feed-forward neural network has been proposed to achieve image compression. The proposed neural network splits down a large image into small blocks and each block applies the normalization process as the preprocessing technique. Since this is an autoencoder-based neural network, each normalized block of pixels has been initialized as the input and the output of the neural network. The training process of the proposed network has been done for various block sizes and different saving percentages of various kinds of images by using the backpropagation algorithm. The output of the middle-hidden layer will be the compressed representation for each block of the image. The proposed model has been implemented using Python, Keras, and Tensorflow backend.

Keywords*:Image Compression, Deep Learning, Autoencoder, Backpropagation Algorithm.

I. INTRODUCTION

Data Compression is one of the fascinating areas all over the world since people have limited storage to store huge data files such as high-quality images and videos. Development of the fields such as Image Processing, Computer Vision, and Multimedia directly affect the creation of images and videos which have excellent quality with sharp details. Compression can be performed on various multimedia components such as text, video, image, audio, and graphics. In this research, we have proposed an image compression algorithm based on an autoencoder model using a deep fully connected feed-forward neural network. A digital image is basically a two-dimensional array of pixels arranged on a two-dimensional space. Those pixels of an image may have redundant or irrelevant pixels. In image compression, what we do is reduce those redundant pixels and remove irrelevant pixels. Those pixels are ignored by the human visual system in such a way that the compressed array of pixels consists of a smaller number of pixels than the original array of pixels. Image compression may be lossy or lossless in terms of the nature of the decompressed image.

An artificial neural network (ANN) is one of the main tools heavily used in machine learning and deep learning. In this research, we used a deep neural network that consists of multiple hidden layers. When it comes to image compression, deep learning is heavily used. Because it has the ability to learn patterns from an input image and transform it into another pattern with fewer components. The neural network learns by adjusting weights (synapses) of interconnections between layers. In general, the learning process of a neural network can be done using two different kinds of learning mechanisms; supervised learning and unsupervised learning. An autoencoder based method is one of a preferable unsupervised learning method which uses in various research domains [1].

An autoencoder neural network is a special kind of unsupervised learning algorithm that has three types of layers as the traditional neural network: an input layer, a hidden layer (encoded), and an outputlayer(decoded). An autoencoder neural network should have at least one hidden layer for the encoded representation over traditional neural networks. The network is trained in such a way that the output values are equal to the input values. In order to do that, in the learning process, autoencoder uses dimensional reduction technique and try to preserve the essential features of the original data while removing the non-essential features.

Image compression is one of the applications of an autoencoder. In this, an autoencoder algorithm has two functions called the compression (encoded) and decompression (decoded). Instead of having two separate algorithms,here autoencoders can perform both processes within one particular neural network. Our research work was done using a deep fully-connected autoencoder neural network.

In basic compression algorithms, there are two separate algorithms to perform both compression and decompression processes. Therefore, in order to achieve compression, they use a lot of techniques and algorithms inside of their original compression algorithm. It needs a lot of computational power and hardware resources with high-performance GPUs. Another problem is that in most of the case studies related to the image compression using neural networks, they have used a convolutional neural network to achieve the compression and it also needs a lot of computational power. Therefore, a deep autoencoder-based multi-layer feed-forward neural network is proposed to achieve the image compression. In image compression,

Revised Manuscript Received on February 17, 2020.
* Correspondence Author
G.G.H.M.T.R. Bandara*, Department of Statistics & Computer Science, Faculty of Science, University of Peradeniya, Peradeniya, Sri Lanka. Email: thushararanjana@gmail.com
R. Siyambalapitiya, Department of Statistics & Computer Science, Faculty of Science, University of Peradeniya, Peradeniya, Sri Lanka. Email: ruwanthini@gmail.com

it is pretty difficult to train an autoencoder-based neural network that gives a better performance than the other basic compression algorithms. The performance of the existing compression algorithms using autoencoders is greatly low compared with the basic algorithms.

In this study, the main objective is to develop an autoencoder neural network to handle both the compression and decompression processes as a single compression engine. In order to do that implement a deep autoencoder-based multi-layer feed-forward neural network and train the model using a block-based technique using minimal hardware performance. The training processes of the model will be done for different block sizes and the saving percentages. The performance of the model will be measured by using standard measuring components such as PSNR, SSIM, and MSE. The optimal model will be tested using a benchmark and try to achieve an acceptable level of generalization for a specific domain of images.

II. RELATED WORKS

The predictive compression of images using neural networks has been studied by Florin Alexa, VasileGui, CatalinCaleanu, and Corina Botoca [2]. In order to compute the predicted pixels, the backpropagation algorithm is used. Linear prediction compression used in the JPEG algorithm is considered to compare the validation of the results. The entropy of prediction error images of the neural networks is slightly higher than the entropy of prediction error of images in linear prediction. However, the linear prediction is better whereas the neural network one in terms of the entropy value. Since the neural networks minimize the mean squared error, the neural network's prediction leads to better results. They expect that the performance of the neural network predictor would be improved using an advanced lossless compression technique.

Autoencoders need for compression algorithms which are more flexible than existing codecs but are difficult to optimize directly due to the inherent non-differentiability of the compression loss. For that LucasTheis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszar [3] have been introduced a new approach for lossy image compression to the problem of optimizing autoencoders. They showed that minimal changes to the loss are sufficient to train deep autoencoders competitive with JPEG 2000 and outperforming recently proposed approaches based on RNNs. This performance was achieved using an efficient convolutional architecture, combined with simple rounding-based quantization and a simple entropy coding scheme. This method is suitable for high-resolution images. This is in contrast to previous work on autoencoders for compression using coarser approximations, shallower architectures, computationally expensive methods, or focusing on the small image.

III. METHODOLOGY

The proposed model is trained using different learning rules for various kinds of images. Since our research is an image-based learning process, it is important to explain the phases which have been done during the processes of compression and decompression. The proposed algorithm has five main phases and each of those phases can be further divided into sub-phases.

Phase 01 - Preprocessing Image Data

The preprocessing phase of an image can be performed under several sub-phases. First, resize the image into a 256×256 image. Then convert it into a grayscale image. Next, normalize the pixel values between 0 and 1. Finally, split the normalized image into several blocks that are not overlapped. The purpose of having a block-based approach is that the associated image can be excessively large to load fully into memory. If we tried to load such a large image using a deep learning pipeline as there may not be enough RAM on the GPU for both image data and the filter activations. In this research, we have used several sizes of blocks such as 4×4, 8×8, and 16×16 in order to perform the splitting of an image.

Phase 02 - Building and Training the Autoencoder-based Model

Develop an algorithm that can be operated on each block of an image using a deep autoencoder-based feed-forward neural network. Suppose that we have a set of unlabeled training samples $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$ where $x^{(i)} \in R^n$. Then the autoencoder tries $y^{(i)} = x^{(i)}$ to achieve the final trained model of the neural network. An autoencoder can do; the encoding process and the decoding process. The encoded representation of an image can be obtained at the middle-hidden layer (fewer number of neurons) and the decoded representation of an image can be obtained at the output layer (number of neurons as same as the input layer).

The autoencoder is trying to learn a function $h_{w,b}(x) \approx x$, so as to the output of the neural network $\hat{x}$ that is similar to the input of the neural network x. In order to do that the autoencoder repeats the "**Backpropagation Algorithm (BPA)**" iteratively until the error is less than a particular threshold value or absolute rate of change in the mean squared error (MSE) per epoch is sufficiently small.

The training process of the proposed neural network is done for different block sizes and different saving percentages by varying the number of neurons in the middle-hidden layer. In order to do that, 75% of the total number of blocks are assigned to the training set and the remaining 25% of the total number of blocks are assigned to the testing process. In order to handle the training process, we implement early stopping criteria as a callback function. Those callback functions apply at the end of each epoch. Especially, in our solution, we include early stopping to define what we wanted to monitor the validation loss at each epoch and the test loss has not improved after two epochs, the training process is interrupted. In order to get the optimal model, we use a model checkpoint which saves the best model to a file after every checkpoint.

Phase 03 - Reconstructing Image Data

The reconstruction of the image can be obtained by compiling the steps in the reverse order. In order to do that the pixels of the output layer reshape into 2D blocks of pixels. Then merging the split blocks in order to construct the image withoriginal dimensions (256×256). Next, perform the denormalization process to transform the values of pixels to the

original range by multiplying each pixel value by 255. Finally, convert the image into its original format and the resultant image will be the reconstruction of the original image which is decoded by the proposed algorithm.

Phase 04 - Measuring Compression Performance

Depending on the nature of the image compression algorithm there are various kinds of criteria that can be used to measure the performance of the algorithm and the quality of the decoded image. When measuring the performance of an image compression algorithm the main concerns would be space efficiency and time efficiency. In our research, we discuss a lossy image compression algorithm, for that the following measuring components are used to evaluate the performance.

A. Compression Time

Time taken for both compression and decompression processes.

B. Compression Ratio

The ratio between the size of the compressed image and the size of the original image.

C. Compression Factor

The ratio between the size of the original image and the size of the compressed image.

D. Mean Squared Error (MSE)

The cumulative squared error between each corresponding pixel in the original and the decompressed image [4, (1)].

$$MSE = \frac{1}{M \times N} \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I'(x,y)]^2 \qquad (1)$$

Where,

$I(x,y)$ -The intensity value in the original image

$I'(x,y)$-The corresponding intensity value in the decompressed image

M, N-The dimensions of the images

E. Peak Signal to Noise Ratio (PSNR)

The ratio between the maximum possible intensity value of an image and the power of distorting noise [5, (2)] that affects the quality of its representation as a logarithmic decibel scale.

$$PSNR(dB) = 20 \times \left(\frac{I_{Max}}{\sqrt{MSE}}\right) \qquad (2)$$

Where,

I_{Max} -The maximum intensity value of the image

F. Structural Similarity Index Measure (SSIM)

The SSIM is a perceptual metric that quantifies the degradation of images. It needs two images from the same image capture as same as measuring PSNR. Unlike PSNR, SSIM based on the visible structure in images and measure the similarity between the original image and the decompressed image[6] [7, ((3), (4), (5), (6), and (7))].

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \qquad (3)$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \qquad (4)$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \qquad (5)$$

$$c_3 = \frac{c_2}{2} \qquad (6)$$

It calculates the weighted combination of the (3), (4), and (5) measures.

$$SSIM(x,y) = [l(x,y)^{\alpha} \times c(x,y)^{\beta} \times s(x,y)^{\gamma}] \qquad (7)$$

Where,

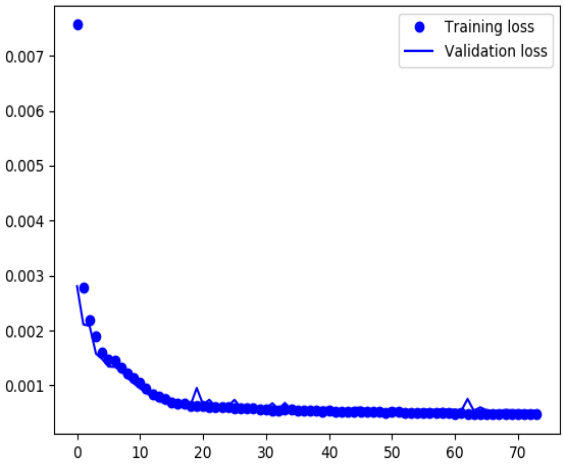
$l(x,y)$ - Luminance
$c(x,y)$ - Contrast
$s(x,y)$ - Structure
α, β, γ - Weights

IV. RESULTS AND DISCUSSION

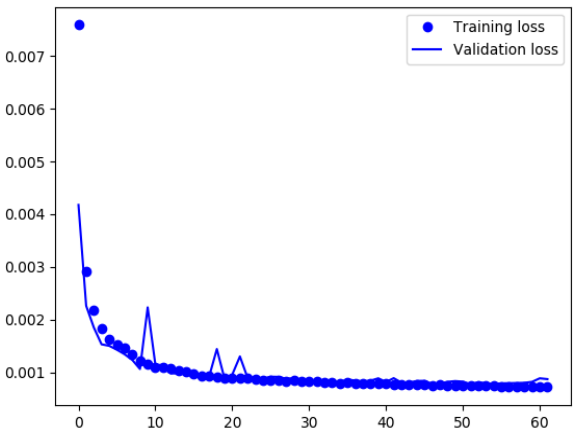
In this research, the proposed model was tested with the Lena image, the Head-MRI image, and the Moon image. The experimental results obtained from the Lena image are given below.

Split into 4 × 4 blocks

The experimental results were obtained for the variation of the training loss and the validation loss for the Lena image over different saving percentages. The saving percentages are50%, 62.5%, 75%, and 87.5%.

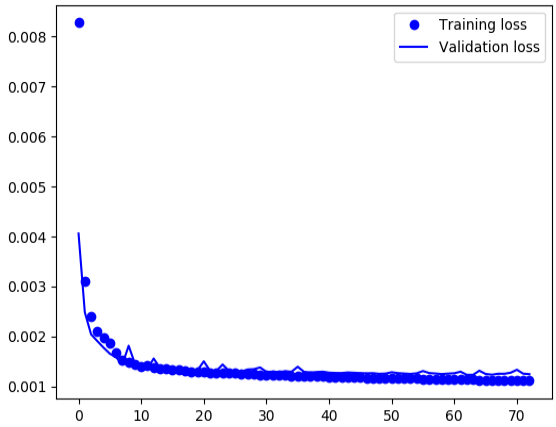


(A) 50% Saving

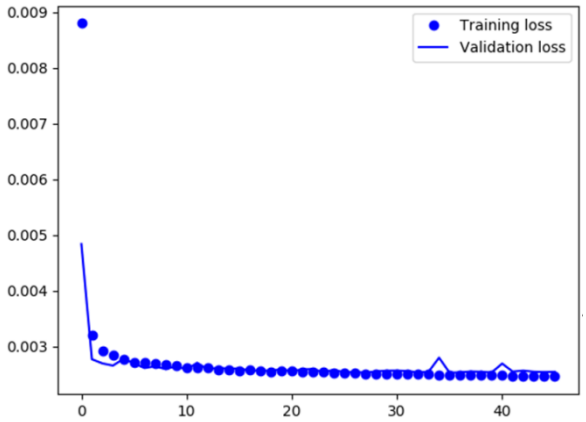


(B) 62.5% Saving

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication



(C) 75% Saving



(D) 87.5% Saving

Fig. 1. Variation of the graph of the training loss/validationloss vs. epoch for the Lena image

When we consider all the saving percentages, the training loss and the validation loss showed a steady and significant decrease over the period and both are in sync. After the model was trained, the image was predicted by reconstructing the decompressed images as given by Fig. 2.

(A) Original (B) 87.5% Saving (C) 75% Saving (D) 62.5% Saving (E) 50% Saving

Fig. 2. Reconstruction of the Lena image for different saving percentages

Table-I: Evaluation of compression performance of all test images using 4 × 4 block

	87.5%			75%			62.5%			50%		
	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon
Compression Ratio	0.125	0.125	0.125	0.25	0.25	0.25	0.375	0.375	0.375	0.5	0.5	0.5
PSNR	26.07	24.92	26.25	29.59	29.17	28.49	31.38	31.95	30.61	33.22	34.21	32.77
MSE	144.8	209.46	154.35	66.71	78.76	92.04	42.6	41.5	56.45	27.9	24.64	34.34
SSIM	0.67	0.49	0.6	0.8	0.62	0.76	0.85	0.65	0.84	0.89	0.67	0.91
Processing Time (seconds)	3.55	3.64	3.35	3.77	3.57	3.63	4.18	3.73	6.52	4.44	5.08	3.41

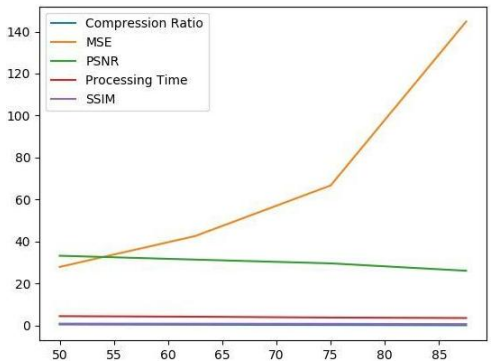


Fig. 3. Overview of the graph of the value vs. saving percentage of each measuring component for the Lena image

According to the original image and the decompressed image, the measures for PSNR, SSIM, MSE, and the compression time were computed. The computed measures were given by Table I.

As stated in Table I, the highest measures for the PSNR in each saving percentage are obtained by the Lena image and the Head-MRI image whereas the highest measures for the SSIM in each saving percentage are obtained by the Lena image and the Moon image. The compression time always depends on the hardware performance and the work-load handled by the OS. The overview of the measures obtained by the Lena image is given by Fig. 3.

Similarly, the experiment was done using 8×8 blocks and 16×16 blocks. The results taken from those 8×8 blocks and 16×16 blocks are given by Table II and Table III respectively.

Split into 8 × 8 blocks

Table-II: Evaluation of compression performance of all test images using 8 × 8 blocks

	87.5%			75%			62.5%			50%		
	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon
Compression Ratio	0.125	0.125	0.125	0.25	0.25	0.25	0.375	0.375	0.375	0.5	0.5	0.5
PSNR	27.66	26.35	26.89	29.61	29.54	29	31.11	30.83	30.71	33.17	32.01	32.43
MSE	105.02	150.58	132.98	64.07	72.4	81.95	45.74	53.66	55.26	35.5	40.91	37.16
SSIM	0.68	0.52	0.61	0.76	0.6	0.76	0.81	0.62	0.83	0.84	0.64	0.88
Processing Time (seconds)	3.61	3.84	4.43	4.55	5.76	4.6	3.9	3.11	6.18	3.84	5.29	3.84

Split into 16 × 16 blocks

Table-III: Evaluation of compression performance of all test images using 16 × 16 blocks

	87.5%			75%			62.5%			50%		
	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon	Lena	MRI	Moon
Compression Ratio	0.125	0.125	0.125	0.25	0.25	0.25	0.375	0.375	0.375	0.5	0.5	0.5
PSNR	25.05	23.45	24.16	24.49	23.87	24.12	24.5	23.86	24.6	23.9	23.93	24.98
MSE	195.7	293.74	249.72	219.13	266.8	251.8	216.7	267.5	225.9	250.4	263	206.72
SSIM	0.5	0.37	0.33	0.48	0.4	0.33	0.49	0.4	0.38	0.44	0.4	0.42
Processing Time (seconds)	3.24	2.89	3.02	2.96	3.11	2.99	3.33	3.08	2.77	2.54	3.01	2.82

But the resultant values for the PSNR and the SSIM were not better compared with the results obtained from the 4 × 4 blocks. Therefore, the proposed model with 4 × 4 blocks was the best model with high performance. In order to perform the validation of that model, the JPEG compression algorithm was used. Since the modern approach of quantifying the degradation of images is SSIM, the performance of the JPEG for the Lena image and the Moon image was computed. The computed values are given as follows.

Table-IV: The SSIM values of the Lena image and the Moon image in different saving percentages

	87.5%		75%		62.5%		50%	
	Lena	Moon	Lena	Moon	Lena	Moon	Lena	Moon
Proposed neural network with 4 × 4 blocks	0.67	0.6	0.8	0.76	0.85	0.84	0.89	0.91
JPEG compression	0.93	0.92	0.95	0.942	0.97	0.964	0.996	0.992

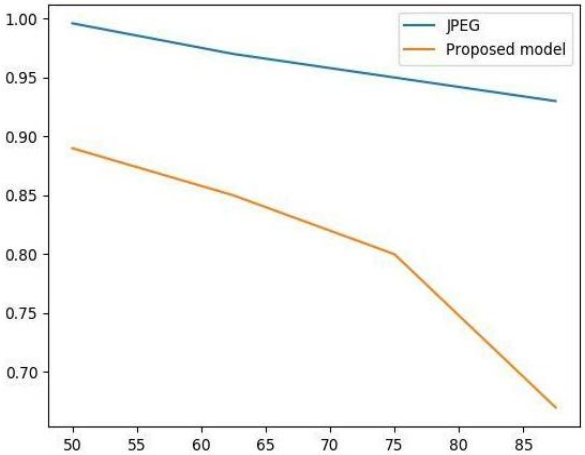


Fig. 4. The graph of the SSIM measure vs. saving percentage for the proposed model and the JPEG compression from the Lena image

The values obtained by both the proposed model and the JPEG compression algorithm are decreased over saving percentage. The SSIM values obtained by the proposed model are considerably low when the saving percentage goes to 87.5%. But when we compare the values obtained in 50% and 62.5% savings, the difference between the SSIM values of the proposed model and the JPEG compression is relatively small. Also, the reconstructions of the Lena image and the Moon image by both compression algorithms are comparatively similar in terms of the human visual perception. The variation of the SSIM values for both algorithms over saving percentage is given by Fig. 4.

Since the results obtained from the Lena image and the Moon image were in a better range, the model proposed with4 × 4blocks were retrained against a specific category of an image set with the 87.5% saving percentage. In order to do that the faces category of the Caltech 101 image set [8] was used. In this training process, there were 435 images used and it gave us 1.78 million blocks. 75% of the resultant blocks were used to perform the training process; that is 1.34 million, and the testing was done using the remaining number of blocks; that is 0.44 million. After fully trained the neural network, untrained images of faces were used to achieve the reconstruction of the images. All the results obtained from those images tended to be in the same value with a slight difference in the PSNR and the SSIM measures depend on the distribution of their grey levels.

V. CONCLUSION

In this research work, the proposed neural network model is trained and tested for three different block sizes in different saving percentages. The model with 4×4 blocks gives better performance on the reconstruction of the Lena image and the Moon image in terms of the PSNR, SSIM, and MSE measures over other models with 8×8 blocks and 16×16 blocks. Therefore, the block size is an important fact even with the same saving percentage. But according to the human visual perception, both models with 4×4 blocks and 8×8 blocks give similar performance. Also, when the model was compared with the JPEG compression algorithm, the results obtained are quite similar and the reconstruction of images from both algorithms looks the same in terms of the human visual perception. Therefore, the proposed model has been checked for the capability of handling the generalization for a set of specific data set. In order to do that, the proposed model was trained for a large set of face images from the Caltech 101 image set. The results obtained from the untrained images are much better in terms of all the measuring components such as PSNR, SSIM, and MSE. Therefore, the proposed model has the capability of handling the generalization for a set of specific images and it has an ability to hold the large enough of training data and small enough for the size of the network. On the other hand, when we consider the time taken to finish the training process of the neural network is quite low even the algorithm is run on a low-performance machine without having a GPU.

As a further improvement, the proposed model can be implemented using a convolutional neural network (CNN) to enhance the performance of the existing compression algorithm. Since the ability to handle the multiple components of an image, the CNN model can be used to achieve the compression of a color image such as RGB. Therefore, the improvement of handling the compression of color images benefit to develop the existing system to accomplish the Video Compression (VC) in the context of the multimedia systems.

REFERENCES

1. Supervised vs. Unsupervised Learning, Understanding the differences between the two main types of machine learning methods https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d by Devin Soni
2. Lossless Data Compression Using Neural Networks; by Florin Alexa, VasileGui, CatalinCaleanu and Corina Botoca (Romania); Proceedings of the 7th WSEAS Internation Conference on CIRCUIT, SYSTEMS, ELECTRONICS, CONTROL and SIGNAL PROCESSING (CSECS' 08)
3. Lossy Image Compression with Compressive Autoencoders; by Lucas Theis, Wenzhe Shi, Andrew Cunningham and Ferenc Huszar (London, UK); Conference paper at ICLR 2017
4. Todd Veldhuizen; Measures Image Quality; http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html
5. Peak Signal-to-Noise Ratio as an Image Quality Metric; http://www.ni.com/white-paper/13306/en/
6. SSIM: Structural Similarity Index; http://www.imatest.com/docs/ssim/
7. Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004
8. Caltech101 – Computational Vision at Caltech; http://www.vision.caltech.edu/Image_Datasets/Caltech101/

AUTHORS PROFILE

G.G.H.M.T.R. Bandara received a B.Sc. degree (Hons) in Computer Science from the University of Peradeniya, Sri Lanka. He is currently a Temporary Lecturer in Computer Science at the Department of Statistics & Computer Science, University of Peradeniya, Sri Lanka. His work is mostly related to the Research and Development (R&D) projects, and his interest covers in the fields of Computer Vision, Deep Learning, Signal Processing, and Data Science.

R. Siyambalapitiya received a Ph.D. in Computer Engineering from the Faculty of Engineering, University of Peradeniya. She is currently a Senior Lecturer in Computer Science at the Department of Statistics & Computer Science, University of Peradeniya, Sri Lanka. Her areas of interest and research cover in the fields of Design of Algorithms, Optimization Scheduling, and Operating Systems.