**An unstructured MRI compression technique using a convolutional approach**

Debkumar Chowdhury1\*, Tirtharaj Sinha1, Arnobrata Ghosh1, Anurag Unnikannan1, Susmit De1, Devesh Raj1and Siddhartha Bose1

1University of Engineering and Management, Kolkata, India

1[debkumar.cse@gmail.com](mailto:debkumar.cse@gmail.com),1[sinhatirtharaj@gmail.com](mailto:sinhatirtharaj@gmail.com) ,1[ghosharnobrata@gmail.com](mailto:ghosharnobrata@gmail.com),1[anuragunnikannan7@gmail.com](mailto:anuragunnikannan7@gmail.com),1[desusmit101@gmail.com](mailto:desusmit101@gmail.com), 1[rajdevesh437@gmail.com](mailto:rajdevesh437@gmail.com) ,1[bosesiddhart1234@gmail.com](mailto:bosesiddhart1234@gmail.com)

**Abstract.** Currently, different methods are available for the purpose of MRI compression technique. Most of these techniques are well appreciated by society and in response to the demand of society, almost every year different techniques are introduced by different researchers. But it does not satisfy the demand of current requirements. Under such a situation, we are going to propose a new MRI compression and storing technique using a convolutional approach. This technique starts with the MRI preprocessing step. In the next step, the segregation of the dataset into the training and testing phase is performed. Then the convolutional model architecture is trained using the preprocessed MRIs from the dataset. After that our proposed algorithm performs encoding and decoding of preprocessed MRI. We have calculated the performance parameters and found that our proposed algorithm can be used for MRI compression technique using a convolutional approach and has a compression ratio of 1.4593%. Novel preprocessing steps and modifications in the convolutional architecture make the proposed methodology unique. Due to high performance, novelty, ease of use, our proposed method is useful to develop any mobile or web application in the future.

**Keywords:** Preprocessed image, Convolutional, Compression Ratio, Image Compression

# Introduction

In today’s world with the advancement of medical procedures, and the new researches in this field has helped save hundreds of thousands of more patients with every passing year. Medical equipment such as a Computed Tomography (CT scanner), Magnetic Resonance Imaging (MRI scanner), Positron Emission Tomography (PET scanner), Ultrasound scanner, etc., are used to generate images of the organs in the body. These images in turn are analyzed by specialists and experts in the medical field to provide a proper diagnosis of the patient. These images are very crucial as they can show if the patient is suffering from any ailment, or a disease, or has a tumor, etc., and can thus help provide proper treatment in time.

Therefore, these images need to be in high resolution, as every little detail has to be visible for it to be analyzed. This means that every image has a large size, and there are hundreds of images alone for every patient from each test alone. But space is limited, especially in medical facilities that don’t have enough resources or funds to sustain the storage of so many large sized images.

Our project aims to provide a remedy to this problem. We originally take all the original images and store it in a dataset. Then we use an Autoencoder to compress the images, hence producing an encoded image, and then we decode it and then check how close it is to the original image.

Our proposed methodology aims to improve on existing technologies and make healthcare better and more accessible to more individuals across the globe. It displays the classified output using convolutional model architecture along with the satisfactory performance parameters (SSIM [8], PRD, PSNR, Compression Ratio [1][2], BPP, etc.).

Our paper is divided into various sections. In section 2, we explain the advantages and disadvantages of some pre-existing techniques, in section 3 we explain our proposed methodology which consists of the main architecture of the method and the algorithm, whereas the experiment result and analysis are explained in section 4.

1. **Literature Survey**

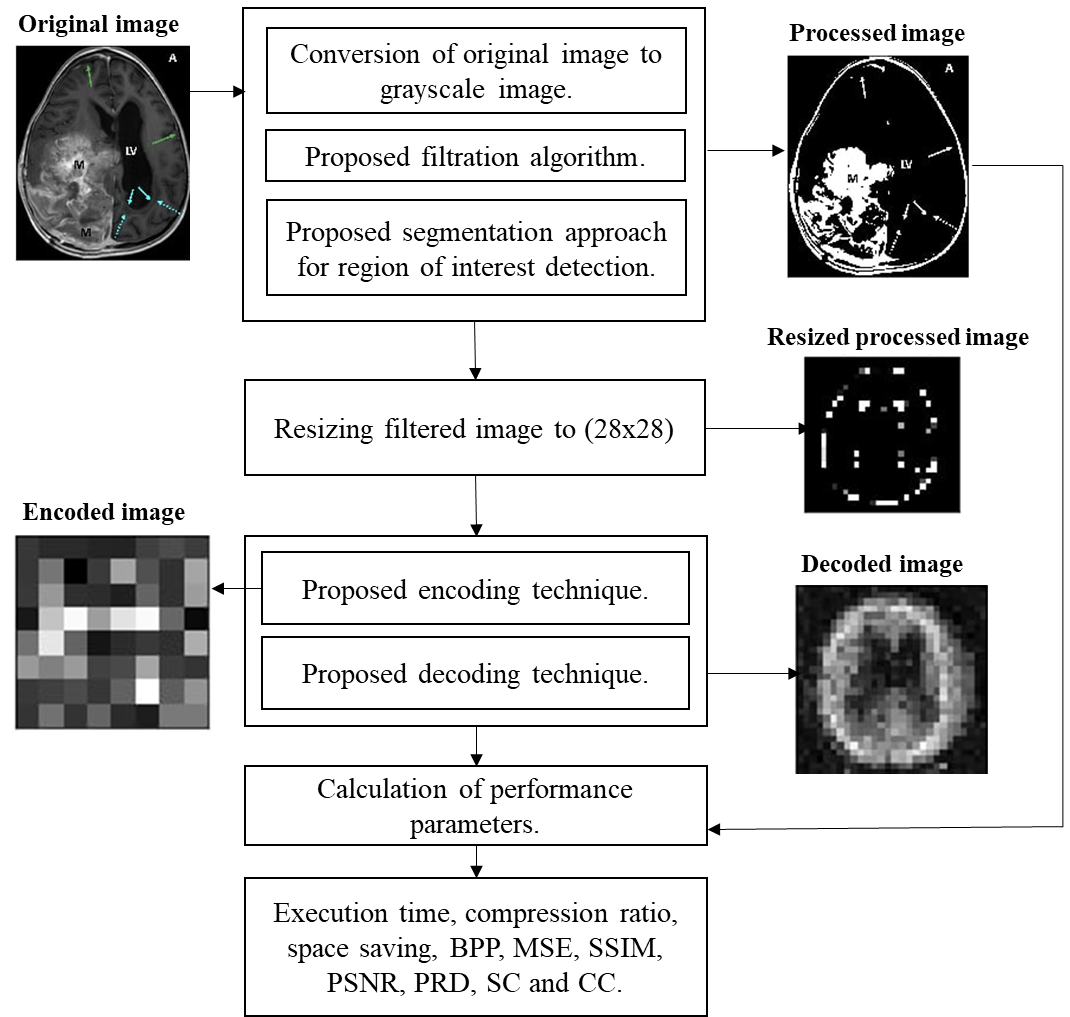
This section represents various pre-existing methods [1] [2] [3][5], along with their advantages and disadvantages as shown in the analysis Table (**Table. 1**). All these methods are well appreciated but in context with our problem the results can be improved. In-depth analysis of these methodologies has proven to be very competent to identify the downsides. Identification of these drawbacks helps us to update, modify our algorithm and code and to calculate the performance parameters. The Survey Table is given below:

**Table. 1.** Existing Methodology Analysis Table

|  |  |  |
| --- | --- | --- |
| **Methodology** | **Advantages** | **Disadvantages** |
| LZW Compression [1][3] | It provides greater compression ratio and appropriate for larger data. | Requires more compression  and decompression time and inappropriate for smaller data. |
| Huffman Coding [1][3] | Requires less compression  and decompression time and appropriate for smaller data. | It provides less compression ratio and inappropriate for larger data. |
| Embedded Zerotree Wavelet (EZW) [2] | User can control desired bit rate. | Properties of the image affects the performance. |
| Shannon–Fano Encoding [3][4] | For Shannon Fano coding procedure we do not need to build the entire codebook. | Shannon–Fano Encoding sometime fails to produce an optimal tree. |
| Run-length encoding [3] | It works well when an image contains long runs of identical samples that usually do not appear in an authentic image. | It’s inappropriate for larger data.[9] |
| Arithmetic coding [3][5] | provides a better compression ratio. | It can corrupt the whole image for a minute error because it has very poor error resistance. |

1. **Proposed Methodology**

Our proposed methodology focused on medical image compression from a custom hydrocephalus dataset using image compression algorithm using a convolutional approach. We have proposed a block diagram to show the main concept of the methodology at a glance as shown in **Fig. 1.**



**Fig. 1.** The built-in architecture of MRI compression technique using a convolutional approach

* 1. **Algorithm**

Our algorithm is divided into 8 steps. The algorithm takes a custom hydrocephalus dataset, as an input from the dataset [10] and produces an encoded image and a decoded image. The algorithm is as follows:

**Algorithm:** An unstructured MRI compression technique using a convolutional approach

**Input:**

**Output:**  encoded image and decoded image

1. Read an image from the dataset.
2. Convert the image to a grayscale image,.
3. Preprocessed using the following formula and 5x5 kernel:

(1)

Where,

= Processed Image

= Standard Deviation

= th row in the grayscale image

= th column in the grayscale image

1. Perform segmentation on using the following equation:

If pixel then,

,

Otherwise,

, provided maximum pixel value = 255

Where,

= grayscale image

= a pixel of an image in position

= selected threshold value in our algorithm. It is tested after the trial-and-error method.

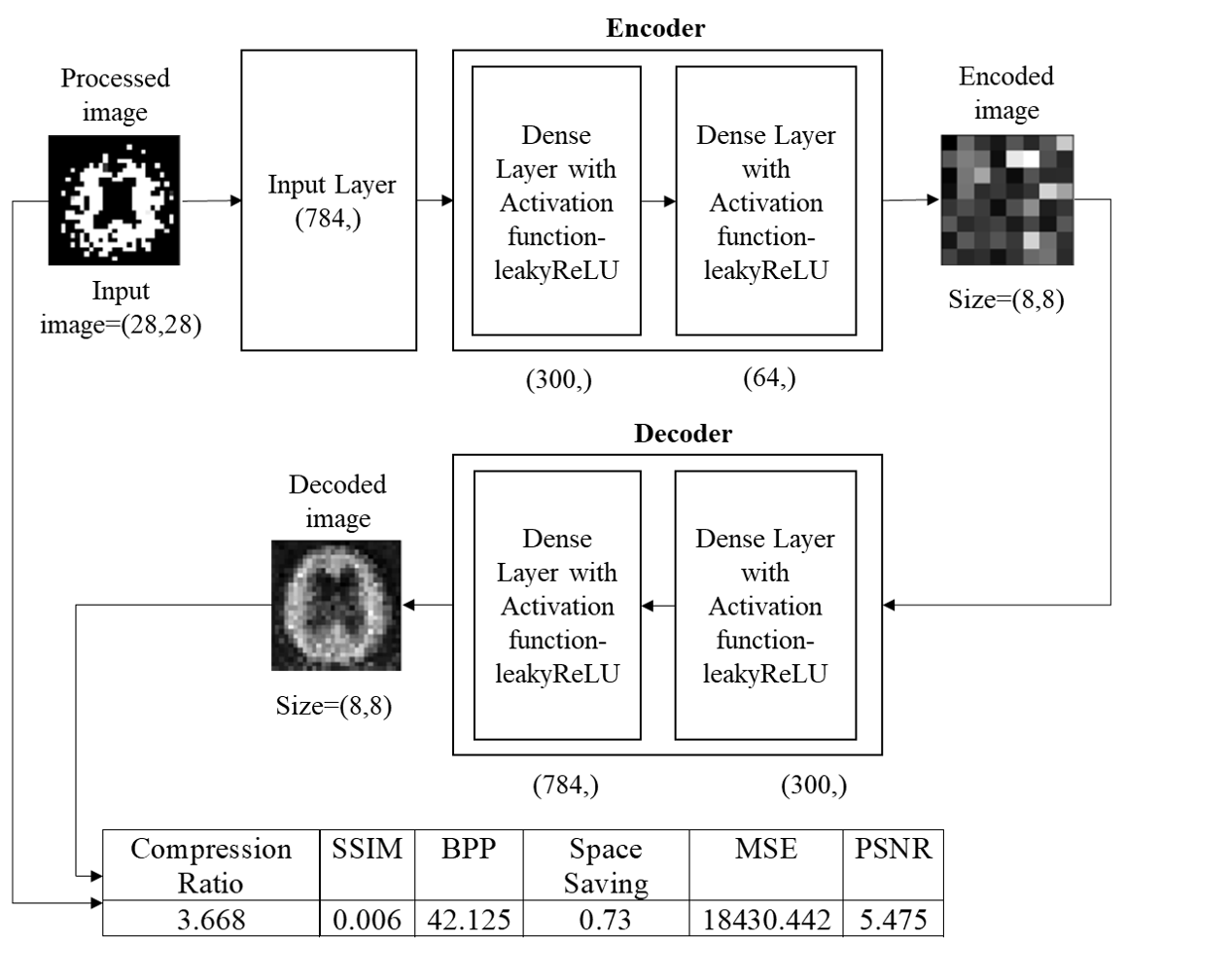
1 = Light

0 = Dark

1. Resize into a 28 X 28 image.
2. The proposed convolutional architecture each has an input x of size 28x28x3 is feed into it. In the convolutional architecture following layers are observed:
   1. Encoder Architecture
   2. One input layer which accepts input of size 784.
   3. One dense layer of output size 300 with an activation leakyReLU function.
   4. One dense layer of output size 64 with an activation leakyReLU function.
   5. Decoder Architecture
      1. One input layer which accepts input of size 64.
      2. One dense layer of output size 300 with an activation leakyReLU function.
      3. One dense layer of output size 784 with an activation leakyReLU function.
3. Segregate the whole dataset [10] into two sections. We select 80% MRI from the dataset [10] for training purposes and 20% MRI from the dataset [10] for testing purposes.
4. We feed the training and testing data into the designed convolutional model architecture to generate result.

**3.2 Model Architecture**

Training and testing phase can be explained with the help of a model architecture as shown in **Fig.2.** the architecture accepts preprocessed MRI image and displays encoded and decoded images along with performance parameters for the same.

~~~~

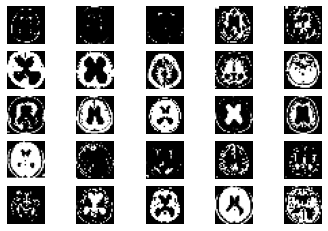
**Fig.2.** Training and testing phase using proposed model architecture

1. **Experimental results**

We consider a custom hydrocephalus dataset, which contains 132 images. The size, colour, and format of images in the dataset are similar in nature, whereas the resolutions of the images are different. The format of the images is ‘.jpg’ by nature.

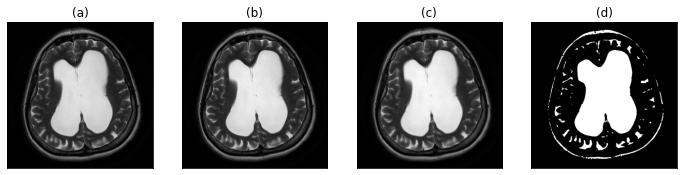
We have applied our algorithm in the python environment, version 3.8, with the hardware configuration of the Intel Core i3 5th Generation processor,4GB DDR3 primary memory (RAM), and an integrated graphics card. Anaconda as a distributor of Python version 3.8 is used. jupyter notebook version 6.3.0 as an open web interface is used as a programming platform for the implementation of our algorithm.

After reading the images from the local machine we have achieved the following results as shown in **Fig. 3.**



**Fig. 3.** Samplesof hydrocephalus dataset

After reading all the images from the dataset each image will pass through the various steps of the algorithm as shown in **Fig. 5.**

****

**Fig. 5.** (a) original image (b) gray-scale image (c) image after applying proposed filtration method (d) proposed segmentation approach for the region of interest detection

We are splitting our dataset [10] into training and testing sets. 80% of the MRIs are used for training and the rest is used for testing. After the execution of our proposed algorithm, we observed that the number of training samples are 104, number of testing samples are 26 whereas training shape values are (104, 28, 28). Our proposed convolutional layered architecture is trained. It is used to train on n number of samples (104 in our case). With the changes of several iterations or epochs, the total time consumed by the algorithm is calculated.

**Table 2:** An example of the training phase.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNo.** | **Epoch No.** | **Time Taken (s)** | **Loss** | **Value Loss** |
| 1. | 1 | 1 | 0.2193 | 0.1604 |
| 2. | 2 | 0 | 0.1407 | 0.1219 |
| 3. | 3 | 0 | 0.1159 | 0.1086 |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
|  |  |  |  |  |
| 18. | 18 | 0 | 0.0517 | 0.0842 |
| 19. | 19 | 0 | 0.0494 | 0.0835 |
| 20. | 20 | 0 | 0.0473 | 0.0831 |

* 1. **Performance parameters**

In this work, the compression techniques use a wide number of performance measures to compute their efficiency and performance.

**Table 3:** Various performance parameters

|  |  |  |
| --- | --- | --- |
| **Serial No**. | **Name of the performance parameter** | **Equation** |
| 1. | Compression Ratio [10][1] | CR = |
| 2. | Mean Square Error [10] | MSE= |
| 3. | Bits per Pixel [10] | CR = |
| 4. | Structure Similarity Index [10] | SSIM = |
| 5. | Correlation coefficient [10] | CC = |
|  | Peak Signal to Noise Ratio [10] | PSNR=20 |
| 7. | Percent rate of distortion [10] | PRD = |
| 8. | Structural Content [10] | SC = |

**Table 4:** Calculation ofperformance parameters for each image

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SNo** | **BPP** | **Compression Ratio** | **MSE** | **SSIM** | **PSNR** | **PRD** | **Structural Content** | **CC** |
| 1. | 338 | 1.538 | 17142.986 | 0 | 5.79 | 99.665% | 10.081 | 0.0040 |
| 2. | 333 | 2.261 | 1421.007 | 0 | 16.605 | 99.55% | 475.408 | 0.0065 |
| 3. | 333 | 1.967 | 3585.557 | 0 | 12.585 | 99.541% | 129.134 | 0.0072 |
| . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . |
| 25. | 341 | 1.757 | 37420.317 | 0 | 2.4 | 99.659% | 4.243 | 0.0020 |
| 26. | 339 | 1.664 | 35073.048 | 0 | 2.681 | 99.625% | 3.669 | 0.0023 |

**Table 4** displays performance parameters of 26 randomly selected original samples from the dataset. At the end, we have found that our proposed algorithm can be used for medical image compression from MRIs. As the data is balanced, we are considering the results to be satisfied. Comparing it with other existing methodologies a satisfactory result is observed as shown in **Table 5.** We can conclude that our proposed methodology overpowers the existing compression algorithms.[1][2][3][5]

**Table 5:** Comparison chart

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Serial No**. | **Name of the compression algorithm** | **Compression ratio** |
| 1. | LZW [1] | 5.319% |
| 2. | Huffman coding [1] | 2.203 % |
| 3. | EZW [2] | 1.6647% |
| 4. | RLE [3] | 1.5766% |
| 5. | Shannon–Fano [3] | 1.9825% |
| 6. | Arithmetic coding [3][5] | 2.2059% |
| **7.** | **Proposed methodology** | **1.4593%** |

1. **Conclusion**

In this paper, we have proposed an Autoencoder algorithm. This algorithm is capable of compressing and storing generated images from medical equipment contained in the Custom hydrocephalus dataset, collected from the web resource. It is responsible for encoding the original image, and decoding the image, and finally comparing the decoded and original image. In addition, we have also generated BPP, Compression ratio, MSE, SSIM, PSNR, PRD, Structural content and CC values, through which we can compare our proposed method with existing methods [1] [2] [3] [5]. The experimental result shows that after applying the proposed and existing methods [1] [2] [3] [5] on the Custom hydrocephalus dataset, our technique is producing a Compression ratio of 1.4593%. This result is considered to be satisfactory, and based on this result we can say that the proposed algorithm overpowers the efficiency of the existing method. Due to high performance, novelty, ease of use, our proposed method is useful to develop any mobile or web applications in the future. Our method can be tested on various medical equipment generated images datasets to identify the generic performance of the proposed method in the future. The performance of our method may be increased by making necessary modifications in the algorithm.

# References

1. Gajendra Sharma, “Analysis of Huffman Coding and Lempel–Ziv–Welch (LZW) Coding as Data Compression Techniques,” International Journal of Scientific Research in Computer Science and Engineering, Vol.8, Issue.1, pp.37-44, 2020.
2. A.M.Raid, W.M.Khedr, M. A. El-dosuky and Wesam Ahmed, “Image compression using embedded zerotree wavelet” Signal & Image Processing : An International Journal (SIPIJ) Vol.5, No.6, December 2014, pp. 33-39, doi : 10.5121/sipij.2014.5603
3. Rahman, Md. A., and Mohamed Hamada. 2019. "Lossless Image Compression Techniques: A State-of-the-Art Survey" Symmetry 11, no. 10: 1274. doi : 10.3390/sym11101274
4. S. S. Yu, M. N. Wernick and N. P. Galatsanos, "Lossless compression of multi-dimensional medical image data using binary-decomposed high-order entropy coding," Proceedings of 1st International Conference on Image Processing, 1994, pp. 351-355 vol.2, doi: 10.1109/ICIP.1994.413590.
5. G. Langdon and J. Rissanen, "Compression of Black-White Images with Arithmetic Coding," in IEEE Transactions on Communications, vol. 29, no. 6, pp. 858-867, June 1981, doi: 10.1109/TCOM.1981.1095052.
6. A. K. Jain, "Image data compression: A review," in Proceedings of the IEEE, vol. 69, no. 3, pp. 349-389, March 1981, doi: 10.1109/PROC.1981.11971.
7. P. Roos, M. A. Viergever, M. C. A. van Dijke and J. H. Peters, "Reversible intraframe compression of medical images," in IEEE Transactions on Medical Imaging, vol. 7, no. 4, pp. 328-336, Dec. 1988, doi: 10.1109/42.14516.
8. Z. Xiao and C. Zheng, "Medical Image Fusion Based on the Structure Similarity Match Measure," 2009 International Conference on Measuring Technology and Mechatronics Automation, 2009, pp. 491-494, doi: 10.1109/ICMTMA.2009.558.
9. Amin Mubarak Alamin Ibrahim\* et al. ,(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH, Volume No.3, Issue No.1, December – January 2015, 1808 – 1812.
10. Pardeep Kumar, Ashish Parmar, Versatile Approaches for Medical Image Compression: A Review, Procedia Computer Science, Volume 167, 2020, Pages 1380-1389, ISSN 1877-0509, doi : 10.1016/j.procs.2020.03.349