Data Compression using SVD Technique and Data Storage and Distribution using AWS

Santhosh B.
Assistant Professor
Telecommunication Department
Dayananda Sagar College of Engineering
Bengaluru, India

Supreeth BI.
Telecommunication Department
Dayananda Sagar College of Engineering
Bengaluru, India

Sharath R.
Telecommunication Department
Dayananda Sagar College of Engineering
Bengaluru, India

Varun Hebbar B. C..
Telecommunication Department
Dayananda Sagar College of Engineering
Bengaluru, India

Shivani Sinha
Telecommunication Department
Dayananda Sagar College of Engineering
Bengaluru, India

***Abstract*—Storage and data trafficking have grown a great deal over the past decade. Therefore, there is a need to reduce the size of the data for better speed and proper utilization of the bandwidth. There are compression techniques that happen online as well as some happen statically. The online based compression is in greater demand, since an online platform is much easier and faster. Static methods have a requisition of the entire file to be present at the time of transmission, while online compressions don't. Compressions make use of an algorithm to effectively shrink the data. Data compression can be used on any form of data. Be it images, or videos to simple text files, we can compress the data. Few examples are: ZIP, RAR, GZIP, PNG, JPEG, WAV, etc. There are two methods of compression, a lossless compression and a lossy compression. Few algorithms that are most widely used in lossless compression is Huffman encoding and LZ77 encoding. In this paper, Singular Value Decomposition is experimented on to achieve best results. The reconstructed file is saved on AWS for best accessibility.**

Keywords—Singular value decomposition, Compression, reconstruction, AWS, Eigenvalues, S3, IAM.

I. INTRODUCTION (*HEADING 1*)

Data compression means reducing the size of a data file by removing some number of bits in a unique and decodable manner. Data can be represented in the forms of bits. The bits can be removed in a convenient manner that would not affect the data's meaning. The removed bits may or maynot be added back based on the application. Reduction of the size of data file by deleting bits is called compression; restoring the bits to the original format is called decompression. Compression and decompression are normally interlinked in a mathematical method, either by formulas or some form of an algorithm.

Data compression for different file formats are different again. A compression used for text file cannot easily be adopted to use for image, since image is 2 dimensional and text is one dimensional. Compression also depends on the arrangement of data. Streams of redundant bits compress highly while a lot of shuffled bits may even increase the file size. Compression needs to be done smartly.

Sometimes it is the same if we reduce the size or not, and sometimes, worse. For example, consider the data aababbabab. Once compression is done, the data would be a2b1a1b2a1b1a1b1. The data was 10 bytes before compression and is 16 bytes post compression. This is larger than the original file.

Need for compression right now in the world is self explanatory. With appropriate compression techniques, the data can be transmitted with lesser friction over a lower quality internet. After all, faster speed does not essentially mean better internet or better reception. In the end, it all depends on how fast the information was able to go from one end to other, while still maintaining the quality. 4k quality videos can be streamed on 3G internet if proper compression techniques are adopted. Or, in other words, one can say that based on the application if the compression would be able to adapt itself, the internet even with current standards would seem faster.

There are basically two standards in compression: Lossy and lossless compression. In lossless compression, after a file is compressed and decompressed, one would get back the exact size of the original file. This is used in places where loss of bits in the original file would change the meaning of the entire file. Lossy compression removes the redundant bits in such a manner that on the whole, the meaning of the original file would not change. Even when the file is decompressed again, one would not get back the original file size. Taking image compression for example, PNG and GIF are lossless compression technique while JPEG is a lossy compression technique.

There are certain drawbacks for compression as well. The intensive calculation and processing may affect the performance of the hardware used. So, appropriately the compression techniques have to be used.

Linux distros and Microsoft employ different compression algorithms. First off, it depends on the file type used in the systems. In Linux distros, the file types are Reiser4, ZFS and btrfs, while in Windows it is NTFS. It is wiser to use compression on files that are not normally accessed since compressing and decompressing a file every time it is used is a very tedious and intensive process.

The popular software like WinZip employs compression on files while zipping them into an archive. BZIP2 and GZIP also are employed widely for compression. There are many more software that employ compression.

Compression techniques employ mathematical formulas and algorithms to compress a file in a logical and uniquely decodable manner. In the modern day, there are so many rising techniques that has potential to be better if used smartly. One such technique is Singular Value

Decomposition, in short SVD. SVD employs a method of smartly removing unwanted data and reconstructing the data with only important elements based on the application required.

For storage, we know that the current hardware is not good enough. Cloud computing is the leading storage right now and there are certain concerns about cloud. Cloud's security is one of the biggest concern. Few leading platforms are Amazon Web Services (AWS) and Microsoft Azure. For the purpose chosen here, AWS seemed to be a wiser choice as it has a more cost effective scheme. Though Microsoft Azure's engine is more powerful, AWS is sufficient for storage purpose. AWS's S3 service and IAM service is chosen as it is the simplest and cost effective choice.

In this paper, we are choosing to use medical images (Dicom) for compression. Though this compression technique and storage idea can be used generally, here, the application that seemed best suitable is medical purposes.

II. LITERATURE SURVEY

Lossless pressure is a class of information pressure calculations that enables the first to be actually recreated again from the packed information at the less than desirable end. This technique is utilized in situations where it is significant that the first and the decompressed information ought to be indistinguishable for example there ought to be any deviations from the first information. Lossless pressure calculation has two stages: the first is to create measurable model for the information, and in the second step we utilize this model to outline information to bit successions such that most much of the time experienced information will deliver shorter yield than "unlikely" information. There are two essential methods for building factual models: in static model the information is examined and model is built and this model is put away with the packed information. This methodology is easier yet it is too costly to even consider storing and can't be utilized for heterogeneous information types as just single model is utilized to store the information.
Though, the versatile models powerfully update the model as the information is packed. Encoders and decoders start with unimportant a model which yields poor pressure of introductory information, however as they get familiar with the information, execution improves.
Some of the lossless compression methods are:

1. Run-length encoding (RLE): - provides good compression of the data containing lots of runs of the same value. 2. Huffman coding: - it is a particular type of optimal prefix code. 3. Prediction by partial matching (PPM): - used for compressing plain text 4. Lempel-Ziv compression (LZ77 and LZ78): - dictionary based algorithm which forms the basis for other algorithms.

Lossy compression or irreversible compression is the class of data encoding methods that uses inexact approximations and partial data discarding to represent the content of the data. This method is applied to reduce the data size for storing, handling and transmitting content. There are two basic lossy compression methods:

- Lossy transform codecs: - tests of picture or sound are taken which will be separated into little fragments. These fragments are changed into another premise space, and quantized. The subsequent quantized information esteems are then entropy coded.
- Lossy predictive codecs: - past or consequent decoded information is utilized to anticipate the present sound example or picture outline. The blunder between the anticipated information and the genuine information, together with extra additional data required to replicate the forecast is quantized and coded. Pressure proportion of lossy pressure is far better than that of lossless pressure. Recordings can be compacted in the proportion 100:1 with little perceivability misfortune. Sound can be compacted at 10:1 with impalpable loss of value and pictures are packed at the proportion 10:1.

Some of the lossy compression methods are: -

1. Better Portable Graphics (BPG). 2. Cartesian Perpetual Compression (CPC). 3. Fractal Compression. 4. JPEG (Joint Photographic Experts Group).

A lossless LZ77 data compression technique is used to obtain the compressed data by eliminating the redundant data (generally represented in binary format) so that the data can be stored in SSD's effectively. It employs test window with look ahead buffer which acts as dictionary. Here Novel method is employed where the pointers are effectively selected and encoded so that longest matched string in the dictionary is compressed which results in greater compression ratio. In this method the single window is categorized into search buffer and look ahead buffer. Search buffer contains recently encoded sequence and in look ahead buffer the next sequence to be encoded is stored. The algorithm checks for consecutive symbols in both the windows and then encoder encodes the longest match in with triple (o,l,c) where 'o' stands for offset, 'l' for length of the match and 'c' for code length. Using the same triple, the encoded data will be decoded at the receiving end. LZ77 decoders expand the compressed file by expanding the copy items in the dictionary into larger strings. By adopting this algorithm technique, we obtain the compression ratio which is one-third of the original file.

Data compression works only for text, voices or images data. If data consists of binary files, numbers or executable program files then no known algorithm can compress such data. Some algorithms require less memory where some require more.

Shannon algorithm: Input data is first encoded into bits for transmission, which the receiver then uses to reconstruct original input data.

Autosophy algorithm: Both receiver and transmission have prior knowledge of the data to be communicated.

Huffman algorithm: it is the first lossless data compression based on Shannon library containing prior knowledge of data in the form of relative frequency. Compression requires a memory, look-up table or library which increase system and hardware cost.

Ziv-Zempel codes: Compression method known as LZ-1 or LZ-77 code. It is based on incorrect assumptions and thus delivers relatively low data compression. In this method, transmitted data is accumulated in a character shift register. During subsequent transmissions the shift register is searched to find longest matching string from previous transmitted data. If matching string found, then it is encoded into a string start address and a string length code. Receiver has its own identical shift register or library both libraries remain identical all times.

Self-learning Autosophy data trees: The fundamental idea can be envisioned as a library of words put away in a PC. Every content word is recognized by a one of a kind location code. Both transmitter and beneficiary have indistinguishable library. Transmitter looks library for that word and transmits its location code. Recipient utilizes address code to recover information.

Reversed Leading Bits Coding and Huffman Coding together are used to get a better compression ratio. In this technique, firstly, each don't-care bit in the original test data is filled with the value of the bit before it. Secondly, the test data generated without any don't-care bit is divided into codeword segment the and codeword segment is matched with RLBC pattern and symbols are generated accordingly. The break codes are asserted looking at the different leading bit value of the two neighboring patterns. The compression efficiency can be improved by keeping the size of the pattern and the number of symbols appropriate. Finally, Huffman Coding is achieved efficiently with high compression ratio. The compressed data consists of encoded data and break codes. Using these break codes, we can convert the encoded symbols back to original test data. Comparing with the different test data compression techniques,

RLBC-HC technique helps us in achieving a higher compression ratio and the compression efficiency is improved consequently.

For storing and transmission of data, compression techniques that use online platform are becoming the popular choice. There are several static methods of compression but they are time consuming and require the entire file before compression whereas, the online platform requires a one pass for both encoding as well as decoding. Dictionary based compression makes the file much smaller and the bandwidth is utilized more effectively. In the paper, they make a dictionary of the incoming data (first come first serve). First block of data is compressed using adaptive tree method and a dictionary is created simultaneously. A unique code of 9 bits is generated and given to each string generated by LZW method and forward move basis method. The frequency of occurrence of strings is vital in this technique for the efficiency to improve. A median is used to store data while previous data is being transmitted after compression. This allows us to create block of data of particular size and apply compression to it.

III. Proposed Method

Straight Polynomial math is an investigation of grids. A network is a table that holds information, stores numbers in segments and lines. Straight Variable based math at that point takes these lattices and controls them which enable specialists to break down vast parts of information. SVD considers one of these extensive segments of information for picture pressure in which every pixel can be spoken to as a number, and the sections and columns of the framework hold the situation of that esteem in respect to where it is on the picture. Single Value Decomposition (SVD). is, where there are a few significant qualities that mathematicians and specialists can get from them that can be utilized to clarify what the information speaks to, arrange the information into families, and they can likewise control grids by maneuvering them separated into qualities that are simpler to work with, at that point sewing those qualities back together toward the finish of the calculation to get some kind of result. The SVD is one such calculation which mathematicians and specialists find incredibly valuable. SVD parts a network into three significant sub grids to speak to the information. Given the lattice A, where the span of an is $m \times n$ where m speaks to the quantity of lines in the grid, and n speaks to the quantity of sections, A can be separated into three sub networks $A=USV$, where U is of size $m \times m$, S is of size $m \times n$ and is inclining, and VT is of size $n \times n$. It is required for lattice duplication that the extent of the sections of the principal network must coordinate with the measure of the lines of the second grid. When you increase a grid of size
$a \times b$ and a framework of size $b \times c$, the subsequent lattice will yield a network of size $a \times c$. So, abstracting the matrices into their size components, and by multiplying them will yield a matrix of the same size:
$m \times n=[(m \times m)(m \times n)](n \times n)$ $m \times n=(m \times n)(n \times n)$ $m \times n=(m \times n)$

The significance of these grids "$USVT$" are that the information will be masterminded so that the most significant information is put away on the top. U is a network that holds significant data about the lines of the lattice, and the most significant data about the grid is put away on the main section. VT is a grid that holds significant data about the sections of every network, and the most significant data about the framework is put away on the primary line. S is an askew network which will just have at most "m" significant qualities, the remainder of the grid being zero. Since the significant quantities of this grid are just put away on the slanting, we will overlook this for size examination. The significance of these grids "$USVT$" are that the information will be masterminded so that the most significant information is put away on the top. U is a network that holds significant data about the lines of the lattice, and the most significant data about the matrix is put away on the main section. VT is a grid that holds significant data about the sections of every network, and the most significant data about the framework is put away on the primary line. S is an askew network which will just have at most "m" significant qualities, the remainder of the matrix being zero. Since the significant quantities of this grid are just put away on the slanting, we will overlook this for size examination.

$U'S'V^{T}{}'= [(m \times 1)\ (1 \times 1)]\ (1 \times n) = (m \times 1)\ (1 \times n) = (m \times n)$

The resulting computation is the same size as the original matrix. This resulting matrix, which we will call A', is a good approximation of the original matrix A. For an even closer approximation, the next column of U and the next row of V^{T}.

The screen of the PC is a mysterious gadget. The shading white on the screen, isn't really white, and something very similar for the shading yellow. There is quite white or yellow color in the screen. It is a blend of the hues red, green, and blue shown by amazingly little pixels on your screen. These pixels are shown in a matrix like example, and the immersion of every pixel fools your mind into believing it's an alternate shading totally when taken a gander at from a separation.
The red, green, and blue pixels go in immersion on a size of 0 to 255; 0 being totally off, and 255 being totally on. They can likewise be written in hexadecimal organization like #F5C78A for instance. In hexadecimal, an is the esteem 10, and F is the esteem 15, in this way 0E = 14 and A0 = 16. The initial two numbers in this series of numbers speaks to the red esteem, the following two speaking to the green esteem, and the last two speaking to the blue esteem. To place reference into what these are doing, here are some simple shading precedents:
#000000 = Dark, #FFFFFF = White, #A0A0A0 = Dim, #FF0000 = Red, #00FF00 = Green #0000FF= Blue.
Due to a pixel's lattice like nature on the screen, an image can

really be spoken to as information in a framework. Considering a dark scale picture for this moment. To make a picture dim, the qualities for red, green, and blue should be the equivalent. In this manner, to speak to an estimation of pixel from 0 through 255 (in hexadecimal 00 through FF), and afterward rehashing that esteem over the red, green, and blue immersion to get the relating shade of dim. Considering a dark scale picture that is 120 × 120 pixels in measurement. Every one of those pixels can be spoken to in a lattice that is 120 × 120, where the qualities in the framework extend from 0 to 255. In the event that to store that picture, would need to monitor precisely 120 × 120 numbers or 14,400 distinctive pixel esteems. It doesn't appear to be a great deal, however in the event that the picture as your work area foundation is a picture of 1280 × 1024 in which it would need to store 1,310,720 diverse pixel esteems and that is on the off chance that it was a dim scale picture, on the off chance that it was shaded, it would be triple that, monitoring 3,932,160 unique numbers, which is around one of those numbers likening to a byte on your PC that approaches 1.25MB for a dark scale picture or 3.75MB for a hued picture. Simply envision how rapidly a motion picture would increment in size in the event that it was refreshing at the rate of 30-60 outlines for every second.

By using and computing SVD algorithm on this image memory and space can be saved. In an image that is 120 × 120 pixels would look really quite good with only 12 modes of precision using the SVD computation.

Considering example mentioned above, "The reason why the SVD is computed is because it can be used as the first components of these matrices to give a close approximation of what the actual matrix looks like."

Then the calculation of the first components of these matrices can be done by taking the first column of *U* and multiplying it by the first row of V^T. This resulted in a matrix with the dimensions of the original matrix *A*.

$U'S'VT' = [(m \times 1)\ (1 \times 1)]\ (1 \times n) = (m \times 1)\ (1 \times n)$
$= (m \times n)$

Modes are how many columns of the matrix *U* that is required to use and how many rows of the matrix V^T it is required to calculate the specified level of precision.

Therefore, if a matrix 100 × 100, and a level of precision of 10 modes is used, the matrices obtained are:

$U' = (120 \times 12)$, $S' = (12 \times 12)$, $V^T{}' = (12 \times 120)$

So now to follow just 3,024 distinct numbers rather than 14,400 which significantly expands the capacity of memory. 'S' it is a corner to corner grid with qualities along the slanting and zeros wherever else. Subsequently, to speak to S as just being the initial ten estimations of σ, and sparing just those qualities in memory, remaking the lattice when opening the document, and S goes from size 120 to measure 12. Be that as it may, there may not be the same number of σ in the calculation as the measure of the grid, in a 5 × 5 lattice, you can have at most five

σ's, yet additionally it can have as meager as one, the remainder of the qualities on the corner to corner likewise being zero. So truly $\sigma \leq$ # modes, which will be so little in any case, it tends to be consulted amid calculation.

IV. METHODOLOGY

METHODOLOGY FOR COMPRESSION OF DICOM IMAGES:

The image to be compressed is taken. This image ideally can be represented in the form of a matrix. Let this matrix be D of size nxp. This matrix is taken and is represented in the form of A = USV', where U and V are square matrices of the size nxn and pxp. S is a diagonal matrix of size nxp. Compute U and V according to SVD definition, i.e., by finding the eigen matrix and then eigenvector and then ultimately the eigenvalues using the eigenvectors. The result is actually a matrix that helps us determine the region of interest (diagonal elements), while the rest of the elements are not of that much importance. This image is nothing but the compressed form of the previous original image. • This image is reconstructed using blocks of size N (N=1,2,4, 8,….). The image is divided into N parts. The values inside these N images are pixelated and are rearranged with respect to the diagonal matrix defined by S. N is the value that defines the success rate of compression. N has to be adjusted in such a way that it does not compromise on quality for space or space for quality. It should be ideal based upon the application it will be used for. After that, parameters such as MSE, PSNR and Compression ratio are calculated with respect to their formulas. Graph of error rate of compressed image vs original image is plotted. This gives us good insight on what size of the image can be used ideally for what purpose. The image is then moved on to the next step.

METHODOLOGY FOR CLOUD STORAGE USING AMAZON WEB SERVICES:

S3:

Log into the root user account by providing the login credentials. After logging in, go to services >> choose S3. It will guide you to the S3 service. Here, you can create a bucket by clicking on "Create Bucket". Then, enter the requisite details. In the "Set Permissions" tab, untick all the conditions there since we need public access to access the elements inside the bucket. Finish the creation of the Bucket and click on it to select the Bucket. Add the required dicom image.

IAM:

Go to "IAM" in services tab. Choose "Users" and then click on "Add user". Tick on both the conditions below to create Programmatic access and AWS management console access. This is what adds security to the user end. Let console password be auto-generated so the user can change it himself when he logs in for the first time. Finish the creation of new user. Go to the newly created user and there, click on "Add inline policy". Then, choose the service as S3. Provide complete access to the IAM user. Then, choose the specific bucket the IAM user can have access to. Along with the specified object. (Add ARNs) Then choose "Review policy" and then click on "Create policy".

IAM user:

The IAM user will be mailed a CSV sheet by the root user. This contains his one-time login credentials. Once logging in, he has to change whatever credentials he needs to change. Re-login and then go to services to choose S3. The bucket access will be provided specific to that IAM user. The IAM user can take his dicom files from there or add any

documents into the bucket. This bucket is securely shared between the root user (hospital) and the end user (patient end).

V. RESULTS

Block Size N	Compression Ratio	PSNR [dB]	MSE [dB]
2	27.49	-24.86	57.26
4	13.713	-20.69	47.64
8	6.82	-18.06	41.6
16	3.38	-16.588	38.19
32	1.66	-14.84	34.18
64	0.8	-12.06	27.8
128	0.3768	-6.7	15.40
256	0.1675	10.56	-24.33

Fig: Original image

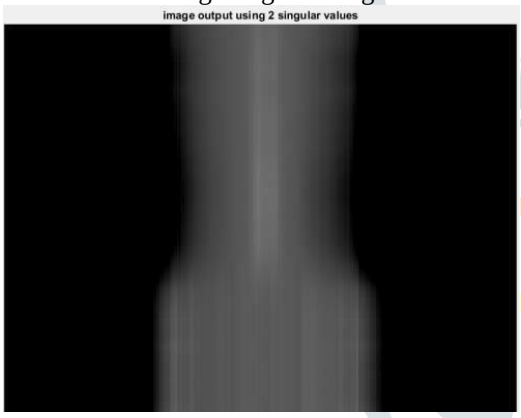

Fig: For N=2

Fig: For N=128

Fig: For N=256

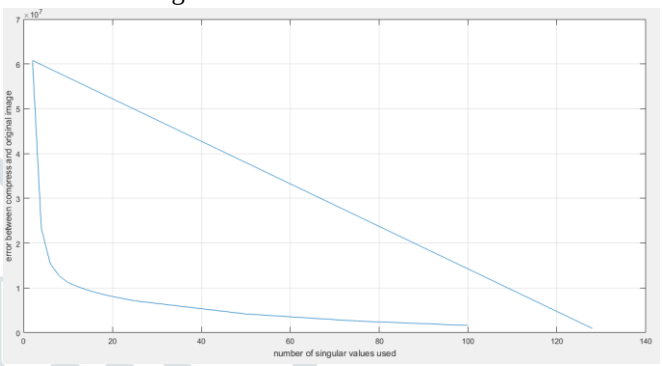

Fig: Plot of Error between Original Image and Compressed Image

Conclusions that can be drawn from the above Summary Table: -

- N is the value which is used to represent the number of Eigen values which is used in the reconstruction of the compressed image.
- If the value of N is small, more is the compression ratio, so that it requires less storage space to store the image but the image quality will be deteriorated.
- On the other hand, if value of N increases, image quality significantly increases but more storage space is required to store the compressed image.
- Therefore, it is necessary to maintain balance between image quality and storage space for good image compression. From the above table we can concur that for N=128 MSE is approximately 16 dB, a balance between quality and storage can be obtained.
- Choice of N depends upon the application.

VI. FUTURE SCOPE

SVD is one of the best algorithms present right now with respect to efficiency, robustness and multiple other factors. They provide best results with very less processing time and the hardware required does not need to be high end. This method is very underrated. It is a comparatively new technique and has a lot of room for improvement.

Amazon web services or AWS in short, is one of the leading big guns in the cloud industry. It almost single handedly defeated most other cloud engines in terms of power and efficiency. AWS stands as the most cost-efficient cloud engine. It is fairly secure as well. Combining these two platforms gave a good result in terms of cost and robust nature all while maintaining quality of the file.

This program and cloud methodology can be integrated together to form a platform that is most needed for medical purposes right now. Consider a scenario where a person is travelling and abruptly falls ill. The person will be treated at a hospital and the doctors require some recent medical documents of that patient. The patient has to call his relatives who are staying at his hometown and ask them to visit his personal home physician on his behalf and get an authorized signature along with the medical documents and will have to courier the documents to him. If there is no time, then the doctors will advise him to retake all scans. This would just mean extra cost for him.

Instead if he had this application, the hospitals can be the root user, where they can create IAM accounts for their patient and share that account with the user. The hospital can drop in his scans in the S3 bucket and the patient can access as well as drop in other files in case he gets scans and documents from a secondary hospital. This would save time, cost and save everyone from a long and tedious process.

REFERENCES

[1] D.R. Vasanthi, R.Anusha, B.K.Vinay, "Implementation of Robust Compression Technique using LZ77 Algorithm on Tensilica's Xtensa Processor", 2016.

[2] Klaus Holtz, Eric Holtz, "Losless Data Compression Techniques", Proceedings of WESCON'94, 1994.

[3] Haoqi Ren, "A data compression technique based on Reversed Leading Bits Coding and Huffman Coding(RLBC-HC)", 2015 10th International Conference on Communications and Networking in China (ChinaCom), 2015.

[4] Deepa Ray and Seema Gupta, "Adaptive Lossless Forward Move Dictionary based Compression" 2015.

[5] MedicineNet.com, "X-ray definition and facts", 2018. [Online].
Available: https://www.medicinenet.com/x-rays/article.htm#x-ray_defi nition_and_facts.

[6] Brady Mathews, "Image Compression using Singular Value Decomposition(SVD)", December 2014.

[7] Boqiang Liu, Minghui Zhu, Zhenwang Zhang, Cong Yin, Zhongguo Liu and Jason Gu, "Medical Image Conversion with DICOM", 2007 Canadian Conference on Electrical and Computer Engineering, ISSN 0840-7789, April 2007.

[8] Jaeger, T. Lin, and JM. Grimes, "Cloud computing and information policy: computing in a policy cloud", Journal of Information Technology & Politics, vol 5, no. 3, pp. 269-283, 2008.

[9] S. Hludov, C. Meinel, "DICOM- Image Compression", Proceedings 12th IEEE Symposium on Computer-Based Medical Systems (Cat. No.99CB36365), ISSN 1063-7125, June 1999.

[10] WD Bidgood Jr, Hori Sc, FW Prior, Syckle Van De "Understanding and using DICOM, the data interchange standard for biomedical imaging", Journal American Medical Informatics Association, vol. 4, no. 3, pp. 199-212, 1997.

[11] Gunjanbhai Patel, "DICOM Medical Image Management the Challenges and Solutions: Cloud as a Service (CaaS)", 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), July 2012.

[12] Srinivasa Rao, Nageswara Rao, E Kusuma Kumari, "Cloud Computing: An Overview", Journal of Theoretical and Applied Information Technology, vol. 9, no. 1, pp. 72-76, 2009.

[13] Steve G. Langer, "Challenges for data storage in medical imaging research", Journal of Digital Imaging, vol. 242, pp. 203-207, April 2011.

[14] S. Hamm, "How cloud computing will change business", Business Week, June 2009. [Online]. Available: http://www.businessweek.com.magazine/ content/09 241b413504294227 0.htm.