BIG IMAGE DATA PROCESSING IN CLOUD AND DISTRIBUTED ENVIRONMENTS

Submitted in partial fulfilment of the requirements of the degree of
(Bachelor of Technology)
by

Karthik Kothuri (177230)
Mounika Kukudala (177231)
Shashikar Anthoniraj (177259)

Supervisor (s):

Dr. U.S.N Raju

Assistant Professor, Department of CSE, NIT Warangal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

(2021)

APPROVAL SHEET

This Project Work entitled **Big Image data processing in cloud and distributed environments** by **Karthik Kothuri, Mounika Kukudala, Shashikar Anthoniraj** is approved for the degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology Warangal during the year 2020-2021

Examiners

Supervisor (s)

Dr. U.S.N Raju,
Assistant Professor,
CSE Department

Chairman

Dr. P. Radha Krishna
Head of Department, CSE
NIT Warangal.

Date:

Place:

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)
Karthik Kothuri
177230
Date:

(Signature)
Mounika Kukudala
177231
Date:

(Signature)
Shashikar Anthoniraj
177259
Date:

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **“Big Image Data Processing in Distributed and Cloud Environments”** is a bonafide record of work carried out by Karthik Kothuri (177230), Mounika Kukudala (177231) ,and Shashikar Anthoniraj (177259), submitted to the faculty of Computer Science and Engineering Department in fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology, Warangal during the academic year 2020-2021.

Dr. U.S.N Raju	Dr. P. Radha Krishna
Project Guide	Head of Department
Department of CSE	Department of CSE
NIT Warangal	NIT Warangal

ACKNOWLEDGEMENT

We consider it a great privilege to express our deep gratitude to many respected personalities who guided, inspired, and helped us in the successful completion of our project.

We would like to express our deepest gratitude to our guide, **Dr. U.S.N Raju,** Department of Computer Science and Engineering, National Institute of Technology, Warangal, for their constant supervision, guidance, suggestions, and invaluable encouragement during the project. They have been a constant source of inspiration and helped in each stage.

We are grateful to **Dr. P. Radha Krishna**, Head of Department, Computer Science and Engineering, National Institute of Technology, Warangal for his moral support to carry out this project.

We are very thankful to the project evaluation committee for the strenuous efforts to evaluate our project.

We wish to thank all the staff members of the department for their kind cooperation and support they have given throughout our project work. We are also thankful to all our friends who have given their valuable suggestions and help in all stages of the development of the project.

Karthik Kothuri (177230)

Mounika Kukudala (177231)

Shashikar Anthoniraj (177259)

ABSTRACT

Distributed Computing is a technique of using distributed systems to solve a computational task, which otherwise would not be feasible by using a single system. A distributed system is a set of computers in a network which communicate with each other and coordinate together to complete the task.

Big Data is a term used to describe data which is very huge and cannot be fit inside of a single system. With the ever growing use of computers to solve various tasks with various sizes and scale, there is a huge amount of data being generated.And as such we need a way to process such huge amounts of data.This is where Distributed Computing comes into the picture. Distributed computing helps in processing such data.

Image processing is a field of computer science which deals with algorithms and methods used to process images. Today image processing is used in many fields such as health and research, industries, various organizations etc., on a day to day basis.

In this paper we combine all the above mentioned concepts to create and analyze various Big Image processing methods in various sub-fields of image processing i.e, Image compression,Content based Image retrieval etc .We do this by using various industry standard Distributed Computing tools such as SPARK, HADOOP, etc. We also compare and contrast our results using different cluster configurations and also cloud technologies such as Microsoft Azure.

By the end of the project we were successful in compressing and decompressing **more than 200 Million images**.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 IMAGE PROCESSING

Image processing, in the contemporary domain, is now emerging as a novel and an innovative space in computing research and applications. Today, the discipline of "computer science" may be termed as "image science", because in every aspect of computer application, either science or humanities or management, image processing plays a vital role in varied ways. It is broadly now used in all the industries, organizations, administrative divisions; various social organizations, economic/business institutions, healthcare, defense and so on. Image processing takes images as input and image processing techniques are used to process the images and the output is modified images, video, or collection of text, or features of the images. Images are taken as input and the properties of the images are changed to enhance the image or features are extracted to make them less complex to study.
There are many existing algorithms or techniques to process images for various purposes. Even a small operation such as complementing all pixel values of a binary image is considered as an Image processing technique until it lets us analyse the image more clearly.

1.2 BIG IMAGE DATA

Since usage of the web has increased a lot in recent years,large amounts of Images are being generated every day on various web platforms. The amount of data is so huge that it can't be processed on a single system,which is why it's called 'Big Image Data'. Big Image data can be stored on a distributed system and processed accordingly. Every single domain in the world is dealing with the problem of Big Image data, which has been considered an important enough research topic recently .Big Image data has successfully become a central theme applied to large-scale computing problems in modern years.We have used various popular Datasets and replicated them in large number inorder to produce Big Image Data.

1.3 IMPORTANCE OF DISTRIBUTED COMPUTING

Distributed computing is a model in which components of a software system are shared among multiple computers. Even though the components are spread out across multiple computers, they are run as one system.While there is some complexity in this multi-computer model, there are greater benefits around. This is done in order to improve efficiency and performance. Distributed computing often brings down the execution time of algorithms because of their ability to distribute independent tasks among groups of computers.Distributed computing offers advantages in scalability (through a "scale-out architecture"), performance (via parallelism), resilience (via redundancy), and cost-effectiveness (through the use of low-cost, commodity hardware).The current study aims at analysing execution times of various image processing algorithms on Distributed systems with different configurations.

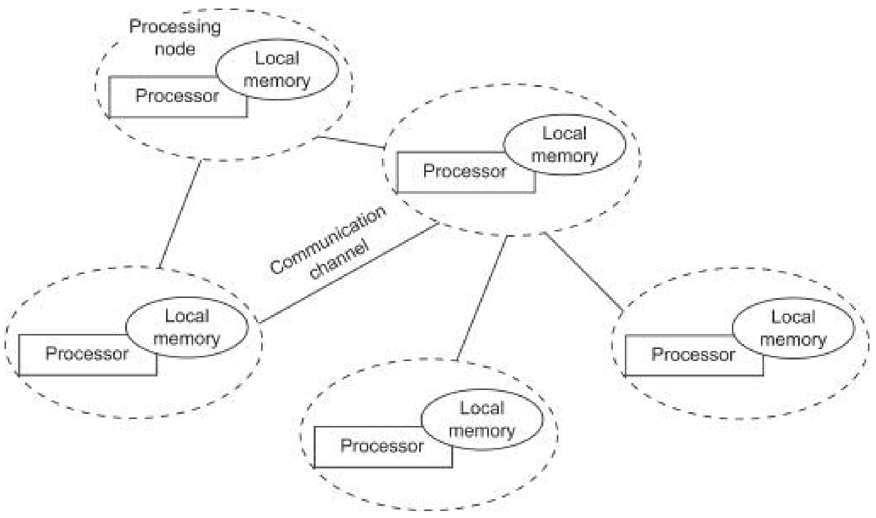


Fig 1.1 shows architecture of distributed systems

1.4 BIG DATA TECHNOLOGIES

Big data technology and Hadoop is a big buzzword as it might sound. As there has been a huge increase in the data and information domain from every industry and domain, it becomes very important to establish and introduce an efficient technique that takes care of all the needs and requirements of clients and big industries which are responsible for data generation. Earlier the data was being handled by normal programming languages and simple structured query language but now these systems and tools don't seem to do much in case of big data.
Big Data Technology can be defined as a Software-Utility that is designed to Analyse, Process and Extract the information from extremely complex and large data sets which the Traditional Data Processing Software could never deal with. We are dealing with two kinds of Big Data technologies in our project, they are Hadoop and Apache Spark which are based on map reduce paradigm.

1.4.1 HADOOP ECOSYSTEM

Hadoop Framework was developed to store and process data with a simple programming model in a distributed data processing environment. The data present on different high-speed and low-expense machines can be stored and analyzed.

This is based on map-reduce architecture and helps in the processing of batch related jobs and process batch information, where the data is processed in parallel with others.MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, the reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers.

1.4.2 APACHE SPARK

Apache Spark is a lightning-fast cluster computing designed for fast computation.The in-memory computing technique is what makes it different from other tools and components and supports a wide variety of applications. It was built on top of Hadoop MapReduce and it extends the MapReduce model to efficiently use more types of computations which includes Interactive Queries and Stream Processing.

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format. Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations.

1.5 CLOUD TECHNOLOGIES

A Cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications. Workloads can be deployed and scaled out quickly through rapid provisioning of Virtual Machines. Virtualization of server resources has enabled cost effectiveness and allowed cloud systems to leverage low costs to benefit both users and providers.
Traditional distributed computing systems provided for on-premise computing and were owned and operated by autonomous administrative domains (e.g. a company).These traditional systems encountered performance bottlenecks, constant system maintenance, poor server (and other resource) utilization, and increasing costs associated with hardware/ software upgrades.Cloud computing as an on-demand computing paradigm resolves or relieves many of these problems.
In this report, Microsoft Azure has been used as a cloud computing service provider. Microsoft Azure is an example of a public cloud. Public clouds are owned and operated by third-party cloud service providers, which deliver their computing resources like servers and storage over the Internet. With a public cloud, all hardware, software and other supporting infrastructure is owned and managed by the cloud provider. You access these services and manage your account using a web browser.

CHAPTER 2

PROBLEM STATEMENT

Image Processing or Digital Image Processing is a technique to improve image quality by applying mathematical operations. Image processing techniques are in demand recently because a vast number of Industries are using Big Image data to process and analyse the information and put it into their business use.
We have selected Image compression as one of the image processing techniques for this project since Image compression improves efficiency in storing and transmission of images. For Example, with upcoming satellites like GISAT that will be downlinking data 24x7 to the tune of 5-6TB per day, data and metadata storage, cataloguing and search becomes a daunting challenge.Distributed computing solves this problem by bringing down the time for processing the images.
With information technology developing rapidly,the variety and quantity of image data is increasing fast.The problem of retrieving desired images among massive image storage has become a crucial task. The paper by Jing Zhang et al. establishes a Distributed Image Retrieval System (DIRS), in which images are retrieved in a content-based way, and the retrieval among massive image data storage is speeded up by utilizing MapReduce distributed computing model. We are trying to analyse performance of content based image retrieval on distributed systems using various image processing techniques such as local binary patterns,uniform local binary pattern,interchannel voting.
Among various ANN's Back propagation neural network (BPNN) is one of the most popular algorithms due to its sensational function approximation and generalization abilities.However, BPNN needs to process huge amount of data through large number of layers, so it is both computationally intensive and data intensive which impacts it's performance significantly. The paper by Yang Liu et al. proposes an efficient way of implementing BPNN algorithm by parallelizing back propagation among clusters of nodes based on data separation.In the current study, one of our targets is to successfully execute deep neural networks in a distributed manner.

CHAPTER 3

IMAGE COMPRESSION

3.1 LZW(Lempel-Ziv-Welch) AlGORITHM

3.1.1 Definition

The LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix's 'compress' command, among other uses. It is lossless, meaning no data is lost when compressing. The algorithm is simple to implement and has the potential for very high throughput in hardware implementations. It is the algorithm of the widely used Unix file compression utility compress, and is used in the GIF image format.The Idea relies on recurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility. It is the basis of many PC utilities that claim to "double the capacity of your hard drive".

3.1.2 How does it work?

LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression.Characteristic features of LZW includes,

- LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file.
- When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.
- As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table.
- Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

3.1.3 Implementation

The idea of the compression algorithm is the following: as the input data is being processed, a dictionary keeps a correspondence between the longest encountered words and a list of code values. The words are replaced by their corresponding codes and so the input file is compressed. Therefore, the efficiency of the algorithm increases as the number of long, repetitive words in the input data increases.

3.1.3.1 LZW encoding

PSEUDOCODE

```
1    Initialize table with single character strings.
2    P = first input character
```

```
3   WHILE not end of input stream
4       C = next input character
5       IF P + C is in the string table
6         P = P + C
7       ELSE
8         output the code for P
9       add P + C to the string table
10        P = C
11      END WHILE
12  output code for P
```

3.1.3.2 LZW Decompression

The LZW decompressor creates the same string table during decompression. It starts with the first 256 table entries initialized to single characters. The string table is updated for each character in the input stream, except the first one.Decoding achieved by reading codes and translating them through the code table being built.

LZW Decompression Algorithm

PSEUDOCODE

```
1   Initialize table with single character strings
2   OLD = first input code
3   output translation of OLD
4   WHILE not end of input stream
5       NEW = next input code
6       IF NEW is not in the string table
7               S = translation of OLD
8               S = S + C
9     ELSE
10              S = translation of NEW
11      output S
12      C = first character of S
13      OLD + C to the string table
14      OLD = NEW
15  END WHILE
```

3.2 HUFFMAN CODING

3.2.1 How does it work?

Huffman coding can be used to compress images. It is an entropy-based algorithm that relies on an analysis of the frequency of symbols in an array.Lossless JPEG compression uses the huffman coding to encode images.To encode a symbol using the tree, start at the root and traverse the tree until you reach the symbol to be encoded—the encoding is the concatenation of the branch labels in the order the branches were visited.

3.2.2 Implementation

Step I - Building a Huffman tree using the input set of symbols and weight/ frequency for each symbol

A Huffman tree, similar to a binary tree data structure, needs to be created having n leaf nodes and n-1 internal nodes

Priority Queue is used for building the Huffman tree such that nodes with lowest frequency have the highest priority. A Min Heap data structure can be used to implement the functionality of a priority queue.

Initially, all nodes are leaf nodes containing the character itself along with the weight/ frequency of that character.

Internal nodes, on the other hand, contain weight and links to two child nodes

Step II - Assigning the binary codes to each symbol by traversing Huffman tree

Generally, bit ‘0’ represents the left child and bit ‘1’ represents the right child

Algorithm for creating the Huffman Tree-

Step 1- Create a leaf node for each character and build a min heap using all the nodes (The frequency value is used to compare two nodes in min heap)

Step 2- Repeat Steps 3 to 5 while heap has more than one node

Step 3- Extract two nodes, say x and y, with minimum frequency from the heap

Step 4- Create a new internal node z with x as its left child and y as its right child. Also frequency(z)= frequency(x)+frequency(y)

Step 5- Add z to min heap

Step 6- Last node in the heap is the root of Huffman tree

3.3 CLUSTER CONFIGURATIONS

Configuration for a cluster of 1 Master + 15 slaves at NITW-DE Lab

NIT Warangal-D.E.Lab(1+15)				
Node Type	**Ram Size**	**Processor**	**CPU Cores**	**Processor Speed**
Master	16	Intel i7-4770	8	3.40GHz
Slave1-Slave10	16	Intel i7-4770	8	3.40GHz
Slave11-Slave15	8	Intel i7-4770	8	3.40GHz

Configuration for a cluster of 1 Master + 4 slaves at NITW-DE Lab

NIT Warangal-D.E.Lab (1+4)				
Node Type	**Ram Size**	**Processor**	**CPU Cores**	**Processor Speed**
Master	16	Intel i7-4770	8	3.40GHz
Slave1-Slave4	16	Intel i7-4770	8	3.40GHz

Configuration for 1 Master + 4 slaves cluster in Microsoft Azure

Microsoft Azure 1+4 Cluster				
Node Type	**Ram Size**	**Processor**	**CPU Cores**	**Processor Speed**
Master	16	Intel® Xeon® Platinum 8272CL	4	Burstable
Slave0-Slave3	16	Intel® Xeon® Platinum 8272CL	4	Burstable

Configuration for 1 Master + 18 slaves cluster in Microsoft Azure

Microsoft Azure 1+18 Node Cluster				
Node Type	**Ram Size**	**Processor**	**CPU Cores**	**Processor Speed**
Master	16	Intel® Xeon® Platinum 8272CL	4	3.40GHz
Slave0-Slave17	16	Intel® Xeon® Platinum 8272CL	4	3.40GHz

3.4 DATASETS USED

3.4.1 Salzburg Texture Image Database(STex) for Image compression algorithms:

- 476 texture images with 512x512 dimension
- split into 1,21,856 32x32 non overlapping tiles and then finally converted into grayscale.
- When less than 1 lakh dataset is required, the appropriate portion of the 1,21,856 Dataset is taken out, otherwise the 1,21,856 images are replicated accordingly.

3.5 DISTRIBUTED IMAGE COMPRESSION

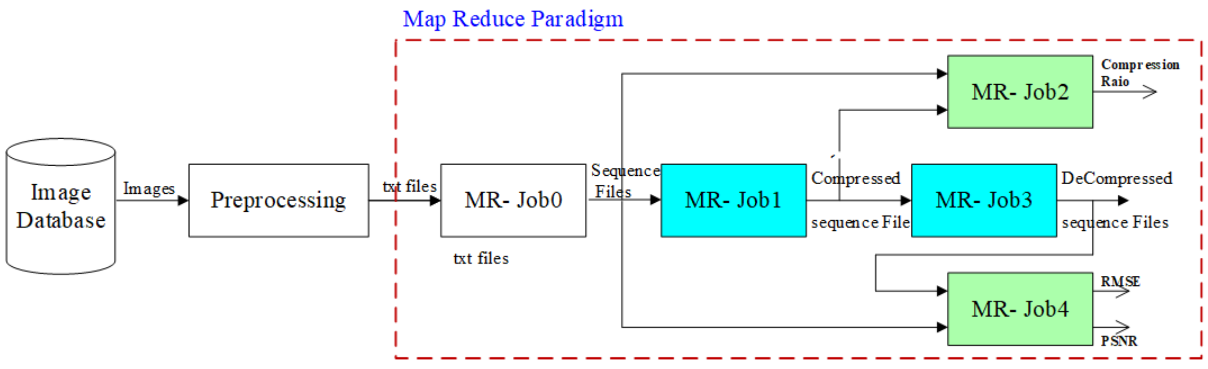


Fig 3.5 shows Map Reduce paradigm for Image compression and decompression in both Hadoop and Spark.

The database contains 32*32 grayscale image files.These images are preprocessed i.e,replicate the generated 1 lakh images to as many as needed,convert them to single text file to solve the small file problem and store them in HDFS.MapReduce_Job0 reads this text file and creates a sequence file of uncompressed images.it has key values pairs<filename,original pixel values>.MR_Job1 is compression.it outputs <filename,decompressed pixel values>.MR_Job2 takes in the uncompressed data and compressed data to calculate the compression ratio.The outputs of Job0 and Job1 are the inputs of the Job2.MR_Job4 takes in the uncompressed and decompressed data and outputs the RMSE and PSNR values.The outputs of Job0 and Job3 are the inputs for Job4.

3.6 RESULTS

We were successful in compressing and decompressing more than 200 million images using the LZW and huffman algorithm.The results are shown below.

3.6.1 Results Obtained for LZW Compression by varying cluster configuration and size of Dataset.

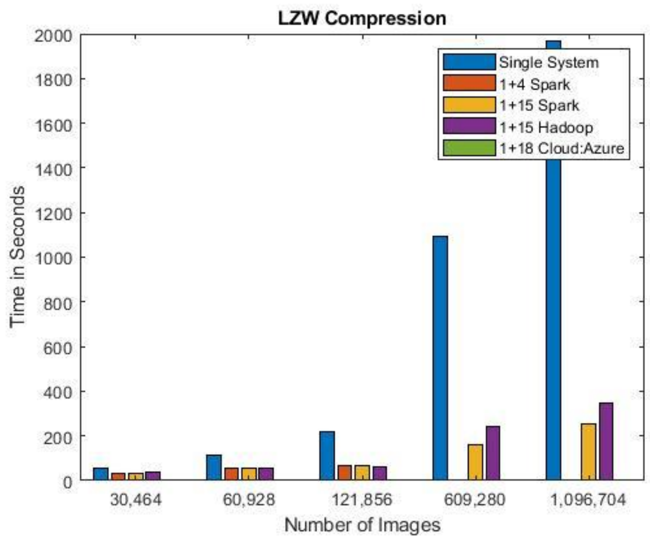


Figure 1:LZW compression results in a bar graph

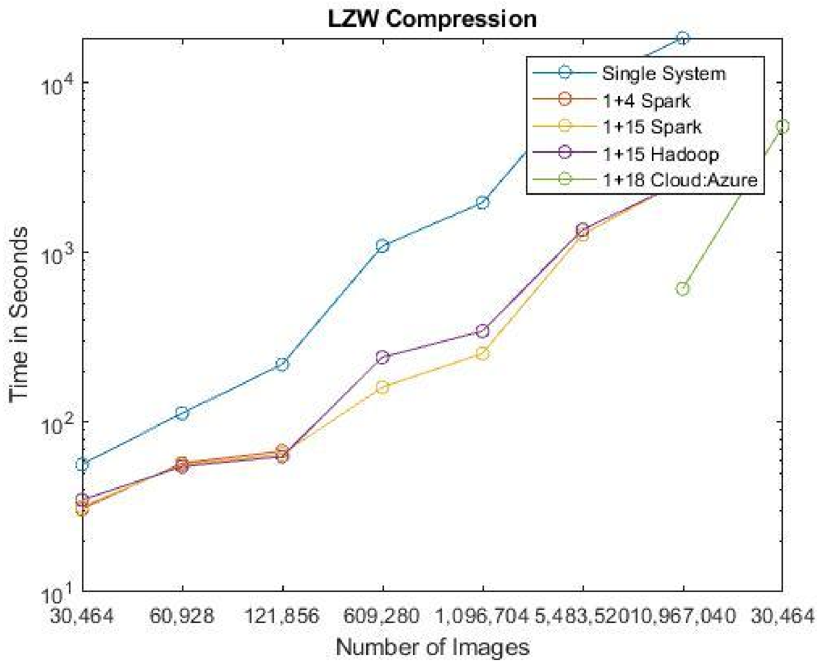


Figure 1:LZW compression results in a bar graph

Figure 2:LZW compression results in a semi-log graph

3.6.2 Results Obtained for LZW Decompression by varying cluster configuration and size of Dataset:

Figure 3:LZW decompression results in a bar graph

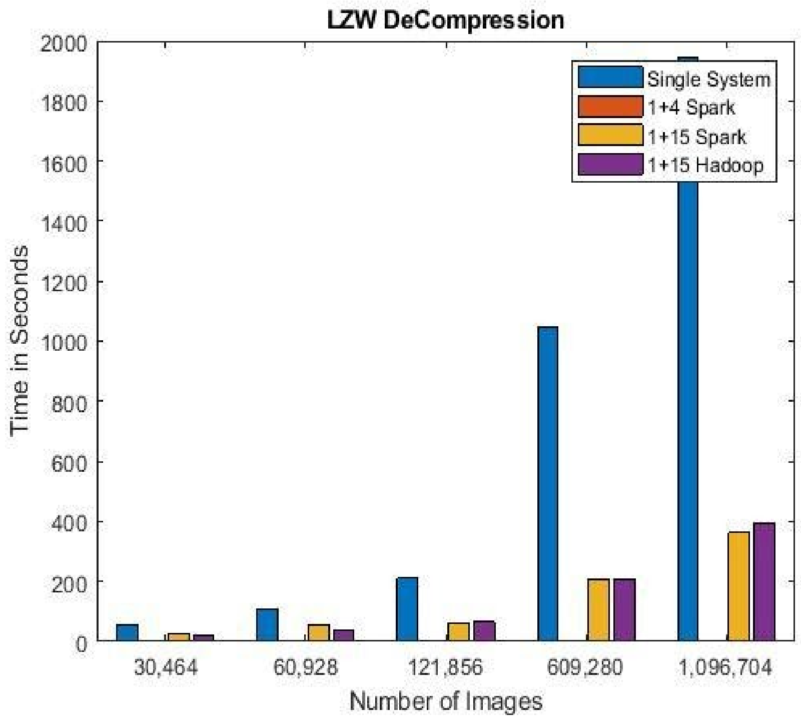
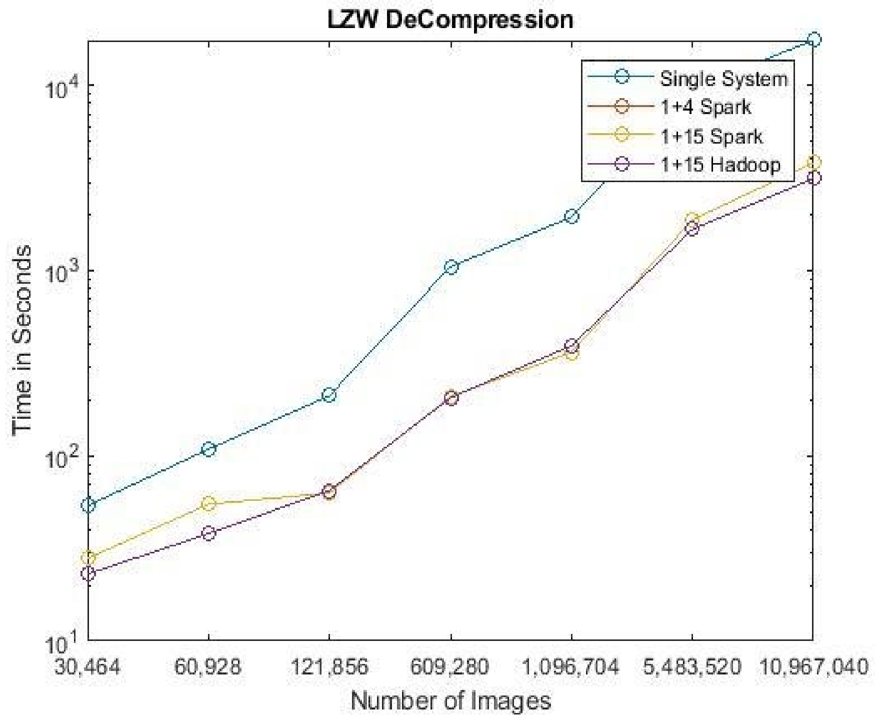


Figure 4:LZW decompression results in a semi-log graph



3.6.3 Results Obtained for Huffman Compression by varying cluster configuration and size of Dataset:

Figure 5:Huffman compression results in a bar graph

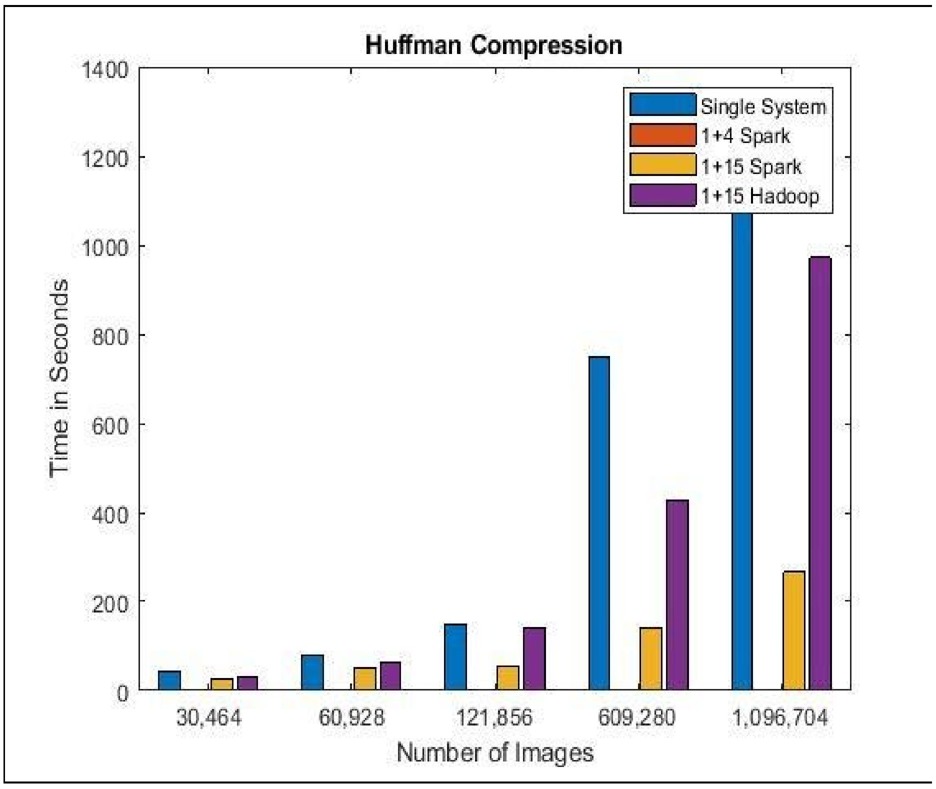
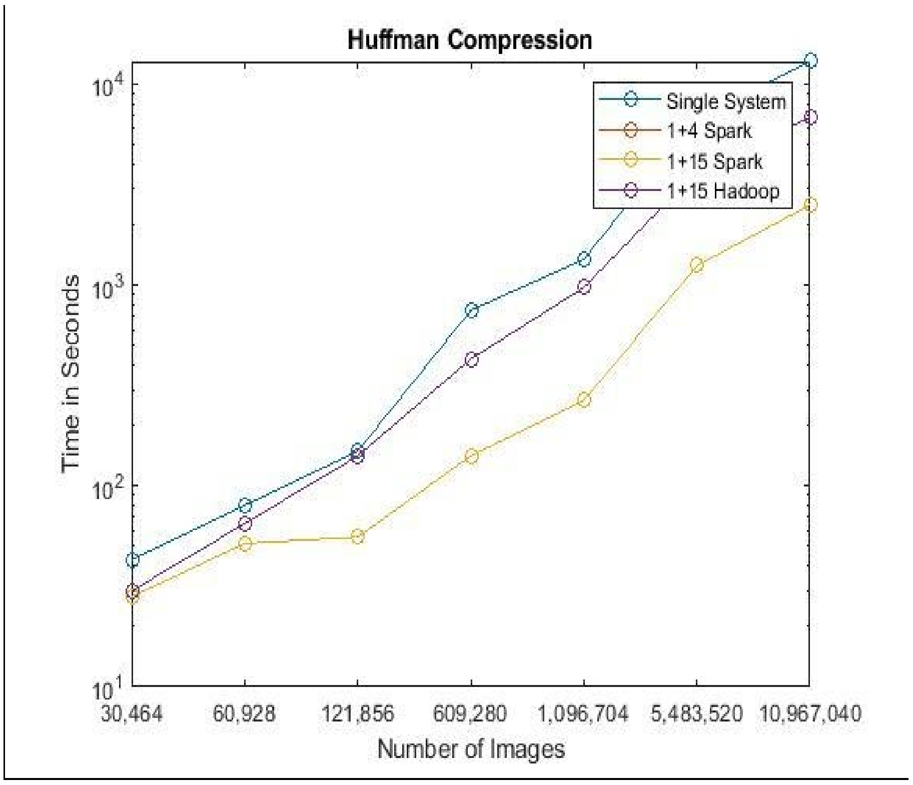


Figure 6:Huffman compression results in a semi-log graph



3.6.4 Results Obtained for Huffman DeCompression by varying cluster configuration and size of Dataset:

Figure 7:Huffman decompression results in a bar graph

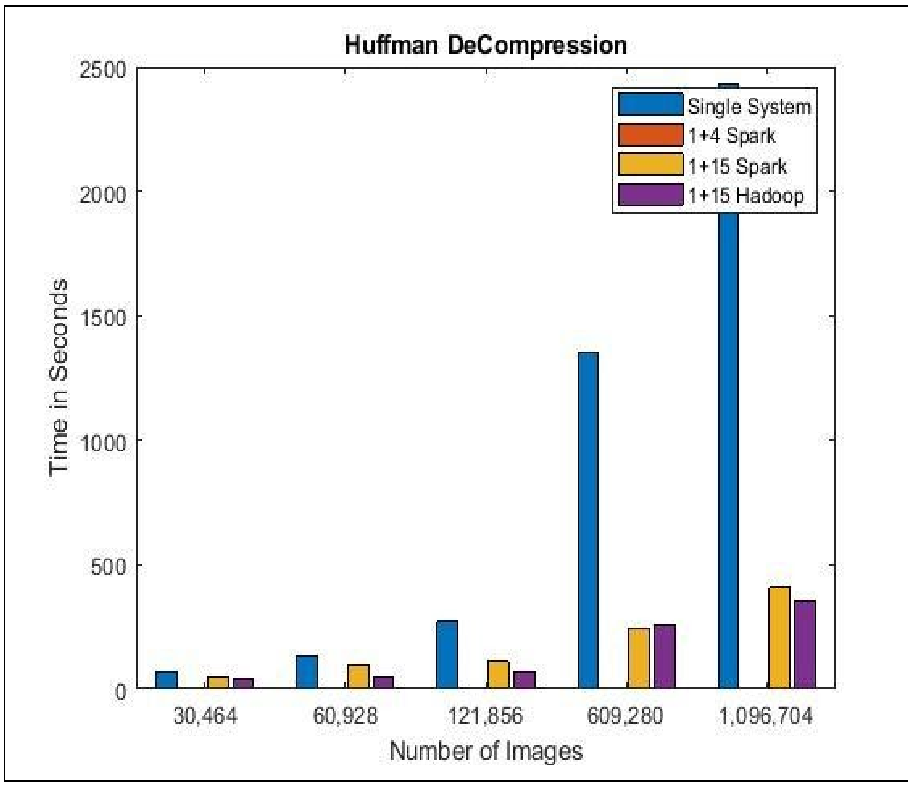
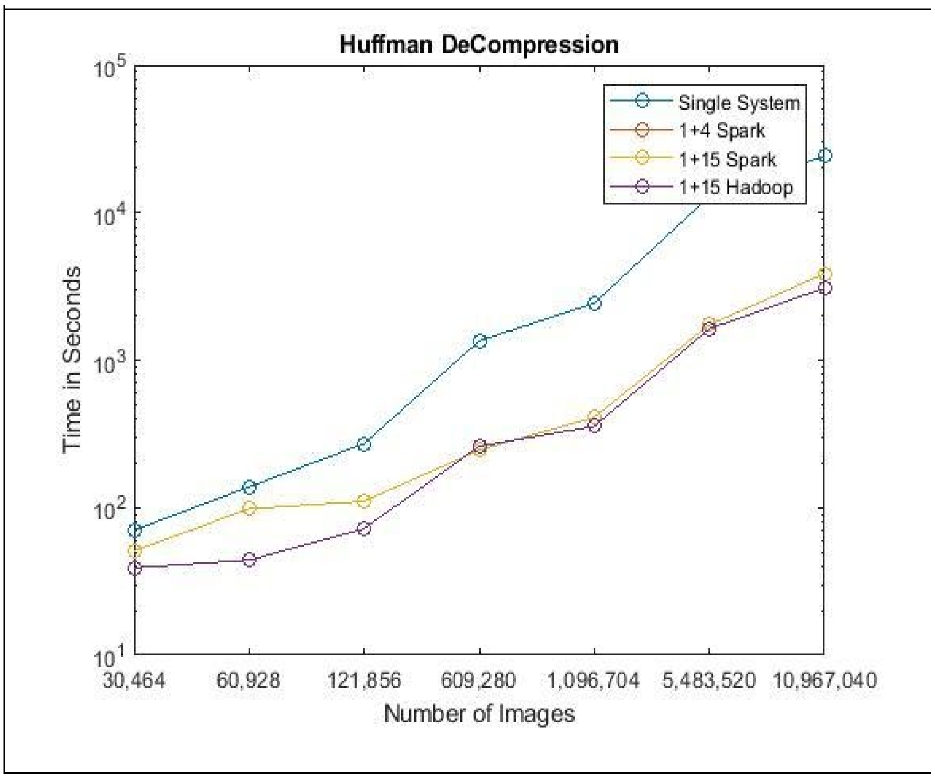


Figure 7:Huffman decompression results in a semi-log graph



3.7 Observations

We have observed that spark outperforms Hadoop by a huge margin and both spark and hadoop are outperforming single node computations.We have also observed that by using SSDs in the azure 1+18 cluster instead of Hard disk we have seen a significant improvement.

CHAPTER 4
CONTENT-BASED IMAGE RETRIEVAL

4.1 INTRODUCTION

Content-based image retrieval is the application of computer vision techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases. "Content-based" means that the search analyzes the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image. The term "content" in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. CBIR is desirable because searches that rely purely on metadata are dependent on annotation quality and completeness. The term content can also be referred to as features which is the important information extracted from each image.We have used various feature extraction techniques for content based image retrieval in the current report.

4.2 HANDCRAFTED FEATURE EXTRACTION ALGORITHMS

Texture plays an important role in the characterization of regions in digital images. It carries information about the microstructure of the regions and the distribution of the grey levels. Texture is an inherent property of any image and of medical images in particular, given that the tissue itself carries a dominant textural appearance.So we are focusing on algorithms outputting texture based features.

4.2.1 BLOCK DIAGRAM

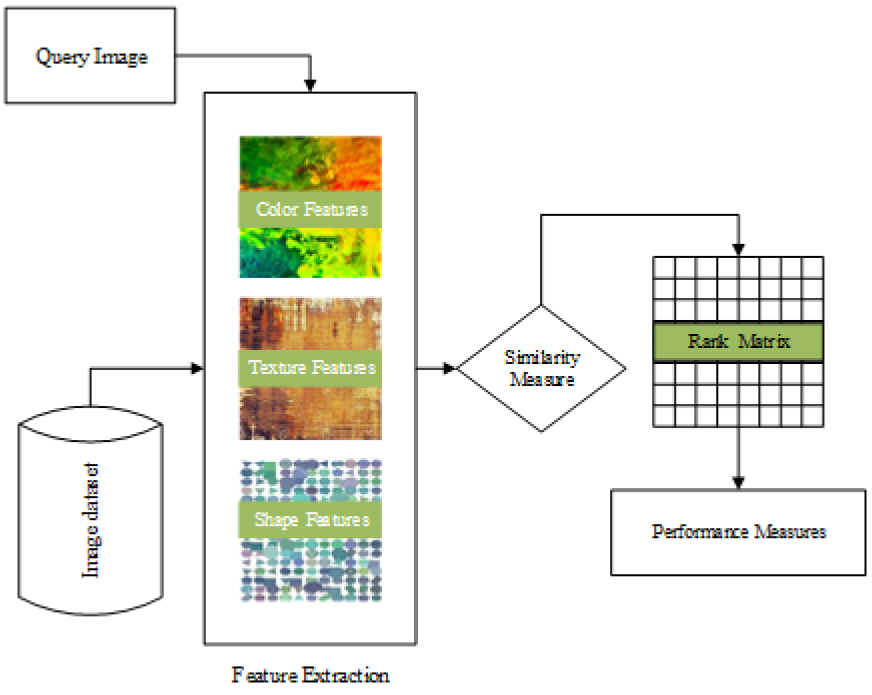


Fig 5.1 shows architecture of CBIR using handcrafted feature extraction.

4.2.2 LOCAL BINARY PATTERN (LBP)

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. It outputs texture features of an image. These feature vectors can be fed into the CBIR similarity measure step,where the current feature can be compared with the already existing features of images in the database and obtain the most similar images.

The LBP feature vector, in its simplest form, is created in the following manner:

1. Divide the examined window into cells (e.g. 16x16 pixels for each cell).
2. For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
3. Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
4. Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
5. Optionally normalize the histogram.
6. Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

4.2.3 UNIFORM LOCAL BINARY PATTERN (ULBP)

A useful extension to the original operator is the so-called uniform pattern,[8] which can be used to reduce the length of the feature vector and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, 00010000 (2 transitions) is a uniform pattern, but 01010100 (6 transitions) is not. In the computation of the LBP histogram, the histogram has a separate bin for every uniform pattern, and all non-uniform patterns are assigned to a single bin. Using uniform patterns, the length of the feature vector for a single cell reduces from 256 to 59. The 58 uniform binary patterns correspond to the integers 0, 1, 2, 3, 4, 6, 7, 8, 12, 14 etc. The feature vectors obtained this way are fed into CBIR and most similar images are obtained.

4.2.4 INTERCHANNEL VOTING

The authors in [1] presented a generic architecture to get highly useful information from the images. This work explores the method of quantization, inter-channel voting, DSP and GLCM. It also exploits the properties and characteristics of the channels in HSI color space that give color and texture features of the image.The motivation for shifting focus from segregating H, S and I channels individually into bins and concatenating their histograms to perform the operation of voting (inter-channel voting) is that we obtain the relationship among all three H, S and I channels. We have implemented inter channel voting and used the extracted features to feed into the CBIR model and get the desired results.

4.3 FEATURE EXTRACTION USING DEEP CNN

In this report, we propose to use features derived from pre-trained network models from a deep learning convolution network trained for a large image classification problem. This approach appears to produce vastly superior results for a variety of databases, and it outperforms many contemporary CBIR systems.

4.3.1 BLOCK DIAGRAM

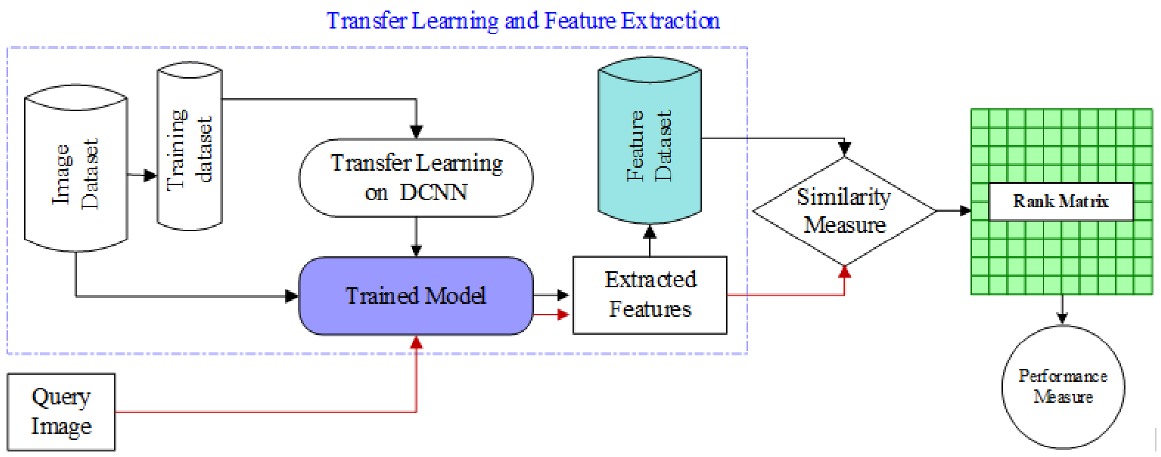


Fig 5.3 shows General CBIR Framework using DCNN Features.

4.3.2 DEEP CONVOLUTIONAL NEURAL NETWORKS (DCNN)

Convolutional Neural Networks take input as images and they handle the architecture in a more sensible way. The layers of a CNN have neurons which are arranged in 3 dimensions: width, height, depth.There are mainly three types of layers in CNN architectures: Convolutional Layer, Pooling Layer and Fully-Connected Layer. We will stack these layers on top of each other to form a fully operational CNN architecture. A Deep CNN is nothing more than too many of these layers stacked up for training purposes.

4.3.3 TRANSFER LEARNING

Transfer learning[39] is a method where instead of starting the training process of a model from scratch, we use the learned weights of an already trained model to solve a different but similar problem. In this way we leverage previous learning through the learned weights and save a considerable amount of time. Usually much better results are also achieved compared to training from scratch. In computer vision, transfer learning is usually executed by the use of pre-trained models. A pre-trained model[22] is a model that was trained on a large benchmark dataset to solve some specific problems similar to the ones we want to solve. Accordingly, because of the high computational cost of training such deep learning models, it is a common practice to import and use models from a published architecture (e.g. AlexNet, ResNet, Xception etc).Being motivated by this, we have used a pre-trained Neural Network model called AlexNet Classifier.We propose to use these higher-level features for the feature representation of images in a CBIR system. So, we removed the last softmax activation layer used for calculating probabilities of each class and selected the preceding fully connected layer to be our feature vector representation for CBIR. As this vector is the deepest layer of the model, this represents the most learned high-level features.

4.3.3.1 ALEXNET CLASSIFIER

AlexNet is a deep learning model and it is a variant of the convolutional neural network.The AlexNet proposed by Alex Krizhevsky in his work has eight layers including five convolutional layers followed by three fully connected layers. Some of the convolutional layers of the model are followed by max-pooling layers. As an activation function, the ReLU function is used by the network which shows improved performance over sigmoid and tanh functions.The AlexNet employing the transfer learning which uses weights of the pre-trained network on ImageNet dataset has shown exceptional performance.

4.3.3.2 DARKNET CLASSIFIER

DarkNet-53 is a convolutional neural network that is 53 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 256-by-256. Darknet-53 is a convolutional neural network that acts as a backbone for the YOLOv3 object detection approach. The improvements upon its predecessor Darknet-19 include the use of residual connections, as well as more layers.

4.4 CLUSTER CONFIGURATIONS

AWS Single system Node

RAM Size	Processor	vCPU	Processor Speed	GPU	GPU Size	OS Available
61 GB	Intel Xeon E5-2686v4	8	2.7 GHz	NVIDIA Tesla V100 GPU	16	Ubuntu 18.04.5-64

Personal Single system Node

System configuration:
12GB RAM, Intel i7-7700HQ, 4GB Nvidia GTX 1050Ti Mobile

Common Configurations: Hadoop 3.3.0, Spark 3.1.1, Ubuntu 20.04

4.5 DATASETS USED

Corel-1 K Dataset

Corel-1 K database [50], which is the first database to evaluate the performance of the proposed method with other existing methods, comprises 10 categories of 100 images each resulting in a total of 1000 images in the database. This database includes images of Africans (1–100), Beaches (101–200), Buildings (201–300), Buses (301–400), Dinosaurs (401–500), Elephants (501–600), Flowers (601–700), Horses (701–800), Mountains (801–900) and Food (901–1000). Size of images in this database is either 384 × 256 or 256 × 384.

4.6 DISTRIBUTED CBIR

The feature extraction part,whether it is using handcrafted features or features extracted through transfer learning, is done in a distributed manner and the remaining part of the CBIR, that is calculating rank matrix and evaluating performance, is done on a single system.The Input image i.e the RGB intensity values are fed into the MAP phase where feature extractor is executed in distributed manner.

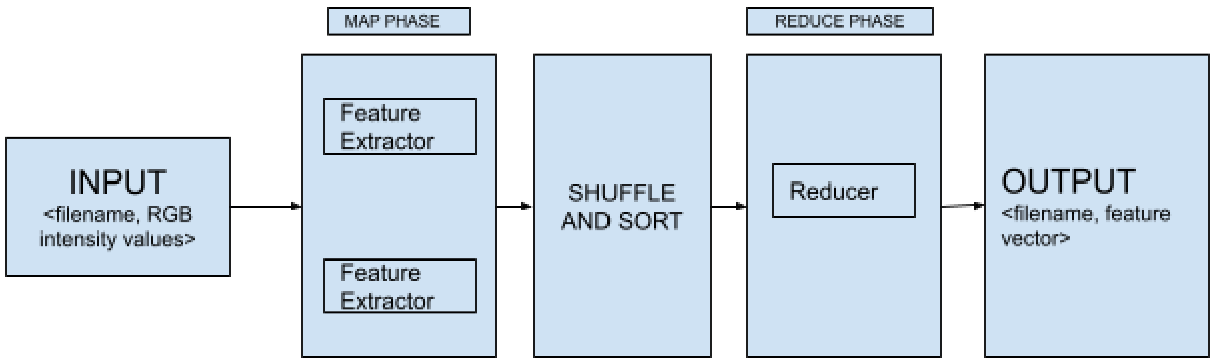


Fig 4.6 shows the architecture of feature extraction in distributed manner

4.7 RESULTS & OBSERVATIONS

Type	Method Name	APR	ARR	F-Measure	ANMRR	TMRE
Handcrafted Features	LBP	69.18	38.69	31.23	0.52	803.17
	ULBP	69.26	44.42	33.93	0.46	749.06
	Interchanel Votting HSI	78.56	50.29	55.41	0.39	728.89
Deep Learning Features	ALexNet	76.22	41.59	34.55	0.49	984.68
	Darknet-53(Python)	94.22	75.55	79.99	0.13	372.90
	Darknet-53(AWS)	79.75	51.16	56.16	0.38	909.01
	Inception Darknet-53(AWS)	88.32	63.64	68.58	0.25	630.61
Concatination	Interchanel Votting HSI +AlexNet	81.24	40.49	46.85	0.50	987.23
	Interchanel Votting HSI +DarkNet-53(Python)	91.60	64.33	70.08	0.25	885.49
	Interchanel Votting HSI +DarkNet-53(AWS)	78.36	47.62	52.76	0.42	940.67
	Interchanel Votting HSI + Inception DarkNet-53(AWS)	82.73	55.21	60.32	0.34	684.83

APR: Average Precision Rate
ARR: Average Recall Rate
ANMRR: Average Normalized Modified Retrieval Rank
TMRE: Total minimum Retrieval Epoch.

Higher the APR and ARR is,the better the algorithm performs.We can observe that InterChannel Voting HSI performs best among the Handcrafted algorithms due to its ability to extract greater level texture features than LBP and ULBP. Among transfer learning algorithms,Darknet-53 performs best in terms of precision and recall. Over here,python refers to the single system we have used which is our personal desktop and AWS is the single system we have used on AWS.

CHAPTER 5
DISTRIBUTED DEEP LEARNING

In recent years there have been a lot of advancements in the field of deep learning, and a lot of deep (convolutional) neural networks are being used for the various image processing tasks. But there is also an inherent problem with these networks, that is they take a lot of time for training. Some deep networks like AlexNet, Darknet have more than 60 million weights and training these networks is a tedious task for many high-performance computers too. Also, the dataset sizes are also getting very large, for example the ImageNet dataset which is one of the most popular datasets used to train a lot of deep neural networks consists of more than 14 million images.

To solve the above-mentioned problem of training, distributed deep learning can be used. With distributed deep learning we can leverage multiple computers to train the network, in a way we are parallelizing the training to reduce the training time drastically. There are two types of parallelism when it comes to training the networks 1) Model parallelism, where the model is split into different nodes of a cluster and trained on the dataset, 2) Data parallelism where each slave node gets its own copy of model from the master node but the data is split among the node. In this project we focus on data parallelism

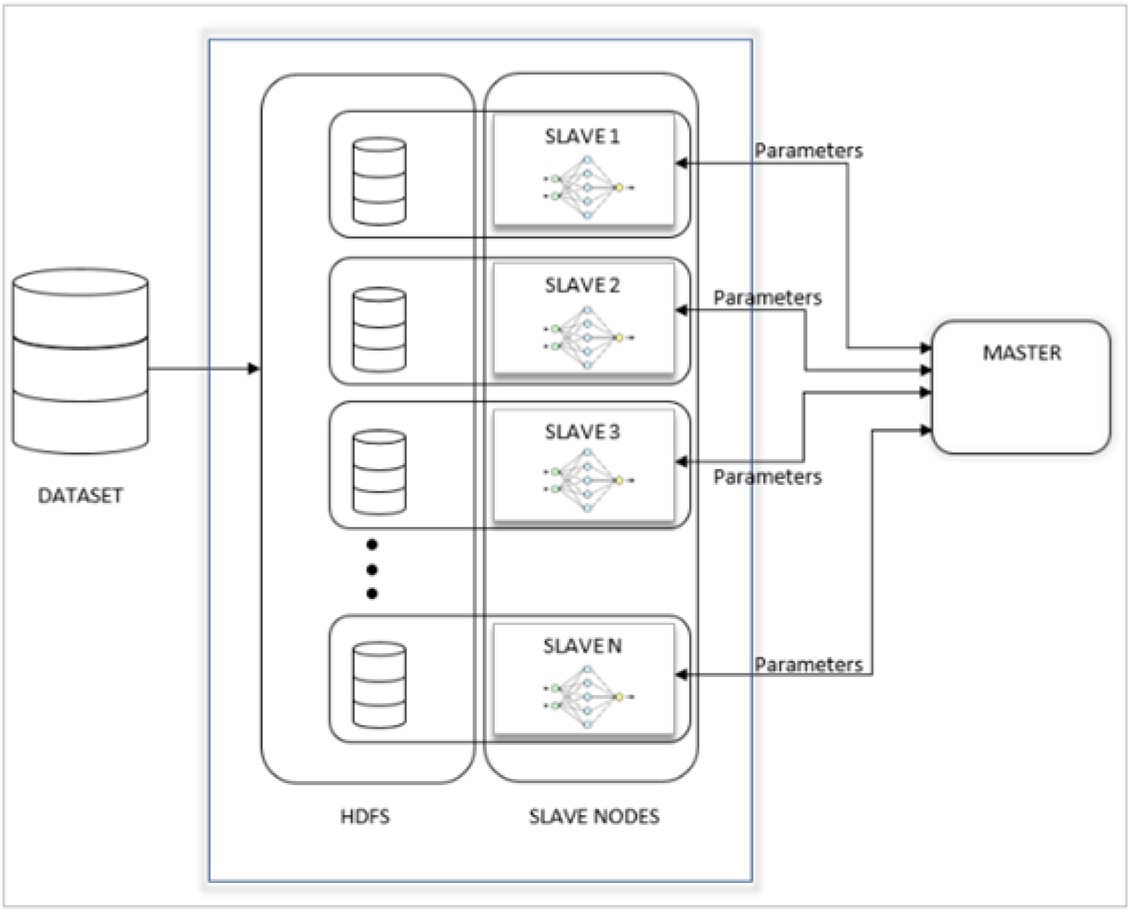


Figure 24 Data Parallel Training

5.1 MODES OF EXECUTION

SYNCHRONOUS:

All the slave nodes return the weight updates at the same time to the master node which calculates the updates and transfers the updates to all the slave nodes at the same time. If any of the nodes are slow then all the other nodes have to wait for them. This results in the wastage of their computational time.

ASYNCHRONOUS:

When the master node receives an update from a slave node(s) it applies a MUTEX lock on the shared variable to perform operations based on the update received from the node(s). After the operations are finished, the corresponding updated model is sent to the respective slave node(s). Even though this is better than Synchronous, it still has a computational overhead due to MUTEX locks.

HOGWILD:

In Hogwild, processors are allowed equal access to shared memory and are able to update individual components of memory at will. Such a lock-free scheme might appear doomed to fail as processors could overwrite each other's progress. However, when the data access is sparse, meaning that individual SGD steps only modify a small part of the decision variable, it has been shown that memory overwrites are rare and that they introduce barely any error into the computation when they do occur.

5.2 METHODOLOGY

For this project, we have trained AlexNet for CBIR feature extraction using distributed deep learning. We used Elephas library to train the model parallelly on a distributed cluster. All the images are first put in HDFS. We then create RDDs containing all 10 classes of the dataset. The images in RDDs are pre-processed before training. While creating the Spark model, the mode of execution is also specified. After the model is created and training data is pre-processed, training starts and the final model is saved in the Master Node. Using the test data, the accuracy of the model is computed. In the master node, the normal training (training in single node) is also done with the training images. Similar to before, the accuracy of the model is also computed.

5.3 DATASET USED

Corel-1K:

- Classes: 10
- Images in each class: 100
- Total Images: 1000

5.4 CLUSTER USED

Microsoft Azure 1+4 Cluster				
Node Type	**Ram Size**	**Processor**	**CPU Cores**	**Processor Speed**
Master	16	Intel® Xeon® Platinum 8272CL	4	Burstable
Slave0-Slave3	16	Intel® Xeon® Platinum 8272CL	4	Burstable

Table 8 Cluster-4

Few other specifications:

- OS – Ubuntu 20.04
- Hadoop – 3.3.0 (Released 6th July 2020)
- Spark – 3.1.1 (Released 2nd march 2021)

5.5 RESULTS

Corel-1K			Time				Model Accuracy			
	S. No.	Setup	Normal	Synchronous	Aynchronous	Hogwild	Normal	Synchronous	Aynchronous	Hogwild
	1	Single		N/A	N/A	N/A		N/A	N/A	N/A
	2	1+4	N/A	303.800793647766	2,004.768443822860	2,031.846085786820	N/A	0.182761643082	0.117177767213	0.182472874410
	3	1+15	N/A		2,011.906435251240	2,017.566215038300	N/A		0.106060605496	0.189265333116
	4	Azure	N/A				N/A			

Table 9 AlexNet CBIR Results on Corel 1K Dataset

5.6 OBSERVATIONS

Elephas does not support transfer learning. Hence we have to train Alexnet from scratch using the given data. This amount of data is not enough to get good results hence we see such low accuracies. In cases where a large amount of data is present we can see good accuracy with a significant improvement in training time.

CHAPTER 6

CONCLUSION

In this report, we have discussed the distributed processing of few most used image processing algorithms.

We were able to successfully reduce the amount of time taken to perform Image compression on more than 200 million images using Huffman and LZW compression methods in significantly less time without the change in algorithm. We were also able to execute the same on both Hadoop and Spark clusters.

We were also successful in using distributed computing for feature extraction for CBIR. Successfully used distributed computing for content based image retrieval using Alexnet classifier.

Distributed Deep Learning is a field in which research is still going on and there is a long way ahead in this field. As of now we were able to verify that the time it takes to train the network has reduced drastically with the help of Distributed Deep Learning.

We were able to execute all these on cloud clusters as well. Also, we have observed Spark performs significantly better than Hadoop on these tasks.

We further wish to test the limits of the proposed methodology applied on compression and classification.

CHAPTER 7

REFERENCES

[1] J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, QV. Le, MZ. Mao, M'A. Ranzato, A. Senior, P. Tucker, K. Yang, and AY. Ng. Large Scale Distributed Deep Networks.

[2] F. Niu, B. Recht, C. Re, S.J. Wright HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent

[3] C. Noel, S. Osindero. Dogwild! — Distributed Hogwild for CPU & GPU

[4] LZW Compression Article from Dr. Dobbs Journal: Implementing LZW compression using Java, by Laurence VanhelsuwÈ

[5]Maji, Subhadip & Bose, Smarajit. (2020). CBIR using features derived by Deep Learning.

[6]Khawaja Ahmed, Shahida, and Muhammad Iqbal. "Content Based Image Retrieval using Image Features Information Fusion". In: Information Fusion 51 (Nov. 2018). DOI: 10.1016/j.inffus.2018.11.004.

Digital Image Processing by Rafael C. Gonzalez 3rd Edition