BIG IMAGE DATA PROCESSING IN CLOUD AND DISTRIBUTED ENVIRONMENT

TEAM:
Karthik kothuri (177230)
Mounika Kukudala (177231)
Shashikar Anthoniraj (177259)

PROJECT GUIDE: Dr. U.S.N Raju

PROBLEM STATEMENT

- Image compression is a great way of storing images and efficient transmission, considering the compression ratio. With upcoming satellite like GISAT that will be downlinking data 24*7 of the tune of 5-6 TB per day, data and metadata storage, cataloguing and search becomes a daunting challenge. Processing images on distributed cloud computing infrastructure for quick archival and retrieval of remote sensing data utilizing lossless data compression techniques is an interesting way to address this issue.

- Current state of the art techniques use millions of images for training and billions of weights to be stored, training networks on such a large scale is very time consuming even on high performance computers.

- Distributed computing solves the problem of Big Image processing.

OBJECTIVES

- Implementation and Analysis of execution of Image compression algorithms such as LZW and Huffman on distributed cloud environments.
- Comparison of results obtained from CBIR(content based Image retrieval) from handcrafted features and DCNN features.
- Training Deep learning Algorithms in a distributed manner.

Cluster configurations used for compression and DDL:

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

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

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

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

Cluster configurations used for CBIR:

AWS								
Node Type	**Ram Size**	**Processor**	**vCPU**	**Processor Speed**	**GPU**	**GPU Size**	**Operationg System Used**	**OS Availabe**
Single	61	Intel Xeon E5-2686v4	8	2.7 GHz	NVIDIA Tesla V100 GPU	16	Ubuntu 18.04.5-64	Ubuntu 18.04.5-64

Datasets used:

- Salzburg Texture Image Database(STex) for Image compression algorithms:
  - 476 Color 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, appropriate portion of the 1,21,856 Dataset is taken out, otherwise the 1,21,856 images are replicated accordingly.

- COREL Dataset for CBIR Execution:
  - 1000 Color images with 256x384 dimension.

Map Reduce Paradigm for Image Compression and Decompression for both Hadoop and spark

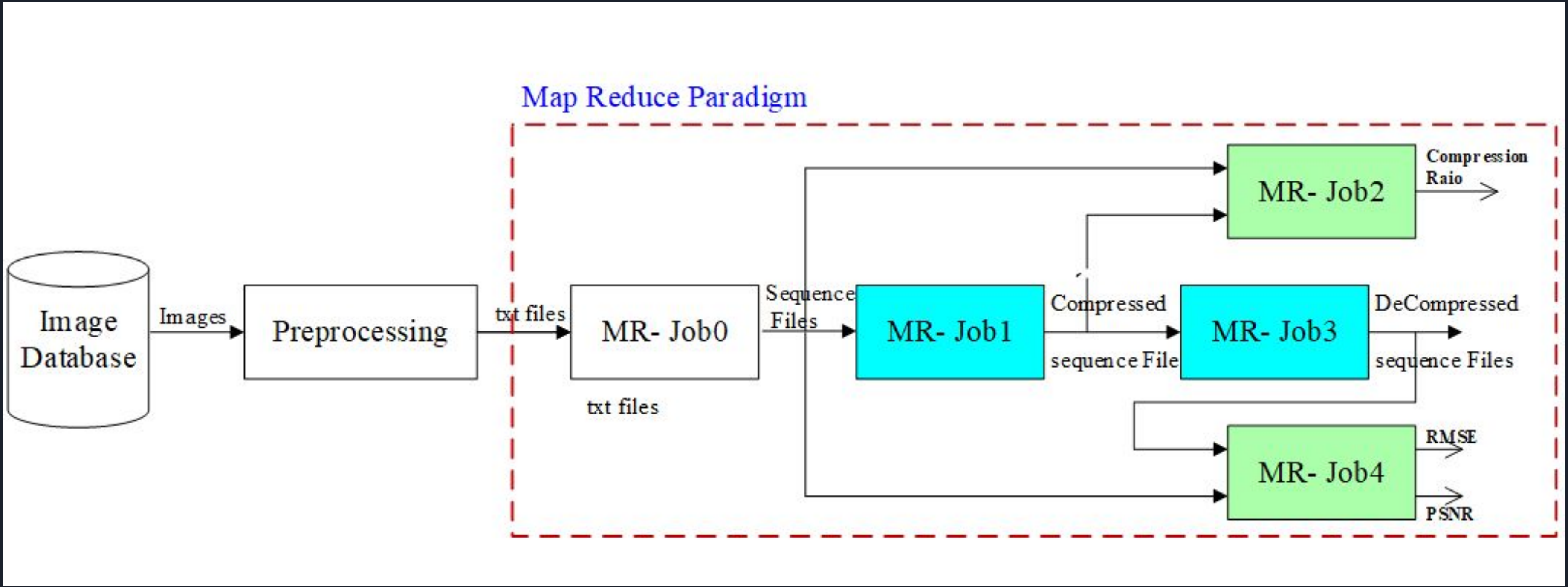


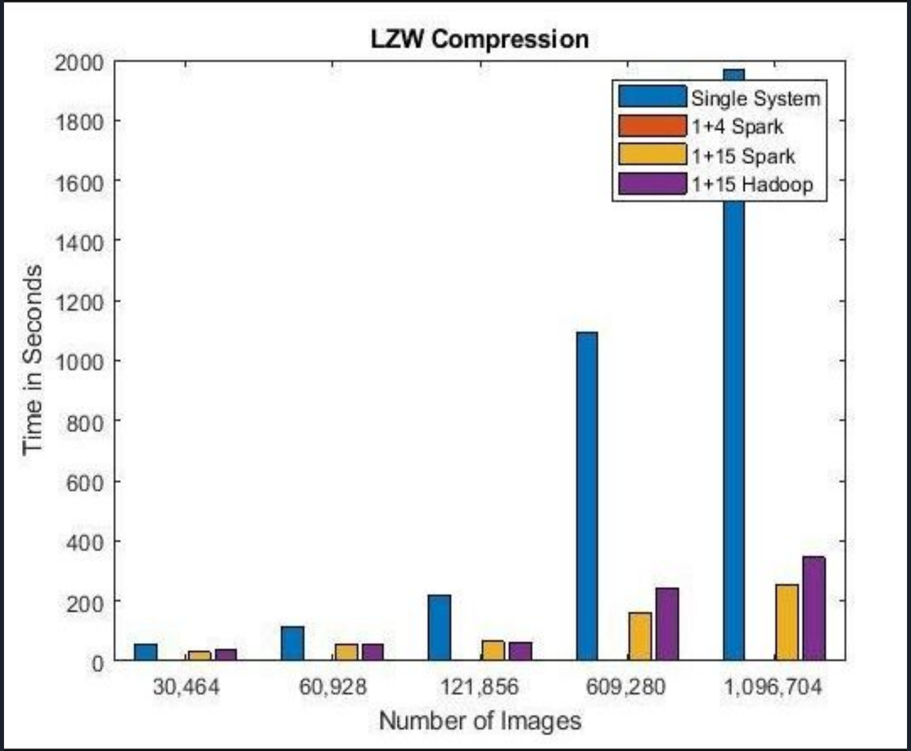
IMAGE COMPRESSION ALGORITHMS:

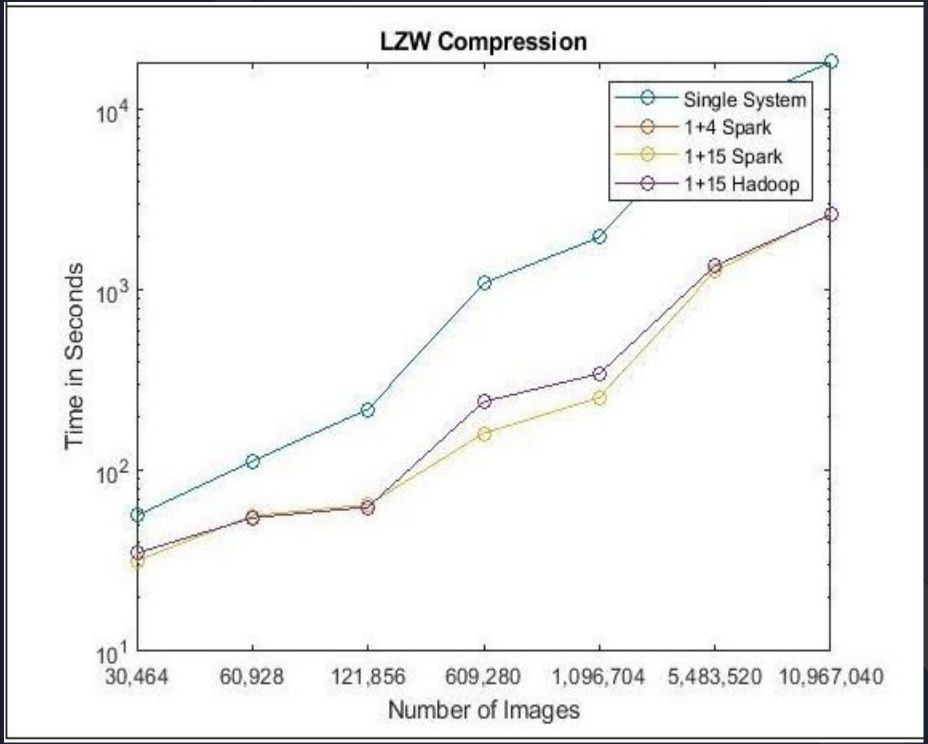
- **LZW (Lempel-Ziv-Welch)** :
  - ➢ LZW is one of the dictionary based compression algorithms.
  - ➢ Dictionary-based algorithms do not encode single symbols as variable-length bit strings, instead they encode variable-length strings of symbols as single tokens.
  - ➢ A dictionary that is indexed by ‘codes’ is used and is assumed to be initialized with 256 entries.

- **HUFFMAN** :
  - ➢ Huffman coding can be used to compress to 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

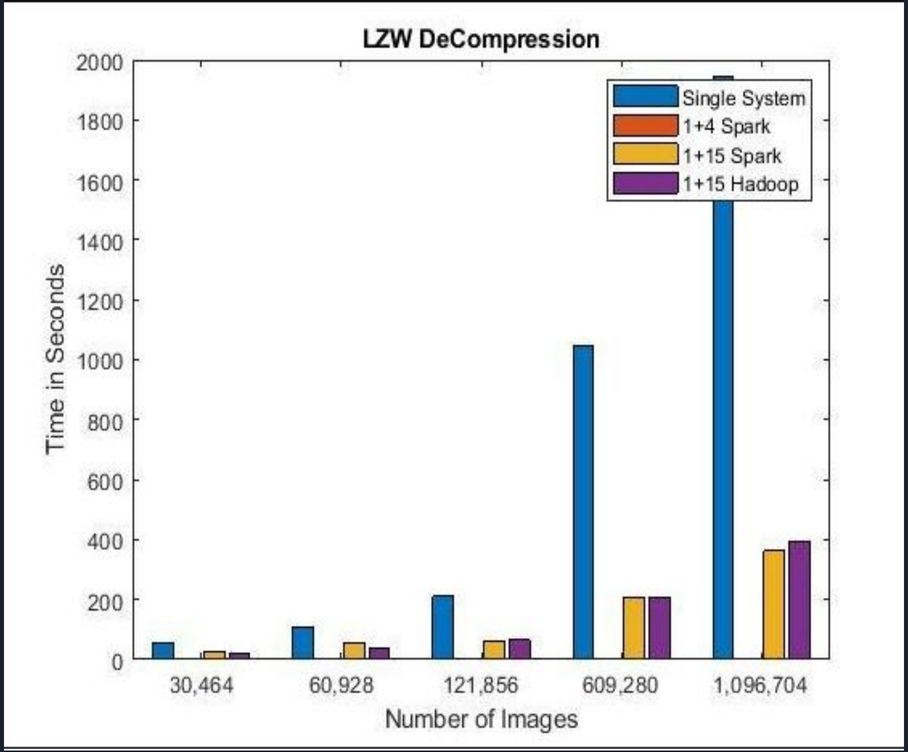
Statistics of Image Compression Algorithms: Link

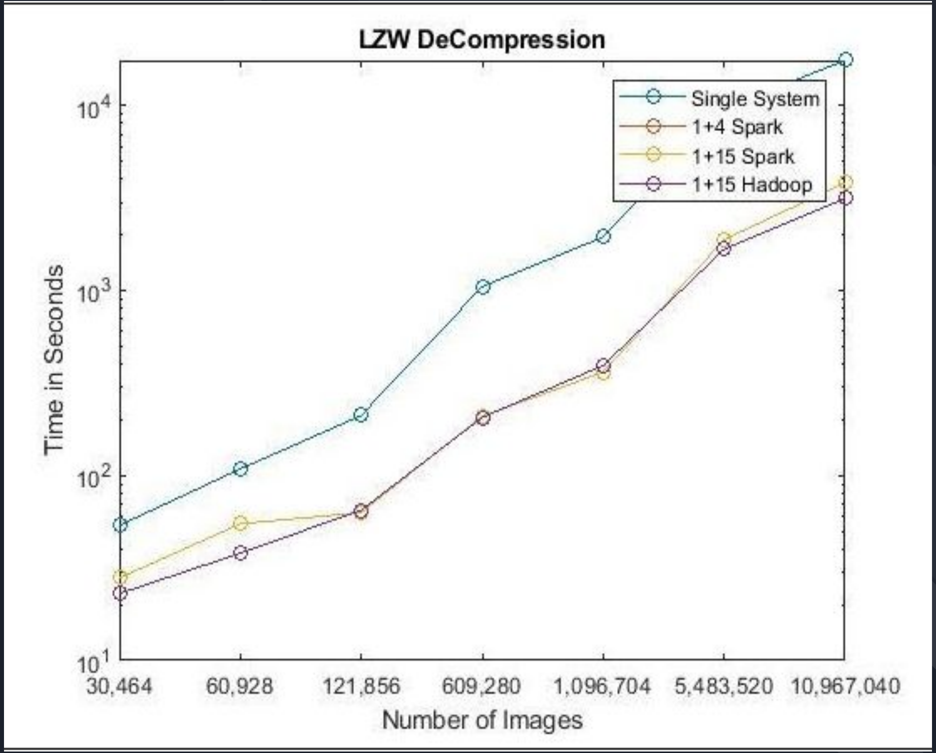
Results Obtained for LZW Compression by varying cluster configuration and size of Dataset:



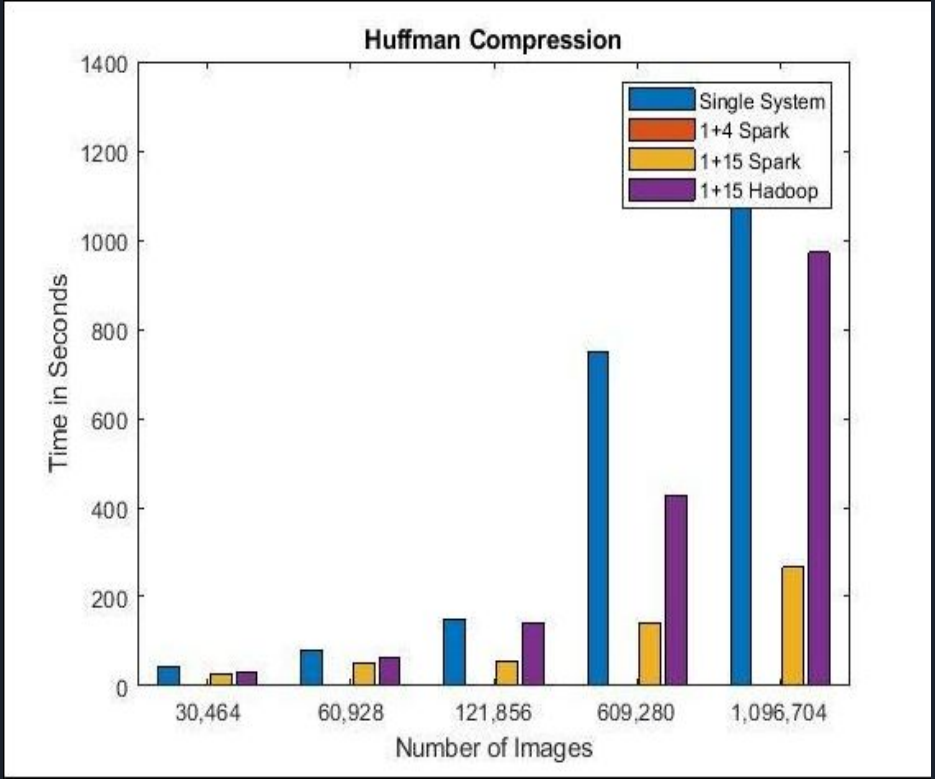


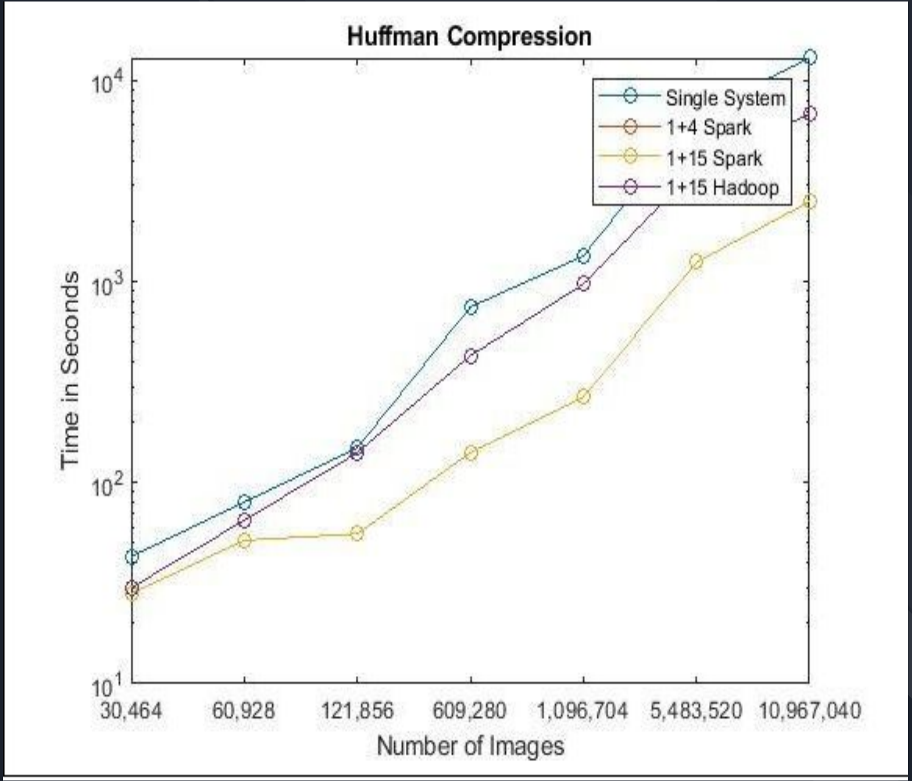
Results Obtained for LZW Decompression by varying cluster configuration and size of Dataset:



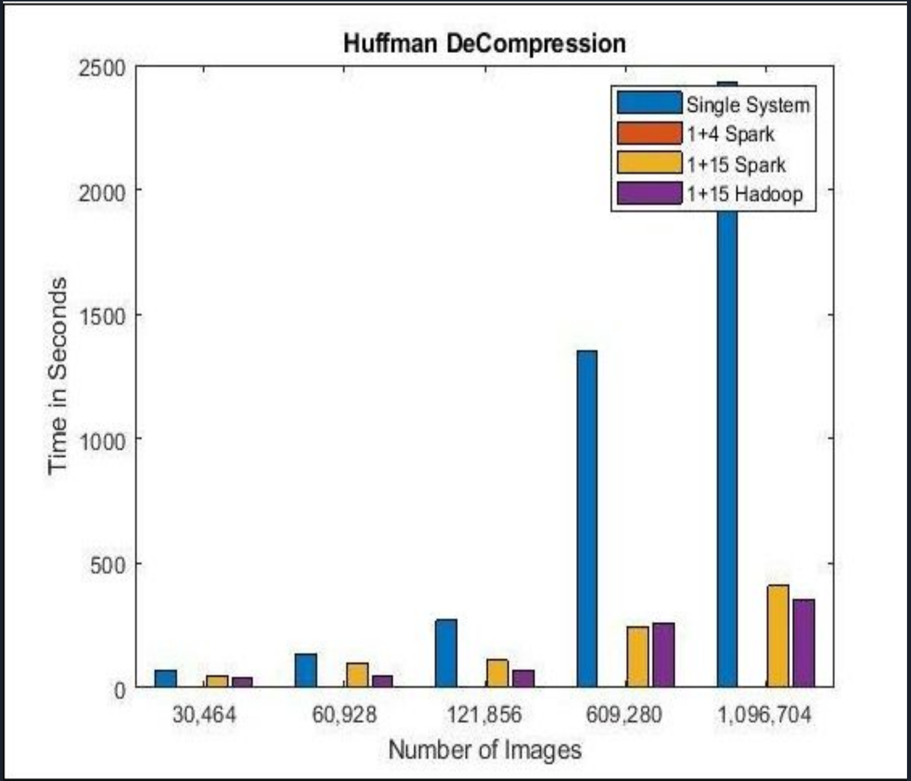


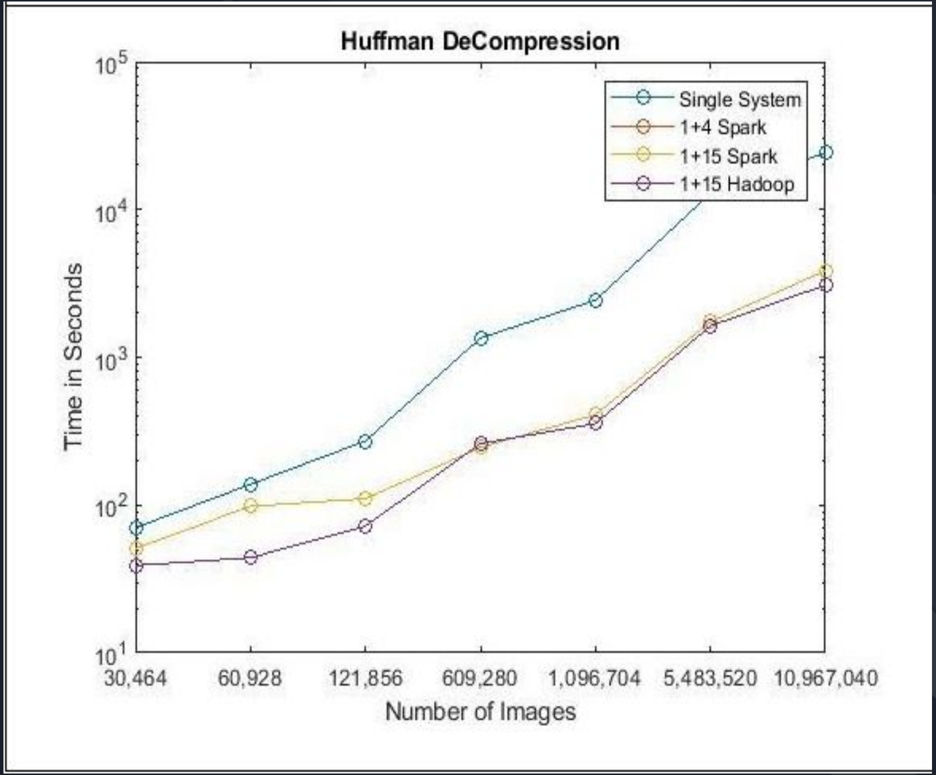
Results Obtained for Huffman Compression by varying cluster configuration and size of Dataset:





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

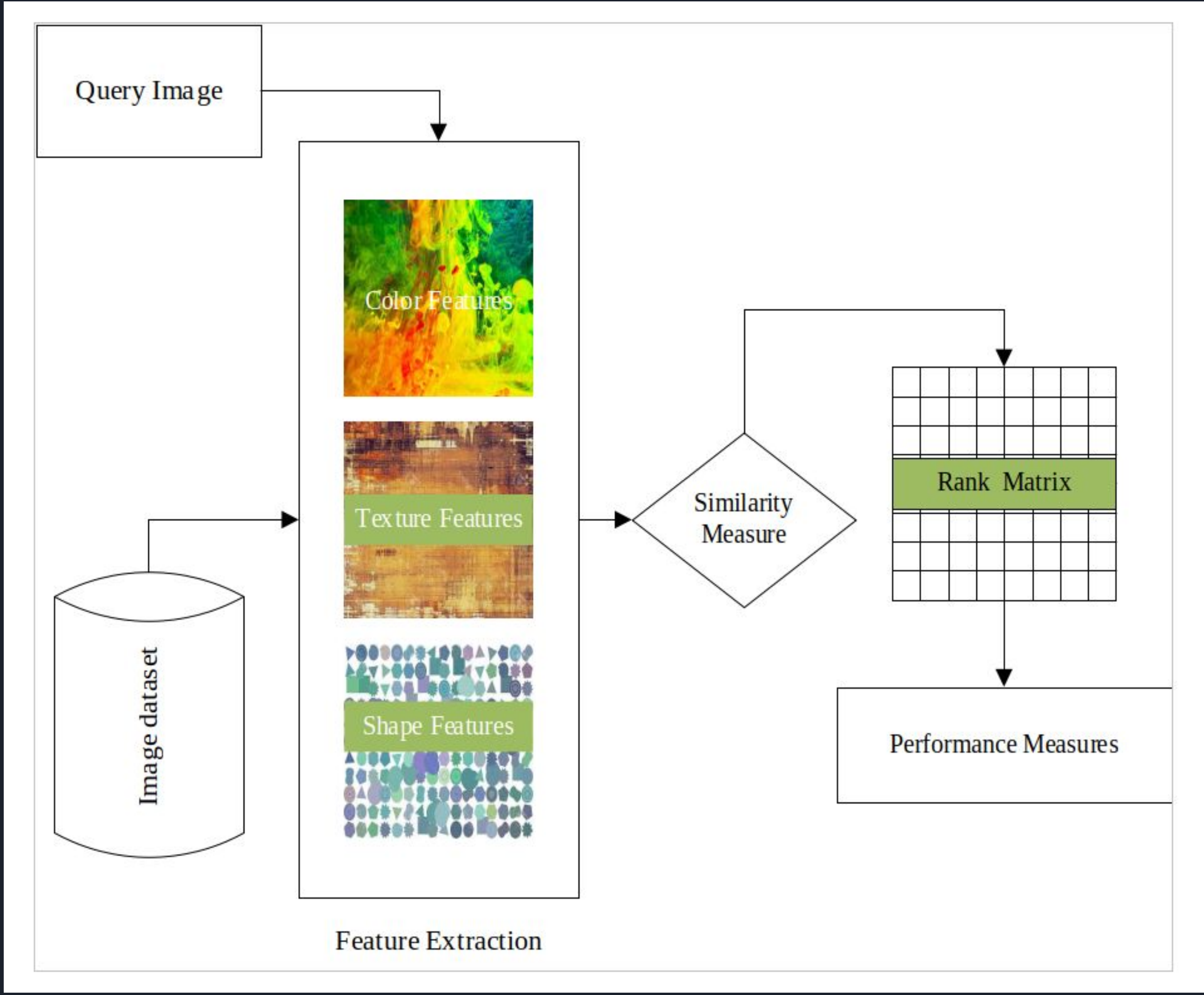




Content-based Image retrieval

- In a Content Based Image Retrieval (CBIR) System, the task is to retrieve similar images from a large database given a query image.
- The usual procedure is to extract some useful features from the query image, and retrieve images which have similar set of features.
- Features can be either handcrafted or extracted from a pre-trained deep learning model.
- LBP, ULBP and interchannel algorithms have been used for handcrafted features since they work on pixel values directly.
- Alexnet, which is a pre-trained deep learning model has been used to extract features using DCNN(Deep CNN).

Block dig for CBIR Using handcrafted features:



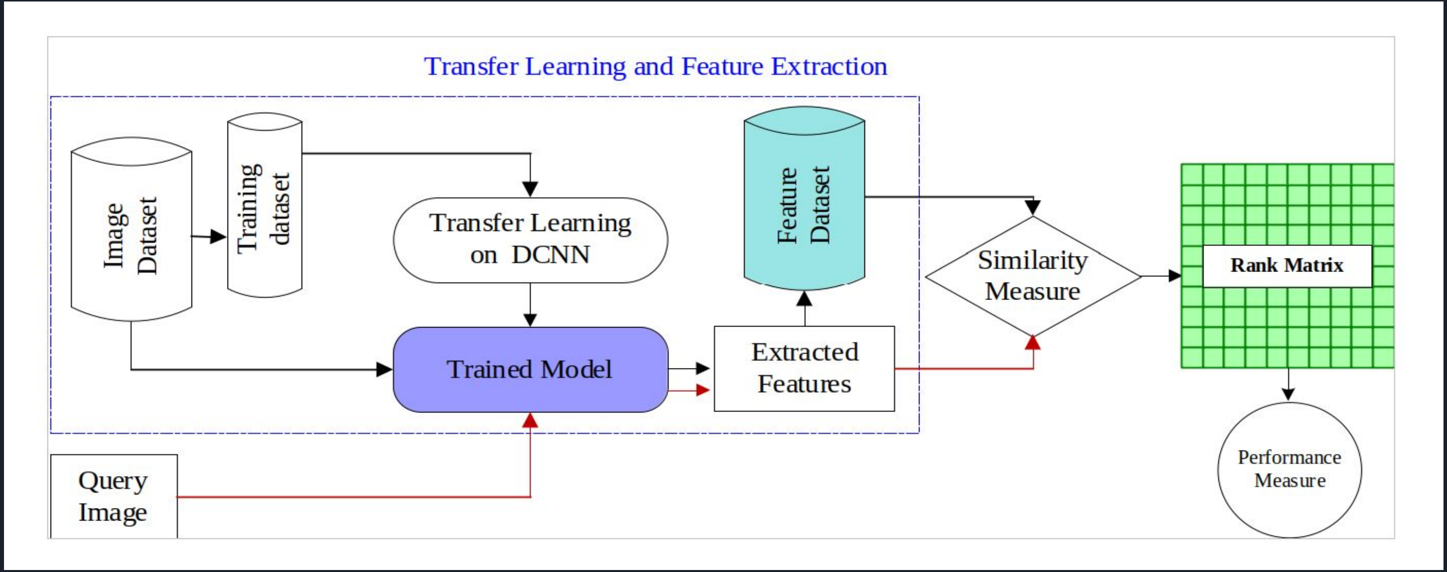
Handcrafted Feature extraction Algorithms

- **LBP :**
  - ➢ LBP/Local Binary Pattern is an effective texture pattern descriptor algorithm ,which describes the local texture patterns of an image.
  - ➢ LBP reflects the correlation among pixels within a local area (e.g., 3 × 3 area for $LBP_{8,1}$), which mainly represents the local information.

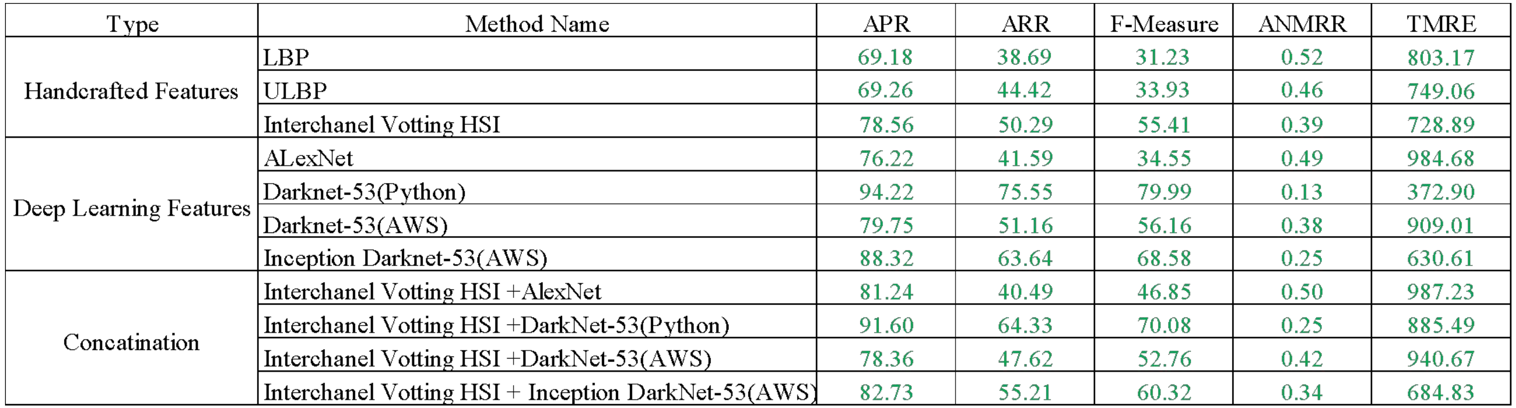
- **ULBP :**
  - ➢ Uniform local binary pattern is an extension to LBP,which can reduce length of the feature vector.
  - ➢ A LBP is called ULBP if the binary pattern consists of at most two 0-1 or 1-0 transitions.

- **INTERCHANNEL**:
  - ➢ The motivation behind inter-channel voting is that we obtain relationship among all three H,S,I Channels.
  - ➢ H,S,I channels are segregated individually into bins and their histograms are concatenated to perform Inter-channel voting.

General CBIR Framework by using DCNN features:



CBIR RESULTS:



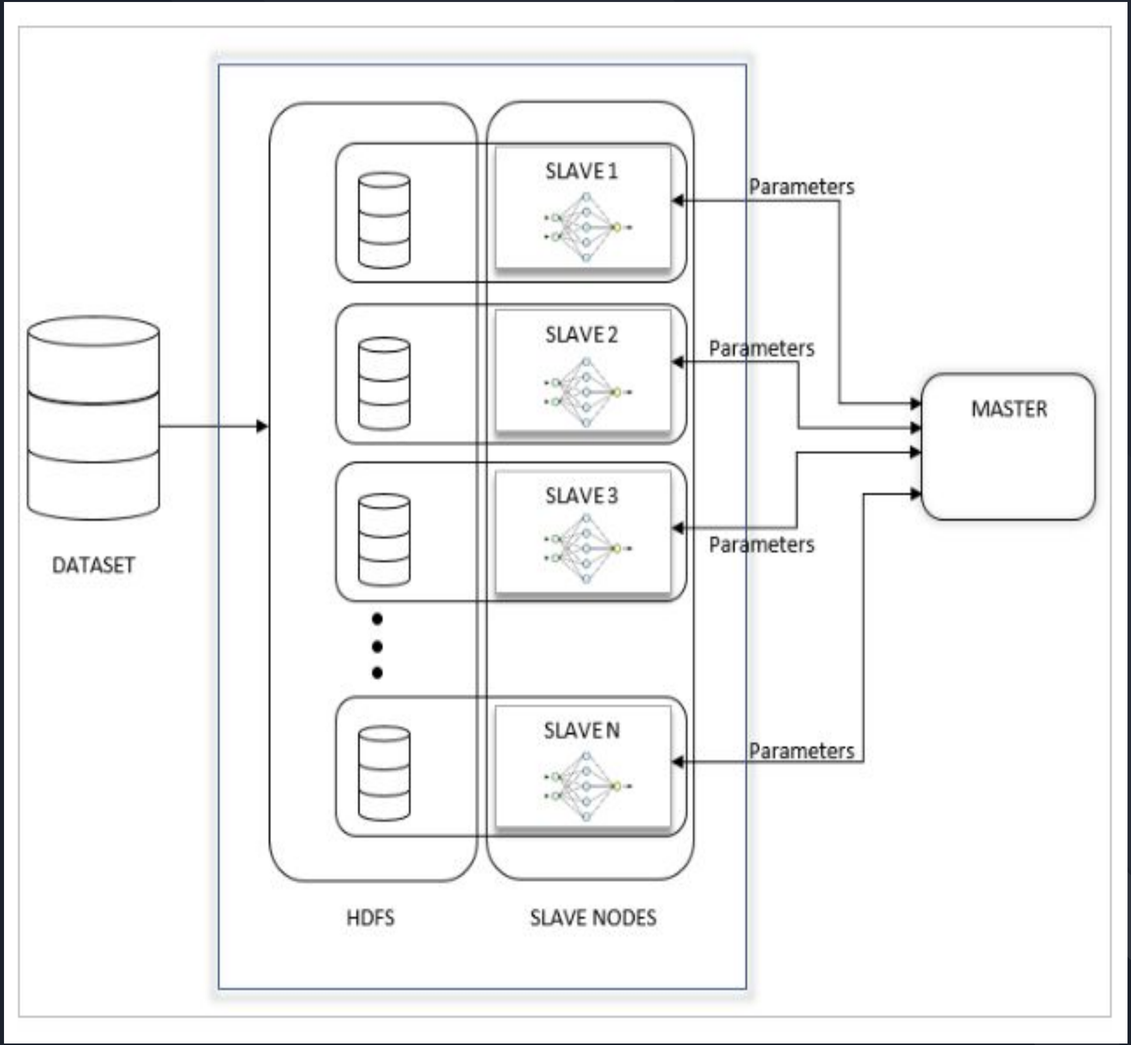
Type	Method Name	APR	ARR	F-Measure	ANMRR	TMRE
Handcrafted Features	LBP	69.18	38.69	31.23	0.52	803.17
	ULBP	69.26	44.42	33.93	0.46	749.06
	Interchanel Votting HSI	78.56	50.29	55.41	0.39	728.89
Deep Learning Features	ALexNet	76.22	41.59	34.55	0.49	984.68
	Darknet-53(Python)	94.22	75.55	79.99	0.13	372.90
	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

Distributed Deep Learning

- Deep learning is a growing field with huge number of applications and is proved to be effective in classification/recognition/detection tasks.
- With Distributed Deep learning, we can train a deep neural network model on a distributed cluster of computers.
- Training Deep learning models on a single system is often computationally expensive and time consuming which impacts it’s performance significantly.
- Distributed Deep learning brings down the execution time for large scale of images by distributing computation among slave nodes.

BLOCK DIG FOR DDL:



Types of executions for DDL in Elephas:

- **Synchronous:**
  The master receives the parameters from all the save nodes at the same time and calculates the updates and transfers the updates to all the slave nodes at the same time.Since all nodes are working together, if any of the nodes are slow then all the other nodes have to wait for them. Therefore, wasting their computational time.

- **Asynchronous:**
  The master receives the parameters from the slave node(s) and applies MUTEX locks on the Shared variable to perform operations based on the parameter received from the node(s). After performing operations on the parameters, the master sends the corresponding updates to the respective slave node(s).
  MUTEX Locks Causes a Computational overhead.

- **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, we show that memory overwrites are rare and that they introduce barely any error into the computation when they do occur.

Digit Classification using Distributed Deep learning:

- Digit classification is a classic example of using deep learning model for classification task.
- Distributed Deep learning would help the model train faster.
- We are using Distributed deep learning library known as elephas which makes our work lot easier.
- MNIST handwritten digit dataset has been used and the deep learning model is constructed using tensorflow keras.
- MNIST contains 70,000 images of handwritten digits: 60,000 for training and 10,000 for testing. The images are grayscale, 28x28 pixels, and centered to reduce preprocessing and get started quicker.
- Keras is a high-level user friendly neural network API. It works with deep learning frameworks like Tensorflow.

RESULTS FOR DDL:

		Accuracy				Time			
S.No.	MNIST	Normal	Synchronous	Asynchronous	Hogwild	Normal	Synchronous	Asynchronous	Hogwild
1	Single	0.9613999724	N/A	N/A	N/A	5.4744529724	N/A	N/A	N/A
2	1+4	N/A	0.8671594696	0.9402519058	0.2616642131	N/A	25.6060221195	21.5338511467	19.7548880577
3	1+15	N/A	🟨	0.0957276255	0.1009811601	N/A	🟨	40.7909429073	35.5843484402
4	Azure	N/A				N/A			

CONCLUSION

- Analysed the execution times of various distributed frameworks on multiple clusters by compressing and decompressing over 200 million images using compression algorithms such as LZW and Huffman.
- Performance of hadoop and spark has been observed under various circumstances.
- Successfully used distributed computing for content based image retrieval using Alexnet classifier.
- Understood the working of Distributed deep learning and used it for training networks for CBIR and Handwritten digit classification tasks.

REFERENCES

Digital Image Processing by Rafael C. Gonzalez 3rd Edition

[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.

THANK YOU!

Distribution of work

- Execution of Image compression algorithms
- Dataset creation, Execution of content-based image retrieval.
- Distributed Deep learning for digit classification,CBIR and establishment of VM’s in cloud environment