2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP - 2021)

An Improved Content Based Image Retrieval System using Unsupervised Deep Neural Network and Locality Sensitive Hashing

Prabavathy Balasundaram
Associate Professor, Department of CSE
SSN College of Engineering
Chennai, INDIA
prabavathyb@ssn.edu.in

Sriram Muralidharan
UG Student, Department of CSE
SSN College of Engineering
Chennai, INDIA
sriram18168@cse.ssn.edu.in

Sruthi Bijoy
UG Student, Department of CSE
SSN College of Engineering
Chennai, INDIA
sruthi18170@cse.ssn.edu.in

***Abstract*— Nowadays, many types of digital devices generate data in the form of text, image and video. Hence, it is essential to have a data archival system that maintains and accesses the data. In this context, Content Based Image Retrieval (CBIR) is the problem of searching for images in a large database of images. The CBIR involves feature extraction and identification of similar images. Existing works have utilized *supervised deep neural networks* to learn the features and used *exhaustive searching* technique to identify the similar images. However, since real datasets of images are unlabelled and large in size, the usage of supervised deep neural networks and exhaustive searching technique will be a time-consuming process. Thus, this paper proposes an Improved CBIR system that first utilizes an unsupervised deep learning approach to extract the features. Further, it utilizes Locality Sensitive Hashing (LSH) to reduce search space and K Nearest Neighbor (KNN) algorithm on the reduced search space to find the required number of similar images.**

Keywords— ***Content based image retrieval, Denoising Autoencoder, Locality sensitive hashing, KNN algorithm***

I. INTRODUCTION

Image retrieval [1] is a technique that searches and retrieves similar images from a database based on a query image. TBIR and CBIR are the two ways in which the similar images can be retrieved. In TBIR, each image in the database will be annotated. An annotation is basically the textual description of the image. Once the annotations for the images in the dataset and the query image are done, the annotation for the query image will be used to search for similar images. Example queries in TBIR would be *to search for flowers* or *to search for flowers which are inserted in the database on 21.04.2020*. Here, the annotation is the name of the image or the date of insertion of the image. The annotations may have misspellings, unexpressed emotions and feelings. Hence, it may affect the accuracy of the retrieved images. However, in CBIR, the content of the image itself is utilized to retrieve the similar images.

Fig. 1 shows the design of the traditional CBIR system. In this method, a query image will be given to retrieve the required number of similar images. The database has *n* number of images. The *Feature Extraction* process is used to find the feature vector for every image in the database based on its color and texture. Similarly, the feature vector is found for the query image. Further, the given query image is compared with each and every image in the database based on their feature vectors in order to retrieve the required number of similar images.

Existing works on the CBIR system extracted the feature of an image using any one of the low-level visual features such as color, texture, spatial layout or shape. Due to the diversity of images in the dataset, a single feature may not be sufficient to represent an image. Hence, few other works have attempted to improve the performance of the CBIR system by fusion of two or more low-level features.

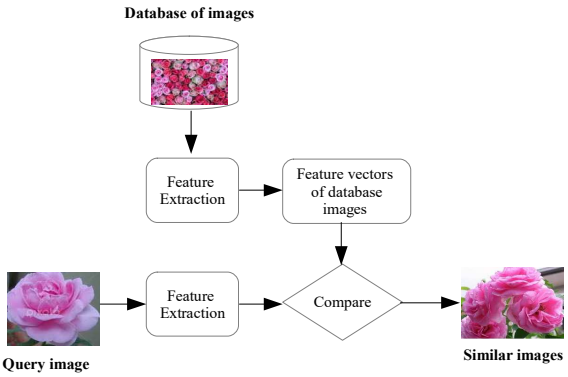

Fig. 1. Design of traditional CBIR system.

The recent research on CBIR has shifted to the use of deep neural networks [2] [3] in order to extract features. These works have shown good results on many datasets and outperformed the methods which have handcrafted the features subject to the condition of fine-tuning of the deep neural networks.

Generally, CBIR has two parts namely, feature extraction and identification of similar images. In the context of feature extraction, supervised deep neural networks can be used to extract the features. However, real datasets are large in size and unlabeled. Hence, the usage of supervised deep neural network demands a time-consuming labeling process [4].

In the context of identification of similar images, the traditional CBIR system compares the feature vector of the query images with the feature vector of every image in the database. However, this exhaustive search increases the number of comparisons, if the dataset is large in size. This

978-1-6654 -3277-1/21/$31.00 ©2021 IEEE

exponential increase in the number of comparisons reduces the efficiency of the CBIR system. Hence, it is necessary to improve the present CBIR system with efficient methods to extract the features and to identify the similar images. Thus, the objective of this paper is to propose an Improved CBIR (I-CBIR) system to handle large real-time datasets.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the design of the proposed I-CBIR system. Section 4 deals with the evaluation of the proposed system and section 5 conclude this paper.

II. EXISTING METHODS

Guo et al. [5] have proposed a CBIR system with ODBTC which is an improved Block Truncation Coding method. Experiments in this work have used a dataset of 1000 color images from Corel Photo Collections. The ODBTC method obtains the description of the image based on its content. Further, the Hamming distance measure was used to find the similar images and the proposed method provided the best average precision rate when compared to the other feature extraction methods.

Saritha et al. [6] have proposed a CBIR system using a deep learning process. This work has combined the features of color histogram, edge, edge directions, edge histogram and texture features. To constitute the feature vector, a set of feature maps were extracted from the input image by n-way convolutions of a deep belief network. Further, the Euclidean distance measure was used to find the distances between the different features in order to retrieve similar images. The proposed system achieves an accuracy of 98.6% and 96% for small (contains 1000 images) and large datasets (contains >$ 10,000 images) respectively.

Rajkumar et al. [7] have presented a CBIR System using a deep neural network. Experiments were carried out on a crawled dataset consisting of 10,000 images with 50 categories, each having 200 images. A Siamese neural network was used to extract the features. The extracted features were stored in a feature repository. The Chi-square similarity metric was used to find the distances between the features. The mean average precision was reported to be 98.62%.

Kuo et al. [8] have presented a CBIR System using a deep neural network. The experiment was carried out for the CIFAR-10 and CIFAR-100 datasets. The CIFAR-10 dataset contains 60,000 images with 10 categories, each having 6,000 images. The CIFAR-100 dataset contains 60,000 images with 100 categories, each with 600 images. A convolution neural network was used to extract the features. The extracted features were stored in a feature repository. Different distance metrics namely, Euclidean, Manhattan and Cosine distances were used to find the distances between the features. The mean average precision rate for the cosine distance measure was found to be better than the other two distance measures. The mean average precision using cosine distance for CIFAR-10 and CIFAR-100 were 0.707 and 0.244 respectively.

Weihong et al. [9] [10] [11] have proposed a scalable content-based image retrieval scheme using locality sensitive hashing (LSH). The experiment was carried out on 500,000 images crawled from the Web. An additional image set containing 50 categories, each category consisting of 100 images from COREL image CDs were used as the query set for performance evaluation. The proposed solution used three visual features - color, shape, and texture. Grid Color Moment, edge direction histogram and Gabor feature were adopted to extract the three visual features respectively. 238-dimensional feature vectors were employed to represent each image in the database. The proposed model was evaluated using precision and recall. A CPU speedup of up to 4.640 was achieved for the proposed solution against an exhaustive linear search model.

To summarize, all the above-mentioned existing CBIR systems have dealt with the datasets of size varying from a few hundred up to a million images. Further, the features for the images are extracted either by handcrafting or with the utilization of deep neural networks. All the datasets used were labelled datasets. However, currently, the number of images generated on the Internet is large and unlabelled. Hence, the CBIR systems which utilize supervised deep neural networks are insufficient to handle this unlabelled big data. However, extracting the features using deep neural networks outperformed the feature extraction using handcrafting. Hence, it is essential to build an Improved CBIR system that will perform well for real-time unlabelled datasets with an unsupervised deep neural network.

III. DESIGN OF THE PROPOSED IMPROVED CBIR SYSTEM

The proposed I-CBIR system consists of three processes namely, the feature extraction, the bucketization of feature vectors and the identification of similar images as shown in Fig. 2. First, the I-CBIR builds the denoising autoencoder model with a set of training images. Further, this model has been utilized to learn the feature vectors for the training set. The number of feature vectors will be more if the data is more. Hence, the LSH algorithm has been used to bucketize the feature vectors. These feature vectors are mapped to bins in different hash tables, each with a set of feature vectors of nearly similar images. When the query image is given to the I-CBIR system, it is mapped into several bins of different hash tables. Further, the feature vectors from all the bins are obtained to form a group. Subsequently, KNN algorithm has been applied on this group of feature vectors to find the required number of *k* similar images.

A. Feature Extraction

The I-CBIR system chooses to utilize a deep neural network to extract the features from the image as it outperforms the handcrafted feature extraction methods. Though convolutional neural network can be used to extract the features from the image, it requires labelled data to train the neural network. However, the I-CBIR system is designed to work for big data. As labeling task is costly and time-consuming for big data, an unsupervised deep neural network, namely, the denoising autoencoder is utilized in the proposed I-CBIR system to extract the features.

The denoising autoencoder corrupts the data on purpose by randomly turning some of the input values to zero. This helps to avoid the autoencoders from copying the input to the output without learning the features about the data. Hence, the autoencoder now has to remove the corruption to generate an output that is similar to the original input. Once the output is generated, it is compared with the original input and not with corrupted input. This process will be continued until the convergence, to minimize the loss.

Algorithm 1 explains the sequence of steps required to build the proposed feature extraction model. Subsequently, this noise image will be given as an input to the Denoising Autoencoder. It has encoder and decoder parts. The *encoder part* has CNN followed by ANN to generate the *feature vector*.

In CNN, a sequence of convolution and pooling layers are added to generate the feature maps. In this context, the convolution layer utilizes filters to detect the basic features like edges and lines (to name a few). Max pooling layers are added after every convolution layer to reduce the size of the feature map to its half.

In every convolution layer, filters are utilized to capture the hidden patterns in the image. For example, in the first layer, filter captures the basic patterns like lines, corners and dots (to name a few). In the subsequent layers, filters are increased to form complex patterns using the basic patterns. Hence, in this work, the number of filters is increased in every layer. In every convolutional layer, an activation function namely, Exponential Linear Unit (ELU) in (1) has been utilized to produce the negative outputs also. This helps the network to slowly adjust the weights and biases in the right directions. Further, the output of the convolution process is flattened to provide the input to the dense layer.

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & x <= 0 \\ x & x > 0 \end{cases} \qquad (1)$$

The dense layer is created with 32 neurons to provide the *feature* vector. The activation function used in this layer is *linear*. Basically, this function takes the input multiplied by its weight for each neuron. Hence, it creates an output value that is proportional to the input. The decoder has ANN followed by CNN. The dense layer of the encoder will serve as an input for the decoder. This supplies the input to another dense layer with 1024 neurons. Further, the output of the dense layer is reshaped into 2x2x256. This reshaped input is supplied to the sequence of convolutional transpose layers to reconstruct the image. Basically, the convolutional transpose layer reverses the convolution process and does the upsampling operations.

Further, mean squared error loss function is used to train the neural network through standard back propagation procedure. In the loss function, the output of the decoder is compared with the original input. An optimizer function Adamax is used to find the optimal weight to train the network. The hyper parameter, namely, the *number_of_epochs* is tuned manually based on loss / validation loss in order to avoid the overfitting of the network. Further, this optimal *number_of_epochs* is utilized to retrain the model.

Fig. 3 shows the detailed architecture of the proposed feature extraction model consisting of various layers with the number & size of filters, activation functions used and number & size of feature maps produced. The encoder part of the proposed feature extraction model is used to retrieve the feature vectors for all the images in the dataset.

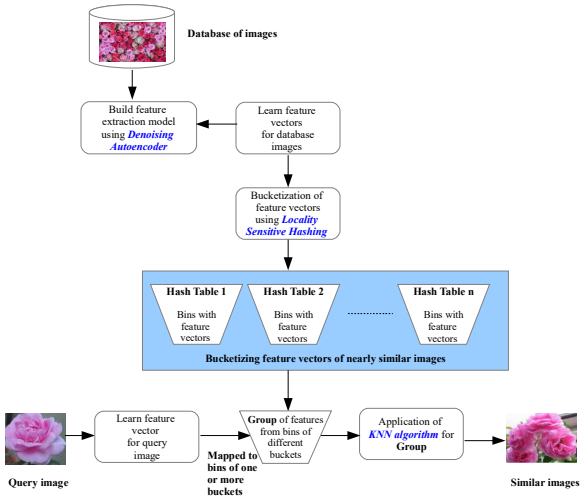

Fig. 2. Design of Improved CBIR System.

B. Bucketization of feature vectors with LSH Projection Method

In traditional CBIR system, the feature vector of a query image is compared with the feature vector of every image in the database. However, this has a huge impact on the query response time when the dataset is large. This query response time can relatively be reduced when the feature vector of the query image is compared with the feature vectors of only nearly similar images. Hence, the proposed I-CBIR system has utilized the LSH technique with random projection method [12] to bucketize the nearly similar feature vectors in a set of hash tables.

It is possible to find the nearly similar feature vectors by representing them in an n-dimensional space, based on their cosine distances. The cosine similarity between two images combines all the dimensions and returns a single value that is minimal for the same image. For example, Fig. 4 shows the cosine distances [13] between the feature vectors that are represented in red, green and blue colors in a 3-dimensional space. The feature vector which is represented in green color is more similar to the red color as the cosine distance between them is small when compared to the feature vector which is represented in blue color. The application of cosine similarity in the proposed model is due to the fact that it combines many dimensions that are learned implicitly, like, the shape of the object and the presence of eyes or legs, into a numeric value. As a result, the network will be able to classify an input image that does not belong to any of the training classes as belonging to the closest class based on cosine similarity.

Hence, the idea of this proposed work is to project the feature vectors in a multidimensional space with each dimension

representing a basic feature. However, projecting *n* feature vectors in a high dimensional space with *d* features would be complex. Hence, LSH with random projection method is used to project the feature vectors in k-dimensional space with *k* random vectors, where k<<d, while preserving the same cosine distances. Algorithm 2 explains the working of LSH using random projection method. Fig. 5 shows an example of a set of hash tables generated by LSH using random projection method. In each table, a different number of bins might be created with their corresponding feature vectors.

Once when the query image comes to the I-CBIR system, the proposed feature extraction model is used to generate the feature vector. Further, the feature vector will be hashed into the corresponding bins of the set of hash tables by using Algorithm 2. For example, if the feature vector for the query image is fv6, it is mapped into bin1 and bin2 in hash tables 1 and 2 respectively. Further, all the feature vectors (*fv*1*; fv*3*; fv*4*; fv*6*; fv*11) from bin 1 of hash table 1 and bin 2 of hash table 2 are grouped as nearly similar feature vectors.

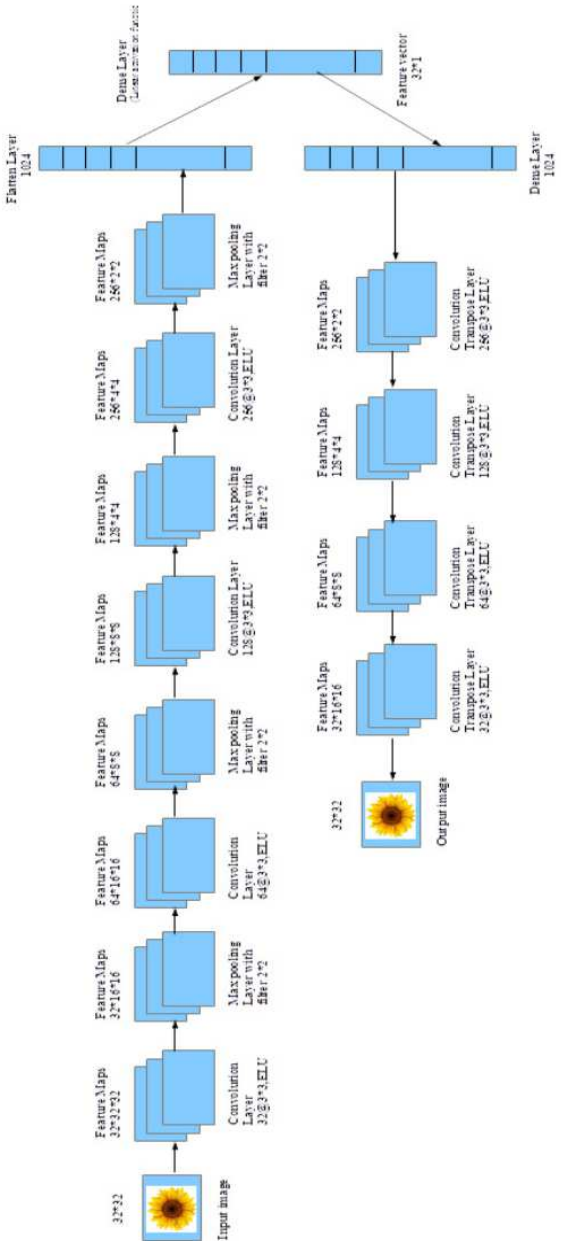

Fig. 3. Architecture of Feature Extraction Model

C. *Identification similar images using KNN algorithm*

The proposed I-CBIR system receives a query image and retrieves *n* number of similar images which are relevant to that. In order to implement this, the KNN algorithm [14] has been applied over the group of nearly similar feature vectors. This process is explained in Algorithm 3.

Algorithm 1: Proposed Feature Extraction model

Input:
Images from CIFAR-10 dataset with each image of size 32x32

Output:
Feature extraction model

```
for each epoch do
    for every image in training set do
        //Encoder part Add noise to the image ;
        Feature maps are generated with the application of
          sequence of convolution and max pooling layers
          from noisy image ;
        Flatten the output of the final convolution layer ;
        Provide this flattened output to the input layer with
          1024 neurons ;
        Dense layer is added with 32 neurons to generate the
          feature vector;
        //Decoder part Add another dense layer with 1024
          neurons;
        The output of the dense layer is converted into
          2x2x256;
        Sequence of convolution transpose layers are used to
          reconstruct the features in subsequent steps towards
          the reconstruction of original image;
        //Back propagation Original and the reconstructed
          image are compared to find the loss using Mean
          squared error loss function;
        Adamax optimizer function is utilized to find the
          optimal weight to perform back propagation;
        The hyper parameter namely, number_of_epochs is
          tuned manually based on loss /val_loss;
    end
end
The model is retrained with the optimal number_of_epochs;
```

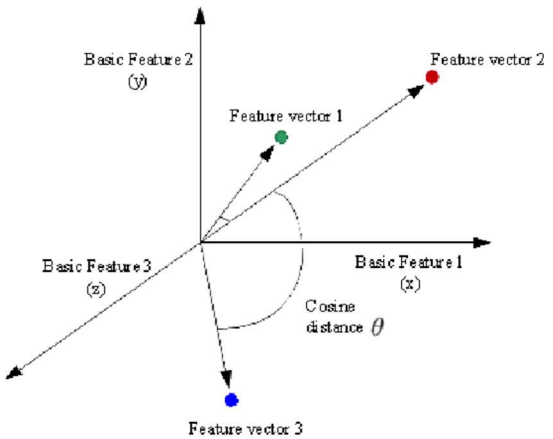

Fig. 4. Cosine distance between two feature vectors

IV. RESULTS AND DISCUSSION

The proposed I-CBIR has 3 processes, namely, the feature extraction model, the bucketization of feature vectors and the identification of the similar images. The entire system has been built using a computing machine with the configuration of Intel Core i5-8265U, 256GB M.2 PCIe NVMe SSD, 8GB

DDR4 RAM, 2400 MHz, NVIDIA 4 GeForce MX150 (2GB GDDR5).

The feature vectors are generated by the feature extraction model using denoising autoencoder [15]. Subsequently, they are bucketized with the LSH using random projection method into a set of hash tables. In order to bucketize, at first the random vectors were generated using the *Random module*. These random vectors are multiplied with the feature vectors using *dot product* and the results are stored as bit vectors which are lists. The feature vectors having the same bit vectors are mapped into the same bin in a hash table. The number of hash tables to be generated has been found empirically. The above-mentioned process was repeated to generate the required number of hash tables.

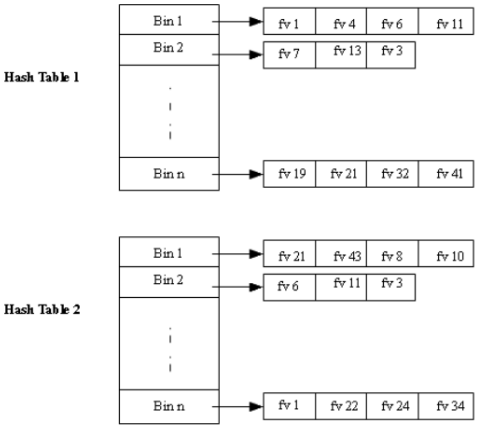

Fig. 5. Bucketization of feature vectors

Algorithm 2: Locality Sensitive Hashing using Random Projection method

Input:
Set of feature vectors
Output:
Set of hash tables, each with a set of bins associated with corresponding feature vectors

```
Create n hash tables;
for x in n do
    Generate k random vectors, each with length d;
    l=len(feature vectors);
    Bit Vector[l][k];
    // Compute the dot product of random vector with every
      feature vector to produce corresponding bit vector;
    for i in k do
        for j in l do
            Compute the dot product of the random vector i
              with the feature vector j;
            If the result of the dot product is positive,
              Append Bit Vector[j] with the bit value as 1 else
              append 0;
        end
    end
    Compare the bit vectors of the feature vectors;
    Assign the feature vectors into a bin, if their bit vectors
      are equal;
    Map the bin into x^th hash table;
end
```

Once there is a query image for the I-CBIR system, its bit vector is found with the LSH using random projection method. Further, the feature vectors of the bins relevant to the bit vector from different hash tables are grouped. Subsequently, the similar images are found from this group using the KNN algorithm.

Algorithm 3: KNN Algorithm

Input:
Group of nearly similar feature vectors
Output:
n similar images

```
q = feature vector corresponding to the query image;
for every feature vector f in the Group do
    Calculate the distance between f and q using Euclidean;
end
Sort the feature vectors in the Group in ascending order
  based on their Euclidean distances;
Choose the top n rows from the sorted feature vectors;
Display the images corresponding to the feature vectors;
```

A. Dataset Description

The proposed I-CBIR utilizes the CIFAR-10 dataset, MNIST Handwritten digits and MNIST Fashion datasets. CIFAR-10 consists of 60,000 images with 10 classes, each having 6,000 images of size 32x32. These classes include the images of airplanes, cars, birds, 25 cats, deer, dogs, frogs, horses, ships, and trucks. This dataset is loaded from Keras datasets. The MNIST handwritten digits dataset consists of 60,000 images of handwritten single digits between 0 and 9. MNIST Fashion dataset consists of 60,000 images with 10 different classes.

B. Evaluation

This section describes the performance of the proposed I-CBIR system in comparison with the traditional CBIR system using a set of common evaluation metrics.

1) Impact of the proposed I-CBIR system on the Number of comparisons:

Objective: To analyse the number of comparisons used to find the required number of similar images in traditional CBIR and I-CBIR systems.

When an image is being queried, it is firstly hashed into the LSH tables to retrieve all the relevant bins. These bins are used to retrieve a group of features vectors which serve as an input to the KNN algorithm. The count of feature vectors in the group determines the number of comparisons that have to be made to find the required number of similar images. This number of comparisons has been recorded. This process has been repeated for a query image from every image class and plotted as shown in Fig. 6.

It is clear from the graph that, in the traditional CBIR system, the number of feature vectors sent to the KNN algorithm is always 50,000. Hence, the query image has to be compared with all the 50,000 feature vectors. Whereas, in I-CBIR system, only the nearly similar feature vectors corresponding to the query image sent to the KNN algorithm. Since, the number of nearly similar feature vectors is much lesser when compared to the total, the comparisons to be made to find the required number of similar images is lesser in the I-CBIR system.

2) Impact of the proposed I-CBIR system on Response time:

Objective: To compare the response time to retrieve the required number of similar images given a query image in traditional CBIR and I-CBIR systems.

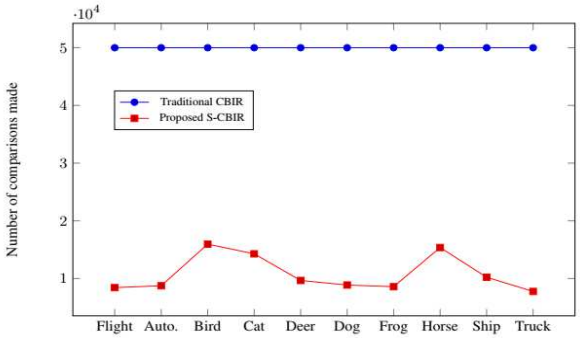

Fig. 6. Number of comparisons to find the required number of similar images

The response time is the total amount of time taken to respond to a query image with a set of similar images. The function *time.time()* has been utilized to retrieve the current system time before the query image arrives and after retrieving similar images. The difference between these times has been calculated as the response time. This experiment was repeated 5 times to calculate the average response time for the systems with LSH and without the usage of the LSH algorithm. Fig. 7 shows the average response times plotted for various cases with the increase in the number of retrieved images.

From the graph, it is seen that the response time for the CBIR system implemented *with LSH* (I-CBIR system) has an improvement when compared with the CBIR system implemented *without LSH* (CBIR system). Hence, the I-CBIR has an improvement in the response time. This is due to the fact of attempting to retrieve similar images, from the group of nearly similar images instead of total number of images.

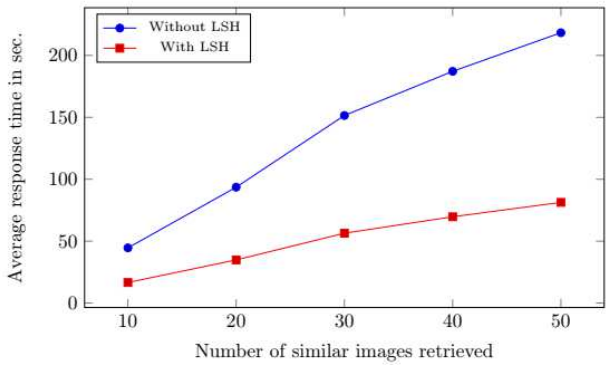

Fig. 7. Response time to find the required number of similar images

3) Impact of I-CBIR system on CPU Speedup with different datasets:

Objective: To study the performance of I-CBIR for different datasets in terms of CPU speedup time.

A query set consisting of 100 images from each of CIFAR-10, MNIST handwritten digits and MNIST Fashion dataset is considered to evaluate the performance. Response time to retrieve query set for each dataset is calculated and the average response time is found. This experiment was repeated 5 times to calculate the average response time for the CBIR and I-CBIR systems. The results are tabulated as shown in the Table. 1.

It is seen from the table that response time increases with the number of images. Further, it also observed that the proposed I-CBIR system was able to achieve a CPU speedup of up to 2 times when compared with the traditional CBIR system. CPU Speedup for the I-CBIR system is calculated as the ratio between response time of CBIR systems with LSH and response time without LSH.

TABLE I. TIME PERFORMANCE OVER DIFFERENT DATASETS

Samples	CBIR without LSH (in Sec)	I-CBIR with LSH (in Sec)	CPU Speedup
10	13.09	5.06	2.58
20	18.15	10.12	1.79
30	23.21	15.18	1.53
40	28.28	20.24	1.40
50	33.33	25.30	1.32

4) Impact of the proposed I-CBIR system on the Mean Average Precision Score:

Objective: To compute the mean average precision score for the I-CBIR system.

Mean average precision score was calculated using Label Ranking Average Precision (LRAP). Given a query image, the similar images are ranked label-wise based on the similarity score. For the query image of every class, similar images are found using label_ranking_average_precision method in scikit-learn package by passing similarity scores and labels. Mean average precision score will be the mean of average precision scores for all the image classes.

An experiment was carried out for the proposed I-CBIR model on the CIFAR-10 dataset to calculate the mAP for n random samples in order to retrieve 100 images. This n varies like 10, 20, 30, 40, and 50. Table. 2 shows the results of the experiment. From the table, it can be observed that an average mAP of 0.848 was obtained for the I-CBIR model, which is 16% higher than the system proposed by the work [8].

Fig. 8 shows a query image and 11 similar images retrieved by the proposed I-CBIR system. It also displays the *Cosine distance* between the query image and every retrieved image.

TABLE II. MEAN AVERAGE PRECISION VALUE OF I-CBIR SYSTEM

Samples	Average Precision without LSH	Average Precision with LSH
10	0.924980	0.898588
20	0.910343	0.864634
30	0.866277	0.838660
40	0.812474	0.816952
50	0.816209	0.823151

5) Impact of the proposed I-CBIR system on the homogeneity of the images:

Objective: To analyse the performance of the I-CBIR system for homogeneous images.

Image augmentation is the process of creating new training examples from the existing ones. To perform augmentation, the images were warped at different angles. Using this method, 10,000 images of cats and dog were taken from the dataset and transformed to obtain 30,000 augmented images. The model was trained by using the newly obtained images. Fig. 9 shows the similar images retrieved by the proposed I-CBIR system with augmentation. The results show that the given system can handle homogeneous images taken from different angles.

Fig. 8. Image retrieval in I-CBIR system.

Fig. 9. Image retrieval in I-CBIR system with image augmentation.

V. CONCLUSION

In the last few years, Content Based Image Retrieval (CBIR), has received wide attention. The proposed I-CBIR system has been implemented using an unsupervised deep neural network namely, Denoising Autoencoder to extract the features from the images. Further, it utilized the LSH using random projection method to bucketize the features into several groups of nearly similar features. Subsequently, the KNN algorithm is used to find *n* similar images from the relevant group. The proposed I-CBIR system was implemented for the CIFAR-10 dataset. The traditional CBIR and I-CBIR systems have been evaluated with the metrics, namely, response time, CPU speedup and the mean average precision score. The proposed I-CBIR system improves the response time and mean average precision score by 93.12% and 16% respectively when compared to the traditional CBIR system.

Conflict of Interest: The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We report that the submission is original work and is not under review at any other publication.

REFERENCES

[1] G. F. Ahmed, and R. Barskar, "A study on different image retrieval techniques in image processing", Journal of Soft Computing, vol. 1(4), pp. 247-251, 2011.

[2] J. Wan, D. Wang, S. C. Hoi, P. Wu, J. Zhu et al., "Deep learning for content based image retrieval: A comprehensive study", 22nd ACM International Conference on Multimedia, Florida: Orlando, 2014, pp. 157-166.

[3] M. Ramprasath, M. V. Anand, and S. Hariharan, "Image classification using convolutional neural networks", International Journal of Pure and Applied Mathematics, vol. 119(17), pp. 1307-1319, 2018.

[4] A. Latif, A. Rasheed, U. Sajid, J. Ahmed, N. Ali et al, "Content based image retrieval and feature extraction: A comprehensive review", Hindawi Mathematical Problems in Engineering, vol. 2019, pp. 1-21, 2019.

[5] J. M. Guo, and H. Prasetyo, "Content based image retrieval with ordered dither block truncation coding features", IEEE International Conference on Image Processing. Australia: Melbourne, 2013, IEEE. pp. 4006-4009.

[6] R. R. Saritha, V. Paul, and P. G. Kumar, "Content based image retrieval using deep learning process", Cluster Computing, vol. 22(2), pp. 4187-4200, 2019.

[7] R. Rajkumar, and M. V. Sudhamani, "Content based image retrieval system using combination of color and shape features, and siamese neural network", International Journal of Innovative Technology and Exploring Engineering, vol. 9(28), pp. 71-77, 2019.

[8] C. H. Kuo, Y. H. Chou, P. C. Chang, "Using deep convolutional neural networks for image retrieval", Journal of Electronic Imaging, vol. 2, pp. 1-6, 2016.

[9] W. Weihon, W. and Song, "A Scalable Content-based Image Retrieval Scheme Using Locality-sensitive Hashing", International Conference on Computational Intelligence and Natural Computing, pp. 151-154, 2009.

[10] J. Buhler, "Efficient large-scale sequence comparison by locality-sensitive hashing", Journal of Bioinformatics, vol. 17(5), pp. 419-428, 2001.

[11] S. Chafik, I. Daoudi, H. E. Ouardi, M. A. E. Yacoubi, and B. Dorizzi, "Locality sensitive hashing for content based image retrieval: A comparative experimental study", International Conference on Next Generation Networks and Services (NGNS). Casablanca, 2014, pp. 38-43.

[12] S. Hari, "Locality sensitive hashing for similar item search", Website https://towardsdatascience.com/locality-sensitive-hashing-for-music-search-f2f1940ace23, 2018.

[13] S. Prabhakaran, "Cosine similarity - Understanding the math and how it works (with python codes)", Website https://www.machinelearningplus.com/nlp/cosine-similarity/, 2018.

[14] O. Harrison, "Machine learning basics with the K-nearest neighbour algorithm," Website https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, 2018.

[15] F. Chollet, "Building Autoencoders in Keras", Website: https://blog.keras.io/building-autoencoders-in-keras.html,