A Deep Learning Based Interval Type-2 Fuzzy Approach for Image Retrieval Systems

Highlights

A Deep Learning Based Interval Type-2 Fuzzy Approach for Image Retrieval Systems

Yosr Ghozzi,Tarek M. Hamdani,Hani Hagras,Khmaies Ouahada,Habib Chabchoub,Adel M. Alimi

- An interval type-2 beta fuzzy membership function with deep features extracted.
- The Multi-Variational Auto-Encoder is used for dimension reduction.
- Type-2 fuzzy similarity measures are used to ameliorate the retrieved images.
- Performance of the strategy after combining Deep fuzzy features with dimensionality reduction.

A Deep Learning Based Interval Type-2 Fuzzy Approach for Image Retrieval Systems

Yosr Ghozzi[a], Tarek M. Hamdani[b], Hani Hagras[c], Khmaies Ouahada[d], Habib Chabchoub[e] and Adel M. Alimi[a]

[a]University of Sfax, National Engineering School of Sfax (ENIS), Sfax, 3038, Tunisia
[b]University of Monastir, Institut superieur d'informatique de Mahdia , Mahdia, 3038, Tunisia
[c]University of Essex, School of Computer Science and Electronic Engineering, Essex, 5B.524, UK
[d]University of Johannesburg, Faculty of Engineering and the Built Environment, Auckland Park Kingsway Campus, B2 Lab 107, South Africa
[e]University of Science and Technology, College of Business, Al Ain, United Arab Emirates,

ARTICLE INFO

Keywords:
Interval-Type-2 Fuzzy Sets,
Deep learning,
variational Auto-encoder,
reduction dimensionality,
Function Beta,
Image Retrieval.

ABSTRACT

Deep learning, that one of its key benefits is automated feature extraction, has become a principal solution for computer vision. This paper presents a Deep Type-2 Beta Fuzzy (DT2F) approach for Content-Based Image Retrieval (CBIR) systems. Firstly, the suggested architecture uses Inception-ResNetv2 a pre-trained deep learning model on Image-Net data as a feature extractor. Secondly, the obtained feature space is fuzzified to handle the uncertainties associated with the extracted values of deep features. Thirdly, the reduction dimensionality step is efficiently applied using a Multi-Variational Auto-Encoder (MVAE) to reduce computational complexity and achieve better performance. Ultimately, we retrieve images using the nearest neighbors rule based on type-2 fuzzy similarity to having higher proximity sensitivity. Extensive experimentations were accomplished on various image-retrieving datasets of different scales the proposed system achieved an average precision of 97.15% exceeding other state-of-the-art methods over many systems on Corel datasets, which can open the door for several hybridization breakthroughs in the area of image retrieval.

1. Introduction

Visual similarity is the main purpose of the approach in content-based image retrieval (CBIR) [2], [52]. Measuring visual similarity between images described by feature descriptors is one way in which visual content similarity can be measured. The search performance of a CBIR system depends critically on feature extraction and similarity measures. The development of a CBIR system that simulates human vision is a very difficult and complex process. Indeed, a major challenge for CBIR systems is the semantic gap between the low-level visual information extracted from images and the high-level information perceived by human evaluators. Traditional methods proceed the low-level or high-level feature extraction and use handcrafted methods to reduce this gap. The image contents on behalf of the low-level visual descriptor, are described by a feature vector [5]. Various research works, including [57, 25, 5], have also been guided every so often to introduce the progress in content-based image retrieval. We need to design a feature extraction framework that combines low-level and high-level features to bridge this gap without using hand-crafted features. After 10 years, we observed a switch from hand-craft to deep learning (DL) in functional representation. For representing features, deep learning is extremely powerful and can fully map high-level and low-level information while extracting meaningful insights, which are modeled after the human brain. Since the success of DL depends above all on classification tasks, e.g. the emblematic success of the ILSVRC'12[2] challenge [26]. ImageNet-based convolutional networks are also very powerful visual representations called "Deep Features", whose use in content searching has recently shown very promising results [20] which inspired us to work on this axis to solve the problem of CBIR. Pre-trained deep learning neural networks are the latest developments of the recently applied convolutional neural networks, demonstrating high accuracy and good results in many research areas. The greater ability of pre-trained networks emanates from training on large images with many classes. Over the past decade, great strides have been made in harnessing the power of deep learning for content-based image retrieval [55, 37, 34, 21, 17, 39, 10, 12]. Indeed, in [12] Gkilos et Al., present a process that gives image retrieval features by the latest architectures of pre-trained models. In [46], the authors used a deep CNN model to obtain feature representations while activating the convolutional layers. They have proposed a model retraining method for learning more efficient convolutional representations for Content-Based Image Retrieval.

Motivation: The motivation behind this study stems from the acknowledged limitations of existing deep learning methods, particularly in terms of robustness and handling uncertainty in high-dimensional data. To address these challenges, the study incorporates feature normalization to maintain robustness. Additionally, the utilization of fuzzy logic is considered a technique to provide more flexible representations for effectively handling the underlying uncertainty in the data. The motivation is further reinforced by existing

ORCID(s):
[1]Email:yosr.ghozzi@regim.usf.tn
[1]URL:https://sites.google.com/regim.usf.tn/adel-alimi

[2]https://image-net.org/challenges/LSVRC/2012/

works, such as the deep type-2 fuzzy logic system proposed by Ravikiran et al. in [7] and the Deep Fuzzy Hashing Network (DFHN) method presented by Lu et al. in [29] to overcome weaknesses in deep hashing approaches.

Innovation: The primary innovation of this research lies in presenting an interval type-2 fuzzy deep learning framework as a novel feature extraction method. The hybridization of deep and fuzzy methods is positioned as a solution to several problems, leveraging the advantages of type-2 fuzzy sets in effectively handling uncertainty. The study aims to address the limitations associated with high-dimensional data by emphasizing the essential task of dimensionality reduction following the feature extraction process. Classic methods like principal component analysis (PCA) and Auto-Encoders (AE) are acknowledged for their role in reducing data dimensionality. The ultimate goal of the study is clearly defined as enhancing search efficiency, reducing data size, minimizing storage requirements, and decreasing query processing time in the context of Content-Based Image Retrieval (CBIR). Previous works, such as in [54] and in [46], are referenced to highlight the ongoing efforts in this direction. Distance (similarity) measurements are identified as crucial for CBIR systems' efficiency, with Euclidean distance and Manhattan distance being common metrics. The successful application of Fuzzy Logic in image retrieval systems is recognized, citing instances in various applications. The study introduces the use of interval type-2 similarity measures to represent multiple levels of relevance, offering a more flexible approach than absolute similarity. The optimization of interval type-2 similarity measures is emphasized for better comparison of deep fuzzy features. The experimental validation on three standard datasets demonstrates the proposed system's high performance for image retrieval. The study commits to empirically proving the efficiency of the proposed method in real-time CBIR contexts compared to other deep fuzzy methods.

Contribution: The originality of this study can be elaborated more specifically as follows:

1) Pretrained deep neural networks InceptionResNetV2 can be used to extract rich and transferable representations from fully connected layers from images.

2) Computing the fuzzy deep features using the lower and upper membership degrees of interval type-2 beta fuzzy sets which improved the novel method proposed by this paper, and can alleviate the negative impact of parameter uncertainties on the extraction feature step.

3)A study of dimensionality reduction methods was explored to minimize the computational complexity (from 1536 to 100 features) and increase the performance (from 95.35% without reduction to 96.94% with reduction) on Corel dataset in terms of the precision of retrieval. Comparing these results with other existing works [52, 2, 46, 41, 19, 1] in the literature, we notice that the resulting approach MVAE-DS-IR is efficient. Finally, the final degree of belonging of the feature vector extraction is obtained after this reduction.

4) The similarity between images was evaluated using type-2 fuzzy similarity measures based on fast library nearest neighbor (FLANN).

Sections of the manuscript: This paper is structured as follows: The theoretical foundations of the used methods are presented in section II. Section III is reserved for the details of the adopted research methodology. Section IV presents the results of extensive experiments and their discussions. The conclusion is presented in section V.

2. Preliminaries

2.1. Literature review of image retrieval

Thanks to the effective techniques and methodologies from artificial intelligence, which can intelligently solve complex problems related to image retrieval systems, researchers have shown significant interest in utilizing these approaches. We aim to provide a succinct survey of several notable research works in this field, highlighting their objectives to not only improve the accuracy and efficiency of image retrieval systems but also to optimize their overall performance. For instance, AI-driven techniques have been used to enhance the precision and speed of image retrieval systems, leading to improved user experiences and more efficient data management.

2.1.1. Recent Literature of Feature Extraction Techniques for Image Retrieval

Numerous traditional methods have been proposed to address the challenge of feature extraction in continuous image description, utilizing handcrafted features [46]. Handcrafted features are the ones that were manually designed and engineered. The most used handcrafted features are the histogram of oriented gradients (HOG), the local binary pattern (LBP), the Gabor filter, and other features such as color histograms, color moments, wavelet statistical histograms, and local phase quantization [31]. Recent work has shown that by using methods similar to trained networks, creating deep learning features to represent image content for content-based image retrieval (CBIR) leads to better retrieval performance. Typically, for the final layers, the trained networks are replaced with random networks, allowing pre-trained weights to be used separately. The final layers are trained using some final features in an attempt to model the new output signal or label [11]. However, a weakness exists with this type of training and data: some images that the trained network has to classify are not similar to the available data, causing the network parameters to change incrementally to correct the classification. Table 1 gives a performance comparison of some of the results obtained in CBIR. These approaches, however, struggle to adapt to feature ambiguities and uncertainties, creating inefficiencies in retrieval systems. Consequently, researchers are increasingly turning to Fuzzy Logic as a valuable tool for handling such uncertainties [19].

Table 1
Relevant Related Works on Technique of Feature Extraction for Image Retrieval

Refs	Year	Techniques of feature extraction	Data	Performance Indicators	Results
[19]	2020	fuzzy class membership	Corel-1k/5k/10k, CIFAR-10	Precision	-
[21]	2020	combining DCNN and Fuzzy c-means	Oxford5K, Inria Holidays	mean average precision (mAP)	83%, 86%
[30]	2021	InceptionResNetV2 pre-trained model	Corel, Caltech	Average precision (mAP)	93.39%, 81%
[1]	2021	fusion of handcrafted and deep features (VGG-19, GoogLeNet)	Caltech-256, ALOT(250), Corel1K, Cifar-100/10	Precision	66%, 99%/95%, 99%, 99%, 99%
[11]	2022	CNN (AlexNet)	Corel-1k/10k, Caltech-101, Scene-67	mean average precision (mAP), Accuracy	mAP=97.04%, P=98.81%, P=96.09%
[9]	2023	GoogleNet + Gabor	Harbour and Iceberg class	mAP	63.3%
[40]	2024	fusion handcraft and DCNN	Corel-1/5k	Precision	96,68%, 94,56%

2.1.2. Synthesis

Extracting good features is crucial because of the difficulty for machines to understand the content of images before analyzing low-level structures. Manually designed features, or handcrafted features, are developed by experts based on their knowledge of the relevant fields. They are quick and easy to use, but their effectiveness may be limited. In contrast, deep learning features, extracted from the intermediate and final layers of neural networks, are very effective in describing image content. However, these features are complex and their extraction is time and memory intensive. Recent methods generally rely on the combination and fusion between the two methods handcraft and deep learning. Fuzzy logic-based feature extraction strikes a balance by providing robust and efficient features that improve the overall performance of CBIR systems. Our proposed fuzzy deep features build upon this concept to offer a comprehensive solution to the challenges in feature extraction.

2.2. Deep Feature Extraction Using Pre-Trained CNN Models

2.2.1. Convolutional Neural Network

CNNs are a class of deep neural networks that use convolutional layers to filter inputs to obtain useful information. A convolutional layer in a CNN applies convolutional filters to the input to compute the output of neurons connected to local regions of the input. It helps extract spatial and temporal features of images. Convolutional Layers in CNNs use weight-sharing techniques to reduce the total number of parameters [4].

2.2.2. Pre-trained models

In general, CNNs perform better on large datasets than on small datasets. Transfer learning can be used when it isn't possible to create a big training dataset. Learning transfer has become more popular with deep learning, especially in convolutional neural networks. Not only does it effectively reduce training time, but also improves the accuracy of the models designed for jobs with minimal or inadequate training data. The learning transfer can be used as a pre-trained model to extract feature vectors.

A pre-trained model is a model inspired by the process with which he learned to solve a specific problem similar to the one you want to solve. This approach can save significant time and resources compared to training a new model from scratch. But, training is computationally expensive and it is common to import while using different models (VGG, ResNet, Inception, etc.). A comprehensive review of the performance of pre-trained models on computer vision problems using ImageNet data [36] is provided by Canziani et al. [6]. This motivated us to use a pre-trained model trained on the ImageNet dataset containing over 14 million images belonging to over 20,000 classes.

2.3. Fuzzification of Feature Vector

After the feature extraction step, and for a better result, we move on to the feature standardization step. The standardization of feature vectors is widely used in machine learning. It is used to eliminate the effects of scale, rotation, and translation between all the features using the fuzzy logic model to correctly determine the relevant information of the image, which will greatly increase and improve the precision rate of the proposed system. This step consists of normalizing the features using the fuzzification method. A fuzzy logic system has been widely used in image processing. According to [33], by conducting various pre-processing on the data, it is possible to make the model learn faster. Indeed, fuzzification (normalization) of data allows for the simplification of a problem that would otherwise need more in-depth learning. Fuzzification, in particular, allows the data to be re-ordered in the same order of magnitude, simplifying a portion of the work to be completed. Thus, it is proven that a neural

network trained using normalized input converges to an optimum quicker than one trained with the same non-normalized data [43]. Many standardization techniques exist. Following our research work in [52] we will use interval type-2 beta fuzzy (IT2BF) MF. The beta functions' primary significance stems from their ability to mimic a wide range of common functions (triangular, trapezoidal, or gaussian shapes) [3].

2.4. Feature Dimensionality Reduction

Machine learning algorithms are based on factors known as variables. The more characteristics there are, the more challenging it is to visualize and interact with training data. Sometimes most of the characteristics are correlated resulting in redundancy. This is where dimensionality reduction algorithms come in handy in tackling the scourge of dimensionality. The dimension reduction techniques can be classified by linear aspect or not of the methods [53].

2.4.1. Principal Component Analysis (PCA)

PCA one of the most popular techniques for processing, compressing, and visualizing multidimensional data originates in [18]. Given a set of vectors y_i, $i = 1, ..., N$ of dimension d, the PCA consists in looking for the orthogonal projection axes along which the variance is maximum. The optimal approximation, in the sense of the mean squared error, of a vector y_i by a $\tilde{t_i}$ vector of dimension $r < d$ is given by:

$$W_r^t = (y_i - \mu) \tag{1}$$

where μ is the mean of y_i and W_r is the projection matrix composed of the r first eigen vectors of the data covariance matrix $\sum_y$, corresponding to the r largest eigenvalues given in descending order λ_i, $i = 1, ..., r$ The reduced data covariance matrix is diagonal of elements λ_i, $i = 1, ..., r$. The quadratic error of the approximation is given by the sum of the separated eigenvalues (the smallest):

$$e^2 = \sum_{i=r+1}^{d} \lambda_i \tag{2}$$

The choice of r can be based on equation 2, or in an equivalent manner on the choice of a threshold p between 0 and 1 such that:

$$\frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq p \tag{3}$$

Although PCA is a widely used technique in data analysis, it has the disadvantage of relying only on a very narrow geometric approach.

2.4.2. Auto-Encoder

Although principal component analysis can reduce dimensions, PCA transforms data using linear algebra. With their nonlinear activation function and numerous layers, Auto-Encoder approaches, on the other hand, can conduct nonlinear transformations. Using an Auto-Encoder to create numerous layers is more efficient than using PCA to create a large transformation [49]. As a result, when the data is complicated and non-linear, auto-encoder approaches reveal their value. Furthermore, a seminal article by Geoffrey Hinton (2006) in [14] demonstrated that a trained Auto-Encoder produces a lower error and better cluster separation than the first 30 major components of a PCA. On the other hand, an Auto-Encoder (AE) [49] is a sort of neural network that is trained to duplicate its input to its output. It converts the input to a reduced-dimensional latent space before codifying the latent representation to the output. An Auto-Encoder

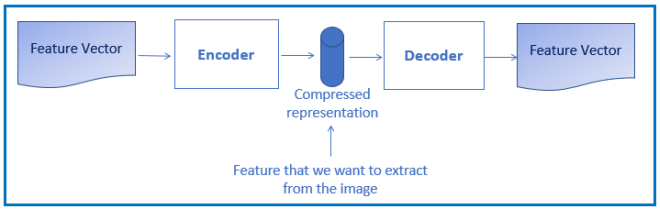

Figure 1: Auto-Encoder for Features Reduction

(AE) [49], learns to compress data by reducing reconstruction error. Figure 1 illustrates the simple architecture of an Auto-Encoder with an Encoder-Decoder structure. An Auto-Encoder includes an input layer, a hidden layer with a finite number of units (at least one), and an output layer composite of the same unit number as the input layer. Examples are regularized Auto-Encoders (AE) [24]: Sparse, Denoising, and Contractive.

However, two things must be born in mind. To begin with, a substantial reduction in dimensionality without sacrificing reconstruction often comes at a cost: the latent space lacks interpretable and exploitable structures. Second, in most cases, the goal of dimensionality reduction is to fight against the scourge of dimension while maintaining the quality of information (altered to achieve the ultimate goal of noise dimensionality reduction all while must be careful).

2.5. Image Retrieval

2.5.1. k-Nearest Neighbors Approximation

The k-Nearest-Neighbors (kNN) algorithm is a non-parametric and supervised classification method introduced in [32]. It is widely used in classification in general and image segmentation, in particular. It is based on a simple and intuitive principle of grouping people together according to their neighborhood. The KNN follows two main elements:

1. the number of the closest cases (K) to use and a metric to measure the nearest neighbor.

2. The value of K is specified each time the algorithm is used as it determines the number of the existing cases that are considered to predict a new case.

The KNN is based on the distance concept. A distance measure is required to calculate the similarity rate; the latter is both important and arbitrary because the choice of metric significantly impacts the quality of the forecasts. The k nearest neighbors method groups pixels by their proximity: each

point has designated the class with the most representation among its k closest neighbors.

This method requires the establishment of a similarity measure and the determination of the number of neighbors to be considered, as well as a training set representing the different classes. The k-NN algorithm necessitates retaining all of the points of the learning base in memory, resulting in expensive storage (in O(n)).

2.6. Similarity by Distance and Similarity Measures

Human perception of similarity can be modeled by an appropriate distance measurement in a multidimensional metric space. A normalized distance is defined mathematically as a function with a value in the range [0; 1] and which satisfies the three conditions listed below.

Definition: The properties of a distance measure between three sets X, Y and Z of fuzzy sets are suggested in [50], as shadows:

- Property of Reflexivity: If $d(X, Y) = 1$, then $X = Y$
- Property of Symmetry: $d(X, Y) = d(Y, X)$
- Property3. Transitivity: If $X \subseteq Y \subseteq Z$ then $d(X, Z) \leq d(X, Y)$ and $d(X, Z) \leq d(Y, Z)$
- P4. Overlapping: If $X \cap Y = \varnothing$ then $S(X, Y) > 0$ otherwise $S(X, Y) = 0$

Many distance metrics exist in the literature (defined for scalar, set, vector values, etc.) including absolute difference, cosine, Dice, Jaccard, Manhattan, Hamming, Euclidean, etc. The principles of similarity and distance measurement are both related to proximity (see equation 4). The value of similarity can range from -1 to 1. A unit similarity measure between two fuzzy sets A and B indicates that the two sets are comparable, whereas a similarity value of -1 indicates that the two sets are opposed.

In [52], we used three type-2 fuzzy similarity metrics: IT2FSM1, IT2FSM2 and IT2FSM3. They are defined as:

Interval Type-2 Fuzzy Similarity Measure 1 (IT2FSM1) The Interval Type-2 Fuzzy Similarity Measure is the mean of Interval Type-2 Fuzzy-upper Nearness Measure and Interval Type-2 Fuzzy-lower Nearness Measure. Since measure similarity is :

$$Similarity = 1 - Distance \tag{4}$$

Then

$$IT2FSM1 = 1 - \frac{(\bar{d} + \underline{d})}{2} \tag{5}$$

where

$$\bar{d}_{\tilde{\cong}_{B,\epsilon}}(\tilde{X}, \tilde{Y}) = (\underbrace{\sum_{\bar{\mu}_{\tilde{A}} \in H_{B,\epsilon}(Z)}} \left|\bar{\mu}_{\tilde{A}}\right|)^{-1} \underbrace{\sum_{\bar{\mu}_{\tilde{A}} \in H_{B,\epsilon}(Z)}} \left|\bar{\mu}_{\tilde{A}}\right| \frac{min(\left|\bar{\mu}_{\tilde{A}} \cap \bar{\mu}_{\tilde{X}}\right|, \left|\bar{\mu}_{\tilde{A}} \cap \bar{\mu}_{\tilde{Y}}\right|)}{max(\left|\bar{\mu}_{\tilde{A}} \cap \bar{\mu}_{\tilde{X}}\right|, \left|\bar{\mu}_{\tilde{A}} \cap \bar{\mu}_{\tilde{Y}}\right|)} \tag{6}$$

and

$$\underline{d}_{\tilde{\cong}_{B,\epsilon}}(\tilde{X}, \tilde{Y}) = (\underbrace{\sum_{\underline{\mu}_{\tilde{A}} \in H_{B,\epsilon}(Z)}} \left|\underline{\mu}_{\tilde{A}}\right|)^{-1} \underbrace{\sum_{\underline{\mu}_{\tilde{A}} \in H_{B,\epsilon}(Z)}} \left|\underline{\mu}_{\tilde{A}}\right| \frac{min(\left|\underline{\mu}_{\tilde{A}} \cap \underline{\mu}_{\tilde{X}}\right|, \left|\underline{\mu}_{\tilde{A}} \cap \underline{\mu}_{\tilde{Y}}\right|)}{max(\left|\underline{\mu}_{\tilde{A}} \cap \underline{\mu}_{\tilde{X}}\right|, \left|\underline{\mu}_{\tilde{A}} \cap \underline{\mu}_{\tilde{Y}}\right|)} \tag{7}$$

where $H_{\tilde{\cong}_{B,\epsilon}}(Z)$ is the set of fuzzy tolerance classes, as described in [52].

Interval Type-2 Fuzzy Similarity Measure 2 (IT2FSM2) According to Jaccard's similarity and Henry's metric-free description-based nearness measure, IT2FSM2 measure is defined by Ghozzi et al. in [52] as follows:

$$IT2FSM2 = 1 - \frac{(\bar{d} + \underline{d})}{2} \tag{8}$$

where

$$\bar{d}_{\tilde{\cong}_{B,\epsilon}}(X, Y) = (2 \times \underbrace{\sum_{\bar{\mu}_A \in H_{B,\epsilon}(Z)}} \left|\bar{\mu}_A\right|)^{-1} \underbrace{\sum_{\bar{\mu}_A \in H_{B,\epsilon}(Z)}} \left|\bar{\mu}_A\right| (\frac{\left|min(\left|\bar{\mu}_A \cap \bar{\mu}_X\right|, \left|\bar{\mu}_A \cap \bar{\mu}_Y\right|)\right|}{max(\left|\bar{\mu}_A \cap \bar{\mu}_X\right|, \left|\bar{\mu}_A \cap \bar{\mu}_Y\right|)} + \frac{\left|min(\left|\bar{\mu}_A \cap \bar{\mu}_X\right|, \left|\bar{\mu}_A \cap \bar{\mu}_Y\right|)\right|}{2 - max(\left|\bar{\mu}_A \cap \bar{\mu}_X\right|, \left|\bar{\mu}_A \cap \bar{\mu}_Y\right|)}) \tag{9}$$

and

$$\underline{d}_{\tilde{\cong}_{B,\epsilon}}(X, Y) = (2 \times \underbrace{\sum_{\underline{\mu}_A \in H_{B,\epsilon}(Z)}} \left|\underline{\mu}_A\right|)^{-1} \underbrace{\sum_{\underline{\mu}_A \in H_{B,\epsilon}(Z)}} \left|\underline{\mu}_A\right| (\frac{\left|min(\left|\underline{\mu}_A \cap \underline{\mu}_X\right|, \left|\underline{\mu}_A \cap \underline{\mu}_Y\right|)\right|}{max(\left|\underline{\mu}_A \cap \underline{\mu}_X\right|, \left|\underline{\mu}_A \cap \underline{\mu}_Y\right|)} + \frac{\left|min(\left|\underline{\mu}_A \cap \underline{\mu}_X\right|, \left|\underline{\mu}_A \cap \underline{\mu}_Y\right|)\right|}{2 - max(\left|\underline{\mu}_A \cap \underline{\mu}_X\right|, \left|\underline{\mu}_A \cap \underline{\mu}_Y\right|)}) \tag{10}$$

where $\bar{d}$ and $\underline{d}$ define the upper and lower values respectively.

Interval Type-2 Fuzzy Similarity Measure 3 (IT2FSM3) For two IT-2 FSs X and Y, the following formula is hence obtained:

$$IT2FSM3 = 1 - \frac{(\bar{d} + \underline{d})}{2} \tag{11}$$

where

$$\bar{d}_{\tilde{=}_{B,\epsilon}}(X,Y) = (2 \times \underbrace{\sum_{\bar{\mu}_A \in H_{B,\epsilon}(Z)} |\bar{\mu}_A|}_{\bar{\mu}_A \in H_{B,\epsilon}(Z)})^{-1} \cdot \underbrace{\sum_{\bar{\mu}_A \in H_{B,\epsilon}(Z)} |\bar{\mu}_A|}_{\bar{\mu}_A \in H_{B,\epsilon}(Z)} \left(\frac{\left|min(|\bar{\mu}_A \cap \bar{\mu}_X|, |\bar{\mu}_A \cap \bar{\mu}_Y|)\right|}{max(|\bar{\mu}_A \cap \bar{\mu}_X|, |\bar{\mu}_A \cap \bar{\mu}_Y|)} + \frac{\left|min(|\bar{\mu}_A \cap \bar{\mu}_Y|, |\bar{\mu}_A \cap \bar{\mu}_X|)\right|}{max(|\bar{\mu}_A \cap \bar{\mu}_Y|, |\bar{\mu}_A \cap \bar{\mu}_X|)}\right) \tag{12}$$

and

$$\underline{d}_{\tilde{=}_{B,\epsilon}}(X,Y) = (2 \times \underbrace{\sum_{\underline{\mu}_A \in H_{B,\epsilon}(Z)} |\underline{\mu}_A|}_{\underline{\mu}_A \in H_{B,\epsilon}(Z)})^{-1} \cdot \underbrace{\sum_{\underline{\mu}_A \in H_{B,\epsilon}(Z)} |\underline{\mu}_A|}_{\underline{\mu}_A \in H_{B,\epsilon}(Z)} \left(\frac{\left|min(|\underline{\mu}_A \cap \underline{\mu}_X|, |\underline{\mu}_A \cap \underline{\mu}_Y|)\right|}{max(|\underline{\mu}_A \cap \underline{\mu}_X|, |\underline{\mu}_A \cap \underline{\mu}_Y|)} + \frac{\left|min(|\underline{\mu}_A \cap \underline{\mu}_Y|, |\underline{\mu}_A \cap \underline{\mu}_X|)\right|}{max(|\underline{\mu}_A \cap \underline{\mu}_Y|, |\underline{\mu}_A \cap \underline{\mu}_X|)}\right) \tag{13}$$

where $\bar{d}$ and $\underline{d}$ defined the upper and lower values respectively.

3. THE PROPOSED DEEP INTERVAL TYPE-2 FUZZY APPROACH FOR CBIR SYSTEM (DIT2F-CBIR)

Content-based image retrieval system searches the similarity of feature vectors that describe the image. In effect, a new fuzzy module is proposed to combine deep fuzzy features extraction which includes a dimensionality reduction and fuzzy similarity measure. The retrieval process is shown in Figure 2: Start with inputting the images to be retrieved.

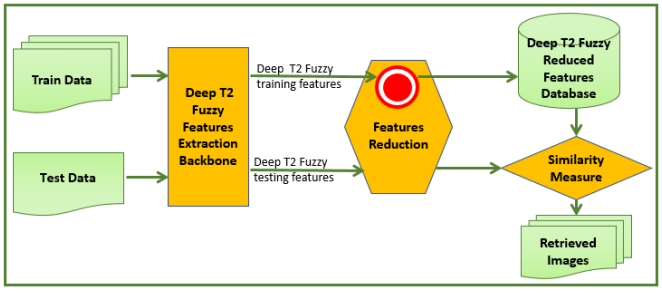

Figure 2: Architecture of DT2F for CBIR system

The steps of our research are as follows:

1. Extract the deep features using the transfer learning of the image and use these features to perform content-based image retrieval.
2. Normalized the deep feature vector.
3. Reduce the dimensional of the features vector, with the PCA or the Auto-Encoder method.
4. Retrieve similar images by calculating the distance between the feature vectors and using the fuzzy similarities measures proposed previously.

3.1. Deep Feature Extraction Backbone and Fuzzification

3.1.1. Deep Feature Extraction Backbone

Deep Learning fulfills âĂIJend-to-endâĂİ learning: from input data, a network is assigned tasks to accomplish (a classification, for example) and learns how to automate them. Thus, it begins with low-level features learning to get the high-level ones. Feature extraction, in the case of transfer learning, consists of taking the convolutional basis of the previously trained network, passing new data through it, and finally removing the new classifier's last layer and assuming this network output as a feature vector. Figure 3 shows a flowchart of this process with more details for understanding the goal. As in most cases, we removed the layer of classification which is the last softmax activation layer. Thus we obtained the feature vector for CBIR system. This vector represents the most learned high-level features as it is the deepest layer of the model. We encoded the images

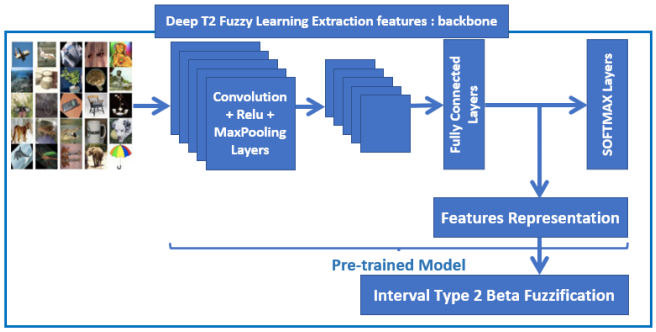

Figure 3: The flowchart of the proposed feature extraction backbone method

of the database by a pre-trained model and obtained for each image an n-dimensional feature vector. Thus, various pre-trained models are employed to extract deep features for comparisons.

Comparison between Deep ImageNet pre-trained Models

The implementation of the specified architectures within Keras provided the use of pre-trained network as a feature extractor of the last fully connected layer of the network [13]. As the images in our problem are comparable to those of ImageNet, we suppose that features extracted by pre-trained models would particularly be useful in similar image research.

There are various pre-trained deep CNNs. However, the CNN architecture is one of the most important changes as it helps improve efficiency. This typically involves using

Table 2
Different model network architectures

Model	Number of params	Number of features
VGG19	143.667.240	512
VGG16	138,357,544	512
NasNetLarge	84,916,818	4032
InceptionResNetV2	55.873.736	1536
InceptionV3	23,851,784	2048
Xception	22,910,480	2048
DenseNet	20,242,984	2048
MobileNetV2	3,538,984	1280

depth (adding new layers) and space. We have evaluated the proposed method on 8 pre-trained deep CNN architectures in table 2 (VGG16, VGG19 [42], NasNetLarge [58], InceptionV3 [45], Inception-Resnet-V2 [44], Xception [8], DenseNet [16], MobileNetV2 [15]).

We conducted a comparative analysis to pick the best deep learning model to check the influence of pre-trained models, as indicated above. We provide a comparative evaluation of the average precision for a range value of **20** on the Corel database for six selected ImageNet pre-trained models. The Euclidean Distance is the dissimilarity metric utilized (L2 norm). We test different pre-trained models (with high/medium/low number of parameters) using Keras libraries as itâĂŹs described in table 2.

Figure 4 below shows the comparison curve of the different models we tested as follows: Figure 4 shows that

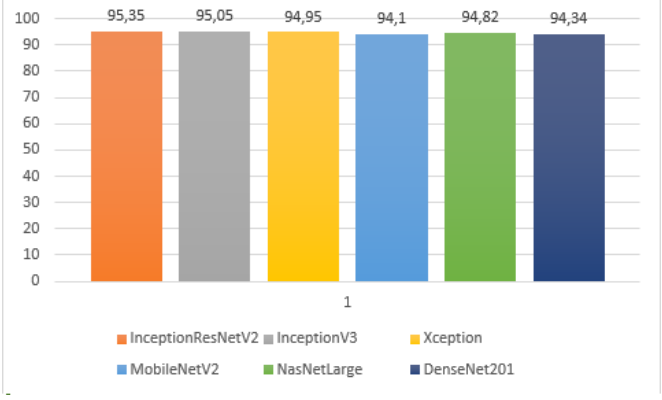

Figure 4: Performance comparison of pre-trained models on Corel Dataset

all the feature extraction backbones used to have similar performance for CBIR system. We will next use Inception-ResnetV2 backbone to carry out experiments taking into account the simple architecture described in section III.

3.1.2. Fuzzification of the Data

Fuzzification interval type-2 beta is introduced to normalize features. We denote the absolute comparison result as y, from a set Y of all the features for the corresponding search, and the corresponding normalized feature as $\mu_{\tilde{\beta}}(y)$. The Interval type-2 Beta upper and lower MF (LMF, UMF) is characterized by four variables, the uncertain center $r \in [r_1, r_2]$, a fixed width σ, m and n, and as is expressed in [52]:

$$\bar{\mu}_{\tilde{\beta}}(y) = \begin{cases} \beta(y; r_1, \sigma, m, n) & y < r_1 \\ 1 & r_1 < y < r_2 \\ \beta(y; r_2, \sigma, m, n) & y > r_2 \end{cases} \quad (14)$$

$$\underline{\mu}_{\tilde{\beta}}(y) = \begin{cases} \beta(y; r_1, \sigma, m, n) & y \leq (r_1 + r_2)/2 \\ \beta(y; r_2, \sigma, m, n) & y > (r_1 + r_2)/2 \end{cases} \quad (15)$$

Where the beta MF can be expressed by:

$$\beta(y) = (1 + \frac{(m+n)(y-r)}{\sigma m})^m (1 - \frac{(m+n)(r-y)}{\sigma m})^m \quad (16)$$

Since, the Footprint of Uncertainty $FOU(\tilde{\beta})$ is the set of all T1FS embedded within IT2FS, the outermost embedded T1FS is the UMF($\tilde{\beta}$), and the innermost embedded T1FS is the LMF($\tilde{\beta}$). According to Mendel et al., 2006, the uncertainty of the footprint of FOU is expressed as the uncertainty range in (i.e. [UMF($\tilde{\beta}$), LMF($\tilde{\beta}$)]) at y shown below $[\underline{\mu}_{\tilde{\beta}}(y), \bar{\mu}_{\tilde{\beta}}(y)]$. And the FOU($\tilde{\beta}$) is the interval linear function described by the lower and upper degree UMF($\tilde{\beta}$) and the LMF($\tilde{\beta}$). Hence, FOU($\tilde{\beta}$) = $\bigcup_{y \in Y} [\underline{\mu}_{\tilde{\beta}}(y), \bar{\mu}_{\tilde{\beta}}(y)]$.

3.2. Dimensionality Reduction of Derived Interval Type-2 Fuzzy Deep Features

To decrease the dimension of a derived interval type-2 fuzzy deep features, we can distinguish two approaches namely PCA, the linear dimension reduction method, and the Auto-Encoder that allows a non-linear dimension reduction [35].

Our proposed algorithm computes the aggregation embedding the upper and lower bounds that is the range for imprecision *FOU()*. It has the property that neighboring points stay close in high-dimensional space and are similarly located to each other in low-dimensional space. Expressly, the embedding of the reduction dimensionality step preserves the *FOU()* presentation of the feature vector while reducing its dimensionality.

3.2.1. PCA Dimensionality Reduction

Principal component analysis (PCA), presents a method of analyzing and visualizing multivariate data with several quantitative variables. It is a question of summarizing the information included in a large database in a certain number of synthetic variables called: Principal components. The goal of PCA is to determine the directions (projection) along which the data variation is maximum. It is a form of data compression that preserves as much information as possible.

3.2.2. Dimensionality Reduction by Auto-Encoder

The general idea of "Auto-Encoders" is very simple, and involves which allows to build an output representation very close to the input one via a new short intermediate representation.

Simple Auto-Encoder

A different approach to nonlinear dimensionality reduction

using autoencoders, as a generalization nonlinear of PCA. An autoencoder has two main functions: the first is to encode the input data to generate an output that has fewer descriptors, and the decoder function recreates the input data from the last coded representation.

Deep Auto-Encoder (DAE)

A DAE is a model of a neural network designed to retain a compressed representation of input data. It consists of multiple layers of encoding and decoding, where each layer reduces the input data dimension and learns a compressed representation. The final layer of the encoder produces the lowest dimensional representation, which is then fed into the decoder layers to reconstruct the original input. The dimensionality reduction process involves training the deep autoencoder on a dataset while minimizing the reconstruction error between the input dataset and its compressed representation. The model learns to encode the most important features input into a lower-dimensional space while discarding irrelevant information, as shown in figure 5 below.

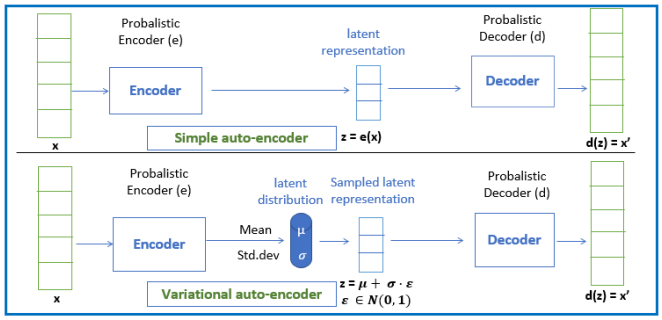

Figure 5: Difference between Simple Auto-Encoder AE and Variational Auto-Encoder VAE

Variational Auto-Encoder (VAE)

A Variational Auto-Encoder can be defined as an Auto-Encoder whose formation is regularized to avoid over-fitting and to ensure good properties for the latent space allowing a generative process. The VAE differs from the simple auto-encoder in that the objective is to estimate the latent space with a Gaussian distribution (mean and variance). To generate X', it is, therefore, necessary to draw a Z code from the distribution and pass it to the decoder. The VAE are powerful tools for reducing dimensionality in large datasets and extracting meaningful features from them.

Multi-Variational Auto-Encoder (MVAE)

A multi-variational autoencoder (MVAE) is an extension of the traditional variational autoencoder, which includes a latent space that captures the underlying structure of the input data. This latent space is learned through variational inference, which allows the MVAE to generate new samples from the learned distribution. Overall, the MVAE architecture is an effective way to reduce dimensionality in high-dimensional datasets while preserving essential features and relationships between different aspects of the data [28].

3.3. Image Retrieval process

3.3.1. Fast Library of Approximation Nearest Neighbors (FLANN)

FLANN[3] is a library for performing fast approximate closest-neighbor searches on huge datasets. The complexity of algorithm is $O(n)^2$ in comparison to $O(2)^n$ for division approach. FLANN provides a collection of algorithms for nearest neighbor search, including hierarchical k-means tree, randomized kd-tree, and locality-sensitive hashing. These algorithms are designed to be efficient in both memory usage and query time, making them suitable for large-scale applications.

FLANN is written in C++ and provides interfaces for several programming languages, including Python, MATLAB, and Java. It also includes support for parallel processing using OpenMP. FLANN has been used in a variety of applications, including computer vision, robotics, bioinformatics, and information retrieval. It is widely regarded as one of the fastest and most accurate libraries for approximate nearest-neighbor searches.

3.3.2. Distance Metric Learning

To find images in a database, search systems perform comparisons between a query and the descriptors extracted from the images. Hence, the user is offered a list of images sorted by the degree of their resemblance to the query. For this task, there is a large variety of similarity measures to choose from. However, we can play with FLANN parameters. Various similarity measures, such as Euclidean, Manhattan, and Canberra distance, are employed in metric learning for image matching. It is trustworthy to note that the fuzzy similarity measures perform quite well.

3.3.3. Retrieving Images

The best similarity metric is used to determine how similar the feature vectors are. Our system returns the set of most similar query images sorted by similarity to any image in the source group. In addition to iterating over each image in the source set, the program computes the degree of similarity to each image in the query set and ranks her with the shortest distance to the source images higher.

4. Experiments and Results

4.1. Results Setting

This section presents a discussion and a comparison of the experiment results of SD-CBIR system. We used Corel-1k (Corel database) [48] which is a very popular database of **1000** images captured from outdoor scenes. It is divided into **10** categories, such as Africans, Foods, Mountains, dinosaurs, Horses, Elephants, Flowers, Buildings, Beaches, and Buses. This database is divided into **90%** for database train and **10%** database test. Each image of the database test can be selected as a query image. Indeed, we first extract its feature vector with InceptionResNetV2 model pre-trained with ImageNet. Then, we fuzzify this vector using IT2B

[3] https://github.com/flann-lib/flann

fuzzy function. After that, we reduce its dimensionality with MVAE. Finally, we compare the query image with all images in the database using a fuzzy similarity measure. Thereafter, the images will be sorted into the query image based on their similarity. In this research, Average Precision (AP) was used as the evaluation criterion, which is shown in Equation 17:

$$AP = average(\frac{Number Relevant Images Retrieved}{Retrieved Images}) \quad (17)$$

The development of our algorithms, we used Python in Colaboratory, or "Colab" for short, which is a product from Google Research. For our experiments, we developed our system with the Python programming language. All networks pre-trained on ImageNet are provided by the Keras[4] library. The mapping is performed by the approximate nearest neighbor algorithm KNN and "FLANN" (Fast Library Approximations Nearest Neighbor) [5].

4.2. Results and Analysis

Image Results of Sample Query Image for SD-CBIR

In this experience, we use InceptionResNetV2 model on the Corel database for the feature extraction. Then, we use the KNN with Euclidean distance to calculate the similarity between images.

In Figure 6, the query image is in the first row and the remaining images show **20** images retrieved for a sample query image belonging to "beach" category on Corel dataset. We find that all **20** retrieved images are relevant (belong to the same class of the query image), hence the precision rate is **1**.

When considering another test image belonging to "food" category in Figure 7 from Corel image dataset, it can be observed that out of the first **20** images retrieved, **15** images are from the query image class and the other ones are not.

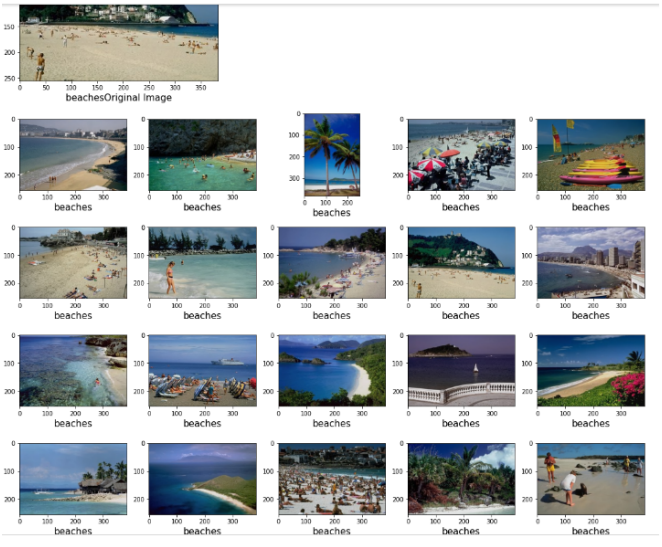

Figure 6: SD-CBIR Top 20 Images Retrieval of Simple Query Image (beach)

[4]https ://keras.io/api/applications/
[5]https://github.com/flann-lib/flann/tree/master/src/python

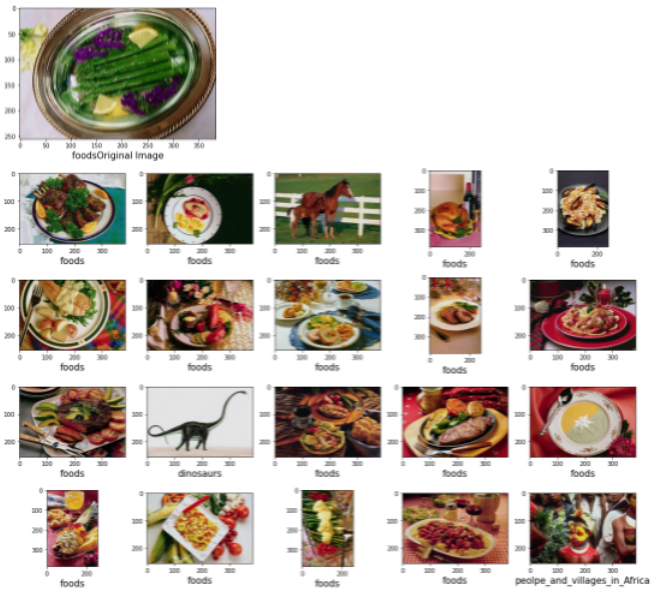

Figure 7: SD-CBIR Top 20 Images Retrieval of Simple Query Image (food)

AP wise category performance of SD-CBIR:

This part reports the category-wise performance on Corel database for a range of the first **20** image retrieved while using Euclidean Distance as a similarity metric.

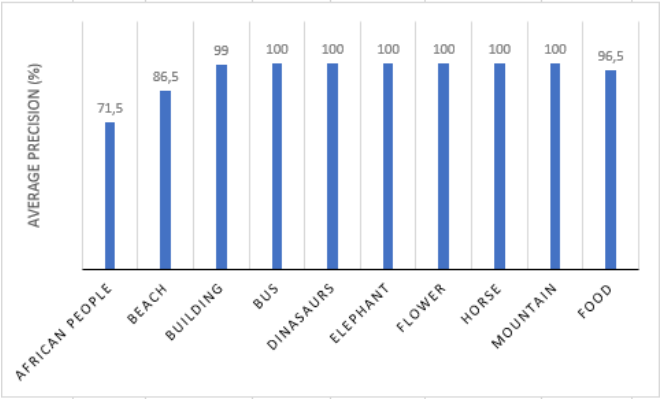

Figure 8: AP wise category performance assessment of SD-CBIR system of Corel database

Figure 8 illustrates that Bus, Dinosaurs, Elephant, Flower, Horse, and Mountain categories have **100%** precision for Corel database. With the pre-trained chosen model InceptionResNetV2, our system retrieves all the images, but we obtained **96.5%**, **86.5%** and **71.5%** precision rate of the food, Beach and African People categories, respectively. The overall average precision for Corel database is **95.35%**. As can be seen in Table 3, it appears that categories with similar features such as "African," "Beach," "Building," and "Foods" may be more susceptible to confusion. The reason for this confusion likely stems from the presence of overlapping visual elements, characteristics, or patterns within these categories. Images within these groups may share certain

Table 3
Comparison of our proposed method with recent papers methods precision/category on Corel database

Categories	SD-CBIR	[41]	[2]	[19]	[52]	[46]	[1]
African	71.50	74.19	81.41	90.00	89.02	84.00	90.00
Beach	86.50	75.38	79.21	75.00	70.50	75.00	85.00
Building	99.49	75.82	91.65	95.00	78.14	87.00	100
Bus	100	81.59	76.54	100	92.30	99.00	99.00
Dinosaurs	100	100	99.76	100	98.00	99.00	95.00
Elephant	100	96.70	96.78	95.00	100	98.00	100
Flower	100	93.21	97.49	95.00	100	99.00	98.00
Horse	100	85.25	84.13	95.00	100	97.00	90.00
Mountain	100	80.47	97.42	70.00	82.00	88.00	100
Food	96.50	81.32	91.46	60.00	80.00	96.00	80.00
Average	**95.35**	**84.39**	**89.58**	**87.50**	**88.65**	**92.20**	**93.70**

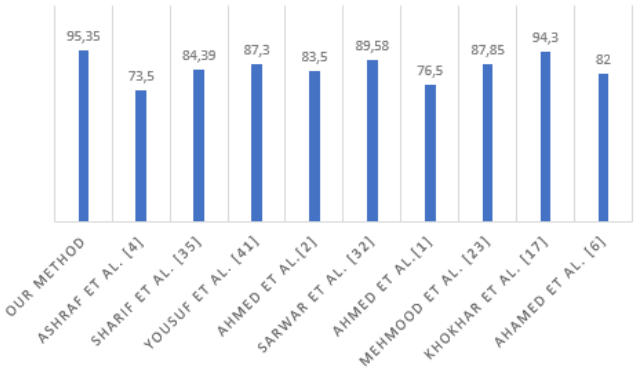

Figure 9: Comparison of SD-CBIR with recent methods works category/precision for the scope of 20 on Corel database

visual attributes that make it challenging for the model or observer to accurately distinguish between them. The African people category has a poor precision value, which can be justified by the nature of the content of the images in this category which are quite complicated. Indeed, these images pose a challenge because they contain objects, with color patterns, which makes it hard to accurately distinguish and separate them. Consequently, it is important to note that our algorithm is capable of accurately recognizing visual features of a wide range of objects, including natural scenes, artificial structures, animals, and food products.

Result achieved from the Proposed Systems:

In the literature, several research works on Corel Database have demonstrated that researchers extracted different types of features and calculated similarity distances using different methods. Two types of methods were performed to compare the obtained results: Category/precision comparison Figure 9 and comparison of average precision with other works in table 3. According to table 3, we observe that our proposed system achieved the best performance with **100%** AP in many categories in comparison with other literature methods on the Corel dataset. The lowest accuracy was recorded for the African (71.5%) and Beach (86.5%) categories. The work of Ahmed et al in [1] achieved a rate of 93.7%, with precision values that vary between 80% and 100% which used deep features. Also in [46], the authors used deep features, the highest accuracy was recorded for the Buses, Dinosaurs, and Horses categories, all achieving 99% accuracy. The lowest accuracy was recorded for the beach category (75%). Overall, the average accuracy across all categories was found to be highest in the SD-CBIR method (95.35%), followed by method [1] (93.7%), method [46] (92.2%), and other works have accuracy rates of less than (90%). These results indicate that our method achieved better results in terms of overall accuracy compared to other methods. These findings support our primary hypothesis, as our SD-CBIR system outperforms that of [52] with 88.65% which its CBIR system uses features extracted in hand-craft.

These findings support our primary hypothesis, as our SD-CBIR system outperforms that of [52] by **95.35%** with **88.65%** which its CBIR system uses features extracted in hand-craft.

4.2.1. SD-CBIR with Dimensionality Reduction

SD-PCA-CBIR : Dimensionality Reduction with PCA
We have already seen that introducing a very deep neural network model (InceptionResNetV2) significantly improves the results, but the image acquisition time is questionable. Our ultimate goal, regardless of the model we use, is to restore the image in a reasonable amount of time. Indeed, we explore the time complexity and demonstrate that our system with deep models, can retrieve images from the Corel dataset in a fair amount of time. It also illustrates how using Principal Component Analysis can speed up image retrieval without sacrificing precision. We employ principal component analysis PCA to evaluate the performance of our SD-CBIR system with dimensionality reduction in the first application. First, to explain the experiment using PCA method, we extract the features of all dataset images through the InceptionResNetV2 model, then reduce the dimension of the features vector to **100** of each image.

When a query image is received, it is now passed via the CBIR model and the PCA. The extracted features from the query image will then be compared to each of the dataset's feature vectors, to retrieve those images whose features are closest to the query image features as measured by the Euclidean Distance metric. The image retrieval time is the time required to search for the most similar images to the query. This time is shown in table 4. We take all of the test query photos and calculate the retrieval time for each one, then provide the average retrieval time. The length of time it takes to get an image is determined by the database's size and dimensionality. Table 4 shows the effect of the reduction with PCA by average precision and response time with the variation of features dimensionality reduction. From Figure 4, we have noticed better precision of **97.27%** and **97.29%** for the dimensions of **100** and **500**, respectively. That is why, we will apply it on the test base. Figures 10 shows the result: For several categories such as building, bus, dinosaurs, elephant, flower, horse, and mountain the average precision value is **100%**. From Table 4, we choose the **dimension = 100** because the average precision is better than for **dimension**

Table 4
PCA Derived Deep Learning Features to CBIR: Setting

Dimensions	Accuracy(%)	Time(s)
1	45.80	0.48
5	87.02	0.51
10	96.90	0.49
50	97.09	0.77
75	97.17	0.82
100	**97.27**	**0.88**
200	97.25	0.95
300	97.25	1.05
400	97.26	1.15
500	**97.29**	**1.26**
600	97.29	1.41
800	97.29	1.60

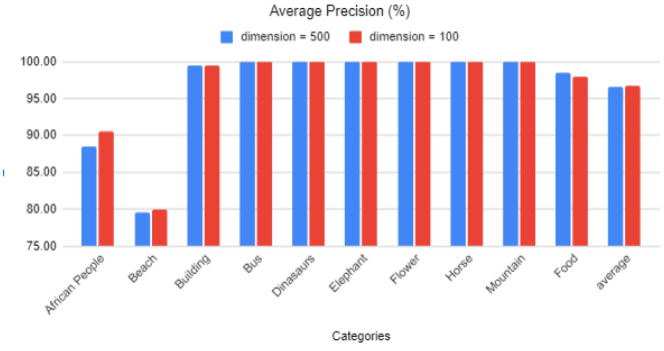

Figure 10: SD-PCA-CBIR: Result on Corel database

Table 5
Different architectures using AE

Method	Architecture	Total param
AE	[(1536, 100, 1536)]	308,836
DAE	[(1536, 864, 256, 100, 864, 256, 1536)]	3,151,652
SVAE	[(1536, 100, 1536)]	158,712
MVAE	[(1536, 864, 256, 100, 256, 864, 1536)]	3,126,968

= 500. Rate of average precision with PCA: **96.8%** and the rate of average precision without PCA: **96.44%**. Briefly, the application of PCA improves precision while reducing computational time because we are dealing with relatively low dimensionality.

SD-AE-CBIR : Dimensionality Reduction with Auto-Encoder

In this experiment, we use the Auto-Encoder to reduce the extracted features from dimension **1536** to **100** features of each image. Then these features are compared with the feature list in the bank of features. To this end, we used 4 types of auto-encoder as shown in Table 5.

Experiment 1: Simple Auto-Encoder

In the first experiment, we will train a neural network with only one hidden layer h to encode input x into output z. Indeed, the hidden layer should contain information relevant to the reconstruction. After pre-training our network (Auto-Encoder), the obtained results are shown in Figure 6, in

Table 6
SD-AE-CBIR : Results

CategoriesAP (%) Method	SAE	DAE	VAE	MVAE
African	89.49	86.00	85.00	91.50
Beach	80.99	76.50	82.00	79.50
Building	100	98.00	100	99.49
Bus	100	100	100	100
Dinosaur	100	99.49	100	100
Elephant	100	100	100	100
Flower	100	100	100	100
Horse	100	100	100	100
Mountain	100	100	98.50	100
Food	94.50	98.00	96.00	99.00
Average	**96.50**	**95.80**	**96.14**	**96.94**
Epoch	1000	50	500	500
Time of training	2min	9s	42s	1min

which we notice that although the performance of AE is not satisfactory compared to the **PCA 96.8%** they encourage us to modify the network configuration. We can see that the precision has slightly increased with an epoch of **1000**, it reaches **96.50%** compared to our system **without reduction** with **95.35%**. Additionally, We can see that several categories reached a precision rate of **100%**. This stimulates us, together with the parameters adjustment, to proceed to the investigation of another kind of auto-encode to achieve better performance. Therefore, we will try to use deep Auto-Encoders.

Experiment 2: Deep Auto-Encoder:

It is possible to perform the reconstruction by passing the information through several hidden layers $h_1, ..., h_n$, and thus we obtain a deep Auto-Encoder. As this type of Auto-Encoders contains multiple layers, the number of parameters increases significantly, as shown in Figure 5. According to the experimental results in Table 6, while employing deep Auto-Encoder to retrieve images, there is a minor improvement of **95.80%** precision for **50** iterations compared to our method**without reduction** of **95.35%**. Although deep Auto-Encoder does not achieve the best result in terms of precision compared to simple Auto-Encoder, the complexity of the hidden layers and the parameterization in the used Deep AE can affect the performance of the obtained results. To have better performance, we decide to use Variational Auto-Encoders.

Experiment 3: Simple Variational Auto-Encoder:

In its simplest form, a Variational Auto-Encoder (VAE) is a three-layer network, which is equivalent to one-hidden-layer neural network. The input and output have the same dimension and VAE learns the reconstruction of the input. Figure 5 shows that the number of VAE parameters is close to that of a simple Auto-Encoder since they have the same number of layers. According to the experimental results in Table 6, there is a slight improvement achieved of **96.14%** precision after **500** epochs, but it doesn't outperform **PCA reduction with 96.8%**. Since a hidden layer is not enough to achieve satisfactory results, we extend the Variational Auto-Encoder with several hidden layers (Multilayer Variational

Table 7
Comparison of the average precision values on the Caltech-101 database

Works	SD-MVAE-CBIR	DFPCA [30]	[2]	[11]
AP (%)	**83.64**	82.54	62.00	83.00

Auto-Encoder (MAE)) to achieve better performance as discussed in the next subsection.

Experiment 4: Multi-layer Variational Auto-Encoder (MVAE):

MVAE is a VAE with several hidden layers. An influence of the hidden layers was noticed on the number of the used parameters in Figure 5. Table 6 reveals that applying MVAE improves image retrieval performance significantly. Indeed, the acquired precision index of **96.94%** for **500** epochs is the best of all experiences, even when compared to PCA reduction results. It can also be noticed that the precision rate for certain categories, such as bus, dinosaur, elephant, flower, horse, and mountain, is **100%** regardless of the number of iterations. Besides, the performance for the categories building and food is close to **99%**. As a result, it can be observed that the MVAE reduces the dimension of the features vector and improves the outcomes in one minute, which is enough time. The importance of our strategy is demonstrated by experimental results, which show that the MVAE reduction method used does not lead to significant information loss. We thus obtain the best result of **96.94%** for **epochs = 1000** with MVAE. Since this result outperforms all the previous ones, we will take it for all our subsequent experiments.

Experiment 5: Results on Caltech-101

To demonstrate the efficiency of adding a dimensionality reduction module, We apply our method to Caltech-101 database and record image retrieval time since Corel-1k (corel database) is a small image database. In this experiment, the dimension of the InceptionResNetV2-model encoded feature vector is **1536**, which looks large. Therefore, we performed MVAE reduction on the feature vector to reduce its dimensionality to **100**. Table 7 shows the average precision values on the Caltech-101 database. With the MVAE reduction dimensionality, the performance is **83.64%**. This value takes 52 minutes of execution time which is a reasonable time according to the number of database images. Also, our method outperforms [30], [2] and [11] respectively by **1.1%**, **21.64%** and **0.64%** on Caltech-101 dataset. Our algorithm can be said to perform well by comparing this average precision value to other previous research in the literature on the Caltech-101 database.

4.2.2. Results of Similarity Measures

In this experiment, we process all the images of Corel database through the SD-MVAE-CBIR model with InceptionResNetV2 and Multi-Variational Auto-Encoder models, then save the **100** pertinent features extracted from each image in a database. Now when a test image is received, feature extraction is applied through deep fuzzy method and Auto-Encoder. After that, using the K-Nearest Neighbor method, we compare the features extracted from the query image with each feature vector in the feature vector database to find images with features close to those of the query image measured by various methods. In the second phase, we will introduce the fuzzy logic in the vector features and use the proposed fuzzy similarity measures defined in section 2.5. In the next phase, we will employ these proposed fuzzy similarity measures to integrate fuzzy logic into the vector features.

Experiment 1: Effects of the distance measures

This section presents the results of SD-CBIR system with some distance measures that allow us to find the closest images to a given query image. As shown in Figure 11 we compare some metrics based on different criteria[6]. We need a mechanism to measure the outcomes' performance by providing an average precision score to evaluate them. The first experiment will be conducted using Metric distances. Except for the beach category, these results obtained a rate of **100%** for most of the categories, and for the various distance measurements, we notice a decrease of **20%** for all distance measures compared to other categories. This is proven by a near precision for all measures and better average precision for Euclidean distance.

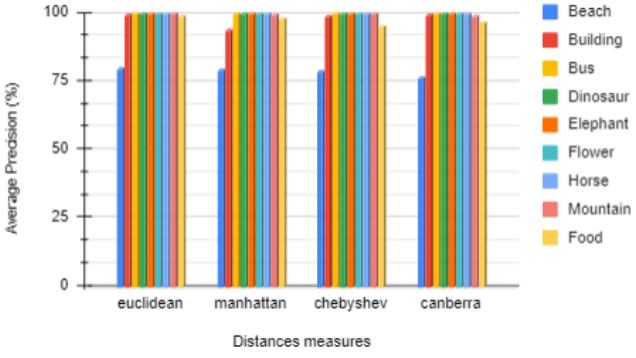

Figure 11: Effects of the distance measures to SD-CBIR: Results

Experiment 2: Effects of the fuzzy similarity measures

In the previous tests, we studied the obtained performance using some distance metrics with the kNN nearest neighbor search algorithm. Further, we note that similarity metrics fail with the k-nearest neighbor search algorithm. For this, we use the nearest neighbor approximation FLANN instead of KNN. Indeed, FLANN is a fast library allowing the use of some similarity measures as well as distance measures. In this part, we will study the results obtained with the fuzzy similarity measures described in section II in the context of similar image searches using SD-MVAE-CBIR. With a deep learning feature extraction backbone, all of the features of the request picture and the database images are

[6]https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

Table 8
Time and Precision of different clustering criteria in FLANN algorithm

Method	KD-trees	K-means	Composite	Hierarchical clustering
AP (%)	95.20	**96.40**	96.30	93.80
Time (s)	1.2	1.2	1.1	1

extracted. After that, from these vectors, an affine interval type-2 beta fuzzification step is performed. If our data is on various scales, fuzzy similarity metrics will fail. The use of feature normalization can help to ensure that results are not skewed. The FLANN method is then applied with Euclidean distance to compare its performance to that of the KNN. The FLANN algorithm has numerous initialization methods that can be used as parameters. To adjust these criteria, we test the four FLANN implementations (KD-trees, Kmeans, composite, hierarchical clustering). Table 8 shows the results of this test. All four FLANN implementations take about the same amount of time to compute. The results of all four techniques are delivered in approximately a second, showing that the implementations are extremely comparable, except for the clustering criterion. It is also noted that the k-means clustering criteria have the best average precision. As a result, we used K-means to implement neighbor search.

Effects of the normalization step on reduction

The questions we have considered are: how does the fuzzification method influence the ability of the trained model SD-CBIR: What is the power of the dimension reduction modules on SD-MVAE-CBIR system?

After the deep extraction features backbone, we need to normalize our data set. Several distance methods will not work if we use structurally different scales of data. In this part, we check the precision of retrieval images for the Corel database after using the proposed fuzzy similarity measures with and without a dimensionality reduction step. Besides, we compare the performance of the proposed fuzzy similarity measures with and without dimensionality reduction. According to the experimental details presented in Table 9, the best performance in terms of average precision is **97.15%**, which is obtained with IT2FSM2 with a feature reduction step to **100** features per image. While without reduction, the achieved average precision of this similarity measure is **97.14%** (the same rate). For IT2FSM1, the average precision achieves **96.25%** with reduction and **94.4%** without reduction. Furthermore, IT2FSM3 achieves an average precision of **94.95%** with reduction and **95.7%** without reduction. Thus, we can conclude that with dimensionality reduction, our system gives better results. However, on the Corel dataset, the results show the best real-time performance across all categories. This is an unexpected side benefit, as our method does not accurately optimize mAP scores that rely on fuzzy models trained on fuzzy similarity measures.

The second experiment was performed on Caltech-101 data collection. We selected randomly fifteen categories; the contents include airplanes, ferry, camera, brain, minaret, motorbrikes, etc. To use the Caltech database. The average precision rates for 15 categories of the Caltech-101 database are shown in 10. The best performance in terms of average precision is **86.70%** with reduction and **78.44%** without reduction, both with IT2BFNS2. The reason for these performance indicators is that many classes have images that are less than one hundred, and some even have less than fifty. Nevertheless, the proposed system's effectiveness remains intact and it proves to be a practical and feasible solution to address the image retrieval problem.

Now we consider the experiment results on CIFAR-10 dataset. Our results in 11 do suggest that the deep fuzzy system is more performant in accuracy terms (achieving the highest accuracy of **93.43%**) than other deep fuzzy systems in the literature. Furthermore, our method improves the state-of-the-art [27], [23] and [51] respectively by **0.51%**, **0.75%** and **1.55%** on Cifar-10 dataset. From these experiments, we can conclude that we have explored the most used datasets in the literature. In all these experiments, we give performances acceptable and superior to other works of the state of the art.

We reinforced the presented results, by comparing our fuzzification method with other normalization methods. The application of fuzzy theory in the context of deep learning is constrained and the broader context of Content-Based Image Retrieval (CBIR). The assessment of various normalization techniques enables an evaluation of their impact on the feature vector extracted and their effectiveness in enhancing the performance of CBIR. We present the results obtained from experiments conducted to investigate data fuzzification (normalization). To evaluate the performance of different normalization methods, we compared four techniques: the fuzzy mathematical algorithm applied to color and texture feature extraction, as presented in [57], the l2-norm normalization from [38], the transitional normalization method introduced in [47], the linear normalization technique discussed in [22] and the fuzzy deep function with a lower membership degree used in this work. 12 displays the results of comparing these normalization methods based on the precision metric in the CBIR context. Our work with IT2FSM2 outperformed other methods with a precision of 97.14%. The linear normalization technique achieved a close precision of 96.67% in [38]. The linear normalization technique was given a good precision rate of 93% in [22] The remaining normalization methods yielded lower precisions of 86% and 78.31%, respectively. These contributions were introduced in [47] and [57]. Lower and upper fuzzy deep features are typically computed to investigate how different levels of uncertainty affect CBIR performance. By experimenting with using either the lower membership degree with the fuzzy similarity measure 7, we have obtained a precision rate of 87%, which does not lead to better retrieval results but is slightly higher than the degree of the upper membership degree with the fuzzy similarity measure which is 86.4%. This result justified the precision with a type 2 fuzzy function is more efficient than with one of its members (upper or

Table 9
Effects of the proposed fuzzy similarity measures to CBIR: Results on Corel dataset

	without reduction			with reduction		
AP(%) / Cat	IT2FSM1	IT2FSM2	IT2FSM3	IT2FSM1	IT2FSM2	IT2FSM3
African	84.00	86.99	82.00	86.00	87.00	78.50
Beach	75.00	88.00	83.00	89.00	89.00	87.50
Building	98.50	100	99.00	100	100	97.50
Bus	95.50	100	100	93.00	99.00	98.00
Dinosaurs	99.00	100	100	100	100	100
Elephant	100	100	100	100	100	100
Flowers	100	100	100	100	100	100
Horses	97.50	100	100	100	100	100
Mountains	100	100	93.49	100	100	100
Food	94.50	96.49	99.49	96.50	94.50	98.00
Average	94.40	**97.14**	95.70	**96.25**	**97.15**	**95.95**

Table 10
Effects of the proposed fuzzy similarity measures to CBIR: Results on Caltech-101 database

	without reduction			with reduction		
AP(%) / Cat	IT2FSM1	IT2FSM2	IT2FSM3	IT2FSM1	IT2FSM2	IT2FSM3
Airplanes	95.00	91.10	87.00	94.00	94.00	92.18
Ferry	66.00	68.50	66.00	71.51	78.00	71.50
Camera	85.00	71.10	72.00	84.02	84.00	87.11
Brain	76.00	71.20	70.00	74.50	81.00	76.22
Cougarface	83.00	78.90	71.00	91.05	93.00	85.00
Grad piano	90.00	84.05	73.00	93.00	92.00	95.00
Dalmation	73.00	66.00	68.00	76.00	90.00	79.00
Dollar bill	35.00	41.78	40.00	44.21	56.00	47.11
Starfish	49.00	37.00	44.00	42.20	58.00	47.68
Soccer ball	80.00	71.22	71.00	83.05	87.00	81.55
Minaret	68.00	63.88	79.00	73.50	88.00	75.05
Motorbikes	78.00	81.00	72.00	81.78	90.00	83.00
Revolver	63.00	59.00	71.00	72.00	75.00	76.66
Sunflower	76.00	78.44	68.00	83.00	92.00	80.05
Windsorchair	73.00	78.44	73.00	73.00	84.00	80.05
Average	69.50	78.44	72.80	83.00	**86.70**	79.05

Table 11
Comparison of the average precision values with recent works on the CIFAR-10 dataset

Works	DIT2F-MVAE-CBIR	PSCH [27]	[23]	Deep GK [51]
Accuracy (%)	**93.42**	92.91	92.67	91.87

Table 12
Comparison of normalization methods for precision of COREL dataset

AP(%) / Cat	[38]	[47]	[56]	[22]	IT2FSM2	Lower-Fuzzy-Deep	Upper-Fuzzy-Deep
African	96.67	98.00	77.9	89.00	87.00	83.50	88.20
Beach	93.33	66.00	60.10	84.00	89.00	87.00	79.33
Building	100	90.00	69.10	82.00	100	76.20	100
Bus	100	100	87.60	100	99.00	99.00	82.50
Dinosaurs	100	100	99.40	100	100	97.50	100
Elephant	100	76.00	59.25	97.00	100	74.88	68.00
Flowers	100	100	95.80	100	100	94.00	100
Horses	100	100	91.85	100	100	93.00	91.33
Mountains	90.00	40.00	64.00	91.00	100	88.75	90.50
Food	90.00	90.00	78.10	88.00	94.5	78.90	64.33
Average	96.67	86.00	78.31	93.00	**97.15**	87.27	86.41

lower) and confirmed our choice of the Interval type 2 Beta fuzzy function for the fuzzification function.

5. Conclusion

In comparison to the features produced by traditional approaches in prior works, this research shows that employing deep fuzzy learning features with decreased dimensionality delivers improved precision outcomes. Hence, using pre-trained deep learning features was confirmed to improve the result in our CBIR system. Because we are dealing with high dimensions, the dimensionality reduction improves precision and saves computational time. We performed extensive experiments on two widely used image datasets to validate the performance of what was proposed MVAE to achieve competitive performance. In addition, the feature representation is based on InceptionResNet-V2 model and Interval type-2 beta fuzzy function, with the use of interval type-2 fuzzy similarity measures. The proposed method achieved **97.15%** performance that represents **8.5%**, **1.03%**, **3.45%** and **0.55%** increase compared to the results of respectively [52], [30], [1] and [11] on the Corel data set. Furthermore, our method improves the state-of-the-art [27], [23] and [51] respectively by **0.51%**, **0.75%** and **1.55%** on Cifar-10 dataset. Also, our method outperforms [30], [2] and [11] by **1.1%**, **21.64%** and **0.64%** on Caltech-101 dataset. Which can open the door for several hybridization breakthroughs in the area of image retrieval. Thus, an extensive evaluation using several databases shows that deep fuzzy method-based retrieval system framework convincingly outperforms deep or fuzzy methods-based retrieval. The results are motivating and indicate that our proposed approach can be applied for several real live applications such as medical diagnosis.

Acknowledgment

The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under grant agreement number LR11ES48.

References

[1] Ahmed, K.T., Jaffar, S., Hussain, M.G., Fareed, S., Mehmood, A., Choi, G.S., 2021. Maximum response deep learning using markov, retinal & primitive patch binding with googlenet & vgg-19 for large image retrieval. IEEE Access 9, 41934–41957.

[2] Ahmed, K.T., Ummesafi, S., Iqbal, A., 2019. Content based image retrieval using image features information fusion. Information Fusion 51, 76–99.

[3] Alimi, A.M., Hassine, R., Selmi, M., 2003. Beta fuzzy logic systems: Approximation properties in the mimo case. International Journal of Applied Mathematics and Computer Science 13, 225–238.

[4] Batyrshin, I., 2019. Towards a general theory of similarity and association measures: similarity, dissimilarity and correlation functions. Journal of Intelligent & Fuzzy Systems 36, 2977–3004.

[5] Bhoir, S.V., Patil, S., 2021. A review on recent advances in content-based image retrieval used in image search engine. Library Philosophy and Practice , 1–45.

[6] Canziani, A., Paszke, A., Culurciello, E., 2016. An analysis of deep neural network models for practical applications. CoRR abs/1605.07678. URL: http://arxiv.org/abs/1605.07678, arXiv:1605.07678.

[7] Chimatapu, R., Hagras, H., Kern, M., Owusu, G., 2020. Hybrid deep learning type-2 fuzzy logic systems for explainable ai, in: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE. pp. 1–6.

[8] Chollet, F., 2016. Xception: Deep learning with depthwise separable convolutions. CoRR abs/1610.02357. URL: http://arxiv.org/abs/1610.02357, arXiv:1610.02357.

[9] Devulapalli, S., Potti, A., Krishnan, R., Khan, M.S., 2023. Experimental evaluation of unsupervised image retrieval application using hybrid feature extraction by integrating deep learning and handcrafted techniques. Materials Today: Proceedings 81, 983–988.

[10] Dubey, S.R., 2021. A decade survey of content based image retrieval using deep learning. IEEE Transactions on Circuits and Systems for Video Technology .

[11] Giveki, D., Shakarami, A., Tarrah, H., Soltanshahi, M.A., 2021. A new method for image classification and image retrieval using convolutional neural networks. Concurrency and Computation: Practice and Experience , e6533.

[12] Gkelios, S., Sophokleous, A., Plakias, S., Boutalis, Y., Chatzichristofis, S.A., 2021. Deep convolutional features for image retrieval. Expert Systems with Applications 177, 114940.

[13] Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Zhang, L., Han, W., Huang, M., Jin, Q., Lan, Y., Liu, Y., Liu, Z., Lu, Z., Qiu, X., Song, R., Tang, J., Wen, J.R., Yuan, J., Zhao, W.X., Zhu, J., 2021. Pre-trained models: Past, present and future. AI Open URL: https://www.sciencedirect.com/science/article/pii/S2666651021000231, doi:https://doi.org/10.1016/j.aiopen.2021.08.002.

[14] Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural computation 18, 1527–1554.

[15] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR abs/1704.04861. URL: http://arxiv.org/abs/1704.04861, arXiv:1704.04861.

[16] Huang, G., Liu, Z., Weinberger, K.Q., 2016. Densely connected convolutional networks. CoRR abs/1608.06993. URL: http://arxiv.org/abs/1608.06993, arXiv:1608.06993.

[17] Jing, L., Tian, Y., 2020. Self-supervised visual feature learning with deep neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence .

[18] Jolliffe, I., 2005. Principal component analysis. Encyclopedia of statistics in behavioral science .

[19] Kale, M., Mukhopadhyay, S., 2020. Effective image retrieval method of natural images in a large database using fuzzy class membership. Journal of Electronic Imaging 29, 053012.

[20] Kapoor, R., Sharma, D., Gulati, T., 2021. State of the art content based image retrieval techniques using deep learning: a survey. Multimedia Tools and Applications , 1–23.

[21] Karamti, H., Shaiba, H., Mahmoud, A.M., 2020. Hybrid shallow learning and deep learning for feature extraction and image retrieval., in: ICEIS (1), pp. 165–172.

[22] Kenchappa, Y.D., Karibasappa, K., 2021. Dual phase cbir model using hybrid feature extraction and manhattan distance measure. International Journal of Intelligent Engineering and Systems 14, 72–81.

[23] on CIFAR-10 using kerasâĂİ, â.R., . Keras examples. [Online].Available: https://github.com/keras-team/keras/tree/master/keras/datasets. Accessed: Oct. 27, 2019.

[24] Kunapuli, S.S., Bhallamudi, P.C., 2021. Chapter 22 - a review of deep learning models for medical diagnosis, in: Kumar, P., Kumar, Y., Tawhid, M.A. (Eds.), Machine Learning, Big Data, and IoT for Medical Informatics. Academic Press. Intelligent Data-Centric Systems, pp. 389–404. URL: https://www.sciencedirect.com/science/article/pii/B9780128217771000070, doi:https://doi.org/10.1016/B978-0-12-821777-1.00007-0.

[25] Latif, A., Rasheed, A., Sajid, U., Ahmed, J., Ali, N., Ratyal, N.I., Zafar, B., Dar, S.H., Sajid, M., Khalil, T., 2019. Content-based image retrieval and feature extraction: a comprehensive review. Mathematical Problems in Engineering 2019.

[26] Le, Y., Yang, X., 2015. Tiny imagenet visual recognition challenge. CS 231N 7, 3.

[27] Li, H., Li, Y., Xie, X., Gao, S., Mao, D., 2020. Pseudo labels and soft multi-part corresponding similarity for unsupervised deep hashing. IEEE Access 8, 53511–53521.

[28] Liu, G., Xie, L., Chen, C.H., 2020. Unsupervised text feature learning via deep variational auto-encoder. Information Technology and Control 49, 421–437.

[29] Lu, H., Zhang, M., Xu, X., Li, Y., Shen, H.T., 2020. Deep fuzzy hashing network for efficient image retrieval. IEEE transactions on fuzzy systems 29, 166–176.

[30] Maji, S., Bose, S., 2021. Cbir using features derived by deep learning. ACM/IMS Transactions on Data Science (TDS) 2, 1–24.

[31] Mathews, A., Sejal, N., Venugopal, K., 2022. Analysis of content based image retrieval using deep feature extraction and similarity matching. International Journal of Advanced Computer Science and Applications 13.

[32] Nigsch, F., Bender, A., van Buuren, B., Tissen, J., Nigsch, E., Mitchell, J.B., 2006. Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization. Journal of chemical information and modeling 46, 2412–2422.

[33] Pyle, D., 1999. Data preparation for data mining. morgan kaufmann.

[34] Rodrigues, J., Cristo, M., Colonna, J.G., 2020. Deep hashing for multi-label image retrieval: a survey. Artificial Intelligence Review 53, 5261–5307.

[35] Roweis, S.T., Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. science 290, 2323–2326.

[36] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 211–252. doi:10.1007/s11263-015-0816-y.

[37] Saritha, R.R., Paul, V., Kumar, P.G., 2019. Content based image retrieval using deep learning process. Cluster Computing 22, 4187–4200.

[38] Sayed, M.S., Elrab, A., Fathy, K.A., Raslan, K.R., 2023. A deep learning content-based image retrieval approach using cloud computing. International Research Journal of Engineering and Technology (IRJET) 29, 1577–1589.

[39] Shahzad, A., Raza, M., Shah, J.H., Sharif, M., Nayak, R.S., 2021. Categorizing white blood cells by utilizing deep features of proposed 4b-additionnet-based cnn network with ant colony optimization. Complex & Intelligent Systems , 1–17.

[40] Shamsipour, G., Fekri-Ershad, S., Sharifi, M., Alaei, A., 2024. Improve the efficiency of handcrafted features in image retrieval by adding selected feature generating layers of deep convolutional neural networks. Signal, image and video processing , 1–14.

[41] Sharif, U., Mehmood, Z., Mahmood, T., Javid, M.A., Rehman, A., Saba, T., 2019. Scene analysis and search using local features and support vector machine for effective content-based image retrieval. Artificial Intelligence Review 52, 901–925.

[42] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. URL: https://arxiv.org/abs/1409.1556, doi:10.48550/ARXIV.1409.1556.

[43] Sola, J., Sevilla, J., 1997. Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Transactions on nuclear science 44, 1464–1468.

[44] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A., 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. URL: https://arxiv.org/abs/1602.07261, doi:10.48550/ARXIV.1602.07261.

[45] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the inception architecture for computer vision. CoRR abs/1512.00567. URL: http://arxiv.org/abs/1512.00567, arXiv:1512.00567.

[46] Tarawneh, A.S., Celik, C., Hassanat, A.B., Chetverikov, D., 2020. Detailed investigation of deep features with sparse representation and dimensionality reduction in cbir: A comparative study. Intelligent Data Analysis 24, 47–68.

[47] Wang, C., Liu, L., Tan, Y., 2020. An efficient content-based image retrieval system using knn and fuzzy mathematical algorithm. CMES-Computer Modeling in Engineering & Sciences 124.

[48] Wang, J.Z., Li, J., Wiederhold, G., 2001. Simplicity: Semantics-sensitive integrated matching for picture libraries. IEEE Transactions on pattern analysis and machine intelligence 23, 947–963.

[49] Wang, Y., Yao, H., Zhao, S., 2016. Auto-encoder based dimensionality reduction. Neurocomputing 184, 232–242. URL: https://www.sciencedirect.com/science/article/pii/S0925231215017671, doi:https://doi.org/10.1016/j.neucom.2015.08.104. roLoD: Robust Local Descriptors for Computer Vision 2014.

[50] Wu, D., Mendel, J.M., 2009. A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets. Information Sciences 179, 1169–1192.

[51] Yeganejou, M., Dick, S., Miller, J., 2019. Interpretable deep convolutional fuzzy classifier. IEEE Transactions on Fuzzy Systems 28, 1407–1419.

[52] Yosr, G., Baklouti, N., Hagras, H., Alimi, A.M., et al., 2021. Interval type-2 beta fuzzy near sets approach to content-based image retrieval. IEEE Transactions on Fuzzy Systems .

[53] Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., Saeed, J., 2020. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. Journal of Applied Science and Technology Trends 1, 56–70.

[54] Zhang, Y.y., Feng, Y., Liu, D.j., Shang, J.x., Qiang, B.h., 2020. Frwcae: joint faster-rcnn and wasserstein convolutional auto-encoder for instance retrieval. Applied Intelligence 50, 2208–2221.

[55] Zheng, L., Yang, Y., Tian, Q., 2017. Sift meets cnn: A decade survey of instance retrieval. IEEE transactions on pattern analysis and machine intelligence 40, 1224–1244.

[56] Zhou, J.X., Liu, X.d., Xu, T.W., Gan, J.h., Liu, W.q., 2018. A new fusion approach for content based image retrieval with color histogram and local directional pattern. International Journal of Machine Learning and Cybernetics 9, 677–689.

[57] Zhou, W., Li, H., Tian, Q., 2017. Recent advance in content-based image retrieval: A literature survey. arXiv preprint arXiv:1706.06064 .

[58] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V., 2017. Learning transferable architectures for scalable image recognition. CoRR abs/1707.07012. URL: http://arxiv.org/abs/1707.07012, arXiv:1707.07012.