Categorization of Images using Content-based Features: A Data Mining Approach

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

OF

MASTERS OF TECHNOLOGY

IN

COMPUTER SCIENCE

BY

ADITYA N

(MTC0512)

Under the Guidance of

Prof. Aditya Bagchi
(Computer and Statistical Services Centre)
Dr. Pinakpani Pal
(Electronics and Communication Sciences Unit)

INDIAN STATISTICAL INSTITUTE

203, Barrackpore Trunck Road

KOLKATA – 700 035

Acknowledgements

It gives me immense pleasure and satisfaction to express my heart-felt gratitude and respect to my professor and supervisor, Prof. Aditya Bagchi, for his invaluable guidance, supervision and encouragement throughout my project work. I will be grateful to him forever, for his most timely suggestions, and help regarding many aspects during this period. Working under him is a wonderful experience.

I express deep regards and thanks to all my teachers who have given me invaluable knowledge. I would like to express my special thanks to Dr. Pinakpani Pal for his valuable suggestions during all stages of my dissertation and support in Image Processing Lab. I thank all my colleagues and my classmates for their support and maintaining an excellent work environment in the Lab.

Finally I express my heart-felt thanks to my parents who have unconditionally encouraged and supported me from the beginning to end.

Date:

ADITYA N
Indian Statistical Institute,
Kolkata – 700 035.

Contents

Synopsis: Categorization of Images using Content-based Features: A Data Mining Approach

Aditya N (MTC0512)

Under the supervision of
Prof. Aditya Bagchi
Dr. Pinakpani Pal

Objective:

The basic objective of this project is to classify the images based on the extracted visual contents or features from a collection of large set of images present in the image database. It also aims at continuously restructuring the class hierarchy dynamically upon the arrival of new instances into the existing structure. In our present algorithm we used shape, texture and color features for the purpose of classification.

Over View of Work:

In the context of our algorithm shape, texture and color features are combined to form 57-dimensional feature vector. These features have values in the range of (0 to 8) or (0 to16). The absence of a particular feature is marked by 0 and the presence is indicated by a value in the range of (1 to 8) or (1 to 16), signifying the strength of that feature. Each of these feature values is converted into respective binary strings of 8-bit or16-bits where, if the feature value is i, then the i^{th} bit is set to 1 and rest of the bits to 0 in the corresponding 8-bit/16-bit vector. This reduces the given problem into a single level processing. Based on these features a classification tree has been constructed that performs continuous restructuring of a class hierarchy. The class hierarchy represents relationship among different items of the database.

The system starts with an initial class hierarchy consisting of a global root and its children as the representatives of each class of images under consideration. The class representatives are obtained by the initial training. In the training phase, for each class we find out the corresponding identifying features from the training data. This identifying feature becomes the feature vector for the root node corresponding to that class (representative). The system keeps on modifying the class hierarchy as new instances are considered. These new instances are either taken from the underlying database or inserted afresh. When a new instance (transaction) arrives in the database, the system tries to place it in the existing hierarchy. However, if it fails to classify the instance exactly, it adds the instance as an exception to the class found to be closest. The present system initiates restructuring only when the number of exceptions to a class exceeds a predefined threshold value. The threshold value is related to support and confidence of an association rule in the context of data mining.

Evaluation of rate of Miss Classification:

Our test set consists of 312 images comprising of the images from each of the cars, fishes, flowers, deer, elephants and planes classes.

The no. of images considered from each of the classes is shown in the flowing table:

CAR	AIRPLANE	FLOWER	DEER	ELEPHANT	FISH
99	*39*	*36*	*20*	*21*	*97*

The error rate corresponding to each of the classes is described below:

Percentage of misclassification with cars	*= 08.08%*
Percentage of misclassification with Planes	*= 30.07%*
Percentage of misclassification with Flowers	*= 22.22%*
Percentage of misclassification with Deers	*= 30.00%*
Percentage of misclassification with Elephants	*= 09.52%*
Percentage of misclassification with Fishes	*= 31.95%*

The average percentage of misclassification *=21.47%*

Conclusion:

The system is tested with a collection of images in our database consisting images of cars, planes, flowers, animals and fishes. It has been established that along with shape features, successive inclusion of texture and color features improves the system performance. In our present work shape features are given more weightage against color and texture. So the shape feature predominantly separates among the different classes.

It has been observed that, as the precision of the features is increased to higher values (e.g. from 8-bit to 16-bit), the algorithm yields better separation between feature values and hence better results. It has also been observed that subdivision of the classes into more sub classes reduces the error rate.

From the figures showing misclassification rates from the previous section it is apparent that some classes show higher percentage of misclassification. This higher rates of misclassifications are observed because; the objects present in the corresponding classes are of different shapes than those which are used for training (e.g. Deers in sitting posture are misclassified against the training set which was fed to recognize the deers in the normal standing posture).

In case of fishes, by subdividing *Fish class* into two sub classes consisting of Normal Fishes and Circular shaped fishes, it has been observed that the misclassification rate has reduced considerably.

The percentage of misclassification with respect to Fishes after sub classification is observed to have reduced to 14.43%.

Hence the average error rate i.e., the avg. percentage of misclassification (with respect to the total images taken together after sub classification of fishes) has been reduced to 16.02%.

So finally, it has been concluded that similar sub classification of other data items could reduce the overall misclassification rate to a great extent.

References

[1] Subhamoy Maitra & Aditya Bagchi. Dynamic Restructuring of Classification Hierarchy towards Data Mining.

[2] Sanjoy Kumar Saha. Intelligent Image Retrieval Using Visual Features and Relevance Feedback. PhD thesis, Bengal Engineering. & Science University.

[3] Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules and Sequential Patterns. PhD thesis, University of Wisconsin Madison.

[4] Han Jiawei, Kamber Micheline. Data mining: concepts and techniques, Morgan Kaufmann, 2001.

[5] Mitra Sushmita, Acharya Tinku. Data mining - multimedia, soft computing, and bioinformatics, John Wiley, 2003.

[6] RaghuRam Krishnan.Database Management System, McGraw-Hill, 1998.

Chapter 1

1.1 Introduction

Images are being extensively used in every sphere of our life. Apart from overwhelming influence of television, common people look for images in newspapers, advertisements, item catalogues, entertainment, education, architecture, painting and many others. Professionals use image in criminology (e.g., fingerprint identification, face recognition), medicine (e.g., case-based diagnosis from radiographs or scan data), education (e.g., searching for material in Library), fashion design, historical archiving, fine arts and so on. Most of the cases the problem is to find a desired image from a large collection or, in other words, retrieve images similar to the image at hand from large number available in some collections. Image search and retrieval is a field of very active research since the 1970's. However, the field has observed a steady exponential growth in recent years as a result of unparalleled increase in the volume of digital images. Thousands of images are generated everyday for different applications. These images are either stored in a local database or are available from remote ones. Thus a huge amount of information is out there and can easily be accessed through world-wide web. Professionals of various fields intend to access and utilize these images for their purpose. However, we cannot access to or make use of the information unless it is properly organized for efficient browsing and retrieval, because searching and locating a desired piece of image from varied and large collection usually result in a total frustration. Two major research communities, namely Database Management and Computer Vision, are putting considerable effort towards the solution of this problem. Accordingly two major approaches have emerged: one being text based and the other visual based respectively.

Early systems of image retrieval exploited the capabilities of text based Database management Systems. Images are first manually annotated using a set of keywords that describe the content of the image best. Images are indexed and arranged using these keywords, finally images are retrieved based on text based query. Major research in this direction includes Data Modeling, Indexing Structure, Multi-dimensional Indexing, Efficient Searching and Query Design and Evaluation. However, these text-based image retrieval techniques face two major problems: labor intensiveness and annotation impreciseness. When image collection is large, enormous amount of man-hour is required to annotate those images manually. Problem became more and more acute since early 1990's when world-wide web allow access remotely placed image databases. The second problem is more crucial and is due to semantic of image content. Because of rich content in the images and the subjectivity of human perception, same image may be perceived differently by different persons. As a result, same image may be annotated by different set of keywords by different persons. Thus image annotation in general is neither unique nor adequate; hence affects the performance of image retrieval system to a large extent. This leads to development and flourishing the alternate approach, namely Content Based Image Retrieval (CBIR) system.

1.2 What is CBIR?

Content-based image retrieval (CBIR), also known as query by image content (QBIC) and content-based visual information retrieval (CBVIR) is the application of computer vision to the image retrieval problem, that is, the problem of searching for digital images in large databases. The term CBIR seems to have originated in 1992, when it was used by T. Kato to describe experiments into automatic retrieval of images from a database, based on the colors and shapes present. Since then, the term has been used to describe the process of retrieving desired images from a large collection on the basis of syntactical image features. The techniques, tools and algorithms that are used originate from fields such as statistics, pattern recognition, signal processing, and computer vision.

In CBIR systems the term “content-based” means that the search will analyze the actual contents of the image. The term 'content' in this context might refer colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords, which may be laborious or expensive to produce. There is growing interest in CBIR because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image retrieval. Textual information about images can be easily searched using existing technology, but requires humans to personally describe every image in the database. This is impractical for very large databases, or for images that are generated automatically, e.g. from surveillance cameras. It is also possible to miss images that use different synonyms in their descriptions. Systems based on categorizing images in semantic classes like "cat" as a subclass of "animal" avoid this problem but still face the same scaling issues.

In case of classical information retrieval system, the data in the text databases are logically structured. But in CBIR system, the image database is essentially unstructured. The digitized image consists of an array of pixel intensities without any inherent meaning. Thus, like any kind of image processing, in CBIR system also one of the key issues is to extract useful information from the raw data. Subsequently, the issue like storage of data and efficient scheme for retrieval of desired images come into picture. Thus the major issues of CBIR are as follows:

- **a.** Identification of suitable ways of describing the image content
- **b.** Extracting such features from raw image.
- **c.** Providing compact storage for large database.
- **d.** Matching query and stored images in a way that reflects human similarity judgment.
- **e.** Efficient accessing of stored images by content.

The CBIR system architecture is essentially divided into two parts.

- i. In the first part images from the image database are processed off-line and indexing and meta-data construction is done.
- ii. In the second part , the query image goes through the same process and compared with the metadata info to find out the similar images

1.2.1 Fundamental modules of CBIR system:

Generally the CBIR system can be grouped into three fundamental modules:

1. **Visual content or feature extraction :**

   Visual contents or features such as color, texture, pattern, image, topology, shape of objects and their layouts and locations within the image etc., are extracted from the images of an image database by applying various image processing algorithms.

2. **Multidimensional indexing :**

   An image can be represented by a multidimensional vector of the extracted features. The feature vector actually acts as the signature of the image. These extracted image features are stored as meta-data, and images are indexed based on these meta-data information. This feature can be assumed to be associated to a point in the multidimensional space.

3. **Retrieval:**

   In this step the query image is posed and asked to find the images similar to the given image. Retrieving similar images to the query image then boils down to finding the indices of those images in the N- dimensional search space, whose feature vectors are within some threshold of proximity to the point representing the query image.

1.2.2 Applications of CBIR Systems:

CBIR system can be used in a wide range of applications. Some of them are described as follows:

Crime Prevention:

Law enforcement agencies maintain a database of facial photographs of past suspects, fingerprints etc. In case of crime, they can compare evidence with the database images for identify matching.

Intellectual property:

In case of trademark registration, the new mark is to be checked with the existing marks to ensure that it will not create any confusion. It has been recognized as a prime application area of CBIR.

Architectural and engineering design:

The designer can search and find out similar designs from the design archives and those can be adapted to the current problem.

Medical diagnosis:

Modern medical Science relies heavily on diagnostic techniques like radiology, histopathology, computerized tomography etc. By comparing these medical images CBIR techniques can identify the similar past cases and thereby can aid diagnosis.

Geographical Information and remote sensing systems:

CBIR techniques can be used to aid the GIS. Searching by spatial attributes (finding target of interest in the close vicinity), analyzing the images to locate the regions of interest (area growing crops, flooded regions etc.) may be few such applications.

Journalism and advertising:

News paper agencies maintain archives of photographs illustrate articles or advertising copy, it is expensive to maintain detailed keyword. In this case CBIR has an important role to access the desired photographs.

1.3 Objective:

The basic objective of this project is to classify the images based on the extracted visual contents or features from a collection of large set of images present in the image database. It also aims at continuously restructuring the class hierarchy dynamically upon the arrival of new instances into the existing structure. In our present algorithm we used shape, texture and color features for the purpose of classification.

1.4 Over View of Work:

In the context of our algorithm shape, texture and color features are combined to form 57-dimensional feature vector. Based on these features a classification tree has been constructed that performs continuous restructuring of a class hierarchy. The class hierarchy represents relationship among different attributes (items of the database). When a new instance (transaction) arrives in the database, the system tries to place it in the existing hierarchy. However, if it fails to classify the instance exactly, it adds the instance as an exception to the class found to be closest. The proposed system initiates restructuring only when number of exceptions to a class exceeds a predefined threshold value. The threshold value is related to support and confidence of an association rule in the context of data mining. The system starts with an initial class hierarchy and class descriptions obtained from the initial training and keeps on modifying them as new instances are considered. These new instances are either taken from the underlying database or inserted afresh.

The system is tested with a collection of images in our database consisting images of cars, planes, flowers, animals and fishes. It has been established that along with shape features, successive inclusion of texture and color features improves the system performance.

Chapter 2

This Chapter deals with the description of basic terminology and definitions required for the understanding of the algorithms and implementation details which are described in the next chapter. Here we give introduction to basics of data mining and various types of features considered in our implementation.

2.1 Problem Formulation

The proposed algorithm works for a certain class of images. So, first we define the how the images are classified.

Depending on the contents, images may be grouped into following three classes:

1. Class of images containing a single dominant object (Class-1).
2. Class of images containing many objects of more or less equal significance (Class-2).
3. Class of images containing no objects of specific interest, but their combination appears very picturesque (Class-3).

The Class-3 is exemplified by outdoor scenery consisting mostly of sky, water bodies (like sea, river, Lake etc), grass field, beach etc. none of which is particularly important, but surely the combination is.

Images of a group of people, cluttered objects, busy area (e.g., railway station, market, city streets etc), business meeting etc. belong to the Class-2.

Finally, Class-1 contains images of our friend, relative, home, car, pet, object of our interest (e.g., ancient building, monument, sculpture and statue, biomedical mage, animal, bird etc), famous personality and so on. These objects, in the image, occupy the major area mostly at the centre and are sharply focused. There could be other objects too in the image, but those are given usually less emphasis while photographed and are treated as background. Hence, we say that Class-1 is by far large than that of Class-2 and Class-3 together.

Now, based on the background, the images of Class-1 can further be divided into two groups consisting of:

1. Images with non-textures or smooth textured background.
2. Images with highly textured background.

First group is most common, because when photos are taken usually uniform backdrop (such as curtain or wall in the studio or in the hall or even sky in case of outdoor object) is used as much as possible to assign more prominence of object of interest. However, in some cases it is not possible to have such controlled background (e.g., while photographing a wild animal, bird on a tree, a building surrounded by trees etc.) and we have dealt with most of the photos of the second group. However, it is more apparent that the images of first group are more common.

2.2 Feature Categories:

In our present algorithm the features used for the classification are broadly divided into three categories namely,

i. Shape Features.

ii. Texture Features.

iii. Color Features.

2.2.1 Shape Feature:

Major information content of an image is the shape of the objects present in it and their relative arrangement. Infant, there is considerable evidence that natural objects are primarily recognized by their shape. Shape can roughly be defined as the description of an object minus its position, orientation and size. Therefore, the shape features used for the object identification or matching should be invariant to translation, rotation and scale. In short, shape features are those properties of objects or of its parts that are same for visually similar type of objects and are different for visually dissimilar objects. The shape features are computed by measuring some geometrical attributes of the regions corresponding to the objects. Shape features may be calculated for the whole image or for each individual objects. In the former case we have mainly topological attributes like, number of objects, Euler number etc. In latter case features like area, perimeter, convexity, aspect ratio, circularity, elongated ness etc are computed. It should be noted that the design of a feature extraction algorithm depends on scheme of shape representation. Shape representation schemes can be divided into two categories:

i. Boundary based.
ii. Region based.

The former uses the outer contour of a region, whereas the latter uses the whole of it.

Some basic definitions of most commonly used shape features are described below.

Linear Symmetry: This visual property indicates, when an axis passing through the centre of gravity divides the object into two parts, how well the mirror reflection of one part about that axis matches with the other part. The axis for which the best match occurs is called the axis of symmetry and the error in matching provides a measure of symmetry.

Circularity: For a perfectly circular object, all the contour points are equidistant from the centre of gravity and the centre of gravity coincides with the center of the smallest circle encompassing the object.

Aspect ratio: Aspect ratio of an object signifies overall shape structure like some kind of elongated ness etc. It can be measured in terms of the ratio between height and width or length of major axis and minor axis etc. If the minimal bounding box of an object is square then aspect ratio is 1.

Concavity: Conventionally objects are concave or convex. Thus a measure of concavity (convexity) is a strong feature for describing the shape. Contour of an object consists of a collection of inward and outward curve segments, i.e., curve segments with positive or negative curvature. Some segments have zero curvature indicating straight line segment. In place of concavity, the curve segment is inward with respect to center of gravity. As a result, any arbitrary contour can be thought of as a segment of circular arc indicating concavity or convexity.

2.2.2 Texture Feature:

Texture is another feature that has been extensively explored by various research groups. Texture is an innate property of virtually all object surfaces, including fabric, bark, water ripple, brick, skin etc. In satellite images texture of a region can distinguish among grass land, beach, water body, urban area, etc.

The term texture is used to specify the roughness or coarseness of object surface. In an intensity image texture puts its signature as the variation in intensity from pixel to pixel. Texture measures look for visual patterns in images and how they are spatially defined. Textures are represented by texels which are then placed into a number of sets, depending on how many textures are detected in the image. These sets not only define the texture, but also where in the image the texture is located.

2.2.3 Color Feature:

Another widely used visual feature for CBIR is color. Main advantage of this feature is its invariance to size, position, orientation and arrangements of the objects. On the other hand, disadvantage is its immense variation within a single image. Several methods of color representation and estimating similarity measures are present in the literature. Color searches will usually involve comparing color histograms, though this is not the only technique in practice

2.3 Related Concepts

This section would discuss about some background materials to be used in the proposed system. Basic ideas about data mining have been covered.

2.3.1 Data Mining

Data mining covers the methods for finding interesting trends or patterns in large datasets. These discovered patterns help and guide the appropriate authority in taking future decisions. Generally data mining tools are expected to identify interesting patterns in the data with minimal user intervention. Since data mining efforts usually assume a very large volume of data, efficiency and scalability are two very important criteria for data mining algorithms.

The pattern discovered by data mining should properly portray the contents of the dataset and the nature of the application under consideration. The imperfectness should be expressed by approximate rules and should also be quantifiable. Discovering or mining association among different features present in an application domain has recently attracted a lot of attention.

Definition:

An *association rule* is of the form LHS $\Rightarrow$ RHS, where both LHS and RHS are sets of items.

This identifies that if every item of LHS is present in a transaction, then it is likely that the items in RHS will also be present. There exists two important measures for an association rule, one is support and another is confidence.

***Definition*:**

The *support* for a set of items is the percentage of transactions that contain all of these items.

***Remark*:**

The support for a rule LHS $\Rightarrow$RHS is support for the set of items LHS$\cup$RHS

Low support may imply that a rule has arisen purely by chance, whereas high support value may identify some relational pattern among the items.

Definition:

The *confidence* for a rule LHS $\Rightarrow$ RHS is $\frac{\text{Support (LHS} \Rightarrow \text{RHS)}}{\text{Support (LHS)}}$.

Remark:

Out of the transactions that have LHS, the percentage of transactions that have RHS as well, is the measure of confidence of the rule LHS $\Rightarrow$RHS

It indicates the degree of correlation between presences of these set of items.

2.4 Different Matching Procedures of Classification:

***Definition*:**

The *universal attribute* set U, $|U| = n$ is the set of all the attributes that will be considered for the application domain.

Here we are interested to find out relevant relationships among the items in U. Once the universal attribute set is identified, the attributes are considered in a specific $a_0, a_1, \ldots, a_{n-1}$ for convenience of representation. Since $a_i \in \{0, 1\}\ \forall i$, the ith bit of an n bit string represents presence and absence of an attribute.

Definition:

An *instance* (transaction) *I* is considered as a string of length n containing 0-1 values ($I \in \{0, 1\}^n$) implying the presence (1) and absence (0) of the items in the transaction.

Definition:

A *class* C consists of a set of attributes $C^A \subseteq U$. A class hierarchy consists of a set of classes, with a parent-child relationship among them.

Remark:

A class may be identified by a 0-1 bit string vectors of length n, where the ith vector is 0 if $a_i \notin C^A$, and corresponding feature value otherwise. C^A represents the set of attributes belonging to class C only and not the attributes inherited from the parent classes. The root class has no parent. The intermediate classes have one parent and one or more children. The leaf classes have one parent and no child.

2.4.1 Exact match

When the system considers a new instance, it is compared against the leaf level classes for exact match. If it fails, top-down search is done from the root for approximate classification. In this case matching may be possible only up to some class at intermediate level. Since further matching down the hierarchy fail, the new instance is flagged as an exception to the class corresponding to the intermediate node. In this method a lot of exceptions may get accumulated in each class. If the count of exceptions for a class exceeds the application specific threshold values the class hierarchy needs to be restructured. The different matching procedures for inserting an instance into the system are discussed below.

Definition:

The *total attribute* set C^{tA} attached to a class C is the union of all the attribute sets of the classes lying on the path from root to that class. Corresponding to total attribute set an n bit binary number is formed, with 1 at the i^{th} position if the attribute $a_i \in C^{tA}$ and 0 otherwise. This is called the *path identification* number of the class.

Definition:

An instance I is an *exact match* with a leaf level class C, if the binary number corresponding to I is equal to the path identification number of C.

Remark:

Here equality means the similarity of the bit patterns of the two n bit strings, one the instance I and another the path identification number of the class.

Remark:

Once an instance becomes exact match with one leaf class C, the respective count increases for all the classes on the path from the leaf class C to the root.

2.4.2 Perfect match

An instance *I* will be a perfect match with root class if all the attributes of root class are present in that instance. In case of *Perfect match the* system can classify an instance up to some depth in the tree. For all the classes, from the root downwards up to which the perfect match is found the match count is incremented by 1 for those classes.

For the class beyond which perfect match cannot be found an exception is flagged and corresponding exception value is stored and its count is incremented.

Let Fr(C) =$\bigcup C_i^A$, where C_i's are the peer classes of C, excluding C.

Definition:

An instance *I* will be a *perfect match* with a class C at depth i $\geq$ 1 if

1. It is a perfect match to a class C_p at depth i-1.
2. C_p is the parent of C.
3. All the attributes of C are present in the instance *I* i.e., $C^A \subseteq I$.
4. Let $A' = Fr(C) - C^A$. In all the attributes of A' the instance *I* should contain 0 value.

Remark:

For all the classes, from the root downwards up to which the perfect match is found the match count is incremented by 1.For those classes, if the threshold value with respect to no. of exceptions is crossed, a new class is created as a child to the corresponding class. The child class consists of the features with maximum exception count, ties are resolved arbitrarily.

Chapter 3

This Chapter deals with the description of various algorithms used in the implementation. The details of algorithms are described in the following sections. Here we describe different versions of each algorithm, their short comings and the enhancements in the successive versions.

3.1 Implementation Details:

The actual implementation process proceeds in two steps:

1. In the first step, the input features having feature values in the range (0 to 8) or (0 to16) are given. We convert each feature values into a binary string of 8-bit or16-bits. This reduces the given problem into a single level processing. The conversion is done as follows:

   Given a feature whose value is 'i' we make ith bit to be 1 and rest all bits to be 0s in the corresponding 8-bit/16-bit vector. Details of the algorithm are described in the following section.

NOTE: Here the feature values represent the strength corresponding to a feature i.e., we are not only considering the presence or absence of an item in a transaction, but also the exact quantity of it. Higher the feature, value higher is the strength.

2. In the second step, once the input in the binary format is available, we proceed to apply our algorithms on the modified input. We describe the details of the algorithms used in the rest of the sections.

3.1.1 Basic Class Structure:

The structure of main class, along with the description of each field in the comments is shown below:

```
struct class_node
{
 BYTE features [MAX_BYTE];   // feature vector corresponding to class //
 Char *class_name;                    //  Represents the name of the class //
 class_node *child [MAX_FEATURES];              // link to child nodes //
 int no_child;                                  // no. of children available //
 BYTE path_id [MAX_BYTE];           // path identification no. of class //
 int match_count;                     // counts the no. of perfect matches //
 int no_ex;                                // no. of exceptions initially -1 //
 struct exception_list *ex_list [THRESHOLD];
                                           // exception list with its count //
 struct class_node *parent;                      // points to parent node //
 class_node *next;                        // this points to next node of leaf//
}
```

This structure is designed to handle the exceptions:

```
struct exception_list
{
        BYTE exception [MAX_BYTE];                   // actual exception //
        int count;                          // no. of occurrences of exception //
 }
```

3.2 Preliminary Algorithmic Details:

3.2.1 Algorithm for converting Feature values into binary format:

Algorithm 1:

Input: File containing ordered features values

Output: File containing binary ordered feature values

```
Begin Algorithm
    Open the input file to read data
    Open the output file to write data
    While (! EOF)                    // Read word by word
    Do begin
    Read 'ch' from the input file
        For i = 8 to 0               // In case of 16-bit replace 8 by 16
        Do begin
          j = i-ch
          If (j=0)
              Print 1 on output file
          Else
              Print 0 on output file
        End for

    End while
    Close input file
    Close output file

End Algorithm
```

Description:

In case of input features, the absence of a particular feature is marked by value 0. The feature presence is indicated by a value in the range of (1-8) or (1-16) signifying the strength of the feature.

The above two level feature presence/absence value is mapped into a single level representation by converting it into 8-bit/16-bit binary string for each feature.

The features used in our problem are given below.

1-8	foldreg_len []
9	regslope
10	regerror
11	regconcavity
12	regasp_ratio
13	regsymmetry
14	circarea
15	circlen
16	circmin
17	asp_ratio
18	asp_ratio_median
19	symmetry_1
20	symmetry_2
21	concavity_len
22-49	texture []
50-57	color []

Here the number against each feature name indicates the position(s) corresponding to that feature in the given order.

3.2.2 Algorithm for Building Class hierarchy:

Algorithm2 V 1.0:

Input: Modified binary input feature file

Output: Class hierarchy

```
/* This is the first version of building dynamic class hierarchy. */

 Begin Algorithm
        Call init_build ()
        For i = 0 to no_items
        Do begin
            Call exact_match (identifying_feature)
            If there doesn’t exist an exact match with any of leaf nodes
            Then
             Call Perfect_match (identifying_feature)
       End for
 End Algorithm
```

3.2.2.1 Algorithm init_build V 1.0:

Input: File containing ordered binary feature values of training data

Output: Finds the identifying features w.r.t given data & creates Corresponding root Class

```
/* Here we consider 100% support for finding identifying features */
 Begin Algorithm
   While (! EOF)
   Do begin
        Read values row wise from the file into an array features
   End while
     For i=0 to Max_Columns
```

```
            For k=0 to Max_Rows
             Do begin
                 identifying_feature [i] = identifying_feature [i] & features[k][i]
             End for
      End for
          Update Root with identifier and node information
End Algorithm
```

Description:

Here *init_build ()* will take file containing training set as input and finds identifying features with respect to that class. This identifying feature becomes the feature vector for the root corresponding to that class.

Root node is updated with the information corresponding to that node.

Note: Identifying Feature in this version is identified as the feature with 100% support.

3.2.2.2 Algorithm exact_match V 1.0:

Input: Key vector corresponding to test data

Output: Return status corresponding to match

```
Begin Algorithm
   Temp = leaf_list
    While (Temp)
    Do begin
        For I = 0 to Max_Columns
        Do begin
                Result = Temp ^ Key_vector
                If Result ! = 0
                Then
```

```
            Next (Temp)
        End for
         If there exists a Temp with Result 0 for all columns
        Then
            Update Temp
            return 1
        Otherwise
            return 0
          End while
End Algorithm
```

Description:

Here *exact_match ()* will search all the leaf nodes in the leaf list and tries to find whether the features corresponding to any node has exact match with the query features i.e. key features.

If there is an exact match then the match count of that class is incremented and the information corresponding to that node is updated, otherwise perfect match method is invoked.

3.2.2.3 Algorithm perfect_match V 1.0:

Input: Key vector corresponding to test data.

Output: Flags an exception at the class where perfect match fails,
Creates a new child class, if no. of exceptions cross threshold

```
Begin Algorithm
        Temp = root
        If key_vector is not perfect match for root
           Then
             return 0
```

```
    While (True)
    Do begin
       If key_vector is perfect match for Temp
       Then
            Update match_count
            Remove matched attributes from the instance of key_vector

            For all children (Temp)
            Do begin
                    Check if key_vector is perfect match to child (Temp)
            End for

            If no. of perfect match is 1 with $i^{th}$ child
            Then
                    Temp = child$_i$ (Temp)
                    Continue the loop from beginning
           Else
                    If ( no_exception (Temp) < Threshold )
                    Then
                            Flag exception at Temp
                            Update the Temp
                    Else
                            Find the exception j with max count
                            Create child $C_j$  for the Temp
                            Update $C_j$
                            Update Temp
                            Update leaf_list
    End while
End Algorithm
```

Description:

In case of *Perfect_match ()* the system can classify an instance up to some depth in the tree. For all the classes, from the root downwards up to which the perfect match is found the match count is incremented by 1 for those classes.

For the class beyond which perfect match cannot be found an exception is flagged and corresponding exception value is stored and its count is incremented.

If the threshold value with respect to no. of exceptions is crossed, a new class is created as a child to the corresponding class. The child class consists of the features with maximum exception count, ties are resolved arbitrarily.

Remarks:

- This version deals with only one class at a time, so it is difficult to verify the distinguishing power of the algorithm to separate the elements of different classes.
- In this version we deal with 8-bit data which have some limitations such as precision insufficiency in certain cases.
- It fails to properly classify / prevent features belonging to other classes to enter into current class because of the following reasons:
    a) Considering 100% support acts as a bottle neck because, the training data may not always represent complete real time dynamic data. So same flexibility must be provided.
    b) This support consideration may lead to selection of minimal identifying features which, in most of the cases may not exactly represent the total identifying feature set of

that class. Hence it results in the objects of other classes to enter the query class.

c) The values corresponding to an identifying feature may not be same for all the elements of that class, so the ϵ-neighborhood must be considered at the time of classifying and finding the identifying features for the better results.

In order to overcome the short comings mentioned above we make following modifications / additions to the previous version.

Here we describe the algorithms where modifications / additions are made in the previous version.

3.3 Enhanced Algorithmic Details

Here we describe the full fledged algorithm with all the required changes and/or additions to the previously mentioned algorithms.

3.3.1 Algorithm for Building Class hierarchy:

Algorithm3 V 1.1:

```
Input:    Modified binary input feature file
Output:   Class hierarchy
Begin Algorithm
    Call build_root ()
    For i = 0 to no. of training classes
    Do begin
            Call init_build ()
    End for
```

```
        While ( ! EOF )
         Do begin
            Read the Test data from file into array features
         End while
        For i = 0 to no_items
        Do begin
        Call exact_match (feature)
        If there doesn't exist an exact match with any of leaf nodes
        Then
            Call Perfect_match (feature)
   End for
End Algorithm
```

3.3.2 Algorithm build_root V 1.1:

Input: Global root information

Output: Creates a Global root node

```
Begin Algorithm
        Create a root node
        Update root
End Algorithm
```

Description:

Creates a node corresponding to global root and updates corresponding information.

3.3.3 Algorithm init_build V 1.1:

Input: File containing ordered binary feature values of training data

Output: Finds the identifying features w.r.t given data & creates
Corresponding root Class as a child to global root

```
//** Here we consider neighborhood values and variable two level
     Support (T1, T2) for finding Identifying features.          **//

 Begin Algorithm
   While (! EOF)
   Do begin
         Read training data from the file into an array features
   End while
  Store = Ø     /* store is an array which maps a value with its count */
   For i=0 to Max_Columns
          For k=0 to Max_Rows
           Do begin
              If feature[i] [k] exists in Store
               Then
                   Increment corresponding count value
             Else
                     Insert feature[i] [k] into Store
                     Initialize corresponding count as 1
          End for
      Find $count_j$ > T2
        If there exists such j
        Then
                 For all ϵ-neighborhood ( $feature_j$ ) taken together
                 If count ( $feature_j$ ) = T1
                 Then
                          identifying_feature [i] = $feature_j$
                 Else
                          identifying_feature [i] = 0
  End for
      Update Root with identifier and node information

End Algorithm
```

Description:

In case of finding the identifying features, a two level processing is done where two separate thresholds are set say T1, T2. Here T1 is taken as 100% and T2 value has been set according to experimental results based on inputs.

Initially, to consider ϵ-neighborhood for a particular feature, we check for all the features whose count value satisfies T2 to qualify as a candidate. Once the candidate has been identified, we take the ϵ-neighborhood to verify whether the features with neighborhood taken together has 100% support (here $\epsilon = 2$). If both T1 and T2 constraints are met then the corresponding features becomes identifying features.

The advantage of above method is that it may increase the identifying feature set which means better separability.

Here the n^{th} call of init_build procedure will assign the created class as n^{th} child to the global root.

Note: *The Algorithm for evaluating exact match is similar to the one described in previous version.*

3.3.4 Algorithm perfect_match V 1.1:

Input: Key vector corresponding to test data

Output: Flags an exception at the class where perfect match fails,
Creates a new child class if no. of exceptions cross threshold

```
Begin Algorithm
    Temp = root
    If key_vector is not perfect match for root
      Then
        return 0
    While (True)
    Do begin
      If key_vector is perfect match for Temp
      Then
          Update match_count
          Remove matched attributes from the instance of key_vector
      Else
          If key_vector is perfect match for ϵ-neighborhood (Temp)
          Then
              Update match_count
          Remove matched attributes from the instance of key_vector

          For all children (Temp)
          Do begin
              Check if key_vector is perfect match to child (Temp)
              or ϵ-neighborhood (child (Temp))
          End for

    If temp = root
    Then
          If root has more than 1 match
          Then
           Choose the class with max matches
          If there exists a tie with no. of max matches
```

```
            Then
               Choose the class with max direct match

        Else
            If no. of perfect matches is 1 with ith child
            Then
                Temp = $child_i$ (Temp)
                Continue the loop from beginning
            Else
                If ( no_exception (Temp) < Threshold )
                Then
                    Flag exception at Temp
                    Update the Temp
                Else
                    Find the exception j with max count
                    Create child $C_j$ for the Temp
                    Update $C_j$
                    Update Temp
                    Update leaf_list
        End while

End Algorithm
```

Description:

Here a global root with roots of different classes as children is considered. This helps to properly differentiate within the classes based on maximum match of feature vector.

If tie is observed in the max match while classifying a feature from root to its children, the feature having max direct match is chosen as the candidate.

In this version an additional check is made while comparing for the perfect match i.e., if the key vector fails to have a perfect match directly with a node in question then, the ϵ-neighborhood is considered to check for the perfect match.

The rest of the algorithm for handling exceptions and creating child nodes works similarly as described in the previous version corresponding to this algorithm.

Remarks:

1. In this version we deal with 8-bit data which have some limitations such as precision insufficiency i.e., incase of differentiating elements of different classes having close feature properties, it fails to give good results.

2. Here proper prioritization of the features is missing i.e., identification of the key (important) features corresponding to a specific class is necessary for better results while classification.

Finally the above version has been modified slightly to work with normalized 16-bit feature values. It has been observed that the results produced were good due to better separation among the features because of the extension of precision value.

3.4 Observations:

The following observations were made while working with 16-bit feature values:

1. It provided better separation among the closely related classes. Here we tested on classifying different animals .It provided better results compared to that of the 8-bit feature values.

2. It has given the results with an accuracy up to 90% with some classes while classification.

In the next chapter we show the images corresponding to the training data and the respective results. The results are shown as a classification tree depicting the input images that entered the corresponding classes, exceptions, miss classifications as well as correct classifications.

Chapter 4

4.1 Experimental Results:

The feature classification methods with the dynamic restructuring capability presented in Chapter 3 are applied and tested on a collection of images in our database containing the images of cars, planes, flowers, deer, elephants and fishes.

The classification model is built on the initial tree which is constructed from the training data. First we show the images used as the training data from each of the classes, followed by the classification tree. The notation followed in the construction of the classification tree is described in the next section.

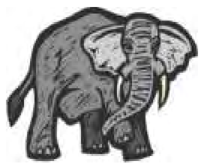
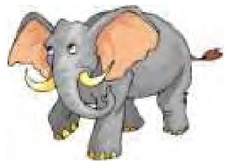
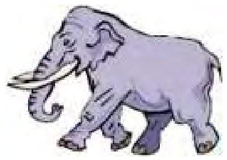
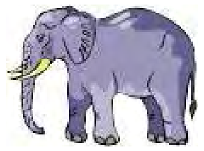
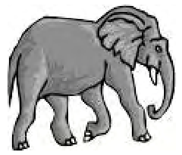
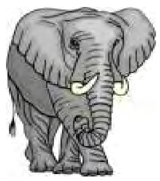
With each class we show the images that entered corresponding class and depict how they are classified i.e., we indicate if the image is classified correctly or misclassified or gone as an exception to some existing class in the hierarchy.

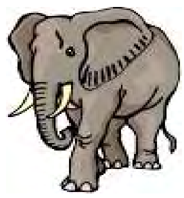
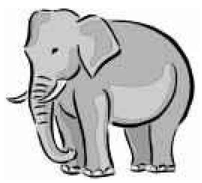
Finally we show the table representing the no. of items taken from each of the classes as the test set and the rate of misclassification corresponding to each of the classes as well as the average rate of misclassification.

4.2 Training Data

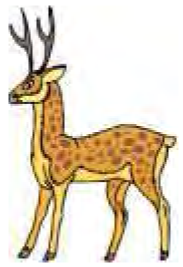
The following images are taken as training data for flowers:

The following images are taken as training data for Elephant:

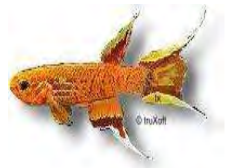







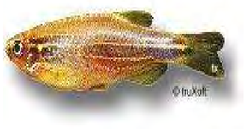



The following images are taken as training data for Deers:



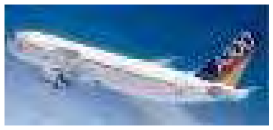
The following images are taken as training data for Fish:

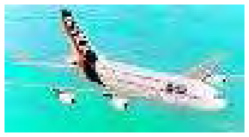
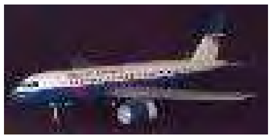




The following images are taken as training data for Cars:

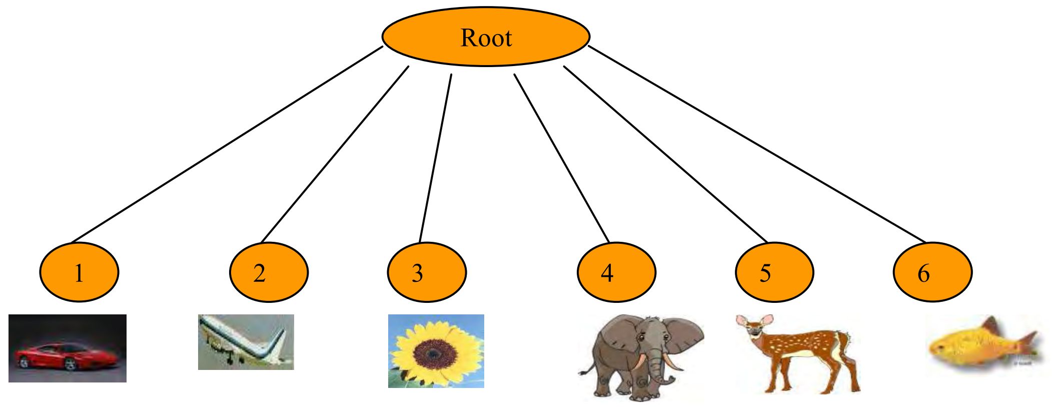
The following images are taken as training data for Aero Planes:



4.3 Class Structure

CLASS – HIERARCHY



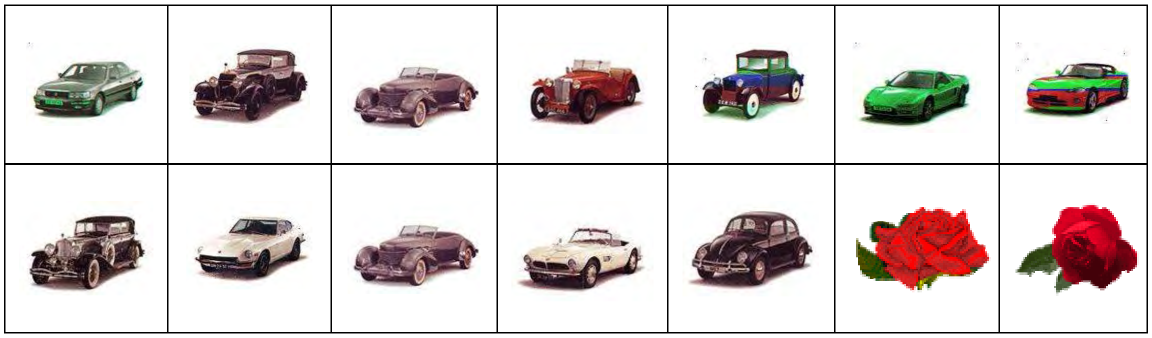
The labels corresponding to the figure are described as follows:

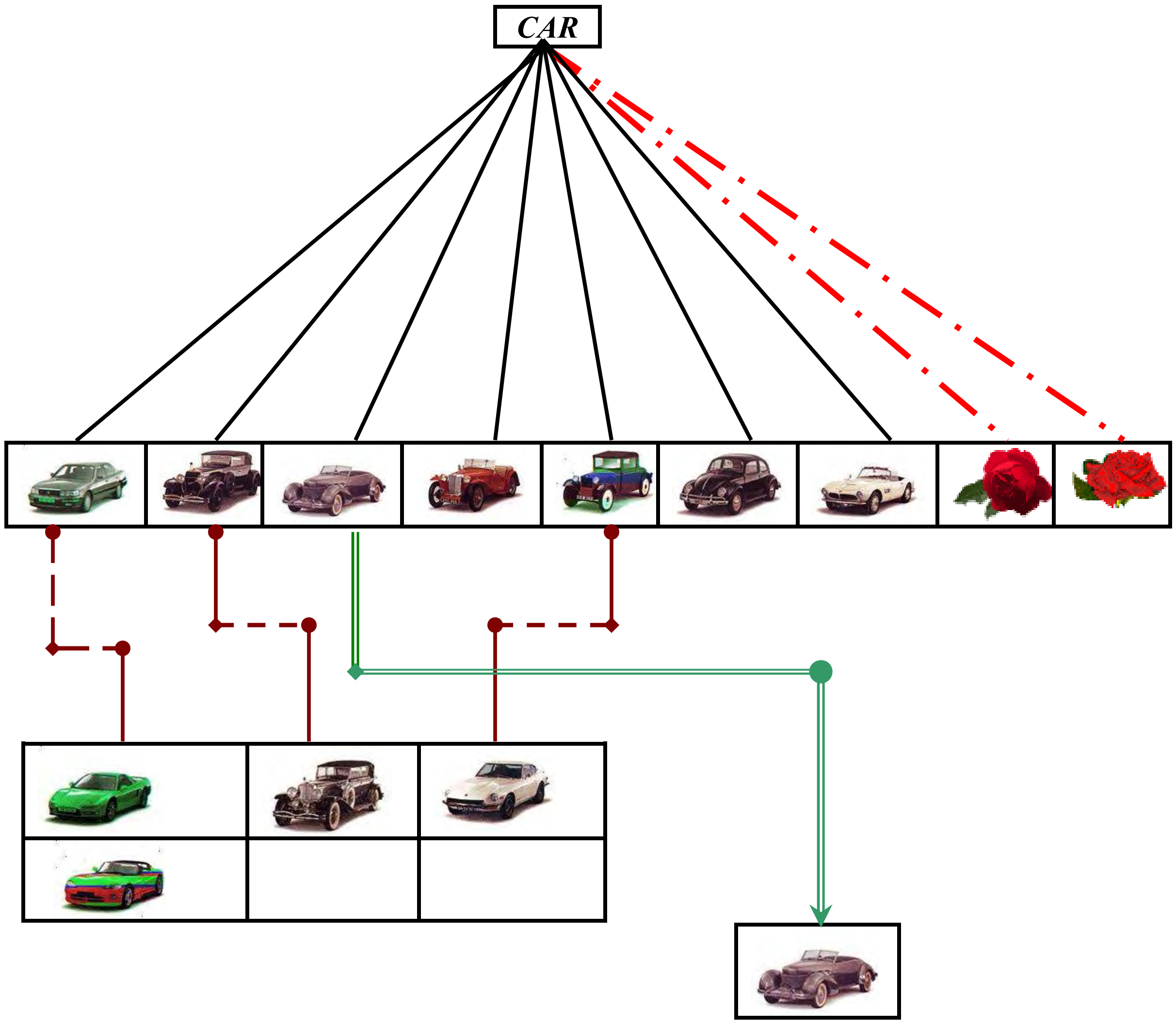
1. CAR
2. PLANE
3. FLOWER
4. ELEPHANT
5. DEER
6. FISH

The following table describes the notations used in the tree construction.

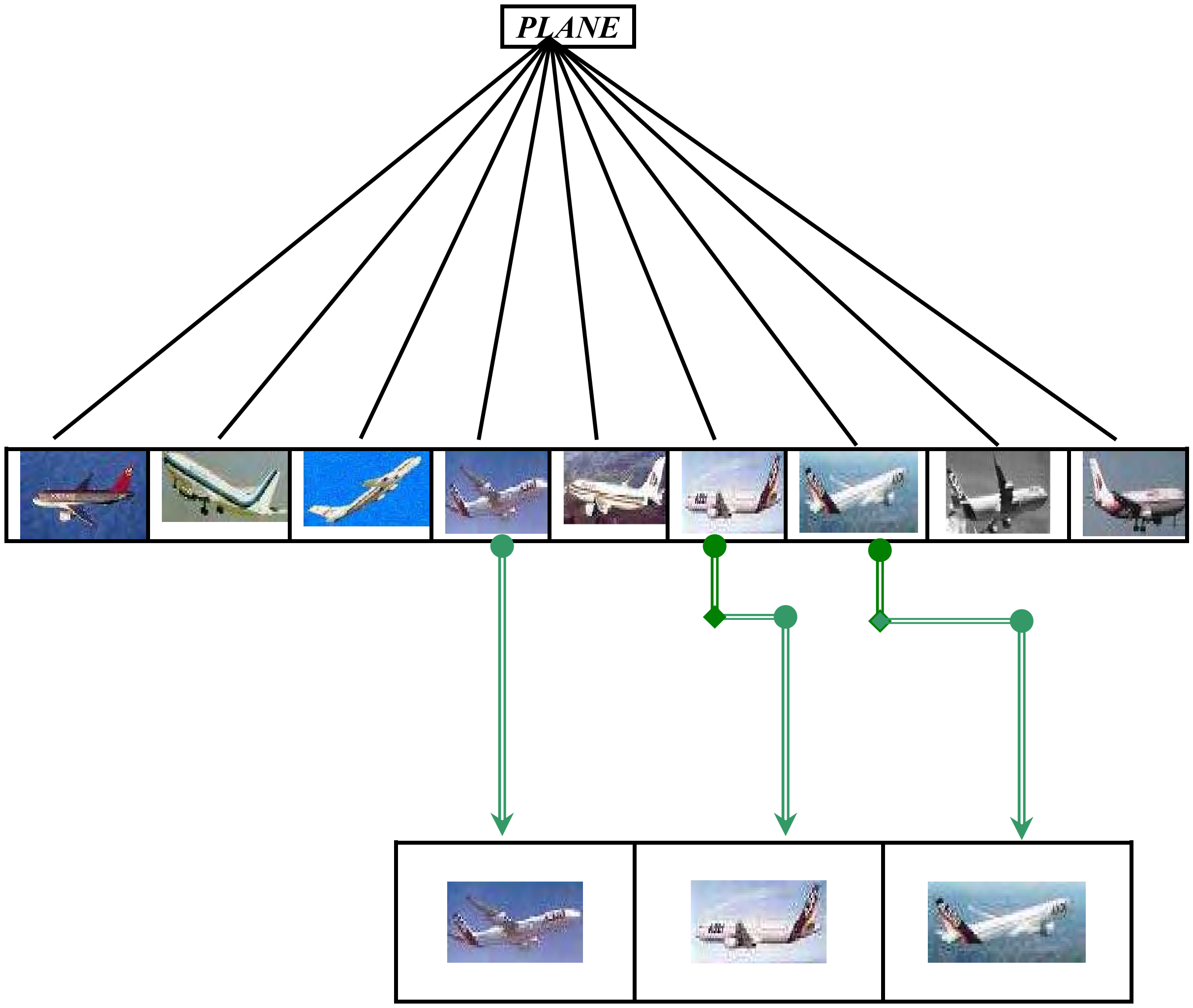
Correct Classification	**Misclassification**	**Exception**	**Exact Match**
———	— · — · —	- - - - -	⟹

The following are the set of images which entered Car class

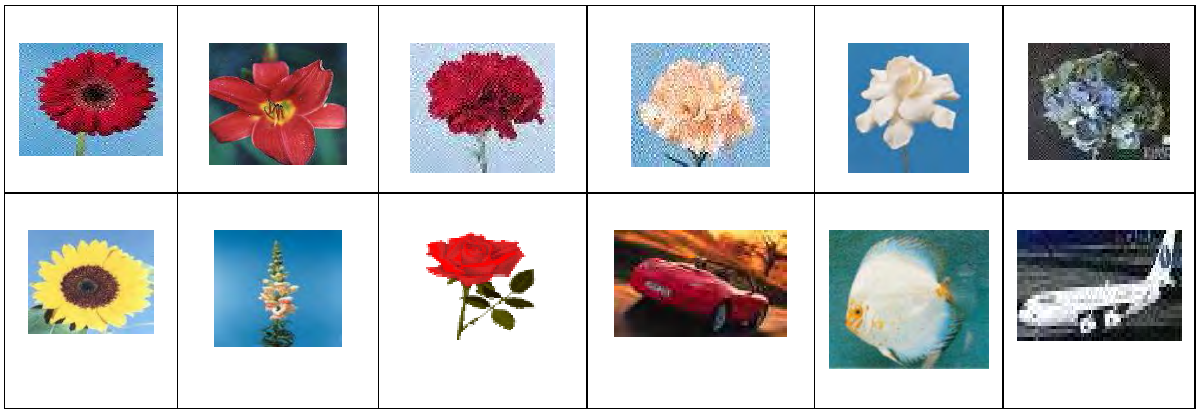


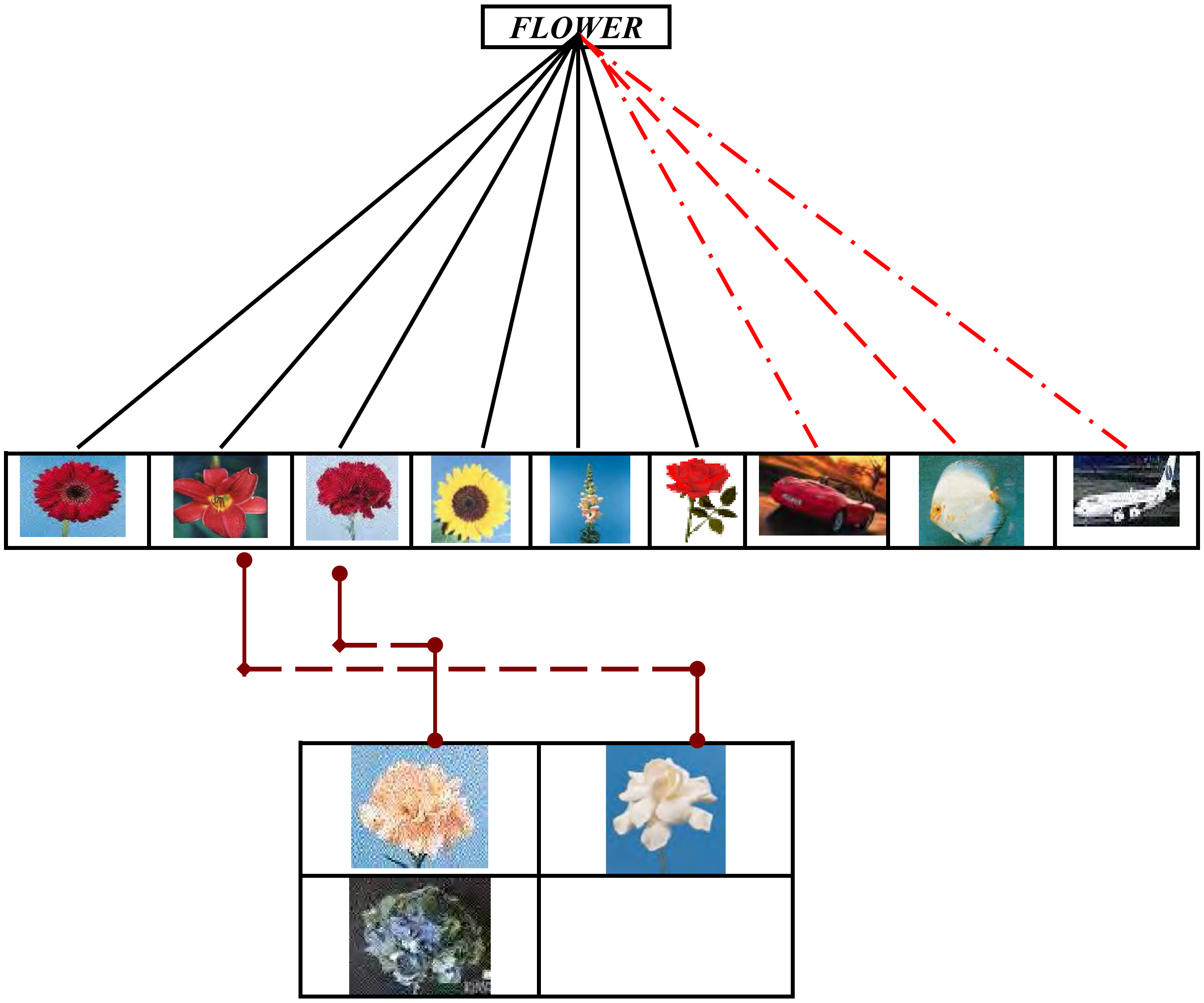


The following are the set of images which entered Airplane class

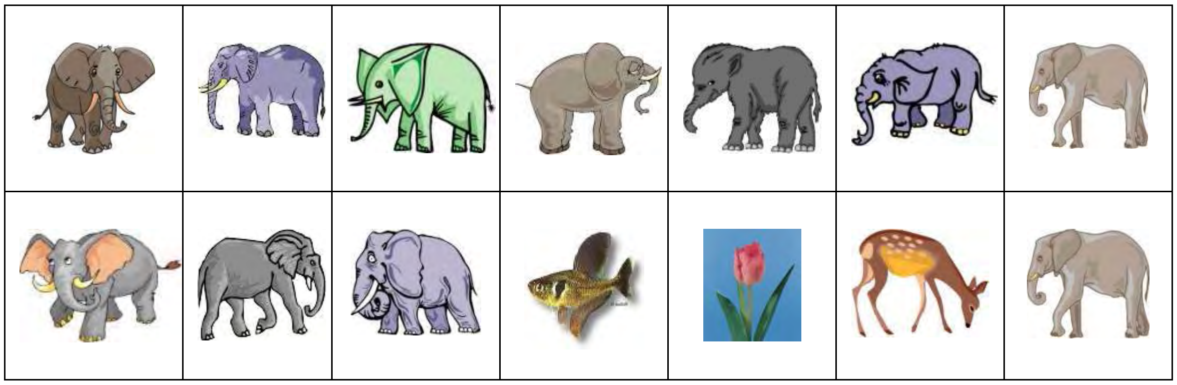


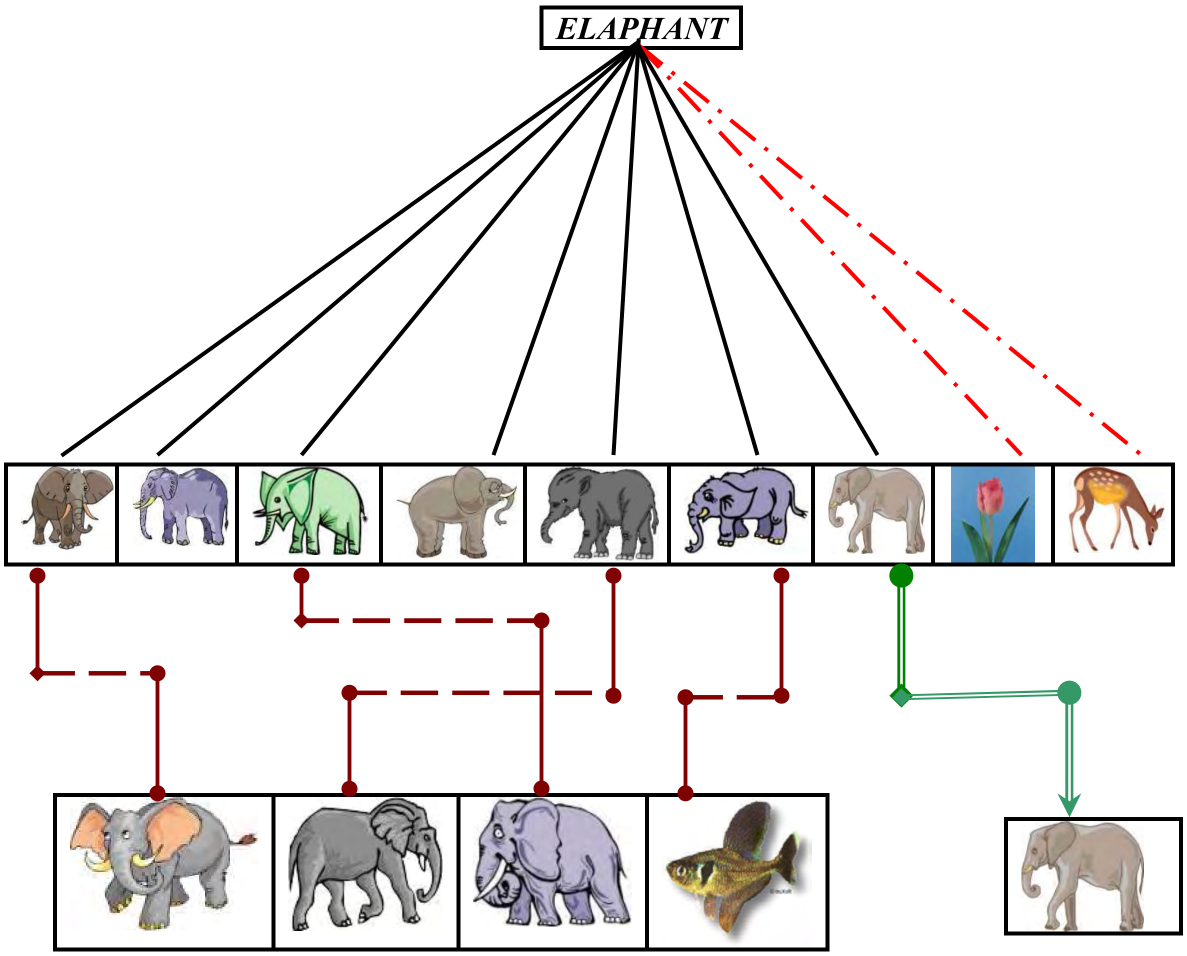
The following are the set of images which entered Flower class



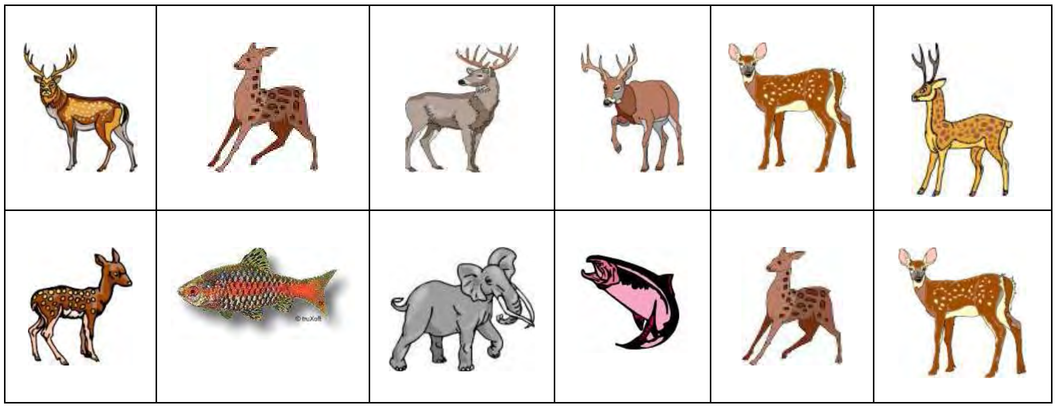


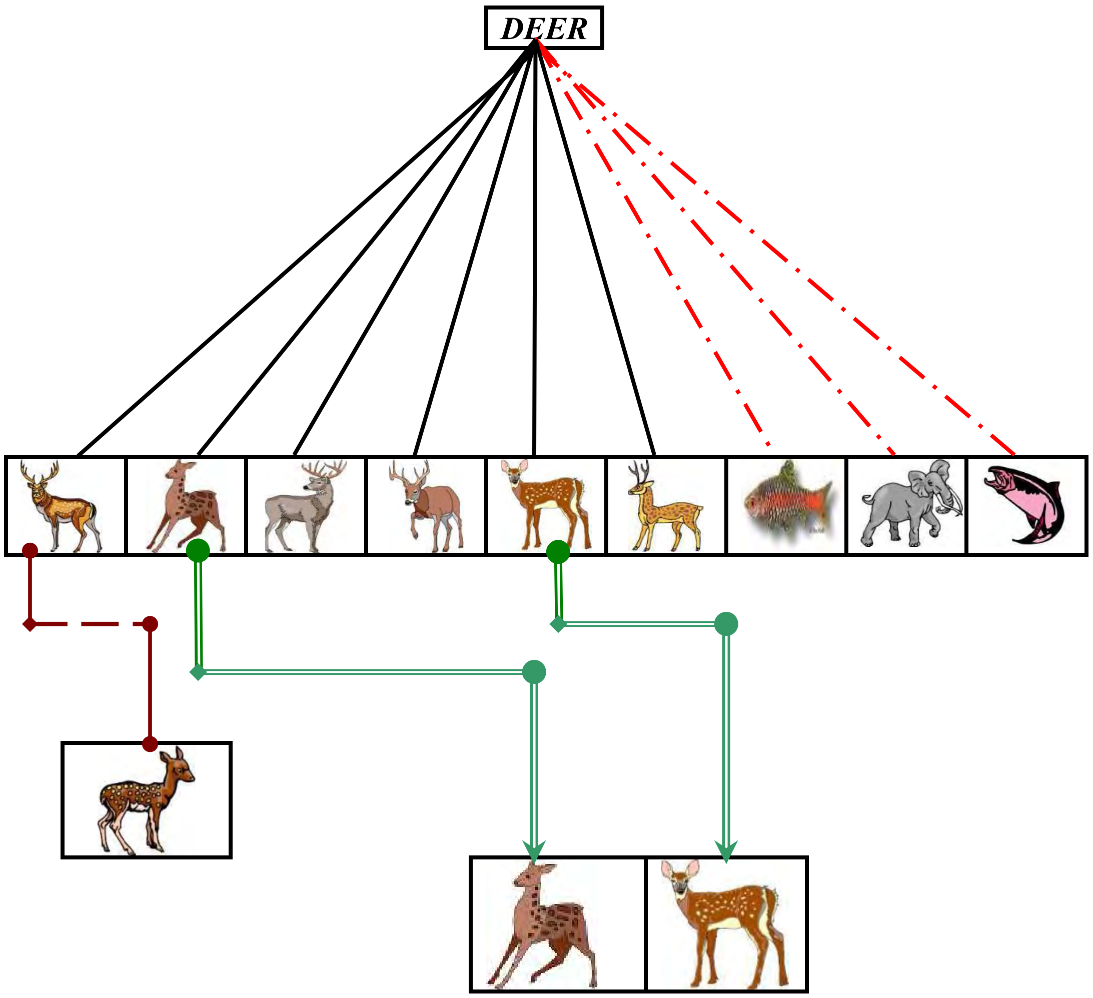
The following are the set of images which entered Elephant class



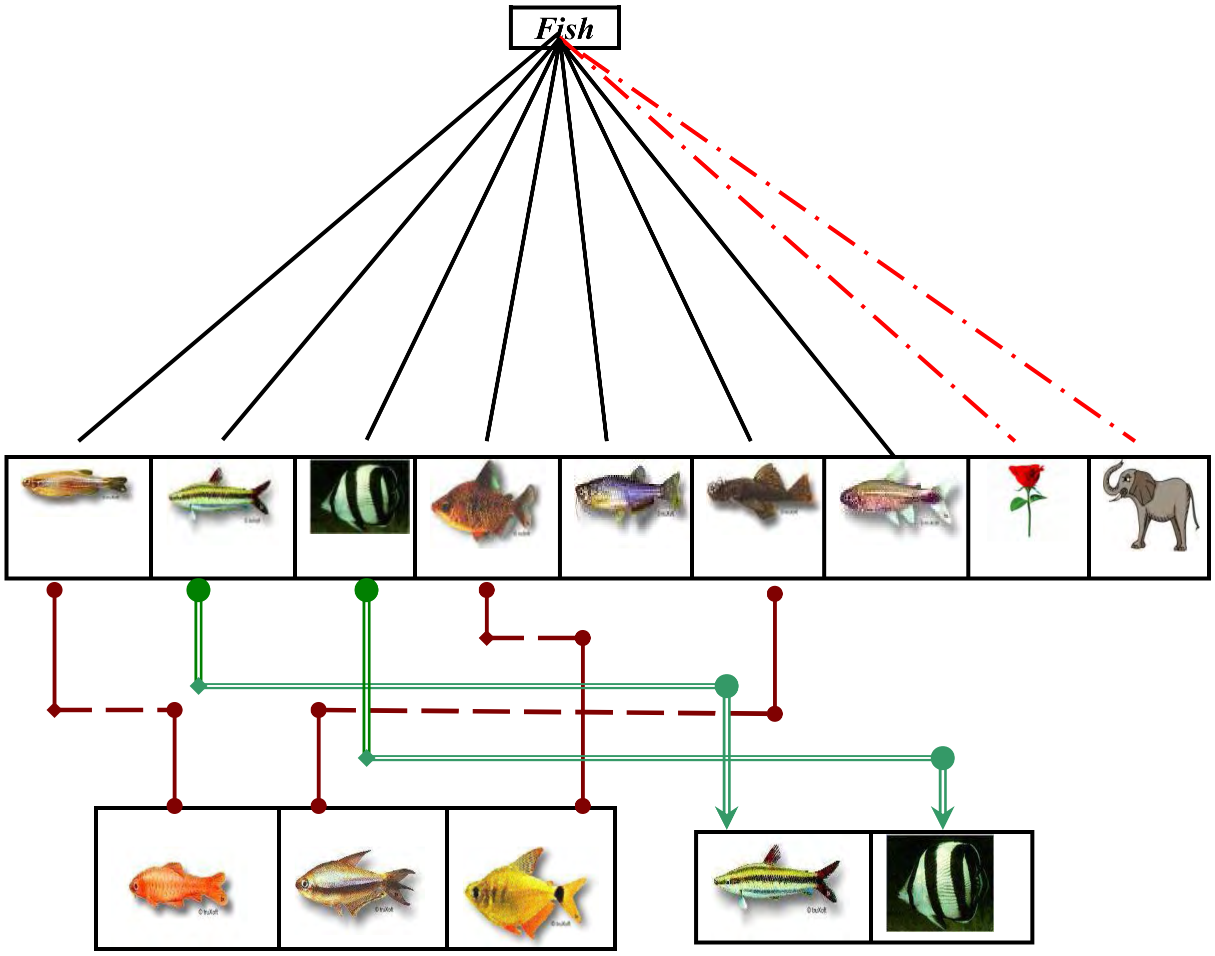


The following are the set of images which entered Deer class





The following are the set of images which entered Fish class



4.4 Evaluation of rate of Miss Classification:

Our test set consists of 312 images comprising of the images from each of the cars, planes, flowers, animals and fishes classes.

The no. of images considered from each of the classes is shown in the flowing table:

CAR	AIRPLANE	FLOWER	DEER	ELEPHANT	FISH
99	*39*	*36*	*20*	*21*	*97*

The error rate corresponding to each of the classes is described below:

Percentage of misclassification with cars	*= 08.08%*
Percentage of misclassification with Planes	*= 30.07%*
Percentage of misclassification with Flowers	*= 22.22%*
Percentage of misclassification with Deers	*= 30.00%*
Percentage of misclassification with Elephants	*= 09.52%*
Percentage of misclassification with Fishes	*= 31.95%*

The average percentage of misclassification *=21.47%*

Chapter 5

5.1 Conclusion:

The system is tested with a collection of images in our database consisting images of cars, planes, flowers, animals and fishes. It has been established that along with shape features, successive inclusion of texture and color features improves the system performance. In our present work shape features are given more weightage against color and texture. So the shape feature predominantly separates among the different classes.

It has been observed that, as the precision of the features is increased to higher values, the algorithm yields better separation of feature values and hence better results. It has also been observed that subdivision of the classes into more classes reduces the error rate.

From the figures showing misclassification from the previous section it is apparent that some classes show higher percentage of misclassification. This higher rates of misclassifications are observed because; the objects present in the corresponding classes are of different shapes than those are used for training (e.g. Deers in sitting posture are present against the training set which was fed to recognize the deers in the normal standing posture). In case of fishes, by subdividing *Fish class* into two sub classes consisting of Normal Fishes and Circular shaped fishes, it has been observed that the misclassification rate has reduced considerably.

Percentage of misclassification with Fishes (after sub classification) is observed to be 14.43%.

Hence the average error i.e., the avg percentage of misclassification (with respect to the total images taken together) has been considerably reduced to 16.02%.

5.2 Scope of Future work:

The following future possibilities can be explored towards the improvement and extension of the above mentioned algorithms.

1. The results corresponding to 32-bit and higher dimensional feature values may be explored and can be compared with the present results.

2. Subdivision of Classes and their (annotations) labeling can be done for providing better semantics to each of the sub classes and improve the rate of misclassification.

3. The results can be studied by varying the weightage corresponding to each of the feature types, (i.e. each of the shape, texture and color features can be assigned different priorities) at the time of classification.

4. Addressing the issues corresponding to queries can be handled and the strategies corresponding to reduction of search and retrieval time can be explored.

References

[1] Subhamoy Maitra & Aditya Bagchi. Dynamic Restructuring of Classification Hierarchy towards Data Mining.

[2] Sanjoy Kumar Saha. Intelligent Image Retrieval Using Visual Features and Relevance Feedback. PhD thesis, Bengal Engineering. & Science University.

[3] Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules and Sequential Patterns. PhD thesis, University of Wisconsin Madison.

[4] Han Jiawei, Kamber Micheline. Data mining: concepts and techniques, Morgan Kaufmann, 2001.

[5] Mitra Sushmita, Acharya Tinku. Data mining - multimedia, soft computing, and bioinformatics, John Wiley, 2003.

[6] RaghuRam Krishnan. Database Management System, McGraw-Hill, 1998.

[7] Adamo Jean-Marc. Data mining for association rules and sequential patterns, Springer-Verlag, 2001.

[8] Albert A J, Wayne Niblack. Image storage and retrieval systems, SPIE Publisher, 1992.

[9] Gonzalez Rafael C, Woods RichardE. Digital image processing, Addison-Wesley, 1999.

[10] A. Blaser, Database Techniques for Pictorial Applications, Lecture Notes in Computer Science, Springer Verlag, 1989.

[11] C. Faloutsos et al, "Efficient and effective querying by image content", Journal of intelligent information system, Vol.3, pp.231-262, 1994.