6-2015

Content-Based Image Retrieval using Deep Learning

Anshuman Vikram Singh
avs5913@rit.edu

Content-Based Image Retrieval using Deep Learning

by

Anshuman Vikram Singh

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Roger S. Gaborski

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

June 2015

The thesis “Content-Based Image Retrieval using Deep Learning” by Anshuman Vikram Singh has been examined and approved by the following Examination Committee:

Dr. Roger S. Gaborski
Professor
Thesis Committee Chair

Prof. Thomas J. Borrelli
Senior Lecturer

Srinivas Sridharan
PhD Student

Acknowledgements

I want to take this opportunity to show my gratitude towards the people who have helped me in completing my Masters capstone Thesis. First and foremost the person who I owe a special obligation is Prof. Gaborski, for his constant guidance in the course of my work. Since the beginning I have worked with him on various projects and he has supported me through each of them. It is with his cognizant efforts and sincerity that my thesis has been successful. I will also take the opportunity to acknowledge the contribution of Kenny Davila for his full support and assistance during the development of the work. He took out time from his busy work load to help me whenever I was stuck and guided me to finish explaining me the mistakes I would make. He was really patient with me during the course of this work. I will also like to thank Professor Borrelli for agreeing to be on my committee and help me out whenever I needed his help. I will not like to miss the opportunity to acknowledge the contribution of my family. It is because of them that I am able to get here and finally am presenting my capstone thesis. I want to thank them for being there for me always.

Abstract

Content-Based Image Retrieval using Deep Learning

Anshuman Vikram Singh

Supervising Professor: Dr. Roger S. Gaborski

A content-based image retrieval (CBIR) system works on the low-level visual features of a user input query image, which makes it difficult for the users to formulate the query and also does not give satisfactory retrieval results. In the past image annotation was proposed as the best possible system for CBIR which works on the principle of automatically assigning keywords to images that help image retrieval users to query images based on these keywords. Image annotation is often regarded as the problem of image classification where images are represented by some low-level features and the mapping between low-level features and high-level concepts (class labels) is done by supervised learning algorithms. In a CBIR system learning of effective feature representations and similarity measures is very important for the retrieval performance. Semantic gap has been the key challenge for this problem. A semantic gap exists between low-level image pixels captured by machines and the high-level semantics perceived by humans. The recent successes of deep learning techniques especially Convolutional Neural Networks (CNN) in solving computer vision applications has inspired me to work on this thesis so as to solve the problem of CBIR using a dataset of annotated images.

Contents

Acknowledgements . . . . . **iii**

Abstract . . . . . **iv**

1 Introduction **1**

2 Background **4**
2.1 Related work 4
2.1.1 Content-Based Image Retrieval 4
2.1.2 Deep Learning 5
2.1.3 Distance Metric Learning 6

3 Previous Approach **8**
3.1 Bag of Words 8
3.1.1 Image Description 9
3.1.2 Query Image 10
3.1.3 Segmentation 11
3.1.4 Region Classification 12
3.1.5 Image Matching 13
3.1.6 Image Retrieval 14
3.2 Hypothesis 14
3.3 Evaluation 15

4 Architectural Overview **16**
4.1 Convolutional Neural Network 16
4.1.1 Sparse Connectivity 17
4.1.2 Shared Weights 18
4.1.3 Convolutional Layer 18
4.1.4 Max-Pooling 20
4.1.5 Full-Model: LeNet 20

4.2 Dataset 21
4.3 Training the Network 22
4.4 System Design 24

5 Analysis 26
5.1 Experiment 26
5.2 Evaluation 27

6 Conclusions 28
6.1 Future Work 28

Bibliography 30

A Retrieval Results 32

List of Tables

5.1 Training Results 27
5.2 Evaluation Table 27

List of Figures

3.1 Annotated Image Description. 9
3.2 Query Image. 11
3.3 Segmented Query Image. 12
3.4 Region Classification. 13
3.5 Image Matching. 14
3.6 Image Retrieval Process. 15

4.1 Sparse Connectivity [6] 17
4.2 Shared Weights [6] 18
4.3 Convolutional Layer [6] 19
4.4 Full LeNet Model[6] 20
4.5 Example of Image and Annotation 22
4.6 System Design 24
4.7 Query Image 24

A.1 Beach Scene 32
A.2 Desert Scene 33
A.3 Snowy Mountain Scene 33
A.4 Forest Road Scene 34
A.5 Highway Scene 34

Chapter 1

Introduction

In the recent past the advancement in computer and multimedia technologies has led to the production of digital images and cheap large image repositories. The size of image collections has increased rapidly due to this, including digital libraries, medical images etc. To tackle this rapid growth it is required to develop image retrieval systems which operates on a large scale. The primary aim is to build a robust system that creates, manages and query image databases in an accurate manner. CBIR is the procedure of automatically indexing images by the extraction of their low-level visual features, like shape, color, and texture, and these indexed features are solely responsible for the retrieval of images [8]. Thus, it can be said that through navigation, browsing, query-by-example etc. we can calculate the similarity between the low-level image contents which can be used for the retrieval of relevant images. Images are a representation of points in a high dimensional feature space and a metric is used to measure the similarity or dissimilarity between images on this space. Therefore, those images which are closer to the query image are similar to it and are retrieved.

Feature representation and similarity measurement are very crucial for the retrieval performance of a CBIR system and for decades researchers have studied them extensively. A variety of techniques have been proposed but even then it remains as one of the most challenging problems in the ongoing CBIR research, and the main reason for it is the semantic gap issue that exists between the low-level image pixels captured by machines and high-level semantic concepts perceived by humans. Such a problem poses fundamental challenge of Artificial Intelligence from a high-level perspective that is how to build and train

intelligent machines like human to tackle real-world tasks. One promising technique is Machine Learning that attempts to address this challenge in the long-term. In the recent years there have been important advancements in machine learning techniques. Deep Learning is an important breakthrough technique, which includes a family of machine learning algorithms that attempt to model high-level abstractions in data by employing deep architectures composed of multiple non-linear transformations.

Deep learning impersonates the human brain that is organized in a deep architecture and processes information through multiple stages of transformation and representation, unlike conventional machine learning methods that are often using shallow architectures. By exploring deep architectures to learn features at multiple level of abstracts from data automatically, deep learning methods allow a system to learn complex functions that directly map raw sensory input data to the output, without relying on human-crafted features using domain knowledge.

In the recent studies like Hinton et al [5] and Wan et al [10] encouraging results have been reported for applying deep learning techniques in applications like image retrieval, natural language processing, object recognition among others. The success of deep learning inspired me to explore deep learning techniques with application to CBIR task for annotated images. There is limited amount of attention focusing on CBIR applications even though there has been much research attention of applying deep learning for image classification and recognition in computer vision.

In this thesis, I will work with a deep learning method for solving the CBIR task for images that have been annotated by humans. There has been no study of this approach being applied to such a dataset. More about the dataset has been discussed in the other Architectural Overview section. I use a framework of deep learning for CBIR by applying a state-of-the-art deep learning method, that is, convolutional neural networks (CNNs) for learning feature representations from image data. I will be training large scale deep convolutional neural networks for learning effective feature representations of images. Each image will be assigned a binary value for each class present in it. I am dealing with 8

classes namely water, sky, snow, car, ground, tree, building and mountain. Each image will be assigned with an 8 bit binary number where each bit represents each class, which I will be storing in an index. When a query image is given, it will also be trained and a 8-bit binary number for it will be generated and will be matched against the index. All the images with the matching binary number will be returned as the closest match for the query image. I expect to get better results compared to other approaches which have been used for this particular problem and dataset.

Chapter 2

Background

2.1 Related work

My study involves the concepts of content-based image retrieval, distance metric learning and convolutional neural network. The work will be evaluated against my previous work with Bag-of-words model. In this section I briefly review the work done previously in these concepts and my Bag-of-words model.

2.1.1 Content-Based Image Retrieval

Content-based image retrieval (CBIR) for decades has been one of the most researched field of computer vision. CBIR aims to search for images through analyzing their visual contents, and thus image representation is the crux of CBIR.[[14], [5], [7]] In the past there has been a variety of proposed low-level feature descriptors for image representation, ranging from global features like color features, edge features, texture features, GIST and CENTRIST, and recent local feature representations, such as the bag-of-words (BoW) models using local feature descriptors (SIFT, SURF). Conventional CBIR approaches usually choose rigid distance functions on some extracted low-level features for multimedia similarity search, such as Euclidean distance or cosine similarity. However, the fixed rigid similarity/distance function may not be always optimal to the complex visual image retrieval tasks due to the grand challenge of the semantic gap between low-level visual features extracted by computers and high-level human perceptions. Therefore, in the recent past there

have been a surge of active research efforts in the design of various distance/similarity measures on some low-level features by exploring machine learning techniques. Among these techniques, some works have focused on learning to hashing or compact codes. There has been a proposed mapping learning scheme for large scale multimedia applications from high-dimensional data to binary codes that preserve semantic similarity.

2.1.2 Deep Learning

Deep learning refers to a class of machine learning techniques, where many layers of information processing stages in hierarchical architectures are exploited for pattern classification and for feature or representation learning [10]. It lies in the intersections of several research areas, including neural networks, graphical modeling, optimization, pattern recognition, and signal processing, etc. [5] Yann LeCun adopted the deep supervised back-propagation convolutional network for digit recognition. In the recent past, it has become a valuable research topic in the fields of both computer vision and machine learning where deep learning achieves state-of-the art results for a variety of tasks. The deep convolutional neural networks (CNNs) proposed by Hinton came out first in the image classification task of Imagenet classification with deep convolutional neural networks. The model was trained on more than one million images, and has achieved a winning top-5 test error rate of 15.3% over 1, 000 classes. After that, some recent works got better results by improving CNN models. The top-5 test error rate decreased to 13.24% in by training the model to simultaneously classify, locate and detect objects. Besides image classification, the object detection task can also benefit from the CNN model, as reported in. Generally speaking, three important reasons for the popularity of deep learning today are drastically increased chip processing abilities (e.g., GPU units), the significantly lower cost of computing hardware, and recent advances in machine learning and signal/information processing research. Over the past several years, a rich family of deep learning techniques has been proposed and extensively studied, e.g., Deep Belief Network (DBN), Boltzmann Machines (BM), Restricted Boltzmann Machines (RBM), Deep Boltzmann Machine (DBM), Deep Neural

Networks (DNN), etc. Among various techniques, the deep convolutional neural networks, which is a discriminative deep architecture and belongs to the DNN category, has found state-of-the-art performance on various tasks and competitions in computer vision and image recognition.

Specifically, the CNN model consists of several convolutional layers and pooling layers, which are stacked up with one on top of another. The convolutional layer shares many weights, and the pooling layer sub-samples the output of the convolutional layer and reduces the data rate from the layer below. The weight sharing in the convolutional layer, together with appropriately chosen pooling schemes, endows the CNN with some invariance properties (e.g., translation invariance). My work is similar to the work of Ji Wan et al.[10] but differs from them in the sense that the dataset I am using is different from the ones they have used in their study. Also my approach of image matching will be completely novel which has not been used in any study similar to mine.

2.1.3 Distance Metric Learning

Distance metric learning (DML) is an important concept of image retrieval which has been studied very extensively in machine learning [[4], [7]]. In this section I will discuss some already existing work for DML which can be organized by different leaning settings and principles.

Most of the current DML studies work with 2 types of data or side information when dealing with training data formats: pairwise constraints where the constraints for must-link and cannot-link are given and triplet constraints which consists of similar and dissimilar pair. There are studies which use the class labels directly for DML by following a typical machine learning scheme like large margin nearest neighbor (LMNN) algorithm. I have gone with the use of class labels directly for DML.

There are typically 2 groups into which distance metric learning can be categorized with respect to different learning techniques: the local supervised approach, where metric learning is done on the local sense when the given local constraints from neighboring

information are satisfied, and the global supervised approach where all the constraints are satisfied simultaneously for metric learning on a global setting.

Most of the current DML studies use the batch learning method as a learning methodology where before the training task the whole collection of training data must be given and a model is trained from scratch. The key concept on which distance metric learning is based is that for an optimal metric the distance between similar images should be minimized and distance between dissimilar images is maximized.

Chapter 3

Previous Approach

3.1 Bag of Words

The image features are treated as words in order to apply the Bag of Words model to image classification [9]. A bag of visual words [13] in computer vision is defined as a vector of occurrence counts of a vocabulary of local image features [7]. In my project I used a dictionary of 40 words. To compute the key-points I first used SURF and then compared the results with SIFT so as to be sure which was working better for our project. SURF is a robust local feature detector. It uses an integer approximation to the determinant of Hessian blob detector, which can be computed extremely quickly with an integral image (3 integer operations). I used the HESSIAN THRESHOLD as 600. SIFT is an algorithm to detect and describe local features in images. The local image gradients are measured at the selected scale in the region around each key-point. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination. The method proposed by me used JSEG segmentation to segment the query image into regions. I then extracted color features to describe each region and SURF features from the entire image. The texture features were extracted using Gabor Filters. After all features are extracted I computed bag of words using the SURF of each region and combine with color and texture to generate feature vector. A random forest classifier was used to assign a class to each region and then we compute a similarity score against every image on the dataset based on the current region labels and rank them by this score. After ranking the images the top n (n is the number of resultant images required by user) images were retrieved based

on the similarity score. The results obtained in this study were not of benchmark standards which inspired me to work on finding a solution for the same problem but with a different approach.

3.1.1 Image Description

The images in our dataset contain annotations of different regions in the form of XML files. The Extensible Markup Language(XML) annotations provide the annotated image description of each image in the dataset as shown in fig.3.1. With the help of XML annotations we generate a mask which gives us the region masks of that image. The combination of region masks and the XML annotations is used to generate descriptions of the image based on 3 main features. The color features, texture features and description of images using key-points and Bag of words. These annotated image description are stored in an index in the form of a dictionary so as to easily access them.

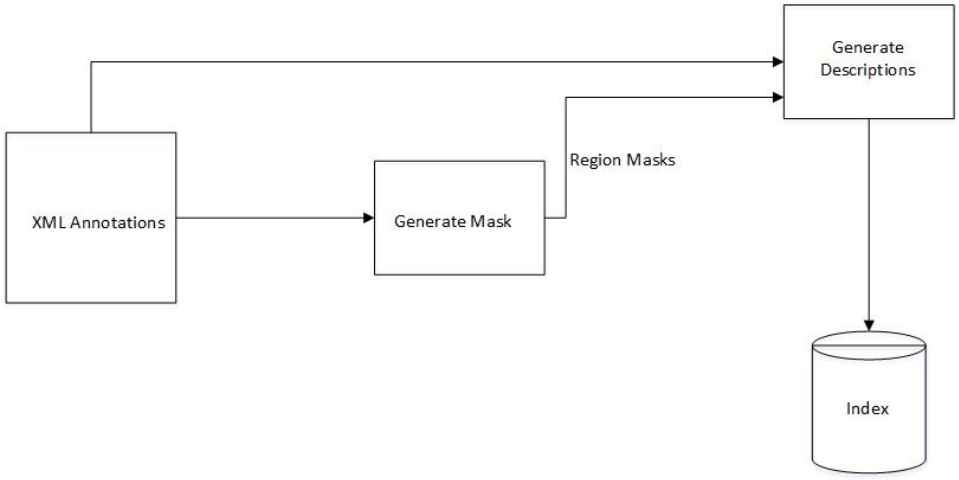

Figure 3.1: Annotated Image Description.

Color Features

We use a combination of dominant color, average color, dominant channel and fuzzy color histogram to describe the color feature. The dominant color uses representative colors to characterize the color information in the required region of an image thus making it a compact and efficient descriptor. Local features of an image can be well represented by a dominant color descriptor which helps in fast and efficient retrieval of images from large datasets. The average color descriptor returns the average of all colors present in the image and compares to it. The dominant channel descriptor takes into consideration the dominant tone per channel and returns the percentage of the dominant channels. Fuzzy 3D color histograms are required to compute dominant color. Fuzzy version are more balanced for colors that fall between color bins. We have used only 8 color bins in this project.

Texture Features

We used Gabor Filter as a texture feature descriptor. Gabor Filter is a linear filter used for edge detection. It is an image filter that can be used to describe texture of the image. The Gabor Filters are of any arbitrary size and orientation and are good to detect edge orientations in images. The only drawback of Gabor Filters is that it is scale-sensitive. We also added the average and standard deviation of brightness for each region to complement the information provided by the Gabor Filter.

3.1.2 Query Image

The query image is a user input image which he wants to use as a sample to retrieve images from the dataset. The query image can be from any source and need not be from our dataset. The system takes the input query image and uses JSEG segmentation which is explained in the next section to segment the image. The segmented image is used to generate the region masks and from these region masks feature extraction takes place which gives a feature vector as explained in fig.3.2. The feature vectors are passed into a region classifier (in our system it is a random forest classifier) which gives classified regions as shown in fig.3.4.

The classified regions give a final description of the image as to which region is mapped to which of the 9 classes.

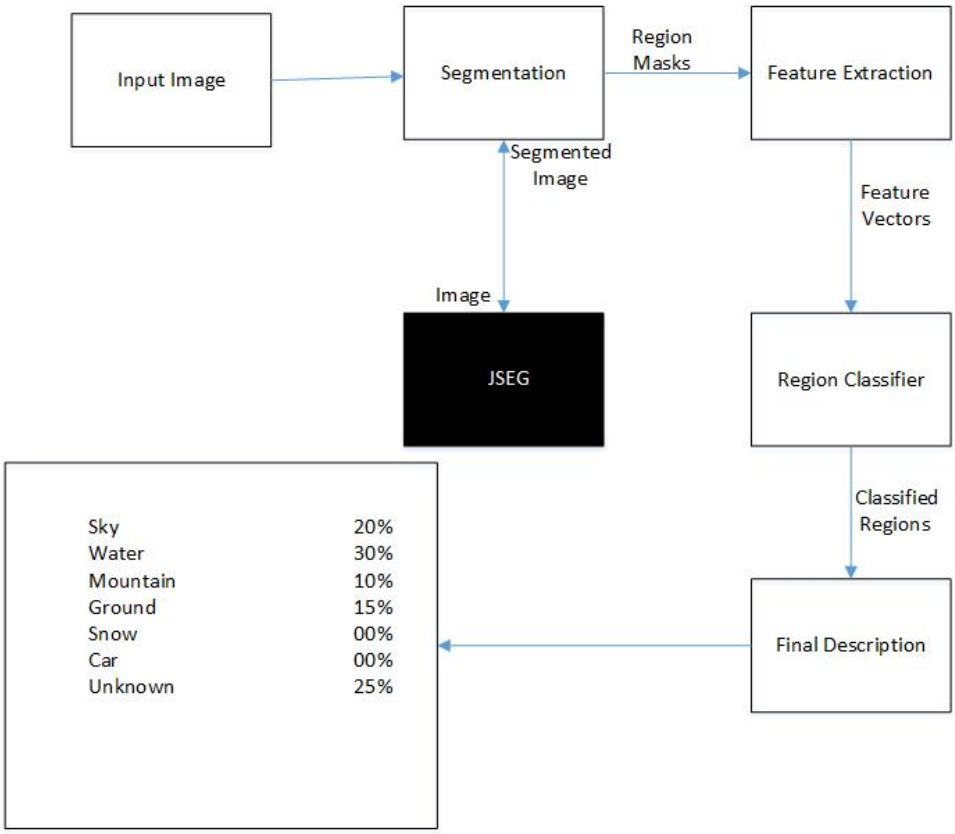

Figure 3.2: Query Image.

3.1.3 Segmentation

We needed segmentation of images for the retrieval part of the project. We used JSEG segmentation for this project as it is considered to be one of the best segmentation algorithm around for segmenting color images. The reason for this is that it takes into consideration not only color but also the texture while segmenting the image. Images are segmented in an unsupervised manner based on color-texture regions by JSEG which is includes performing color quantization and spatial segmentation independently. In the color quantization step,

the regions in the image are differentiated by quantizing the colors in the image to several representative classes. A class map of the image is then formed by replacing the image pixels by their corresponding color class labels [3]. As shown in fig.3.1 the segmentation part is more of a black box method. Fig.3.3 shows segmentation of a sample query image. For this project we used the already implemented version of the algorithm and made a script to process the images with the application. The implemented version of JSEG can be found at:
http://vision.ece.ucsb.edu/segmentation/jseg/software/

Figure 3.3: Segmented Query Image.

3.1.4 Region Classification

Once the image is segmented using JSEG we get the region masks based on the segmentation of region. These region masks are used for feature extraction so as to give a feature vector on which we apply a region classifier i.e. Random Forest in this case so as to give us classified regions like in fig.3.4 .

We have color coded the 9 classes in the region classification for our convenience. In the image shown in fig.3.4 dark blue region is classified as water, light blue is sky, brown is ground, dark gray is unknown and yellow is mountain. We store the region classification of all the images in our dataset match them with the region classification of the query image as mentioned in the next section.

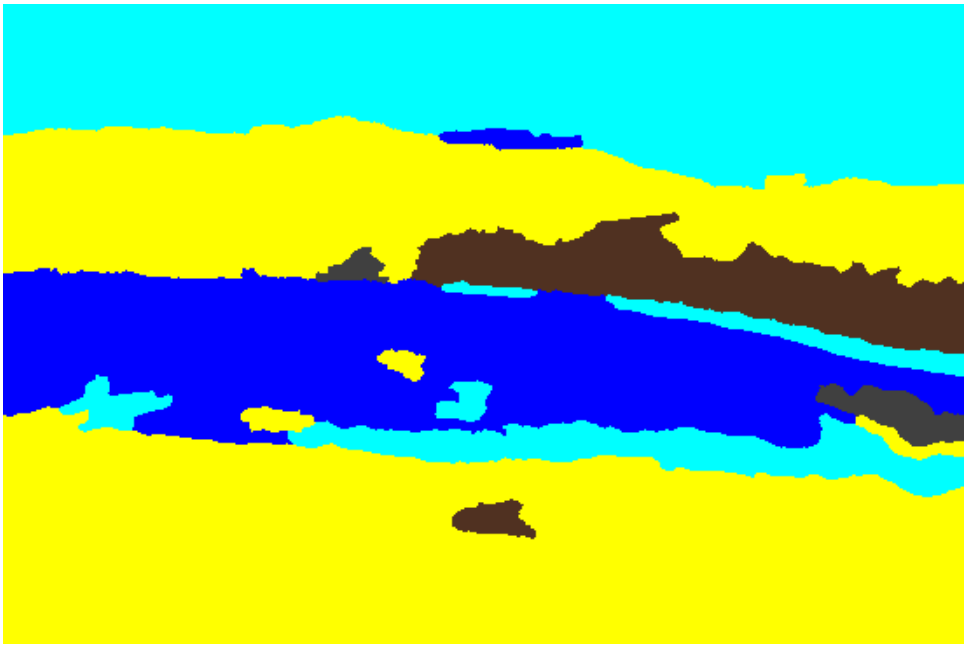

Figure 3.4: Region Classification.

3.1.5 Image Matching

We take a query image and segment it using JSEG algorithm. It returns us segmented region mask as .gif files. As openCV cannot upload the .gif files we use the Pygame library to load it as a 2D array so that it can be processed. Once we get the 2D array we separate out the region masks for each region in the image. We calculate the class percentage of each region in the query image and match it with class percentage of all the other images in the database so as to generate a similarity measure score. It works using ”Histogram Intersection” which means taking two histograms and choosing the minimum value on each bin. Then, you add

those values and the result is the similarity score. The images with the higher similarity score are returned based on how many images you want.

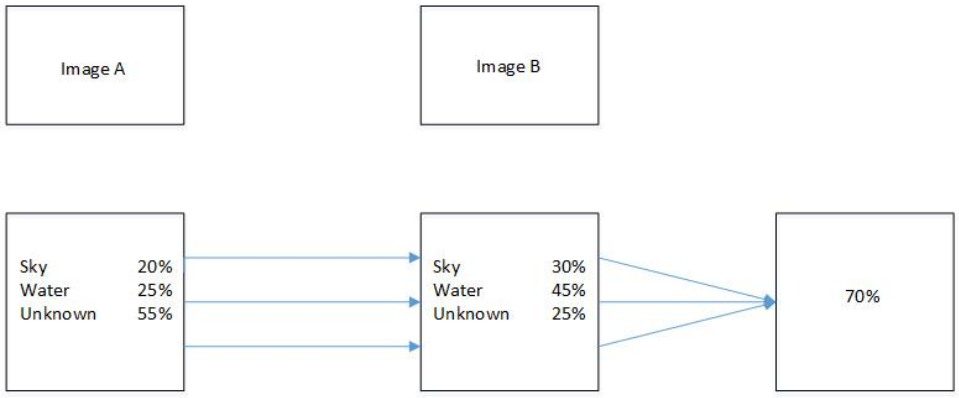

Figure 3.5: Image Matching.

3.1.6 Image Retrieval

Image retrieval is the last part of our system. For this we take the query image and segment the regions to get the description of each region which is explained in fig.3.1. The index of annotated image description is used along with the region description of query image to perform image matching as explained in the previous section. The top n images are retrieved based on their similarity score.[14] The complete retrieval process is explained in fig.3.3.

3.2 Hypothesis

The research in the past decade related to CBIR touched many aspects of the problem and it was seen that deep learning gave the best results. The problem of annotated images was also touched upon but it was not used with the deep learning method. In my thesis I propose to show better results for annotated images using not only the images but also the annotations provided with each image. I will be using convolutional neural network

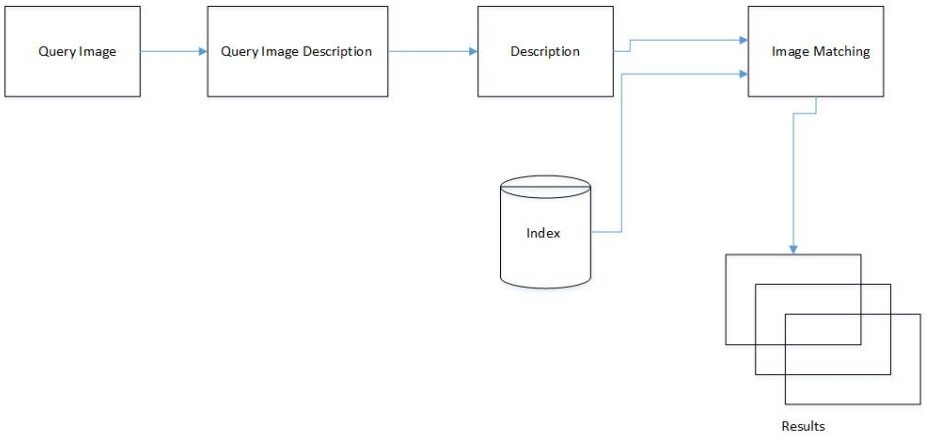

Figure 3.6: Image Retrieval Process.

for training the dataset and save the train model. Once the neural network is trained, a query image will be evaluated against the network and its regions will be classified. It will then be matched against the annotation index which contains the images and the binary values representing each class which is present or absent from that particular image and the most similar images will retrieved. My hypothesis is that deep learning will produce better results for annotated images and which will result in more accurate image retrieval.

3.3 Evaluation

I have worked in the past on CBIR using Bag-of-Words model with the same dataset. The results I get in my study have been evaluated against the results I achieved in my previous study and are discussed in chapter 5. The work has also been compared against the results shown in Ji Wan et al's work [10] where they used deep learning for CBIR with various different datasets but they differ in the sense that the datasets used by them were just plain images without any annotations.

Chapter 4

Architectural Overview

4.1 Convolutional Neural Network

”Convolutional neural network (CNN) is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field” [11]. They are biologically-inspired invariant of Multilayer Perceptrons (MLP) which are designed for the purpose of minimal preprocessing. These models are widely used in image and video recognition. When CNNs are used for image recognition, they look at small portions of the input image called receptive fields with the help of multiple layers of small neuron collections which the model contains [11]. The results we get from this collection are tiled in order for them to overlap such that a better representation of the original image is obtained; every such layer repeats this process. This is the reason they are able if the input image is translated in any way. The outputs of neuron clusters are combined by local or global pooling layers which may be included in convolutional networks. Inspired by biological process, convolutional networks also contain various combinations of fully connected layers and convolutional layers, with point-wise nonlinearity applied at the end of or after each layer [11]. The convolution operation is used on small regions so as to avoid the situation when if all the layers are fully connected billions of parameters will exist. Convolutional networks use shared weights in the convolutional layers i.e. for each pixel in the layer same filter (weights bank) is used which is advantageous because it reduces the required memory size and improves performance. CNNs use relatively less amount of pre-processing as compared to other image classification algorithms,

meaning that the network learns the filters on its own which are traditionally manually-engineered in other algorithms. CNNs have a major advantage over others due to the lack of a dependence on prior-knowledge and the difficult to design hand-engineered features.

4.1.1 Sparse Connectivity

CNNs enforce a local connectivity pattern between neurons of adjacent layers to exploit spatially-local correlation [6]. We have illustrated in fig.4.1 that in **layer m** the inputs of hidden units are from a subset of units in **layer m-1**, units containing spatially adjoining receptive fields.

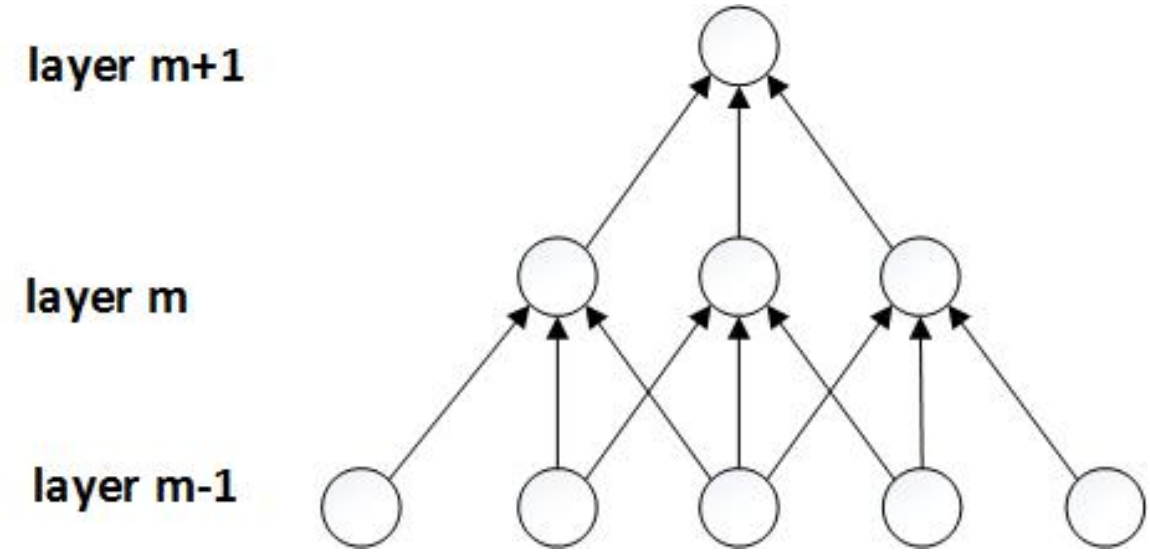

Figure 4.1: Sparse Connectivity [6]

Let us consider **layer m-1** as an input retina. It can be seen in the figure that the layer m have receptive fields of width 3 in the input retina and are thus connected only to 3 adjacent neurons in the retina layer [6]. There is similar connectivity between the units in **layer m+1** and the layer below. It can be said that their with respect to the input receptive field is larger where as with respect to the layer below their receptive field is 3. There is no response in the each unit to variations which are outside their receptive fields with respect to the retina

thus ensuring that the strongest response to a spatially local input pattern is produced by the learnt filter.

4.1.2 Shared Weights

Every filter h_i in CNNs is duplicated across the complete visual field. The duplicated filters consists of the same parameters i.e. weights and bias that form a feature map. We can see in fig.4.2 that same feature map contains 3 hidden units. The weights of same color are shared that are constrained to be identical [6]. We can still use gradient descent to learn such shared parameters by altering the original algorithm by a very small margin. When the gradients of the shared parameters are summed, then it gives the gradient of a shared weight. We can detect the features regardless of their location in the visual field by duplicating the units. The huge reduction of the number of free parameters being learnt can lead to weight sharing increasing the learning efficiency. CNNs achieve better generalization on vision problems due to the constraints on these models.

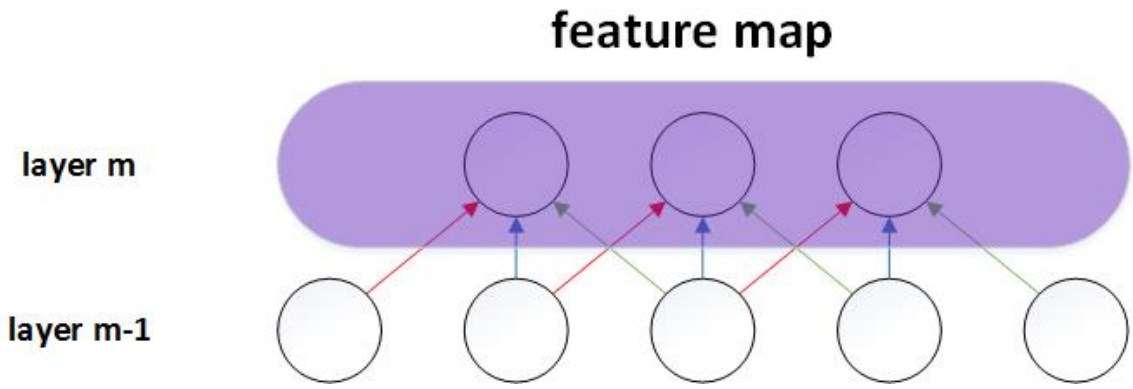

Figure 4.2: Shared Weights [6]

4.1.3 Convolutional Layer

We obtain a feature map by repeatedly applying a function across sub-regions of the entire image, mainly by convolution of the input image with a linear filter, adding a bias term and

then applying a non-linear function [6]. The k-th feature map can be denoted as h^k at a given layer, whose filters we can determine by the bias b^k and weights W^k , then we can obtain the feature map by the given equation:

$$h_k^{ij} = \tanh(W^k * x)_{ij} + b_k) \text{ [6]}$$

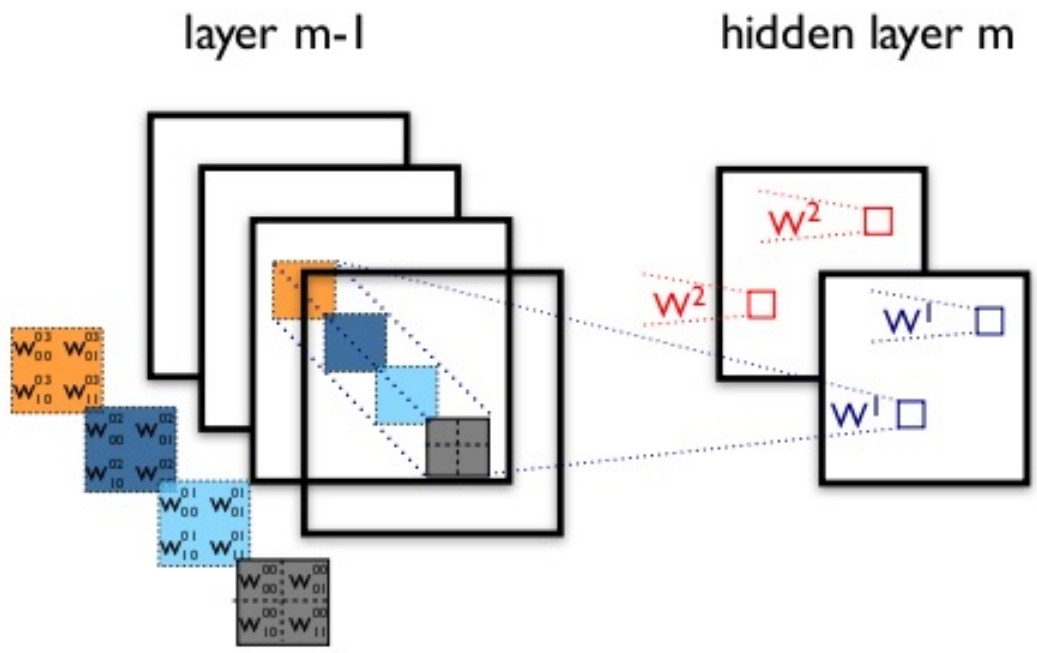

Figure 4.3: Convolutional Layer [6]

The fig.4.4 depicts 2 layers of CNN. There are 4 feature maps in layer m-1 and 2 feature maps in hidden layer m (h^0 and h^1). The pixels of layer (m-1) that lie within their 2x2 receptive field in the layer below (colored squares) are used for the computation of the pixels in the feature maps h^0 and h^1 (blue and red squares). It can be observed that how all 4 input feature maps are spanned by the receptive field. As a result the 3D weight tensors are the weights and of and . The input feature maps is indexed by the leading dimensions, whereas the pixel coordinates is referred by the other two. When we combine it all as shown in fig.4.3, at layer m the weight that connects each pixel of the k-th feature map with the pixel of the l-th layer at layer (m-1) and at coordinates (i,j) is denoted [6] .

4.1.4 Max-Pooling

Max-pooling a form of non-linear down-sampling is an important concept of CNNs. The input image is partitioned into a group of non-overlapping rectangles and a maximum value is given for each such sub-region. We use max-pooling in vision for the following reasons- The computation of upper layers is reduced by the removal of non-maximal values. Suppose a max-pooling layer is cascaded with a convolutional layer. The input image can be translated by a single pixel in 8 directions. 3 out of 8 possible configurations produce exactly the same output at the convolutional layer if max-pooling is done over a 2x2 region. This jumps to 5/8 for max-pooling over a 3x3 region [6]. A form of translation invariance is provided by this. The dimensionality of intermediate representations is reduced by max-pooling because it provides additional robustness to position.

4.1.5 Full-Model: LeNet

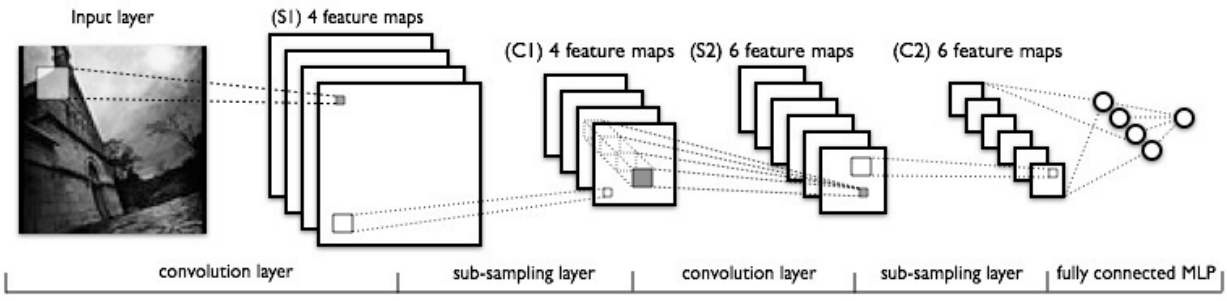

Figure 4.4: Full LeNet Model[6]

The LeNet family of models have sparse, convolutional layers and max-pooling concepts as its core. The exact details of the model shown in fig.4.4 will vary a lot, it shows how a LeNet model will look like. The alternating convolution and max-pooling layers compose the lower-layers of the model. ”The upper-layers however are fully-connected and correspond to a traditional Multi-layer Perceptron which is a combination of hidden layer and logistic regression. The input to the first fully-connected layer is the set of all

features maps at the layer below" [6].

4.2 Dataset

The dataset I chose for this thesis is from the SUN database [12]. The major reason for choosing this dataset was that the images in it were pre-annotated and had annotations as XML files for each image. The SUN database is huge so I had to choose a small subset of it for this study. In this study I am trying to classify images based on 8 classes namely: water, car, mountain, ground, tree, building, snow, sky and unknown which contains all the rest of the classes. I chose only those sets of images which I felt were more relevant to these classes. I collected a database of 3000 images from 41 categories. Each image has its annotations in an XML file. I randomly divided the dataset into 80% training set and 20% testing. There are 1900 training images, 600 testing images and 500 validation images. The training set was further divided into 80% training set and 20% validation set. The major drawback of this dataset is that the images are annotated by humans and the annotations are not perfect thus it may have some effect on the results. I try to handle this problem by getting as many synonyms as I can for each class label. A few examples of the synonyms are lake, lake water, sea water, river water, wave, ripple, river, sea, river water among others which all belong to the class label water. I mapped these synonyms to their respective class labels which are being used. Not all images in every categories were annotated. I filtered out the annotated images from the dataset and used only them for this study. Fig.4.5 shows an example of an image from the dataset and its annotation file where it can be seen how a river is annotated by the user.

A little pre-processing was required on the dataset before it could be trained because of the way the code for CNN training was written. The images were converted to grayscale and resized to 28x28 pixels. I used the annotation files to get a flag for each class present or absent from the image and using the flags I compressed the dataset into a 1D array which contains the image dimensions and binary values for each class where 1 states that class is present and 0 states the class is absent. The compressed data is then trained by the neural

Image

```
<annotation>
    <filename>sun_aajgwojgklrxdpwc.jpg</filename>
    <folder>c/canyon</folder>
    <source>
        <sourceImage>The MIT-CSAIL database of objects and scenes</sourceImage>
        <sourceAnnotation>LabelMe Webtool</sourceAnnotation>
    </source>
    <object>
        <name>river water</name>
        <deleted>0</deleted>
        <verified>0</verified>
        <date>22-Feb-2006 19:53:35</date>
        <polygon>
            <username>anonymous</username>
            <pt>
                <x>139</x>
                <y>129</y>
            </pt>
            <pt>
                <x>135</x>
                <y>134</y>
            </pt>
```

Annotation

Figure 4.5: Example of Image and Annotation

network which returns a train model and the best validation and testing scores.

Constraints

There are a few constraints which I faced while using this dataset. The first constraint I faced was that the dataset was required to be filtered for annotated images. I wrote a script which filtered the images based on the XML annotation files present. Only those images whose annotations were present were copied while the remaining of them were deleted. The next constraint is the size of the images. The code I have used from [6] can only train the model with images of size 28x28 and they have to be grayscale for training. More about the effects of these constraints is explained in Chapter 4.

4.3 Training the Network

I have used the python for coding the convolutional neural network. The code has been taken from [6] and was modified to work for this dataset. The convolutional neural network was built using the Theano library in Python [1]. The model is simplified by this

implementation because it does not implement location-specific gain and bias parameters and also it implements pooling by maximum and not by average. The LeNet5 model uses logistic regression for image classification.

The convolutional neural network was trained by passing the compressed train, test and validation datasets. There is one bias per output feature map. The feature maps are convolved with filters and each feature map is individually downsampled using max-pooling. The compressed dataset is divided into small batch sizes so as to reduce the overhead of computing and copying data for each individual image. The batch size for this model is set to 500. We keep the learning rate which is the factor for the stochastic gradient as 0.1. The maximum number of epochs for running the optimizer is kept as 200 which means the learning for each label goes on for 200 epochs so as to optimize the network.

When the first convolutional pooling layer is constructed, filtering reduces the image size to 24x24, which is further reduced to 12x12 by max-pooling. During the construction of the second convolutional pooling layer the image size is reduced to 8x8 by filtering and max-pooling reduces it further to 4x4. Since the hidden layer is fully-connected it operates on 2D matrices of rasterized images. This generates a matrix of shape (500, 800) with default values. The values of the fully-connected hidden layer are classified using Logistic Regression. The cost which is minimized during training is the negative log likelihood of the model. A Theano function [2] test model is constructed to compute the incorrect calculations that are made by the model. We create two lists, one of all model parameters that have to be fit by the gradient descent and the other of gradients of all model parameter. The updating of the model parameters by Stochastic Gradient Descent(SGD) is done by the Train Model which is a Theano function. Manually creating update rules for each model parameters results in being tedious because of many parameters present in this model. The updates list is thus created by looping over all pairs automatically. We keep a improvement threshold which means that a relative improvement of this much value is considered as significant. Once the training of the convolutional neural network is done, it is the train model that is returned.

4.4 System Design

The fig.4.6 illustrates how the system of retrieval works for this study. The query image is pre-processed and is evaluated with the trained neural network and the regions are classified. It is then matched against the annotation index with images on which the neural network was trained. All the images in the dataset which are similar to the query image are returned to the user based on the number of images required by him. In other words, top N images similar to the query image are retrieved. This section briefly explains the major components of the system design:

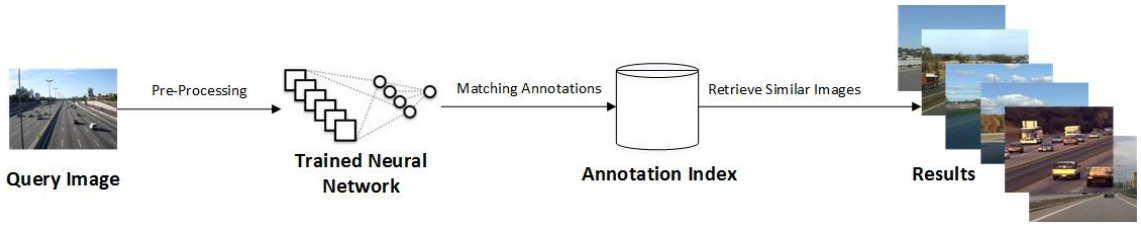

Figure 4.6: System Design

Query Image

The query image is a user input image which he wants to use as a sample to retrieve images from the dataset. The query image can be from any source and need not be from our dataset. An example of query image can be seen in fig 4.7.

Figure 4.7: Query Image

The image has to be pre-processed before it is evaluated by the trained neural network because the dataset on which the network was trained had images pre-processed and works with specific constraints. The image is converted to grayscale and is resized to 28x28 pixels as a part of pre-processing step. Once the query image is a 28x28 pixel grayscale image it can be evaluated with the train model.

Trained Neural Network

I have discussed in section 4.3 how I trained the neural network. The result that is returned after training is a train model which is a Theano function. After the query image is converted to grayscale and is resized it is evaluated with the train model. Based on the training results the regions of the query image are classified according to the class labels. This information is stored and is used for matching against the annotation index.

Annotation Index

I built an annotation index with the help of the annotations provided with every image. The index contains the regions that have been annotated and classified by the users for each image. I check for labels from valid regions to get new labels. These labels are added to the index and for each label only once an image is added to the index. This results in an index which contains the information about all the labels present in each image. The annotation index is also used for generating a mapping for labels based on the existing active labels. I maintain a synonym list which contains the synonyms for all the 8 classes that I am using. These synonyms are names used by users to annotate the images. For example in some cases sky is annotated as cloudy sky, clear sky, sky etc. To handle this situation mapping is done for the labels to be mapped to their synonyms. Once the annotation index is ready the dataset is compressed using the annotation index, images and the mapping.

Chapter 5

Analysis

In this chapter I will analyze the results I got after training and evaluate them against the performance of BoW model on which I had worked before.

5.1 Experiment

The results obtained after training the neural network on the dataset are illustrated in table 5.1. The neural network was trained on the dataset for each label. It can be observed in the table that the validation error rate and testing error rate for each label was quite low. Buildings and trees were two of the worst classified labels in the network with both their error rates in testing and validation being over 40%. In cases like snow the validation and error rate was initially 100% but after some iterations the error rates came down tremendously to 14.6% and 11.4% respectively as shown in table 5.1. This shows that the network learned this class label very well. High error rates can be attributed to the fact that images were downsized to 28x28 pixels which led to loss of information in them.

We can see that in all the classes the best test and validation error rates are achieved on iteration 3, 6 or 9. Training runs for 500 iterations just to validate that there is no change in the error rate after every 100 iterations and once it reaches 500 with a constant rate it stops and returns the best error rates. The training runs for a long time as can be seen in the table. Average time for it to finish training a label is almost 35 minutes. This can be improved by using a GPU and a better processor.

It can be concluded from the results shown in table 5.1 that the network did a good

job in learning the labels Sky, Snow and Ground. Water, Car and Mountain labels showed average classification rate. The network did not train that well for labels Building and Tree. We can say this is due to the fact that loss of information due to downsizing made it difficult for network to classify between trees and buildings because they both overlap each other in many cases. This results in the error rate to go high for these labels.

Table 5.1: Training Results

	Car	Water	Tree	Mountain	Ground	Snow	Sky	Building
Initial Test Score(%)	80.4	28.4	48.4	70.6	22.0	100.0	19.0	43.6
Best Test Score(%)	19.6	28.4	48.4	29.4	22.0	14.6	19.0	42.4
Initial Validation Score(%)	75.2	73.0	52.0	67.2	80.6	100.0	85.8	100.0
Best Validation Score(%)	24.8	27.0	48.0	32.8	19.4	11.4	14.2	44.6
Time (min)	35.4	33.69	33.79	39.68	31.75	32.19	41.9	41.92
Iteration	6	3	3	6	3	9	3	9

5.2 Evaluation

The information shown in table 5.2 is a comparison between the performance of the Bags of Words model and the Convolutional Neural Network model. It is evident from the table that the performance of CNN is much better than that of Bags of Words. Although it can be observed that for some labels like Tree, Building and maybe Sky the error rates for Bags of Words is less or almost equal to that of CNNs, but there is a drastic improvement in the error rates of labels Car, Ground, Snow, Water and Mountain. The evaluation of these results show that using the CNN model for the training of annotated images classifies the images with a lower error rate as compared to Bags of Words model.

Table 5.2: Evaluation Table

Test Error %	Car	Water	Tree	Mountain	Ground	Snow	Sky	Building
Bag of Words	41.89	63.89	37.26	45.81	77.63	42.72	26.3	44.0
Convolutional Neural Network	19.6	28.4	48.4	29.4	22.0	14.6	19.0	42.4

Chapter 6

Conclusions

6.1 Future Work

There is a lot of work that can be done to make this model more efficient and robust for the application of CBIR with annotated images.

Dataset and Classes

In this study I worked with only 3000 images from 41 categories and 8 classes. In future to make the system more generalized and efficient the dataset can be increased and more number of classes such as man, person, plane etc can be added. Just by considering 8 classes I showed that the system is efficient and even if more classes were added to it the performance of the system would only increase.

Image Resizing

The major constraint I had to put is the downsizing of images due to memory issues. The images had to be resized which resulted in loss of data. This problem can be avoided as shown in the work of Krizhevsky et al [5]. In their model the image can be of any square size and can have any number of channels (color channels). Removing this constraint will make the system more robust. One thing that needs to be kept in mind while using original dimensions of the image is that a good GPU will be required for training the convolutional neural network otherwise there will be memory issues which I faced.

Porting Code to C++

The major problem I faced was the speed of training the neural network. The code I have used was written in Python using the Theano library and since Python in itself is known for its slow processing speed the overhead computation time was still high even after taking into consideration parameter reduction to reduce any overhead. To overcome this problem of slow processing the code can be ported to C++ and an already implemented version is available online which is a faster than the Python version.

Bibliography

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[3] Yining Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):800–810, August 2001.

[4] Yahong Han, Fei Wu, Qi Tian, and Yueting Zhuang. Image annotation by input 2013;output structural grouping sparsity. *Image Processing, IEEE Transactions on*, 21(6):3066–3079, June 2012.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] University of Montreal LISA Lab. Deep learning tutorial, 2008.

[7] Ying Liu, Dengsheng Zhang, and Guojun Lu. Region-based image retrieval with high-level semantics using decision tree learning. *Pattern Recogn.*, 41(8):2554–2570, August 2008.

[8] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recogn.*, 40(1):262–282, January 2007.

[9] Chih Fong Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012(1), 2012.

[10] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–166. ACM, 2014.

[11] Wikipedia. Convolutional neural network — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-May-2015].

[12] Jianxiong Xiao, Jianxiong Xiao, James Hays, J. Hays, Krista A. Ehinger, K. A. Ehinger, Aude Oliva, A. Oliva, A. Torralba, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. pages 3485–3492. IEEE, 2010.

[13] Jun Yang, Yu-Gang Jiang, Alexander G. Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the International Workshop on Workshop on Multimedia Information Retrieval*, MIR '07, pages 197–206, New York, NY, USA, 2007. ACM.

[14] Dengsheng Zhang, Md Monirul Islam, Guojun Lu, and Jin Hou. Semantic image retrieval using region based inverted file. In *Proceedings of the 2009 Digital Image Computing: Techniques and Applications*, DICTA '09, pages 242–249, Washington, DC, USA, 2009. IEEE Computer Society.

Appendix A

Retrieval Results

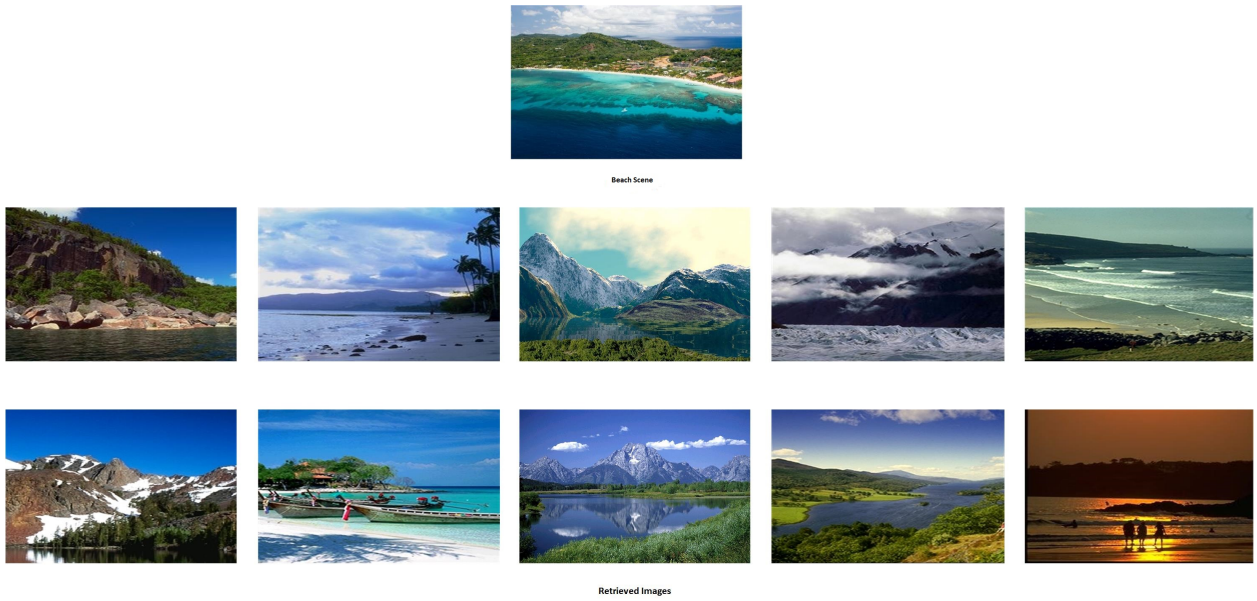

Figure A.1: Beach Scene

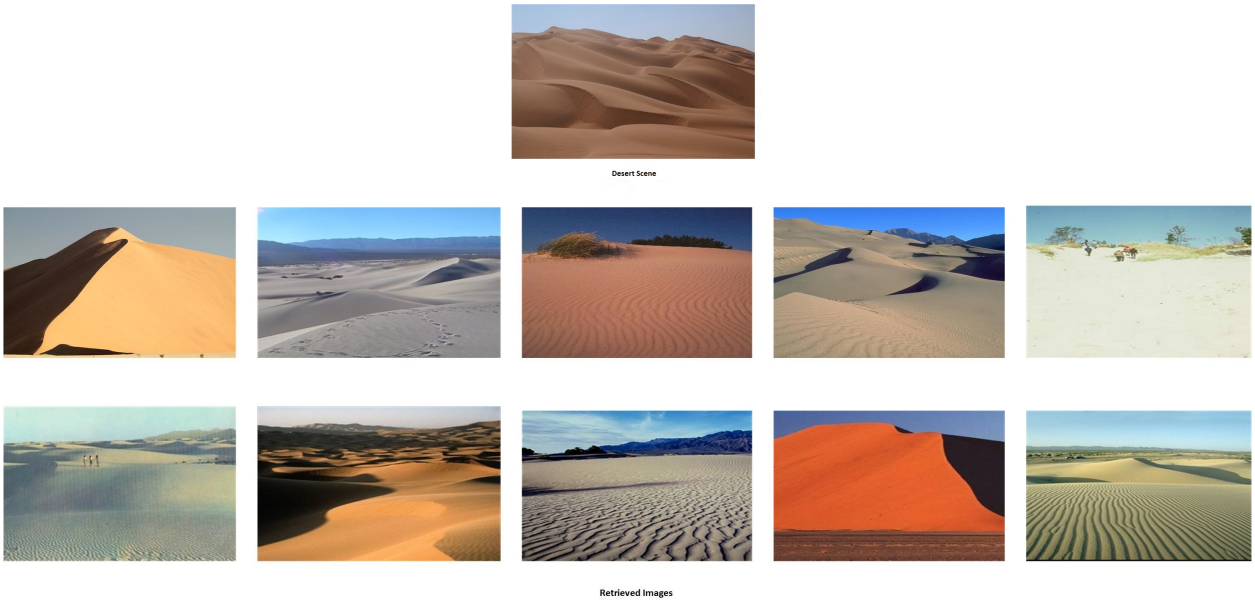

Figure A.2: Desert Scene

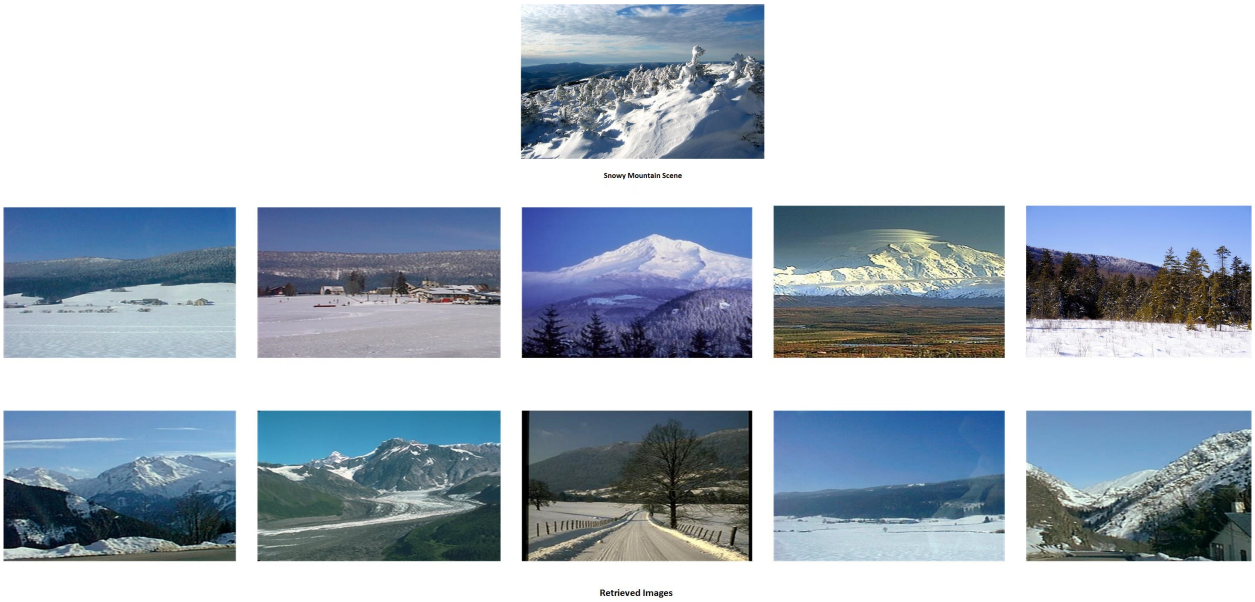

Figure A.3: Snowy Mountain Scene

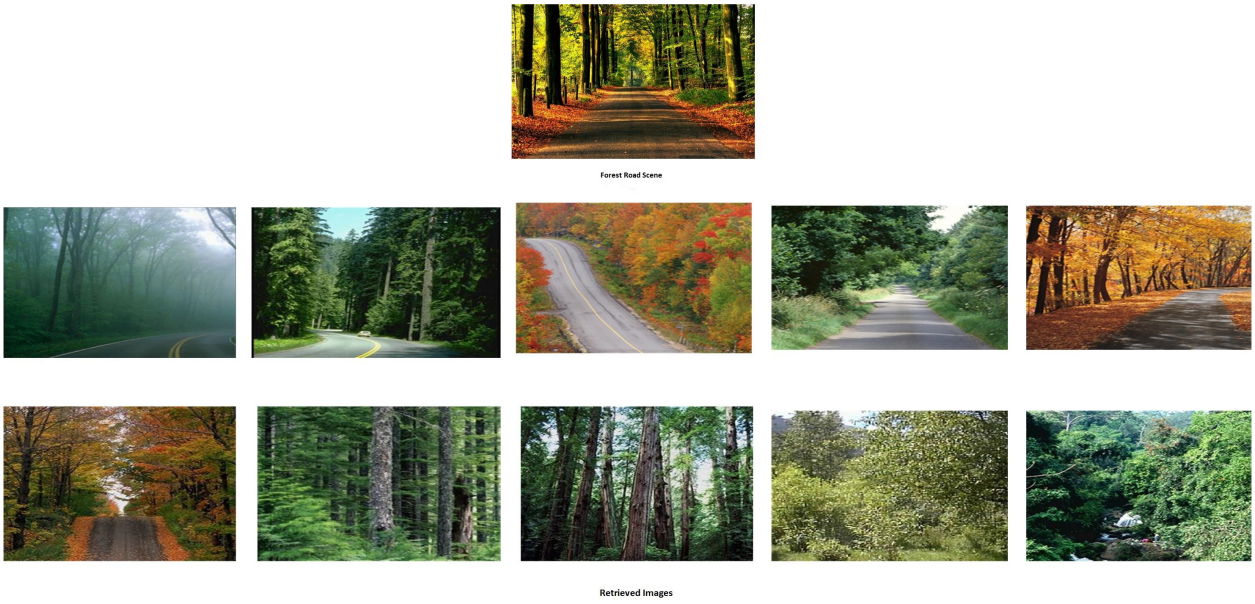

Figure A.4: Forest Road Scene

Figure A.5: Highway Scene