Convolutional Neural Networks for Image Retrieval

A Project Report

submitted by

HARIPRASAD P S

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

under the guidance of
Dr. Kaushik Mitra

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2016

THESIS CERTIFICATE

This is to certify that the thesis entitled **Convolutional Neural Networks for Image Retrieval**, submitted by **Hariprasad P S**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Kaushik Mitra
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 13th May 2016

ACKNOWLEDGEMENTS

I would like to thank Prof. Kaushik Mitra for the guidance, support and encouragement he has provided during the course of this project. I am deeply grateful to all the professors who have helped me and supported me over the last five years at IIT Madras.

Specials thanks to my friends Nikhil Naphade, Abhinav Garlapati, Prasanna S, Vignesh Krishnakumar, Adarsh Tadimari and Kishore Natarajan. I have greatly benefited from my discussions with them. I thank HyperVerge, a startup which I have been a part of, for permission to use GPU computing resources from time to time.

ABSTRACT

Image Retrieval is the procedure of automatically indexing images by the extraction of meaningful features which contain information about the shape, colour and texture, and using these indexed features for retrieval of other similar images. With the advent of deep learning, many computer vision problems have seen a vast improvement in performance, including image retrieval. But with this increase in performance, we have lost the ability to understand why two images are similar, since it is hard to understand the type of information encoded in the features obtained using deep learning. We propose a nearest neighbor search on rich feature representation obtained using Deep Convolutional Neural Networks. We train separate models on different forms of input data to extract features which contain different types of information, and use these features to investigate the type of retrieved images. We further try to combine the various models we have trained to get the optimal performance on a new dataset.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

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. Image Retrieval is the procedure of automatically indexing images by the extraction of meaningful features which contain information about the shape, colour and texture, and using these indexed features for retrieval of other similar images. The representation of the images are 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 an Image Retrieval 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 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 this thesis, I use features obtained from Deep learning models, which are trained using different types of inputs to capture different types of information from the images. These features are tested on a variety of datasets and the type of retrieved images are analyzed to learn more about the models themselves. Further more, I also propose a method to combine the different type of features to obtain the optimal performance on a given dataset.

1.1 Related Work

Traditionally Image Retrieval is performed using features which are hand-crafted to contain specific information, such as colour, shape, etc. These features may be reliable for some datasets, but cannot be scaled as it is hard to generalize features that would work across all types of image datasets.

1.1.1 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. In the past there has been a variety of proposed low-level feature descriptors for image representation, ranging from global features like color features[12], edge features[12], texture features[20], GIST[24], and recent local feature representations, such as the bag-of-words (BoW) models[34] 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[3]. 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[23].

1.1.2 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. It lies in the intersections of several research areas, including neural networks, graphical modeling, optimization, pattern recognition, and signal processing, etc. Deep learning has a long history, and its basic concept is originated from artificial neural network research. The feed-forward neural networks with many hidden layers are indeed a good example of the models with a deep architecture. Back-propagation, popularized in 1980s, has been a well-known algorithm for learning the weights of these networks. For example, LeCun et al. [17] successfully adopted the deep supervised back-propagation convolutional network for digit recognition. Recently, it has become a hot research topic in both computer vision and machine learning, where deep learning techniques achieve state-of-the art performance for various tasks. The deep convolutional neural networks (CNNs) proposed in [16] came out first in the image classification task of ILSVRC-2012. The model was trained on more than one million images, and achieved a winning top-5 test error rate of 15.3% over 1000 classes. Besides image classification, the object detection task can also benefit from the CNN model, as reported in [7]. 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)[10], Boltzmann Machines (BM)[1], Restricted Boltzmann Machines (RBM)[31], Deep Boltzmann Machine (DBM)[30], Deep Neural Networks (DNN)[9], 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.

Several researchers have used deep learning methods for the task of Image Retrieval including [25], [19] and [27]. My work here is related to some of the recent works of [5] and [7] who explore the fully connected layers as possible feature representations for Image Retrieval. However the main aim of this project is not to get the best performance on a dataset but rather to understand and control the type of retrieval that is done using features obtained from deep learning.

CHAPTER 2

Background on Deep Neural Networks

One of the major problems with existing approaches to Image Retrieval is that the features that were used for similarity search were hand-crafted. This meant that the kind of features that worked for a certain kind of dataset would likely be not helpful for a different one. The problem with this approach is that what we see in an image is a lot different from what a machine sees in the image through the features it is fed. In the many to one mapping between the image space and the feature space, reconstruction of the image from these hand-crafted features is unlikely to produce anything close to the original image.

An alternative approach to feature extraction is by learning representations. [28] explains a generic method for learning internal representations from data through error back propagation. This led to the development of neural networks with multiple hidden layers to internally represent data between an input layer and an output layer.

2.1 Deep Neural Networks

As the name suggests, Deep Neural Networks (DNNs) are large neural networks with several hidden layers stacked one on top of the other, giving it a deep architecture. The hidden layers model internal representations that are useful for the task of obtaining the required outputs at the output layer from the raw data sent in at the input layer. Since each hidden layer computes a non-linear transformation of the outputs from the previous layer, a DNN is capable of representing complex relationships between the input and the output.

2.1.1 Training for classification

The most popular way of training a neural network is error back propagation learning. This is outlined in [28]. Referring to Fig. 2.1, if u_j is a neuron, let f_j be the activation function at the neuron and $net_j = \sum_i w_{ij} o_i$ be the net total input. Let p be a particular input-output pair from the training dataset. The local error measure at an output layer neuron is given by:

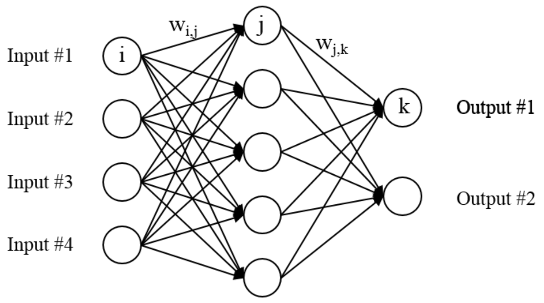

Figure 2.1: A simplified example of a neural network. The sample network has one hidden layer between the input and the output layers.

$$\delta_{pk} = (t_{pk} - o_{pk}) f_j^{'}(net_{pj})$$

where t_{pk} is the target output from the training set and o_{pk} is the predicted output from the model. The local error at a hidden layer neuron is given by:

$$\delta_{pj} = f_j^{'}(net_{pj}) \sum_k \delta_{pk} w_{jk}$$

Finally, the weight updates are given by the generalized delta rule:

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi}$$

By propagating these errors in a recursive manner, the weights of the neural network are updated.

2.1.2 Difficulty in Training Deep Neural Networks

- Problem of diffusion of gradients: The magnitude of the error that is back propagated through the neural network reduces in each layer. Hence the gradients in the initial layers are very small and do not change significantly for a long time at the start of training.
- Local minima: Training a neural network involves non-convex optimization and there are very high chances that the training might get stuck at a poor local minima.
- Insufficient data: Since the model is complex, using insufficient data will result in overfitting.

2.2 Convolutional Neural Networks

The idea of Convolutional Neural Networks (CNNs) was originally refined in [18] and applied to hand-written character recognition. CNNs are characterized by the 2D convolution operation where the input to a neuron is the output of a 2D kernel operation on a local window of neurons from the previous layer. This enables them to deal with patterns of scale, translation and distortion associated with 2D input data. The same kernel is applied throughout a layer, resulting in weights being shared by multiple connections in the network. It has the following major advantages.

- The same kernel is applied throughout the image which helps in achieving translational invariance as the response to a feature present anywhere in the image is similar.
- Since the weights are shared between connections, the number of parameters are reduced when compared to a fully connected network with the same number of connections.
- The convolution operation is parallelizable. There exist efficient GPU implementations for the task thereby reducing the training time significantly.

2.2.1 Types of layers

CNN consists of a combination of convolution and pooling layers, accompanied in the end by a set of fully connected layers. We describe them briefly in this section.

Convolution

The convolution layer performs individual convolution operations. It sends the output of each window of the convolution operation to the neuron in the next layer receptive to that window. The coefficients in the kernel are the parameters to be learnt in a convolutional layer. The convolutional layer is very specific. The neurons in a convolutional layer fire only if specific features are present in their receptive field.

Pooling

Pooling layer computes aggregate operations like max/average over neighbouring windows of outputs and is built for invariance where neurons fire if one of the several patters are found in their receptive field. There are no parameters which need to be learnt in a Pooling layer.

Fully Connected

In a fully connected layer, each neuron is connected to every neuron in the next layer. Most of the parameters which need to be learnt in a convolutional neural network are present in the fully connected layer.

2.2.2 Training

For a large CNN, with several million parameters and more than 500,000 neurons, training times can be extremely high making the use of such a model unfeasible. It becomes imperative to come up with techniques for faster learning. In this section, we look at modifications made to the training process as suggested in [16].

Rectified Linear Units

Neural networks typically use a sigmoidal activation function like $f(x) = tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. These are saturating non-linear activation functions. [8] reports that the use of such activation functions makes training a deep feedforward neural network very slow and ultimately makes the network converge to a poor local minima. [16] proposes the use of the non-saturating non-linearity $f(x) = max(0, x)$. These are referred to as Rectified Linear Units (ReLUs), with the rectification arising out of termination at 0. They show that CNNs trained with such non-saturating activation functions train much faster than the ones using sigmoidal activation functions.

On large datasets, use of ReLUs gives a speedup of several orders in magnitude. This makes training on large image collections feasible.

Local Response Normalization

Normalization of responses locally, when coupled with Rectified Linear Units, lead to better generalization and improved training. Let $a^i_{x,y}$ denote the activity obtained by applying ReLU to the output of kernel i operating at position (x, y) and let $b^i_{x,y}$ denote the normalized output. Then,

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=max(0,\, i-n/2)}^{min(N-1,\, i+n/2)} (a^j_{x,y})^2\right)^\beta}$$

Local Response Normalization can be intuitively understood by looking at the way neurons function in the human brain. It is a form of inhibition where the output through a particular neural connection is suppressed or boosted according to the outputs sent through neighbouring neural connections. This behavior of the neurons in the brain are captured through the function described above.

Secondly, Local Response Normalization also results in normalizing the intensity values. This helps build invariance to lighting conditions into the model. Brightness invariance is an important property required by visual recognition systems as appearance characteristics can change significantly under different lighting conditions.

Parallelization on a GPU

Despite faster learning being made possible by the use of ReLUs, training times can be very high due to the sheer amount of computation involved. However, the convolution operation lends itself to a form that can be parallelized. Powerful GPUs today can execute these operations in parallel giving a huge boost to the training speed. A very efficient GPU implementation of the 2-D convolution operation has been proposed in [16].

The NVIDIA CUDA library for GPU computing provides a way to implement these operations for execution on NVIDIA GPUs. Common NVIDIA GPU cards available today have several thousand cores that can execute threads in parallel. The library made available online through [15] provides a very good framework for implementing CNNs and is used in this work.

[16] also proposes the use of multiple GPU cards and distributing different sets of kernels across the different GPU cards. The neuron from a particular layer in one GPU card is made to ignore the outputs of neurons from the previous layer computed by kernels in a different GPU card. Such parallel streams are maintained until the fully connected layer where the outputs from all the kernels are pooled together. While this results in a loss of invariance, it enables extraction of more number of representative features. In effect, the CNN thus built on 3 GPU cards has the learning capacity of almost 3 times the CNN built on 1 GPU card.

CHAPTER 3

Proposed features for retrieval

In this work, we build several CNN models having the same architecture but contribute different features for the purpose of Image Retrieval. All the models were trained on the ImageNet dataset [29]. The details about the structure and the training of the neural networks are given in this section.

The different models are trained on the same dataset but with different forms of input data. The type of input data determines the type of feature that the model learns.

3.1 Types of Input Data

- RGB: This is the normal 3-channel RGB input which is the default way images are saved.
- CrCb: The input images are transformed in color space from RGB to YCrCb. The Y channel (which represents intensity) is removed and the Cr and the Cb channels are used as input data for the model.
- Intensity: The 3-channel color images are converted to 1-channel grayscale images, which are used as input to train this model.
- Edges: The Edge maps of the images are obtained using [4] and used as input data for the model.

The 4 models mentioned above would be referred to as the RGB model, CrCb model, Intensity model and the Edge model henceforth in this thesis.

Model	Accuracy(top-1)
RGB model	59.10%
Intensity model	58.60%
Crcb model	47.60%
Edge model	44.10%

Table 3.1: Validation Accuracy on the ImageNet dataset

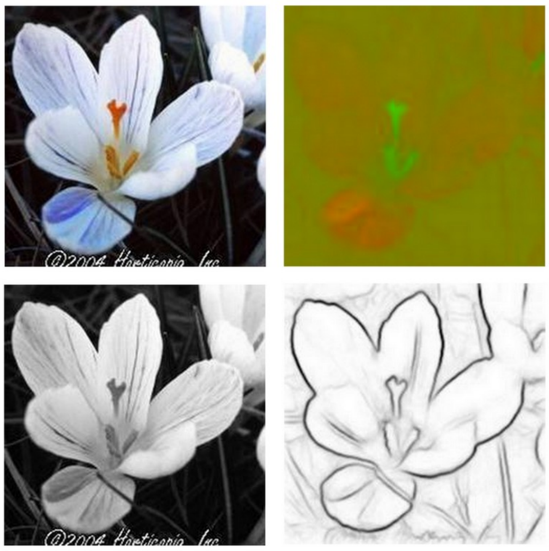

Figure 3.1: A sample image shown in the 4 different input forms for the 4 models. Top left: RGB, Top right: CrCb, Bottom left: Intensity and Bottom Right: Edges

3.2 Architecture

The proposed CNN (shown in Fig. 3.2) consists of 5 convolution layers, interspersed with 3 max pooling layers and finally backed by 3 fully connected layers (including the output layer). This is the alexnet model defined in [16]. The input layer reads the raw data from 227 x 227 dimensional images and a softmax output layer produces class probabilities as outputs. A query image is resized such that the minimum dimension is brought down to 227 pixels and then it is cropped to take the central patch of 227 x 227. This is done so as to preserve aspect ratio in the image.

Every convolution kernel has three dimensions. The first two represent the size of the window over which the kernel operates. The third dimension stands for the number of channels over which the kernel operates. In the RGB model, the first convolutional layer operates on the three channels from the image, namely the RGB channels. In the CrCb model, the first convolutional layer takes the 2-channel input of Cr and Cb. In the Intensity and Edge model, the first convolution layer operates on a single channel, namely intensity and edge map respectively. The output of each kernel becomes a new channel in the next layer. All subsequent kernel operations happen over channels from the previous layer.

The configuration of the convolution layers is as follows: first convolution layer has 96 kernels of size 11x11x(number of channels), second has 256 of size 5x5x96, third has 384 of size 3x3x256, fourth has 384 of size 3x3x384 and fifth has 256 of size 3x3x384. All three max pooling layers use a kernel of size 3x3 with a stride of 2.

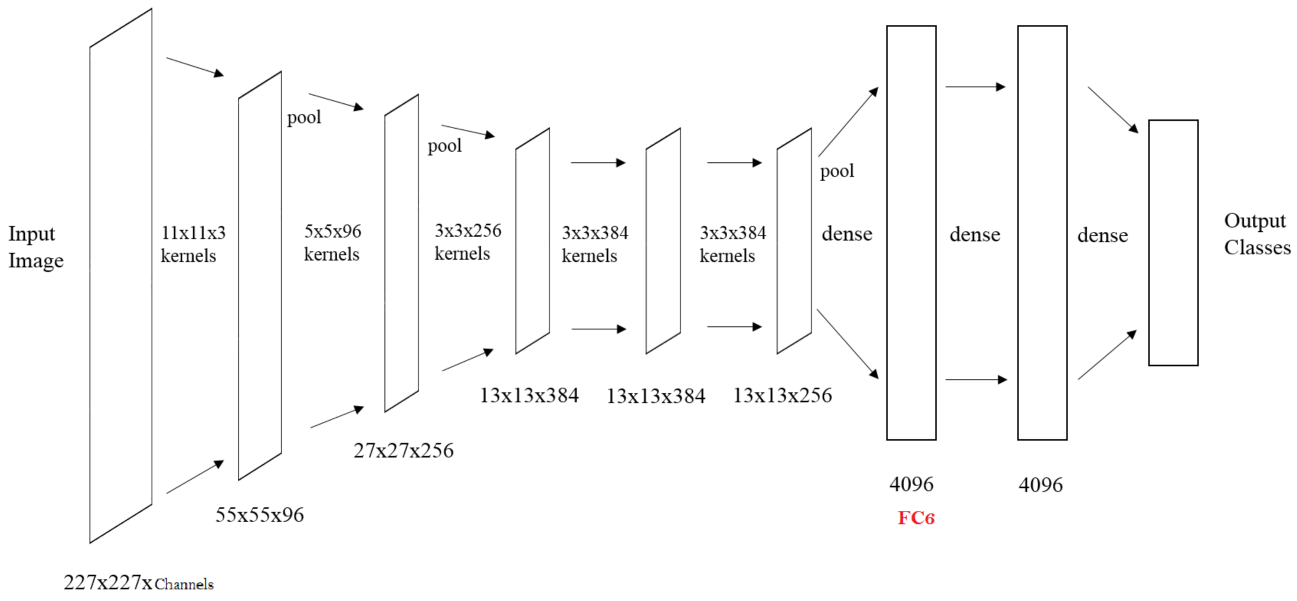

Figure 3.2: An architectural diagram showing the configuration of alexnet. The numbers at the bottom indicate the layer sizes. Max pooling layers are used after the second, third and fifth convolution layers.

We have 2 fully connected layers, each having 4096 neurons. The final softmax output layer is also fully connected with the penultimate layer. The outputs from the softmax layer can be seen as the probability of an image belonging to each of the output classes.

3.3 Learning Details

Our input images for training were random patches of 227 x 227 that we extracted from resized images of size 256x256. We used a batch of 256 images in each iteration of stochastic gradient descent. The weight update rule is given by:

$$v_{i+1} = 0.9.v_i - 0.0005.\epsilon.w_i - \epsilon.\left\langle \frac{\partial L}{\partial w}|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} = w_i + v_{i+1}$$

Here i is the iteration number, w is the weight vector, D_i is the training data sampled for training in iteration i, v is the weight update, ϵ is the learning rate and L is the Cross Entropy Loss function applied on the softmax outputs from the neural network given by:

$$L = -\sum_{j} t_j log(s_j)$$

Here j runs over the output classes. If a particular sample belongs to class c, then $t_c = 1$ and $t_j = 0$ for all $j \neq c$. s_j is the softmax output. In the weight update rule, the first term represents the momentum used to prevent the model from getting stuck at local minima. The second term is a regularization with a weight decay of 0.0005. The third term accounts for the error gradient.

Initialization of weights was done using a gaussian distribution with zero mean and a standard deviation of 0.01. The bias terms for the layers were filled with constants, either 0.5 or 0. Biases were set to 0.5 in the fully connected layer and in some convolution layers in order to accelerate learning as this leads to positive inputs at the ReLUs.

We decayed our learning rate following a step function. The learning rate was initialized to 0.01 and was multiplied by a factor of 0.1 once every 70,000 iterations, unless the error rate increases over a period of time. We test the model once every 2000 iterations to keep a check on the validation error.

3.3.1 Training Dataset

The ImageNet dataset [29](ILSVRC2014) contains around 1.2 million images from 1000 classes. The exact same train and test splits were used for training the 4 different models. A complete list of classes can be found at `http://www.image-net.org/challenges/LSVRC/2014/`. For all images, the minimum dimension was brought to 256 pixels and the central path of 256 x 256 pixels was used. For each model, the corresponding mean image computed from the training dataset was subtracted from each image before use for training or testing.

3.4 Reducing Overfitting

When a model has 60 million parameters, it becomes essential to take measures to avoid overfitting. In this section, we discuss the methods we adopted to handle overfitting.

3.4.1 Data Augmentation

One simple way to combat overfitting is by using large volumes of training data. However, the amount of labelled images is limited. A way to handle this is by replicating data through the introduction of small variations and augmenting this to our training dataset. We extract multiple patches of 227 x 227 from the training image of 256 x 256. This helps us achieve invariance to translation and introduces more variations in the training data.

3.4.2 Dropout

It is a well noted observation that multiple weak models used together in making a decision is better than using a single model that tries to be very strong. Training a large number of CNNs and using them in parallel is however not feasible due to the amount of memory they consume. The several million parameters learnt by a CNN occupy a significant chunk of memory in the RAM. A method proposed in [11] to overcome this limitation is to drop the inputs from certain randomly chosen neurons during training. Since the neurons are selected in a random way, different sets of neurons send outputs through in each batch of input images.

The consequence of this is that each neuron is forced to learn from different combinations of neurons from previous layers. This builds greater generalization. In our implementation, we set outputs of neurons in the first 2 fully connected layers to 0 with a probability of 0.5. When a query image is processed, the outputs of all neurons are taken in but they are multiplied with a factor of 0.5 to weigh their contributions in the training process.

3.4.3 Early Termination

Another general way to prevent overfitting is by terminating the training process early. Along with this, we also drop the learning rate by a factor of 10 whenever the training error increases or is constant for a sustained number of iterations.

3.5 Generic Image Features

A very important observation is that while the CNN might be built for a particular classification task, the model itself is designed in a way to learn representations of images that are best suited for that task. This means that the CNN learns representations that are similar for similar images and dissimilar for dissimilar images. This happens because similar images are to be categorized under the same label, while images that are quite different are likely to be in different classes.

We found that the feature vector produced by the first fully connected layer (fc6 shown in 3.2) best represented this similarity and could be used as a characteristic representation of the image.

3.5.1 Feature Extraction

Once we have the trained CNN models, we need to extract features to perform Nearest Neighbor Search. To extract the features for a given image, we convert it into the 4 different input forms and forward-propagate them through the corresponding models. We extract the output of the first fully connected layer (fc6), which has been found to be a good representation for the image, among other layers. For each model, this fc6 layer gives us an array of 4096 float values per image.

3.5.2 Binarizing Features

An important observation made from our experiments with the CNN was that the raw outputs from the fully connected layers were relatively either too high or too low. This seems to indicate that the neurons in the fully connected layers try to convey some kind of confidence, as is suggested by their spiking behaviour. The possible inference to be made here is that the exact value of the outputs from the fully connected layers do not matter too much beyond the information as to which bucket (high or low) they belong to.

This observation implies that by binarizing the output values according to the bucket they belong to (high or low) hardly causes any loss of information. This sort of binarizing has also been proposed by Jitendra Malik in [2]. This binary feature vector constructed by putting together the binarized output values of the neurons is a good representation of the image. We also note here that nearest neighbour search over a Hamming Space is much more computationally efficient than in the Euclidean Space.

For every image, we now have four 4096-binary feature vectors obtained from four different CNN models. To measure the similarity between 2 images, we compute the hamming distance between the corresponding features. Lower the hamming distance, more similar are the 2 images.

3.6 Implementation Platform

We use the library provided in [15] to implement our CNN. The training was run on a server with an NVIDIA Tesla M2070 GPU card with 2880 CUDA cores, an Intel Xeon X5675 CPU and 5375MB of Video RAM. Training was done for 90 epochs and on the given configuration this took about a week.

CHAPTER 4

Retrieval experiments on various datasets

The models trained in the previous section were used to extract features from images and perform Image Retrieval on several datasets.

4.1 UKB dataset[22]

The University of Kentucky Recognition Benchmark(UKB) consists images of 2550 objects, each of which is represented by 4 images. The most common evaluation metric for this dataset counts the average number of relevant images (including the query itself) that are ranked in the first four positions when searching the 10200 images. This corresponds to 4 times the recall@4 measure, i.e., the best performance is 4.

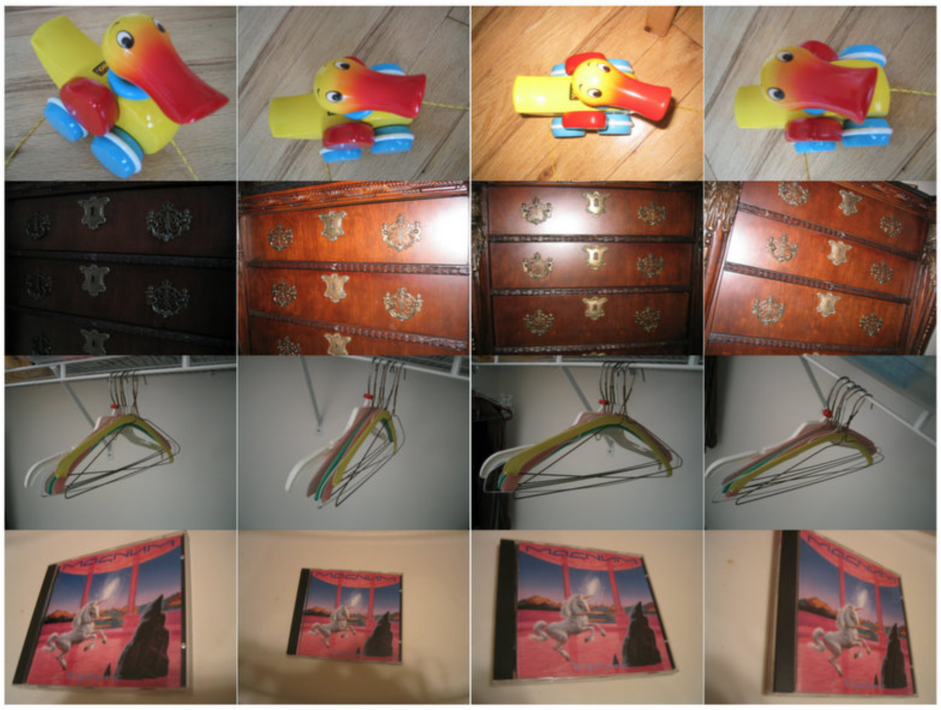

Figure 4.1: Sample Images from the UKB dataset.

Descriptor	Performance (4 x recall@4)
GIST	1.62
BOW	2.95
Fisher	3.33
VLAD	3.35
Edge Model	1.90
CrCb Model	3.14
Intensity Model	3.11
RGB Model	3.38

Table 4.1: Performance on the UKB dataset in comparison with some of the conventional methods of Image Retrieval as reported in [14]

4.2 Holidays dataset[13]

The INRIA Holidays dataset is a collection of 1491 holiday images, 500 of them being used as queries. The dataset includes a very large variety of scene types (natural, man-made, water and fire effects, etc) and images are in high resolution. It contains 500 image groups, each of which represents a distinct scene or object. The first image of each group is the query image and the correct retrieval results are the other images of the group. Since we are using a global feature descriptor, we are only comparing with other global descriptors. The accuracy is measured by the mean Average Precision (mAP), as defined in [26].

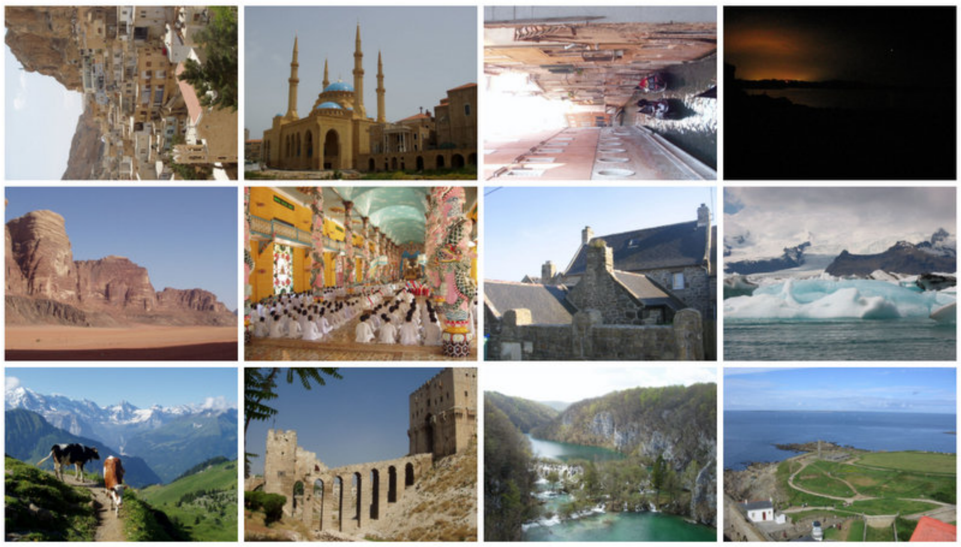

Figure 4.2: Sample Images from the Holidays dataset.

Descriptor	Performance (mAP)
GIST	36.5
BOW	45.2
Fisher	59.5
VLAD	55.7
Edge Model	31.6
CrCb Model	64.7
Intensity Model	65.1
RGB Model	71.0

Table 4.2: Performance on the Holidays dataset in comparison with some of the conventional methods of Image Retrieval as reported in [14]

Descriptor	Performance (precision@1)
Edge Model	41.70
CrCb Model	80.87
Intensity Model	71.90
RGB Model	86.55

Table 4.3: Performance on the Oxford Flowers dataset

4.3 Oxford Flowers dataset[21]

The Oxford Flowers Dataset consists of 17 flower categories with 80 images for each class. The images have large scale, pose and light variations and there are also classes with large variations of images within the class and close similarity to other classes. 50% of the images are used as query images to retrieve from the other 50%. The measure of performance used is precision@1, ie., for every query image we look at the first retrieved image and assign it a precision of 1 if it belongs to the same class, else it is 0. This is averaged over all the query images.

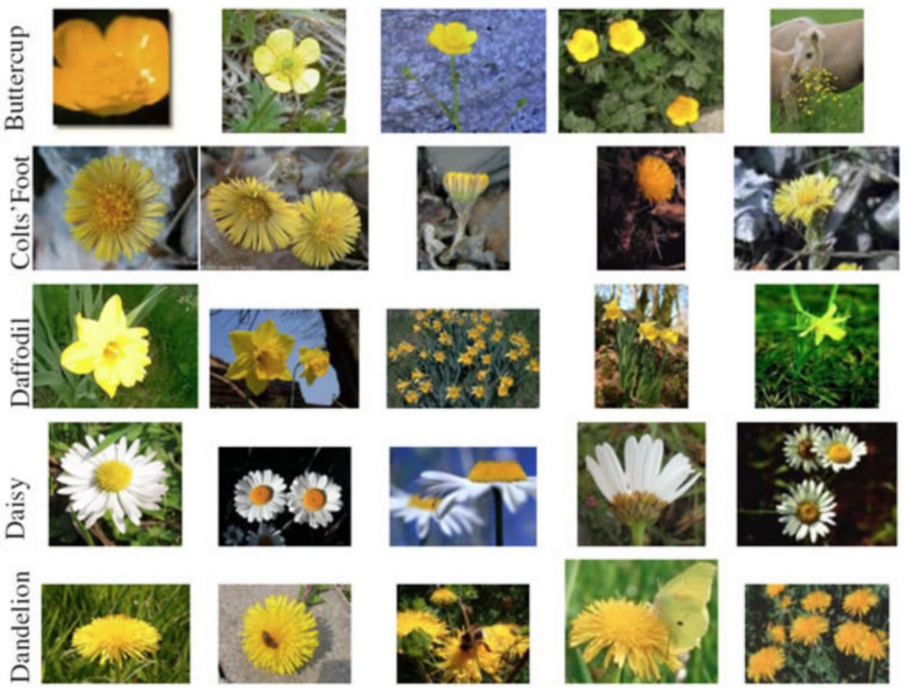

Figure 4.3: Sample Images from the Oxford Flowers dataset.

Descriptor	Performance (precision@1)
Edge Model	54.41
CrCb Model	63.43
Intensity Model	78.19
RGB Model	80.51

Table 4.4: Performance on the Caltech 101 dataset

4.4 Caltech101 dataset[6]

The Caltech 101 dataset contains pictures of objects belonging to 101 classes. There are 40 to 800 images per category. Most categories have about 50 images. 50% of the images are used as query images to retrieve from the other 50%. The measure of performance used here is also precision@1.

Some observations:

- In the Caltech101 dataset, the Intensity model features perform much better than the CrCb model features. However we see the exact opposite observation in the Oxford Flowers dataset. This tells us that the colour information captured by the CrCb model proves to be more useful for a dataset like Oxford Flowers dataset, where there is high inter-class colour variance.
- The RGB model always performs better than the other 3 models, as the image information available to that model is higher than other models.
- In most datasets, the CrCb model and the Intensity model have similar performance although the

information encoded in the features is very different.

- In the Caltech101 dataset, the performance of the Intensity model and the RGB model are almost at par.

CHAPTER 5

Control over type of retrieval

Having control over the type of features used to perform Image Retrieval is very important in several cases. One important real life application of Image Retrieval is in Clothing. When a shopper finds a shirt he is interested in, he would also like to see other shirts which are similar to the one he has selected, to finetune his choice. These other shirts could be similar in colour, pattern, design, etc. Here we use the models we had trained to come up with different suggestions of a query image. The clothing images used were crawled from the internet.

The advantage that conventional retrieval methods have over Deep Learning is that, we know the relation between the query image and the retrieved images. We know exactly what the retrieved images have in common with the query image (colour, shape, etc.). If we were to use only the RGB model, although its performance is much better than the conventional methods, we do not know why the retrieved images are similar to the query image. By having multiple models, each trained on a different type of input data, we can now know why the retrieved images are similar to the query image.

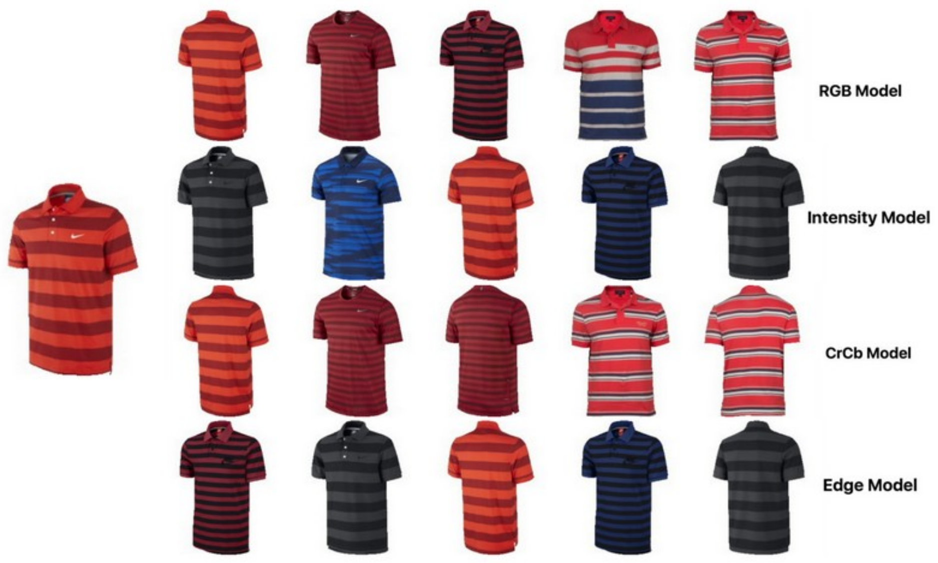

Figure 5.1: Sample 1: Clothing Image Retrieval

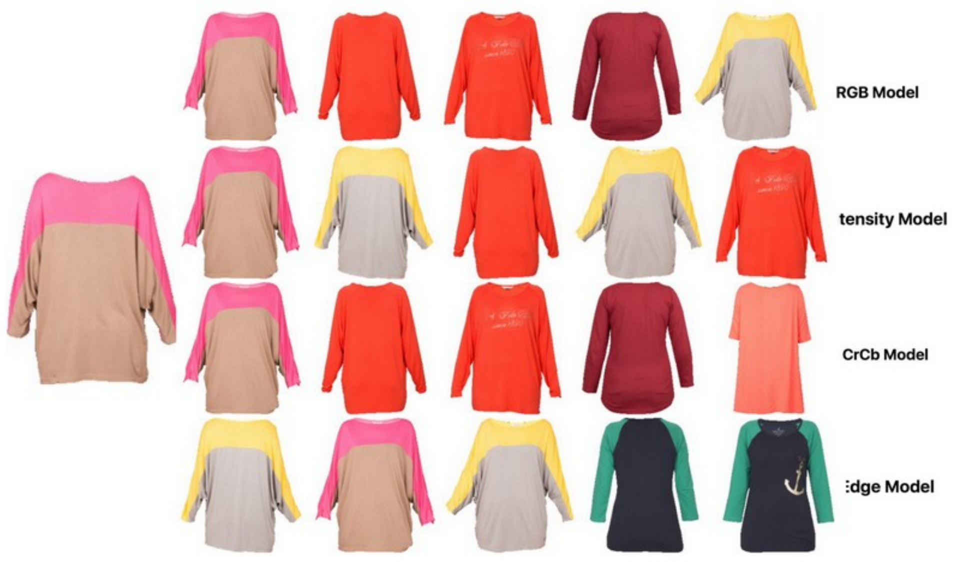

Figure 5.2: Sample 2: Clothing Image Retrieval

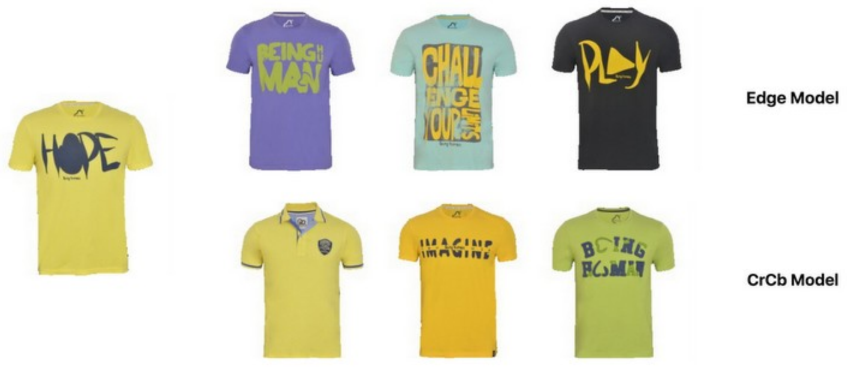

Figure 5.3: Sample 3: Clothing Image Retrieval

From Figure 5.1, we can see that the CrCb model has retrieved images which are similar in colour to the query image, while the Edge model has retrieved images which have the same pattern as the query. We can see a similar observation with others as well.

From the sample figures shown above, it can been seen that the CrCb model gives importance to the colours present in the clothing while the Edge model gives importance to the shape and texture present.

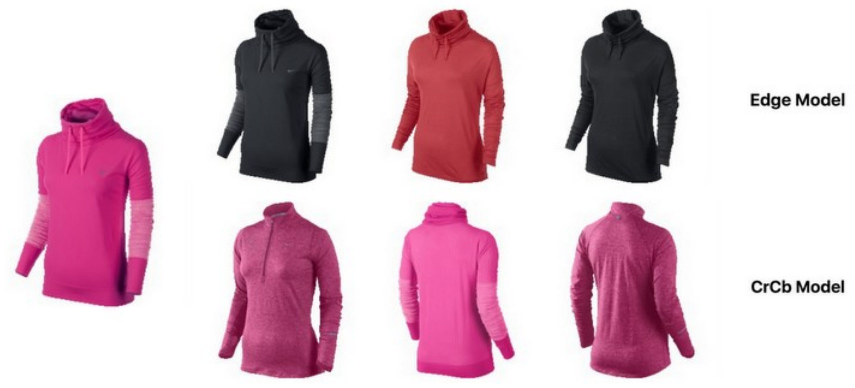

Figure 5.4: Sample 4: Clothing Image Retrieval

The Intensity model uses all information present in an image except colour. The RGB model uses all the information available in an image. Hence it is hard to understand what the retrieved images have in common with the query image.

CHAPTER 6

Optimal combination of the various models

In this section, we try to combine the features from the 4 models in the best way possible, to get the maximum performance on a given dataset. For a given model, we find the similarity between two images by computing the 4096 binary features for each image using that model, and then computing the hamming distance between the two feature vectors. For any 2 images, there would be 4 such hamming distances computed using the features from the 4 different models. Let Hij_{rgb} , Hij_{crcb}, $Hij_{intensity}$ and Hij_{edges} be the hamming distances between 2 images i and j using the corresponding feature vectors. We define a new distance measure for the 2 images, which is a weighted sum of the 4 hamming distances.

$$Hij_{weighted} = w_{rgb} * Hij_{rgb} + w_{crcb} * Hij_{crcb} + w_{intensity} * Hij_{intensity} + w_{edges} * Hij_{edges}$$

$$w_{rgb} + w_{crcb} + w_{intensity} + w_{edges} = 1$$

where w_{rgb}, w_{crcb}, $w_{intensity}$ and w_{edges} are the weights given to each feature vector. For a given dataset, we try to find the optimal weights which would give the best performance in retrieval. The images which are used as query are not used to determine these weights.

6.1 Determining the weights

To determine the optimal weights for a dataset (lets call it dataset A), we sample several images from the non-query part of dataset A and compute the corresponding features for the different models. We then use Genetic Algorithm to find the set of weights which would maximize the performance (precision@1). We start with an initial population size of 10 and run it for 100 generations.

6.1.1 Genetic Algorithm

- randomly initialize population(t)

- determine fitness of population(t)
  - select parents from population(t)
  - perform crossover on parents creating population(t+1)
  - perform mutation of population(t+1)
  - determine fitness of population(t+1)
- until best individual is good enough or generation limit is reached

The entire process of finding the optimal weights using Genetic Algorithm takes anywhere between 10 to 30 minutes.

6.2 Transfer Weights from Another dataset

It is not always viable to sample images from dataset A and find the optimal weights using Genetic Algorithm. To reduce this time, we propose transferring weights from another dataset, which is similar to the given dataset A, whose optimal weights are precomputed. We use the ImageNet 1000 classes dataset[29] for this purpose. We sampled 100 images from the validation dataset of each class from the ImageNet dataset and found the optimal weights for Image Retrieval for each class. At the end of this process, we have seperate optimal weights for each of the 1000 classes from the ImageNet dataset.

Now to find the weights for dataset A, we sample some images from the non-query part of the dataset and classify them using the RGB model which was trained earlier. We take the majority class among these sampled images and transfer the weights learnt for that class to dataset A. This would give a near-optimal combination of different feature vectors for Image Retrieval.

Descriptor	Performance (precision@1)
Edge Model	41.70
CrCb Model	80.87
Intensity Model	71.90
RGB Model	86.55
Optimal combination	88.34
Transfered Weights from ImageNet class (daisy)	88.19

Table 6.1: Performance on the Oxford Flowers dataset with combination

The Optimal weights and performance are given in tables 6.1 and 6.2. The ImageNet class which is closest to the Oxford Flowers dataset is found to be the class *daisy*.

	w_{rgb}	w_{crcb}	$w_{intensity}$	w_{edges}
Optimal Combination	0.3674	0.2496	0.1711	0.2119
Transfered Weights from ImageNet class (daisy)	0.4098	0.0602	0.2382	0.2918

Table 6.2: Optimal Weights for the Oxford Flowers dataset

CHAPTER 7

Conclusion

Our results show a promising approach for large scale Image Retrieval. We have shown how we can control the type of retrieval by changing the type of input data, keeping the architecture the same. Furthermore, we have also shown how to combine multiple features (specialized for different needs) to get the optimal retrieval performance on a given dataset. Some of the possible work that can be done in future with regard to this project are given below.

7.1 Future Work

7.1.1 Using more powerful models

The experiments that were shown in the previous chapters can be repeated with more powerful sophisticated models such as GoogleNet [33] and VGG [32]. I have used the VGG19 model (released publicly by [32]) to extract the first fully connected layer features and check the retrieval performance on the UKB dataset and the Holidays dataset.

Descriptor	Performance (4 x recall@4)
RGB Model (alexnet)	3.38
VGG19 model	3.52
Best accuracy reported [14]	3.47

Table 7.1: Performance on the UKB dataset extended

Descriptor	Performance (mAP)
RGB Model (alexnet)	71.0
VGG19 model	78.12
Best accuracy reported [25]	84.7

Table 7.2: Performance on the Holidays dataset extended

7.1.2 Video Retrieval

We can extend this Image Retrieval system to Video Retrieval as well. One simple method would be to sample frames from the videos and make each frame of the query video vote for the closest frame from another video. I have used the RGB model on the UCF-101 dataset to generate the video retrievals shown below.

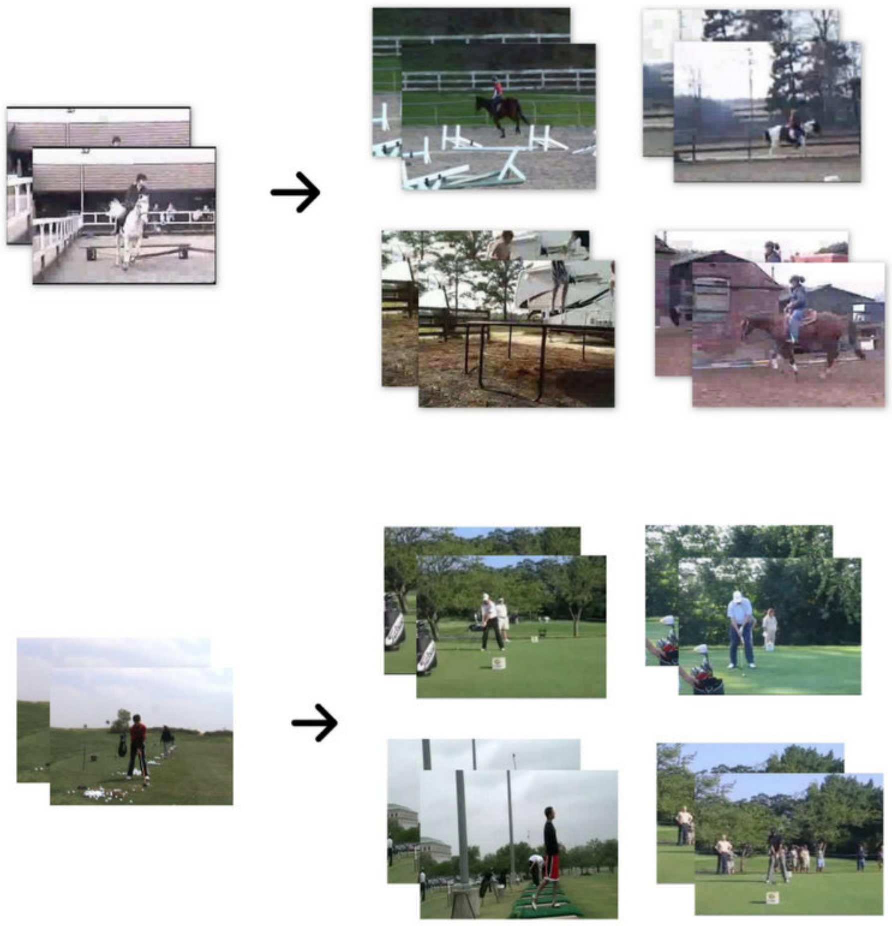

Figure 7.1: Sample Video Retrieval from UCF-101 dataset

REFERENCES

[1] **Ackley, D. H.**, **G. E. Hinton**, and **T. J. Sejnowski** (1985). A learning algorithm for boltzmann machines. *Cognitive science*, **9**(1), 147–169.

[2] **Agrawal, P.**, **R. Girshick**, and **J. Malik**, Analyzing the performance of multilayer neural networks for object recognition. *In Computer Vision–ECCV 2014*. Springer International Publishing, 2014, 329–344.

[3] **Chechik, G.**, **V. Sharma**, **U. Shalit**, and **S. Bengio** (2010). Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, **11**, 1109–1135.

[4] **Dollár, P.** and **C. L. Zitnick**, Structured forests for fast edge detection. *In ICCV*. 2013.

[5] **Donahue, J.**, **Y. Jia**, **O. Vinyals**, **J. Hoffman**, **N. Zhang**, **E. Tzeng**, and **T. Darrell** (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*.

[6] **Fei-Fei, L.**, **R. Fergus**, and **P. Perona**, Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *In Int. Conf. Comput. Vis. Patt. Recogn. Workshop on Generative-Model Based Vision*. 2004.

[7] **Girshick, R.**, **J. Donahue**, **T. Darrell**, and **J. Malik**, Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[8] **Glorot, X.** and **Y. Bengio**, Understanding the difficulty of training deep feedforward neural networks. *In In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*. 2010.

[9] **Hinton, G.**, **L. Deng**, **D. Yu**, **G. E. Dahl**, **A.-r. Mohamed**, **N. Jaitly**, **A. Senior**, **V. Vanhoucke**, **P. Nguyen**, **T. N. Sainath**, *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, **29**(6), 82–97.

[10] **Hinton, G. E.**, **S. Osindero**, and **Y.-W. Teh** (2006). A fast learning algorithm for deep belief nets. *Neural computation*, **18**(7), 1527–1554.

[11] **Hinton, G. E.**, **N. Srivastava**, **A. Krizhevsky**, **I. Sutskever**, and **R. R. Salakhutdinov** (). Improving neural networks by preventing co-adaptation of.

[12] **Jain, A. K.** and **A. Vailaya** (1996). Image retrieval using color and shape. *Pattern recognition*, **29**(8), 1233–1244.

[13] **Jégou, H.**, **M. Douze**, and **C. Schmid**, Hamming embedding and weak geometric consistency for large scale image search. *In* **A. Z. David Forsyth, Philip Torr** (ed.), *European Conference on Computer Vision*, volume I of *LNCS*. Springer, 2008. URL `http://lear.inrialpes.fr/pubs/2008/JDS08`.

[14] **Jégou, H.**, **F. Perronnin**, **M. Douze**, **J. Sanchez**, **P. Perez**, and **C. Schmid** (2012). Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(9), 1704–1716.

[15] **Jia, Y.**, **E. Shelhamer**, **J. Donahue**, **S. Karayev**, **J. Long**, **R. Girshick**, **S. Guadarrama**, and **T. Darrell** (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[16] **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton**, Imagenet classification with deep convolutional neural networks. *In Advances in Neural Information Processing Systems*.

[17] **LeCun, Y.**, **L. Bottou**, **Y. Bengio**, and **P. Haffner** (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

[18] **Lecun, Y.**, **L. Bottou**, **Y. Bengio**, and **P. Haffner**, Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*. 1998.

[19] **Lin, K.**, **H.-F. Yang**, **J.-H. Hsiao**, and **C.-S. Chen**, Deep learning of binary hash codes for fast image retrieval. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015.

[20] **Manjunath, B. S.** and **W.-Y. Ma** (1996). Texture features for browsing and retrieval of image data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **18**(8), 837–842.

[21] **Nilsback, M.-E.** and **A. Zisserman**, A visual vocabulary for flower classification. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2. 2006.

[22] **Nistér, D.** and **H. Stewénius**, Scalable recognition with a vocabulary tree. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2. 2006. **oral presentation**.

[23] **Norouzi, M.**, **D. J. Fleet**, and **R. R. Salakhutdinov**, Hamming distance metric learning. *In Advances in neural information processing systems*. 2012.

[24] **Oliva, A.** and **A. Torralba** (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, **42**(3), 145–175.

[25] **Perronnin, F.** and **D. Larlus**, Fisher vectors meet neural networks: A hybrid classification architecture. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[26] **Philbin, J.**, **O. Chum**, **M. Isard**, **J. Sivic**, and **A. Zisserman**, Object retrieval with large vocabularies and fast spatial matching. *In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007.

[27] **Razavian, A.**, **H. Azizpour**, **J. Sullivan**, and **S. Carlsson**, Cnn features off-the-shelf: an astounding baseline for recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014.

[28] **Rumelhart, D. E.**, **G. E. Hinton**, and **R. J. Williams**, Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X, 318–362. URL http://dl.acm.org/citation.cfm?id=104279.104293.

[29] **Russakovsky, O.**, **J. Deng**, **H. Su**, **J. Krause**, **S. Satheesh**, **S. Ma**, **Z. Huang**, **A. Karpathy**, **A. Khosla**, **M. Bernstein**, **A. C. Berg**, and **L. Fei-Fei** (2014). ImageNet Large Scale Visual Recognition Challenge.

[30] **Salakhutdinov, R.** and **G. E. Hinton**, Deep boltzmann machines. *In International conference on artificial intelligence and statistics*. 2009.

[31] **Salakhutdinov, R.**, **A. Mnih**, and **G. Hinton**, Restricted boltzmann machines for collaborative filtering. *In Proceedings of the 24th international conference on Machine learning*. ACM, 2007.

[32] **Simonyan, K.** and **A. Zisserman** (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, **abs/1409.1556**.

[33] **Szegedy, C.**, **W. Liu**, **Y. Jia**, **P. Sermanet**, **S. Reed**, **D. Anguelov**, **D. Erhan**, **V. Vanhoucke**, and **A. Rabinovich**, Going deeper with convolutions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[34] **Wu, L.**, **S. C. Hoi**, and **N. Yu** (2010). Semantics-preserving bag-of-words models and applications. *Image Processing, IEEE Transactions on*, **19**(7), 1908–1920.