Thesis

RIDMBC for Object Recognition using Convolutional Neural Networks

Submitted by

Nikhil Agnihotri

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2016

Master's Committee:

Advisor: Bruce A. Draper
Co-Advisor: Ross Beveridge

Anthony Maciejewski

Abstract

RIDMBC for Object Recognition using Convolutional Neural Networks

Two trending techniques that are making advances in computer vision research are Convolutional Neural Networks and Visual Hashing. The goal of this paper is to analyze how these two interact in the broad domain of objects. Deep neural nets have proved to broadly represent image features, and binary codes have proved to be a powerful way to represent the intrinsic nature of image content in a compact way. Our research explores what kind of information is contained in feature vectors obtained from deep neural nets and what information can be binarized, in the context of object recognition. We also try to optimize the length of binary codes and select subsets of bit vectors to represent images so as to obtain the best classification results, while trying to bring down computational cost.

Acknowledgements

Thanks to Dr. Bruce Draper, Dr. Ross Beveridge for their guidance and support to help finish this project successfully. Great thanks are also due to Leif Anderson for building this LaTeX document class, allowing me to meet the graduate school formatting requirements with no effort on my part.

Table of Contents

Abstract ii

Acknowledgements iii

List of Tables vi

List of Figures vii

Chapter 1. Introduction 1

Chapter 2. Background 5

2.1. Bit Codes 5

2.2. RIDMBC 6

Chapter 3. RIDMBC for Object Recognition 11

3.1. Dataset and parameters 11

3.2. Features and Classifiers 12

Chapter 4. Experiments 14

4.1. Common Setup 14

4.2. Experiment 1 : How many bits? 15

4.3. Experiment 2: Is there an optimal K? 18

4.4. Experiment 3: Bit Vector vs Floating-point vector 19

4.5. Experiment 4: Binarization of Feature Vectors 21

4.6. Experiment 5: Bit Selection 23

4.7. Experiment 6: Average bit vector 28

4.8. Experiment 7: Dimensionality reduction of features using PCA 29

Chapter 5. Conclusion 32

5.1. Future work 33

Bibliography 35

List of Tables

4.1 Comparision of bit selection methods for object classification when 5 bit vectors are selected out of 10 trained bit vectors. Random bit selection and correlation minmax are compared to best possible accuracy obtainable by exhausting all combinations of selecting 5 out of 10 bits. 28

List of Figures

2.1 Binary code representation of image. Image pairs of same class have smaller Hamming distance as compared to the ones of a different class. 5

2.2 Description of cells created by two-dimensional bitcodes. 6

2.3 Algorithm of RIDMBC [1] 8

2.4 Examples from PaSC and LFW. Top row: aligned images of the same person in LFW. Bottom row: aligned images of the same person in PaSC. 9

2.5 ROC curves of our method (RIDMBC) and two baselines (LRPCA and CohortLDA) on PaSC for the still challenge (a) Entire challenge problem (b) Frontal face images only 10

3.1 Sample images from Caltech-101 to show the diversity in categories and variation in illumination, pose and occlusion 12

4.1 Basic RIDMBC experiment for different number of binary classifiers, while k=1 for kNN 16

4.2 McNemar test on 5 splits of data using chi-square statistic, compared for different number of binary classifiers. Cells marked in blue have p>=0.05, and the ones in red have p<0.05 18

4.3 Experiment to demonstrate effect of changing k of kNN on accuracy, for RIDMBC and kNN classifier 20

4.4 Experiment to demonstrate difference in accuracy between using bit vector and floating point vector 21

4.5 McNemar test on 5 splits of data using chi-square statistic, compared for different number of binary classifiers of bit vectors and floating point vectors. Cells marked in blue have p>=0.05, and the ones in red have p<0.05 21

4.6 Experiment to demonstrate effect of binarizing the feature vectors. 22

4.7 Experiment to demonstrate effect of length of binary code. 24

4.8 Heatmap for correlation among bits selected by bit selection methods 27

4.9 Experiment to evaluate performance of average bit vector 29

4.10 Experiment to demonstrate effect of PCA compression on accuracy 30

4.11 Experiment to demonstrate effect of PCA compression on accuracy on varying bit code length 31

4.12 McNemar test on 5 splits of data using chi-square statistic, compared for different percentage of PCA compression on 1024-dimensional feature vector. The number of binary classifiers is 500. The cells marked in blue have p>=0.05, and the ones in red have p<0.05 31

CHAPTER 1

Introduction

Two trending techniques that are making advances in computer vision research are Convolutional Neural Networks and Visual Hashing. The goal of this paper is to analyze how these two interact in the broad domain of objects. Deep neural nets have proved to broadly represent image features, and binary codes have proved to be a powerful way to represent the intrinsic nature of image content in a compact way. Our research explores what kind of information is contained in feature vectors obtained from deep neural nets and what information can be binarized, in the context of object recognition. We also try to optimize the length of binary codes and select subsets of bit vectors to represent images so as to obtain the best classification results, while trying to bring down computational cost.

Visual categorization is central to computer vision research, and is based on measuring similarity among images of objects and scenes. Image comparison is the fundamental operation for retrieving and recognizing objects. Object recognition is the task of finding and identifying objects in images and videos. Object recognition systems are used for learning visual categories and identifying new instances of those categories.

The similarity of two images can be measured at the pixel, feature, object and semantic level. Most image retrieval and recognition systems have been constructed based on features. Gudivada [2] listed types of features for retrieval, including color, texture, shape and spatial. Feature extraction and representation is crucial, and finding how to represent features that are compact, and reflect the intrinsic content of images is still a challenging problem in computer vision.

An efficient and highly descriptive way of representing images is as binary codes [3]. An image can be represented as a code that discriminates it from other images in a dataset. Bit codes help with large scale object recognition because of their low computational cost. This is because the hash functions are learnt such that samples within the same category are closer in hamming space, and those from different categories are farther in hamming space.

Binary codes have great significance in object classification. Object classification is the task of assigning semantic labels to images using a classifier. Classifiers are trained using sets of images that already have preassigned labels. These classifiers, then can be used to predict category labels for any given test image. The query for classification is a lot faster if the features are compact and accurate, which makes binary codes an ideal choice.

A lot of algorithms have been introduced to represent images using binary codes, where each bit in the code represents a label assigned by a binary classifier [3, 4]. One such algorithm is Randomized Intraclass-Distance Minimizing Binary Codes(RIDMBC), introduced by Zhang et al. [1] in the context of face recognition. RIDMBC learns binary codes that are largely uncorrelated because of random initial assignments of bit labels to classes, by minimizing the Hamming distances within classes. RIDMBC uses simple features such as grayscale and LBP features to make the testing phase faster. The binary labels are obtained using the popular linear classifier SVM.

This paper explores the performance of RIDMBC in the broader domain of object recognition. The goal of the paper is not to beat the state-of-the-art results in the field of Object Recognition, but to explore the performance of RIDMBC in this context. Experiments are performed to examine different components of the algorithm such as binarization and length of binary codes, and to analyze the interaction of RIDMBC with deep neural nets.

We performed experiments to analyze RIDMBC and its variants compared to a baseline experiment on a simple classifier k-Nearest Neighbor. The experiments show that if number of weak classifiers are greater than a threshold, RIDMBC significantly outperforms kNN classifier. Recent research has shown that Convolutional Neural Networks(CNNs) features give state-of-the-art results in the domain of object classification, therefore, we use them for feature extraction instead of simple features like LBP and grayscale, and analyze their interaction with RIDMBC. The features are extracted using Caffe on GoogleNet [5].

The data set selected for the research is Caltech-101 [6]. It has a diverse set of 101 categories, comprising of about 9K images and is a popular choice for conducting object classification experiments. The highest classification performance recorded until April 2015 on Caltech-101 is about 93.5% by He et al. [7], whereas we recorded a maximum performance of about 89%.

A lot of components of RIDMBC have been investigated and varied to analyze its performance for classification. The first basic experiment was to explore the length of binary code i.e. the number of binary classifiers needed to achieve best performance and see if the accuracy saturates after a certain number of binary classifiers. The results showed that about 500 classifiers are sufficient to obtain stable classification results. Another experiment was performed to analyze the very basic component of binarization that the algorithm is based on. We examine how does the binarization of dot product of feature vector and classification hurts the accuracy, as compared to using the float values of dot product and using euclidean distance as the distance metric. The results showed that the binarization did not hurt the accuracy, which means that which side of the classifier the sample lies is important but distance from the classifier is not. Another set of experiments to analyze the

effect of binarization was to binarize the feature vectors obtained from deep neural nets. Values greater than zero in the feature vector were assigned a value of one. Results showed that floating point information contained in the feature vectors is useful and representative of image features.

The next set of experiments were performed to analyze the different ways of selecting a subset of bits that are enough to achieve a comparable accuracy, as when all the bits are used. We look at the trade-off in speed vs accuracy for bit selection. The results showed that random bit selection is as good as other methods of smartly selecting the bits, like decorrelating the bits or choosing best performing independent bits. Another experiment was performed to examine if there is a bit vector that could represent an entire object category. This would help in the prediction of category label of a test image. Instead of making comparison to each training image, now a hamming distance could be computed to just a vector that represents the whole category. This was accomplished using average bit vector which was deduced by averaging the bit vectors of all the training images. This gave a bit vector with weights attached to each bit. Using average bit vector gave about the same classification accuracy as compared to when hamming distance is computed to each training image.

Finally, an experiment was performed to examine the effect of reducing dimensionality of feature vectors using Principal Component Analysis(PCA). The compression was done by continuously halving the length of feature vector and we observed that a compression to about 25% does not hurt the accuracy a lot, and to 12.5% affects the accuracy by about 2-3%. The research is concluded by discussing the possibility of increasing the performance of RIDMBC.

CHAPTER 2

Background

2.1. Bit Codes

RIDMBC are based on the powerful representation of images as bitcodes. Bit codes are easier to store and retrieve. Image similarity is computed by measuring Hamming distance. Images that are similar to each other are closer in the Hamming space as compared to images of different classes. For example, in Fig. 2.1, it can be seen that images of starfish have similar bitcode as compared to the image of an aeroplane.

(a) Bitcode: 100001 (b) Bitcode: 100011 (c) Bitcode: 011111

Figure 2.1. Binary code representation of image. Image pairs of same class have smaller Hamming distance as compared to the ones of a different class.

Binary code is an image representation, obtained from a combination of binary classifiers, where each classifier corresponds to a bit. To demonstrate this, a simple experiment was performed in two dimensional space with two binary classifiers, as shown in fig. 2.3. Each classifier divides the space into two, and images are given a label 0 or 1 depending upon which side of the classifier they lie on. Each bitcode is a cell in bit space, and the number of bitcodes that can be created are 2^n, where n is the number of binary classifiers. In this case,

we have two classifiers, thus we get four cells(bitcodes) : 00,01,10,11. As we can see in the fig. 2.3, we have three classes of objects and each of them gets a bitcode. This experiment was perofrmed on real data by reducing the dimensions of the images to two using Principal Compnent Analysis. Images that are similar seem to be getting similar bitcode. But usually, the dimension of bitcode is much bigger than number of classes, so they are farther apart in the high-dimensional bit space.

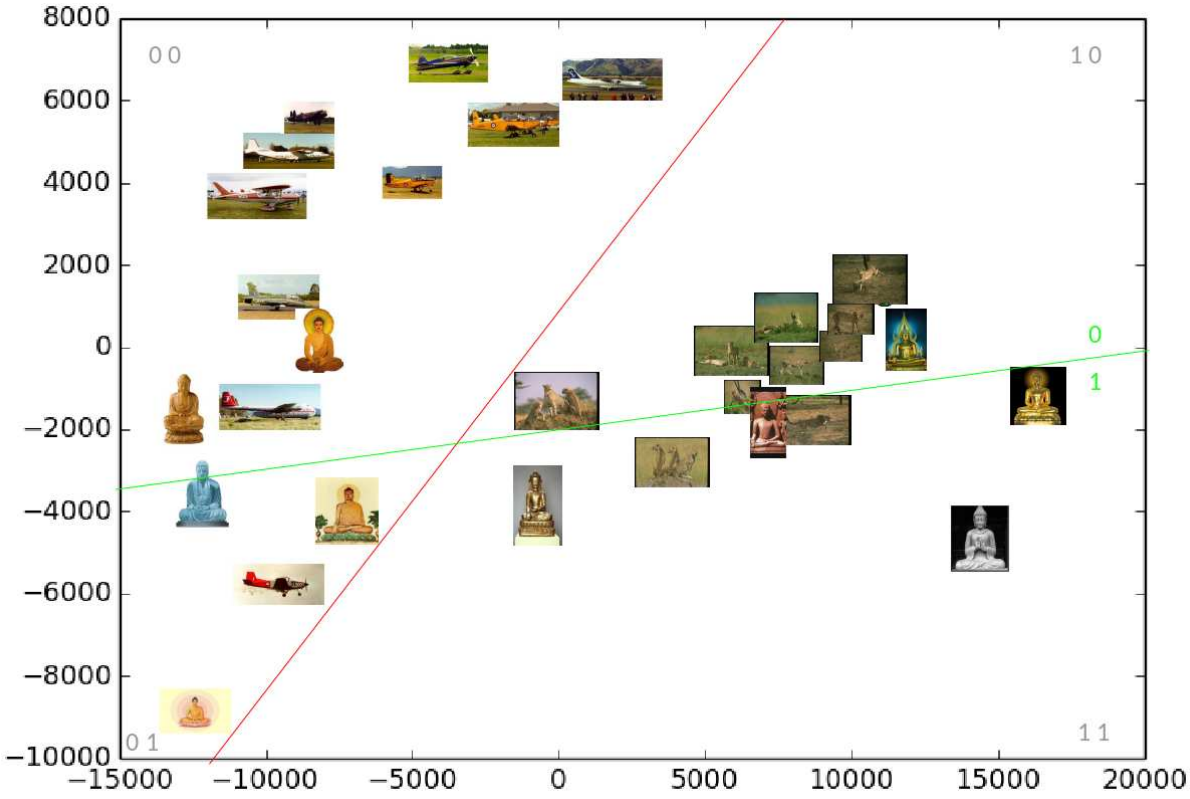


FIGURE 2.2. Description of cells created by two-dimensional bitcodes.

2.2. RIDMBC

The goal of RIDMBC algorithm is to represent face images with binary codes, and perform face recognition using Hamming distances. The binary codes are assigned to images by learning binary classifiers where images of the same category lie on the same side of each

classifier. Each bit in a binary code can be thought of as a split of the feature space into two half-spaces. Each bit is learnt by a binary classifier in a way that the same label is assigned to all images belonging to the same class. In training, we infer codes for training images, and learn classifiers to predict the codes; in testing, we apply those classifiers to a test image to produce a code.

2.2.1. Algorithm. The main strength of RIDMBC is randomization. The process begins by assigning binary values to images in a training set S, where images of the same class get same binary label, 0 or 1. In this process, roughly half of the classes get 0 and the other half gets 1, to balance the number of 0s and 1s, and in every iteration this initialization is randomized among classes. This randomization helps in producing different weak but uncorrelated classifiers.

The next step is to partition all the images of training set S into two disjoint subsets : S1 and S2. Then, a random initial assignment of binary label is made to each category, with a constraint that images of a particular category get the same label. A linear SVM classifier is trained on set S1 and this classifier is used to predict the labels for the set S2. The prediction would make some label assignments such that some images don't get the same label as the other images of the same class. Thus, we need to make label adjustments i.e. all the misassigned images get the same label that was assigned to majority of the images of the same class. After label adjustment, a new classifier is trained on set S2, which is then used to predict labels on set S1. This process keeps on iterating between S1 and S2 until it converges, where convergence would mean that labels of the classes are no longer changing.

The final step is to recombine S1 and S2 into S and train a final classifier on the new S. This SVM classifier represents one bit in binary code representation of images. Any number of such SVM classifiers can be learned by random initialization of labels of training set S.

Algorithm 1 Learning Bit Encoders

INPUT: Two subject disjoint image sets S_1 and S_2,
for i from 1 to k, the number of bits **do**
  Generate random $\hat{L}_1$ and $\hat{L}_2$ desired labels
  Initialize Training Data = $\{S_1, \hat{L}_1\}$
  Initialize Test Data = $\{S_2, \hat{L}_2\}$
  while labels in L_1 or L_2 changing **do**
    Train a classifier on Training Data
    Test the classifier on Test Data
    for Each subject j in Test Data **do**
      Let $L_j = \{l_{j,1}, l_{j,2}, ..., l_{j,m}\}$ be the predicted labels of all the images belonging to that subject
      Let $\hat{L}_j = \{\hat{l}_{j,1}, \hat{l}_{j,2}, ..., \hat{l}_{j,m}\}$, be the corresponding desired labels, satisfying $\hat{l}_{j,1} = \hat{l}_{j,2} = ..., = \hat{l}_{j,m}$
      if majority of L_j not equal to $\hat{L}_j$ **then**
        $\{\hat{l}_{j,1}, \ldots, \hat{l}_{j,m}\} = \{1 - \hat{l}_{j,1}, \ldots, 1 - \hat{l}_{j,m}\}$
      end if
    end for
    Swap the Training Data and Test Data.
  end while
  Combine $\{S_1, \hat{L}_1\}$ and $\{S_2, \hat{L}_2\}$ and train classifier a_i.
end for
Output: bit encoders

FIGURE 2.3. Algorithm of RIDMBC [1]

2.2.2. FEATURES AND CLASSIFIERS. In RIDMBC, the features used to represent images were selected in a way such that the task of face recognition is fast. Thus, grayscale pixel values and Linear Binary Pattern(LBP) were used as features to represent face images. Also, the data was registered, so these features did well on this task.

For the purpose of classification, linear Support Vector Machine (SVM) has been used. Among existing appearance models, linear SVM has proved to be a simple yet effective

choice. We use LIBLINEAR package for implementing SVMs. LIBLINEAR is an open source library for large-scale linear classification.

2.2.3. Results on Face Recognition. RIDMBC was introduced in the context of face recognition. The performance of RIDMBC was analyzed by conducting a cross-database experiment, where training was performed on the Labeled Faces in the Wild (LFW) and tested on the Point-and-Shoot Challenge (PaSC).

The reason RIDMBC wass evaluated on the challenging Point-and-Shoot Challenge dataset (PaSC) [8] is to avoid the possibility of results getting affected by codes which may have become specific to a particular data set. PaSC is a new, unrelated data set, which contains 9376 images of 293 people for testing. The images are taken at nine locations (including indoors and outdoors) using five point-and-shoot still cameras, with a lot of variation in pose of subjects and their distance to the camera. The test set has 4688 query images and 4688 target images. Image pairs are formed by taking one image from the query set and the other from the target set. Figure 2.4 shows example images from PaSC and LFW, which show the variation in pose, lighting and sharpness variations.

Figure 2.4. Examples from PaSC and LFW. Top row: aligned images of the same person in LFW. Bottom row: aligned images of the same person in PaSC.

Algorithms generate similarity scores for all image pairs in the test set. By looking at image pairs of interest, two ROCs can be plotted, one for frontal image pairs and one that combines both frontal and non-frontal images. Figure 2.5 presents the ROCs for RIDMBC along with those for two baseline algorithms provided by Colorado State University as part of the PaSC. ROCs are shown for all still images and just the frontal images. The RIDMBC algorithm performs better than both PaSC baselines.

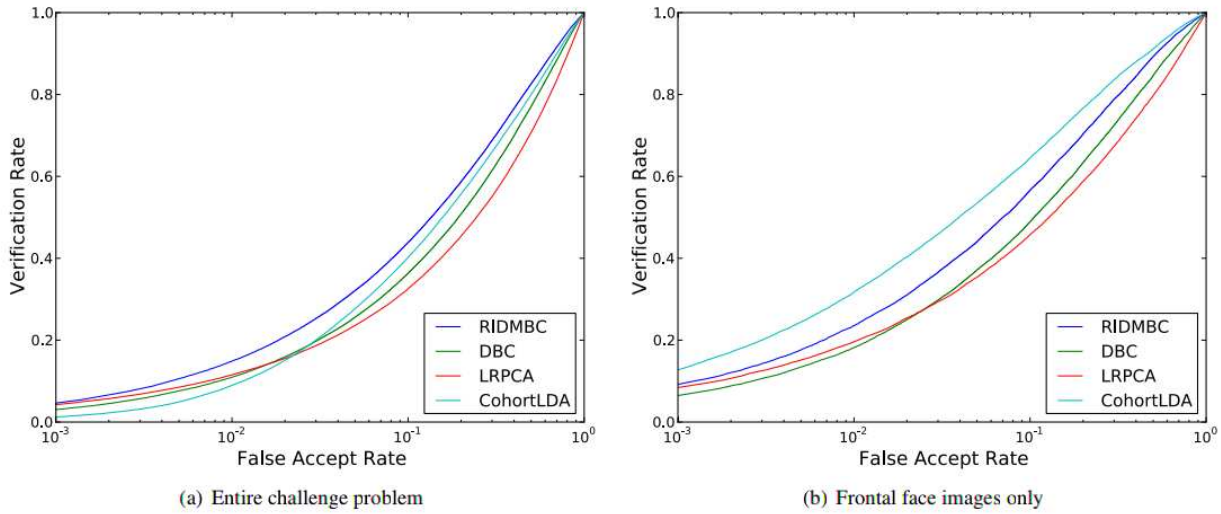


FIGURE 2.5. ROC curves of our method (RIDMBC) and two baselines (LRPCA and CohortLDA) on PaSC for the still challenge (a) Entire challenge problem (b) Frontal face images only

CHAPTER 3

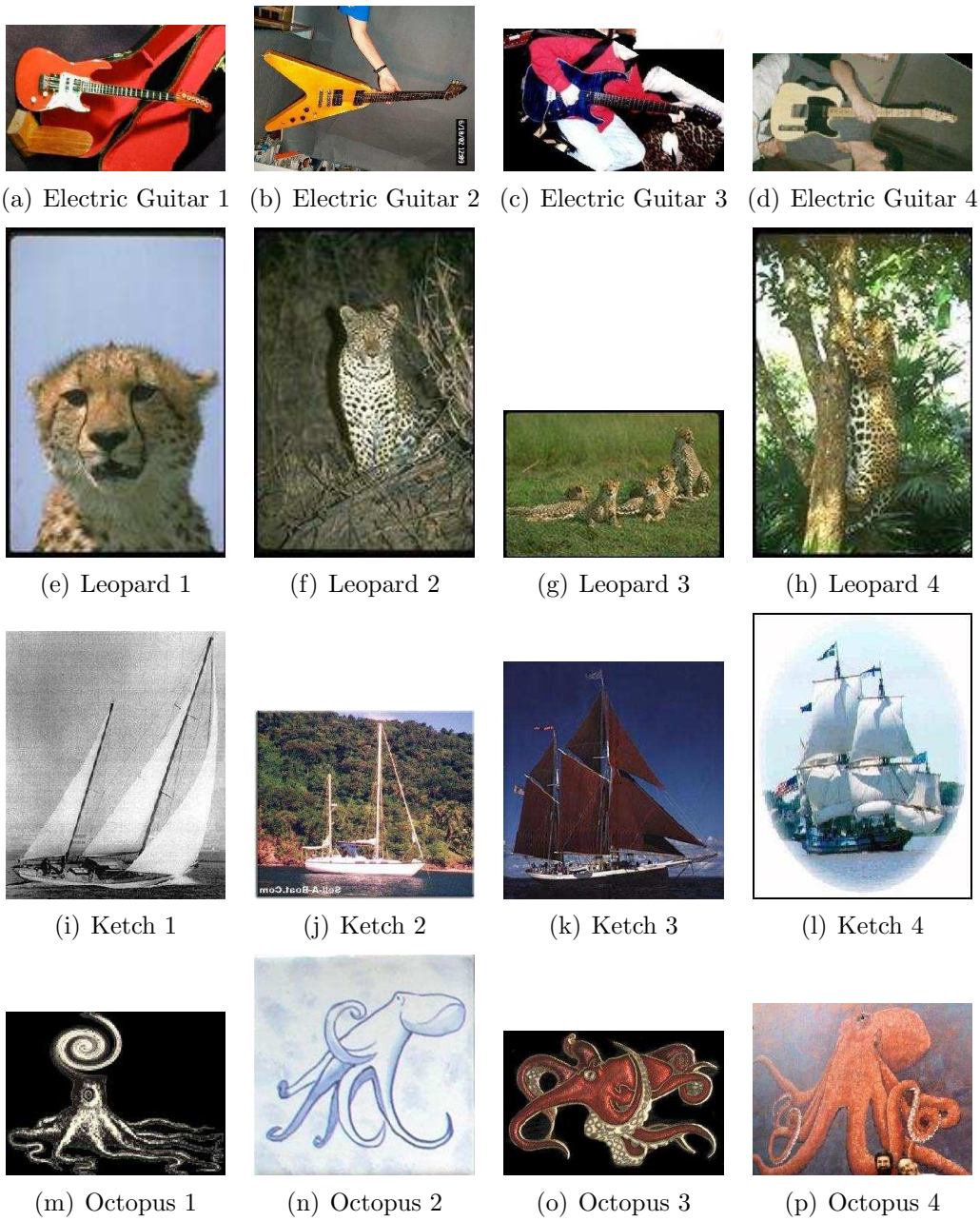
RIDMBC for Object Recognition

RIDMBC was originally developed for face recognition by Hao et al [1]. We have extended it to a broader context of object recognition, to examine how well it performs for objects, which have more diversity as compared to faces. Other objectives of this research is to explore different components of the RIDMBC algorithm and analyze its interaction with Convolutional Neural Networks(CNN).

3.1. Dataset and parameters

Caltech-101 dataset was selected for performing experiments of object classification as it contains a diverse set of objects and is a popular choice for object classification experiments. Caltech-101 data set consists of a total of 9,146 images, split between 101 different object categories with each class having a lot of intraclass variation. Each object category contains between 30 and 800 images. Each image is about 300x200 pixels. Common and popular categories such as faces tend to have a larger number of images than others. Figure 3.1 shows some sample images from categories : electric guitar, leopard, ketch and octopus. We can see the diversity in interclass and intraclass variations in the data set.

Since the smallest size of a data set in Caltech-101 is 31, so we select 31 images from each category. Out of 31 images from each set, 20 images of each class are added to the training set and 11 images are added to the test set. So, the total size of training set is 2020 and that of test set is 1111. The experiments are performed on five splits of the same data,



(a) Electric Guitar 1 (b) Electric Guitar 2 (c) Electric Guitar 3 (d) Electric Guitar 4

(e) Leopard 1 (f) Leopard 2 (g) Leopard 3 (h) Leopard 4

(i) Ketch 1 (j) Ketch 2 (k) Ketch 3 (l) Ketch 4

(m) Octopus 1 (n) Octopus 2 (o) Octopus 3 (p) Octopus 4

FIGURE 3.1. Sample images from Caltech-101 to show the diversity in categories and variation in illumination, pose and occlusion

where samples of training and test data are shuffled. The splits of data are created in a way as to have minimum overlap.

3.2. Features and Classifiers

The features used by Hao et al. [1] for face recognition were grayscale and LBP, which were fast and simple to compute. But for the purpose of object recognition, we picked Convolutional Neural Networks for feature extraction because they significantly outperform all other features in the context of object recognition. Google proposed a deep convolutional neural network architecture codenamed "Inception", which was responsible for setting the new state of the art for classification and detection in ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014). Although CNNs obtained from Googlenet are trained on ImageNet data set, but ImageNet data is so huge and diverse that we expected it to represent the image features of CalTech-101 data set too, and it did give impressive results.

GoogleNet features were extracted using Caffe for each image sample from the topmost layer, which has been trained by the deep neural nets. The feature vector for each image is 1024 dimensional. For the purpose of classification, we pick the SVMs as used by Hao et al. for face recognition, using LIBSVM.

CHAPTER 4

Experiments

This chapter discusses experiments conducted by us to evaluate RIDMBC for object classification, and analyze its interaction with GoogleNet. First section discusses a common experiment setup used for all the following experiments. Each following experiment is predicated upon results of the previous experiment, and uses that information to evaluate, analyze and optimize performance of RIDMBC.

4.1. Common Setup

The common setup for all experiments includes finding feature descriptor for each image, using binary classifier, and finding classification accuracy. Feature descriptor for each image is obtained using GoogleNet Caffe simulation from the topmost layer, which is trained on Imagenet data set. The feature vector is of length 1024. SVM classifiers are used as binary classifiers which produce bits in binary code used to represent each image sample. LIBLINEAR [**?**] library is used to obtain SVM classifiers.

Binary classifiers obtained from the training phase are projected on test data and sign of dot product tells which side of the classifier does the test sample lies. If we take sign of the dot product, we get binary values describing label of a test sample for each binary classifier. Classification is performed using k nearest neighbors. For each test image, we look at k nearest neighbors of the training samples. The nearest neighbor for each test sample are found using Hamming distance. This means that we add 1 to distance if sign of a bit of a test sample has different sign as compared to the corresponding bit of the training

sample, i.e. if they lie on different sides of the hyperplane for that dimension. For k-nearest neighbors of a test sample, the Hamming distance can be same to several training samples, and count of these neighboring training samples can be greater than k. So, for such test samples, classification accuracy is averaged for all the training samples over the remaining k neighbors. If there are already p neighbors, where p¡k, that are closer to the test sample, and there are neighbors that have an equal Hamming distance to that test sample and their count is greater than k-p, then accuracy of kNN for remaining k-p neighbors, is averaged over neighbors with equal distance to that test sample.

4.2. Experiment 1 : How many bits?

Experiment 1 is a basic experiment performed to evaluate accuracy of RIDMBC on the given data set. The goal of this experiment is to examine a basic component of RIDMBC, which is number of binary classifiers.

For this experiment, k is assumed to be one for kNN. The experiment is performed on five splits of the data, so as to make sure that the results are not biased to a particular split of the data set. Classification accuracy is obtained by averaging classification results over all test images.

Fig. 4.1 shows performance of RIDMBC on this basic experiment. Vertical lines in graph represent error bars, since the experiment was performed on five splits of the data set. It can be observed that maximum accuracy is about 88%, and it starts to saturate when number of classifiers is about 500. To make sure that impact on accuracy by increasing the number of classifiers from 500 will not be statistically significant, McNemar's test is performed.

4.2.1. McNemar test. McNemar's test is a statistical test used to compare paired proportions. It is applied to 2x2 contingency tables with a dichotomous trait, with matched

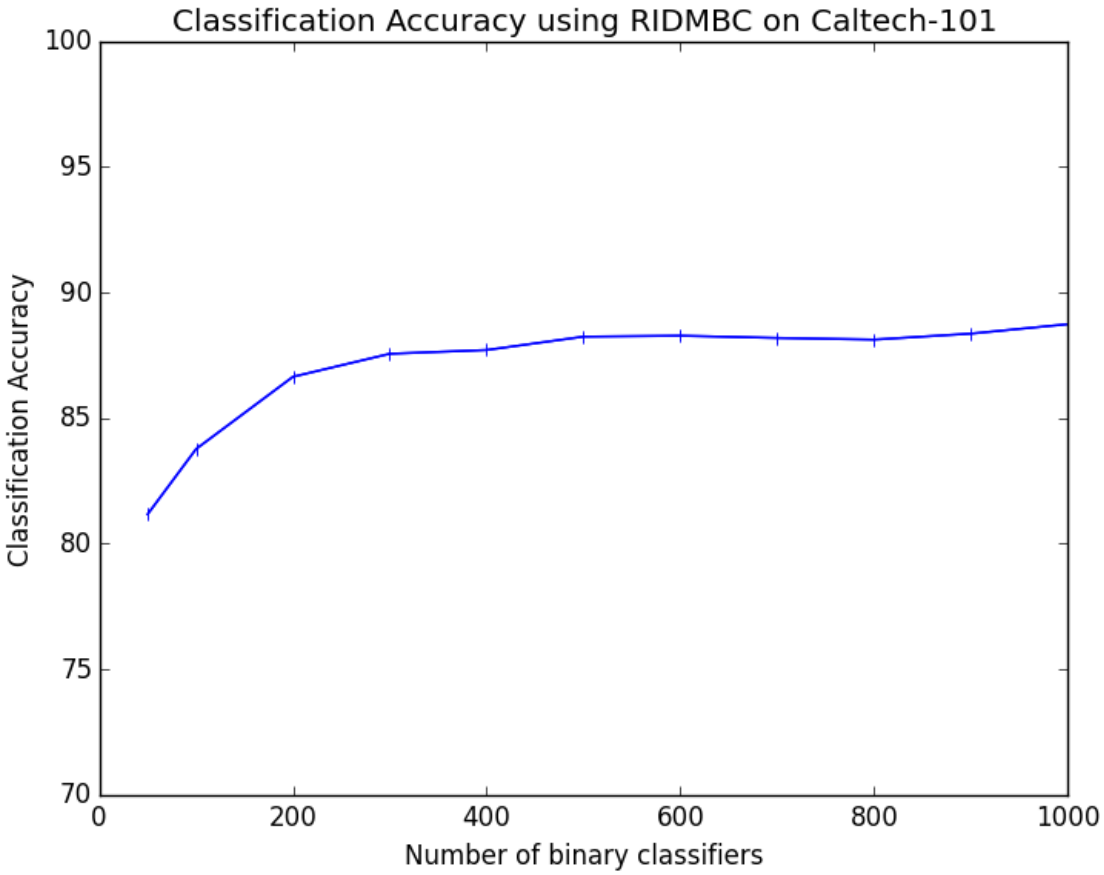


FIGURE 4.1. Basic RIDMBC experiment for different number of binary classifiers, while k=1 for kNN

pairs of subjects, to determine whether row and column marginal frequencies are equal. McNemar's test assesses significance of difference between two correlated proportions, such as might be found in the case where two proportions are based on same sample of subjects for different parameters.

	Condition(Group 2)		
Condition(Group 1)	**Yes**	**No**	**Totals**
Yes	A	B	A+B
No	C	D	C+D
Totals	A+C	B+D	n

The proportions of data under consideration is:

π_1= proportion of respondents in the population who would answer yes to Condition 1.

π_2= proportion of respondents in the population who would answer yes to Condition 2.

These are approximated by the following sample values:

$$p_1 = \frac{A+B}{n}$$

sample proportion of respondents who answered yes to Condition 1.

$$p_2 = \frac{A+C}{n}$$

sample proportion of respondents who answered yes to Condition 2.

McNemar's test is for testing Null hypothesis that there is no difference between two proportions of a population based on frequency counts to two conditions:

$$H_0 : \pi_1 = \pi_2$$

The null hypothesis of marginal homogeneity states that the two marginal probabilities for each outcome are same, i.e. $p_a + p_b = p_a + p_c$ and $p_c + p_d = p_b + p_d$.

The test statistic is given by

$$Z = \frac{B-C}{\sqrt{B+C}}$$

A high p-value prevents from not rejecting Null hypothesis at 5 percent level of significance.

	50	100	200	300	400	500	600	700	800	900	1000	1100	1200
50		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100			0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
200				0.11	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300					0.49	0.05	0.19	0.10	0.04	0.29	0.00	0.00	0.15
400						0.20	0.29	0.37	0.18	0.62	0.02	0.08	0.59
500							0.45	0.67	0.94	0.39	0.29	0.12	0.75
600								0.70	0.36	0.93	0.05	0.43	0.53
700									0.59	0.63	0.11	0.54	0.43
800										0.31	0.30	0.24	0.90
900											0.02	0.07	0.70
1000												0.78	0.11
1100													0.48

FIGURE 4.2. McNemar test on 5 splits of data using chi-square statistic, compared for different number of binary classifiers. Cells marked in blue have p>=0.05, and the ones in red have p<0.05

The observation that can be made from McNemar's experiment is twofold. First, it can be observed that if number of binary of classifiers is about 500, then it will prevent from not rejecting the Null hypothesis. Second, if number of classifiers is less than 200, then Null hypothesis can certainly be rejected.

4.3. EXPERIMENT 2: IS THERE AN OPTIMAL K?

The next experiment was carried out to find out optimal value of k(of k-nearest neighbors of test samples), which is used to find out classification accuracy. If increasing the k does not lead to increase in accuracy, then a small k would be a better choice for experiments as it will be computationally cheap.

Two experiments were performed to evaluate the optimality of k. First experiment was performed to examine a baseline classification experiment where k-nearest neighbor was

used as classifier. Second experiment was performed using the same configuration as in the Experiment 1, where the number of classifiers was fixed to be 500. Both the experiments were performed by varying values of k from 1 to 13.

Fig. 4.3 shows that using higher values of k does not improve accuracy for any of the two experiments. When kNN is used as a classifier, the classification accuracy drops drastically from 81% to 64% as k is increased from 1 to 13. When kNN is used in RIDMBC for classification, the accuracy neither increases nor decreases. Therefore, increasing value of k is not improving accuracy, and makes classification computationally expensive. So, for all the following experiments, k is assigned a value of 1. To confirm the claim statistically, McNemar's test was used and it showed that mistakes made by changing the value of k are on the same data, so that confirms there is no statistical difference.

4.4. Experiment 3: Bit Vector vs Floating-point vector

The goal of this experiment is to use floating point values instead of bit values. To explain further, we are trying to examine if distance of a test image vector from a binary classifier is significant or not. In RIDMBC, we take sign of dot product of test image vector and binary classifiers, and obtain binary labels. Taking sign of dot product is actually an approximation and thus, compression of the result. So, an important step is to analyze trade-off in accuracy of this compression. Since we are throwing away data, it is expected that raw floating point values should perform better than bit values. Raw values obtained from dot product are float values that take 32 bits to store the result. So, taking sign to obtain bit vector gives a compression of 1:32. Also, if floating point values are used, the distance computation becomes expensive because computing Euclidean(L2) distance is expensive than Hamming distance.

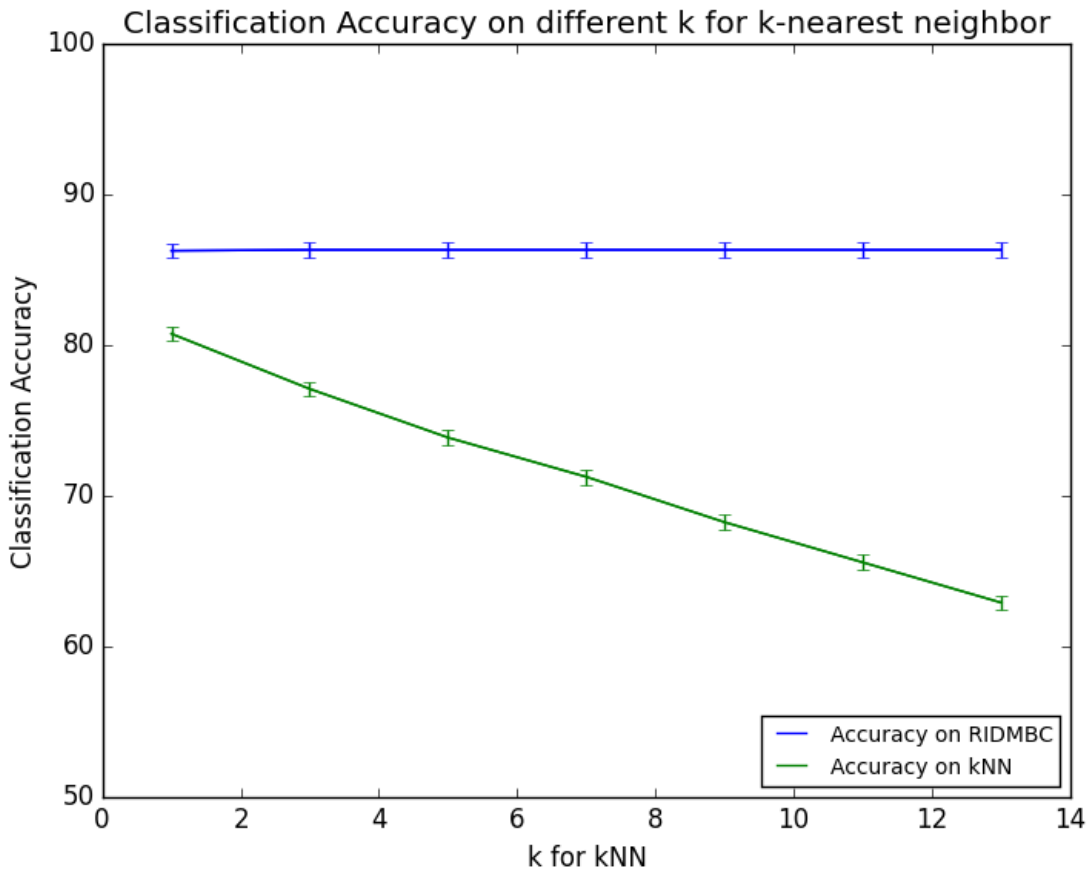


FIGURE 4.3. Experiment to demonstrate effect of changing k of kNN on accuracy, for RIDMBC and kNN classifier

So, an experiment is performed to see effect of using floating point values instead of bit values. Fig. 4.5 shows that using either of bits or float values from the dot product gives similar classification result. Using float values gives slightly better results until the length of binary code is 200 but is outperformed by bits by about 2% after that. This outcome is strange but helpful in the way that throwing away information is not hurting accuracy, it rather improves by 2%. Thus, it can be observed that distance of feature point from the classifier can be discarded. Only the information of which side of the classifier the feature point lies is sufficient for the algorithm to produce good classification results. This shows the strength of using binary codes for classification.

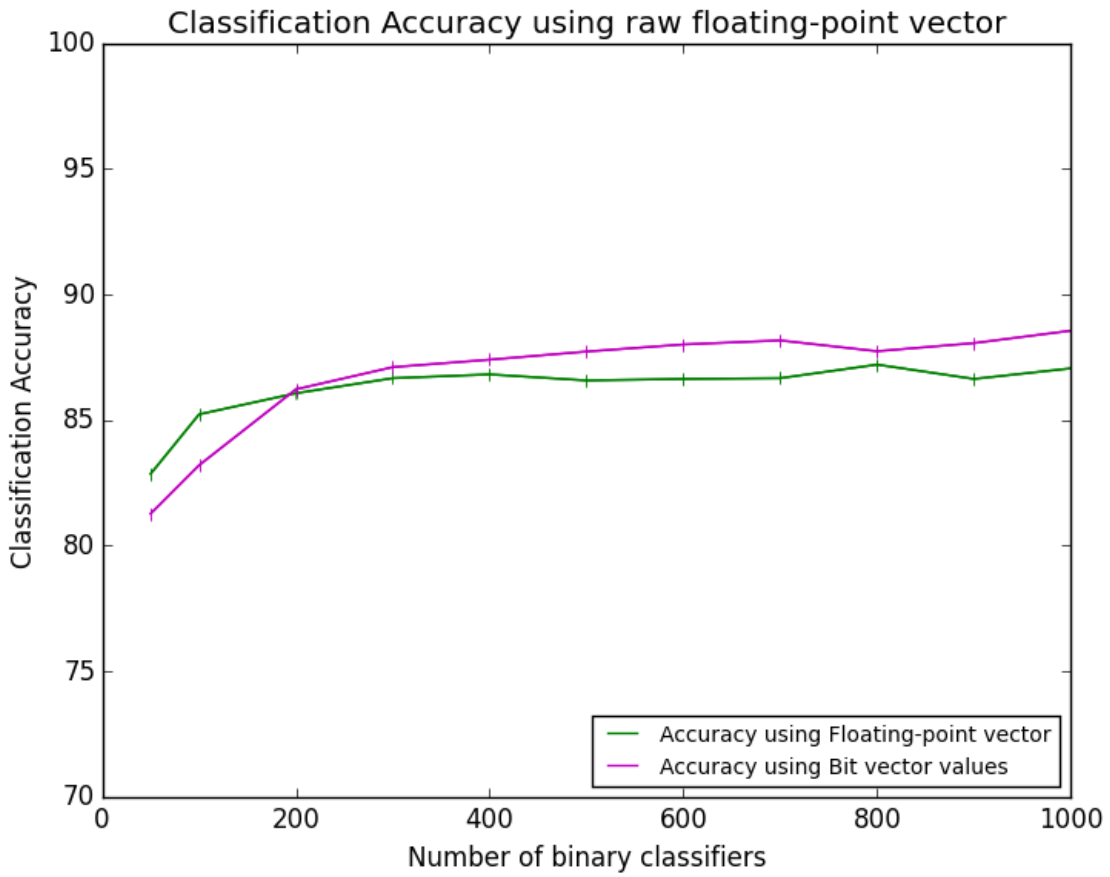


FIGURE 4.4. Experiment to demonstrate difference in accuracy between using bit vector and floating point vector

50	100	200	300	400	500	600	700	800	900	1000
0.86	0.50	0.01	0.01	0.07	0.00	0.02	0.21	0.00	0.02	0.03

FIGURE 4.5. McNemar test on 5 splits of data using chi-square statistic, compared for different number of binary classifiers of bit vectors and floating point vectors. Cells marked in blue have p>=0.05, and the ones in red have p<0.05

4.5. Experiment 4: Binarization of Feature Vectors

From the above experiment described in Section 4.3, we saw that binarization of dot product of feature vectors and weak classifiers did not hamper the classification accuracy. So, it can be observed that binarization could be useful as it helps in throwing away information and still does not has a negative impact on accuracy. Thus, we perform an experiment to

explore effect of binarization on feature vectors, as most of the values in Googlenet features are zero. The binarization of feature vectors means all the value above zero are assigned a value one.

The results of the experiment as depicted in Fig 4.6, show that binarization of feature vectors does not help in increasing the accuracy. The accuracy rather drops significantly from 89% to 78%. Thus, we can conclude that floating point information contained in the feature vector is useful.

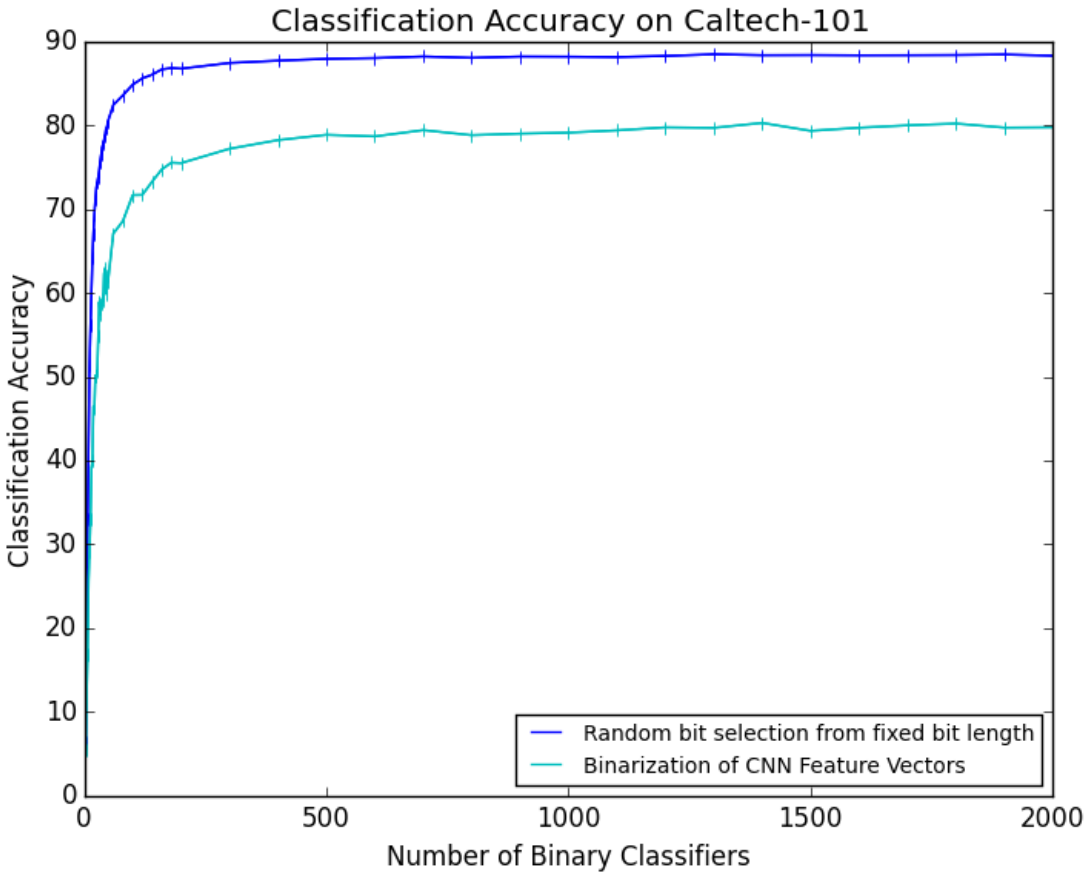


FIGURE 4.6. Experiment to demonstrate effect of binarizing the feature vectors.

4.6. Experiment 5: Bit Selection

The aim of the set of experiments described in this section is to examine effect of length of binary code on classification accuracy, and find ways to select a subset of bits that can achieve the maximum possible accuracy this algorithm can obtain. From Experiment 3, we can say that bit vectors are a better choice than floating point vectors. Now, the next step is to optimize length of binary code. One of the strengths of RIDMBC is that there is no tap on length of bit code, i.e. we can use as many binary classifiers as desired. But having longer codes than necessary will increase space and time complexity. So, we explore different ways to select a subset of binary classifiers to obtain best classification results.

First of all, we compare all methods of optimizing length of binary code using RIDMBC algorithm to a baseline experiment. In the baseline experiment, we pick a simple classifier: k-nearest neighbor classifier. As we can see in Fig. 4.7, the baseline experiment gives an accuracy of about 80%, as shown by yellow line.

Now, we want to analyze effect of length of binary code using RIDMBC on accuracy. To accomplish that, a fixed number of binary classifiers are trained and a subset of bits are selected for classifying test data, and this method is named as bit selection.

4.6.1. Bit Selection. In bit selection, a subset of bits is chosen by training a fixed number of binary classifiers. Then, we analyze how many bits are necessary to obtain the classification accuracy same as when all the binary classifiers are used. For this experiment, the total number of classifiers trained are 2000 and a smaller set of classifiers are picked for classification, depicted by x-axis of Fig. 4.7. The different ways used to select bits are discussed below.

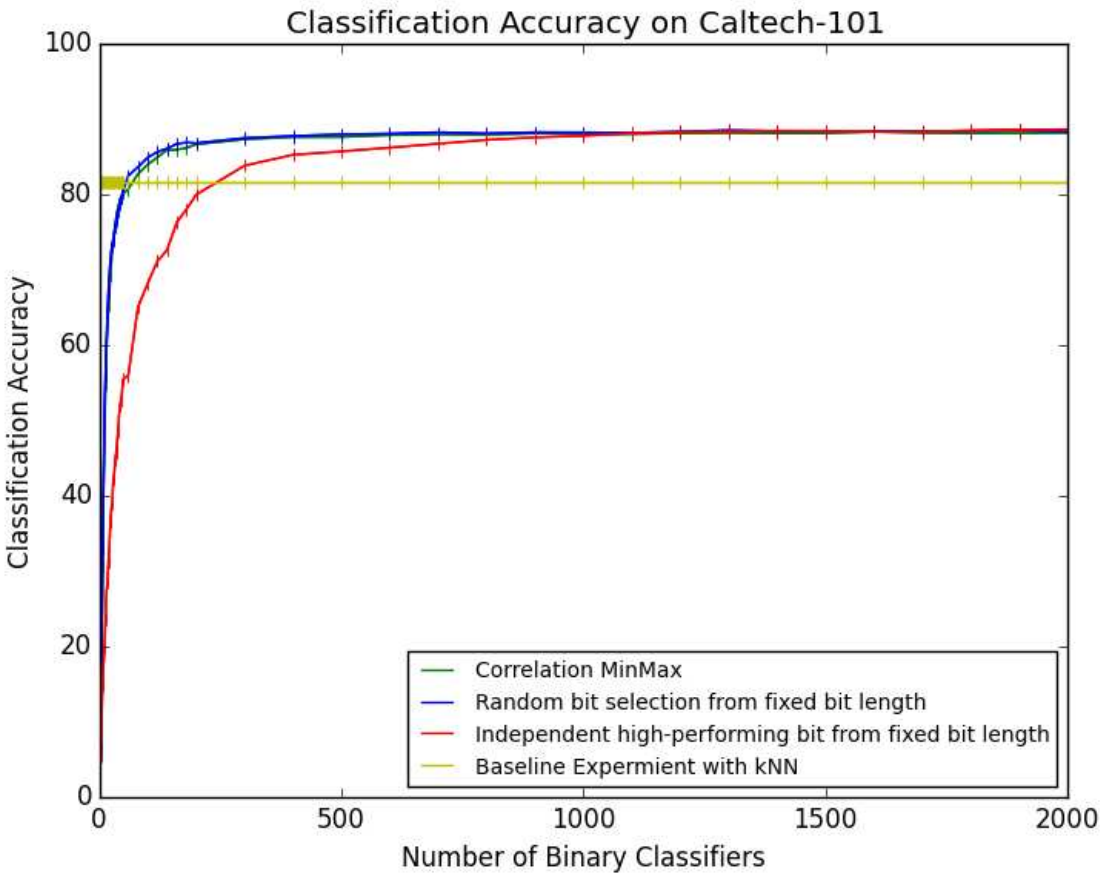


FIGURE 4.7. Experiment to demonstrate effect of length of binary code.

- *Randomized:* The first intuitive way is to pick a subset of classifiers randomly for classification. This is not a smart way to pick classifiers, but since roots of RIDMBC algorithm are based on randomization, it can be expected to perform well. As we can see from classification result in Fig. 4.7, as depicted by blue line, randomly selecting about 400 classifiers reaches the maximum accuracy obtained when all the classifiers are selected. Also, it can be observed that just 100 randomly selected classifiers are enough to beat kNN classifier.
- *Best performing independent bits:* The next step is to pick classifiers smartly, and see if number of classifiers, needed to achieve the maximum accuracy obtained when

all classifiers are used, can be reduced. In this experiment, bits of binary code were chosen on the basis of performance of each individual bit. Accuracy of each bit was measured by considering each test sample and checking if label of each bit of that sample matched with dominant label of corresponding bit of training samples of the class to which that test sample belongs. The bits are sorted in order of their decreasing accuracy, and the subset of best performing independent bits are picked. Fig. 4.7 shows that this way of bit selection, as depicted by the red line, does not perform well. An explanation to justify drop in performance could be that the bits selected are correlated and make the same mistakes. Therefore, a solution to this problem could be picking bits by a greedy method that selects binary classifiers that are not correlated to each other.

- *Greedy Correlation Minmax:* The aim of using Minmax Correlation algorithm is to pick bits that have low correlation with each other. Bit selection begins by choosing first bit using the above algorithm of best performing independent bit. Then, for every non chosen bit, we find correlation with each chosen bit and choose the maximum correlation with each chosen bit of each unchosen one. The unchosen bit which has the minimum value of these maximum correlations would be chosen next. Fig. 4.7 shows that Greedy Correlation Minmax, as depicted by green line, performs better than best performing independent bits but only as good as random bit selection.

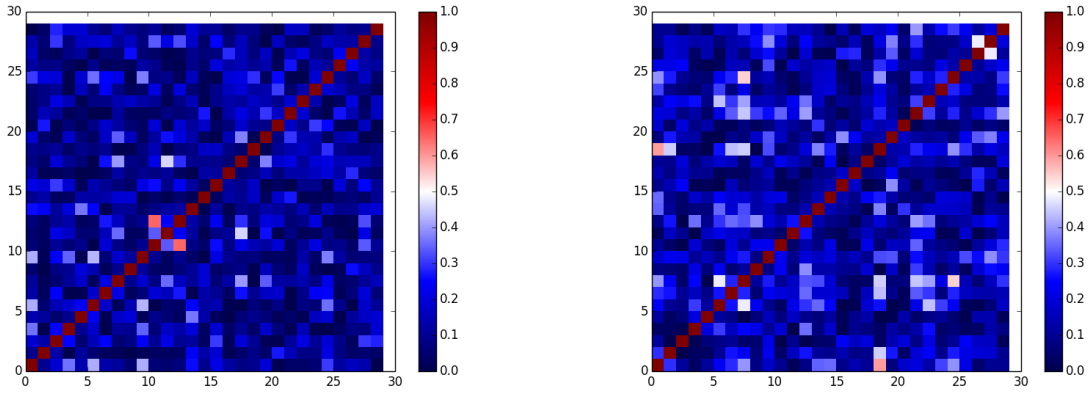
In Fig. 4.7, the presence of error bar at the any point in the graph means that an experiment was conducted at that bit code length(x-axis of graph). It can be seen that there are more error bars for smaller number of classifiers. This is because accuracy changes

drastically when the number of classifiers is low, so the experiments are performed more for smaller numbers, so as to have better understanding of the curve.

4.6.2. Correlation in bit selection methods: As observed in the above bit selection experiments, random bit selection performs as good as greedy correlation minmax, whereas best performing independent bit selection is performing much worse than the other two methods. An explanation could be presence of higher correlation in best performing independent bits, as compared to other two methods. So, an experiment was performed to find the correlation among the binary classifiers selected by these methods.

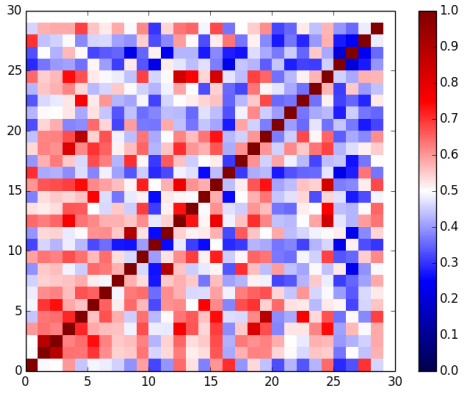
Fig. 4.8 shows heatmap of correlation of binary classifiers with each other, for each method of bit selection. The diagonal is entirely red because correlation with self is 1, which is maximum. In fig. 8(c), we can see that it is much more red as compared to other two heatmaps, which indicates correlation is much higher among the binary classifiers selected by best performing independent method of bit selection. Fig. 8(a) and 8(b) show that correlation among the bits is not too high, and thus, they perform well.

4.6.3. Exhaustive bit selection. From the above experiments, it seems that random bit selection is the fastest method of bit selection and gives as much accuracy as any other method. To determine whether it is the best method of bit selection and gives the best possible accuracy as can be obtained by RIDMBC, the only way to assure that is to exhaust all the possible k sized subsets of a bigger set and see what is the maximum accuracy that can be obtained. But this exhaustive method would take a long time if done on large number of classifiers. So, we train 10 classifiers and select all subsets of 5 bits, so 10 choose 5 gave 252 possible selections.



(a) Random bit selection

(b) Greedy Correlation Minmax



(c) Best performing independent bits

FIGURE 4.8. Heatmap for correlation among bits selected by bit selection methods

Table 1 shows the maximum accuracy that can be obtained by exhausting all possible selections compared to the above bit selection methods. The maximum classification accuracy obtained was 20.34%. The mean accuracy among all the combinations of 5 bits was 17.03, with a standard deviation of 1.15. Since, the difference between mean accuracy and maximum accuracy is not negligible, it can be concluded that random bit selection will have very low probability to obtain the maximum possible accuracy. A couple of experiments were performed using random bit selection and correlation minmax method. Correlation minmax method gave an accuracy of 16.3, which is close to mean accuracy but not close to

maximum accuracy obtainable, thus, not making it any better than random. Hence, there is a scope of selecting bits more smartly to achieve the maximum accuracy.

	Classification Accuracy (%)
Random	15.4
Correlation minmax	16.3
Maximum	20.34
Mean	17.03
Standard Deviation	1.15

TABLE 4.1. Comparision of bit selection methods for object classification when 5 bit vectors are selected out of 10 trained bit vectors. Random bit selection and correlation minmax are compared to best possible accuracy obtainable by exhausting all combinations of selecting 5 out of 10 bits.

4.7. Experiment 6: Average bit vector

The goal of this experiment is to find out if there is a binary code representation that would represent the whole object category, which would help in predicting the category label for a test image. For prediction of category of a test image, the binary code of a test image is compared to binary code of all the training images. In this experiment, the binary codes of all the training images of a particular category are averaged to produce a binary code with weights. The binary code consists of bit labels that are in majority and their weight is the difference between count of the dominating label and the other label, divided by total number of samples. So, a weighted hamming distance is computed for each test image instead of simple hamming distance but number of comparisons are reduced from number of training samples to number of object categories.

Fig. 4.9 shows the result of using average bit vector for object classification, as compared to when normal hamming distance is used. We can see that classification accuracy is about the same. So, using bit vectors, the classification process is speeded up without loss in

accuracy. Thus, it can be concluded that average bit vectors are a good representation of an object category.

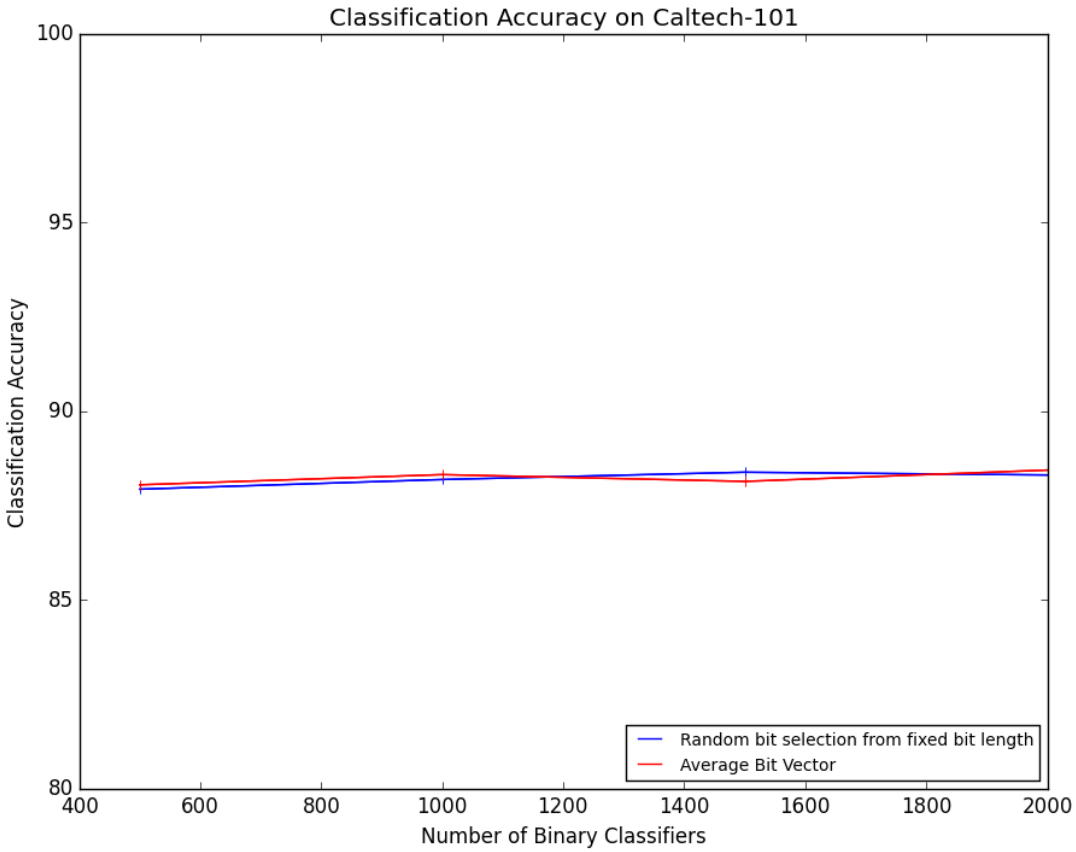


FIGURE 4.9. Experiment to evaluate performance of average bit vector

4.8. EXPERIMENT 7: DIMENSIONALITY REDUCTION OF FEATURES USING PCA

The feature vector obtained from GoogleNet for each image is 1024-dimensional. We conduct an experiment to examine if information contained in 1024 dimensions could be compressed such that it does not impact classification accuracy. To accomplish that, we use Principal Components Analysis(PCA) to reduce the dimensionality by picking the eigenvectors with maximum variance. In this experiment, we reduce dimensionality of feature vector

repeatedly by 50% until we see a considerable drop in accuracy. The original length of 1024 is reduced to 512, 256, 128, 64 and 32. Fig. 4.10 shows that accuracy is not affected until the dimension is compressed to 128. Therefore, a compression to about 12.5% of the original dimension does not hurt the accuracy.

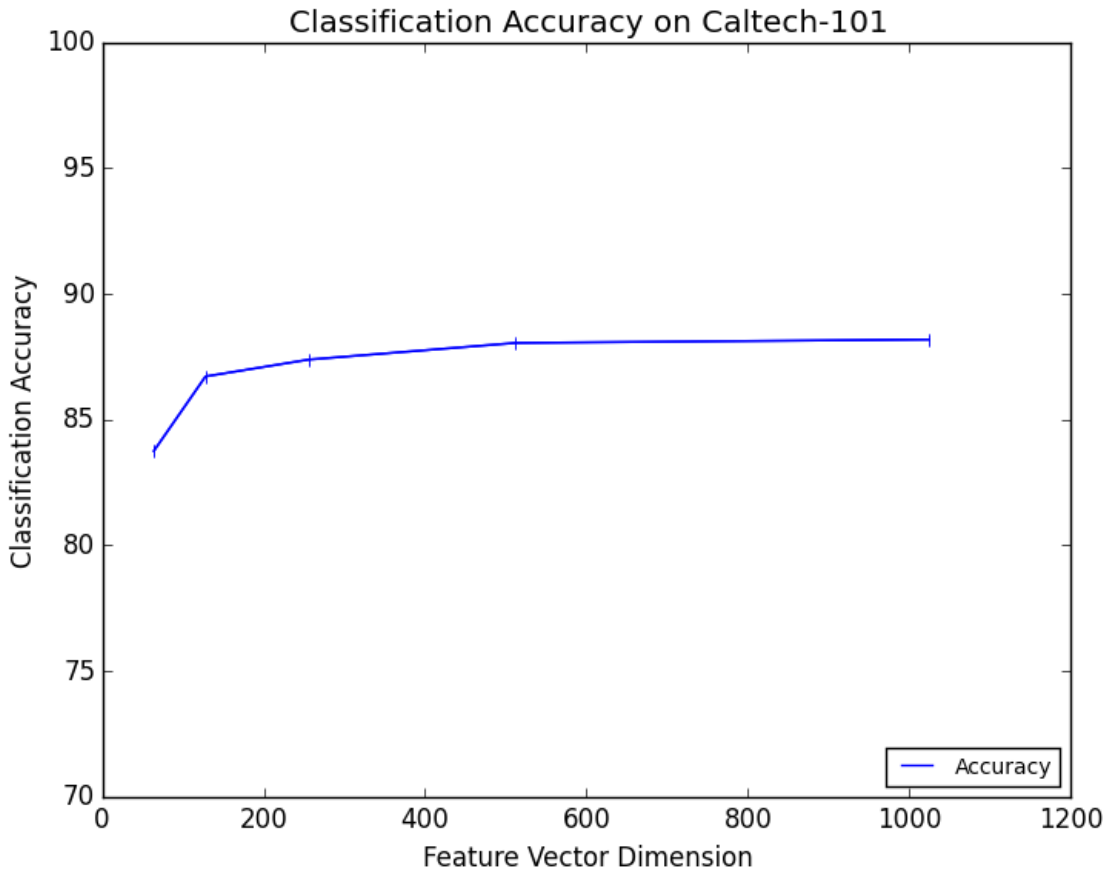


FIGURE 4.10. Experiment to demonstrate effect of PCA compression on accuracy

Now, lets examine the effect on accuracy of reducing dimensionality on varying the length of binary code, i.e. the number of binary classifiers. From the Fig. 4.11, we can see that the accuracy is not affected by compressing the original feature dimension to 12.5% and using just 500 bits. Fig. 4.12 shows that compression of different percentages greater than 12.5% are not statistically significant.

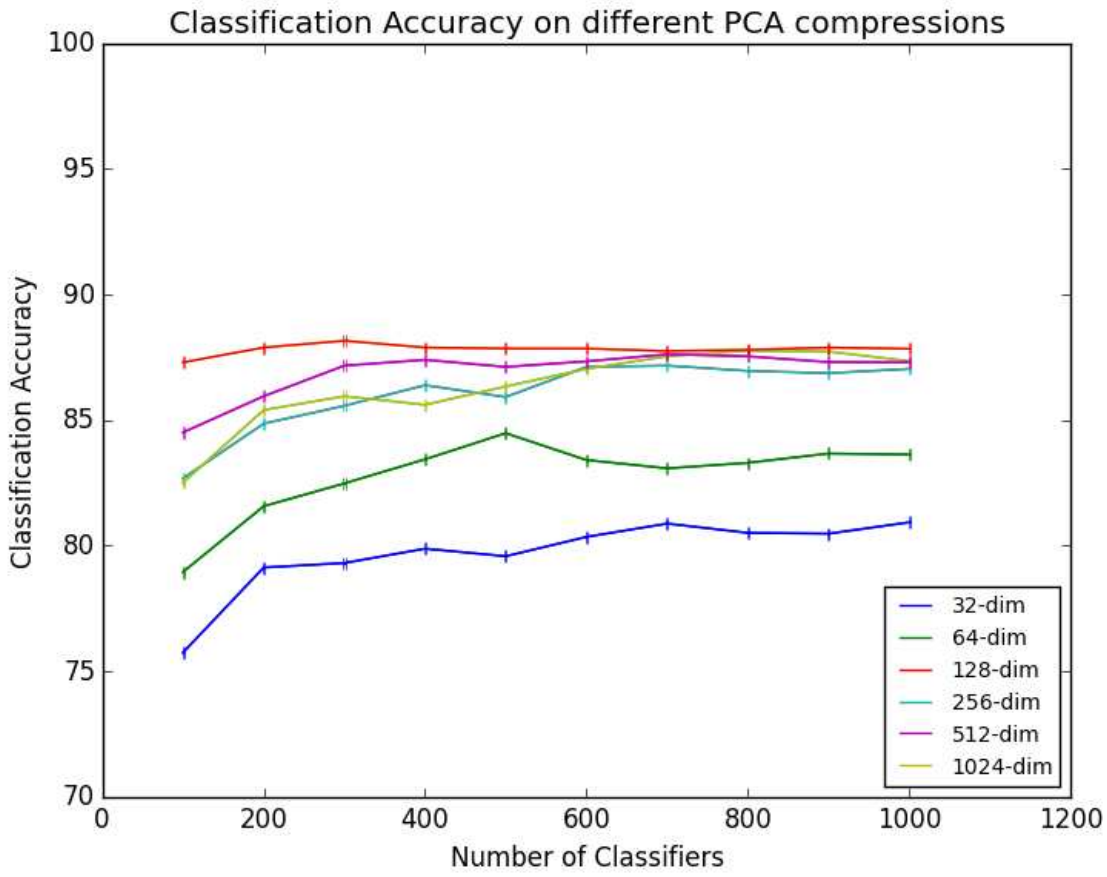


FIGURE 4.11. Experiment to demonstrate effect of PCA compression on accuracy on varying bit code length

	64	128	256	512	1024
64		0.00	0.00	0.00	0.00
128			0.47	0.65	0.19
256				0.84	0.55
512					0.26

FIGURE 4.12. McNemar test on 5 splits of data using chi-square statistic, compared for different percentage of PCA compression on 1024-dimensional feature vector. The number of binary classifiers is 500. The cells marked in blue have p>=0.05, and the ones in red have p<0.05

CHAPTER 5

Conclusion

This paper presents the analysis of interaction of Convolutional Neural Networks and Visual Hashing. The algorithm of RIDMBC which was originally developed for face recognition, was extended to a broader domain of object recognition, to examine the strength of binary codes to represent object images. We explore what kind of information is contained in feature vectors obtained from deep neural nets and which floating point information can be compressed and thus binarized, in the context of object recognition. We also try to optimize the length of binary codes and examine a way of selecting a subset of bit vectors to represent images in a way as to obtain best classification results, while trying to bring down computation cost. The experiments were performed on Caltech-101, which contains a diverse set of 101 categories, comprising of 9K images, that have a lot of interclass and intraclass variations.

The first basic experiment was to explore the length of binary code i.e. the number of binary classifiers needed to achieve best performance and see if the accuracy saturates after a certain number of binary classifiers. The results showed that about 500 classifiers are sufficient to obtain the maximum accuracy of about 88%, which does not change if the number of classifiers is increased.

Another experiment was performed to analyze the very basic component of binarization that the algorithm is based on. We examine how does the binarization of dot product of feature vector and classification hurts the accuracy, as compared to using the float values of dot product and using euclidean distance as the distance metric. The results showed that

the binarization did not hurt the accuracy, which means that which side of the classifier the sample lies is important but distance from the classifier is not.

Another set of experiments to analyze the effect of binarization was to binarize the feature vectors obtained from deep neural nets. Results showed that floating point information contained in the feature vectors is useful and representative of image features.

The next set of experiments were performed to analyze the different ways of selecting a subset of bits that are enough to achieve a comparable accuracy, as when all the bits are used. The results showed that random bit selection is as good as other methods of smartly selecting the bits, like decorrelating the bits or choosing best performing independent bits. Another experiment was performed to examine if there a bit vector that could represent an entire object category, which would help in the prediction of category label of a test image. Instead of making comparison to each training image, now a hamming distance could be computed to just a vector that represents the whole category. This was accomplished using average bit vector which deduced by averaging the bit vectors of all the training images. This gave a bit vector with weights attached to each bit. Using average bit vector gave about the same classification accuracy as compared to when hamming distance is computed to each training image.

5.1. Future work

The extension of RIDMBC into the domain of object recognition using deep neural nets has a lot of scope in achieving better accuracy and faster runtime. Some components of the algorithm can be compressed and modified to obtain better classification results.

The way the features are extracted and choice of classifiers opens a lot of possibilities to improve the results. We have used PCA for dimensionality reduction of feature vectors, but

using methods like Linear Discriminant Analysis(LDA) which discriminate among classes can be useful.

We discussed a few methods of bit selection where we tried to pick bits smartly and randomly but we could not achieve the achieve the highest possible accuracy obtainable. So, there is a scope of coming up with a way of selecting a subset of bits that can achieve the maximum accuracy possible. Also, availability of faster computation resources can help in exhausting more bit selection strategies.

We came up with an average bit vector that helps us producing a bit code with weights obtained by averaging of bit vectors of training images. So, there is scope of coming up with a smarter way of assigning weights to the bit vector that represents a bit vector for a particular category of images.

Bibliography

[1] H. Zhang, J. R. Beveridge, Q. Mo, B. A. Draper, and P. J. Phillips, "Randomized intraclass-distance minimizing binary codes for face recognition," *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, sep 2014.

[2] Gudivada, V.N, Raghavan, and V.V., "Content based image retrieval systems," *Computer*, vol. 28, pp. 18–22, sep 1995.

[3] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, p. 817824, sep 2011.

[4] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," *Machine Learning for Computer Vision*, p. 4987, sep 2013.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015.

[6] J. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1904–1916, Sep 2015.

[8] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Given, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, and K. W. Bowyer, "The challenge of face recognition from digital point-and-shoot cameras," *Biometrics: Theory, Applications and Systems*

(BTAS), 2013 IEEE Sixth International Conference on, pp. 1–8, 2013.