CS554 Project Final Report
Content Based Image Retrieval

Doruk Çakmakçı
Computer Engineering Department
Bilkent University
Ankara, Turkey 06800
ID: 21502293

Kerem Ayöz
Computer Engineering Department
Bilkent University
Ankara, Turkey 06800
ID: 21501569

Abstract*—Image retrieval task has been a major research area of computer vision practitioners since late 20th century. Starting with the MIT's micro computer-based image retrieval system, researchers investigated how to retrieve images of interest effectively. Abundance of large image databases paved the way for content-based approaches instead of meta-data approaches for image retrieval task. From a similar perspective, the development in storage technologies and computational environment made deep learning approaches for content-based image retrieval. In this work, we try and investigate a selection of deep learning and traditional computer vision approaches on content-based image retrieval task. We have tested the approaches on CIFAR-10 and Oxford Buildings datasets. From our experiments we have concluded that Siamese Neural Network and Haar with Cosing Locality Sensitive Hashing perform best on CIFAR-10 and Oxford Buildings datasets respectively. Source codes to our model can be found at: *github.com/dorukcakmakci/cbir

Introduction

Image retrieval has been a major research area for computer vision researhers due to the availability of large scale image datasets almost anywhere. The task is first addressed from the perspective of searching based on image meta-data such as tags and labels. Even though, this approach is adopted for a long time, the need for manual labeling and tagging image data deteriorated its existence. For eliminating the labor required for using meta-data based approach, researchers proposed a content-based approach which utilizes different types of features (e.g. edges, contours colors gradients etc). Content-based image retrieval approach has born due to the abundance of large image databases, storage technologies and computational power.

As briefly mentioned above, content-based image retrieval (CBIR) is an application of computer vision techniques to the image retrieval problem [1]. This approach utilizes the raw image itself using methods and models based on computer vision, image analysis and deep learning. Content-based image retrieval, compared to the meta-data based approaches, removes the need for image labels using a notion of image distance measure on the domain of image representations (e.g. raw pixel values, embeddings) at a cost of robustness. Adversarial attacks may affect performance of a CBIR system, while being indistinguishable to an human eye.

With the availability of more computational power, using deep learning became feasible on image data. Deep learning is suitable for content-based image retrieval task since models are capable of encoding images to a sophisticated embedding domain in which images are more separable than raw pixel domains. Models maybe trained in a end-to-end manner for producing similarity between two images (e.g. siamese neural networks) or for encoding images to a lower dimensional domain (e.g. autoencoder). Embeddings (or feature extractions) are then matched between most probable

In this project, our aim was to compare the effectiveness of different feature extraction approaches, feature matching approaches on content-based image retrieval task. We have used a wide array of methods focusing on different types of information present in the images (e.g. texture, gradient, color and semantic information). On the other hand we investigated on various distance metrics to assess their feasibility on different types of features extracted. Furthermore, we trained a siamese neural network to perform content-based image retrieval task in an end-to-end manner. Methods are experimented on datasets of different backgrounds such as general purpose (i.e. CIFAR10 [2], CIFAR100 [2] and Caltech101 [3]) and task specific (i.e. Oxford Buildings Datasets [4]). From our preliminary results, we inferred that when the number of categories in the dataset is increased, the systems perform worse and worse. Therefore we have not included the preliminary results on CIFAR100 and Caltech101 datasets which contains images from 100 and 101 categories respectively. We narrowed our experiments to the datasets with images from less number of categories (i.e. CIFAR10 and Oxford Buildings Dataset).

Feature Extraction Methods

Denoising Autoencoder

Denoising autoencoder (DAE) is a neural network that utilizes unsupervised learning to learn robust features from the dataset it is trained on. Instead of reconstructing the exact version of input, a noise (i.e. gaussian noise in our setup) is added to the inputs for increasing the robustness of the features extracted by the model. The noise added to the normalized training images can be seen in Equation (1). f_{scale} is the scaling factor for the added gaussian noise (i.e. 1 in our implementation). After addition of noise, resulting images are clipped between 0 and 1. We utilized convolutions in our autoencoder architecture given the well established application of convolutions on image domain. The architecture can be seen in Figure 18 (see Appendix C). The process of extracting features from input images can be seen in Figure

$$I_{train} = I_{train} + f_{scale} \cdot N(0, 0.1) \tag{1}$$

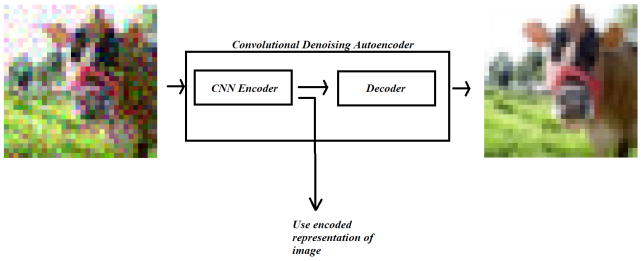


Figure 1: Feature extraction process when denoising autoencoder is used

VGG-16

We used transfer learning with VGG-16 pretrained on ImageNet [5] dataset to extract features from images. Preprocessing steps include: (i) Resizing each image to $224 \times 224 \times 3$ with antialiasing; (ii) min-max normalization of images; and (iii) changing channel format RGB to BGR.

Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) is a feature descriptor commonly used in various computer vision applications such as face detection (i.e. more broadly, object detection) and pedestrian detection (i.e. Dalal Triggs detector [6]). HOG, counts the occurrences of different gradient orientations in localized parts of the image (i.e. a dense grid of uniformly spaced cells) [7]. We used 8 orientations and 14×14 cells in our analysis. Figure 2 shows an example image and corresponding HOG feature extraction.

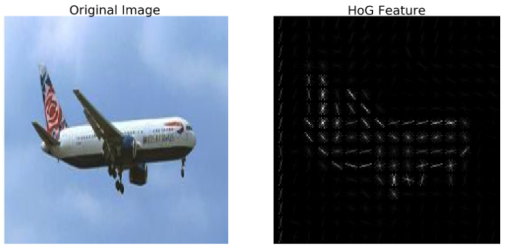


Figure 2: Example of HoG feature extraction.

Gabor Features

Gabor filter based feature extraction applies a bank of filters to the input to analyze the images textures [8]. Gabor filters are designed to activate on images having a specific frequency content in a predetermined orientation of interest [9]. These linear filters The bank of filters a contains various filters with different orientations and sizes. Use of different types of filters in the filter bank accounts for the variety of texture present in the images. Gabor filters process the image locally for every region and does not utilize global features allowing semantic representations extracted from image. We used 8 filters with 4 different orientations (i.e. π, $\frac{\pi}{2}$, $\frac{\pi}{4}$ and $\frac{\pi}{16}$) and 2 different scales (i.e. 5, 10). An example of features extracted by this method and used gabor filter bank can be seen in Figure 3.

Figure 3: Example of Gabor feature extraction.

Sobel Edge Features

This feature extraction method is based on edge detection by using Sobel operator. The pipeline for this method is as follows: First, the input image is blurred with gaussian kernel of size 5×5 and converted to grayscale, see Eq. (2) and Eq. (3). Then Sobel kernels are used to compute gradients along x and y dimensions of the image. Gradients along both directions are averaged to find the edge image, see Eq. (4). Finally, edge intensity image is thresholded to discard weaker edges, see Eq. (5). A sample image and its sobel features can be seen in Figure 4.

$$I_{noisy} = \omega * I \tag{2}$$

$$I_{grayscale} = f(I_{noisy}) \tag{3}$$

$$I_{edge} = \frac{1}{2} \cdot I_x + \frac{1}{2} \cdot I_y \tag{4}$$

$$I_{features} = I_{edge} \geq 120 \tag{5}$$

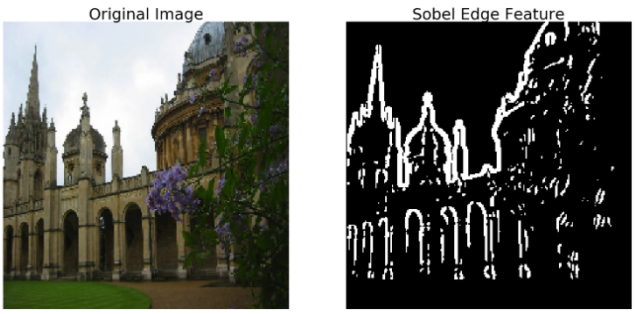


Figure 4: Example of Sobel feature extraction.

Local Binary Patterns

Local binary pattern (LBP) provides a simple way to extract texture analysis based features by considering each pixel in within its local neighborhood. This approach is dependent to the following parameters: (i) Radius for buildng circular local binary patterns; (ii) Number of sample points to build the neighborhood; (iii) Number of cells in horizontal and vertical directions. First an input image is divided to cells. Then each pixel in the cell is compared with its neighbors to produce a binary number by concatenating labels assigned to neighbors. A histogram (i.e. feature vector) is generated using the computed numbers [10]. Since local neighborhood provides an important information for discriminating between edges and corners, we can state that this method is feasible for discriminating between these structures, see Figure 5.

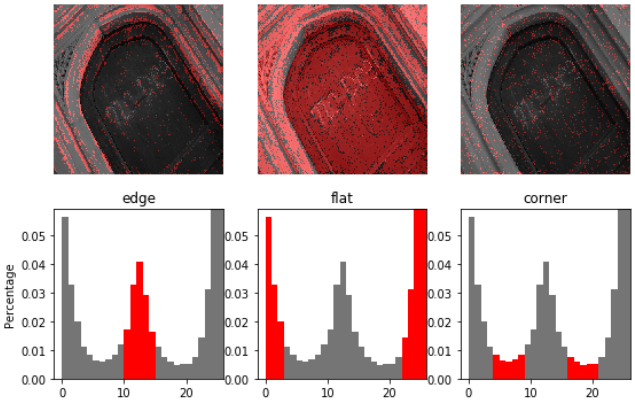


Figure 5: Example of LBP feature extraction.

Haar Like Features

Haar-like features approach owes its name to their intuitive similarity to Haar wavelets [11]. We used, the following types of Haar-like features: type-2-x, type-2-y, type-3-x, type-3-y and type-4. These filters are visualized in Figure 6.

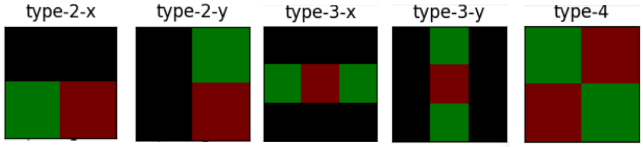


Figure 6: A visualization of Haar-features types used in our analysis

Color Histogram

Color is one of the easiest ways to discriminate between images. Many objects can be separable with respect to their color. However, color information most likely will not provide a good feature space for dataset with intra-class variation. RGB color space is used for generating color histograms. However different color spaces (i.e. Lab) may account for different information in the images such as luminosity. Thus, the color space used should be selected in a task-specific manner. A sample image and corresponding color histogram is shown in Figure 7.

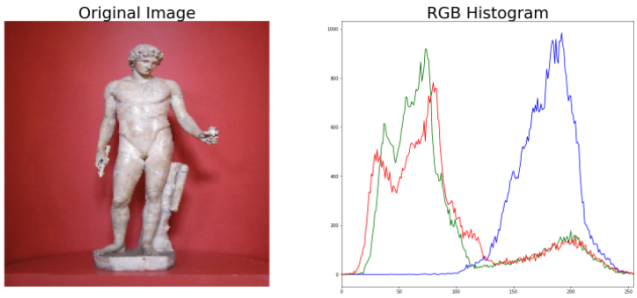


Figure 7: Example of RGB color histogram feature extraction.

Combinations of features

Each feature extraction mentioned above convey different information regarding the images. The information utilized by the approaches are below:

- **HOG**: Gradient based features
- **Gabor and LBP**: Texture analysis
- **VGG-16 and DAE**: Learned features
- **Sobel**: Edge information
- **Haar-like Features**: Edge and Texture information
- **Color Histogram**: Color information

Given this diverse types of features, we investigated the feasibility of combinations of these features. We first normalize each feature according to the training dataset then concatenate the resulting vectors. Furthermore we apply PCA to the concatenated feature vector to: (i) investigate main direction of change; and (ii) reduce dimensionality. We represent each feature vector according to principal components and use these as final feature vectors. We used only the following subset of extracted features to maintain complexity of our approach: HoG, Gabor, LBP, Sobel, Haar and Color Histogram features.

Similarity Measurement Methods

One of the challanges in the content based image retrieval systems is developing a fast and accurate query response mechanisms. Most of these systems contains large number of items in their databases and finding the most related items can be intractable even it has linear time complexity due to large number of items. Therefore, instead of comparing the queried item with each and every element in the database, hashing-based probabilistic algorithms are used. Also, emergence of deep learning methods also contributed to this area to get fast and accurate query responses.

We decided to use Locality Sensitive Hashing algorithms and Siamese Networks to measure the similarity between database items and queried item in a fast and accurate manner.

Locality Sensitive Hashing

Brute-force approach for finding similar image pairs has time complexity $O(n^2)$ when number of images in the dataset is n. When the dataset size is huge, brute-force approach to this problem may not be efficient. Locality Sensitive Hashing techniques aim to do this task in linear time, which means these algorithms have time complexity $O(n)$. In our project we decided to use Locality Sensitive Hashing in order to find similar images based on their feature embeddings obtained from various deep neural network models, namely denoising autoencoder, Siamese networks and image classification networks. Since we have large number of images in our dataset, linear-time algorithms will be quite useful for our project.

We implemented Locality Sensitive Hashing for three different distance metrics; Min Hashing for Jaccard distance, Random Hyperplanes for Cosine distance and Random Projections for Euclidean distance.

Min Hashing.

Min Hashing is the name of the technique which is used to find similar pairs of items based on Jaccard distance between them. Jaccard similarity between two objects A and B is defined as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Min Hashing is the probabilistic way of calculating Jaccard similarity between multiple items in linear-time. In this method, feature embedding of the items should be in binary form.

In the first step, we shuffle rows of binary features of all items. In that step we shuffle items' rows together therefore, feature correspondence information is protected. Then we get the index of occurrence of the first feature which is 1 for all items. This indices produced by single shuffle form a vector which is an item of what is called signature matrix. When we do this process m times, we obtain a signature matrix sized mxn which has m latent features for n items. This signature matrix represent transformed features of the items.

After obtaining signature matrix, we divide this matrix's rows into bands. This bands have k number of rows, so we have $\frac{m}{k}$ number of bands for each item. We declare two images as a candidate pair if at least t of their bands are identical. Here m, k and t are hyper parameters of the algorithm.

We aim to use this algoritm with the binarized neuron activations in our project. We will use feature embeddings from hidden layers of denoising autoencoder and image classification network. Neurons in these networks represent latent features and some of these neurons might be fired, means have a positive activation value, or not fired, means negative or zero activation value. We can binarize these feature embeddings according to the states of the neurons whether they are fired or not fired for particular input image. Then we will use these binarized feature embeddings in the Min Hashing algorithm.

Random Hyperplanes.

In order to calculate Cosine similarity between two items in an efficient way, a probabilistic method called random hyperplanes is used. In this method, feature points are compared with randomly generated hyperplanes to generate binary signature matrix. Then Min Hashing technqiue is applied to this binary signature matrix to find similar items.

In the first step, a plane is generated randomly in feature space. This plane divides the plane into two areas. For each item, we determine whether an item belongs to the first or second area. We apply dot product to find the area that the point belongs to. Then we obtain a binary value for each item. If we repeat this process m times, we obtain a mxn signature matrix. Then Min Hashing technique is applied to find similar item pairs. We aim to apply this method on original neuron activations of the images given into deep models.

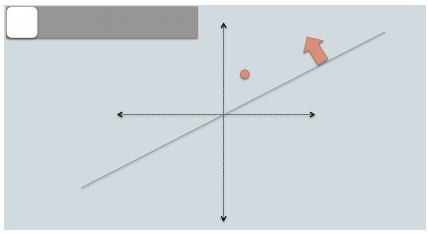


Figure 8: Example of single iteration. Point belongs to the upper area of the generated hyperplane.

Random Projections.

Random projection method is similar to the Random Hyperplanes method. Instead of generating hyperplanes, this time we generate m random lines in feature space. Then we project each item point onto this randomly generated lines. We divide the line into b sized buckets. Feature points that are in the same bucket at least t times are declared as a candidate pair.

Figure 9: Example of projecting two points on to a line. In the first projection, they do not coincide on the same bucket whereas in the second projection, they fall into same bucket.

Content-based Image Retrieval System Using LSH.

Each feature extraction system is combined with locality sensitive hashing method for measuring similarities between training dataset and query images. System figure for this approach can be seen in Figure 10. First, all training dataset images and the query image are fed to feature extraction method under consideration. Then, locality sensitive hashing is used between extracted features to find the most similar images. Similar images can be sorted with respect to the distance measure used.

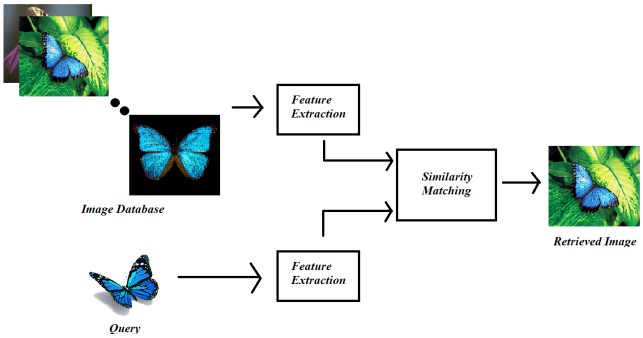


Figure 10: Content-based image retrieval system combining feature extraction methods and locality sensitive hashing

Siamese Network

Instead of extracting features by some method and matching similar images with another, we investigated the effectiveness of combining both processes via Siamese neural networks. Our siamese neural network structure can be seen in Figure 11. CNN networks have shared weights. The network is trained on a set of image pairs. Contrastive loss is used for model training. Euclidean distance is used for similarity measure between extracted features. The content based image retrieval conditioned on the network trained can be seen in Figure 12. We have trained the model from scratch however, as a more robust approach, the sister CNN networks could be VGG-16 networks pretrained on ImageNet dataset. The sister networks can be fine-tuned on this task. The details of the architecture can be seen in Figure

Experimental Setup

All proposed methodology is benchmarked on CIFAR10 and Oxford Buildings datasets. CIFAR10 dataset contains

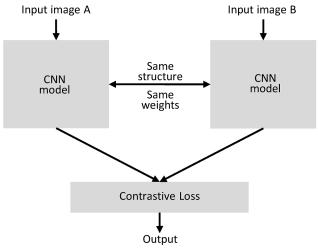


Figure 11: Siamese neural network structure

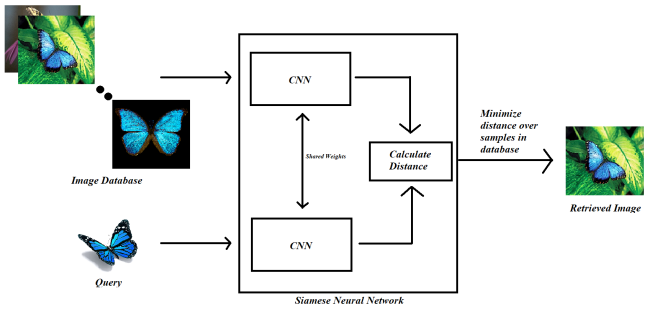


Figure 12: Content based image retrieval system using the trained siamese neural network

$50,000$ training and $10,000$ test images of size $32 \times 32 \times 3$. Images belong to one of 10 classes. Oxford Buildings dataset contains 5063 images. The dataset is shuffled and split to 4050 training and 1013 test images (i.e. $\sim 20\%$ of the data is separated as test data splits are stratified in with respect to class label) of various size. There are 17 different image categories in Oxford Buildings Dataset. Image sizes are set to $224 \times 224 \times 3$ with antialiasing before applying any of the explained methodology. The preprocessing steps for images differ with respect to the used method. Thus preprocessing steps are explained in within the section of each methodology.

All considered deep learning models are implemented in Python language using Keras framework. All models are trained and tested on a Dell XPS 9570 with an 8th-gen Intel Core i7-8750H processor, 16GB RAM and an Nvidia GeForce GTX 1050 Ti Max-Q GPU(4GB). All considered traditional computer vision based approaches are implemented in Python language. LSH algorithms and feature extraction methods were run on a server with 72-core Intel Xeon processor and 256GB RAM. A detailed explanation about task division per frameworks and libraries see Appendix A. Also for a division of project work among teammates, see Appendix B.

Results

We performed a total of 44 experiments on CIFAR10 and Oxford Building Datasets. Our experiments were designed to measure the effectiveness of different feature extraction methods and different distance metrics on content-based image retrieval task. In particular, we experimented with

jaccard, cosine and euclidean distance metrics, in the context of locality sensitive hashing, for each feature extraction method explained in Feature Extraction Methods section. We used mean average precision metric to assess our content based image retrieval systems' performance.

Locality Sensitive Hashing Method

We run three different LSH algorithms with all of the extracted features possible. Jaccard LSH requires features to be binary therefore we could run Jaccard LSH with Sobel, VGG-16 and Denoising Autoencoder features. We binarized the VGG-16 and Denoising AE features by assigning value 1 to the fired neurons, neuron with the positive activation and value 0 to non-fired neurons, neuron with zero or negative activation. Since there are no specific heuristics, we experimentally tuned the hyperparameters of the LSH algorithms by comparing the number of candidates and mAP values between distinct runs.

mAP is calculated from the top-10 candidates of the query image. Quality of the match was decided based on collusions in the hash tables in all of the algorithms; image with the most collusion with query image is the best candidate match for that specific query. Then mAP is calculcated by taking the average of these top-10 precision values. Results of the experiments with Oxford Buildings and CIFAR-10 datasets are shown in Table 1 and Table 2 respectively.

	Cosine LSH	**Euclidean LSH**	**Jaccard LSH**
HoG	0.13	0.15	-
Sobel	0.18	0.14	0.20
Color Histogram	0.15	0.18	-
Gabor Filter	0.15	0.14	-
LBP	0.18	0.16	-
Haar	0.24	0.14	-
PCA Mixture	0.15	0.14	-
VGG-16	0.15	0.14	0.13
Denoising AE	0.14	0.14	0.14

TABLE 1: Mean average precision results of LSH-based similarity matching system with different feature extraction technqiues on Oxford Buildings dataset.

	Cosine LSH	**Euclidean LSH**	**Jaccard LSH**
HoG	0.23	0.16	-
Sobel	0.1	0.12	0.13
Color Histogram	0.17	0.18	-
Gabor Filter	0.14	0.14	-
LBP	0.16	0.14	-
Haar	0.14	0.13	-
PCA Mixture	0.16	0.12	-
VGG-16	0.14	0.15	0.21
Denoising AE	0.20	0.25	0.16

TABLE 2: Mean average precision results of LSH-based similarity matching system with different feature extraction techniques on CIFAR-10 dataset.

Siamese Network Method

Siamese convolutional neural networks are used to perform content-based image retrieval task as a standalone method. We trained models on each dataset and tested on a subsample of test dataset due to computational resource limitations. For each dataset we randomly sampled 10 images per category (i.e. in total 100 images for CIFAR100 and 170 images for Oxford Buildings Dataset). We had to subsample test datasets since using all test data needed 5 days and 7 days for CIFAR100 and Oxford Buildings dataset respectively. Results can be seen in Table 3

	Oxford Buildings Dataset	CIFAR10
mAP@1	0.12	0.46
mAP@3	0.09	0.43
mAP@10	0.07	0.41

TABLE 3: Mean average precision results of Siamese neural network based system

Discussion

During our experiments, we inferred that content based image retrieval task gets harder when the number of categories in the dataset is increased since label information is used to measure similarity. Therefore we eliminated the CIFAR-100 and Caltech101 datasets at the inital stages of the project.

In the LSH-based similarity matching experiments, we faced with two main problems; (i) finding the best set of hyperparameters is difficult (ii) feature extraction methods may fail because of noise in the data. It is faster approach rather than comparing query image with all of the database images however, in order to increase the performance, it needs a lot of effort to tune both the feature extraction methods and LSH algorithms. Even CIFAR-10 dataset performed well with deep learning methods, we fail to achieve a decent performance with traditional methods since image sizes are very small. Filter based feature extraction methods (Gabor, Sobel, Haar) works much better with Oxford Buildings dataset despite the fact that it contains more possible labels since image sizes are much bigger than CIFAR-10 dataset. Haar like feature extraction technique is the best performing method for Oxford Buildings dataset with 0.24 mAP value.

Histogram based feature extraction methods (Color Histogram, LBP, HoG) performed well when we compare their ease of implementation with others. When the images in the dataset are simple, low resolution or less noisy, these methods are much more preferable because they are fast and perform better than filter-based approaches. However, when data gets complicated they also begin to underperform, filter based approaches become better in such case. Also when we applied Principal Component Analysis (PCA) to the vector containing all the extracted features for images, we could not obtained any decent performance. This might be caused by contradictions between the extracted features. For instance one extracted feature set imply that query image is type of

class dog while other extracted feature set implying class automobile.

Deep learning based feature extraction methods are the best performing feature extraction method for CIFAR-10 dataset but not for Oxford Buildings dataset. The main reason is that labels in the CIFAR-10 are more distinct than Oxford Buildings dataset. Since we use label information as ground truth for similarity, VGG-16 features worked well since it is trained to separate the images with different categories. All of the images in Oxford Buildings dataset are buildings of various types whereas CIFAR-10 image categories are much more distinguishable (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). A more complex model could be trained to extract the information that separates the buildings specifically, however our models were more generic types of deep neural networks.

From results of our experiments we conclude that Siamese neural network approach performed the best for content based image retrieval task on CIFAR-10 dataset in terms of mean average precision. We anticipated this result before performing experiments since siamese network is designed to perform matching between similar inputs in and end-to-end manner. In other words, the network not only learns feature extraction, it optimizes itself such that the distance between images of belonging to same class is decreased and the distance between samples of different classes are increased. Compared to other content based image retrieval methods discussed in this work, siamese network learns to eliminate bias due to intra-class and inter-class variance in the dataset. However, siamese network is not able to reach results of state-of-the-art since these methods are much more sophisticated in terms of model design.

Appendix A (Use of Software Libraries)

Python is the only programming language used for this project. Additionally, Keras Functional API is used for deep learning based approaches. DAE model idea is based on this article [12]. Siamese neural network is based on the Mnist siamese implementation on keras documentations [13]. The version on this documentation uses fully connected network as sister networks however, our version uses convolutional neural networks. We used Numpy and Collections libraries to implement Locality Sensitive Hashing algorithms. Feature extraction methods are used from Sklearn library and customized with additional features using OpenCV library.

Appendix B (Contributions)

Contributions of each group member is given below:

- **Doruk Çakmakçı**: Helped designing overall project structure. Designed DAE and Siamese neural network models, Applied transfer learning with VGG16. Contributed to reports and presentations. Performed experiments regarding deep learning approaches.
- **Kerem Ayöz**: Helped designing overall project structure. Implemented Locality sensitive hashing. Extracted features with Gabor, Haar-like features, Color histograms, HOG, LBP and Sobel. Performed experiments with LSH. Contributed to reports and presentations.

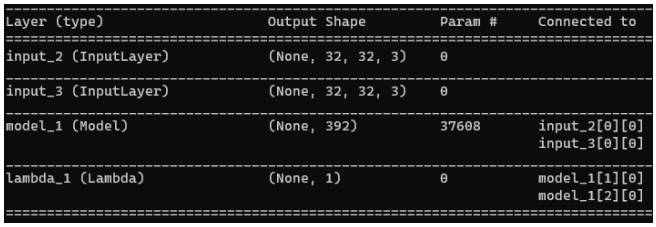
Appendix C (Supplementary Information)

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d_1 (Conv2D)	(None, 32, 32, 8)	392
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 8)	0
conv2d_2 (Conv2D)	(None, 16, 16, 16)	2064
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 16)	0
conv2d_3 (Conv2D)	(None, 8, 8, 32)	8224
encoder (MaxPooling2D)	(None, 4, 4, 32)	0
conv2d_4 (Conv2D)	(None, 4, 4, 32)	16416
up_sampling2d_1 (UpSampling2	(None, 8, 8, 32)	0
conv2d_5 (Conv2D)	(None, 8, 8, 16)	8208
up_sampling2d_2 (UpSampling2	(None, 16, 16, 16)	0
conv2d_6 (Conv2D)	(None, 16, 16, 8)	2056
up_sampling2d_3 (UpSampling2	(None, 32, 32, 8)	0
conv2d_7 (Conv2D)	(None, 32, 32, 3)	387

Figure 13: Denoising convolutional autoencoder architecture used to extract features from images

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d_1 (Conv2D)	(None, 30, 30, 64)	1792
conv2d_2 (Conv2D)	(None, 28, 28, 32)	18464
conv2d_3 (Conv2D)	(None, 26, 26, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 16)	4624
conv2d_5 (Conv2D)	(None, 9, 9, 16)	2320
conv2d_6 (Conv2D)	(None, 7, 7, 8)	1160
flatten_1 (Flatten)	(None, 392)	0

Figure 14: One of the sister networks in Siamese model



Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 32, 32, 3)	0	
input_3 (InputLayer)	(None, 32, 32, 3)	0	
model_1 (Model)	(None, 392)	37608	input_2[0][0] input_3[0][0]
lambda_1 (Lambda)	(None, 1)	0	model_1[1][0] model_1[2][0]

Figure 15: Siamese neural network architecture used

Cosine LSH	Hyperplane	Signature Size	R	B
HoG	1000	512	8	64
Sobel	1000	512	8	64
Color Histogram	768	512	8	64
Gabor	2000	4000	100	40
Local Binary Pattern	200	1024	16	64
Haar	500	2000	25	80
Mixture with PCA	1000	5000	100	50
VGG16	500	600	3	20
Denoising Autoencoder	250	225	15	15

Figure 16: Parameters used with Cosine LSH algorithm.

Euclidean LSH	Lines	Bucket Size
HoG	500	0.01
Sobel	100	0.05
Color Histogram	5000	10
Gabor	2000	2
Local Binary Pattern	100	0.001
Haar	2000	1
Mixture with PCA	1000	0.01
VGG16	500	0.01
Denoising Autoencoder	500	2

Figure 17: Parameters used with Euclidean LSH algorithm.

Jaccard LSH	Signature Size	R	B
Sobel	1050	7	150
Denoising Autoencoder	1050	7	150
VGG16	1050	7	150

Figure 18: Parameters used with Jaccard LSH algorithm.

References

[1] "Content-based image retrieval," Mar 2020. [Online]. Available: https://en.wikipedia.org/wiki/Content-based_image_retrieval

[2] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[3] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 conference on computer vision and pattern recognition workshop*. IEEE, 2004, pp. 178–178.

[4] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[5] O. Russakovsky, 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, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[7] "Histogram of oriented gradients," May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[8] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," in *1990 IEEE international conference on systems, man, and cybernetics conference proceedings*. IEEE, 1990, pp. 14–19.

[9] "Gabor filter," May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Gabor_filter

[10] "Local binary patterns," May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Local_binary_patterns

[11] "Haar-like feature," May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Haar-like_feature

[12] Andrey, W.-C. Adrian Rosebrock, Cheng, Mingxing, Johnny, and Sumit, "Denoising autoencoders with keras, tensorflow, and deep learning," Apr 2020. [Online]. Available: https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/

[13] "Mnist siamese." [Online]. Available: https://keras.io/examples/mnist_siamese/