Zero Shot Sketch Based Information Retrieval

Harsh Thakur
160279

Mayank Sharma
160392

Shreyaansh Gupta
160664

Utkarsh Thukral
160760

Abstract

In this project, we aim to study and implement some state-of-the-art Sketch based Information Retrieval methods and extend them to the zero-shot setting. We then look at some recently developed methods used for the zero-shot SBIR and try to implement those. We particularly try Deep Sketch hashing, extending to the zero-shot setting using appropriate train-test split on the Dataset. We use a benchmark TU Berlin sketch-image database to evaluate and report our results on the same.

Disclaimer: The work carried out in the project undertaken by Group 55 has not been re-used from any of our existing/past projects that we may have done in the past or currently doing. The work is not copied from any other projects done at IIT Kanpur or elsewhere. All the projects or papers that we have used in our project have been sequentially listed in the References Section

1 Problem Introduction and Motivation

An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. The techniques used for retrieving image on the basis of content, the content as text, sketch, color and shape that can describe the image. Queries are often expressed as keywords, requiring the images to be tagged. In view of the ever increasing size of image databases, having an appropriate and complete set of tags requires too much effort and is also prone to human biasing, and thus content-based search techniques become vital. Nowadays, searching desired image by describing hand-drawn sketch is popular with the emerging of touch screen technology.
Sketch-based image retrieval (SBIR) is the task of retrieving images from a natural image database that correspond to a given hand-drawn sketch, which makes for a particularly interesting problem in Computer Vision and Machine Learning. However, SBIR is challenging since freehand sketches only focus on the salient object structures and are unintelligible as such. Also, the shapes and scales in sketches are usually distorted compared to natural images. Another challenge is the large intra-class variance present in sketches due to the fact that humans tend to draw sketches with varied levels of abstraction.
Ideally, an SBIR model should learn to associate components in the sketch (say, feet, tail, etc.) with the corresponding components in the image having similar shape characteristics. However, the traditional methods simply focus on retrieving images which belong to the same class as the sketch but not necessarily having the same shape characteristics as in the sketch. As a result, existing methods simply learn to associate sketches with classes seen during training and hence fail to generalize to unseen classes (the zero-shot setting).

In a real-life scenario, there is no guarantee that the training data categories cover all concepts of potential retrieval queries and candidates in the database. An extreme case occurs when test data are subjected to an absolutely different set of classes, excluding the trained categories. This motivates

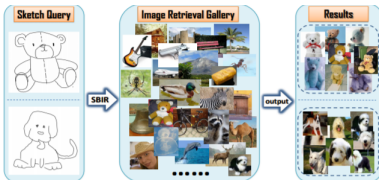


Figure 1: Sketch-based Image Retrieval from Database [1]

us towards a zero-shot setting for the problem where the model is evaluated on images and sketches from some classes which were not present during training.

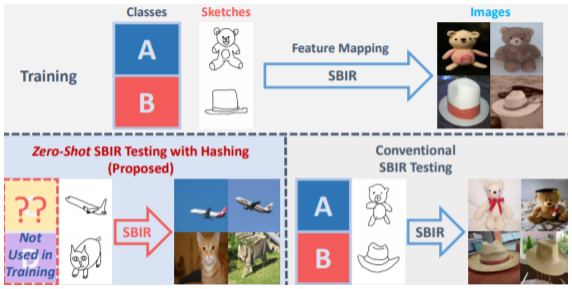


Figure 2: Zero Shot Sketch Based Image Retrieval

2 Dataset Used

We are using the benchmark TU Berlin Dataset [2]. The necessary data is acquired in a controlled user study where subjects rate how well given sketch/image pairs match.
The dataset contains 31 benchmark sketches corresponding to 31 classes (1 sketch per class) and 40 natural images per class. All the reported results are given on this dataset. Some examples from the given dataset:

To set-up the zero-shot setting for this problem, we partitioned this dataset into 25 and 6 classes.

For zero-shot, we use these 25 classes of sketches and images as the training set and the remaining 6 classes are used for testing. For partitioning of this dataset, we handpicked the 6 classes by manually observing the sketches and selecting the 6 sketches that stood out as totally different in the dataset.

3 Problem Formulation

We have data in the form

$$\mathcal{O} = \{\boldsymbol{I}, \boldsymbol{S}, \boldsymbol{Y}^I, \boldsymbol{Y}^S\}$$

where $\boldsymbol{I} = \{\boldsymbol{I}_i\}_{i=1}^{n_1}$ is the set of images $\boldsymbol{S} = \{\boldsymbol{S}_i\}_{i=1}^{n_2}$ is the set of sketches $\boldsymbol{Y}^I$ is an $\mathbb{R}^{C \times n_1}$ matrix consisting of sparse vectors $y_i^I \in \mathbb{R}^C$ such that $y_{ic}^I = 1$ if image $\boldsymbol{I}_i$ is in category c and 0 otherwise $\boldsymbol{Y}^S$ is an $\mathbb{R}^{C \times n_2}$ matrix consisting of sparse vectors $y_i^S \in \mathbb{R}^C$ such that $y_{ic}^S = 1$ if sketch $\boldsymbol{S}_i$ is in category c and 0 otherwise

The Objective: For SBIR, we get a test sketch $\boldsymbol{S}_t$ and we need to learn the data in such a way that we obtain the class c to which it belongs and return the images in that class (retrieval).
In case of zero-shot SBIR we assume a split in the data as $mathcalO = \mathcal{O}^s \cup \mathcal{O}^u$. Also, we have $\mathcal{O}^s \mathcal{O}^u = \phi$

4 Sketch Tokens

We use the algorithm provided by [3], to generate the sketch tokens for the images present in our dataset. We classify each image patch with a token label given a collection of low-level features including oriented gradient channels, color channels, and self-similarity channels. The token class assignments resulting from clustering the patches of hand drawn contours are used for training. The result is an efficient approach that can compute per-pixel token labelling in about one second per image.
We define a set of token classes that represent various local edge structures that may exist in an image. Let us say we have I images and the corresponding set of binary images S. We cluster patches s extracted from the binary images S. Given a set of token classes, we then learn a classifier that takes features which are color patches from the input image. For feature extraction, we compute multiple feature channels per image. We have 2 types of features, one that directly indexes into the channel and self similarity features. For classification we need an efficient algorithm since every pixel in the image is going to be classified. Also, the number of potential classes is large so we implement random forest method which works well given the 2 constraints. Contour detection only requires binary labeling of pixel contours. The random forest classifier predicts the probability that an image patch belongs to each token class or the negative set. Since each token has a contour located at its center, we can compute the probability of a contour at the center pixel using the sum of token probabilities. In summary, we learn tokens which is used for object and contour detection.

As in most Computer Vision tasks, feature extraction from the image is one of the most important tasks in SBIR. In fact, many state-of the art SBIR models essentially convert the image to a sketch using some feature transformation and carry out comparison between the image and sketch for matching. For implementation purposes, we use the generated sketch tokens wherever necessary (as part of **FEATURE EXTRACTION** from the images).

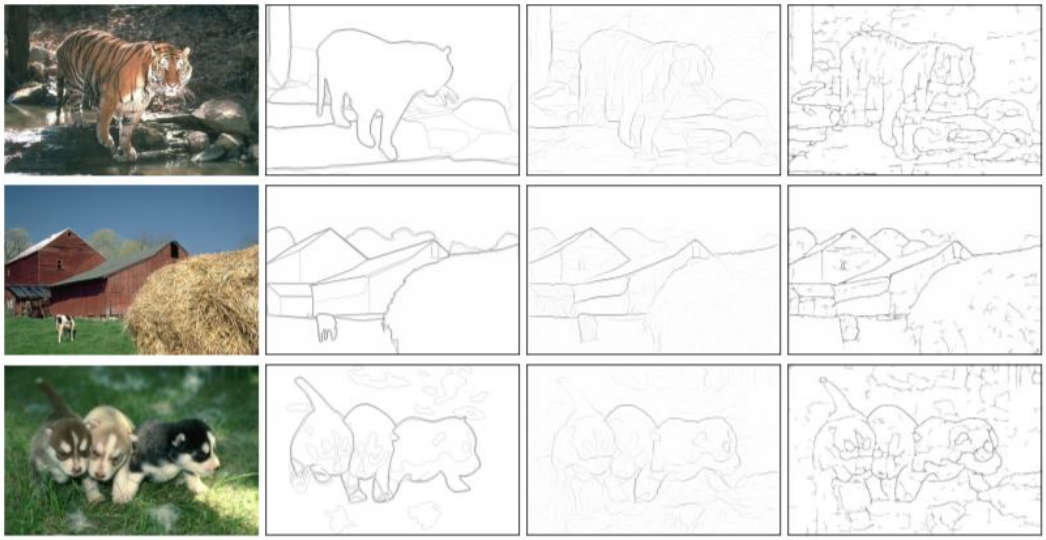


Figure 3: Generated Sketch tokens from images

5 Traditional Methods of SBIR

We studied multiple existing methods for Sketch Based Image Retrieval and have stated them below. We implemented the Principal Component Analysis among them, and extended it to the zero-shot setting to check how it generalizes. However, in the case of Zero shot setting, we got poor results. These methods work well for seen classes but fails when the test data includes images from unseen classes.

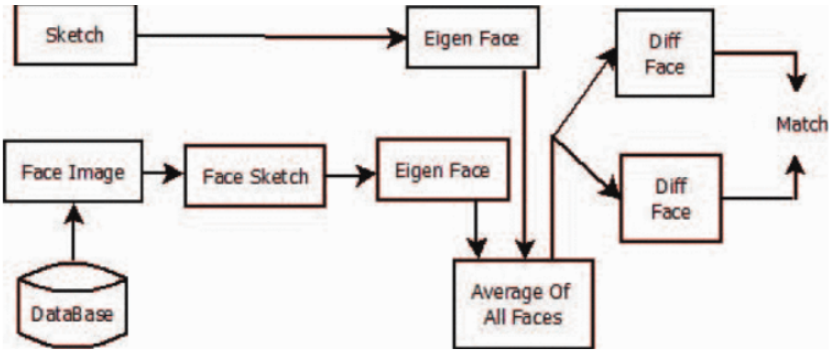
5.1 Principal Component Analysis (PCA)

During the training phase, we convert the images in our database to sketches and then based upon the intensities these sketches are further changed into eigenfaces [4]. Eigenfaces are the eigenvectors of the covariance matrix of the set of images and each image can be represented exactly by a linear combination of the Eigen faces. Eigenfaces can be approximated using the good eigenvectors with the largest Eigen values. We then compute the average of eigenfaces and calculate the eigenface of the given sketch. The eigenface of the given sketch is then mixed with the average of all the eigenfaces. The image is divided into overlapping patches and for each patch, we try to find a closest matching patch from the set of images in our database. We take the assumption that if 2 image patches are similar, then the sketch patches must also be similar. The maximum match image will be returned from the database. In PCA, the aim is to minimize the error in reconstructed data by finding a set of orthogonal components. We compute the vectors which describe the image distribution in the entire image space. Using PCA, we further compute the coefficients by projection of the input photos onto the target photos. We finally Calculate the target sketch by taking the linear combination of weights which are the projection coefficients.

5.2 Siamese Network

The Siamese network [5] maps both the sketches and images into a common space where the semantic distance is preserved.
Let (S, I, Y = 1) and (S, I, Y = 0) be the pairs of images and sketches that belong to same and different class respectively and D_θ(S, I) be the l2 distance between the image and sketch features where θ are the parameters of the mapping function. The loss function L(θ) for training is given by:

$$L(\theta) = Y\frac{1}{2}(D_\theta)^2 + (1-Y)\frac{1}{2}\{max(0, m - D_\theta)\}^2$$



Face recognition with eigen face.

The Siamese approach is much better suited for a fine-grained approach wherein the data is extremely sparse. Hence, using a method like PCA might over-fit for such data, but since we got to predict classes from sketches which vary greatly (much less fine-grained), we used a PCA based setting.

5.3 Multiscale Markov Random Field

This technique[4] is generally used when we need to work with local structures in different scales. The training set consists of photo-sketch pairs. An image region is divided into overlapping patches and for each patch, a similar patch is searched in the database. Apart from the patch-matching, the smoothness between the adjacent patches is also taken into account. The scale of the image structure depends on the patch size. The images are transformed into sketches using sketch synthesis algorithms and then a sketch is searched and transformed into a photo which is then matched with the images in the dataset. A major drawback for this method is that the image conditions in the dataset images needs to be the same as that of the given sketch.

5.4 Local Linear Embedding (LLE)

In this we generate a pseudo-sketch from a photo image which is being used in this scheme [4]. This method's main focus is on sketches of plain style. The weight combination of local nearest neighbours brings some blurs to the pseudo-sketches. It is widely used for nonlinear dimension reduction of high dimensional data and image analysis. The fundamental idea of LLE is to compute neighbour-preserving mapping between a high-dimensional original data space and a low-dimensional feature space, on the basis of simple geometric intuition that each data and its neighbours lie on or close to a locally patch.

6 Introduction to Zero Shot Setting

The major drawback because of which the traditional SBIR methods fail is that these methods simply focus only on retrieving images which belong to the same class as the sketch but not necessarily having the same shape characteristics as in the sketch.
The existing methods simply learn to associate sketches with classes seen during training and hence fail to generalize to unseen classes. This is why there is a need for Zero Shot Setting so that we can also train the data to work for unseen classes.
We studied some approaches to tackle this problem and tried to implement Deep Sketch-Image Hashing. The possible approaches and the one undertaken are summarised as follows:

6.1 Generative Probability Models: Variational Autoencoders

The Variational Autoencoders [5] map a prior distribution on a hidden latent variable $p(z)$ to the data distribution $p(x)$. The intractable posterior $p(z|x)$ is approximated by the variational distribution $q(z|x)$.
We assume that θ is the parameter for conditional distribution $p(x|z)$ and ϕ is the parameter for the variational distribution $q(x|z)$.

$$\implies p(x) \geq \mathcal{L}(\phi, \theta; \S)$$
$$= -D_{KL}(q_\phi(z|x)|p_\theta(z)) + \mathbb{E}_{q_\phi}(z|x)[logp_\theta(x|z)]$$

The bound comes out to be

$$\mathcal{L}(\phi, \theta; \boldsymbol{I}, \boldsymbol{S}) == -D_{KL}(q_\phi(z|\boldsymbol{I}, \boldsymbol{S})||p_\theta(z|\boldsymbol{S})) + \mathbb{E}[logp_\theta(\boldsymbol{I}|z, \boldsymbol{S})]$$

6.2 Generative Probability Models: Adversarial Encoders

Adversarial Autoencoders [5] are similar to Variational Autoencoders, but the KL divergence term is replaced by the adversarial training procedure.
Let E, D be the encoder and decoder respectively, the loss function for this is

$$\mathcal{L} == \mathbb{E}_z[logp_\theta(\boldsymbol{I}|z, \boldsymbol{S})] + \mathbb{E}_{\boldsymbol{I}}[log[1 - D(E(\boldsymbol{I})]]$$

7 Deep Sketch Image Hashing

Using our earlier notation, our aim [1] is to learn two sets of m-bit binary codes $\mathbf{B}^I = \{\mathbf{b}_i^I\}_{i=1}^{n_1} \in \{-1, 1\}^{m \times n_1}$ and $\mathbf{B}^S = \{\mathbf{b}_i^S\}_{j=1}^{n_2} \in \{-1, 1\}^{m \times n_2}$ for both $\mathcal{O}$

Intuitively, this means that given any natural image, we will get an initial sketch-token, with each pixel being assigned a score that how likely that point lies on a contour (i.e. the edge). We define the the cross-view similarity matrix of $\mathcal{O}_1$ and $\mathcal{O}_2$ as $\mathbf{W} \in^{n_1 \times n_2}$ where the element of $\mathbf{W}_{ij}$ denotes the cross-view similarity between $\{\mathbf{I}_i, \mathbf{Z}_i\}$ and $\mathbf{S}_j$. The loss function $\mathcal{J}_1$ and subsequently Deep Sketch Hashing problem is formulated as follows:

$$\min_{\mathbf{B}^I, \mathbf{B}^S} \mathcal{J}_1 := \mathbf{W} \odot m - (\mathbf{B}^I)^T \mathbf{B}^S{}_F^2,$$
$$\text{s.t. } \mathbf{B}^I \in \{-1, +1\}^{m \times n_1}, \mathbf{B}^S \in \{-1, +1\}^{m \times n_2}$$

where we have that, $\mathbf{W}_{ij} = 1$ if $\mathbf{y}_i^I = \mathbf{y}_j^S$ and -1 otherwise. Our loss function $\mathcal{J}_1$ minimizes the difference in natural images and their sketches, so that the ones belonging to the same category will be pulled as close as possible, and rest pushed far away.
The detailed model is given in the following figures:

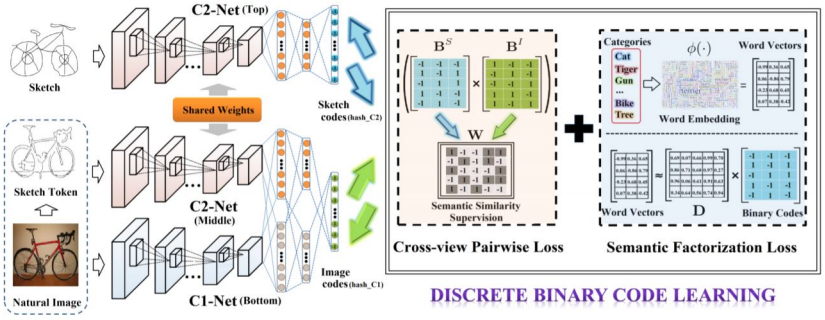


Figure 4: End to End structure [1]

The above figure shows the model proposed in [1], though our implementation of the algorithm is

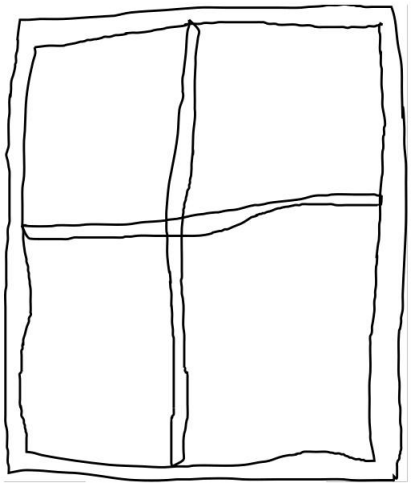
quite different.
The above requires a use of 2 convolutional neural networks which we implemented using the caffe framework:

Net	Layer	Kernel Size	Stride	Pad	Output
C1-Net (Natural Image)	input	-	-	-	3×227×227
	conv1	11×11	4	0	96×55×55
	pooling1	3×3	2	0	96×27×27
	conv2	5×5	1	2	256×27×27
	pooling2	3×3	2	0	256×13×13
	conv3	3×3	1	1	384×13×13
	conv4	3×3	1	1	384×13×13
	conv5	3×3	1	1	384×13×13
	pooling3	3×3	2	1	256×7×7
	fc_a	7×7	1	0	4096×1×1
	fc_b	1×1	1	0	1024×1×1
	hash_C1	1×1	1	0	m ×1×1
C2-Net (Free-hand sketch/ Sketch-tokens)	input	-	-	-	1×200×200
	conv1	14×14	3	0	64×63×63
	pooling1	3×3	2	0	64×31×31
	conv2_1	3×3	1	1	128×31×31
	conv2_2	3×3	1	1	128×31×31
	pooling2	3×3	2	0	128×15×15
	conv3_1	3×3	1	1	256×15×15
	conv3_2	3×3	1	1	256×15×15
	pooling3	3×3	2	0	256×7×7
	fc_a	7×7	1	0	4096×1×1
	fc_b	1×1	1	0	1024×1×1
	hash_C2	1×1	1	0	m ×1×1

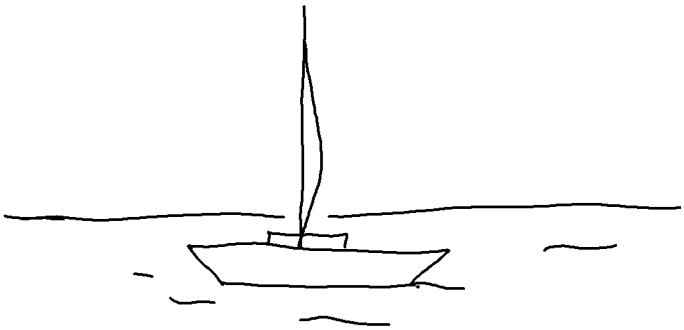
Figure 5: Description of the neural nets [1]

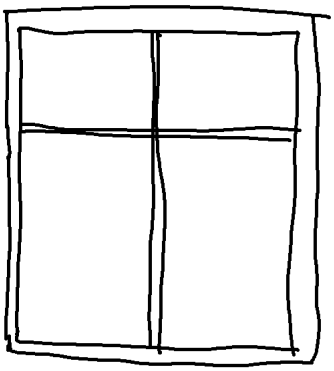
8 Experimental Results:

We did not use any evaluation metric on our framework. Instead we are manually evaluating how our framework performs on the following query sketches:



The sketches in our dataset corresponding to the these query sketches are:





We are evaluating our retrieval performances on these two query sketches. We return the top 5 matches given by different algorithms and report the outcomes.

PCA under normal setting:

PCA under zero-shot setting:

Deep Sketch Image Hashing under zero-shot setting:

We observe that PCA works great during SBIR, however it doesn't work well under Zero Shot setting. However, Deep Sketch Image Hashing gives more efficiency under Zero Shot Setting, thus giving us beter results.

Softwares/Tools used:

Majority of our code was written in Python and we use the following:

- **caffe**:
  This is a deep learning framework. We used caffe for our implementations of almost all of the (convolutional) neural networks.
- **tensorflow**:
  TensorFlow is an open-source software library for dataflow programming across a range of

tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

- **OpenCV**:
  OpenCV is a library of programming functions mainly aimed at real-time computer vision. We have used OpenCV to try various kinds of feature transformations on the images.

We also used Sketch Token Toolkit [3], the code of which was written in MATLAB. The code was modified for our implementation where our the images from our dataset were fed to it and the obtained sketch tokens were used.

Things we Learned

- We worked on the image dataset for the first time and learned how images are stored and how to manipulate the dataset and extract features and implement them according to our requirement.
- We structured the dataset accordingly and learned to feed various algorithms these datasets to get our output.
- We used code snippets on different platforms and modified them accordingly and implemented them according to our needs. Implementing PCA that we were taught in class also helped to gain a better understanding and fully internalize the algorithm.
- Breaking the big task of Zero shot Sketch Based Image Retrieval into smaller sub-problems and tackling each sub-problem provided exposure into Machine Learning implementation and Debugging each sub-problem easily helped to gain insights about various methods that we employed. Then we combined these sub-problems together to develop an end to end framework that solves the problem.
- Teamwork :)

Future Works

- We can improve on our feature extraction method. Currently we are relying on Sketch Token Toolkit [3] which is not specific for our retrieval task. So, we can further work upon finding a better feature transformation which could give more appropriate sketches from our image dataset.
- We can fine tune our model by adjusting various hyperparameters which could give better efficiency with the same model framework over a variety of datasets.
- We can also extend this algorithm to generate images from the sketches.

References

[1] Deep Sketch Hashing: Fast Free-hand Sketch-Based Image Retrieval [Li Liu1, Fumin Shen2, Yuming Shen1, Xianglong Liu3, and Ling Shao1]

[2] Benchmark dataset by TU berlin based on controlled user study `http://cybertron.cg.tu-berlin.de/eitz/tvcg_benchmark/index.html`

[3] Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection [Joseph J. Lim, C. Lawrence, Zitnick Piotr Dollar], pdollar@microsoft.com

[4] Survey on sketch based image retrieval methods [2016 International Conference on Circuit, Power and Computing Technologies [ICCPCT]]. M Indu,indu22.emp@gmail.com

[5] A Zero-Shot Framework for Sketch based Image Retrieval [Sasi Kiran Yelamarthi1 , Shiva Krishna Reddy1 , Ashish Mishra, and Anurag Mittal]