# Sketch Based Image Retrieval using Deep Learning based Machine Learning

Mr. Siddhant Mishra[1], Mr. Aryaan Silva[2], Mr. Aditya Garela[3], Mr. Vishal Patil[4]

[1,2,3,4]*Department of CSE MIT ADT University of Pune, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** Sketch based image retrieval (SBIR) is a sub-section of Content Based Image Retrieval (CBIR) where the user provides or can manually draw a drawing as an input in order to retrieve images relevant to the drawing given. The biggest challenge in SBIR is subject to how well a user can draw as it entirely relies on the user's ability to express information in hand-drawn form. Our project aims to enable detection and extraction of multiple sketches given together as a single input sketch image whereas many of the SBIR models created take only a single input as an image and give an output. The features are extracted from individual sketches obtained using VGG16 which is a deep learning architecture, and classified to its type based on supervised machine learning using Support Vector Machines. Based on the image class obtained, images are retrieved from the database using an OpenCV library, CVLib , which finds the objects present in a photo image. The model analyses the components in the class of images and based on a ranking system gives output of the image class. The model consisting of a combination of VGG16 and SVM provides 89% accuracy

***Key Words*: Sketch Based Image Retrieval, Machine Learning, VGG16, Support Vector Machines, CVLib**

## 1. INTRODUCTION

To obtain information about something, it's required to be ready to express it unambiguously and with relevance. Verbal descriptions, pictographs and sketches are one amongst the few ways to explain one's need for relevant information. With the appearance of technology, many aim to achieve knowledge through information remotely, by using search engines. This explicit description about the thing is additionally employed in search engines to induce relevant ends up in the shape of images, websites, videos etc. Information retrieval may be a domain of interest to several problem solvers where the most difficulty lies within the ability to bridge the gap between information available and also the query given by the user. A sub-domain under information retrieval is content based image retrieval. In Sketch Based Image Retrieval, in order to obtain images, the user draws the images to retrieve an image of the drawing, the images are mapped to particular tags that are of (SBIR) [1], a sketch or a drawing is the input given. For example, to retrieve images of an apple the user needs to draw an apple. These drawings can be freehand sketches, professional drawing, symbols and pictures defined by edges. A hand drawn sketch is just roughly drawn shapes and has a high possibility that it may not contain edges and features as sharply defined in the photos. Thus, relevance in the retrieved images depends on how efficiently a model can figure out and relate to the drawn image in order to give the correct retrieved image. Every user will have their own subjective way of expressing an image, it is the role of a good model to identify it regardless. Hence a wide range of database is require containing both high and low levels of abstraction in terms of features.

## 2. LITERATURE SURVEY

Ayan Kumar Bhunia, Yongxin Yang, and other scholar's address the most concern involving the user's ability to draw a whole sketch. A sketch is already a product of a user's subjectivity and skill in expressing a pictographic content. Thus, completion of a diagram also depends on such a user's skills in drawing a sketch. Hence the technique utilized in [2] could be a reinforcement learning-based cross-modal retrieval framework that learns from the strokes provided and identifies the category to retrieve images. These strokes are compared with photos to get images containing similar stroke patterns, or edges. This helped in retrieving images faster than in conventional SBIR where the user has to draw the sketch completely which takes more time. Most SBIR systems use singular sketches as input and retrieve images. Thus, to retrieve images containing many alternative classes of sketch, one cannot send one sketch image containing various sketches belonging to different classes. a mix of two texts and a couple of sketches are wont to provide input for multiple classes in [3], Sounak Dey, Anjan Dutta et al addressed this where text and sketches are used for singular sketch input image. Thus, a cross-modal deep spec is employed to be told common embedding between text and pictures and between sketches and pictures. An attention model in included to detect the individual multiple input objects within the query on the photo dataset. SBIR models are created by training them with extensive datasets for various classes. Training a model with new classes every time isn't a practical and feasible process. Sasi KiranYelamarthi et al [4] proposed a unique generative model, where the model can classify classes that are not seen during training. rather than trying to suit within the existing classes seen during training, they're considered novel and generalized for further associations this is often described because the Zero-Shot (ZS) framework where a model is able to classify an image from a unique class that wasn't used during the training process, giving the name zero for the number of examples employed in that class. Sketch Based Image Retrieval (SBIR) could be a challenging domain mainly due the anomaly and therefore the abstraction within the sketches. a unique convolutional neural network supported the in [5], Yonggang Qi, Yi Zhe Song, Honggang Zhang, Jun Liu proposed a Siamese network for SBIR. By linking two Convolution Neural Networks (CNN), the triplets contain a Boolean variable denoting True, the class label and an image, if the image belongs to the identical class or False, if different. The idea is to fill the gap between such similar and dissimilar classes sketches specified it can work on novel data as well. Sounak Dey, Pau Riba et al addressed the zero-shot learning process by first suggesting a unique ZS-SBIR dataset [6], QuickDraw-Extended, that consists of 330,000 sketches and 204,000 photos spanning across 110 categories was contributed. Highly abstract amateur human sketches are purposefully sourced to maximize the domain gap, instead of ones included in existing datasets that may often be semi photo realistic. A ZS-SBIR framework was wont to relate the sketch and also the photo classes.

## 3. EXPERIMENTAL SETUP

Sketch Based Image Retrieval can be done in 2 ways:

- Class based image retrieval - Conventional method
- Sketch based image retrieval

In class-based image retrieval the model first takes a drawing as an input and then classifies the input image and retrieves the output from class-wise separated photo dataset. In sketch-based retrieval, all images are put together and the photo dataset is not classified. Here we use sketch-based retrieval of images to obtain the photos from our dataset. The relevant images are retrieved using an object detection library CVlib [8]

### A. Algorithm of Proposed System:

**INPUT:** Sketch containing various drawings

**OUTPUT:** Photos in ranked order

**Step 1:** An input sketch which contains one or more drawings is given to the system

**Step 2:** The various components of the input image is identified individually using contours and then bounding boxes are drawn around each of them to distinguish them.

**Step 3:** The individual sketch images are then cropped and saved separately

**Step 4:** All the cropped images are sent to a classifier system and its class is identified

**Step 5:** Using CVLib, bounding boxes and labels the classes obtained from the classifier system are searched for and then found

**Step 6:** Based on the number of classes classified in each photo, they are given particular ranks.

**Step 7:** The photos are displayed in order of ranking.

### B. Workflow of the Proposed System:

The architecture diagram of the entire system is given in Fig 1. The modules in the system are explained below:

**1. Detection of individual sketches:** In this step the input image containing sketches is given and using contours function, the individual contours are obtained. These contours relate to individual sketch components. From the contours obtained, right and top, bottom and left coordinates are extracted. Bounding boxes are drawn around each of these sketch components and saved separately.

**2. Classification of the Sketch:** The separate sketches are loaded individually and their class is predicted.

**3. Detection of Components in the Images:** Using CVLib, various objects that are detected in the photos or an input image are obtained. This contains a list of classes that may also be irrelevant to the query. So, the class of the query is sketched and the bounding boxes are drawn only around relevant queries

### 4. Ranking:

A photo or drawing that is inputted to the model or system may have more than one sketch. Here the Ranking system comes into play. The class or group with highest number of images in the input will get displayed first and then in descending order till no class or group has an image.

### Dataset Used:

The dataset contains two parts:

**1. Sketches** - Drawings of various levels of abstractions

**2. Photos -** A dataset of photos to obtain from based on input query and order by relevance.

### 1. Sketch Dataset:

The sketches are obtained from three sources namely, QuickDraw! [10], TU-Berlin [11] and Sketchy [9]. Each of these sketch datasets contain 110, 251 and 125 classes respectively and the sketches given in each of the dataset are of varying complexities.

The number of sketch images with size of individual images in each of the dataset is given in Table- I.

**Table -1:** Images per class in each dataset

| SNO | DATASET | IMAGES PER CLASS | IMAGE SIZE (pixels) |
|---|---|---|---|
| 1 | QuickDraw! | 3001 | 256x256x3 |
| 2 | TU-Berlin | 80 | 1111x1111x3 |
| 3 | Sketchy | 500 to 800 | 256x256x3 |

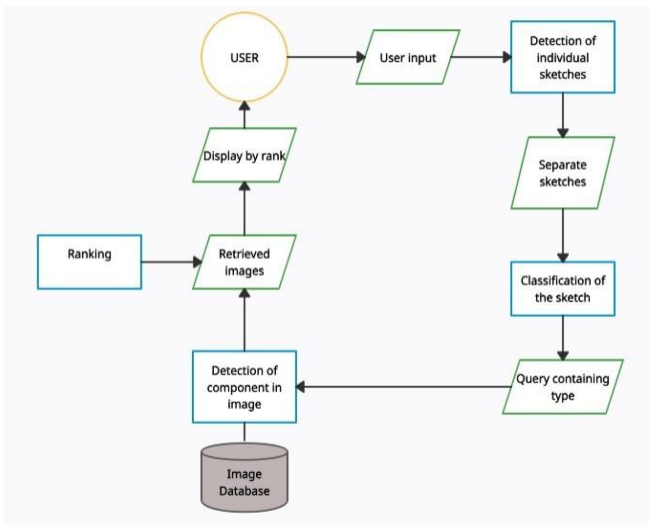


**Fig -1**: Architecture diagram of the Proposed System

Fig 2 - 4 show some sketch images of the class apple from each of the three datasets The level of abstraction between sketches of the same class from different datasets can be observed.

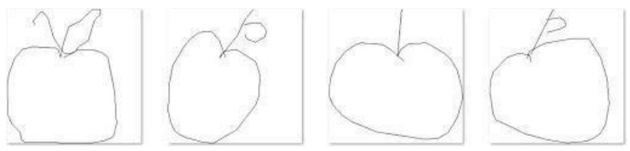


**Fig -2**: QuickDraw! Dataset

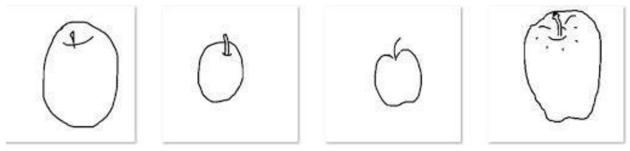


**Fig -3**: Sketchy dataset

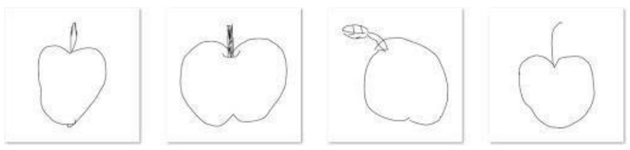


**Fig -4**: TU-Berlin dataset

**2. Photo dataset:** Sketch datasets such as QuickDraw! and Sketchy include sketches and photos as well. Since most photos in the database contain utmost 2 components, a simple database was created such that photos contained more than 3 components. This was done by concatenating different images of different classes randomly. The combinations used are 4 photos, 3 photos and 2 photos, Fig. 5 to 7. These images are concatenated using python and an online software, Peko-Step [12]

**Fig -5**: Photo containing 2 classes

**Fig –6:** Photo containing 3 classes

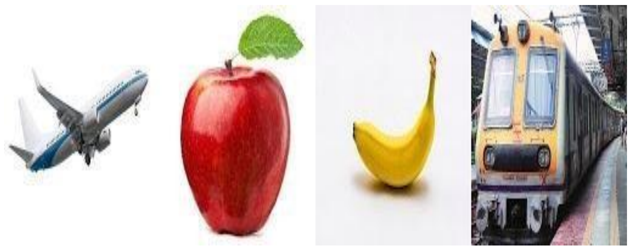


**Fig –7:** Photo containing 4 classes

## 4. METHODOLOGY

### A. Detection of Individual Sketches:

The sketch dataset that we have taken for this project is QuickDraw. It contains a total of 110 classes, of which 15 classes are chosen. The 15 classes are: Airplane, elephant, fire hydrant, knife, pizza, teddy bear, train, apple, banana, bicycle, bird, car, cat, chair, cup. The sketches in this dataset are in binary format. that is, black and white thus each pixel contains values from 0 to 255 where 0 indicates black, 255 indicates white and the intermediate values denote the grayscale range. An input image is given to the system to detect the various sketches present in it. An input image (Fig. 8) is given to the model to detect the various individual sketches present in it.

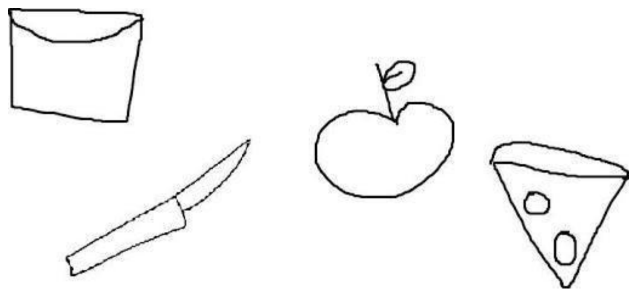


**Fig –8:** Input image containing various sketches

## B. Finding Individual Sketches:

From this input image, the various individual sketches are obtained by finding the contours of every single sketch present in the input image. A contour can be defined as a curve or a line of continuous points containing pixels of the same color or intensity. These contours can be extracted using various edge detection algorithms such as Canny [17] and Sobel [18]. Here we use the OpenCV function findContours [19] to find the contours and obtain them. First the images are converted to grayscale and then a threshold function is applied so that we are able to use this function. The threshold function sets the pixels values above a limit to the said maximum value. The thresholding done here converts pixel values greater than 127 to 255, where 127 is the lowest shade of grey that can be obtained before its transition to white. The thresholded image is inverted and then sent to the findContours function which returns us arrays containing contours of every individual sketch.

## C. Drawing Bounding Boxes:

To obtain the individual sketch components from the input image, a bounding box is drawn. This is drawn using the function, cv2.rectangle(), which uses the top left and the bottom right coordinates. These coordinates can be obtained from the contours array by finding the minimum and maximum values in the first column for the left and right coordinates and in the second column for the top and bottom coordinates. Thus, from the coordinates a bounding box is drawn, Fig. 9, adding 10 extra pixel values to each coordinate and it is displayed to the user. To crop individual sketches, the coordinates obtained before the addition of the 10-pixel units are taken and individual images are snipped and saved on the local system.

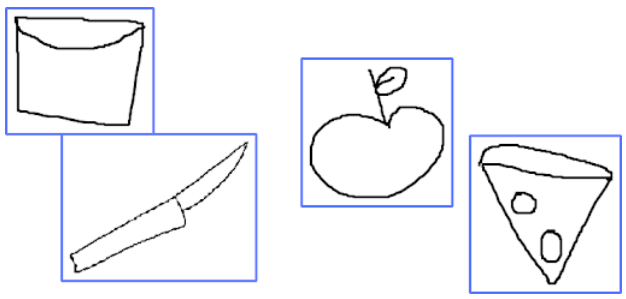


**Fig –9:** Drawing bounding boxes around the individual sketches

The individual sketches are extracted and saved separately as given in Fig. 10 to Fig. 13.

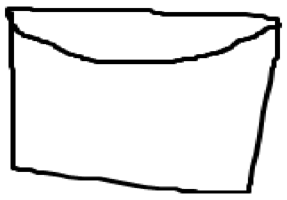


**Fig –10:** Individual sketch – I

**Fig –11:** Individual sketch – II

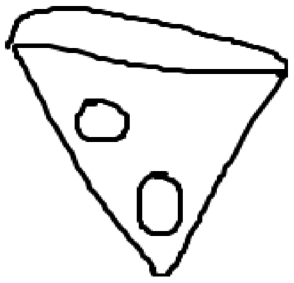


**Fig –12:** Individual sketch – III

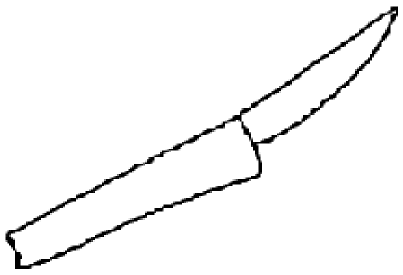


**Fig –13:** Individual sketch – IV

### D. Classifier Model for Sketch Classification:

There are two methods that were implemented to create the classifier. In the first method purely, deep learning is used for the classifier and in the second method we combined both deep learning and machine learning.

#### 1. Deep learning classifier:

The total number of sketch images used in this dataset is 45015 images, which are divided to 38250 training images, 4500 testing images and 2265 validation images.

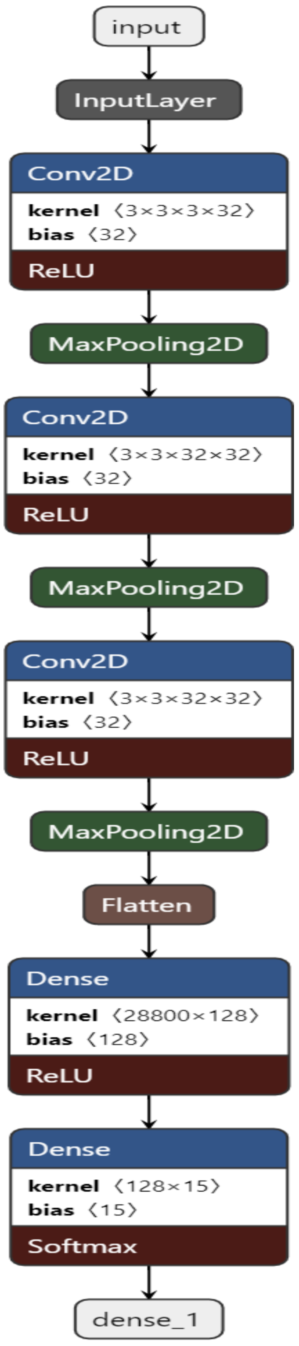


**Fig –14:** Architecture of a Simple CNN

The images are resized to 100*100 for convenience of the users. The model initially designed used a Deep Learning approach such as the Convolution Neural Network (CNN) which has different layers. The total number of layers created in the CNN is 10. But it was inefficient due to heavy loss in data therefore a more productive approach was needed, hence a second method which combined both and machine learning and deep learning was proposed.

#### 2. Deep Learning And Machine Learning Combined:

Due to the extreme loss in data features obtained in the convolution neural network, an alternate method was proposed in which the features are extracted from sketches using deep learning architectures such as VGG16 and InceptionV3 and a basic CNN as well. The basic Convolution Neural Network contains 6 layers. The flattening and dense layers are eliminated because of the need for only features and not a classification from the CNN. The feature vectors obtained were then used to train a Support Vector Machine (SVM) to classify the various classes. The pretrained models were loaded with ImageNet weights. From the classification models created using features from the pretrained architectures and the simple CNN, it was observed that the sketch classifier is more efficient with neural networks and architectures containing a smaller number of layers. Hence VGG16 proved to work better than InceptionV3. Sample input and output for the novel classifier system is given in Fig. 15

```
Predicted Label:
fire hydrant
Actual Label:
Fire_Hydrant
```

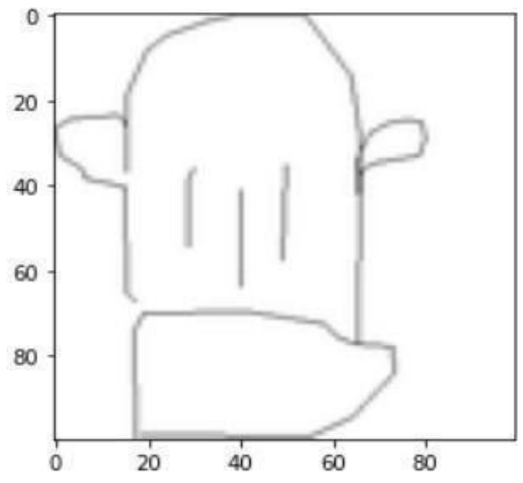


**Fig –15:** Sample Input and Output

### E. Object Detection in Photos:

The OpenCV library CVlib [8] is used to detect objects in an image. The library is trained on COCO dataset [24] and uses the pre-trained YOLO v3 weights. The dataset contains 80 classes of different objects. CVlib can perform face detection, gender detection and object detection.

To detect multiple objects in a photo, the object detection function is used. Given a photo, cvlib's Detect Common Object function will detect all objects seen in the photo, which includes redundant objects as well. The function returns bounding boxes coordinates, labels and confidence scores of the objects. Hence the confidence scores, labels and bounding box coordinates of the objects that are not a part of the input query are removed. Consider an input photo read from the database.

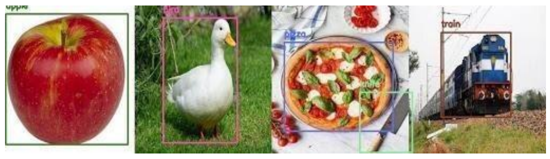


**Fig –16:** CVLib object detection performed on a photo

The detect_common_objects function detected all objects in the photo, Fig. 16. Consider a query: (knife, pizza ) Since the photo contains detection for objects other than the queries, the bounding boxes and labels for those classes other than query are eliminated, thus giving Fig. 17

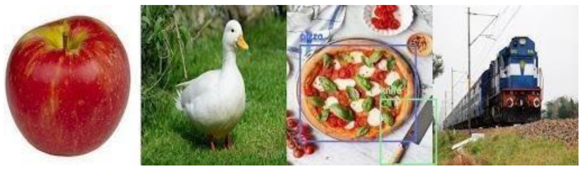
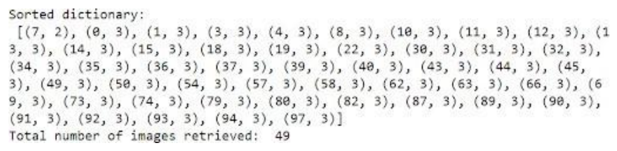


**Fig –17:** Bounding boxes only for query objects

## F. Ranking and Retrieval:

After filtering photos containing only components in the query, they must be ranked according to relevance. Relevance, here is a measure that counts the number of query objects found in the photos counting different instances (objects) of the same class as 1. Based on the number of query objects, given in the input sketch, found in each of the photos, they are given a rank. Rank=1 is given to photos containing all query objects, rank=2 is given to photos missing at-most 1 query object and rank=3 is given for all other images. NOTE: Images that do not contain any of the query objects are discarded in the retrieval process. The tuple in the sorted dictionary contains images in the ranked order. Consider a tuple from the dictionary (7,2) in Fig. 18. Here 7 is the photo number and 2 is the rank.



```
Sorted dictionary:
 [(7, 2), (0, 3), (1, 3), (3, 3), (4, 3), (8, 3), (10, 3), (11, 3), (12, 3), (1
3, 3), (14, 3), (15, 3), (18, 3), (19, 3), (22, 3), (30, 3), (31, 3), (32, 3),
(34, 3), (35, 3), (36, 3), (37, 3), (39, 3), (40, 3), (43, 3), (44, 3), (45,
3), (49, 3), (50, 3), (54, 3), (57, 3), (58, 3), (62, 3), (63, 3), (66, 3), (6
9, 3), (73, 3), (74, 3), (79, 3), (80, 3), (82, 3), (87, 3), (89, 3), (90, 3),
(91, 3), (92, 3), (93, 3), (94, 3), (97, 3)]
Total number of images retrieved:  49
```

**Fig –18:** Images in Ranked Order

## G. Performance Analysis:

### 1. Simple CNN - Deep learning approach:

**Table - II:** Performance analysis for deep learning

| MODEL | F1 SCORE | ACCURACY |
|---|---|---|
| Basic cnn | 0.44 | 0.4352 |
| Vgg 16 | 0.21 | 0.2203 |

### 2. Performance Analysis of 3 models:

F1- Score alongside accuracy is considered as the major metric of efficiency. The classification report for the validation set of the dataset is given in Table- III.

**Table - III:** Performance analysis for validation set

| MODEL | F1-SCORE | ACCURACY |
|---|---|---|
| Vgg16 | 0.89 | 0.8922 |
| Inceptionv3 | 0.69 | 0.6866 |
| Basic 6 layer CNN | 0.65 | 0.76 |

### 3.Performance Analysis VGG-16:

The validation classification report for the 15-class dataset in VGG-16 is included in Table -IV.

**Table - IV:** Performance analysis for validation set

| CLASS | PRECISION | RECALL | F1- SCORE |
|---|---|---|---|
| airplane | 0.85 | 0.87 | 0.86 |
| apple | 0.97 | 0.96 | 0.96 |
| banana | 0.92 | 0.93 | 0.92 |
| bicycle | 0.92 | 0.95 | 0.93 |
| bird | 0.82 | 0.81 | 0.82 |
| car | 0.88 | 0.87 | 0.88 |
| cat | 0.81 | 0.85 | 0.83 |
| chair | 0.95 | 0.91 | 0.93 |
| cup | 0.93 | 0.92 | 0.93 |
| elephant | 0.84 | 0.87 | 0.85 |
| fire hydrant | 0.81 | 0.83 | 0.82 |
| knife | 0.92 | 0.87 | 0.89 |
| pizza | 0.96 | 0.97 | 0.96 |
| teddy bear | 0.92 | 0.92 | 0.92 |
| train1 | 0.90 | 0.86 | 0.88 |
| Accuracy | | | 0.89 |
| Macro avg | 0.89 | 0.89 | 0.89 |
| Weighted avg | 0.89 | 0.89 | 0.89 |

## 5. RESULTS AND DISCUSSIONS

From Table - II, it is inferred that a pure deep learning-based approach proves to be inefficient due to the less features contained in the sketches. Since the sketches contain only black edges on a white background, it causes high data loss and relevantly low details to learn from. Pre-trained models proved to perform poorer than the basic CNN. Table - III shows that the machine learning based approach using SVM works better with deep learning-based feature extraction. Here the feature extraction is efficient when the pretrained model used for extraction contains less number of layers. This method is also useful when the dataset is imbalanced as the SVM only learns from the extracted features and avoids overfitting of the class with majority which occurs in the case of deep learning. To improve the f1- score, classes with lower f1-score can be trained with more sketch images. The need for an extensive computing device occurs when using more data for training the SBIR classifier, which can be eliminated using this method as it requires comparatively less memory to function. Since CVLib is based on the COCO dataset, it is trained to only detect objects belonging to those 80 classes. This limits us to use classes in our dataset common to both the COCO dataset and the sketch dataset. Hence, as an improvement, a separate object detection model can be created that allows for usage of all classes taken in the dataset. This object classifier can be created and deployed using Tensorflow's Object Detection API [25] using the guide given in [26]

## 6. CONCLUSION

From the results obtained, it is understood that a model using pure deep learning does not classify images accurately due to the nature of the sketches being binary i.e.. black and white and the dataset being sparse, when compared to obtaining features using a deep learning model and then training on those features using a support vector machine. The sketch images contain only 0 to 255 as pixel values, which is greatly reduced as we move down the filtering layers from the input convolution layer to the final output dense layer. This is also a possible reason for the lower accuracy of the convolution neural network (CNN). Due to extracting features using pretrained architectures like VGG16, the SVM model is able to predict highly detailed images although the training was done only on heavily abstract and light featured sketch images.

## REFERENCES

1. Hirata, K., Kato. T (1992) 'Query by visual example-content based image retrieval', International Conference on Extending Database Technology: Advances in Database Technology, pp. 56– 71

2. Ayan Kumar Bhunia, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yi- Zhe Song (2020) 'Sketch Less for More: On-theFly Fine-Grained Sketch Based Image Retrieval', IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3. Sounak Dey, Anjan Dutta, Suman K. Ghosh, Ernest Valveny, JosepLlados,Umapada Pal (2018) 'Learning Cross-Modal Deep Embeddings for Multi-Object Image Retrieval using Text and Sketch', 24th International Conference on Pattern Recognition (ICPR)

4. Sasi Kiran Yelamarthi, Shiva Krishna Reddy, Ashish Mishra, and Anurag Mittal (2018)'A Zero-Shot Framework for Sketch Based Image Retrieval',The European Conference on Computer Vision (ECCV)

5. Yonggang Qi, Yi-Zhe Song, Honggang Zhang, Jun Liu (2016), 'Sketch-based image retrieval via Siamese convolutional neural network', IEEE International Conference on Image Processing(ICIP),

6. Sounak Dey, Pau Riba, Anjan Dutta, JosepLlados, Yi-Zhe Song (2019), 'Doodle to Search: Practical Zero-Shot Sketch-based Image Retrieval', The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

7. VGG-16, https://keras.io/api/applications/vgg/

8. CVLib - object detection library, https://www.cvlib.net/

9. Sketchy dataset, https:// sketchy.eye.gatech.edu/

10. QuickDraw! dataset, https://github.com/ sounakdey/doodle2search

11. TU-Berlin dataset, http://cybertron.cg.tuberlin.de/eitz/projects/classifysketch/

12. Peko-Step, https://www.peko-step.com/en/tool/combine-images.html

13. Python 3.8, https: //www.python.org/downloads/release/python-380/

14. Kaggle, https://www.kaggle.com

15. Google colab, https://colab.research.google.com/

16. Support vector machine, https://towardsdatascience.com/httpsmedium-com-pupalerushikesh-svm- f4b42800e989

17. Canny, https://medium.com/sicara/opencv-edge-detection-tutorial7c3303f10788

18. Sobel, http://www.adeveloperdiary.com/data-science/computervision/how-to- implement-sobel-edge-detection-using-python-fromscratch/

19. findContours(), https://www.thepythoncode.com/article/contourdetection-opencv-python

20. Machine learning, https://www.geeksforgeeks.org/machine-learning/

21. Deep Learning, https://machinelearningmastery.com/what-is-deeplearning/

22. Convolution Neural Network, https://towardsdatascience.com/a- comprehensive-guide-to-convolutional- neural-networks-the-eli5-way3bd2b1164a53

23. InceptionV3, https://keras.io/api/applications/inceptionv3/

24. COCO Dataset http://images.cocodataset.org/ zips/val2017.zip

25. Tensorflow Object Detection API, https://tensorflow-object-detectionapi-tutorial.readthedocs.io/

26. Tensorflow Object Detection API usage, https://towardsdatascience.com/how-to-train-your-own-objectdetector-with- tensorflows-object-detector-api-bec72ecfe1d