INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

6th - SEMESTER MINI PROJECT

INTERACTIVE SKETCH RECOGNITION SYSTEM

Submitted to :
Dr. Shirshu Varma

Members:
Subham Raj(IIT2016010)
Garima Chadha(IIT2016020)
Ankit Kumar(IIT2016024)
Vaibhav Srivastava(IIT2016034)
Deepanshu Goyal(IIT2016037)

CERTIFICATE FROM SUPERVISOR

I hereby recommend that the mini project report prepared under my supervision titled "**Interactive Sketch Recognition System**", be accepted as it fulfills the requirements of the mini-project completion of sixth semester of Bachelor of Technology in Information Technology.

Date: 2nd May 2019 **Supervisor**

Place: Allahabad **Dr. Shirshu Varma**

……...

TABLE OF CONTENTS

ABSTRACT

Looking at the advancements and new methodologies being achieved in the field of artificial technology and informatics sector, most of the departments are getting connected with Artificial Intelligence. Machine Learning is nowadays a common tool for a software engineer to monitor over the changes in pattern and do prediction in the required field.

Our project also focus on one such of field that might be used for multiple purposes. “Sketches” are common pen drawings of objects, writings or even some person. They are used at many places like in teaching, planning strategies in games or even in the police departments.

This project also deals with sketches, adding over deep learning (a tool of A.I.) for prediction of different objects. User is predicted an object which resembles mostly to the sketch drawn by him/her. This project will discuss in detail over the methodology , along with attractive implementation and results that we achieved with our model.

INTRODUCTION

One of the most fundamental methods of communication among humans has been via hand drawn depictions and sketches. Sketching is probably the only technique readily available to all humans. Based on various learning experiences and interactions, humans are able to semantically identify hand drawn sketches, but machines cannot function in this manner with considerable accuracy.

Image-based search is closely related to **problem of object recognition and classification**. However, a user may not always have the digital photograph of the object for which he/she desired to perform a search. Such searches can be made using Image Classification based on rough sketches made by the user.

The system is based on taking a query in the form of sketched images and classifying them into various categories. An effective method of recognizing such hand drawn sketches makes use of **Convolutional Neural Networks (CNN)**. In recent years, CNNs have made a huge impact in the field of Computer Vision and Image Processing for classification against a given dataset.

Our system binds the hand drawn sketches with modern technological capabilities. Sketch Recognition is meaningful because it harnesses the potential of modern day computers to classify rough sketches, which are imperfect in comparison with the actual digital images.

LITERATURE REVIEW

In the first attempt to utilize CNNs to recognize hand-drawn sketches [1], two popular CNNs – **AlexNet CNN** [2] and a modified version of **LeNet CNN** [3] were used for experiments, and results showed minor improvements over the conventional state-of-the-art.

The very recent attempt in this area is [4], where another deep convolutional neural network, named SketchNet, was proposed for sketch classification. But the main purpose of the paper is to automatically learn the shared structures that exist between sketch images and real images. In order to do this, a triplet was composed of sketch, positive and negative real image that was developed as the input of the neural network. To construct the auxiliary repository, the real images were collected from the web which covered all the sketch categories in the TU-Berlin sketch dataset.The best classification accuracy, achieved in the paper on TU-Berlin sketch benchmark, was 80.42%.

In 2016, Google released an online game titled “Quick, Draw!” — built an enormous dataset of over a billion drawings.This dataset contains 50 million drawings of 340 categories.The dataset is available on **Google Cloud Console[5**].Notably, it has robust potential in OCR (Optical Character Recognition), ASR (Automatic Speech Recognition) & NLP (Natural Language Processing).As the dataset is very large and obtained from user all over the world so this dataset can be used to get better results.

The whole dataset can be used for training purpose but that will require high efficient GPU system.So we can take some part of it to train CNN model and to check it’s accuracy on the classes on which it is trained.

We have selected 15 categories from the available 340 categories and take almost 12000 strokes of each category in csv format to train the model and some unlabeled data of same classes to check the accuracy of trained model.

MOTIVATION

The first and foremost thing that we learn in our childhood is to draw objects like tree, car, cat etc. These sketch resembles some objects/things in real life. Sketching and drawing is a great way to improve your creative skills and start thinking in a different way. Art shows you that there is normally **more than one way to solve a problem**. These can be really helpful for personal development and solving problems. Skills you learn through sketching can be applied in a number of different areas in your life.

The prediction involves taking an input of a user drawn sketch (identified using color detection techniques) of an object after which possible results are displayed with decreasing order of accuracy. This is achieved with the help of a python GUI.

The project is implemented using **Convolutional Neural Network (Deep Learning).** Sketching is one of the most common rendering technique available. Predicting what a sketch shows with the help of computer systems gathers attraction. “What does a sketch resembles?” is predicted with the help of our model. Since the model is developed using recent emerging technology which is deep learning.

A lot of research work is also not yet made in this topic, as discussed above it still have a good scope of increasing accuracy while predicting any answer. Thus, **both by the topic attractivity and technique used**, we were motivated to choose this topic as to build a sketch detector which might be helpful in different ways.

PROBLEM DEFINITION

Definition:

All humans are not great artists. Everybody has a different way of drawing the same object. Most humans make dramatic exaggerations or simplifications as per their convenience and capabilities. As an example, people have shared iconic descriptions of certains objects looking like stick figures that depict a person.

In this project, we attempt to create the functionality of categorization and image search systems in order to provide the **"closest match"** image category at the object portrayed in the input rough sketch. Also, we use a **dataset that comprises of sketches**, not real images. This increases accuracy of model to some extent.

We use CNN to train our model. Interaction with the user is made possible by **color detection techniques** using system webcam. User draws a sketch using a "**blue colored object"**, using which prediction regarding the class of the sketch is made.

Objective:

The objective of our project is to recognize hand drawn images with high accuracy and in less time. We make use of the most effective technique of recognition available to us, that is, **Convolutional Neural Networks (CNN)**.

We have also developed a **GUI** which will allow the user to draw, and with each stroke, our model will output what the current sketch might represent. Hence, as soon as the user sketches something into the application, the trained model analyses the sketch and outputs the most matched class.

SOFTWARE REQUIREMENTS

We have used Python to implement the **Convolutional Neural Network** used to train the Sketch Recognition Model. Certain modules and libraries of Python that are used in this process are listed below.

- **Keras:** This python library makes it pretty simple to build a CNN. It is a higher level library that operates over Tensorflow.
    - For training our model, we have used the **"Sequential Model"** of keras. This model makes it easier to stack sequential layers of the network one after the other, starting from input to the output layer.

- **OpenCV:** This is an open source library mainly used for handling computer vision problems.
    - We have made use of this library to handle the image data and process it according to requirement. Also, this library has been extensively used to make GUI for our model (for video capturing, noise removal, etc.).

- **NumPY:** This has been used to facilitate faster and easier mathematical computations.

- **Pandas:** This module provides high performance, easy to use data structures and data analysis tools. This handles all significant input data processing methods.

- **Tqdm:** It has been used as a python iteration module. It acts as a fast, extensible progress bar.

Several other python modules such as os, seaborn etc. have also been used to facilitate easier implementation of code to train and test the model.

DATASET

For this project, we started by using the **TU-Berlin Dataset,** which was a collection of 20,000 human sketches (taken from TU-Berlin Sketch Dataset). This Dataset consisted of randomly hand-drawn sketches of 250 different categories like airplane, car, train, cake etc.
Each category consist of 80 images having 1111x1111px PNG files which were converted to grayscale images having 28*28 pixels. But, this dataset led to an unsatisfactory accuracy.

This motivated us to use a bigger dataset, i.e, the **Quick Draw Dataset** provided by Google. This dataset is comprises of **50 million sketches** ranging across **345 categories**. This dataset has been collected as part of a pictionary-like game by Google that prompts the user to create a sketch of a certain object in under 20 seconds. Certain sketches from the Quick Draw dataset can be seen in Fig. 1.

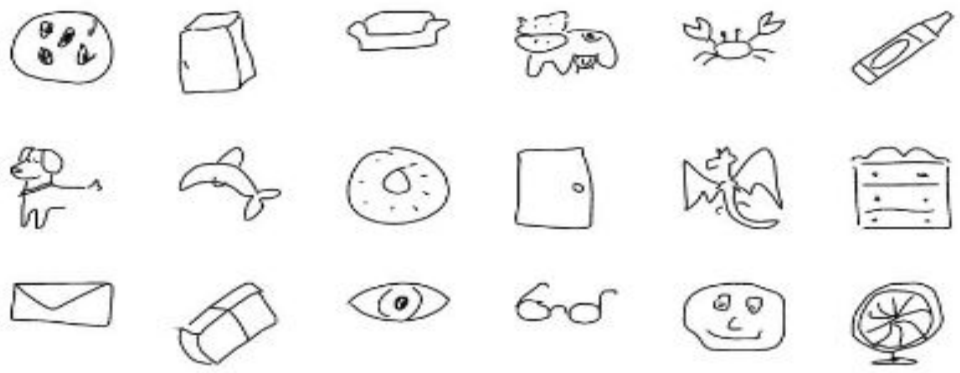


Fig. 1. : Example sketches belonging to Google Quick Draw Dataset

To train our Sketch Recognition Model, we have used sketches belonging to 15 distinct categories. These categories can be seen in Fig. 2.

```
['bicycle', 'camera', 'cup', 'donut', 'eye', 'face', 'flower', 'hockey stick', 'lipstick', 'mermaid', 'pineapple',
'rainbow', 'shoe', 'snake', 'tent']
```

Fig. 2.: Categories used to train model

The data is openly available on **Google Cloud Console**. We have used a preprocessed form of the data available in csv format corresponding to 15 classes shown in Fig .2.

The csv data is available in the following format -

- **Key_id** - It is a unique identifier for each sketch in the dataset.
- **Countrycode** - A string denoting where the creator of sketch was located.

- **Drawing** - A vector representing different strokes made by the user while drawing the sketch. Each tuple in the vector consists of 2 arrays corresponding to the x and y coordinates of the point on the canvas where image is drawn. The stroke information is encoded as shown in Fig. 3.

```
[
  [  // First stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  [  // Second stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  ... // Additional strokes
]
```

Fig. 3.: Drawing Strokes

- **Recognized** - A boolean value indicating whether a particular sketch was recognized by the game.
- **Timestamp** - Time and date when the sketch was created.
- **Word** - Category of a particular sketch.

Data Preprocessing:

The data corresponding to **15 distinct categories** is collected in **csv** format. This data is then read into a pandas dataframe. Next, the data is randomly divided into 100 separate csv files so that the training bias gets reduced. As every image has a key_id, the shuffling is done by dividing the key_id by 100 and taking the remainder as the number of new file the image will be shuffled into. Since the files are large, they are also compressed as .gz (gzip).
The processed data can be seen in Fig. 4. and Fig. 5.

	A	
1	countrycode	drawing
2	DE	[[[0, 28, 36, 49, 76, 83, 124], [0, 74, 108, 144, 247, 251, 255]]]
3	RU	[[[12, 18, 26, 53, 93, 123, 144, 180, 210, 231, 244, 247, 240, 219, 2(
4	US	[[[65, 23, 5, 0, 2, 18, 35, 72, 133, 159, 163, 161, 141, 117, 99, 76, 4
5	CZ	[[[33, 63, 86, 100], [142, 121, 93, 82]], [[39, 73, 132], [153, 135, 97]]

Fig. 4.: Processed Dataset

	D	E	F	G
1	timestamp	word	y	cv
2	2017-03-04 10:58:21.305280	hockey stick	7	0
3	2017-03-04 19:03:05.609420	face	5	0
4	2017-01-27 17:51:59.953920	cup	2	0
5	2017-01-11 09:49:27.832880	mermaid	9	0

Fig. 5.: Processed Dataset

In Fig. 5., the column **“y”** represents the class that a drawing belongs to, eg: 0 is for class “bicycle”, 1 for class “cup”, etc.

METHODOLOGY AND IMPLEMENTATION

After storing the data required for training in the required format as shown in the database description section, our next step is to apply convolutional neural networks to our database. Let's have a look at the concept involved in CNN (Convolution Neural Network).

Methodology:

This section is divided into 2 parts: that contains the convolutional neural network in brief discussion and how it is used in our code, respectively.

The terms and definitions are covered in the following section.

Definitions

1. **Convolution Neural Network :** In the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Various layers in a convNet are defined below.

    - **Convolution Layer :** In convolution layer we take a small window size (kernel) [typically of length 3*3] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.

    - **Fully Connected Layer :** In this layer, each neuron is connected to all the neurons of the previous layer, hence the name. This layer is introduced in our model after flattening of the output of the convolution layers.

    - **Pooling Layer :** We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. This gives us an edge in the number of stages generated and thus helps in reducing the computation to an higher extent. There are two types of pooling :
        - **Max Pooling** : In max pooling we take a window and only take the maximum of all the values present in the window. Well lid this window and continue this process, so we finally get a activation

matrix smaller than the given size, if the pooling matrix size is 2*2 , the final matrix is reduced by half in both the dimensions.

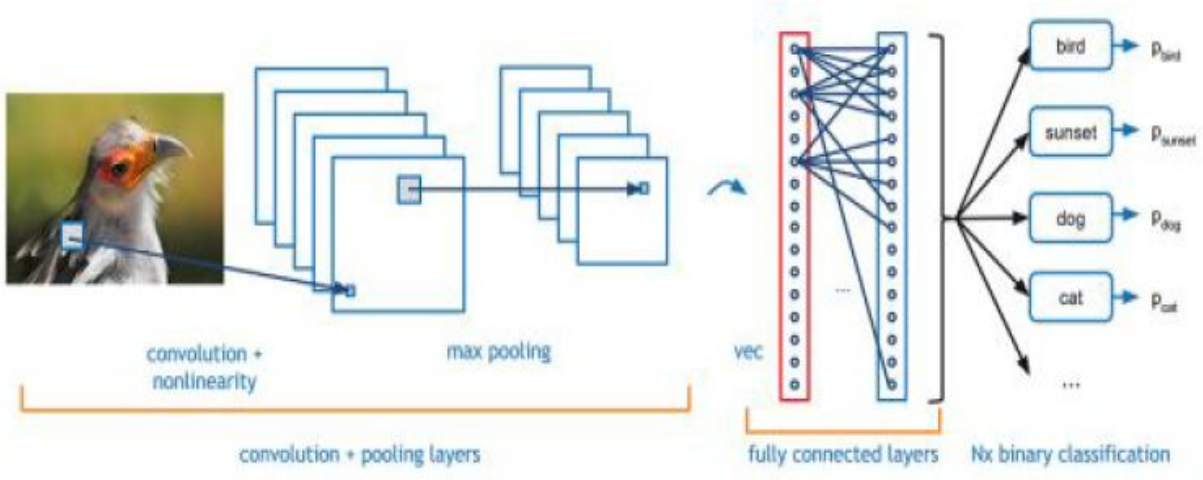


Fig.6. Convolutional Neural Network

- **Average Pooling :** In average pooling we take average of all values in a window, and the remaining process is similar to max pooling. It depends on the use case of the project that when to use an average pooling or a max pooling to reduce the number of layers.

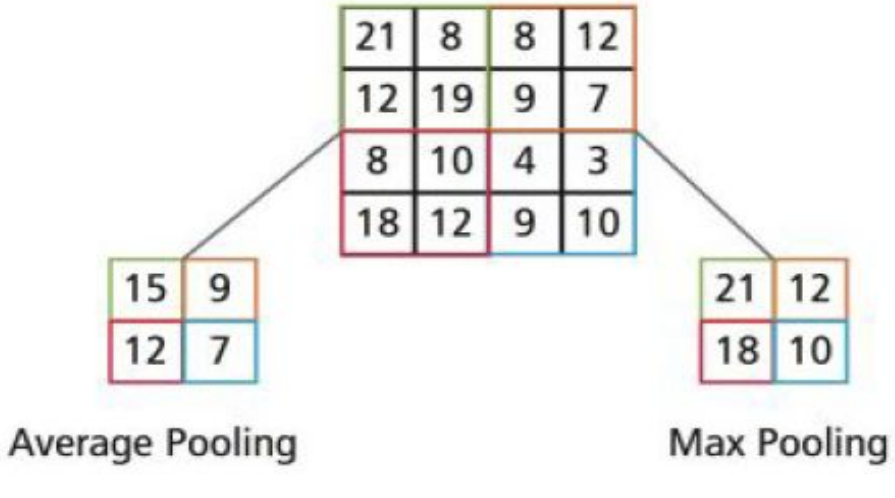


Fig.7. Pooling over 2*2 stride and window

2. Activation Functions:

We have made use of two activation functions so as to introduce non-linearity in our model.

- **ReLU**: It stands for Rectified Linear Unit and is defined by :

  $$f(x) = \max(0, x)$$

  -where f is the output obtained when ReLU is applied to input x. This activation function replaces all negative values by 0. It's graphical representation can be seen in Fig 8.

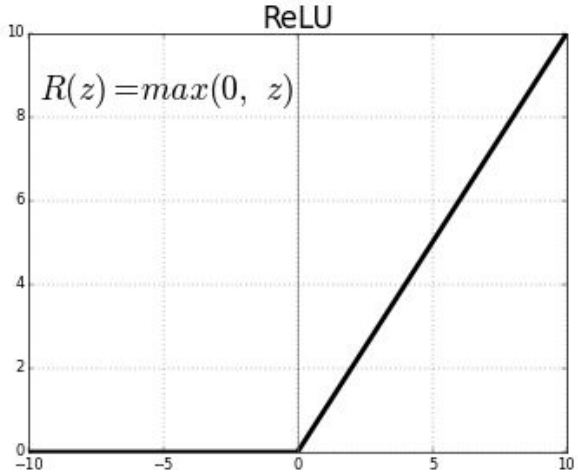


Fig. 8.: ReLU Activation Function

- **SoftMax Function:** This is used in the output layer and it predicts the possibility of each class in the neural network classifier.
  In general, this function can be defined as shown below.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

-where z is the output of a layer before activation is applied.

2. Dropout:

This introduces the concept of regularization in the neural network. Certain neurons are removed based on a dropout probability value. This helps to avoid overfitting in the model.
Hence, the model is generalized to unseen data samples as well. This helps to achieve a better accuracy for the model.

3. Sequential CNN Model (Training Phase):

Designing layers for the convolutional neural networks for our project is done in the following manner.

- The first step is to create an input layer that contains some edge-features. The size of input layer is kept to 82*82*3.
- It is then convolved with 3*3 Matrix so as to attain the second layer of dimension 82*82*128.But since we have applied same padding,output dimension is same as input dimension.
- Then max pooling is applied on that layer to achieve the first hidden pooled layer of dimension 41*41*128.
- Similarly, convolution is performed for the second time and max pooling is also performed resulting into a layer of dimension 19*19*128.
- Again the convolution is performed with the resulting layer with 3*3 kernel with 256 mapping resulting into 17*17*256 dimension.
- Max pooling is again applied to the resulting layer which finally gives us a layer of dimension 8*8*256.
- This layer is then flattened to obtains a single layer of features.
- Dense layer is formed at this stage with a dense layer dropout = 0.10
- Convolutional layer dropout is set to 0.
- Finally, the feature set is passed through softmax activation function so as to get the probability of every class.
- Resulting output is the class with the highest probability.
- The actual class value for any sketch is converted to one-hot notation.
- Cross-entropy error function is used to compute the loss to be used in the back-propagation step while training.

Structure of the layers is shown in the Fig. 9.:

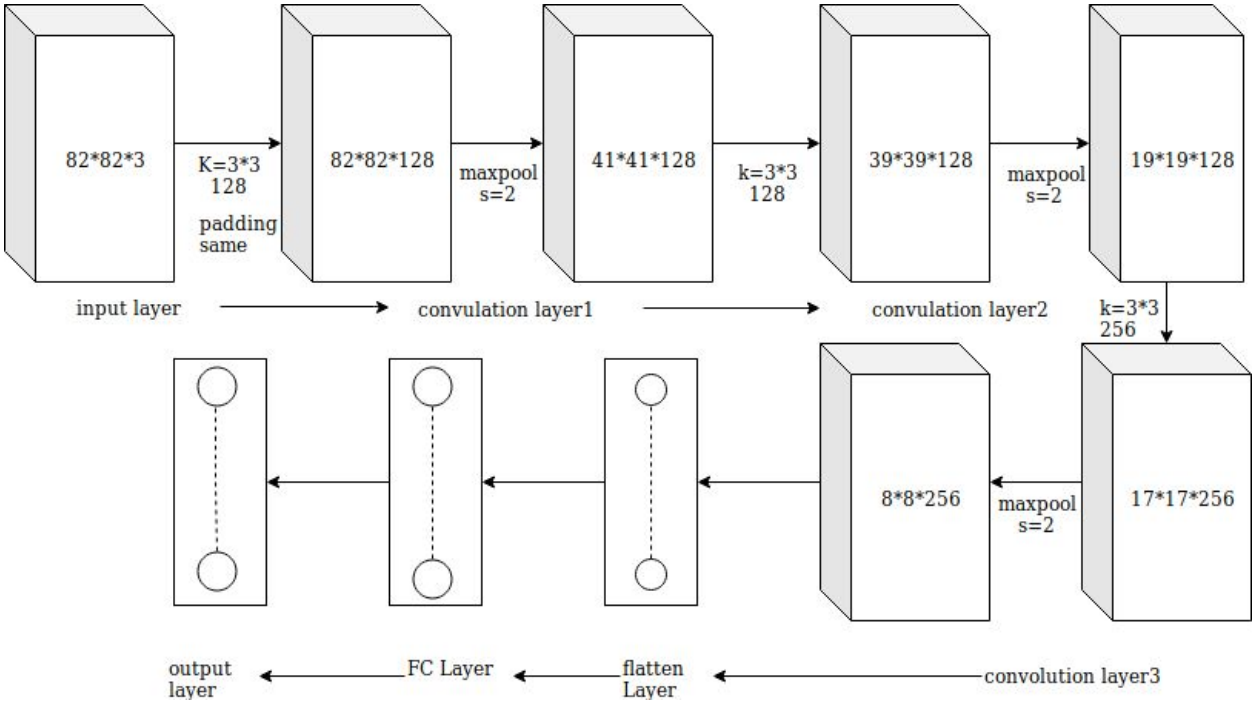


Fig.9. Layers in our model (along dimensions)

Implementation :

After training the dataset and generating all the layers by the code, generates a model named “sketch_user.h5”, that contains the model trained under the training dataset. The second phase involves using the above generated model to predict user drawn sketches. These sketches are taken input by means of a GUI App. Prediction using the model is discussed in the following way :

Camera Input:

- With the help of CV2 library of python, camera is opened to input a sketch for prediction.
- The input is taken through an object with a specific color range, this range is set to dark green to dark blue object in the image as captured by the camera.
- The range of dark blue color is selected in the hsv model of coloring.

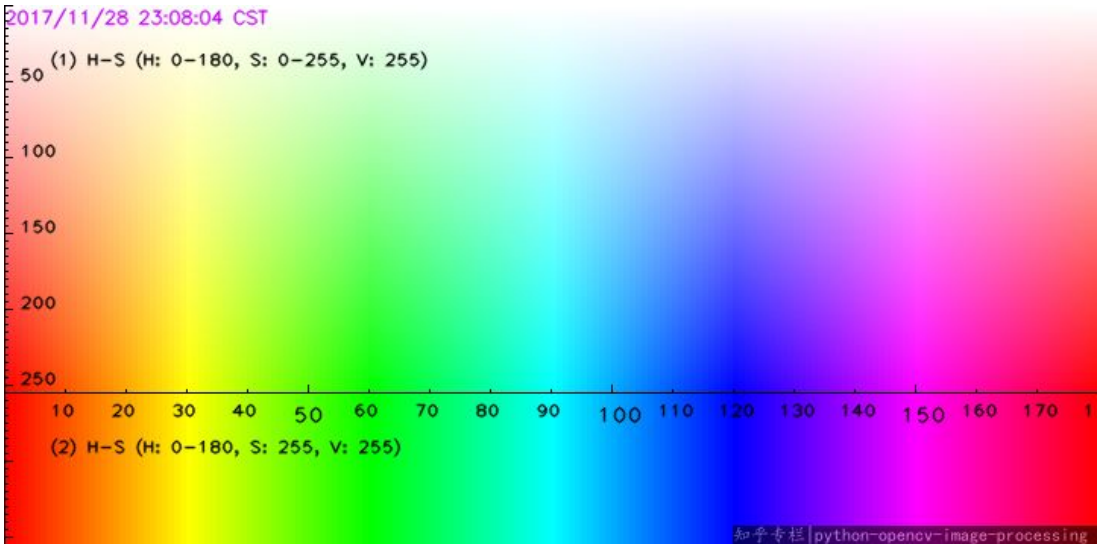


Fig.10. HSV values of different colors

- As we move that object in front of the camera, with the help of that object, a sketch is drawn at the screen that resembles some object in the form of sketch.

Pen Stroke Input and Prediction :

Some steps involved while giving an input via this method are as following :

- The objects in the background should not contain any sort of color that matches to the range of the object used for giving input to the system.
- The image drawing object should be kept to a optimal range so as to give a good flow to the sketch drawn
- The sketch drawn should be made in a single stroke.
- After completing the sketch , just cover the object with your hand, so as to begin the processing for prediction part.

Image in Fig. 11. shows a method to give input to the model.

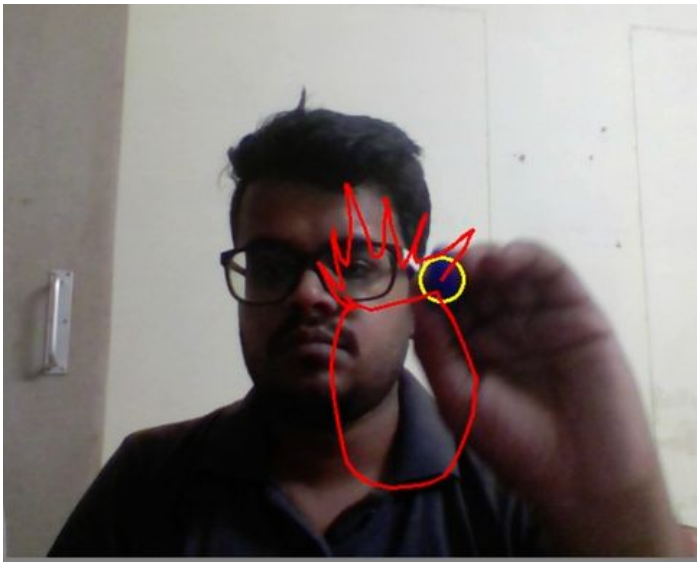


Fig.11. Input given by object movement

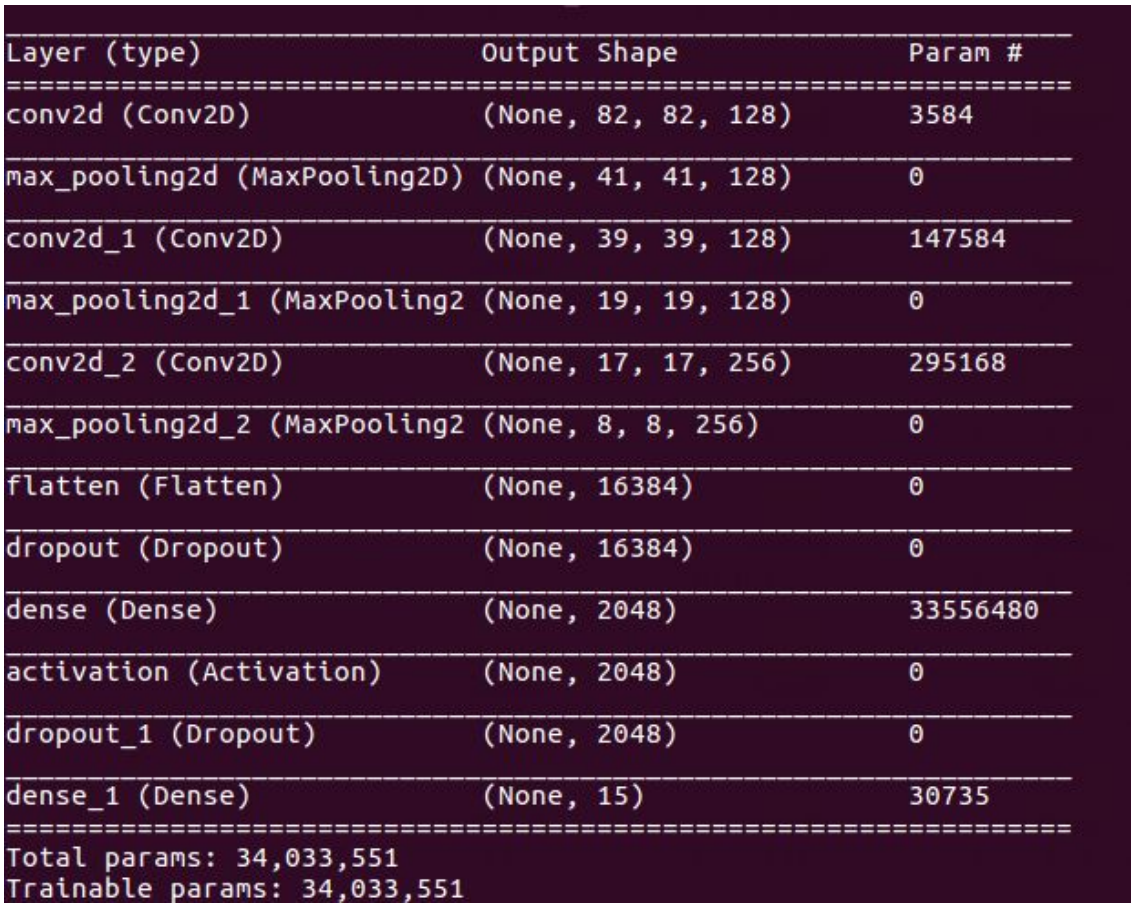
Predicting by the User's Input :

- When image drawing is over, then object drawn is extracted again with help of CV2 library and class label is predicted which matches most to the input figure.
- The sketch is taken and processed so as to match with the dimensions of the data set, this is done with the help of image_resize option in opencv2.
- It is given to the model and then the output is predicted.
- The prediction of the image drawn above which is a **pineapple** is as shown in Fig. 12.

Fig.12. Object predicted of the input sketch

RESULTS AND OBSERVATIONS :

1. **Summary of CNN Model:** The layers obtained after training on the dataset, is matching with the expected forms, the Fig. 13. shows the details.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 82, 82, 128)	3584
max_pooling2d (MaxPooling2D)	(None, 41, 41, 128)	0
conv2d_1 (Conv2D)	(None, 39, 39, 128)	147584
max_pooling2d_1 (MaxPooling2	(None, 19, 19, 128)	0
conv2d_2 (Conv2D)	(None, 17, 17, 256)	295168
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 256)	0
flatten (Flatten)	(None, 16384)	0
dropout (Dropout)	(None, 16384)	0
dense (Dense)	(None, 2048)	33556480
activation (Activation)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 15)	30735

Total params: 34,033,551
Trainable params: 34,033,551

Fig.13. Detail of Layer of CNN Model

2. **Accuracy:** For testing purpose, 10% of the training dataset was taken that contains labelled data of the object class. It is fed into the system and predicted label is matched with the actual label.The model was trained on 10 epochs, with 2000 steps per epoch. Accuracy obtained in the training part is **91.8%.**

3. **Confusion Matrix:** This matrix summarizes the results produced by the classifier. For each classification class, it depicts the number of correct and incorrect predictions made. This depicts the accuracy of the model and how much confused it is while making predictions. Confusion Matrix is shown in the Fig. 14.

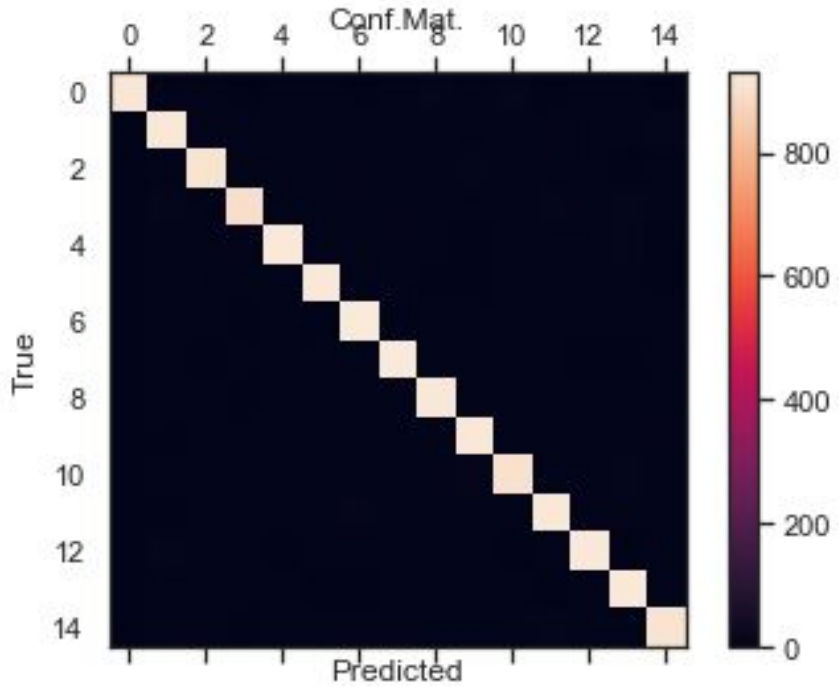


Fig.14. Confusion Matrix

4. **Recall** : This is the number of true positives divided by the sum of number of true positives and false negatives. It depicts the recognition rate of users belonging to a particular class. A higher value for this ratio is desired, which indicates that classification classes are correctly recognized.

$$\mathbf{Recall} = \frac{TruePositives}{TruePositives + FalseNegatives}$$

5. **Precision :**This is the number of true positives divided by the sum of true positives and false positives. It defines the level of correctness of a predicted label.

$$\mathbf{Precision} = \frac{TruePositives}{TruePositives + FalsePositives}$$

6. **F1 Score :** This is defined as the Harmonic Mean of Precision and Recall, hence it takes both the metrics into account.

$$\textbf{F1 Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The metrics calculated for the ConvNet Classifier can be seen in Fig. 15.

	precision	recall	f1-score	support
bicycle	1.00	0.91	0.95	1000
camera	0.90	0.92	0.91	1000
cup	0.90	0.92	0.91	1000
donut	0.91	0.92	0.91	1000
emoji	0.90	0.90	0.90	1000
eye	0.91	0.92	0.92	1000
flower	0.92	0.91	0.92	1000
hockey_stick	0.90	0.91	0.91	1000
lipstick	0.90	0.90	0.90	1000
mermaid	0.89	0.91	0.90	1000
pineapple	0.90	0.91	0.90	1000
rainbow	0.91	0.91	0.91	1000
shoe	0.91	0.90	0.91	1000
snake	0.90	0.90	0.90	1000
tent	0.92	0.92	0.92	1000

Fig.15. Metrics

CONCLUSION

- It was observed that when the model was trained on the TU-Berlin Dataset, a low accuracy was obtained. But, training the model on Google Quickdraw Dataset resulted in considerable improvement in accuracy (91.8%).
- Confusion Matrices are plotted and various other metrics like precision, recall and f1-score are used for analysis purpose.
- The structure of CNN Model is chosen experimentally.

FUTURE WORK

- The model can be extended to predict for all 345 classes (as in QuickDraw Dataset).
- Other neural network models such as RESNet etc. can be used to train the model, so as to increase the accuracy.

REFERENCES

[1] Sarvadevabhatla, R.K, Babu, R.V. [2015] Freehand Sketch Recognition Using Deep Features. arXiv preprint arXiv:1502.00254.

[2] Krizhevsky A, Sutskever I, Hinton GE. [2012] Imagenet classification with deep convolutional neural networks. In:Advances in neural information processing systems, pp.1097-1105

[3] LeCun Y, Bottou L, Bengio Y, Haffner P. [1998] Gradient based learning applied to document
recognition. Proceedings of the IEEE. 86:2278-324

[4] Zhang, H, Liu S, Zhang C, Ren W, Wang, R, Cao X. [2016], SketchNet: Sketch Classification With Web Images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1105-13

[5]https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap//?pli=1