Quick Draw Doodle Recognition

1st Anubhav Shrimal
M.Tech CSE (MT18033)
IIIT Delhi
New Delhi, India
anubhav18033@iiitd.ac.in

2nd Vrutti Patel
M.Tech CSE (MT18020)
IIIT Delhi
New Delhi, India
vrutti18020@iiitd.ac.in

***Abstract*—In Quick Draw the AI system tries to classify the hand-drawn doodle into a predetermined category which is quite similar to pictionary. By this project we are trying to achieve the same using various methodologies such as K-Nearest Neighbor and its variants, CNN and CNN+LSTM, and compare their performance on different evaluation metric.**

***Index Terms*—Quick Draw, doodles, strokes, images, K-NN, K-means++, CNN, LSTM**

I. Literature Review

Comparing human performance against computational recognition methods by developing a bag-of-features sketch representation and using multi-class support vector machines as well as K-NN to train on the sketch dataset, in order to classify sketches is the main concept in paper [2]. In paper [1], the authors have used CNN architecture for freehand sketch recognition and also deep convolutional neural network with some modification in ResNet architecture so that it can be used for sketches instead of images. The results show intra-class variation and inter-class overlap caused by differences in artistic interpretation and scarcity of visual information in different sketches.

Representing images through bag of words model, extracting features using SIFT technique, applying K-means clustering for vector representation of images and generating histogram from that is the approach followed in paper [4] and then comparison is made between KNN and SVM classification method. The authors in paper [5] have presented a recurrent neural network which is able to construct stroke-based drawings of common objects by training the model on a dataset of human-drawn images representing different classes. They described new robust training methods for generating coherent sketch drawings in a vector format.

II. Dataset Used

The Quick Draw dataset is a collection of millions of drawings of 300+ categories which is created by using doodles of different players of the game. The drawings were captured as timestamped vectors, tagged with metadata of the label which the player was asked to draw. There are multiple different representations of the Quick Draw dataset. One dataset represents each drawing as a series of line vectors representing strokes made to draw an object and another contains each image in a 28x28 grayscale matrix. As we focus on classification of the entire doodle, we use the latter version of the dataset.

The complete dataset is huge (73GB) and hence we have used only a subset of the complete data containing 23 classes. The data of each class was balanced by taking 35,000 images for each class and then dividing into 80:20 split for training and testing data and further splitting the training data as 70:30 validation and training split.

TABLE I: Dataset Division

	Training	**Validation**	**Testing**
For Each Class	19600	8400	7000
Total Data	450800	193200	161000

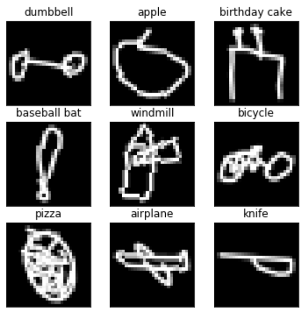

Fig. 1: Dataset Visualization

III. Tasks Completed

We have completed the tasks that were mentioned in our midterm milestone in the proposal which comprises of data visualization and applying K-NN and CNN to classifiy the images. Following is the description of the tasks that have been completed so far:

A. *K Nearest Neighbor*

The intuition for applying this algorithm is that the images of same category should look relatively similar. Based on this assumption, we could determine the category of a given drawing by looking at which training examples are the closest to the doodle under test. This intuition corresponds with K-NN algorithm.

- **KNN with 1-Closest Centroid:** KNN is a computationally heavy approach and is infeasible for such a large dataset. To overcome this we compute centroid for each category using the training dataset and classify test set using the closest centroid based on Euclidean distances. This reduces the number of points we look at for each test image to only 23 which is one centroid for each category.
- **KNN with Kmeans++:** In this approach we create multiple clusters per category using Kmeans which will capture the different variations within a category. We use kmeans clustering to create 5 centroids per category. We initialize the centroids using k-means++ initialization method to ensure that the final centroids are as different as possible. Then, we follow the KNN algorithm to compare each test example with every generated centroid.
- **KNN with Kmeans++ and weighted voting:** This approach is an extension to above approaches and uses a weighted voting scheme along with multiple clusters for each category for final predictions. We give more weightage to votes from closer centroids than votes from the centroids which are away from the test images. The weight is given by following formula: $w_i = 1/\ ||\ x_i - c\ ||_2$

B. *Convolutional Neural Network implemented from Scratch*

The input image size used is 28x28. We have applied Random Rotation of 10 degrees as image transformations which were normalized by dataset mean and standard deviation. We created a CNN architecture from scratch with 5 Convolution layers, 2 Max-Pooling layers and Fully Connected (FC) layers of dimensions 100 and 23. Activation function used is ReLU. Dropout of 0.2 is added in between FC layers. The optimizer used is Adam with a learning rate of 0.001 with Cross Entropy Loss function. The model was trained for 10 and 20 epochs.

IV. Tasks Remaining

The following tasks will be covered in the final milestone:

- Pre-Trained deeper CNN architecture such as ResNet34 and ResNet50.
- Extending CNN with Long Short Term Memory network (LSTM) for pen stroke based recognition.
- Model Comparisons and Analysis based on MAP@3 and CMC.
- Creating OpenCV based doodle recognition application for user interaction.

V. Results

TABLE II: Accuracy on different variations with KNN

KNN Model	**Nearest Neighbor = 5**	**Nearest Neighbor = 1**
1-CC	17.41	59.16
Kmeans++	54.74	70.86
Weighted Voting	65.23	70.86

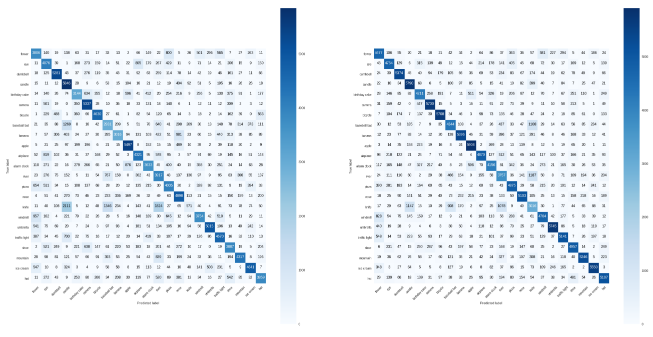
(a) KNN 1-CC (k=1) (b) KNN + Kmeans++

Fig. 2: Confusion Matrices for KNN based approaches

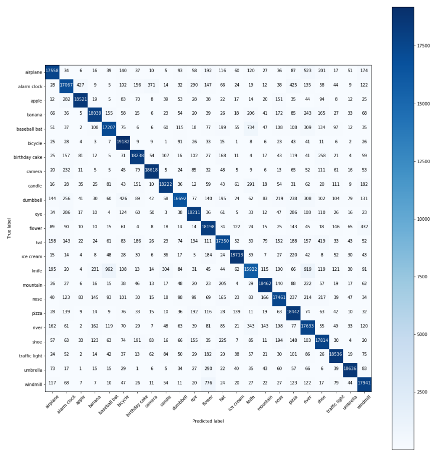

Fig. 3: Confusion Matrix for CNN scratch implementation

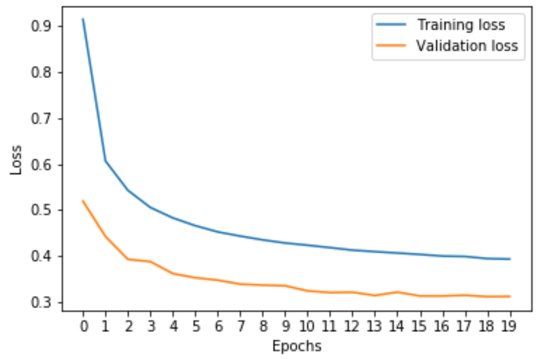

Fig. 4: Training and Validation Loss for CNN

TABLE III: Model Comparison using Accuracy, Precision, Recall and F1-Score

	Accuracy	**Precision**	**Recall**	**F1-Score**
KNN 1-CC (k=1)	59.16	59.36	59.16	59.26
KNN + Kmeans++	70.86	71.2	70.86	71.03
CNN (10 Epochs)	91.54	91.6	91.5	91.6
CNN (20 Epochs)	91.37	91.5	91.4	91.5

VI. Inferences

1-CC makes the assumption that all doodles in a category will be similar to each other but there are many different ways to draw a given object. Thus one type of misclassification comes from categories with multiple versions of the object. So we tried to rectify it with KNN with Kmeans++. The voting in KNN with Kmeans++ often ended up with ties so we extended that with weighted voting. Table II shows the change in accuracy over different values of k for different variations in KNN algorithm and it can be seen that accuracy improves by incorporating the above mentioned changes in KNN.

CNN based architecture is clearly able to beat the KNN based model by a significant margin. Using deeper architecture may result in better accuracy. Also, applying an LSTM over the strokes based dataset intutively should give better performance which will be tested in the final milestone. Table III shows that the performance of different methods implemented.

References

[1] Lu, W. and Tran, E. (2017). Free-hand Sketch Recognition Classification.

[2] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? ACM Trans. Graph. (Proc. SIGGRAPH), 31(4):44:1 44:10, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[4] Kim, J., Kim, B. S. and Savarese, S. (2012). Comparing image classification methods: K-nearest-neighbor and support-vector machines. Ann Arbor, 1001, 48109-2122.

[5] Ha, D. and Eck, D. (2017). A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477.