Recognition of Labels for Hand Drawn Images

Darshan Kothawade[1], Ibtisham Kazi[2], Jatin Mhatre[3], Kavitha Nair[4], Dr. Prashant Nitnaware[5]

[1,2,3,4]B.E. Computer Engineering Student, Pillai College of Engineering, Navi Mumbai, Maharashtra, India
[5]Professor, Dept. of Computer Engineering, Pillai College of Engineering, Navi Mumbai, Maharashtra, India

---***---

Abstract - *People have a brilliant capacity to perceive freehand sketch drawings in spite of their theoretical and meager structures. Understanding freehand sketches with automated methods is a challenging task due to the diversity and abstract compositions of these sketches. In this project, we aim to develop an efficient freehand sketch recognition scheme, which is based on Convolutional Neural Networks (CNNs). Specifically, we seek to create a Keras model to classify sketches using Google's 'Quick, Draw!' dataset, which contains more than 50 million drawings across 345 categories. Further, we aim to integrate a custom model to an Android app using TensorFlow Lite. Such a system will be of great value to a variety of applications, such as human-computer interaction, sketch-based search, game design, and education.*

Key Words: Free Hand Sketch, Quick Draw, Image Processing, Convolutional Neural Networks, Sketch Recognition

1. INTRODUCTION

In November 2016, Google released an online game titled Quick, Draw! that moves players to draw a given object in under 20 seconds. However, this is no ordinary game; while the user is drawing, an advanced neural network attempts to guess the category of the object, and its predictions evolve as the user adds more and more detail. Beyond just the scope of Quick, Draw!, the ability to recognize and classify hand-drawn doodles has important implications for the development of artificial intelligence at large. For example, research in computer vision and pattern recognition, especially in subfields such as Optical Character Recognition (OCR), would benefit greatly from the advent of a robust classifier on high noise datasets. For the purposes of this project, we choose to focus on classification of the finished doodles in their entirety. While a simpler premise than that of the original game's, this task remains difficult due to the large number of categories (345), wide variation of doodles within a category, and confusing similarity between doodles across other categories. Thus, we create a multi-class classifier whose input is a Quick, Draw! doodle and whose output is the predicted category for the depicted object [1].

2. LITERATURE SURVEY

A. Quick, Draw! Doodle Recognition: The paper was developed by the author Kristine Guo, James WoMa, and Eric Xu. In this paper, a multi-class classifier was built to assign hand-drawn doodles from Google's online game Quick, Draw! into 345 unique categories. Multiple variations of k-nearest neighbors and a convolutional neural network were implemented and compared which achieved 35% accuracy and 60% accuracy, respectively [1].

B. Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using TensorFlow and Keras: The paper was developed by the author Karan Chauhan, and Shrwan Ram. In this paper, a large number of different images, which contain two types of animals, namely cat and dog are used for image classification. Four different structures of CNN are compared on CPU systems, with four different combinations of classifiers and activation functions. For Binary image classification, combination of sigmoid classifier and Relu activation function gives higher classification accuracy than any other combination of classifier and activation function [2].

C. Feature-level fusion of deep convolutional neural networks for sketch recognition on smartphones: The paper was developed by the author E. Boyaci, and M. Sert. In this paper, feature-level fusion is implemented that use deep convolutional neural networks (CNNs) for recognizing hand-free sketches and develop a sketch recognition application for smartphones based on client-server application architecture. Results on TU-Berlin hands-free sketch benchmark dataset show that, feature-level fusion scheme achieves a recognition accuracy of 69.175%. This outcome is promising when contrasted and the human acknowledgment exactness of 73.1% on the equivalent dataset [3].

D. Free-hand Sketch Recognition Classification: The paper was developed by the author Wayne Lu, and Elizabeth Tran. In this paper, a publicly available dataset of 20,000 sketches across 250 classes from Eitz et al. is used, Convolutional neural networks (CNNs) is applied in order to improve performance to increase the recognition accuracy on sketches drawn by different people. The effects of several hyperparameters on overall performance are analyzed using a residual network (ResNet) approach [4].

E. Hand Drawn Sketch Classification Using Convolutional Neural Networks: The paper was developed by the author Habibollah Agh Atabay. In this paper, the accuracy of sketch image classification is improved by training a few deep CNNs. The size of inputs in the currently used architectures of CNNs is greater than 200×200 pixels which has limited the accuracy of classification. Input is given in the form of tiny images, thus the architecture of CNNs are simplified and thus be trained in a reasonable time, in CPU mode and increase the speed of training [5].

Table -1: Summary of Literature Survey

Literature	Observations
Kristine Guo et al. 2018 [1]	Quick Draw! dataset contains numerous preprocessed images of various categories that will help in making an efficient model.
Karan Chauhan et al. 2018 [2]	The comparison between different neural networks will help in the optimal selection of the neural network that will be used to train the model with the help of TensorFlow and Keras.
E. Boyaci et al. 2017 [3]	Understanding various datasets for the implementation of the project.
Wayne Lu et al. 2017 [4]	Better understanding of CNN will help in an efficient model creation.
Habibollah Agh Atabay et al. 2016 [5]	Studying CNN on different datasets will help in analyzing the model thus resulting in better outcomes.

3. PROPOSED WORK

We have proposed a Convolutional Neural Network (ConvNet/CNN) based sketch recognition model which is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image to differentiate one from the other. The pre-processing required in a ConvNet is a lot lower when contrasted with other classification algorithms.

In this project, we explore four different convolutional network architectures. The basic architecture consists of an initial 7x7 convolutional layer. This layer is then followed by a series of 12 3x3 residual units, for a total of 25 convolutional layers (not including layers used for residual projection). Every third residual unit, the feature map size is halved by increasing stride while the number of filters is doubled. At the end of the network, global average pooling is used and followed by a fully connected layer to output logits for softmax cross-entropy loss. Dropout is applied on 1) the initial input, 2) every third residual unit, and 3) before the fully connected layer [6][7].

3.1 System Architecture

We implement a convolutional neural network (CNN), a state-of-the-art model known for being able to recognize and quickly learn local features within an image. For a 28 × 28 × 1 doodle, we first run the image through two convolutional filters. Furthermore, we add zero padding border around the image so that the resulting outputs have the same width and height. The output then goes through a max pooling layer with a kernel size of 2 × 2. Following this, we flatten the tensor and feed the result through two fully-connected or dense layers. Each layer uses the ReLu activation function as well as dropout. The output then goes through one more affine transformation before we apply softmax to generate probabilities for each class.

Layers used to build the model architecture: Input, Dense, Dropout, Flatten, Conv2D, MaxPooling2D.

The system architecture is given in Figure 1. Each block is described in this Section.

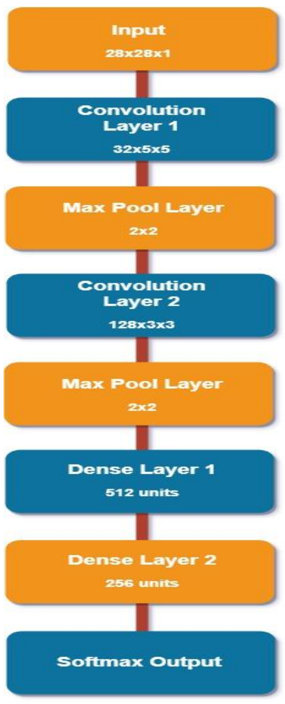

Figure 1: Proposed system architecture layout

3.1.1 Convolutional Neural Network

CNN image classifications take an input image, and classify it under certain categories after processing it. Computers see an input image in the form of an array of pixels. It depends on the image resolution. It will see h x w x d (h = Height, w = Width, d = Dimension) based on the image resolution. For instance, an image of 4 x 4 x 1 array of matrix of grayscale image and an image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values).

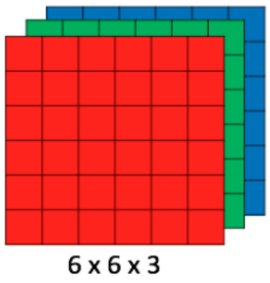

Figure 2: Array of RGB matrix

Each input image will pass it through a series of convolution layers with filters (kernels), pooling, fully connected layers and then apply Softmax function to classify an object with probabilistic values between 0 and 1. The complete flow of CNN to process an input image and classify the objects is explained in the below figure.

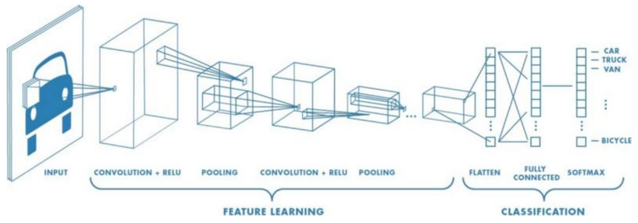

Figure 3: Neural network with many convolutional layers

3.1.2 Convolution Layer

Convolution layer extracts features from an input image. The relationship is preserved by Convolution between pixels by learning image features using small squares of input data. Two inputs such as image matrix and a filter or kernel are taken by a mathematical operation.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

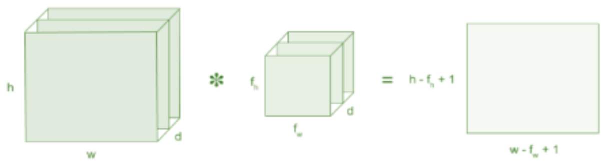

Figure 4: Image matrix multiplies kernel/filter matrix

3.1.3 Strides

Stride is the number of pixel shifts over the input matrix. We move the filters to 1 pixel at a time when the stride is 1. We move the filters to 2 pixels at a time when the stride is 2 and so on. The convolution would work with a stride of 2 as it is shown in the below figure.

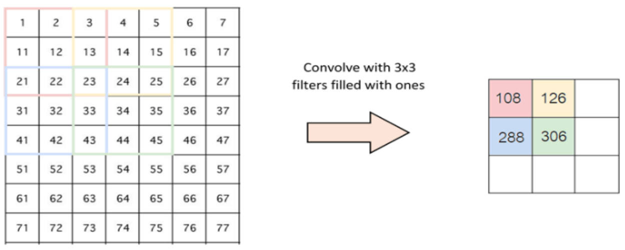

Figure 5: Stride of 2 pixels.

3.1.4 Padding

Sometimes the filter does not fit perfectly with the input image. We have two options:

1. For the picture to fit, pad it with zeros (zero-padding).
2. The part of the image can be dropped where the filter did not fit. This is called valid padding which keeps only the valid part of the image.

3.1.5 Non Linearity (ReLU)

ReLU - Rectified Linear Unit for a non-linear operation. The output is

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

ReLU's purpose is to present non-linearity in our ConvNet. As, the real world data would want our ConvNet to learn would be non-negative linear values.

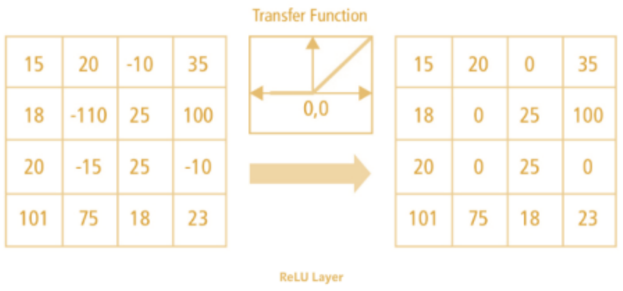

Figure 6: ReLU operation

3.1.6 Pooling Layer

When the images are too large the number of parameters would be reduced by the Pooling layers section. Spatial pooling is also referred to as subsampling or downsampling which reduces the dimensionality of each map but retains the important information.

Types of Spatial pooling:

- Max Pooling
- Average Pooling
- Sum Pooling

The largest element is then taken by Max pooling from the rectified feature map. The average pooling can also be taken by taking the largest element. Sum pooling is the sum of all elements in the feature map.

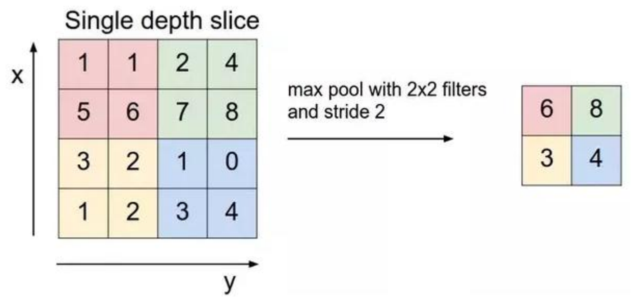

Figure 7: Max Pooling

3.1.7 Fully Connected Layer

The layer we call the FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

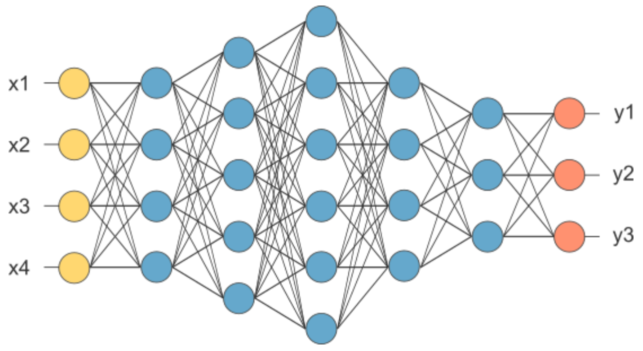

Figure 8: After pooling layer, flattened as FC layer

Feature map matrix will be converted as vector (x1, x2, x3, …) in the above diagram. We combined these features together to create a model, with the fully connected layers. Finally, we classify the outputs using an activation function such as softmax or sigmoid.

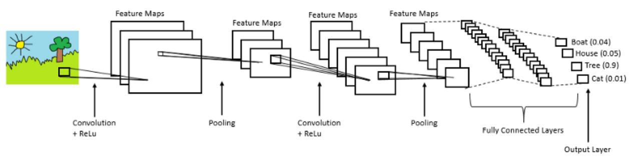

Figure 9: Complete CNN architecture

4. PERFORMANCE AND EVALUATION

The implementation detail is given in this section.

4.1 Dataset and Parameters

Google publicly released a Quick, Draw! dataset containing over 50 million images across 345 categories. There are multiple different representations for the images. One dataset represents each drawing as a series of line vectors, and another contains each image in a 28x28 grayscale matrix. We use the latter version of the dataset because we focus on classification of the entire doodle in this project. We treat each 28x28 pixel picture as a 784-dimensional vector.

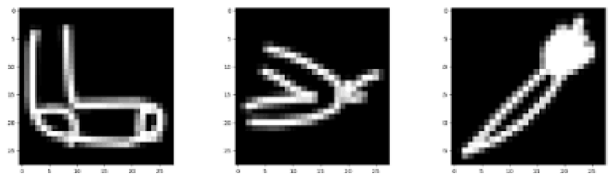

Figure 10: Sample doodles of a sock, elbow, and carrot (left to right) from the training dataset

To test our models, we split the data into two different folds: 80% for training and 20% for testing. To reduce computation time and storage of the data, we decided to create a smaller subset of the original dataset by randomly sampling 10% of the drawings from each category. As a result, we obtain approximately 4,000 examples for the training set and 1,000 examples for the testing set.

While raw accuracy is a good measure of a model's performance, it penalizes harshly for an incorrect prediction (wrong predictions receive 0 points and right predictions receive 1 point). Since we have so many categories, including some that are extremely similar such as "cake" and "birthday cake", we evaluate our methods not only with raw accuracy but also with a scoring metric that is more lenient of incorrect predictions. We use the top_k_categorical_accucary metric provided by Keras which calculates the top-k categorical accuracy rate, i.e. success when the target class is within the top-k predictions provided [1].

4.2 Evaluation Parameters

While raw accuracy is a good measure of a model's performance, it penalizes harshly for an incorrect prediction (wrong predictions receive 0 points and right predictions receive 1 point). Since we have so many categories, including some that are extremely similar such as "cake" and "birthday cake", we evaluate our methods not only with raw accuracy but also with a scoring metric that is more lenient of incorrect predictions. We use the top_k_categorical_accucary metric provided by Keras which calculates the top-k categorical accuracy rate, i.e. success when the target class is within the top-k predictions provided.

4.3 Performance Evaluation

To achieve the best performance for the CNN model, we tuned various hyperparameters including the number of units in each dense layer, dropout rate, and learning rate. Overall, we found that the model producing the best prediction had two dense layers with 512 and 256 units with each layer having a dropout rate of 0.2. Furthermore, we trained our model with a learning rate of 0.001 and batch size of 256 across 10 epochs.

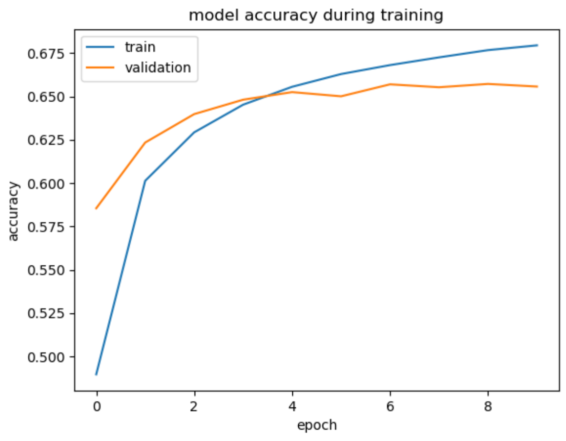

Fig 11: Training Accuracy

As seen from figure, the end architecture fits the data well as the validation accuracy has more or less converged after the 6th epoch.

Furthermore, following were the accuracies achieved on the testing dataset: 1. Final accuracy: 65.57% 2. top-3 accuracy: 82.71%

5. RESULT

Below is the demonstration of the Sketch Recognition application that was built using the model.

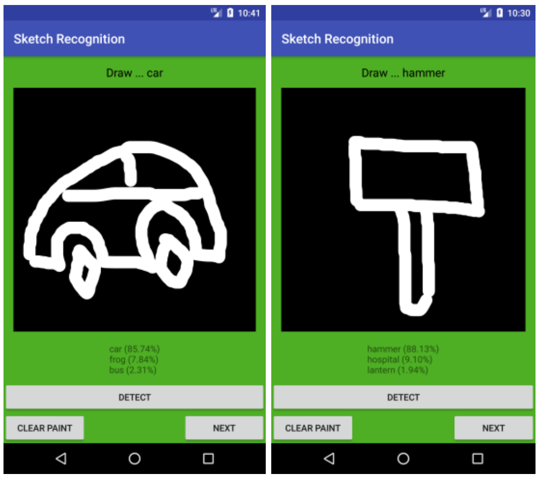

Fig 12: Sketch Recognition App

6. CONCLUSION

We have presented our CNN architecture for freehand sketch recognition. The different pooling layers, strides, padding, ReLu have been explained. The different hybrid approaches have also been described. The comparative study of various techniques mentioned above is presented in this report. The hybrid approach is proposed with a combination of different hybrid strategies for the development of an application with the help of convolutional neural networks by using Keras model, TensorFlow lite. Evaluation parameters which are used to evaluate the performance and accuracy of the system are described within the report. The standard dataset or variable inputs are defined that may be used in an experiment for this system. The two datasets identified for the experiments are 'Quick, Draw!' and 'TU-Berlin sketch'. The learned sketch feature representation could benefit other sketch-related applications such as sketch-based image retrieval and automatic sketch synthesis, which could be interesting venues for future work.

We would like to experiment with advanced CNN architectures such as VGG-Net and ResNet, which have already reached state-of-the-art levels of image classification performance, although not for sketches in particular. Additionally, we have only used approximately 10% of the total Quick, Draw! dataset, and we believe training our models on the complete dataset would improve accuracy.

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our supervisor Dr. Prashant Nitnaware for the valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of this work. We deeply express our sincere thanks to our Head of the Department Dr. Sharvari Govilkar and our Principal Dr. Sandeep M. Joshi for encouraging and allowing us to present this work.

REFERENCES

[1] Kristine Guo, James WoMa, Eric Xu, "Quick, Draw! Doodle Recognition" Stanford University, 2018.
[2] Karan Chauhan, Shrwan Ram, "Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using TensorFlow and Keras", M.B.M. Engineering College Jodhpur, India, International Journal of Advance Engineering and Research Development (IJAERD), 2018.
[3] E. Boyaci and M. Sert, "Feature-level fusion of deep convolutional neural networks for sketch recognition on smartphones," 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2017.
[4] Wayne Lu, Elizabeth Tran, "Free-hand Sketch Recognition Classification", Stanford University, 2017.
[5] Habibollah Agh Atabay, "Hand Drawn Sketch Classification Using Convolutional Neural Networks", Gonbad Kavous University, Iran, 2016.
[6] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
[7] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53