PICASSO- A Smart AI Guessing Game

Rachita V Nayak[1], Madhurika R Upadhya[1], Kruthika K R[1], Hema Jagadish[2]

[1]Student, Dept. of Information Science and Engineering, Bangalore Institute of Technology, Bengaluru, Karnataka, India
[2]Assistant Professor, Dept. of Information Science and Engineering, Bangalore Institute of Technology, Bengaluru, Karnataka, India

---

Abstract – *Sketch recognition is one of the emerging technologies in machine learning built for games or to study psychological behavior of different people based on their drawings. Picasso is an application that predicts the category of an object being drawn in front of a computer. The categories chosen are simple everyday objects that can be drawn with simple hand gestures and movements. The proposed model uses convolutional neural networks (CNN) to recognize hand drawn sketches and predict a suitable output.*

CNNs take in a training dataset and classify it into labels based on unique features. A new input can then be classified using the learned features. The training model was built using Keras which is an open-source neural-network library written in Python. The dataset contains thousands of drawings for 15 classes whose features are extracted and classified by the CNN. The game is designed to be simple and effective in the demonstration of machine learning and artificial intelligence's power in the field of character recognition and game science. The system can lay groundwork for future research in the area of character recognition and predictive algorithms.

***Key Words*: Sketch Recognition, Convolutional Neural Network, Feature Extraction**

1. INTRODUCTION

Pictionary is a game wherein a player guesses the name of an object being drawn by the opponent. This game has been popular for many decades, it is often played with family and friends at parties. The game is usually played with multiple players with the name of the object secretly communicated to a member of the opponent team. A player is only allowed to draw the object or do gestures, any activity done otherwise such as telling out hints loudly or pointing to similar objects leads to either giving half points or no points to the performing team. The team with the highest points wins the game after playing equal number of rounds on both sides. A computer application that builds this game can either take in an input image, run the model and predict it or do it in real time that is, predict the drawing as the user draws. An example of a user's drawing of a street light is shown in Fig -1. As shown in the figure, a street light is a tall structure containing an ovular surface at the top. The light is represented as a set of short lines protruding from the oval. Additionally, two parallel lines can be drawn to make sure that the computer does not confuse the drawing to be a study lamp.

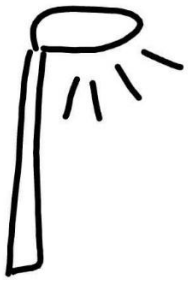

Fig -1: Pictionary drawing of a street light

Drawing inspiration from Pictionary, Picasso was designed to predict the name of a drawing by a computer program. In the proposed work, it is only the machine that guesses what a user draws, not the other way around.

1.1 Sketch Recognition

Sketch recognition is the automated recognition of hand-drawn diagrams by a computer. Research in sketch recognition lies at the crossroads of artificial intelligence (A.I) and human computer interaction (HCI). Recognition algorithms usually are gesture-based, appearance-based, geometry-based, or a combination thereof. Sketches that are input to a computer program for categorization and prediction are usually used in the field of behavioral science to study human behaviors on how they draw a particular sketch. There are instances where people belonging to different countries draw the same object in different ways. This is because of certain environmental factors on how they are taught to interpret objects around them.

In A.I, a machine learning model used for recognizing sketches is in a continuous cycle of learning as more input is given. The efficiency of the program keeps increasing as more data is added and there are less chances of the model failing to map the learned features to the available labels. There is also reduction in false positive cases as more distinct features can be added to a class when the size of the dataset increases. Sketch recognition models commonly employ gesture recognition, multi-touch gestures, pen computing and contour detection technologies. The proposed model detects a contour object in front of the computer's web cam and tracks the drawing.

1.2 Existing Systems

Guo et al. [1] proposed classification models for classifying Google's Quick, Draw! Dataset. The dataset contains 50 million drawing across 345 categories and, images from each dataset were sampled. Their k-nearest neighbour ++ model produced an accuracy of 35% while the CNN model produced 60% accuracy. Although images from all categories were classified, only 1% of images from each dataset were used considering time and complexity constraints.

Seddati et al. [2] presented DeepSketch model for sketch classification which reached an accuracy of 75%. The model used deep convolutional networks (ConvNets) for hand sketch recognition. The features learned by ConvNets are useful for similarity search.

1.3 Problem Statement

To develop a real-time machine learning system that predicts the class of an object being drawn.

Therefore, the aforementioned problem is to:

- Extract and classify features of 15 classes of images by training a model based on CNN.
- Accept a video frame object from a computer's web camera, detect a contour of interest (a blue pen) and track its movements.
- Predict the class of the object after the user finishes drawing.
- Derive the accuracy over the training set and validation set using any reliable and popular analyzing tool.

2. DATASET

There are many datasets available that are open sourced online [3] for the purpose of developing deep learning models to recognize sketches or drawings. Some of them are Sketch Dataset by Elitz et al., Quick, Draw! dataset by Google, Sketch Me that Shoe by researchers from Queen Mary London University and, Sketchy dataset by researchers from Georgia Tech, USA. These datasets contain drawings that span across many categories. After comparing all the datasets to suit the categories decided, Quick, Draw! dataset was selected to train the model. The Quick, Draw! dataset consists of 50 million drawings across 345 categories.

The proposed work presents a CNN model for 15 classes- Apple, Bow, Door, Cycle, Camera, Envelope, Fish, Ice Cream, Mermaid, Circle, Mountain, Pineapple, Rainbow, Shoe and Star. These categories are shown in Fig -2.

The categories were chosen making sure that both simple and complex drawings can be predicted by the model being built. Some of the categories also have certain similarities in features so that the trained model can learn better.

Fig -2: 15 categories of objects

Each image is represented as a time-stamped vector which contain stroke order information and some metadata. Strokes captured at every time interval and their pixel positions are stored in an array. The metadata includes what the player was asked to draw and in which country the player was located. The raw data is available in ndjson format. Google has pre-processed and split the dataset into different files and formats to make it faster and easier to download and explore. The dataset is available as simplified drawing files, binary files and numpy bitmaps.

The proposed work uses drawings in the form of numpy bitmaps to train the CNN model. Numpy files contain numpy array/s and are stored with the extension .npy. All the simplified drawings were rendered into a 28x28 grayscale bitmap in numpy .npy format. The files can be loaded with np.load(). Every category has its own numpy file so, 15 numpy files were used for the dataset. Each image in the file is treated as a 784-dimensional vector that is, the shape of every numpy array representing one image is (784,). The number of images in every category range from 120451 to 162645. The graphical representation of a random numpy array from the .npy file for apple category that is plotted using Python module matplotlib's functions is shown in Fig -3.

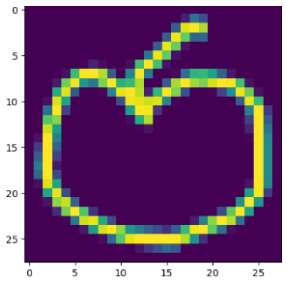

Fig -3: An image from apple class

3. METHODOLOGY

The problem is approached by using a convolutional neural network (CNN) model that goes through feature learning and feature classification. The training model consists of neural network layers, each made up of neurons with weights assigned to them. CNNs use little

pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand engineered. This independence from prior knowledge and human effort in feature design is a major advantage. CNNs are widely used in image and video recognition, recommender systems and natural language processing.

3.1 Backpropagation Algorithm

Backpropagation is an algorithm for supervised learning of artificial neural networks using gradient descent. The algorithm is used to effectively train a neural network through a method called chain rule. Calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

A back propagation neural network produces a local optimized solution, but not a global optimized solution. However, from a general perspective, the result produced by a back-propagation algorithm is usually a satisfactorily optimized solution. Back propagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and speech recognition. The following algorithm provides the basis of how backpropagation works.

Algorithm: BACKPROPAGATION (D, η)
begin
Input: D = $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1 \text{ to } n}$, learning rate η
Randomly initialize all weights and threshold

repeat
for all $(x^{(i)}, y^{(i)}) \in D$ **do**
compute $y^{(i)}{}_j$ according to current parameter
compute $\delta_{\beta j}$
compute $\delta_{\alpha j}$
compute update w_{ji}, v_{kj}
end for
until achieve stopping condition
end

3.2 Adam Optimization Algorithm

The proposed work uses Adam optimization algorithm which is an extension to stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. The Adam optimization algorithm leverages the power of adaptive learning rates methods to find individual learning rates for each parameter. The algorithm also combines RMSprop which maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight. It can handle sparse gradients on noisy problems and uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. Nth moment of a random variable is defined as the expected value of that variable to the power of n:

$$m_n = E[X^n] \quad \text{.....Eq. 1}$$

m-moment, X-random variable

3.3 Activation Function

Neural network activation functions are a crucial component of deep learning. Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model which can make or break a large-scale neural network. Activation functions also have a major effect on the neural network's ability to converge and the convergence speed, or in some cases, activation functions might prevent neural networks from converging in the first place. The activation functions used in the system put forward are ReLU (Rectified Linear Units) and Softmax. ReLU is linear (identity) for all positive values, and zero for all negative values. The softmax function is usually used to compute losses that can be expected when training a data set

4. IMPLEMENTATION

Using the OpenCV module in Python, a video can be read either by using the feed from a web camera connected to a computer or by reading a video file. OpenCV module is a library of programming functions mainly aimed at real-time computer vision. The first step towards reading a video is to create a VideoCapture object. Its argument can be either the device index or the name of the video file to be read. The input is read in the form of a video through the user's webcam. The user draws an object in the air using a blue pointer. The model then tracks the user's gestures to get the drawing. The argument passed in this case will be the index of the system's default camera. The program tracks the drawing in real-time.

4.1 Color Space Conversion

The color space of the frames read by VideoCapture() are converted to another color space. Color spaces are different types of color modes, used in image processing and signals and system for various purposes. Color space conversion is the translation of the representation of a color from one basis to another. The image sequences are captured with BGR color space by default. BGR (Blue, Green, Red) is the same as RGB except that the order of areas is reversed. Red occupies the least

significant area, green the second and blue the third. The BGR color space is converted to HSV (Hue Saturation Value) space. This is achieved by a function of OpenCV module called cvtColor(). The function takes an image as input and converts it into another color space which is mentioned using any of the legal color space conversion codes. There are more than 150 color space codes available by OpenCV. For the proposed work, the code used was COLOR_BGR2HSV. Fig -4 illustrates BGR to HSV conversion.

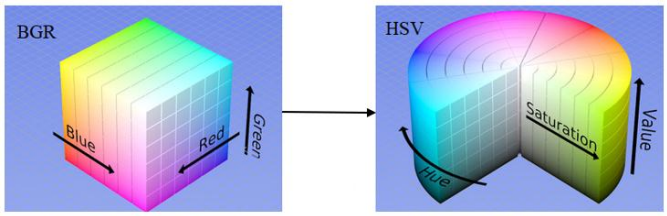

Fig -4: BGR to HSV

4.2 Image Filtering and Tracking the Drawing

In order to recognize the gestures being drawn in the air, the application makes use of contours. A contour is used to detect a blue pointer. The user can use a blue colored pen for this purpose. Contours can be described simply as a curve joining all the continuous points (along the boundary), having same color or intensity. Contour is closely related to the concepts of edge and boundary, which correspond to the discontinuities in photometrical, geometrical and physical characteristics of objects in images.

In the proposed work, a mask is created before identifying the contour object and tracking the drawing. The mask goes through the two morphological functions-erosion and dilation. Morphology is a part of image filtering where images are processed based on shapes. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values. The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

Erosion and dilation are carried out by OpenCV methods erode() and dilate(). They take the mask as input and the number of iterations. The next step is to apply findContours( ) method to the mask to get the final contour. The blue pen is now detected and tracked for a drawing. OpenCV offers four shapes for representing a contour- bounding rectangle, minimum enclosing circle, ellipse and line. The proposed work uses a minimum enclosing circle. After the contour of interest is detected, the coordinates of each and every point the canter of the contour are tracked along with its color. These set of points of different colors are stored in different deques.

4.3 Convolutional Neural Network Model

The proposed work performs a real time recognition of a user's drawing. It tracks the strokes at every second and simultaneously runs the Convolutional Neural Network (CNN) model. While doing so, there are chances that the category assigned may be right or wrong. The user draws an object in a continuous set of strokes and after they stop drawing, the final category is predicted. The CNN model makes use of five different layers-convolutional, maxpool, dropout, flatten and dense layers. The initialized features of each of the categories and their labels are first loaded to help train the model. These features and labels are extracted by reshaping the dataset using NumPy module. An overview of the CNN model used by the proposed work is shown in Fig -5.

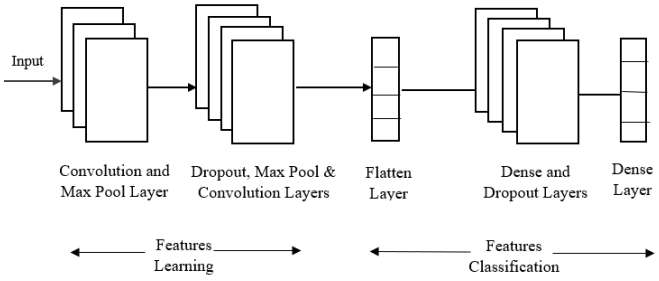

Fig -5: CNN Model

The framework is built using Keras module's sequential model. A sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. Each neural network layer is added on top of the other in a sequential manner. Since the proposed work takes only one input at a time and produces one output, the sequential model is used. Each layer is added using the add() method. This method allows one to mention various parameters such as the activation function, input shape etc. the first layers in a CNN detect large features that can be recognized and interpreted relatively easy. Later layers detect increasingly smaller features that are more abstract and are usually present in many of the larger features detected by earlier layers.

The last layer of the CNN is able to make an ultra-specific classification by combining all the specific features detected by the previous layers in the input data. The dataset goes though features learning and classification to finally label the features into 15 categories or labels.

4.4 Training the Model

The CNN model was built using Keras. Back propagation supervised learning algorithm which was mentioned previously was used to train the model. The dataset is split into training and validation set with a test size of 0.1. The batch size used in the proposed work is 64 so, the dataset is trained for 64 images at a time. The features and labels are loaded by a program that performs some

modifications to the dataset and stores them as pickle files. The pickle files are then used to split the dataset into training set and validation set. The CNN model is then trained and is stored as a HD5 file. The proposed work uses this trained model to make predictions. The model is trained only once and can be used any number of times.

5. RESULTS

The results of the training model over training set and validation set were analyzed using TensorBoard which is a visualization toolkit provided by TensorFlow. Fig -6 shows the accuracy (96.69%) over the training set. Table -1 shows the accuracy over training set and validation set for different epochs.

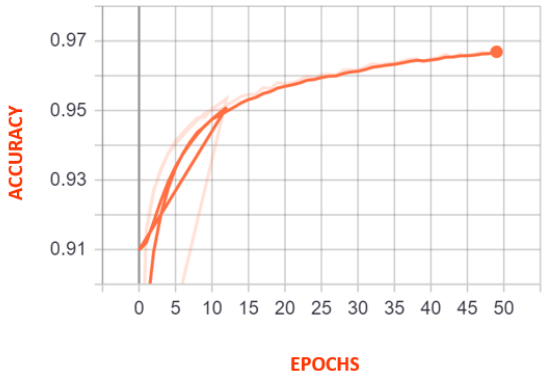

Fig -6: Accuracy over training set

Table -1: Accuracy over different epochs

Accuracy		
Epochs	**Training**	**Validation**
10	95	93.8
30	96.1	94.58
50	97	94.12

6. CONCLUSIONS

The proposed work presented a CNN model for classifying 15 classes of objects. The model takes input in the form of a video with one drawing an object in the air using a blue pointer and performs real-time prediction of the category of the drawing. After training for three different epochs, it was found that the model performed the best for 50 epochs. The accuracy of the trained model stands at 96.69% with a loss of 11.74%. The accuracy of the framework over 10% of the dataset used for validation stands at 94.12% with a loss of 22.32%. The accuracy of the model was calculated using TensorBoard toolkit. It was found out that on an average, 12 categories were predicted out of 15 for 10 runs of the program. The challenge faced when predicting categories was that the model sometimes detected a blue counter of interest in the background initially and then relied on the object in hand. The model was trained for 15 classes as opposed to the 345 categories that are available. In the future, better neural network architectures and hardware resources can be considered to enhance the game.

REFERENCES

[1] K. Guo, J. WoMa, E. Xu, "Quick, Draw! Doodle Recognition", Stanford University, 2018

[2] O. Seddati, S. Dupont, and S. Mahmoudi Deepsketch, Deep convolutional neural networks for partial sketch recognition. In Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on, pages 1–6. IEEE, 2016.

[3] Liping Yang, "Dataset collection for (deep) machine learning and computer vision", Deep Learning Garden, https://deeplearning.lipingyang.org/cvml-dataset-collection/, Accessed Feb 2020

[4] Tsung-Han Tsai, Po-Ting Chi, Kuo-Hsing Cheng," A Sketch Classifier Technique with Deep Learning Models Realized in an Embedded System", IEEE 2019, 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)