Spring 5-20-2019

An Empirical Comparison of Different Machine

Piyush Bajaj
San Jose State University

An Empirical Comparison of Different Machine Learning Algorithms for Classifying Sketches

By

Piyush Bajaj

Presented to

Professor Robert Chun

Department of Computer Science

San José State University

May 2019

The Project Committee Approves the title of the project

An Empirical Comparison of Different Machine Learning Algorithms for Classifying Sketches

By

Piyush Bajaj

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

Spring 2019

ABSTRACT

Sketching has been used by humans to visualize and narrate the aesthetics of the world for a long time. With the onset of touch devices and augmented technologies, it has attracted more and more attention in recent years. Recognition of free-hand sketches is an extremely cumbersome and challenging task due to its abstract qualities and lack of visual cues. Most of the previous work has been done to identify objects in real pictorial images using neural networks instead of a more abstract depiction of the same objects in sketch. This research aims at comparing the performance of different machine learning algorithms and their learned inner representations. This research studies some of the famous machine learning models in classifying sketch images. It also does a study of legacy and the new datasets to classify a new sketch through various classifiers like support vector machines and the use of deep neural networks. It achieved remarkable results but still lacking behind the accuracy in the classification of the sketch images.

Keywords - Sketches, drawing, datasets, machine learning algorithms, neural networks.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Robert Chun for his ongoing support and guidance through this thesis. I would also like to thank other members of the committee, Dr. Chris Tseng and Mr. Rajesh Pradhan, for teaching me the core skills necessary for my project to succeed and review. Finally, I want to thank my parents for the patience and advice they have given me throughout my life.

TABLE OF CONTENTS

I. Introduction 8
II. Related Work 10
III. Datasets 12
IV. Data Preprocessing 16
V. Random forest classifier 18
VI. KNN 23
VII. Neural Networks 28
VIII. Multi-Layer Perceptron 34
IX. Convolutional Neural Networks 36
X. ResNet 41
XI. MobileNet 46
XII. Results 48
XIII. Conclusion 50
References 52

LIST OF FIGURES

Fig 1. Small subset of QuickDraw sketches. 13
Fig 2. Coordinate system for drawing stroke. 14
Fig 3. Size Normalization of the sketches 16
Fig 4. Grayscale and Color encoded sketches 17
Fig 5. Random Forest Algorithm 18
Fig 6. Random Forest Result graph 22
Fig 7. Example of k-NN classification 24
Fig 8. Euclidean Distance formula. 25
Fig 9. KNN Result Graph. 27
Fig 10. Neural network with 2 layers. 30
Fig 11. Activation function ReLU as a function of x 31
Fig 12. Simplest kind of feed-forward network 34
Fig 13. MLP Result Graph 35
Fig 14. A convolutional layer with a kernel 37
Fig 15. Excerpt of a CNN architecture 37
Fig 16. Back propagation in ResNet 42
Fig 17. Indexed results for objects in ResNet 45
Fig 18. Different Convolutional Filters 46

LIST OF TABLES

Table 1. QuickDraw attribute types 14
Table 2. QuickDraw raw data 15
Table 3. Top 3 categories for Prediction ensemble in CNN 40
Table 4. Table for id and output result mapping in ResNet.....40
Table 5. A small subset of the layers in ResNet 34 43
Table 6. A small subset of the layers in ResNet 50 43
Table 7. A subset of values of epochs in ResNet.....44
Table 8. Top 3 categories for Prediction ensemble in ResNet 44
Table 9. Table for id and output result mapping in ResNet.....45
Table 10. Comparisons of results of different Machine learning algorithms 49

I. INTRODUCTION

Sketching is a conventional means of visual communication often used for conveying rough visual ideas in architectural sketches, design studies, comics, or movie storyboards. Humans are incredible when it comes to representing real-world objects and phenomena in simple sketches since primitive times. It has been used for thousands of years to depict ideology and phenomena through since sketched objects. Humans are incredibly accurate in predicting and interpreting human drawn sketches, which are an impoverished version of the actual image representation. Machines significantly lag in accurately predicting the subject of these sketches. It is because of the properties of the sketched object, where its shape and proportion may be substantially different from its corresponding real object. Same object can be drawn in an infinite number of ways depending on the person's imagination and his artistic style and way of depicting it. The recognition of these sketches by machines thus becomes more interesting and challenging than other sectors of image classification.

An important research area in computer vision application is the classification of objects in the human-drawn sketches. Convolutional Neural Networks (CNN) provides a good base neural network model for features recognition for different use cases in the analysis of the images. However, there is little research that utilizes and compares various sets of machine learning algorithms in the areas of sketch-based object classification.

Machine learning can significantly improve and optimize sketch-based object recognition and classification. By learning from different kinds of sketches of the same object, machine learning can help the human mind to understand the simple yet incomplete depictions of the actual object. Over time, in this ongoing performance optimization process, machine learning can be refined to deliver increasingly accurate classification of the object in the sketches.

This research focuses on exploring the various existing machine learning algorithms to predict the categorization of the sketches. This survey uses references from published papers, conference proceedings, and some online articles. The research area which is not addressed yet is the proper comparison of time, performance and accuracy and their tradeoff across various machine learning models. The focus is to analyze and compare different machine learning algorithms and to ensure that the most relevant category gets predicted in the least amount of time and obtains the desired accuracy of the prediction of the human-made sketches.

II. RELATED WORK

Machine learning concepts and algorithms have been designed and developed to predict the human sketches since a decade now. It goes back to the paper [2] where Dr. Paul designed a physical robot where its robotic arm used to draw a portrait by following its designed algorithm to mimic the reality as close as possible. Most of these kinds of work were crafted to replicate the digital photos instead of defining a vectorized model of the top of the algorithm.

Works on sketch recognition go back to the development of Sketchpad[9]. Ever since different computer vision approaches were used to gather improved outcomes in varied sectors of the approach. LaViola et al. [10] investigated the recognition of mathematical sketches. Li et al. [11] exploited local feature representations (using star graph-based ensemble matching strategy) and global structures to address local and global variations. They trained an SVM using bag-of-features (BOF) to select the highest N number of similar drawing categories to the input data.

In addition to the abstract nature of hand drawings (compared to original images), there is also another challenge related to the lack of available databases for model training and benchmarking. Eitz et al. [1] defined a taxonomy of 250 object categories. They gathered 20,000 unique sketches to form the first large scale dataset of human sketches

(TU Berlin benchmark). For sketch category recognition, they used local feature vectors (to encode distributions of sketches), a handful of features representations, and consisting of multiple levels of the support vector machines (SVM). It was shown in that paper that humans could achieve a 73% accuracy on the data set. Results on a classifier showed a 56% accuracy. Schneider et al. [12] worked on the model that Eitz et al. [1] presented and improved the conditions defined to make it more focused on relevant aspects than based on a person's intention. Later on, they used Scale Invariant Feature Transform (SIFT), Gaussian mixture model (GMM) based fisher vector and multi-class SVM to do sketch recognition.

In [6] the same data set was explored by applying CNN, where a CNN was trained to recognize sketches yielding an accuracy of approximately 75%, hence outperforming humans in the same task. The proposed technique makes use of the strokes' order to achieve such a high prediction.

III. Datasets

Wayne and Tran [3] discuss a dataset which comprised of twenty thousand images which were spread across around 240 sets of classes. It was accumulated via an online crowdfunding source from close to thousands of people. Images provided as 1111x1111 pixel PNG files. Every picture was resized to 128x128px using bilinear interpolation to make the dataset more manageable. There are 80 images divided into 48 training, 16 validation, and 16 test examples for each class. Additional samples were generated to augment the low number of training data for a cumulative of around 100 training data defined per class as they were turned upside down horizontally. A small portion of the training set was used for validation and testing examples which were again evenly spread across around 240 classes. The pictures were stored as the shade of grayscale and read into the memory. [4]

David and Eck discuss a dataset called Quick Draw provided by Google and was created through the game Quick, Draw. A user would need to draw an object in 20 seconds from a given category. The data set ranges across 345 categories and consists of more than 50 million sketches. The raw data contains information regarding the type of the drawn object, a time stamp of when the drawing created and the picture itself along with other information.

Though there is not a massive difference in the number of label categories in both the datasets, the number of images in the QuickDraw dataset hosts increased in scope and accuracy.

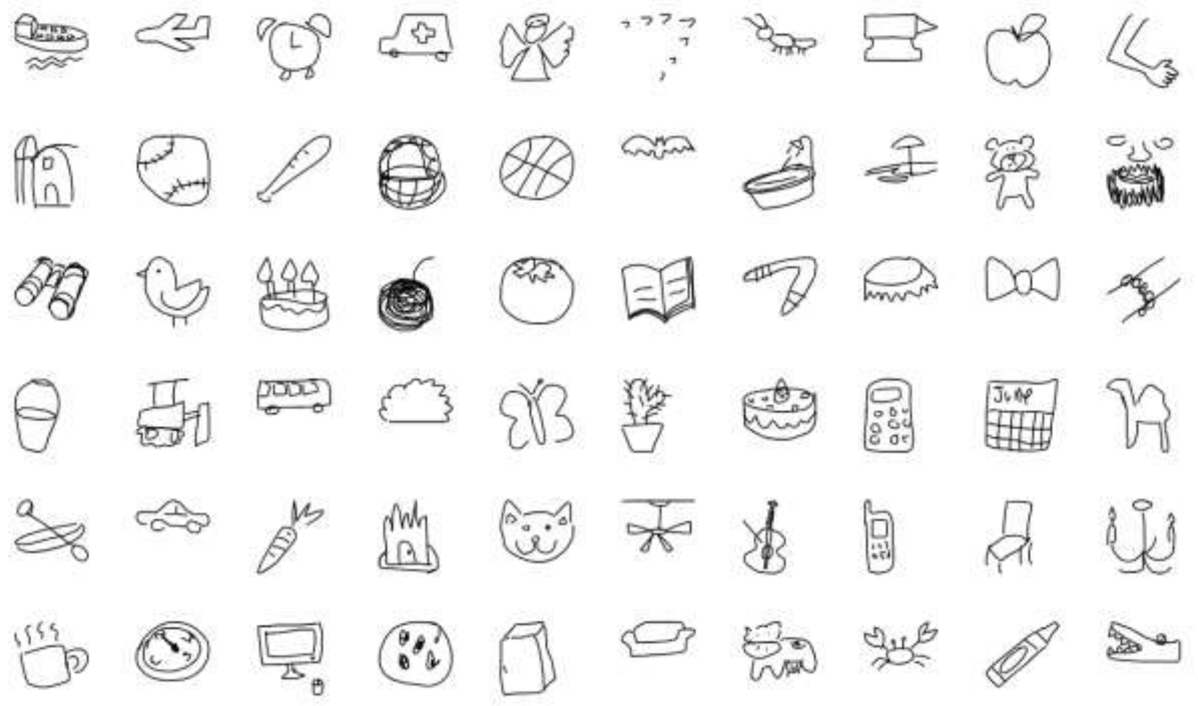


Fig 1. A small subset of QuickDraw sketches [20]

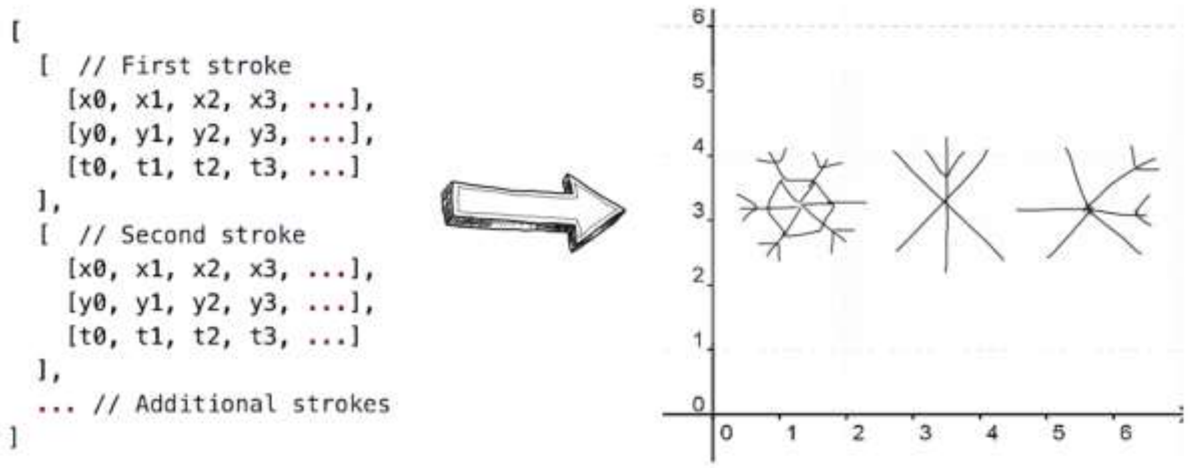
Each point in a stroke compares to an x-axis, y-axis, and time point. Each illustration accompanies explicit factors:

“**word**” - The drawing's class label

“**country code**”—The drawer's country of origin

“**timestamp**” - The timestamp of the drawing

“**recognized**” - The sign of the application's fruitful forecast "**drawing**" - Stroke-base information explicit for doodle pictures; each illustration is comprised of different strokes as grids



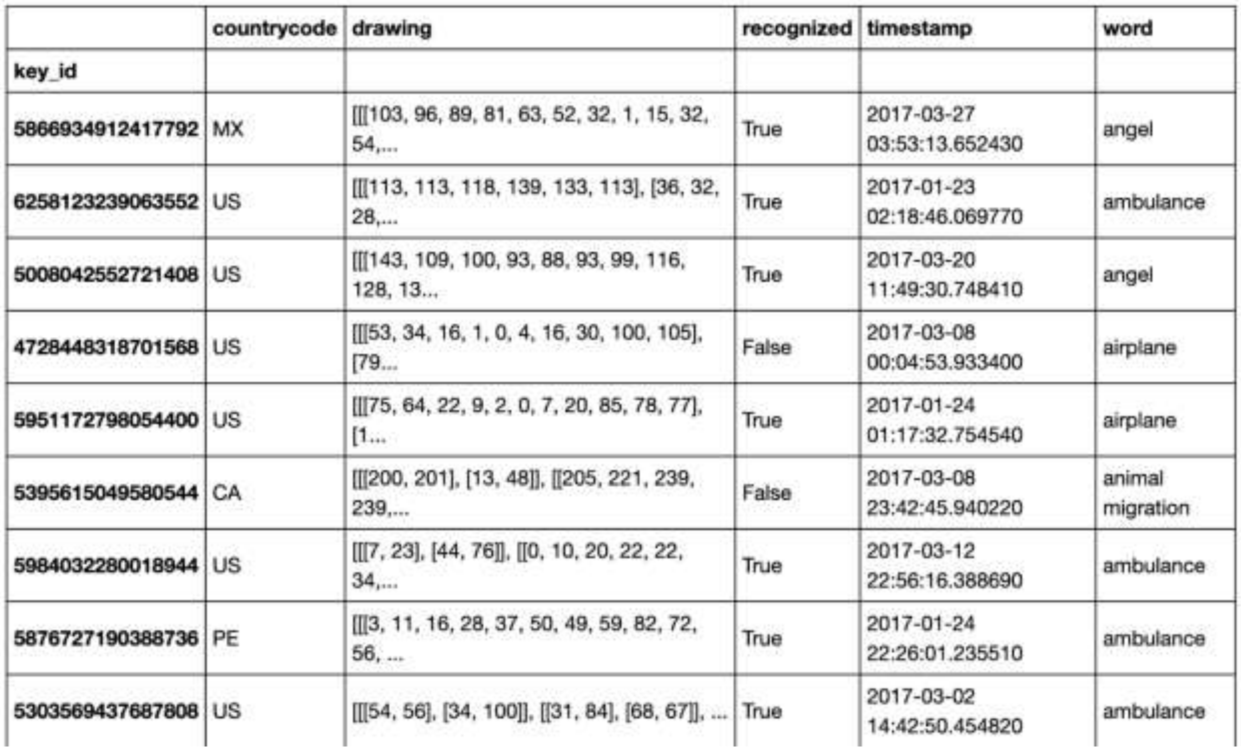
```
[
  [  // First stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  [  // Second stroke
    [x0, x1, x2, x3, ...],
    [y0, y1, y2, y3, ...],
    [t0, t1, t2, t3, ...]
  ],
  ... // Additional strokes
]
```

Fig 2. The coordinate system for drawing a stroke [20]

Key	Type	Description
Key _id	64-bit unsigned integer	A unique identifier across all drawings.
word	string	Category, the player, was prompted to draw.
recognized	Boolean	Whether the word recognized by the game.
timestamp	Date Time	When the drawing was created.
country-code	string	A two-letter country code (ISO 3166-1 alpha-2) of where the player was located.
drawing	string	A JSON array representing the vector drawing

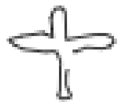
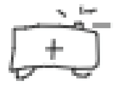
Table 1. QuickDraw attribute types.

This research project began by understanding the structure of the matrix or data that makes up a sketch and looked at ways to pre-process the data. Then, it delved into fitting some simple classifiers and a primary convolutional neural network, or CNN. From there, the work tackled CNN architectures such as ResNet and MobileNet.



	countrycode	drawing	recognized	timestamp	word
key_id					
5866934912417792	MX	[[[103, 96, 89, 81, 63, 52, 32, 1, 15, 32, 54,...	True	2017-03-27 03:53:13.652430	angel
6258123239063552	US	[[[113, 113, 118, 139, 133, 113], [36, 32, 28,...	True	2017-01-23 02:18:46.069770	ambulance
5008042552721408	US	[[[143, 109, 100, 93, 88, 93, 99, 116, 128, 13...	True	2017-03-20 11:49:30.748410	angel
4728448318701568	US	[[[53, 34, 16, 1, 0, 4, 16, 30, 100, 105], [79...	False	2017-03-08 00:04:53.933400	airplane
5951172798054400	US	[[[75, 64, 22, 9, 2, 0, 7, 20, 85, 78, 77], [1...	True	2017-01-24 01:17:32.754540	airplane
5395615049580544	CA	[[[200, 201], [13, 48]], [[205, 221, 239, 239,...	False	2017-03-08 23:42:45.940220	animal migration
5984032280018944	US	[[[7, 23], [44, 76]], [[0, 10, 20, 22, 22, 34,...	True	2017-03-12 22:56:16.388690	ambulance
5876727190388736	PE	[[[3, 11, 16, 28, 37, 50, 49, 59, 82, 72, 56, ...	True	2017-01-24 22:26:01.235510	ambulance
5303569437687808	US	[[[54, 56], [34, 100]], [[31, 84], [68, 67]], ...	True	2017-03-02 14:42:50.454820	ambulance

Table 2. QuickDraw raw data.

Recognized **Unrecognized** **Unrecognized** **Recognized**

These are some of the recognized and unrecognized samples in the QuickDraw dataset.

IV. DATA PREPROCESSING

The data for each of the class label of drawing exists in a format of separate CSV files. The CSVs were first shuffled. This process helped in making a good number of random data to craft the model in a way that it helps in eliminating bias.

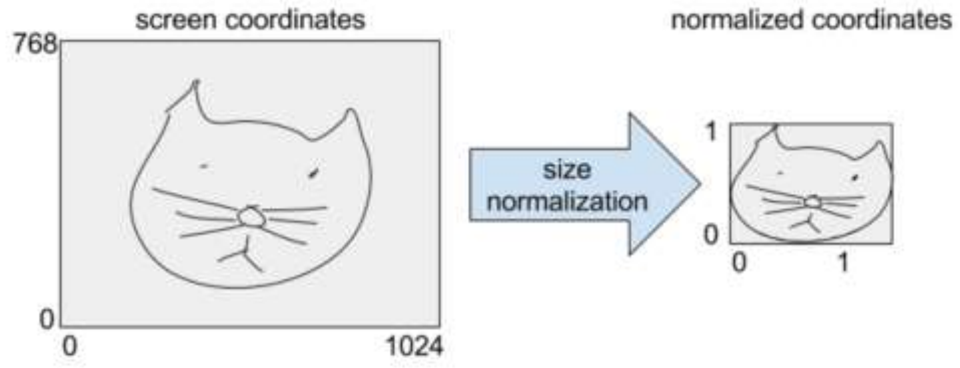


Fig 3. Size Normalization of the sketches [20]

People learn via seeing and observing things. The same phenomena apply when it comes to drawing sketches. People think in similar fashion of the common, occurring or day to day objects in life. When one is asked to draw a face, he or she would start with a circle or oval shape, and then draw two small inner circles to represent eyes and a small line in the middle to draw the nose and a horizontal oval to depict mouth of the person. For the model to recognize the differences between each stroke, this information was needed to be captured. Color encoding and gray-scale technique were quite useful when designing the CNN model. A color was assigned to each chronological stroke of a sketch; this helps the model in obtaining information on different strokes instead of just the actual whole image. To introduce noise into the images, they were spontaneously flipped, rotated and the capacity of the model was increased to handle noise. Both the OpenCV and Image Generator libraries of Keras were used, which helped in loading batches of raw input data from CSV files first and later transforming them into pictures. This methodology was used by both greyscale/color encoding and image augmentation in this research.

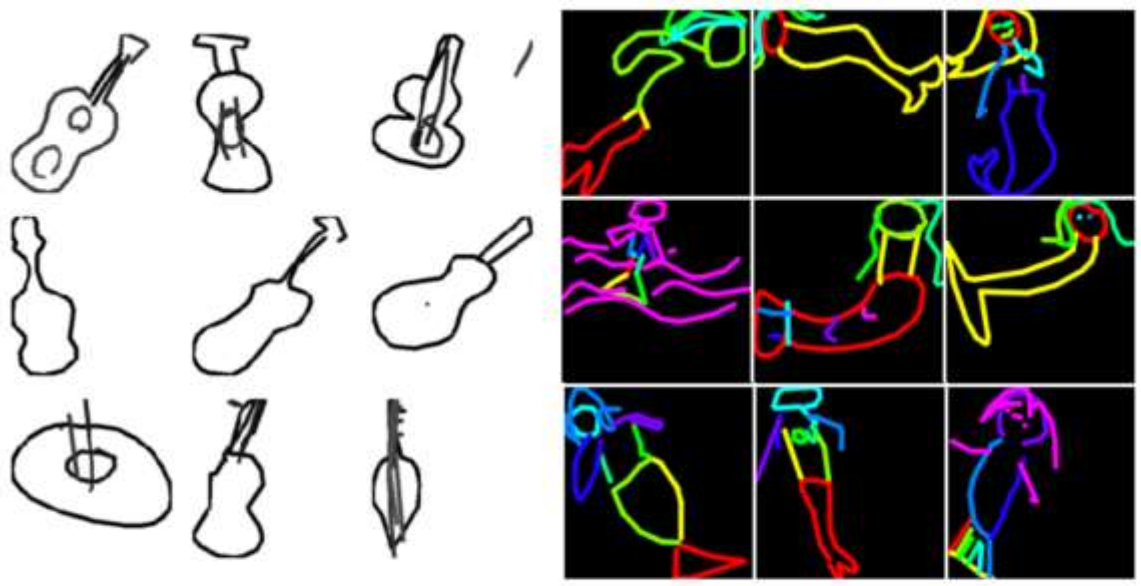


Fig 4. Grayscale and Color-encoded sketches [20]

V. RANDOM FOREST CLASSIFIER

Random forest methodology would probably be the first classification that would come in mind when we think of selecting and starting with a classifier for any machine learning model. It is a supervised learning algorithm. It is generally used for both regression and classification for its predictive learning models. It is one of the easiest and flexible technique to start with.

Random forests on randomly selected information samples create selection trees. Predictions are acquired from the trees, and it selects the quality consequences through voting. Trees are pruned with the aid of setting a hindering criterion for splitting nodes, and its goals to make the trees de-coupled. Various learning models are used in Ensemble models like Random Forest to get higher predictive solutions. To arrive at the viable solution, a large fleet of these de-coupled decision trees gets generated in this model.

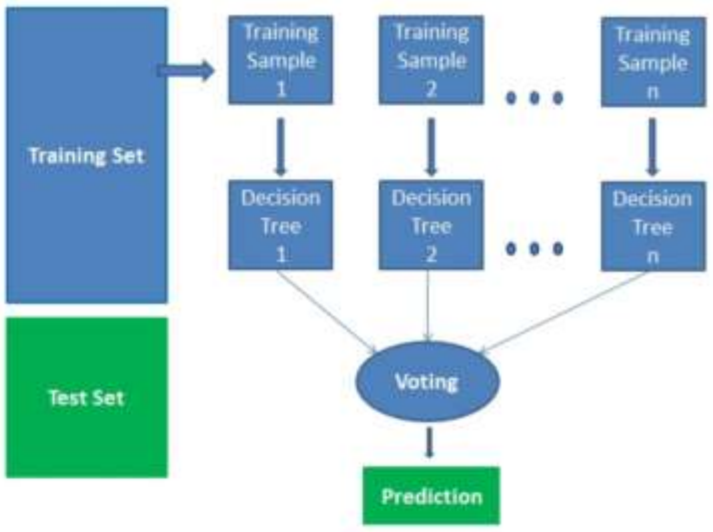


Fig 5. Random Forest Algorithm.

Random forests offer an excellent feature selection indicator. It is designed to calculate the importance of each feature by using a mean decrease in impurity (MDI) or Gini importance. It is a factor of the change of fitting of a model with a decrease in the accuracy when a variable is being dropped. The importance of the variable becomes substantial with a higher decrease in accuracy.

As it involves a good number of decision trees which participates in the process, the random forest is known to be as robust and highly accurate. It is truly beneficial when it comes to eliminate or minimize overfitting. The reason why it doesn't suffer the problem is that it cancels out the biases by taking the average of all the predictions.

Random forest is slow in generating decision trees as it considers all the possibilities recursively. Whenever it makes a prediction, for the same given input, trees follow the same process, and then voting is taken into consideration. It makes the whole process time-consuming. In a nutshell, these are one of the algorithms which are quick to train on, though on the other side, it is quite slow but when predictions are taken into consideration, to create projections once trained. The model turns out to slower in predicting the outcome when it is being traded with more accuracy. There is a tradeoff to consider when planning to opt in or opt out of this algorithm. The random forest algorithm is fast enough in most real-world applications, but there can be scenarios where other approaches might be preferred because run-time performance is essential.

To make the model faster or to enhance the rate of the prediction of the designed model, the hyperparameters in the random forest described below:

1. **Increasing the Predictive Power**

Trees get generated amidst considering the mean of the predictions which is termed as "n_estimators." The range of values used in this research started with the value of 10 and went till 100. The observation was that performance was enhanced with a steady increase in the number of trees thus leading to a stable outcome of predictions, trading computation as a factor.

"max_features" is another parameter that was considered. It is the highest quantity of the features that can be taken into consideration when the node is being split using the algorithm. This parameter was used in the implementation, but there was a significant change in the results considering the additional field to the computation.

When an internal node gets split, a number of leaves are required which is considered as "min_sample_leaf." This parameter was used to set a base condition for the model.

2. Increasing the Models Speed

In order to decide on the number of computing processors which can be required and allowed to use in the computing environment, the parameter “n_jobs” can be useful. Initially, the value of “1” was used because it uses only one processor. Later, the value of “-1” was used because the system can harness the full power of all the processors. It is no more limited to just one processor and can use all if available.

The attribute "random state" makes the output of the model similar. It will generate the same outcome provided a certain parameter with the same value. It has the same training data and same hyperparameters. The state was assigned the value of “0” to produce consistent results because it doesn’t generate random values every time and the value itself doesn’t matter.

Another hyperparameter is “oob_score” is suited to be more of a cross-validation technique. By the usage of this attribute, only two - thirds of the raw input data is used when it comes to training and evaluating the model. The advantage that it doesn’t use additional computation.

GridSearchCV (Keras library) was utilized to enhance the attributes and was useful in model validation. During the experimentation, it was concluded that the accuracy got stable after a defined number of trees. The “n_estimators” parameter was assigned a value of ‘100’ in the final model, returning an accuracy of 0.83.

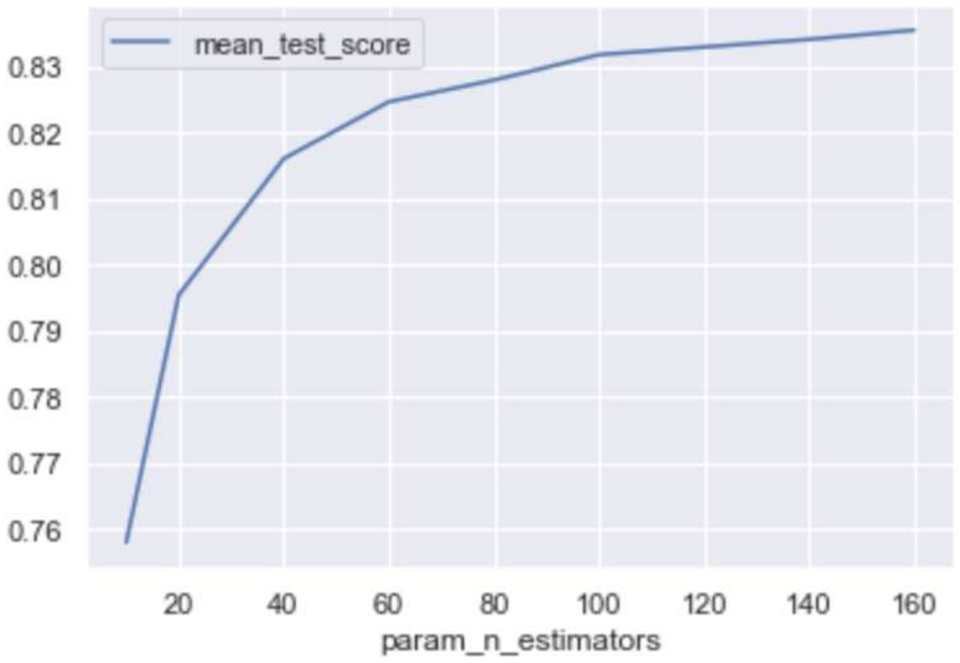


Fig 6. Random Forest Result graph

VI. KNN

In pattern recognition, the most popular classifier for predicting based on neighborhood is k-nearest neighbors (KNN) classifier. It is highly efficient in the areas of machine learning, pattern recognition, text categorization, data mining, object recognition, etc. Moreover, the technique is straightforward to implement. Both the classification and regression functionality can be effectively solved using this supervised algorithm. It assumes that same, or similar attributes/things exist nearby. It shall be closer to each other. A supervised machine learning algorithm when given a new unlabeled data produces an appropriate output by learning from a function which relies on labeled input data.

To predict a category of classification from a range of classes, these classes are formed by the data which gets separated by this algorithm, being non-parametric in nature. No assumptions of how the data gets distributed are considered. This research would be using KNN for the classification, where the output is a class membership. A majority vote of its neighbors classifies an object or sketch. The factor which defines how the features which are out of sample plays a role in categorizing the data point is based on the similarity of the features.

In any case, it accompanies a few restrictions, for example, time multifaceted nature, memory management, and prerequisite, since it is totally reliant on each model in the preparation set.

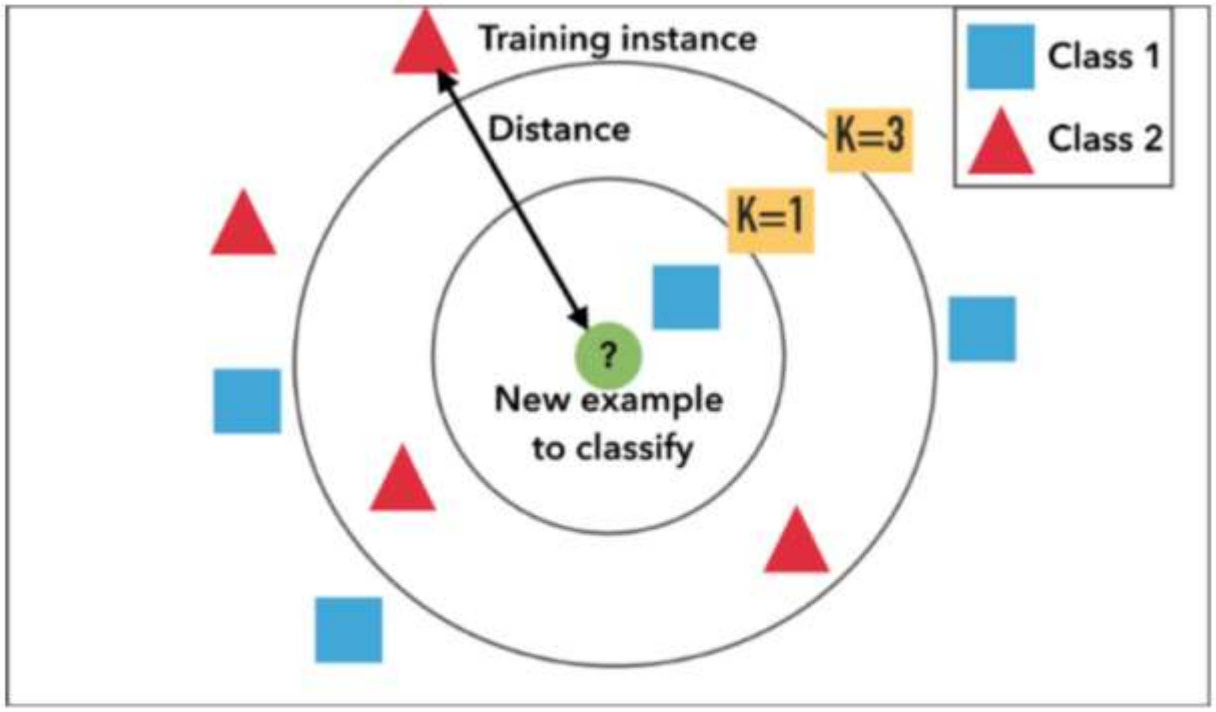


Fig 7. Example of k-NN classification [21]

The KNN Algorithm

1. Loading the dataset/subset.

2. The numbers of the neighbors are assigned to K.

3. Computing for each data point:

3.1 Calculating the length between the unknown and the current instance from the point.

3.2 Adding the length and the index to a defined group.

4. Sorting the index by the measures and ordered a collection of ranges in increasing order.
5. Picking the first K sets from the above-computed array.
6. Getting the tags of the enclosed K points.
7. Returning the mode of the K labels for the classification.

Euclidean Distance

It is the distance which gets computed between two points (new sample and all the data we have in our QuickDraw dataset).

The Euclidean distance's formula is like the image below:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Fig 8. Euclidean Distance formula

Choosing the right value for K

The KNN algorithm was run several times with different amounts of K starting from '1' up to '15,' and the value of K was chosen which minimize the error rate. It is done to let the algorithm shoot for the highest accuracy as possible for the data it has never seen before.

The stability of the prediction was decreased when the value of K was decremented to 1. When the value of K was '1', the sample was incorrectly depicted with a wrong label as the nearest single data belonged to a different class. The value of K was needed to be an odd digit to eliminate the even number condition as the majority vote was considered among labels.

The predictions become more stable as the value of K was increased, due to majority voting, and accuracy was increasing to a certain value. At some point, an increased number of errors were observed. It indicated that the value of K was increased which crossed the threshold.

The cross-validated of the “n_neighbors” was carried out and observed that the best model given is k = 5, which returned with an accuracy of 0.8752.

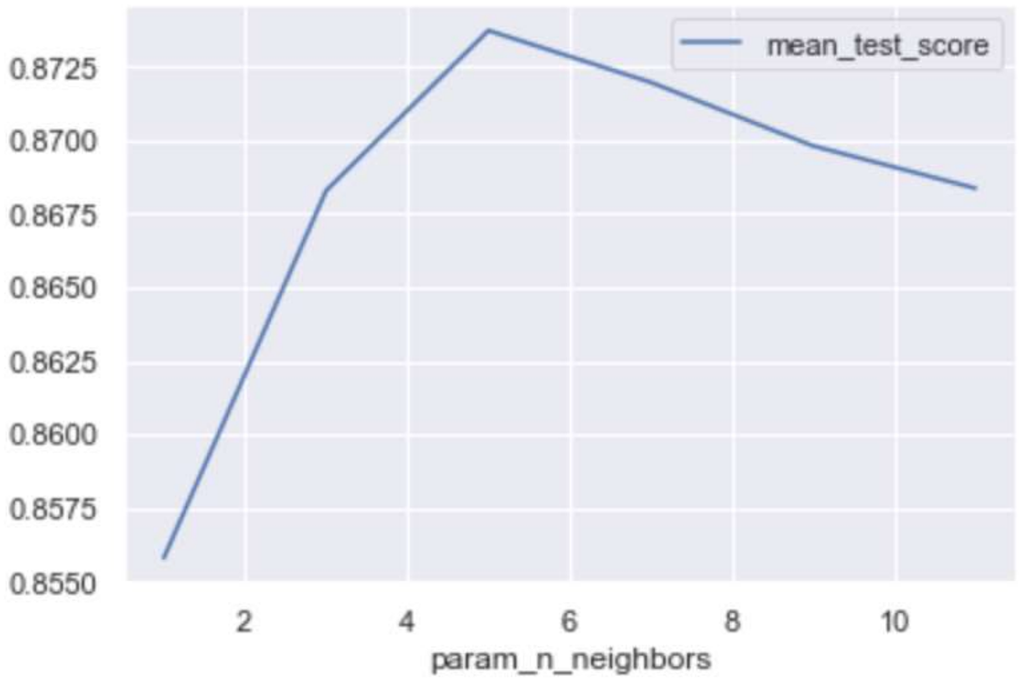


Fig 9. KNN Result Graph

VII. NEURAL NETWORKS

A neural network is a composition of several linear models that connect with activation functions. The input information to one layer of the neural networks is the output information from the previous layer. This way the data is processed through the layers and finally gives a result. Neural networks can be used for both regressions as well as classification. Neural networks are composed of neurons that are connected with different weights which scale the input. The neurons are collected in layers and form the neural network which consists of several layers, some of which are hidden; so-called hidden layers. A bias term, a constant, is added to shift the input to the activation function along the x-axis. For a neural network, the weights and biases are the parameters which are optimized.

A neural network with two layers, the output from the previous layer is given by

$$H = \sigma\left(W^{(1)T}X + b^{(1)T}\right)$$

Where H is the output, the activation function, W(1) the weights, X the input and b(1) the bias, in matrix notation, these variables can be expressed as stated below.

$$b^{(1)} = \begin{bmatrix} \beta_{01}^{(1)} & \dots & \beta_{0M}^{(1)} \end{bmatrix}, \quad W^{(1)} = \begin{bmatrix} \beta_{11}^{(1)} & \dots & \beta_{1M}^{(1)} \\ \vdots & \dots & \vdots \\ \beta_{p1}^{(1)} & \dots & \beta_{pM}^{(1)} \end{bmatrix}, \quad X = \begin{bmatrix} X_1 \\ \vdots \\ X_p \end{bmatrix}$$

The output from the second layer is given by

$$Z = W^{(2)T} H + b^{(2)T}$$

Where Z is the output from the second layer, H the output from the previous layer and b(2) the bias. In matrix notation these variables can be expressed as follows:

$$b^{(2)} = \begin{bmatrix} \beta_0^{(2)} \end{bmatrix}, \quad W^{(2)} = \begin{bmatrix} \beta_1^{(2)} \\ \vdots \\ \beta_M^{(2)} \end{bmatrix}, \quad H = \begin{bmatrix} H_1 \\ \vdots \\ H_M \end{bmatrix}$$

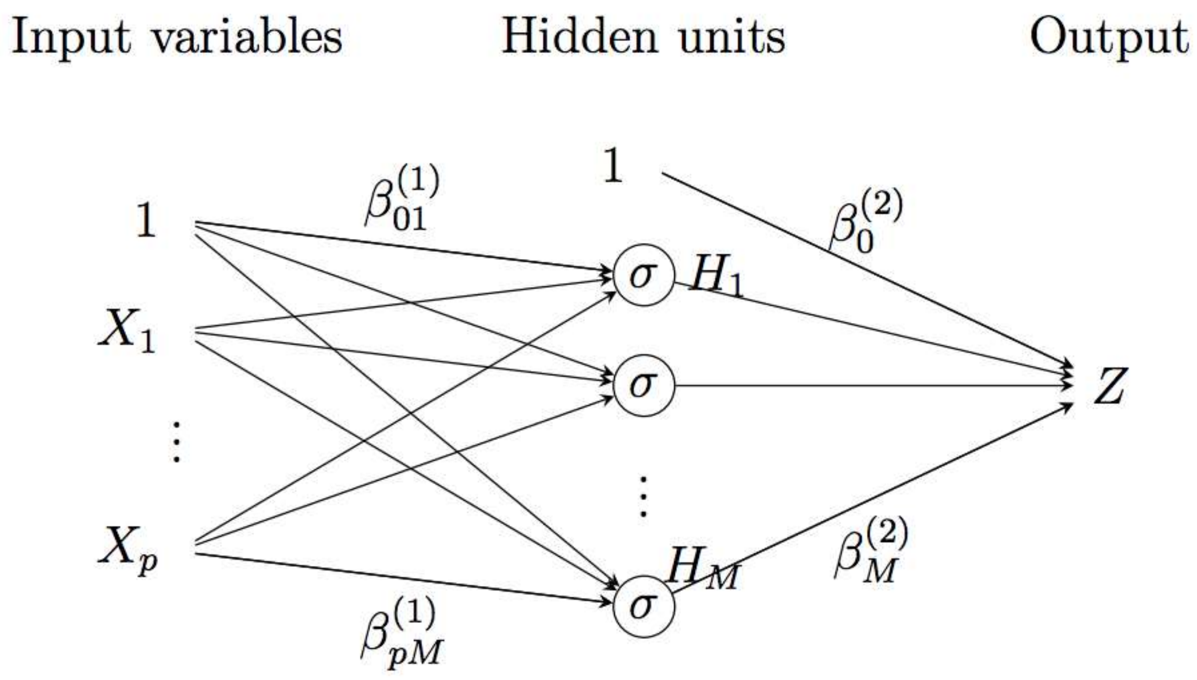


Fig 10. A neural network with two layers [22]

3.5.1 Activation Functions

An activation function scales the input and thereby decides when a neuron is activated, that is passing information to the next layer. The rectified linear unit (ReLU) is an activation function which has become increasingly popular due to its simplicity and high performance. The function is zero for all negative inputs and equal to the input if it is positive, as expressed below:

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

where (x) is the activation function dependent on the input. A graphical representation of this function as seen in figure 11.

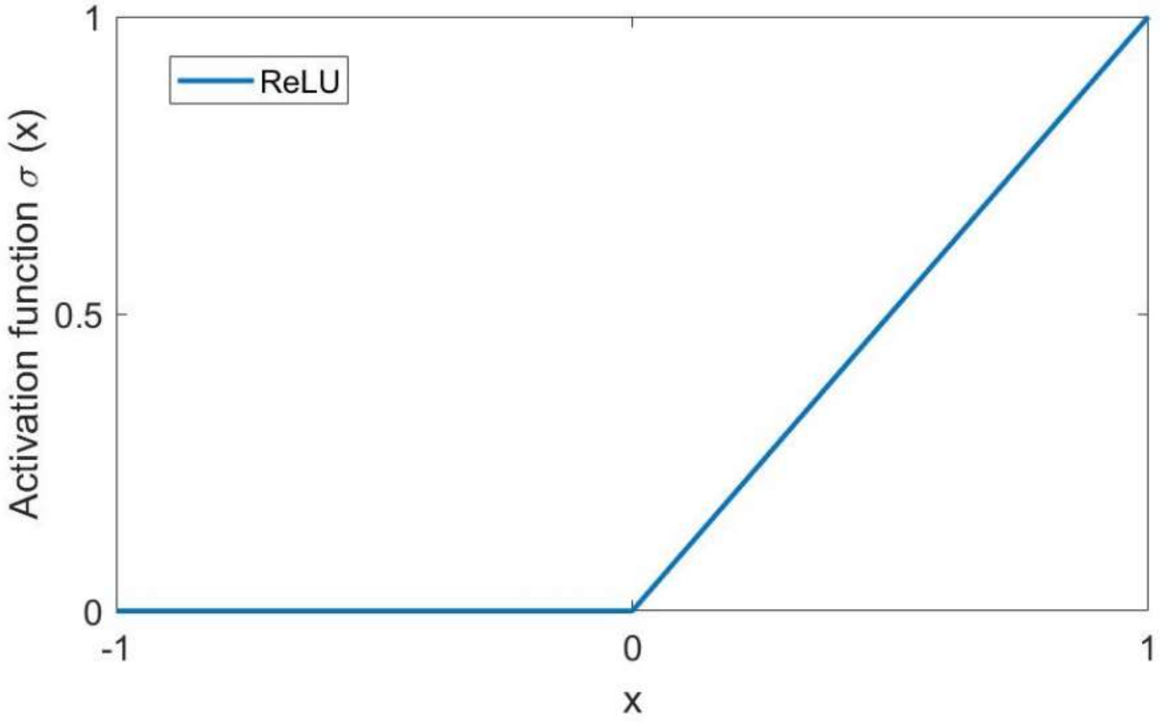


Fig 11. Activation function ReLU as a function of x.

Weighted inputs are summarized and passed through an activation function, sometimes referred to as the transfer function. An activation function is a basic mapping of the summed weighted input to the output of the neuron. It is an activation function because it regulates the threshold at which the neuron is activated and the output signal strength.

For example, if the summed input is above a threshold, 0.5, a step activation function is used which the computed value by the neuron would be close to 1, or else, it would be 0.

Activation functions that are traditionally non-linear are used. It allows the network to combine the inputs in more complex ways. It gives a progressively useful ability in the capacities they can display. Non-linear functions like the logistic, also known as the sigmoid capacity were utilized that yield an incentive somewhere in the range of 0, and 1 with S-molded dissemination and the hyperbolic digression work likewise that yields a similar circulation over the range - 1 to +1.

When working with a classification problem, the result should be a class probability to achieve a qualitative output. In a neural network, this can be done by using softmax as the activation function in the last layer of the network. The Z in the Softmax (Z) function is a vector containing the input to the function and range of [1, K] where K is the number of classes. Softmax gives an output between 0 and one which can be interpreted as a class probability.

$$\mathrm{softmax}(Z) = \frac{e^{Z_k}}{\sum_{l=1}^{K} e^{Z_l}}$$

3.5.2 Fully Connected Layers and Dropout

All connections to the output units in a densely connected layer, described by unique parameters contributing to a large number of parameters that requires computation. Generally, these connected neural layers get placed at the tail of a neural network. Adding so-called dropout to a layer removes a fraction of the neurons according to a specified percentage. Dropout prevents the model from overfitting because it highly adapts to the training data, by making the result more random. One sign of overfitting is a massive difference between the training and validation loss, where the training loss is significantly smaller.

After perhaps the most helpful kind of neural system, the field of artificial neural networks is often referred to only as neural networks or multi-layer perceptrons. A perceptron is a solitary neuron display that was a forerunner to bigger neural systems. It is a field that explores how straightforward neural models can be used to understand complex computational assignments such as the prescient display errors found in AI. The objective isn't to make practical models of the mind, yet instead to create hearty calculations and information structures that one can use to display serious issues.

Neural networks are learning to map in this sense. They can learn mathematically any mapping function and have proven to be a universal approximation algorithm.

Neural networks' predictive capacity comes from the networks' hierarchical or multi-layered structure. The data structure can select and combine features at different scales or resolutions into higher - order elements, for example, from lines to collections of lines to shapes.

VIII. MULTI-LAYER PERCEPTRON

The typical type of neural network is multilayer perceptrons or short MLPs. They consist of one or more neuron layers. Data is provided to the input layer. One or more hidden layers can provide abstraction levels, and the output layer, also known as the visible layer, is predicted. Most of the details about the building blocks of this neural network were shared in the previous section.

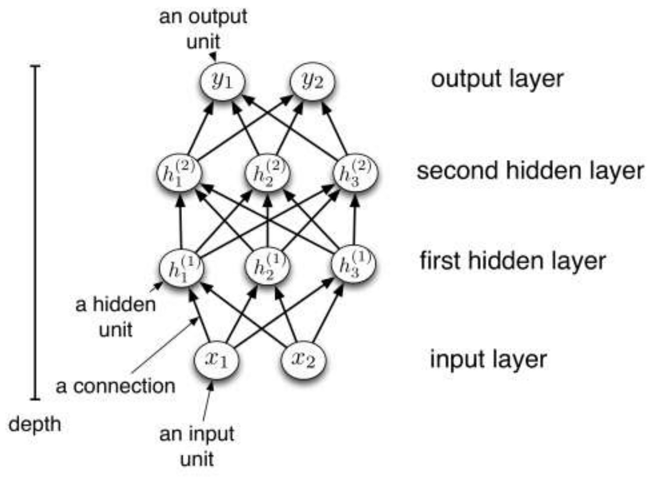


Fig 12. Simplest kind of feed-forward network [23]

The units are arranged into a set of layers. Each layer contains a certain number of identical units. The network is fully connected when every unit is connected to its subsequent layers. The input is the first layer, and its units take the values of the input features. The last is the output layer, and it has one unit for each value the network outputs. Hidden layers lie in between these input and output layers, as one cannot know ahead of time what these units should be computed, and it gets discovered during the learning phase.

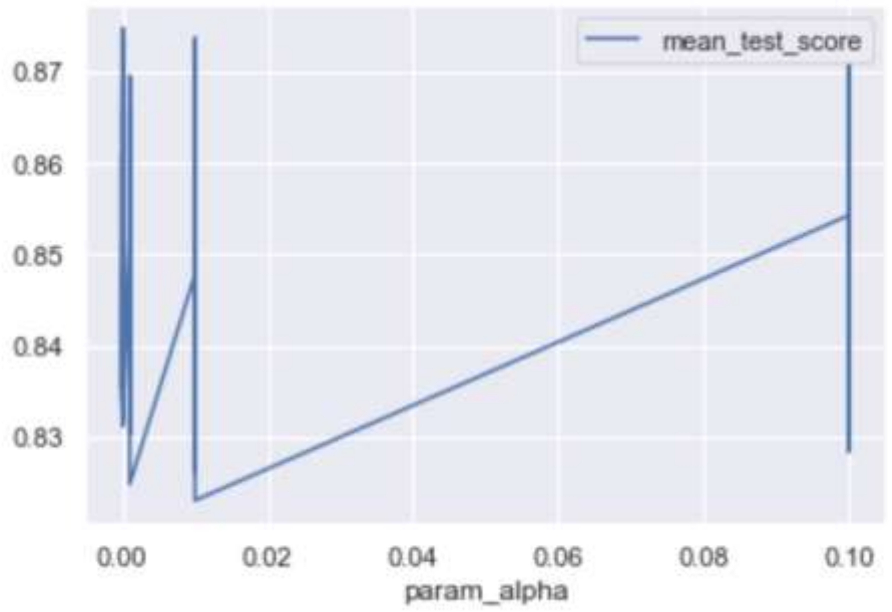


Fig 13. MLP Result Graph.

The Multi-Layer Perceptron (MLP) from sci-kit-learn was used. After the cross-validation which was conducted over many different hidden layer sizes and their learning rates. It came out with a 0.865 accuracy with alpha = 0.001 learning rate and consisting of 648 hidden layers.

IX. CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) is a robust and effective technique in the fields of the neural network. It was very well suited to digit recognition in the space of object detection as it reserves the dimensional representation and structure of the underlying object. It has become widely popular as it helps in solving complex computer vision challenges like in the case of natural language processing. CNNs expect and preserve the pixel-spatial relationship by using small squares of input data to learn internal representations of features.

The item in the image can be dynamically transformed and translated by use of feature learning which happens in the whole image across all the pixels, and the network is still able to detect this process.

CNNs are a type of neural network typically used for data with a grid-like structure such as 2D-images. The structure of a CNN consists of several layers such as convolutional layers, pooling layers, and dense layers. These layers are used for processing the input. As suggested by the name, convolutional neural networks are connected to the mathematical operation convolution. For a 2D-image the convolution operator is given by

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} \beta_{ij} \cdot I_{x+i-1,y+j-1}$$

where I is the image and K a set of parameters (ij), known as a kernel, that has height h and width w. The kernel is stepped with a specified size (stride) and applied to all pixels of the image, as illustrated in figure 14.

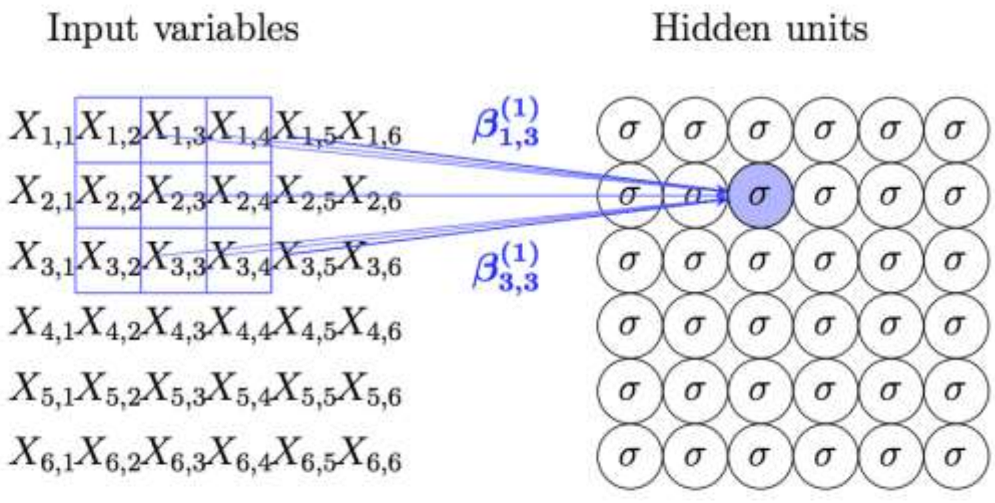


Fig 14. A convolutional layer with a kernel with parameter's size (3, 3) [24]

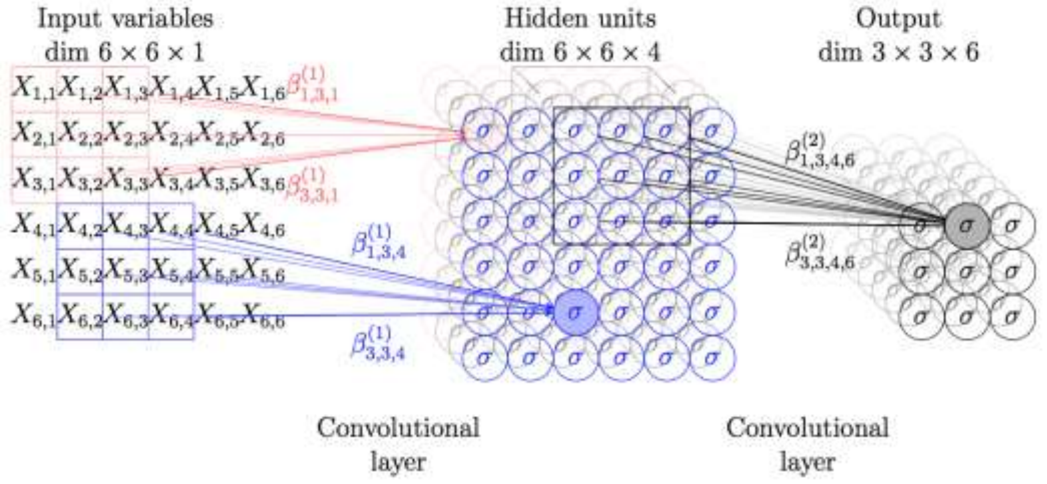


Fig 15. Excerpt of a CNN architecture consisting of two convolutional layers [24]

Below are three kinds of convolutional neural layers:

1. Convolutional Layers

Filters and feature maps consist of conventional layers. The filters are the layer's "neurons." They have weights of input. The size of the incoming data is a defined set of reception. The pixel values will be the input patch considering the incoming layer is the convolution layer. The input from the previous layer's feature map will be taken from the layer of convolution.

Each pixel is processed and taken one at a time, and filter is applied to the preceding layer and each index location outcomes the activation of the neuron, and it gets collected which occurs in the feature map. If the receptive field moves from one activation to another, one can see that the input values (field width–1) will overlap the area with the previous activation. The step is defined by each occurrence of the activation, and the length of the removal of the filter from the preceding layer is termed as zero paddings.

The reception tries to access and read off the circumvent of the incoming feature map when the size of the step and the filters of the receptive field cannot be mathematically divided by the capacity of the preceding layer. Methods such as zero paddings defined above come in the picture which can help to generate mock data input to read off these receptions.

2. Pooling Layers

These layers help in downgrading the layers before it. These pooling layers help the feature map that was defined earlier by merging the similar features and following a sequence of one or more convolutional layers.

It generally reduces the model's overfitting of training data as pooling can be a technique for compressing or generalizing representations of features.

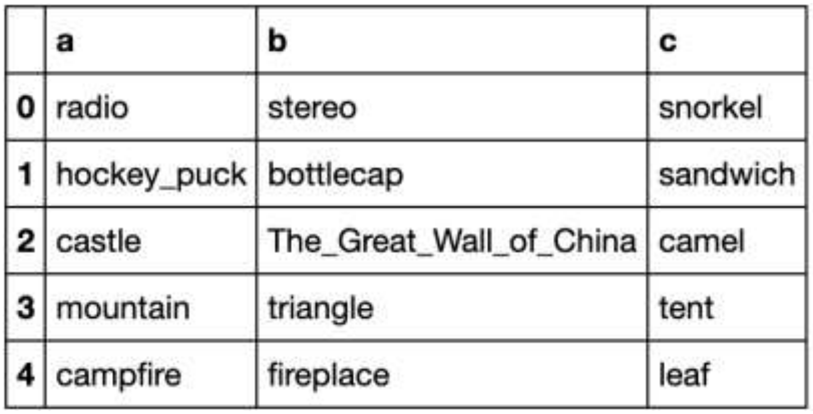
The layer of convolution is significantly small with the field of reception it has. To avoid the condition of overlapping, the magnitude of the reception is generally equal to the number of inputs of each receptive activation. These layers are usually straightforward to create their own feature map, taking the average or maximum input value.

3. Fully Connected Layers

This is one of the typical flat feed-forward layers of the neural network. These layers use a softmax activation or non-linear activation function to give output probabilities of class predictions. The convolutional and pooling layers perform extraction of features and consolidation, and at the end of the network, these fully connected layers come into the picture. They are used to create non-linear endings.

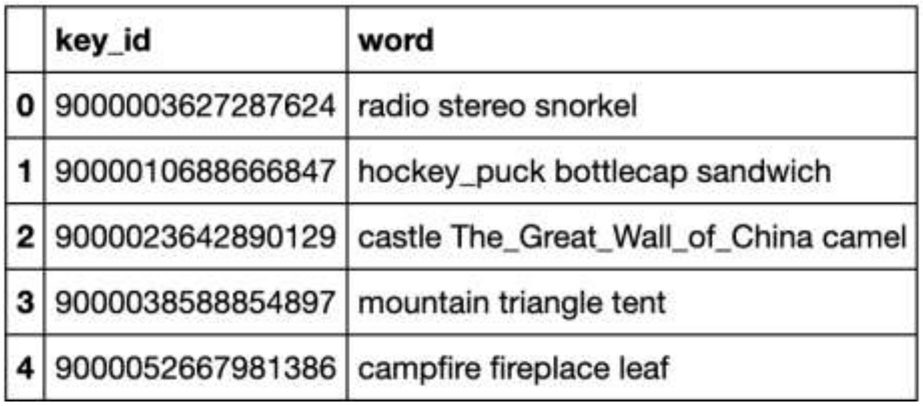
Selecting an Optimizer

A few optimizers were tried to see if any improvements in accuracy could be noticed. The Adaptive Moment Estimation (Adam) optimizer and the Stochastic Gradient Descent (SGD) optimizer were performed on the data and comparison was made. It was wise to choose Adam optimizer after multiple iterations because it yielded slightly better results and converged faster than SGD.



	a	b	c
0	radio	stereo	snorkel
1	hockey_puck	bottlecap	sandwich
2	castle	The_Great_Wall_of_China	camel
3	mountain	triangle	tent
4	campfire	fireplace	leaf

Table 3. Top 3 categories for Prediction ensemble in CNN



	key_id	word
0	9000003627287624	radio stereo snorkel
1	9000010688666847	hockey_puck bottlecap sandwich
2	9000023642890129	castle The_Great_Wall_of_China camel
3	9000038588854897	mountain triangle tent
4	9000052667981386	campfire fireplace leaf

Table 4. Table for id and output result mapping in ResNet

X. ResNet

This section would consist of both the SE-ResNet-34, SE-ResNet-50 architectures. When the previous model's depth was increased, the first noticeable variations were a gradient of degradation and vanishing. In other words, for deeper models, the variations were worse than for simpler ones. A Residual Network is an architecture of a neural network that uses deep residual learning in the simplest way possible to solve the problem of gradient vanishing and degradation.

During back propagation, in the stage, the gradient gets clear via f(x) when the signal is tracked in reverse. Here f(x) could be batch normalization, matrix multiplication, convolution, etc. It could cause issues as it involves non-linearities.

A shortcut is applied where it allows the gradient to pass backward directly. It is denoted by "+ x" at the end is the shortcut. The gradient could make it the bottom of the layer crossing over the middle layers without being diminished when these layers are stacked.

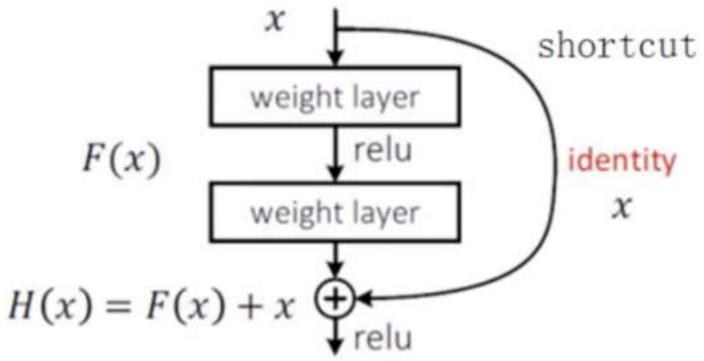


Fig 16. Backpropagation in ResNet

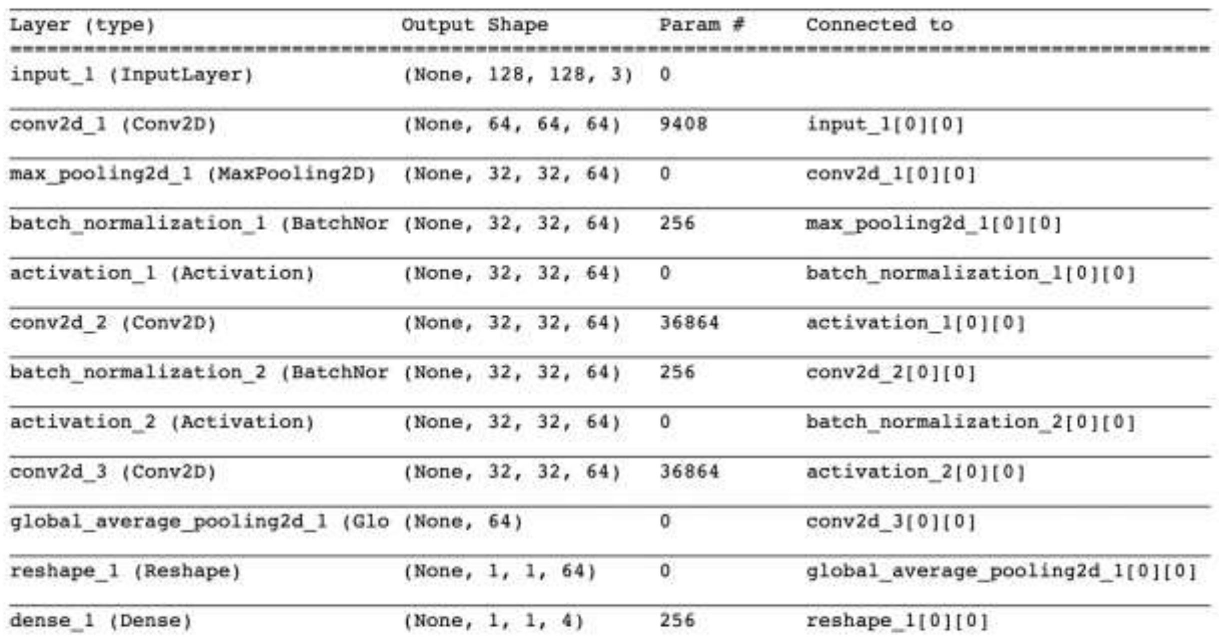
Squeeze and Excitation Net (SE) -ResNet-34 and 50 were trained further from using simple CNN. An additional block gives various channel of weights. It has been proven that the SE blocks provide additional accuracy by providing the weights but only by increasing less than 10 percent of the total parameters.

During SE - ResNet-50 training, various parameters were tried for 50 to 60 epochs. Finally, the batch size of 512 and the image size of 128x128 gave the score the best improvement, boosting it to 0.91 out of all the combinations.

STEPS per Epochs = 500, 800 or 1000

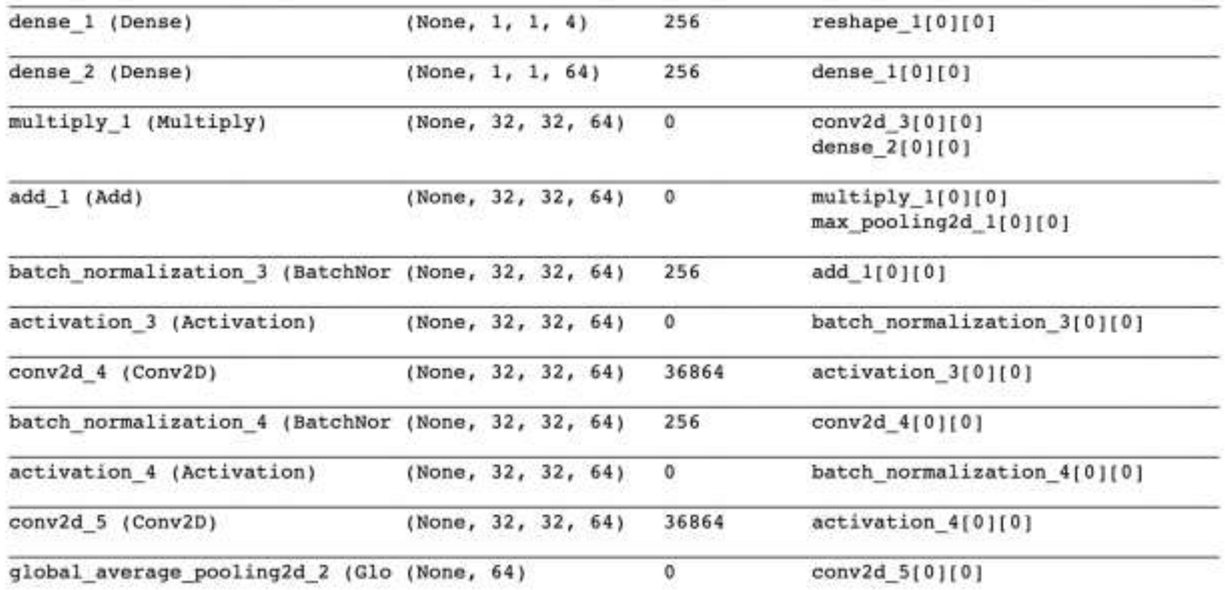
Size = 82*82, 96*96, or 128*128

Batchsize = 256 o 512



```
Layer (type)                     Output Shape          Param #     Connected to
==================================================================================================
input_1 (InputLayer)             (None, 128, 128, 3)   0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)                (None, 64, 64, 64)    9408        input_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)   (None, 32, 32, 64)    0           conv2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor  (None, 32, 32, 64)    256         max_pooling2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)        (None, 32, 32, 64)    0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)                (None, 32, 32, 64)    36864       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor  (None, 32, 32, 64)    256         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)        (None, 32, 32, 64)    0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)                (None, 32, 32, 64)    36864       activation_2[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo  (None, 64)            0           conv2d_3[0][0]
__________________________________________________________________________________________________
reshape_1 (Reshape)              (None, 1, 1, 64)      0           global_average_pooling2d_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                  (None, 1, 1, 4)       256         reshape_1[0][0]
```

Table 5. A small subset of the layers in ResNet 34



```
__________________________________________________________________________________________________
dense_1 (Dense)                  (None, 1, 1, 4)       256         reshape_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                  (None, 1, 1, 64)      256         dense_1[0][0]
__________________________________________________________________________________________________
multiply_1 (Multiply)            (None, 32, 32, 64)    0           conv2d_3[0][0]
                                                                   dense_2[0][0]
__________________________________________________________________________________________________
add_1 (Add)                      (None, 32, 32, 64)    0           multiply_1[0][0]
                                                                   max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor  (None, 32, 32, 64)    256         add_1[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)        (None, 32, 32, 64)    0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)                (None, 32, 32, 64)    36864       activation_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor  (None, 32, 32, 64)    256         conv2d_4[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)        (None, 32, 32, 64)    0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)                (None, 32, 32, 64)    36864       activation_4[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_2 (Glo  (None, 64)            0           conv2d_5[0][0]
```

Table 6. A small subset of the layers in ResNet 50

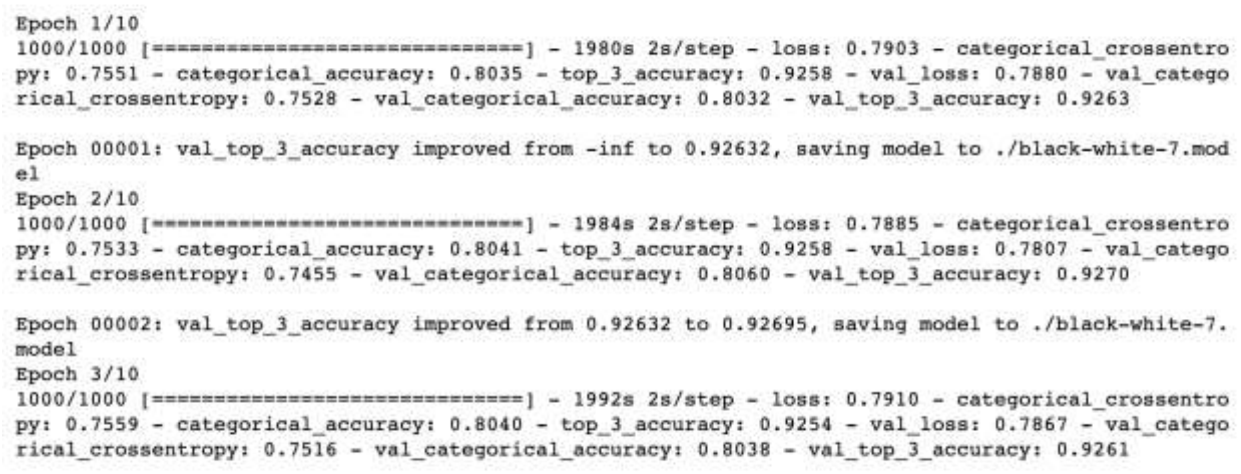
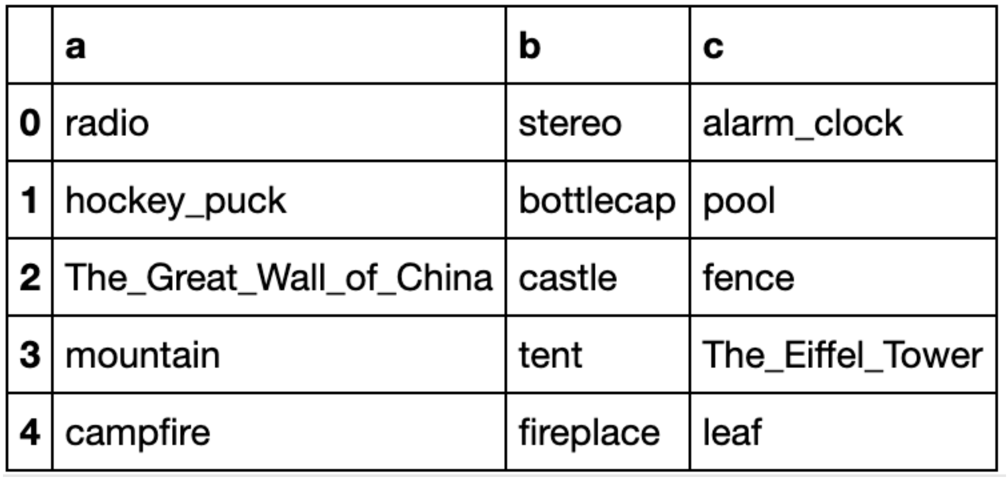
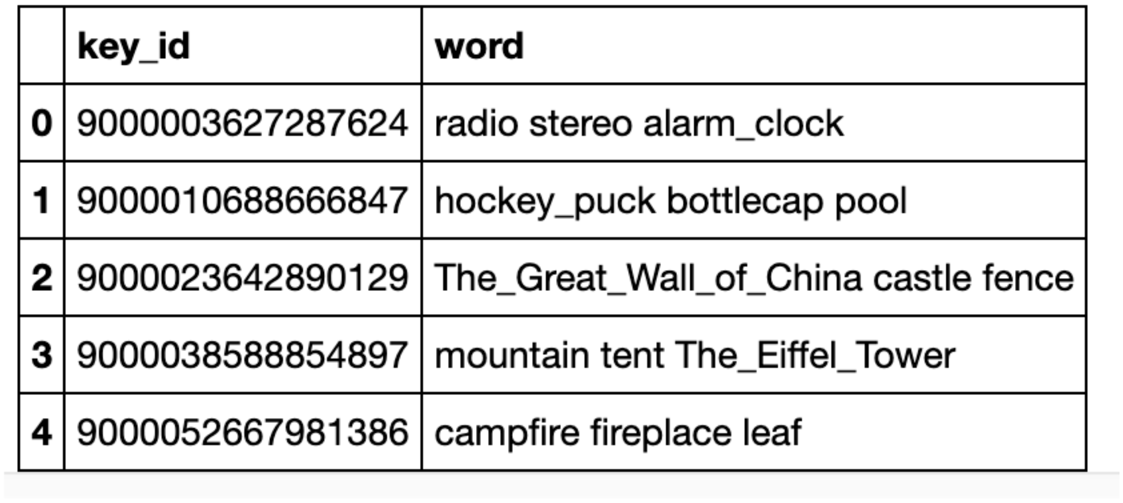

```
Epoch 1/10
1000/1000 [==============================] - 1980s 2s/step - loss: 0.7903 - categorical_crossentro
py: 0.7551 - categorical_accuracy: 0.8035 - top_3_accuracy: 0.9258 - val_loss: 0.7880 - val_catego
rical_crossentropy: 0.7528 - val_categorical_accuracy: 0.8032 - val_top_3_accuracy: 0.9263

Epoch 00001: val_top_3_accuracy improved from -inf to 0.92632, saving model to ./black-white-7.mod
el
Epoch 2/10
1000/1000 [==============================] - 1984s 2s/step - loss: 0.7885 - categorical_crossentro
py: 0.7533 - categorical_accuracy: 0.8041 - top_3_accuracy: 0.9258 - val_loss: 0.7807 - val_catego
rical_crossentropy: 0.7455 - val_categorical_accuracy: 0.8060 - val_top_3_accuracy: 0.9270

Epoch 00002: val_top_3_accuracy improved from 0.92632 to 0.92695, saving model to ./black-white-7.
model
Epoch 3/10
1000/1000 [==============================] - 1992s 2s/step - loss: 0.7910 - categorical_crossentro
py: 0.7559 - categorical_accuracy: 0.8040 - top_3_accuracy: 0.9254 - val_loss: 0.7867 - val_catego
rical_crossentropy: 0.7516 - val_categorical_accuracy: 0.8038 - val_top_3_accuracy: 0.9261
```

Table 7. A subset of values of epochs in ResNet



	a	b	c
0	radio	stereo	alarm_clock
1	hockey_puck	bottlecap	pool
2	The_Great_Wall_of_China	castle	fence
3	mountain	tent	The_Eiffel_Tower
4	campfire	fireplace	leaf

Table 8. Top 3 categories for Prediction ensemble in ResNet



	key_id	word
0	9000003627287624	radio stereo alarm_clock
1	9000010688666847	hockey_puck bottlecap pool
2	9000023642890129	The_Great_Wall_of_China castle fence
3	9000038588854897	mountain tent The_Eiffel_Tower
4	9000052667981386	campfire fireplace leaf

Table 9. Table for id and output result mapping in ResNet

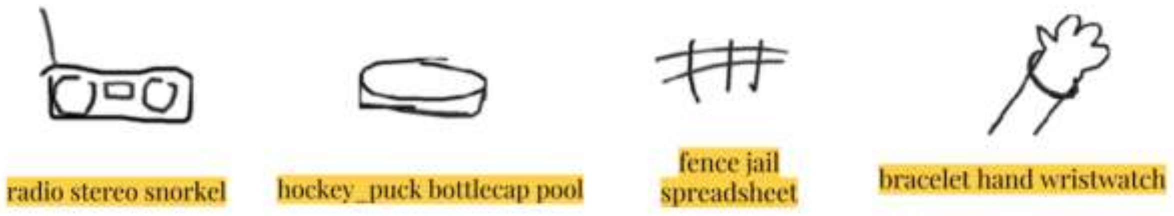


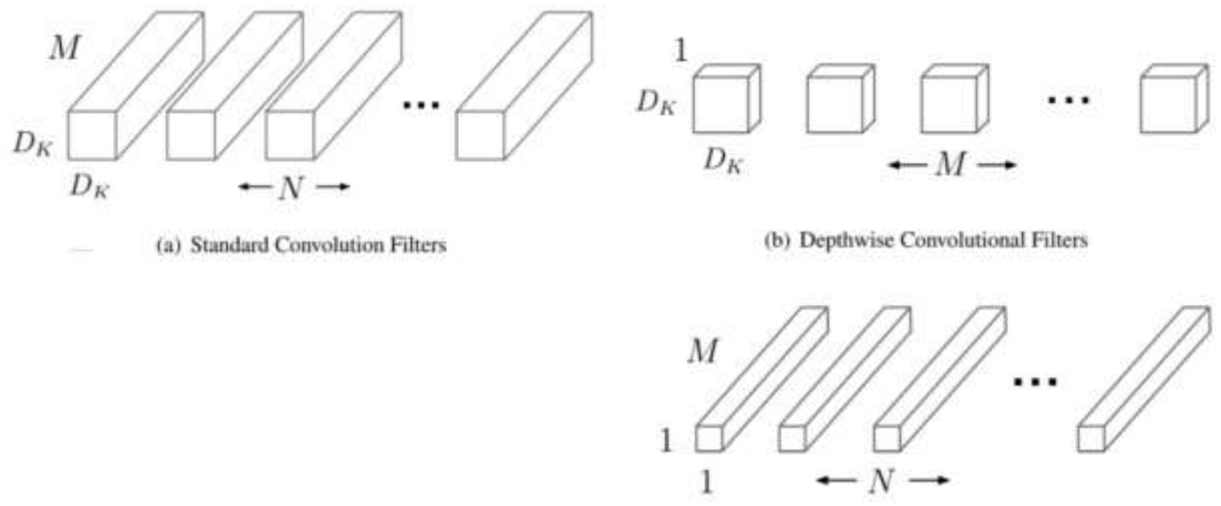
Fig 17. Various indexed results for objects in ResNet

XI. MOBILENET

Google introduced MobileNet uses depth-wise separable convolutions which are designed based on a streamlined architecture which helps in building lightweight deep neural networks. In a single step, a standard convolution applies filters across all input channels and combines these values. On the other hand, a depth-wise separable convolution performs two different stages:

1. A single filter to each input channel is applied in depth-wise convolution.

2. Later, pointwise convolution (1×1 convolution), is used to create a linear combination of the output of the depth-wise layer.



M
D_K
D_K
N
(a) Standard Convolution Filters
1
D_K
D_K
M
(b) Depthwise Convolutional Filters
M
1
1
N

Fig 18. Different Convolutional Filters [25]

MobileNet also provides two parameters which have a significant reduction in the number of computations. The width multiplier thins the number of channels. It produces alpha * N instead of N channels. This multiplier can be used to deal with a trade-off between the desired performance and latency. Another is the multiplier of resolution. It scales the image's input size from 128 to 224. The MobileNet was trained on 224x224 images as it uses a global average pooling instead of a flatten. It depends only on the number of channels and not on the feature maps spatial dimension with a global scope of poling at the tail of the network model.

Choosing the right MobileNet model to fit is crucial. The memory and disk size of the model is proportional to the number of attributes. The model scales latency and power usage with the number of Multiple - Accumulates (MACs) measuring the amount of fused multiplication and addition operations. This factorization weights the combination of the capacity of the kernel and the outcome channels and helps in disengaging the connection as it reduces computation and model size.

The original stats from the MobileNet paper [19] showed a reduction in computation costs by at least nine times. The standard module of MobileNet in Keras was used in this implementation, and it achieved an astonishing 92.1% accuracy.

XII. Results

The deep neural network architecture used by the MobileNet and its aesthetics of a dropout rate of 10% has been seen as the best model in terms of performance and tradeoff in the conducted experiments and in comparison, to different algorithms. It came up to 92% in accuracy. This algorithm works better than the simple convolutional neural networks proposed in [7] which utilized activation functions and ADAM optimizer. However, state-of-the-art ResNet models achieve better accuracy than CNNs reaching 90% in accuracy with validation and with its convolutional Resnet 34 and ResNet 50 architecture [26]. On top of that, non-neural network-based models like Random Forest were able to reach 83% cross-validation accuracy using decision trees trained on traditional image features [20]. Some of the incorrect classification observed was for the 'angel' object/class in the animal migration category where it was classified incorrect for around twice the factor in the sample dataset. The sketches vary depending on the artistic representation, but more intensely for some type of objects than another highly common set of objects like the sun, where the idea is simple of sketching an of a circle with radiating lines in a clockwise direction.

Different machine learning models serve different purposes, so a straight-forward comparison would be inherently biased. Random Forests are excellent classifiers for handling binary classification tasks along with SVMs and Gradient Boosting. On the other hand, issues in the fields of vision and speech can be modeled much better-using networks like those found in deep learning frameworks.

Model	Main parameters	Accuracy
Random Forest	100 trees Max depth = 8	0.83
MLP	Hidden Layer (,784) Alpha = 0.001	0.865
KNN	N_Neighbors = 5	0.875

Simple CNN	NCSVS = 100
	NCATS = 340
		0.821

Greyscale CNN	Factor=0.5
	Min_delta=0.005
		0.87

ResNet 50	Classes = 1000
	Bottleneck = True
		0.90

ResNet 34	Classes = 1000
	Bottleneck = False
		0.909

MobileNet	Classes = NCATS
	Alpha = 1.0
		0.921

Table 10. Comparisons of results of different Machine learning algorithms

XIII. CONCLUSION

The work analyzed the prediction accuracy of eight different machine learning models on the most recent and popular dataset. The study focused on five classes of the dataset. The primary purpose was to find out the accuracy of the different models on the same dataset and evaluating the consistency of prediction by each of these machine learning models. The work adds to the comprehension of the machine learning applicability in different but related domains to that of the training set. It can further be summed up that neural networks are new and best emerging techniques for making a machine intelligent for solving many real-life object categorization problems.

The sketch recognition and classification are quite challenging and daunting as it is highly dependent on the creative skills and interpretation by humans who have been depicted in [1] [2] [3]. The images used in [2] are larger than 200,200 pixels, and extensive efforts were made to achieve the results. But at this point, there is no need for such a large size (for deep CNN to work on), especially on sketch datasets. With the modification of current state-of-the-art CNN architectures, it is possible to work with smaller sizes and achieve even better results. In [3], the neural network with a deeper model was presented for better accuracy order to eliminate overfitting and take in an average rate of dropout.

The future work should train these CNN's in more restricted categories and vary the diversity of examples, including untargeted sketches, related to the performance of classification. Future research should also seek to implement these findings as a basis and build a hybrid model combining various models and cherry-picking features based on the type of requirements and application. The new models such as ShuffleNet consisting of group convolutions and channel shuffles, EffNet using separable spatial convolutions could be taken into consideration. Recently improved version of MobileNet can be used to the existing research. The goal would be to ensure that the most relevant category gets predicted in the least amount of time without reducing the speed with which the machine learning model gets trained and achieving the desired accuracy of the prediction of the human-made sketches.

REFERENCES

[1] M. Eitz, J. Hays and M. Alexa, "How do humans sketch objects?" ACM Trans on Graphics, vol. 31, no. 4, pp. 1-10, 2012. [Online] Available: https://dl.acm.org/citation.cfm?id=2185540

[2] H. David, and D. Eck. "A neural representation of sketch drawings." arXiv preprintarXiv:1704.03477, 2017. [Online] Available: https://dl.acm.org/citation.cfm?id=2185540

[3] L. Wayne and E. Tran, "Free-hand Sketch Recognition Classification," CS 231N Project Report, 2017.

[4] R. Schneider and T. Tuytelaars, "Sketch classification and classification-driven analysis using Fisher vectors," ACM Trans on Graph., vol. 33, no. 6, pp. 1-9, 2014.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[6] Q. Yu, Y. Yang, F. Liu, Y. Song, T. Xiang, and T. Hospedales, "Sketch-a-Net: A Deep Neural Network that Beats Humans," Int. J. of Comput. Vision, vol. 122, no. 3, pp. 411-425, 2016.

[7] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse, "Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression," Comput. & Graph., vol. 71, pp. 77-87, 2018.

[8] P. Ballester and R.M. de Araújo, “On the Performance of GoogLeNet and AlexNet Applied to Sketches.”, In AAAI, pp. 1124-1128, 2016.

[9] K. T. Yesilbek, C. Sen, S. Cakmak, and T. M. Sezgin. SVM-based sketch recognition: which hyperparameter interval to try? In Proceedings of the workshop on Sketch-Based Interfaces and Modeling, pages 117–121. Eurographics Association, 2015.

[10] MathPad2: A System for the Creation and Exploration of Mathematical Sketches.

[11] LI, Yi, SONG, Yi-Zhe, et GONG, Shaogang. Sketch recognition by ensemble matching of structured features. In: In British Machine Vision Conference (BMVC). 2013.

[12] SCHNEIDER, Rosália G. et TUYTELAARS, Tinne. Sketch Classification and Classification-driven Analysis using Fisher Vectors. ACM Transactions on Graphics (TOG), 2014, vol. 33, no 6, p. 174.

[13] Ebrahimi, Mohammad S., and Hossein A. "Study of residual networks for image recognition." arXiv preprint arXiv:1805.00325, 2018.

[14] Nguyen, Anh, Jason Y., and Jeff C. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 427-436. 2015.

[15] Cireşan, Dan, Meier U., and Schmidhuber J. "Multi-column deep neural networks for image classification." arXiv preprint arXiv:1202.2745, 2012.

[16] Szegedy, Christian, Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., and Rabinovich A. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

[17] Krizhevsky, Alex, Hinton G. "Learning multiple layers of features from tiny images." Vol. 1, no. 4. Technical report, University of Toronto, 2009.

[18] Srivastava, Nitish, Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15, no. 1, 2014.

[19] Howard, Andrew G., Menglong Zhu, Bo C., Dmitry L., Weijun W., Tobias W., Marco Andreetto, and Hartwig A. "Mobile nets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861, 2017.

[20] "Quick, Draw! The Data," Google. [Online]. Available: https://quickdraw.withgoogle.com/data. [Accessed: 22-Sept-2018].

[21] A. Bronshtein and A. Bronshtein, "A Quick Introduction to K-Nearest Neighbors Algorithm," Noteworthy - The Journal Blog, 11-Apr-2017. [Online]. Available: https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7. [Accessed: 02-Dec-2018].

[22] Multilayer Shallow Neural Network Architecture - MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/ multilayer-neural-networkarchitecture.html; [Accessed: 17-Jan-2019].

[23] S. An, "Feedforward Neural Networks," Sungtae's awesome homepage, 08-Oct-2017. [Online]. Available: https://www.cc.gatech.edu/dlhc-fnn/. [Accessed: 27-Jan-2019].

[24] "Deep Learning – Computer Vision and Convolutional Neural Networks," Anh Vo, 02-Feb-2018. [Online]. Available: https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/. [Accessed: 10-Oct-2018].

[25] Z. Li, "Reading Note: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Joshua's Blog - Welcome! [Online]. Available: https://joshua19881228.github.io/2017-07-19-MobileNet/. [Accessed: 23-Mar-2019].

[26] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).