# Milestone
# Drawing: A New Way To Search
## (Computer Vision)

Nguyet Minh Phu
*Department of Computer Science*
*Stanford University*
*minhphu@stanford.edu*

Connie Xiao
*Department of Computer Science*
*Stanford University*
*coxiao@stanford.edu*

Jervis Muindi
*Department of Computer Science*
*Stanford University*
*jmuindi@stanford.edu*

***Abstract*—In this project milestone, we summarize our initial experimental results as well as outline our next steps to classify doodles in an effort to develop a new way of interacting with computers.**

## 1. Motivation

### 1.1. Problem Statement

Using words can be limited when communicating across cultures and literacy levels. Images are a shared medium of communication that can beneficially bridge those divides. Instead of using words, we can draw to communicate what we want to computers. This form of communication, however, requires computers to understand our quick drawings (i.e. doodles). Thus, our project aims to recognize the meaning of hand-drawn doodles. More specifically, we first want to develop a system to recognize labels of hand-drawn images based on Google's QuickDraw dataset.

Our project is within the relatively successful domain of image recognition. However, we want to focus on not just accuracy but also efficiency (i.e model size, training time). Efficiency is critical to allow deployment of the system in real-life applications, yet it has not received sufficient attention in research.

We believe this is possible by taking advantage of two distinct characteristics of our doodle dataset which are:

- Image has only two colors, black and white
- Image consists only of lines

## 2. Methods

The first stage of our project is characterization to see how well these general methods on image classification do on our specific problem of doodle classification. This involves implementing popular methods used for image classification and creating a benchmark to document their performances. We want to analyze performance to understand what works in each method, so we have a basis to which we can compare and come up with improvements applicable to our specific problem of doodle classification.

So far, we have started benchmarking popular methods used for image classification, including Logistic Regression, Support Vector Machines (SVM) with different kernels to see how they perform in the context of classifying doodles.

Our intuition is that while these general methods for image classification can work in the same manner as on doodle classification, some parts of these methods may be extraneous and add unnecessary complexity for classifying doodles. A doodle is like an image with only black or white color options and a focused subject matter. Some feature extractors used for Support Vector Machine to build a histogram of gradient based on color variation. We may not need as many layers in the Convolutional Neural Net (CNN) because our data does not contain as much information. By removing or altering these components of existing methods, we may be able to increase accuracy and speed up training and inference time.

We need to gain a good understanding of how the existing methods work. We focus on the 3 areas when trying out these methods:

- Accuracy: via accuracy
- Runtime: via train time
- Specific cases of classification success and failure: via plotting of confusion matrix and manual examination of classes that are often identified wrongly

## 3. Preliminary Experiments

### 3.1. Setup

The full size of our dataset weighs in over 40GB of data even when compressed. To make training more tractable on modest computing resources, we elected to work with a subset of the data. Specifically, this meant constraining the overall number of examples to look at within a class and the total number of classes. The classes selected were chosen randomly from the overall pool of classes and are fixed throughout our experimentation. The chosen limit for maximum number of examples per class is 70,000 examples. For number of classes, we picked three choices of 3, 10, 50 classes roughly corresponding to small, medium, large data set sizes.

**3.1.1. Computing Hardware.** We decided to use Google Cloud Compute Engine for running our machine learning experiments. Even with the restricted data set size, we had encountered issues of running out of memory. To address that, we settled on an instance with 64GB of RAM, 256GB of disk and 10 Cores of CPU. The software image selected for the VM was the Deep Learning Image[2].

## 3.2. Baseline: Logistic Regression

For baseline, we implemented Logistic Regression using a bitmap of the data. This was implemented using Python SciKit Learn framework. The solver for the logistic regression was 'lbfgs' and the 'multi_class' setting was multinomial as we have several classes that we're trying to classify. To keep training times reasonable on the larger dataset, maximum number of iteration was set to 100. The results for Logistic Regression are discussed below.

**3.2.1. Outcomes.** The accuracy scored reported below are normalized with 1 representing 100% accuracy. The best performance for Logistic Regression came with 3 classes where we achieved 79% accuracy and it took less than a minute to train. On the other hand, with a larger dataset of 50 classes, training time increased 40x to half an hour and the overall accuracy fell to 43%.

Putting these numbers in context, for the 50-class dataset, a classifier that randomly guesses the class would have expected accuracy of 2%. In that context, we see that the Logistic Regression classifier did relatively well.

That said, there is still room for improvement and we perform some error analysis next to understand the types of errors that the classifier was making.

| Number of Classes | Accuracy | Training Time (s) |
|---|---|---|
| 3 | 0.79 | 25 |
| 10 | 0.64 | 122 |
| 50 | 0.43 | 1089 |

TABLE 1: Logistic Regression Result

**3.2.2. Error Analysis.** Linear Regression performs reasonably well.

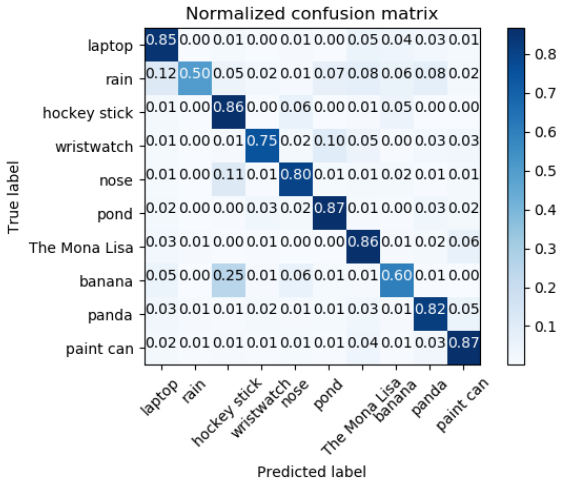


Figure 1. Linear Regression's Confusion Matrix

Looking at the confusion matrix, we can see that Linear Regression performs relatively well. The diagonal of the confusion matrix carries the most weight, indicating it often makes the correct prediction. We notice that in the wrongly classified region, the true label *banana* is highly misclassified, and is often confused with *hockey stick*. This is expected as a hand-drawn banana is very similar to a hand-drawn hockey stick. Thus, this experiment suggests that in order to predict hand-drawn doodles, contrary to our initial belief, we may need a more sophisticated model instead of a simpler model because the quality of the drawing may not be very good.

Figure 2. Banana Doodles

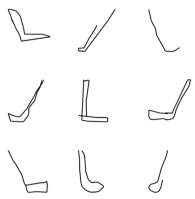

Figure 3. Hockey Stick Doodles

## 3.3. SVM

We implemented Support Vector Machine (SVM) with different four different kernels: Linear, RBF, Polynomial, and Sigmoid.

**3.3.1. Outcomes.** We normalized the inputs to the range $[-1, 1]$ and trained for 500 iterations. The initial parameters that we used are:

- polynomial: degree of 5, coefficient of 1
- RBF: gamma of 1
- sigmoid: coefficient of 1

| Number of Classes | Accuracy | Training Time (s) |
|---|---|---|
| 3 | 0.41 | 204 |

TABLE 2: SVM with Linear Kernel Result

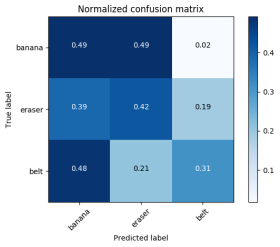


Figure 4. Confusion Matrix for the SVM with the Linear Kernel

| Number of Classes | Accuracy | Training Time |
|---|---|---|
| 3 | 0.62 | 317 |

TABLE 3: SVM with RBF Result

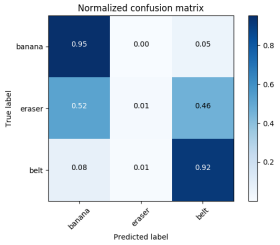


Figure 5. Confusion Matrix for the SVM with the RBF Kernel

| Number of Classes | Accuracy | Training Time (s) |
|---|---|---|
| 3 | 0.52 | 447 |

TABLE 4: SVM with Polynomial (with degree 5) Result

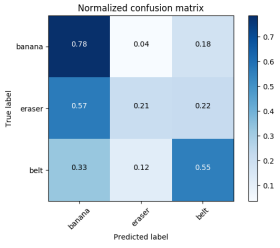


Figure 6. Confusion Matrix for the SVM with the Polynomial Kernel

| Number of Classes | Accuracy | Training Time |
|---|---|---|
| 3 | 0.33 | 478 |

TABLE 5: SVM with Sigmoid Kernel Result

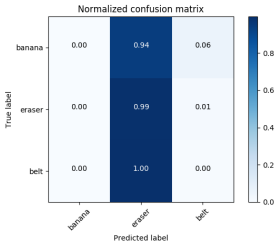


Figure 7. Confusion Matrix for the SVM with Sigmoid Kernel

**3.3.2. Error Analysis.** To our surprise, the SVM performed worse than Linear Regression overall. However, this could be due to the fact that we have not done hyperparameter tuning for SVM.

Among our different choices of kernels, the RBF kernel performs the best, followed by the polynomial kernel with degree 5, then the linear kernel. The sigmoid kernel performed the worst with an accuracy equitable to assigning a category at random.

This is result is consistent with what we expected. The accuracy corresponds to the complexity of the feature space with RBF corresponding to an infinite feature space and the polynomial and linear having fewer features. Although the sigmoid corresponds to a higher dimensional feature space, its corresponding kernel matrix is not guaranteed to be positive semi-definite. If the chosen parameters are not well tuned, the algorithm can perform worse than random [1].

## 4. Next Steps

We will implement a Convolutional Neural Network (CNN) as the last popular image classification method that we want to later improve upon.

The next stage of the project involves proposing, implementing, and evaluating modifications to all these methods, taking into account the aforementioned two distinct characteristics of our doodle dataset as well as the fact that image may be poorly drawn, and does not resemble the object it is trying to depict very well. There can be some confusions (for example between *banana* and *hockey stick*.)

## References


[1] R. Amami, D. B. Ayed, and N. Ellouze. Practical selection of svm supervised parameters with different feature representations for vowel recognition. *arXiv preprint arXiv:1507.06020*, 2015.

[2] Google. Cloud deep learning vm image. 2018. https://cloud.google.com/deep-learning-vm/.