Drawing: A New Way To Search
Nguyet Minh Phu, Connie Xiao, Jervis Muindi
{minhphu, coxiao, jmuindi}@stanford.edu

OVERVIEW
Motivation: Using words can be limited when communicating across cultures and literacy levels. Images are a shared medium of communication that can benefit bridge these divides.

We want to develop an efficient system that recognizes labels of hand-drawn images based on Google's QuickDraw dataset. We implemented a variety of models and found that an altered CNN was best for this task.

DATA
Google's QuickDraw is the world's largest doodling dataset, consisting of hand-drawn images from over 15 million people all over the world.

Examples:
- banana
- hockey stick
- squirrel
- watermelon
- bathtub

Label/Class:

Data features: We used the npy bitmap version of the data. Each drawing consists of raw pixel inputs with values from 0 to 255. We are take advantage of the fact that each image has only two colors, black and white to binarize the pixels.

RESULTS
Training Time Vs Accuracy tradeoff
- Judging purely on the basis of training time / training resources, Logistic Regression was the fastest with a training time of about half an hour.
- However, if one has more resources available to train more sophisticated models, we find that CNN does the best for this task with a training time of 0(hours) ~ 4+ hours.

CONVOLUTIONAL NEURAL NET
- A doodle is a simple image, thus some components of a CNN may be removed. We implemented a CNN, then simplified it by progressively removing layers and dense units to analyze the impact on accuracy and runtime.

Training Time (s)
- The Convolutional Neural Network is a good candidate as it can perform worse than random [1] parameters are not well tuned, the algorithm can perform worse than random [1]

Transfer Learning
- Training a deep-learning model from scratch is time-intensive. Transfer learning is one way to leverage pre-trained models for our task. We explored whether using pre-trained winning models from the ImageNet competition could help save time and improve accuracy for our task of doodle classification.

FUTURE WORK
- Develop our most promising approach: More extensive experiments to determine the effect of each layer in CNN
- Explore a different approach of transfer learning: using transfer learning as a fixed feature extractor for logistic regression
- Develop other efficiency metrics: data efficiency - working on efficiency in conjunction with smaller datasets

REFERENCES

SUPPORT VECTOR MACHINE
- In Support Vector Machine with Kernel, some kernels may be more suited for the task of doodle classification, thus we implemented a SVM with four different kernels (Linear, RBF, Polynomial, Sigmoid) to identify the best one for this task empirically.

SVM
- Surprisingly SVMs performed worse than linear regression overall.
- Suspect it is due to lack of parameter tuning: For the sigmoid kernel, if the chosen parameters are not well tuned, the algorithm can perform worse than random [1]

MODEL
- For baseline, we used Logistic Regression, a simple and fast to train model using numpy bitmap of raw image pixels.
- In Support Vector Machine with Kernel, some kernels may be more suited for the task of doodle classification, thus we implemented a SVM with four different kernels (Linear, RBF, Polynomial, Sigmoid) to identify the best one for this task empirically.

LOGISTIC REGRESSION
- Linear Regression performs relatively well.
- Banana is often confused with hockey stick which shows that there is a need for a more sophisticated model to make up for training quality

PLATT'S RBF KERNEL

Table 1. Results for linear regression

<table>
<thead>
<tr>
<th>Classes</th>
<th>Accuracy (%)</th>
<th>Training Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>64.64</td>
<td>43.89</td>
</tr>
<tr>
<td></td>
<td>122</td>
<td>1089</td>
</tr>
</tbody>
</table>

Table 2. Results for SVMs

<table>
<thead>
<tr>
<th>Number of classes</th>
<th>Accuracy (%)</th>
<th>Training Time (s)</th>
<th>Binarized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>v1: full CNN</td>
<td>86.59</td>
<td>82.12</td>
<td>3047</td>
</tr>
<tr>
<td>v2: remove 2nd comLayer</td>
<td>85.5</td>
<td>76.1</td>
<td>2450</td>
</tr>
<tr>
<td>v3: v2 + remove 1st MaxPoolLayer</td>
<td>82.16</td>
<td>70.27</td>
<td>2332</td>
</tr>
<tr>
<td>v4: v2 with a dense layer of 64 units</td>
<td>85.6</td>
<td>70.29</td>
<td>628</td>
</tr>
</tbody>
</table>

Table 3. Results for CNNs

Table 4. Results for transfer learning

Table 4. Results for transfer learning

<table>
<thead>
<tr>
<th>Models on 3 classes</th>
<th>Accuracy (%)</th>
<th>Training Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception v3</td>
<td>48.77</td>
<td>(stopped at 20 epochs)</td>
</tr>
<tr>
<td>MobileNet</td>
<td>75.4</td>
<td>102.95</td>
</tr>
<tr>
<td>VGG</td>
<td>N/A</td>
<td>(ran out of memory)</td>
</tr>
<tr>
<td>ResNet50</td>
<td>62.72</td>
<td>152.32</td>
</tr>
</tbody>
</table>