Guess-a-Sketch

Lucia Zacek (llz22), Urvashi Deshpande (uad3), John Fernandez (jmf433)

New changes and status updates in blue.

AI Keywords: Machine Learning, Deep Learning, Convolutional Neural Networks, Support Vector Machines, Random Forest, Game Playing

Application Setting:

We are planning to develop an AI to classify drawings based on the online game Skribbl.io, which is similar to pictionary. The AI should take in a word length and an image (potentially one that is in the process of being drawn) and classify it according to a set of possible words.

Our application setting has not changed, although we are currently limiting the models to only a word bank of size 250, and have not implemented word length yet.

Description:

Section 1: The Big Picture

The AI we are creating will have two main objectives: classify drawings correctly with high accuracy and classify them correctly at a faster speed than a human. We will provide it with a large dataset of images and labels (using the dataset from the first reference below and further images from google if necessary). Initially, we will focus on optimizing the first goal of high accuracy. We will have a built-in limit on how many words it can guess in a set amount of time (for example, we might limit it to typing only as fast as the average human).

When the AI receives an image and word length, it should match the image to the most likely label from words of the same length. We will try convolutional neural networks, SVM, and random forests as machine learning classification techniques. Once we began implementing both CNNs and random forests, it was clear that each would take a substantial amount of time, so we decided not to attempt using SVM on the data. Eventually, we want it to be able to take as input a ~~video~~ or set of images of a drawing in process and classify it. We will start with a small set of words from the first reference paper, and hopefully later expand to the entire set of words in Skribbl.io.

Section 2: The AI Core

(2-3 paragraphs)

The AI Core of this project stems from the image classification task. The three main machine learning techniques we are planning to try for this part of the project are convolutional neural networks, ~~SVMs~~, and random forests. The benefit to using convolutional neural networks is that they are translation-invariant, so the location of the drawing on the canvas does not matter for

identification. We will be coding this in Python and using libraries such as numpy and pytorch. Additionally, we used libraries skLearn for encoding string categories as numeric, and pickle to save our trained models.

~~When training the SVM, it will be structured as several binary classification tasks, where task i addresses the question of whether the drawing belongs to label i (1) or to any of the other labels (-1).~~ The benefit to having black and white images (of standardized size) is that we can represent them simply as a vector, with each index being the black intensity in a particular pixel of the image. So, if we use 50x50 drawings for example, we can represent those as a 1-by-2500 dimensional vector. Then, we can do the classification task using ~~SVM~~ and Random Forest on this vector. ~~Since SVM only does binary classification, we would need to train one SVM classifier per word.~~ We are also open to exploring other techniques as the project develops, and are not yet set on using any one machine learning technique. After completing initial CNN and random forest models, we trained them on very small samples to verify that they run correctly. Later on, we will train each model on larger datasets and then determine which model is more effective.

When we move on to classification of the drawing process (instead of just the completed image), we will try running the same models above on individual frames of the drawing as it is in process. As we progress in the development of this classification AI, we will be testing and modifying hyperparameters related to time limits and number of guesses allowed to determine what is the right balance of time and guesses for our model. If we find that the model is guessing too quickly for a human to compete with, we can further limit its number of guesses. If it is unable to guess correctly on a significant proportion of images, we will consider increasing the guessing speed.

Section 3: Evaluation

We will evaluate the performance of our project in two steps. First, we will measure the speed and accuracy with which the model can correctly classify an unseen, complete drawing of a given word. ~~For our test data, we will get members of WICC and URMC to provide testing sketches for all the labels present in our training data~~. Second, we will expand the models to provide sketches one line at a time to simulate a human drawing an object in real-time, and have the model guess the correct label for the drawing under timed conditions. We will ask human players to play against the model and evaluate the model on how quickly it can guess the word compared to human opponents. We plan to ask students in WICC and URMC to try playing against the AI for this step. We were not able to generate data from the collaboration of WICC or URMC and have instead decided to split up the data of our original 200000 image dataset into a training set and validation set. We had 80 images to a label in our original dataset and have decided to split it up into a training set with 75 images to a label and a validation set with 5

images to a label. This leaves us with a training set of 18750 images and a validation set of 1250 images to train our models on.

In terms of measuring the accuracy of our project on completed images, we will have two ways of measuring the accuracy of our model. The first of which is just a standard test error calculation, where images that are classified correctly have 0 error and misclassified images have an error of 1. The second calculation will take into account the difficulty of the words. We will pull the verified statistics of the difficulty of words within the Skribble.io database (see second reference) and use them to weigh each testing image differently. For example, a word in the database that has been shown to be more difficult to guess will receive a higher weight than one that has been shown to be easier to guess. We will evaluate the accuracy of each of the three models described above and compare the results, settling on the one with the lowest overall classification error.

Section 4: Timeline

- February 17: Create framework for project, put small training dataset into usable form (done)
- February 24: Write CNN for image classification (done)
- March 3: Initial testing of CNN with small dataset (done)
- ~~March 10: Write SVM model~~ (removed)
- ~~March 17: Test SVM model with small dataset~~ (removed)
- March 10: Write random forest model (moved up, done)
- March 17: Test random forest model with small dataset (moved up, done)
- March 24: Separate existing data into training/testing (added, done)
- March 24: ~~Collect Data from WICC and URMC~~ (removed)
- March 24: Complete Milestone 3 (done)
- **March 29: Milestone 3: Status Report**
- ~~March 31: Write random forest model~~ (moved up)
- April 7: Train and test CNN with large datasets (added)
- April 7: Train and test Random Forest with large datasets (added)
- ~~April 7: Test random forest model with small dataset~~ (moved up)
- April 7: Decide on best model out of the ~~three~~ two
- April 14: Evaluation step 1: test final model on complete drawings, expand dataset to include all words in Skribbl.io dictionary
- April 21: Evaluation step 2: Modify test data to display drawings one line/one piece at a time
- April 28: Evaluation step 2: Test model on modified test data, start working on final presentation
- May 5: Additional testing with human opponents, practice final presentation, finish **Milestone 5**

- May 8: **Milestone 4: Project Presentation**

Section 5: Resources

Software

For writing the machine learning models, we plan to use Python's PyTorch and ~~TensorFlow~~ libraries. We will also use GitHub for project management. VSCode will be used as our main IDE.

Data Resources

We will use the dataset of images from the first reference below. We separated this dataset into training and testing datasets, of size 18750 and 1250 respectively. When we expand the project to include all words in the Skribbl.io dictionary, we will data-mine for sketches on google to use in our training data. We may use Skribble.io (the game) to gather data from other peoples' guesses.

Images for testing will be collected from having other students draw sketches of various labels. ~~We also eventually plan to ask students in URMC and WICC to play against our model, providing testing data.,~~

Section 6: Previous Related Projects: N/A

Section 7: References

How do Humans Sketch Objects?
http://cybertron.cg.tu-berlin.de/eitz/projects/classifysketch/
Skribbl.io Base Words
https://skribbliohints.github.io/
Feature Extraction Using Convolution
http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/