Data Selection Proposal

Parth Gupta

Feb 20, 2022

1 Dataset

Hand-drawn doodle recognition is the problem that I plan on tackling, and I will be using a subset of Google's Quick Draw dataset to solve this problem. The reason I chose this dataset is because it contains over 50 million drawings across 345 classes which is sufficient to achieve a high model accuracy. In addition, this dataset is easy to use as all the data has been formatted into labelled `JSON` objects.

2 Methodology

I plan on creating a classifier that is trained on the Quick Draw dataset so that it can classify doodles/sketches drawn by a user. The dataset is already labelled, so a supervised approach is suitable for this project. The raw dataset contains vectorized drawings in a `JSON` format, but also comes with preprocessed drawings. These drawings have been scaled into a 256 x 256 region and vectorized in a `ndjson` format. Furthermore, the dataset also has these simplified images in 28 x 28 greyscale Numpy bitmap format which can be used to train CNNs if that route is opted for. I will be using the preprocessed drawings for training.

Pre-existing research for sketch drawing recognition shows that convolutional neural networks and recurrent neural networks are best suited for this task. The Quick Draw game developed by Google uses an RNN which takes the input as a sequence of strokes and recognizes the class that the user tried to draw. Convolutional layers, LSTM layers and a softmax output layer are used to classify the doodles. Another research showcases a multi-graph transformer (MGT) that represents the drawings as sparsely connected graphs. I will be using a CNN since there are pretrained models such as `MobileNet, ResNet` etc. After browsing some Kaggle projects, CNNs seem to work well with this problem as people get around 80-85% accuracy with them. Another advantage of using a CNN over an RNN is that the CNNs have already been trained on popular datasets like `COCO` and `ImageNet` which makes them more robust. I will get some metrics before deliverable 2 by trying these models out myself and then choosing the right one for my final application.

As this is a classification problem, I will be reporting a confusion matrix along with accuracy, precision recall to evaluate the model's performance. Cross-entropy loss is another metric that I'm thinking of reporting. I hope to beat baselines of 70% which are the results of several `K-means` and `Random-Forests` classifiers from implementations on Kaggle.

3 Application

I will be integrating this project into a web application. The user will be able to draw on the screen and there will be an option to submit the sketch. The model will then make a prediction on the possible sketch class and report that back to the user using a speech API (`react-speech-kit` can be used). For the front-end of the app, I am planning on using `React.js` since it will allow for the use of states. I'm still undecided for the back-end, I do want to explore `tensorflow.js` so that I can seamlessly integrate models with JavaScript but it would limit the variety of models available. The option to use `Flask` and Python for the backend is always available if I don't plan on going with `tensorflow.js`.