Initial results

Parth Gupta

March 16, 2022

1 Problem statement

The problem that I will solve is recognition of free-hand sketches using the Google Quick Draw dataset. This dataset is open source and contains around 50 million doodles which makes it ideal to use.

2 Data processing

There will be no-need for preprocessing as the dataset already contains processed images in a variety of different formats. The doodles have been simplified into a 256 x 256 region in an `.ndjson` format. Also, these drawings are also offered as Numpy bitmaps. Specifically, they have been rendered into 28 x 28 greyscale bitmaps with the `.npy` extension.

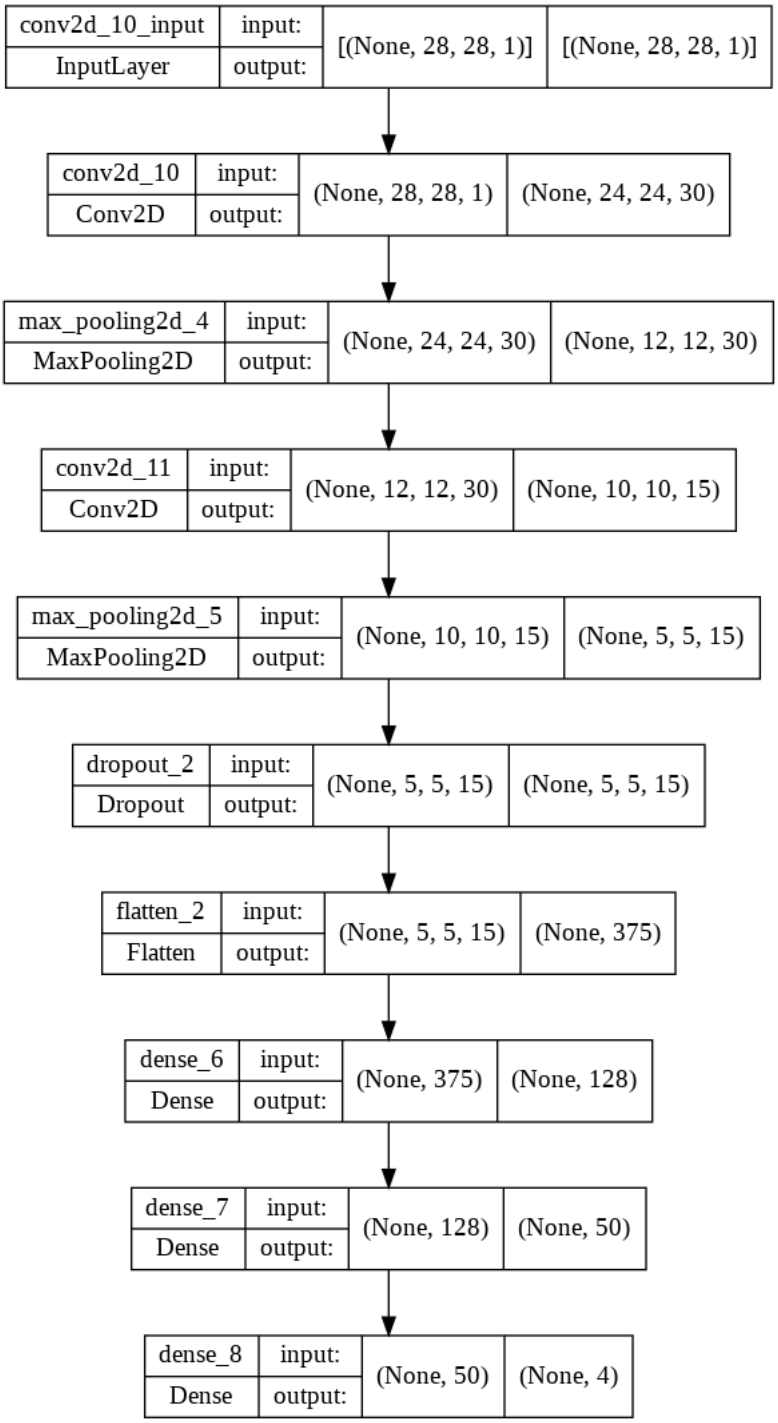
3 Models and Results

3.1 Baselines

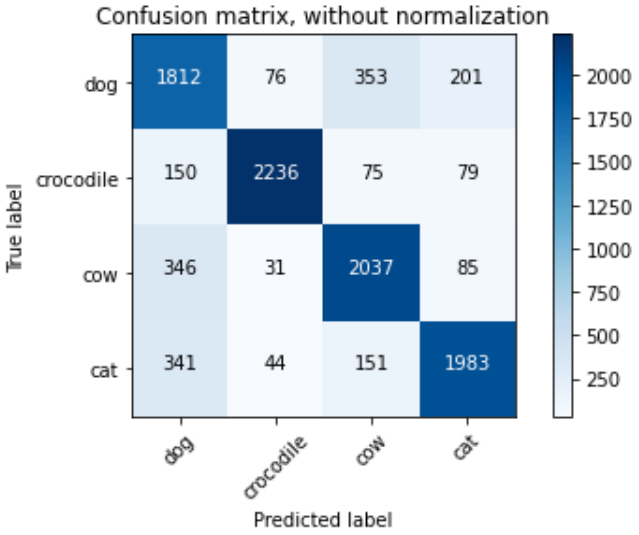
I used a `RandomForests` classifier and a Support Vector Machine as baseline models for the image classification task. `RandomForests` is known for its classification strength which is why it was chosen as a baseline and the addition of SVM was just for exploratory purposes. `Sci-kit learn` was the chosen library for these tasks as it makes it fairly easy to implement these models. The `RandomForests` classifier achieved a classification accuracy of 69% and the SVM an accuracy of 61%. I hope to perform `GridSearch` in the future to make these baselines more robust and improve accuracy.

3.2 Convolutional Neural Network

I used a `CNN` as my main model for the classification task. I built the CNN using the `Keras` library since it makes it easy to add layers and other features. I used a simple train-test split to generate the dataset and I had to convert the 28 x 28 images back to one-hot encoders as Keras uses that. A difficulty that I faced was to decide the architecture of the model. To solve this, I browsed online tutorials for CNNs and chose to combine some of the features from each of the tutorials. Mostly the architecture is built using this tutorial. Here's an image that describe the architecture of the model.


conv2d_10_input
InputLayer
input:
output:
[(None, 28, 28, 1)]
[(None, 28, 28, 1)]
conv2d_10
Conv2D
input:
output:
(None, 28, 28, 1)
(None, 24, 24, 30)
max_pooling2d_4
MaxPooling2D
input:
output:
(None, 24, 24, 30)
(None, 12, 12, 30)
conv2d_11
Conv2D
input:
output:
(None, 12, 12, 30)
(None, 10, 10, 15)
max_pooling2d_5
MaxPooling2D
input:
output:
(None, 10, 10, 15)
(None, 5, 5, 15)
dropout_2
Dropout
input:
output:
(None, 5, 5, 15)
(None, 5, 5, 15)
flatten_2
Flatten
input:
output:
(None, 5, 5, 15)
(None, 375)
dense_6
Dense
input:
output:
(None, 375)
(None, 128)
dense_7
Dense
input:
output:
(None, 128)
(None, 50)
dense_8
Dense
input:
output:
(None, 50)
(None, 4)

For an evaluation metric, I decided to use a confusion matrix to see how well the model performs.



Some of the changes I faced were deciding on the architecture for the CNN, and the type of evaluation metrics I want to report. I researched different architectures and this particular architecture gave an accuracy score about 80%.

4 Next Steps

I plan on using `GridSearch` for tuning the hyperparameters and try experimenting with different numbers of layers to see how the CNN performs. For this initial testing, I only used 4 classes and they were all animal classes. For my final project, I would like to create an animal doodle recognizer so I'm definitely going to be using more classes than (maybe around 10-15). I also plan on reporting more metrics and performing cross-validation to make the model perform better.