Advanced Machine Learning
(UE15CS415)

Easy-Peasy Draw

Team members:
Aishwarya Manjunath (01FB15ECS019)
Akanksha G (01FB15ECS023)
Anusha Kamath (01FB15ECS048)

Problem Statement:

This is a game built with machine learning. You draw a Geometrical Figure, and a neural network tries to guess what you're drawing. Of course, it doesn't always work. But the more you play with it, the more it will learn. So far we have trained it on a few concepts, and we hope to add more over time. We made this as an example of how you can use machine learning in fun ways. Just like Google's Quick Draw, we have managed to do a classification of what shape the user is drawing and also prediction of next possible stroke to help user in his drawing. This project is a fun way to see how Machine learning techniques can be applied in fun areas.

Potential Approaches:

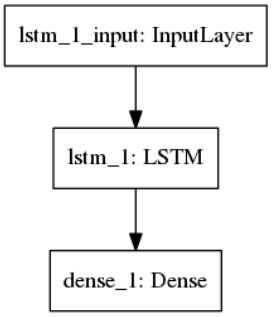
There can be many approaches in solving the two perspectives of the problem. Some approaches that are already formulated and built are:

1. Sketch RNN , the original model developed by Google's Brain team for Quick Draw. This paper originally picks the idea of combining Variational AutoEncoders and Recurrent Neural Networks, with a slight modification of using Bidirectional RNN and modelling the target output as a sample from a GMM. [1]
2. Using CNN+LSTM to classify the handwritten sketches, the developer uses a stack of 3 convolution layers followed by a LSTM layers and then a Dense layer with "c" neurons for "c" classes and softmax as the activation function as a model. The accuracy obtained in the following approach is 92.20%. [2]
3. The same dataset has been used for many other creative and analytical projects. The link [3] will be useful to look through them.

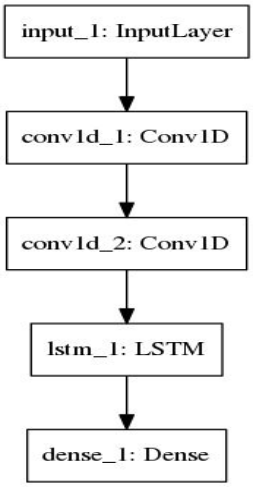
Approaches Used:

1. Classification:
    a. Geometrical figures:
    5 classes comprising of line, square, circle, hexagon and triangle were considered

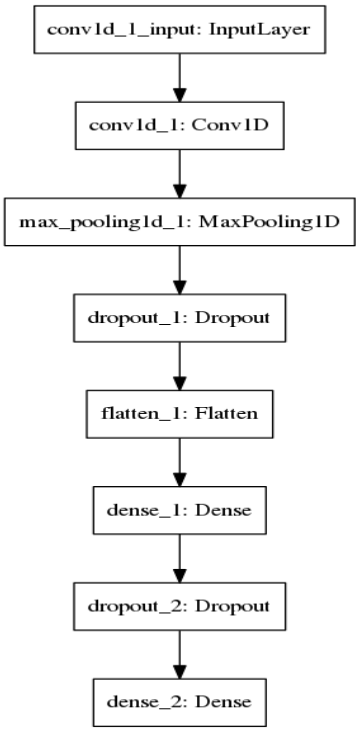
i. LSTM:



ii. CNN + LSTM



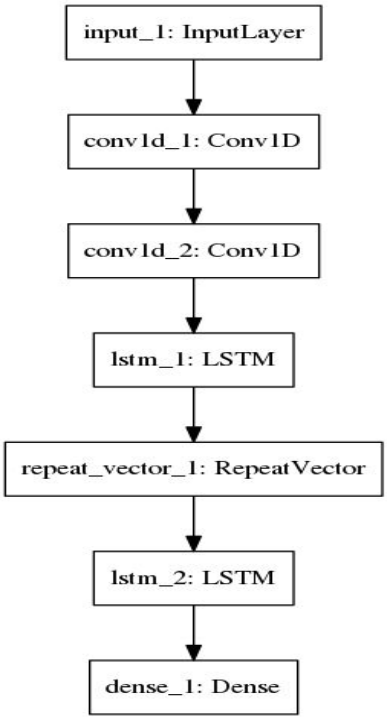
iii. CNN



b. Animals

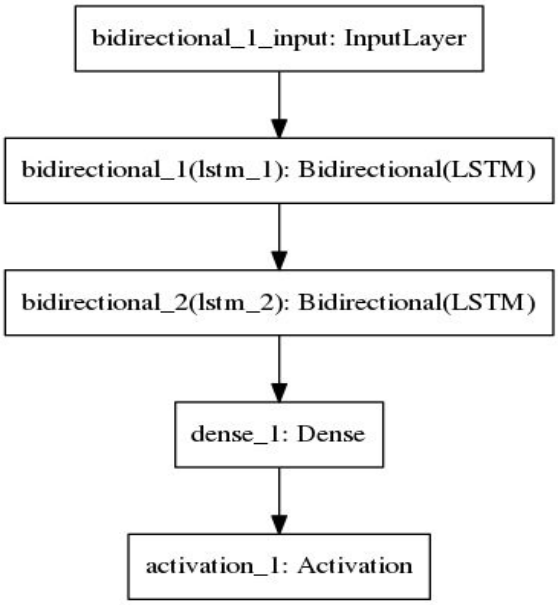
i. CNN + LSTM

Here 7 classes comprising of ant, bear, dog, fish, bird, butterfly, and cat were considered

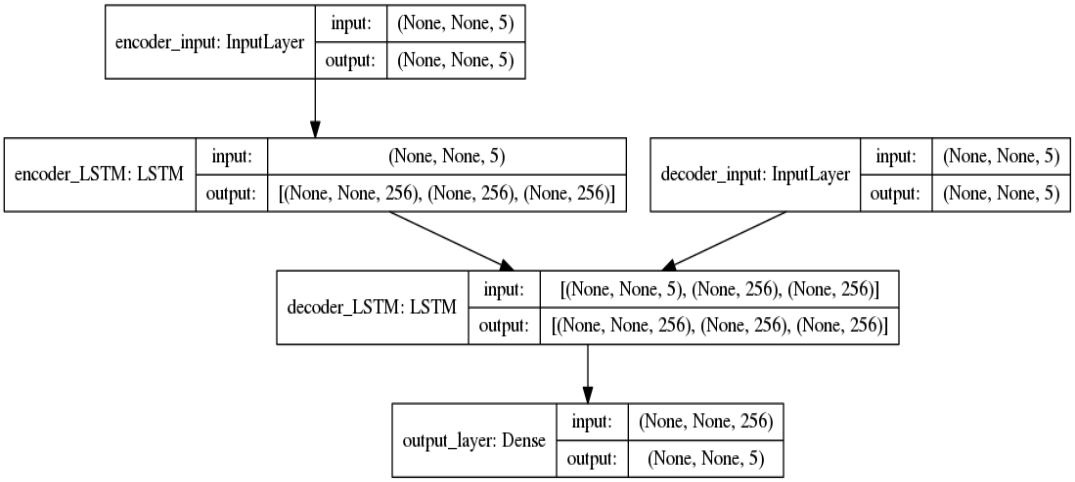


ii. Bi-directional RNN

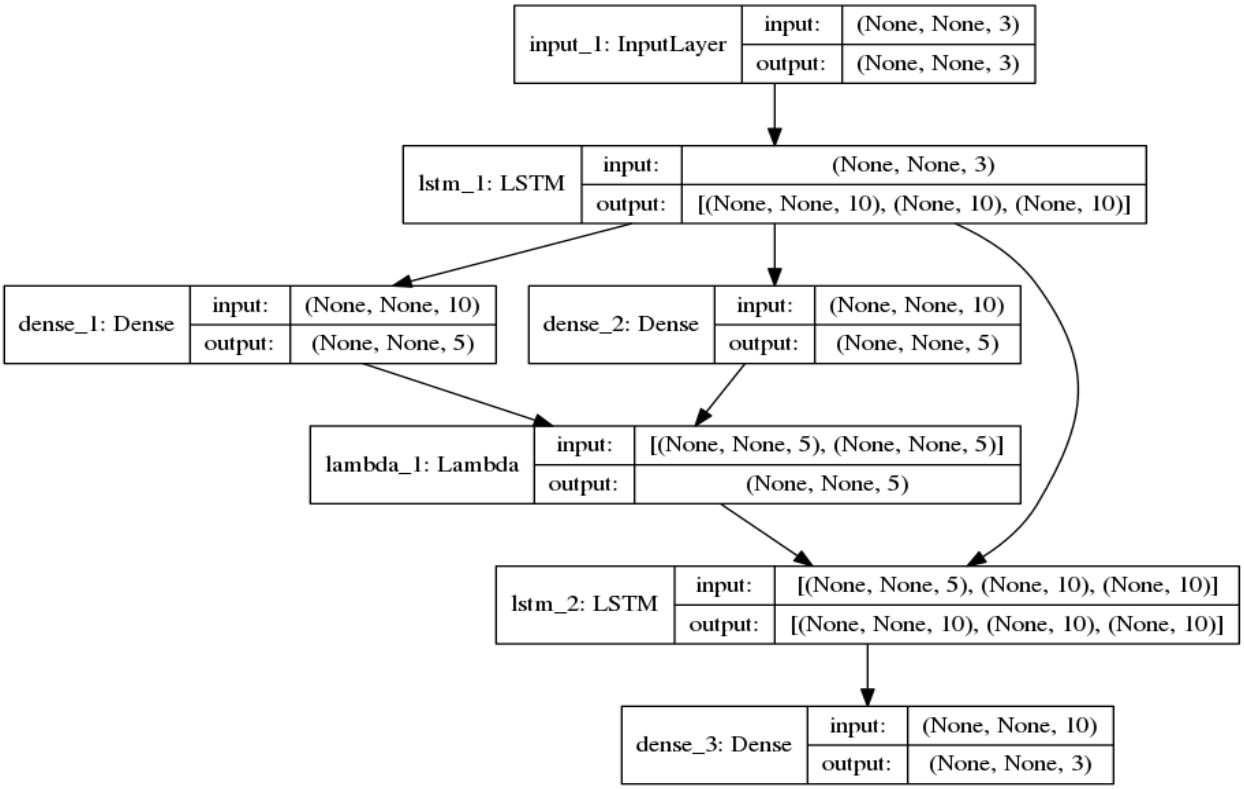
Here 5 classes comprising of ant, bear, fish, butterfly and cat were considered



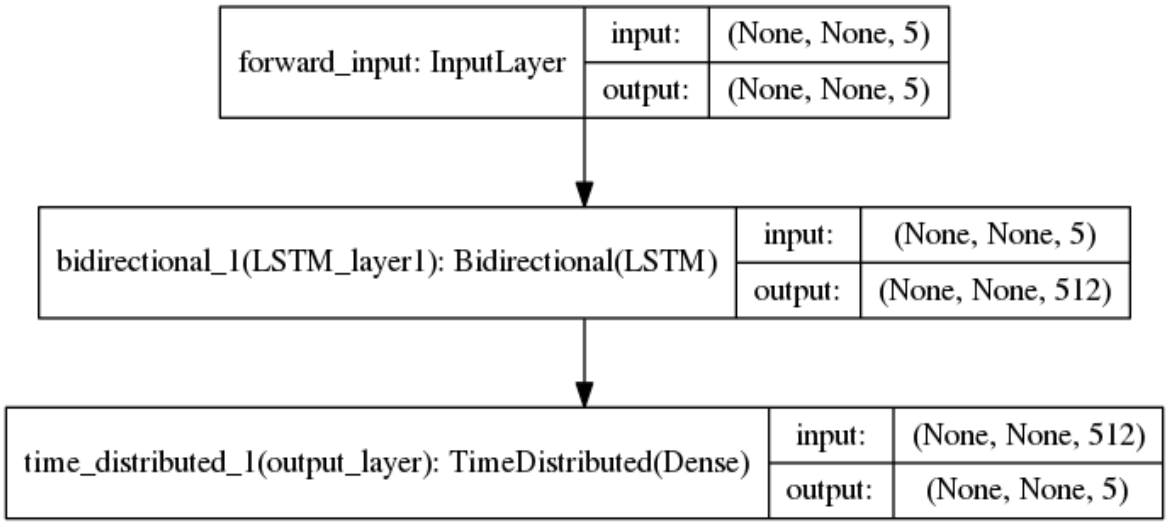
2. Prediction:
   a. LSTM Autoencoder (Seq 2 Seq) :



b. Variational Autoencoder (Seq 2 Seq)



c. Bidirectional LSTM model



Dataset:

We have taken the dataset of Google's Quick Draw contributed by the players of the game. The drawings were captured as timestamped vectors, tagged with metadata including what the player was asked to draw.
For each of the classes, a ndjson contains data of number of drawings, where each line contains one drawing with following details:


```
{
  "key_id":"5891796615823360",
  "word":"ant",
  "countrycode":"AE",
  "timestamp":"2017-03-01 20:41:36.70725 UTC",
  "recognized":true,
  "drawing":[[[129,128,129,129,130,130,131,132,132,133,133,133,133,...]]]
 }
```


And each drawing array as:

```
[
 [  // First stroke
   [x0, x1, x2, x3, ...],
   [y0, y1, y2, y3, ...],
   [t0, t1, t2, t3, ...]
 ],
 [  // Second stroke
   [x0, x1, x2, x3, ...],
   [y0, y1, y2, y3, ...],
   [t0, t1, t2, t3, ...]
 ],
 ... // Additional strokes
]
```

Tests and Results:

1. Classification
    a. Geometrical figures classes:
    Here 5 classes comprising of line, square, hexagon, circle and triangle were considered
    Design Parameters:
        - LSTM - 50 LSTM cells , 10 epochs
        - CNN+ LSTM - CNN layers with 5 filters(filter size = 2) and LSTM layer with 50 cells, 10 epochs

- CNN - Convolutional Layer(32 filters (filter size = 2)) and MaxPool layer (pool size=2), 100 epochs
- For all the above models, Adam was used as the optimizer and Categorical crossentropy was used as the loss function.

Model	**Training Accuracy**	**Training Loss**	**Testing Accuracy**
LSTM	96.32	13.10	96.25
CNN + LSTM	96.22	13.10	96.11
CNN	91.79	26.98	94.33

Comparison of precision and Recall of the three models

Model	**Recall**	**Precision**
LSTM	96	96
CNN + LSTM	96	96
CNN	94	94

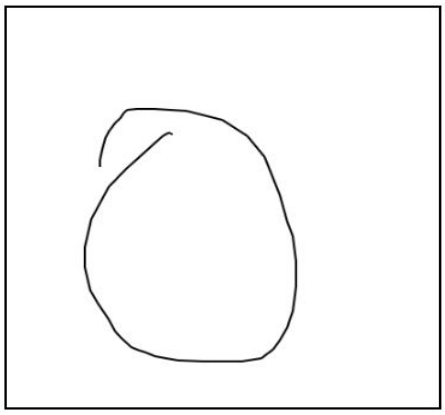
b. Animals :
Here we increase the no of classes to 7 and categories like ant, bear, bird, cat, butterfly, fish and dog were considered.
Design Parameters:

- CNN + LSTM :2 CNN layers with 8 and 16 filters and filter size = 2, 2 LSTM layers with 50 cells were considered. No of epochs = 30.
- Bidirectional RNN - 2 bidirectional layers with 50 LSTM cells were considered. No of epochs = 30.
- For all the above models, Adam was used as the optimizer and Categorical crossentropy was used as the loss function.

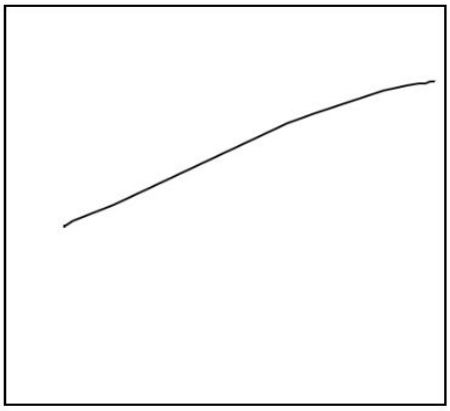
Model	**Training Accuracy**	**Training Loss**	**Testing Accuracy**
CNN + LSTM	75.35	69.9	74.69
Bidirectional RNN	89.96	29.9	89.7

Results: (Geometrical Figures - From LSTM Model)

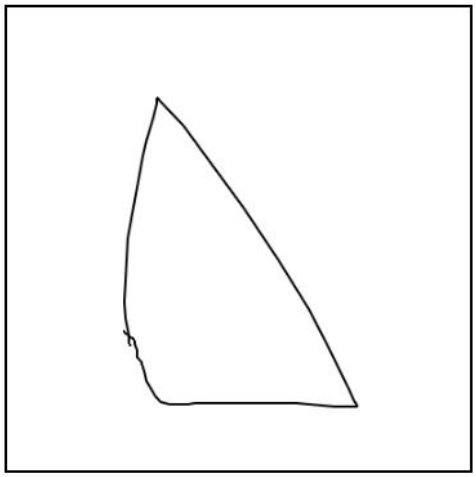
START
circle



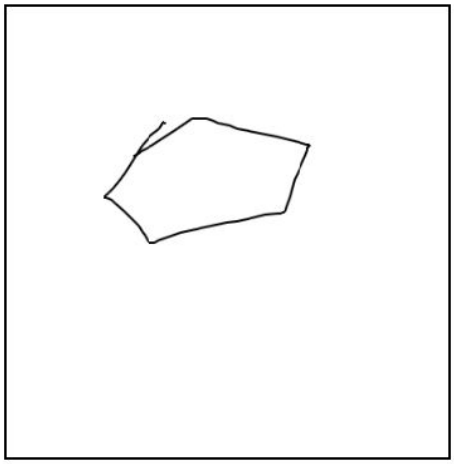
START
line



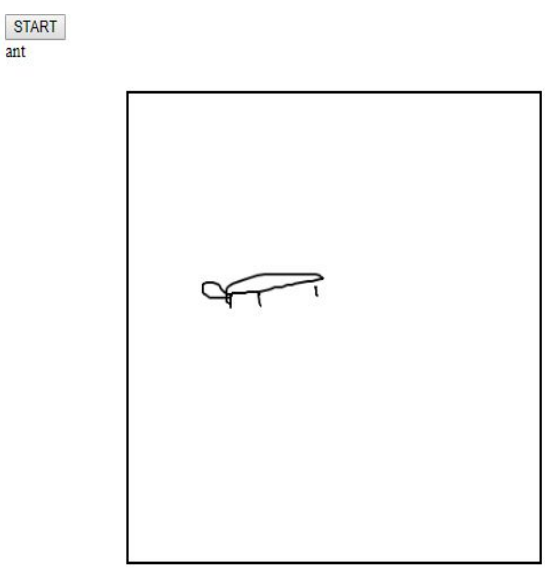
START
triangle

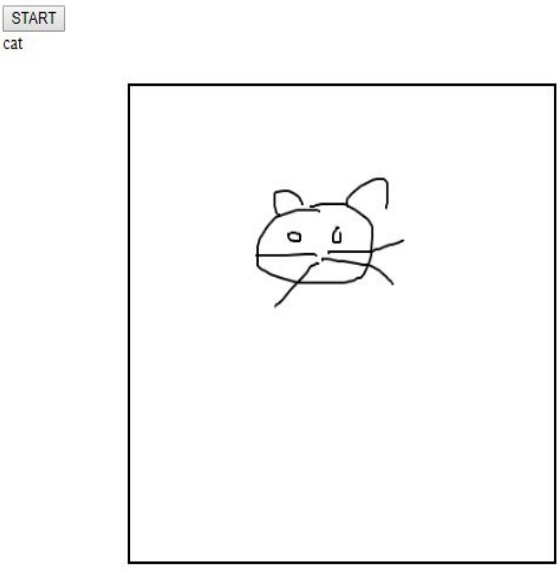


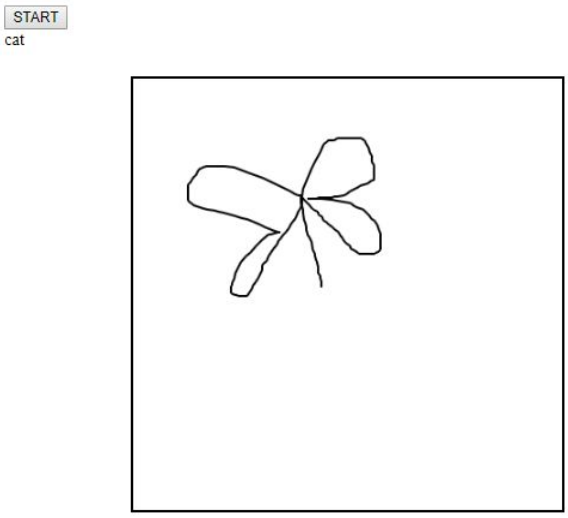
START
hexagon

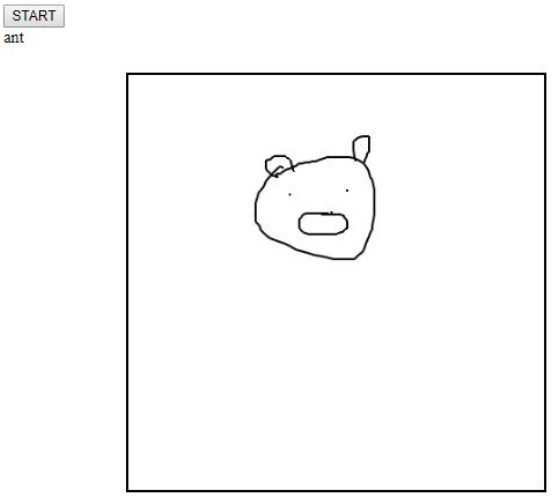


Results (Animals - From Bidirectional model):









As it can be seen the animal prediction was not good enough, some of the reasons could be:

- When two classes have similar strokes while drawing, the models are not able to capture the subtle details.
- In the above model, for each drawing 400 points was considered. This was due to limitations in computing resources. So this limits the learning of the model. It was also observed that when no of points was increased from 208 to 400, the accuracy improved. If number of points considered were greater than 400, the accuracy could have been better.

2. Prediction:
   Epochs = 30, Optimizers - adam / rmsprop, loss - mse/ (mse+KLDivergence)

Model	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
Autoencoder	88.37	248.06	88.19	252.52
VAE	79.09	306.09	81.55	313.54

Even though the accuracies of the above are high they failed to construct the sketch on the UI when the predictions were plotted. As we can observe the loss in all the above is very high and these simple models couldn't capture the pattern in the data.

Possible Reasons:

1. We believe that instances of the hand drawn sketches are very different and a simple model was not able to take in the complexity with such limited training epochs.
2. In a sequence to sequence training we need the entire encoder input sequence to get the context which might not be as productive in situations as this where we have a part of encoder input.
3. The Bidirectional LSTM's output was comparatively visible that other predictions this can be due to the fact that it takes into account both the possible ways of starting a drawing. The below images are predictions of our bidirectional model, it shows that it's able to capture only a small part of the train data(eg: a straight line in square and a curve for a cat)

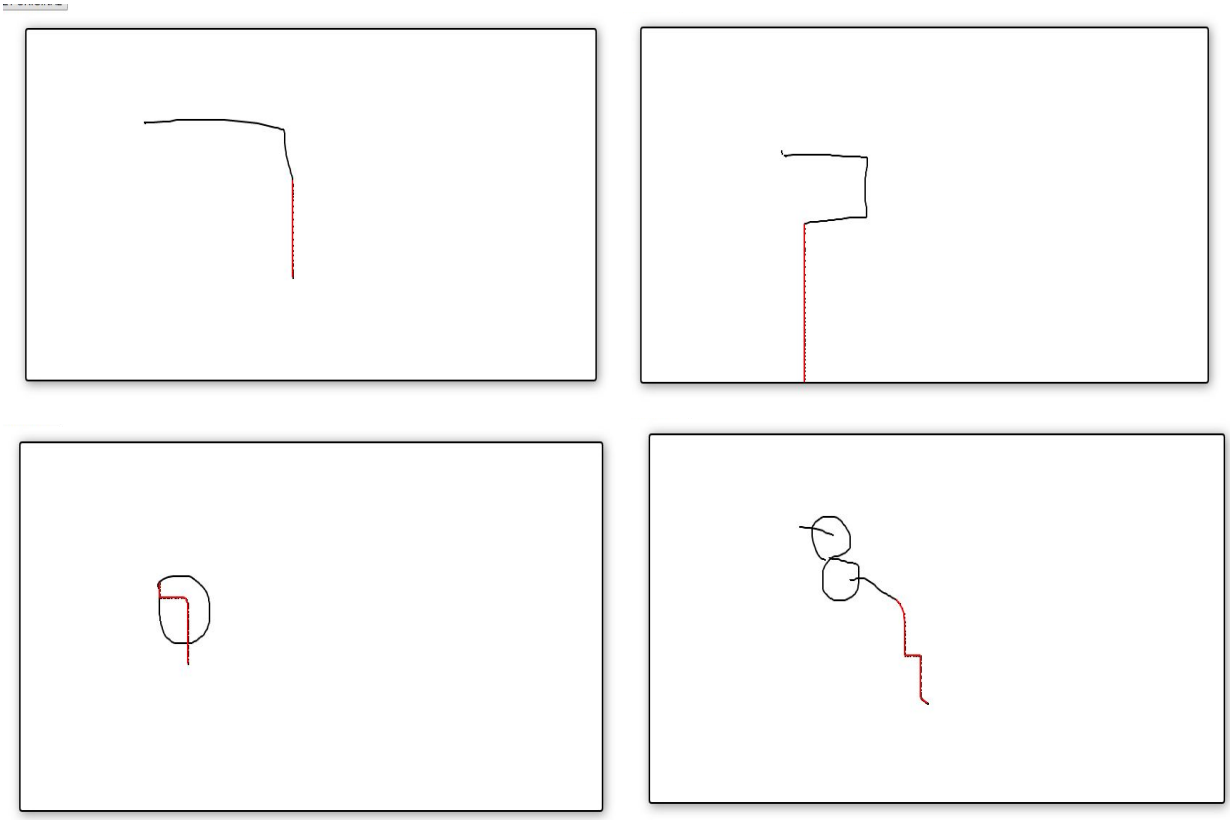


Figure: Prediction of line in a square model and a curve in cat model

References:

[1] Ha, David, and Douglas Eck. "A neural representation of sketch drawings." *arXiv preprint arXiv:1704.03477* (2017).

[2] https://github.com/zaidalyafeai/zaidalyafeai.github.io/tree/master/sketcher

[3] https://github.com/googlecreativelab/quickdraw-dataset#projects-using-the-dataset

[4] https://www.tensorflow.org/tutorials/sequences/recurrent_quickdraw