Machine Learning Project Report

Team Member:

Chang Ye NetID: cy1365
Zuolun Xiao NetID: zx798
Mingda Du NetID: md4012
Zhaoyan Li NetID: zl2532

Abstract

In this project, we aim at designing a model to classify our data. We use training data to build our LSTM model and use train_and_evaluate estimator to figure out the accuracy.

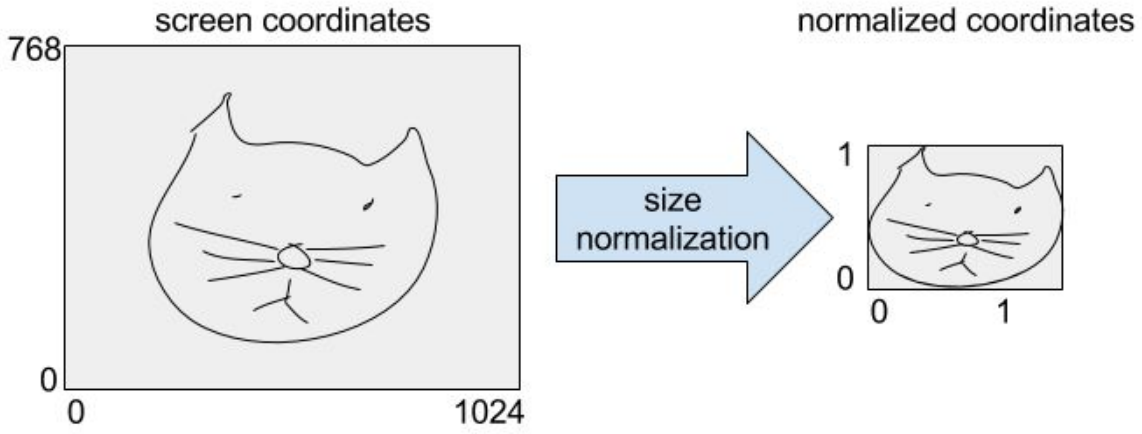
1. Introduction

The purpose of the project is to build a machine learning software to recognize sketch drawings of objects.
The recognition is performed by a classifier that takes the input, given as a sequence of strokes of points in x and y, and recognizes the object category of the drawing.

need to modify
We analyze the problem, and choose to use recurrent neural networks for the classification.
Our training and testing result...
need to modify

2. Data

The data source of the project is from:
https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/ .



```
[
  [  // 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
]
```

Images are ".ndjson" format files, which have features such as "key_id", "word"(means the image belongs to which species), "countrycode", "timestamp", "recognized", "drawing"(strokes of points in x and y, and it's time), etc. First, We use 550000 sketch images of 50 classes, separate it into training and evaluating part: 10000 train images and 1000 test images each class. We extract the features from strokes, simplified every sketch's information, and transformed them into images, scaled into 256 x 256 region, aligned to the top-left corner, each pixel normalized into [0, 255].

Then we change image files into ".tfrecord" files as the input of model training and evaluating, using "create_dataset.py" using Colab (the process is in "new_convert.pdf" or "new_convert.ipynb"). The format of input data of training and testing is a ".tfrecord" shard and a ".classes" file.

```
eval.tfrecord-00000-of-00001  training.tfrecord-00000-of-00001
eval.tfrecord.classes         training.tfrecord.classes
```

3. Approach

The goal is to recognize the category of the object in the drawing. This is a classification problem, so we will not be using regression models.

PCA is not suitable for image data processing. And SVM could be used to process image data, but accuracy is low.
Neural Network could capture all different aspects in image, that is, it could automatically select useful features in image.
Since we need to recognize a drawing, which need capture the relation in a sequence of strokes. RNN is the most suitable model for sequence problems.
And it has several image classes. so, it's a multiclass classification. We uses fully connected layer with softmax to predict the class label.

4. Model

Our model uses LSTM(Long Short-Term Memory) model, which is used to recognize the data more easily and flexibly. In this model, we design four different layers:

Input layer

First all the data are put into a input layer. This part we design for both parse and storage. In the parse function, every record will be transformed in to a tensorflow, which keeps the features like “ink” and “shape”. According to the ModeKeys, we extract the feature “class_index” as labels for future classification use. Furthermore, we use the map function to process 10 files uniformly at the same time. In the end, we shuffle the data. Since inputs are different in size, we use padded_batch to pad them.

Convolution layer

After preprosessing the input, we begin our main model. The first layer is convolution layer: we extract the ink from features. We set the convolution layer number to 3. To avoid over-fitting, we add a dropout layer. For convolution function, we use 1-dimension function tf.layers.conv1d().

CUDNN RNN layer

In order to improve the computing and executing efficiency, we take the advantage of CUDNN for GPU computing. For all the images, they consist of different inks. And inks consist of points. So when we draw the picture, we have different priorities in points. We uses them as sequences. So our data fit RNN model more.

Fully connected layer

In the fully connected layer, each neuron in this layer is fully connected to all neurons in the previous layer.

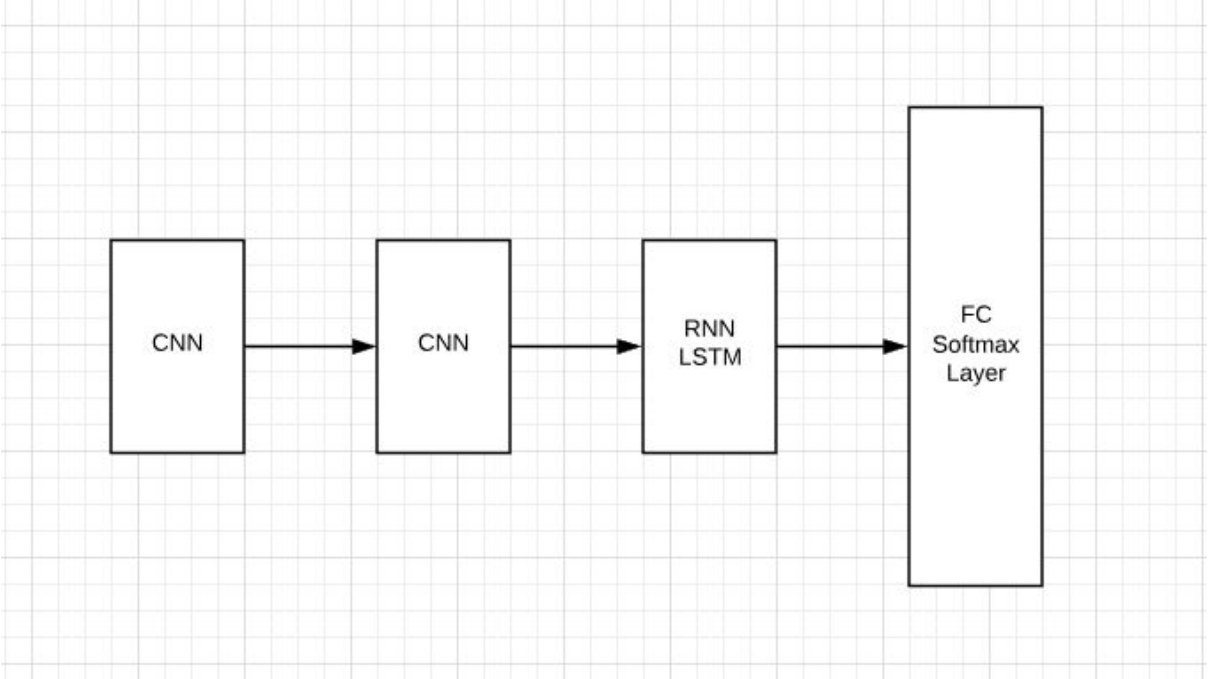
Optimizer

In order for the data to converge quickly, we choose tf.contrib.layers.optimize_loss in Tensorflow.

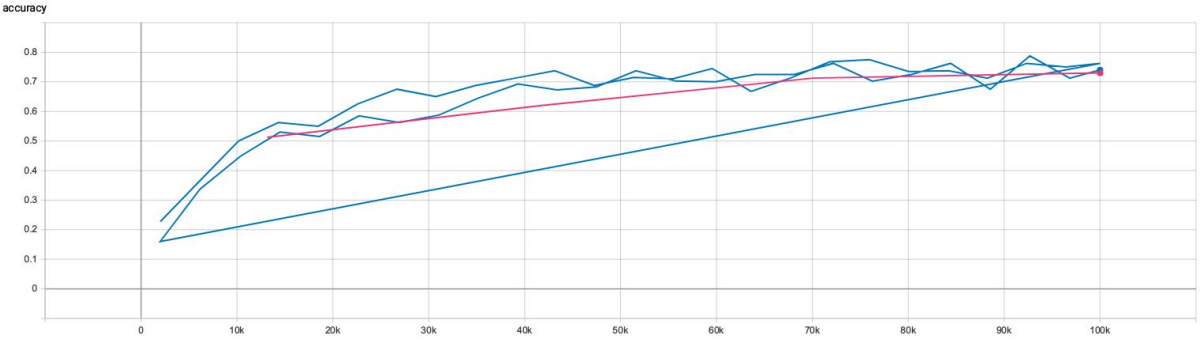
Estimator

For esitmator part, we customized an estimator in tensorflow. It has three parameters in both training and evaluating estimator, such as mode, tfrecord and batch_size.

stimator



5. Training & Result



Blue line is model with GPU, pink line is model withou GPU
Using CNN+LSTM+FC, after 100000 steps. The accuracy is 72.8%, and the running time is around 10hrs.

Using CNN+Cudnn LSTM+FC, in same configuration. it gives a 72.3% accuracy, and running time is 1 hr.

6. Evaluation

Using GPU accelerated model gives a huge performance improvement and decrease the running time dramatically.

7. Conclusion

CNN is a good model for image processing. And RNN is proven to be one of the best model for sequence data problem.

And as we can see. GPU acceleration really helps the performance and efficiency of model, gives a decent amount of improvement.