Capstone Project

Machine Learning Engineer Nanodegree

Sungju Kwon
December 31st, 2016

Definition

Project Overview

Single digit recognition problem like MNist is most common example for machine learning algorithm. Multi digit recognition is more challenging problem because. In this project, I'll implement convolutional neural network that recognize multi digits and upload that model to Android app based on Tensorflow Android demo app[1].
This project based on Goodfellow et al. (2014)[2] and Tensorflow CIFAR-10 tutorial[3]. First, I convert tensorflow cifar-10 example model accept SVHN dataset and do multi digit prediction. And then tuning hyperparameters for more accuracy.
Goodfellow's research shows that Convolutional network's accuracy can reach to over 95%. But my goal of this project is 90% accuracy and upload that model to Android app.

Problem Statement

Main goal is to create Android application that recognize multi digits. The tasks involved are the following:

1. Convert Tensorflow CIFAR-10 tutorial model to some custom model that can accept SVHN dataset as input.
2. Tuning hyperparameters.
3. Convert Tensorflow Android camera App can accept my model.
4. Show real world result.

Metrics

I define accuracy as whole digits is correct. According to Goodfellow et al. (2014), partial recognition has "no credit". I follow that policy. So, Accuracy is

$$accuracy = \frac{fully\ correct\ counts}{total\ counts}$$

[1] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android
[2] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In Proc. ICLR, 2014.
[3] https://www.tensorflow.org/versions/r0.11/tutorials/deep_cnn/index.html#cifar-10-model

I'll check Minibatch/validation/test accuracy for SVHN dataset. And get some real world prediction result by Android Application.

Analysis

Data Exploration

SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting.[4]

- 10 classes, 1 for each digit. Digit '1' has label 1, '9' has label 9 and '0' has label 10.
- 73257 digits for training, 26032 digits for testing, and 531131 additional, somewhat less difficult samples, to use as extra training data
- Comes in two formats:
    - Type 1. Original images with character level bounding boxes.
    - Type 2. MNIST-like 32-by-32 images centered around a single character (many of the images do contain some distractors at the sides).

SVHN provides images and mat files(Matlab file). But i'm not familiar deal with that file in python. So, i use fuel library for load that dataset[5]. It reads whole dataset(train, test, extra) and return one large dataset as result. Fuel's result dataset size is 248822.
For each data in fuel's result dataset has 6 column. Each column's data is like following:

1. Column 0 : image height.
2. Column 1 : labels.
3. Column 2 : image left boundary.
4. Column 3 : image top boundary.
5. Column 4 : image width.
6. Column 5 : image byte array.

Also, i use some real world data by Android application.

Exploratory Visualization

SVHN numbers.

[4] http://ufldl.stanford.edu/housenumbers/
[5] https://fuel.readthedocs.io/en/latest/api/dataset.html#module-fuel.datasets.svhn

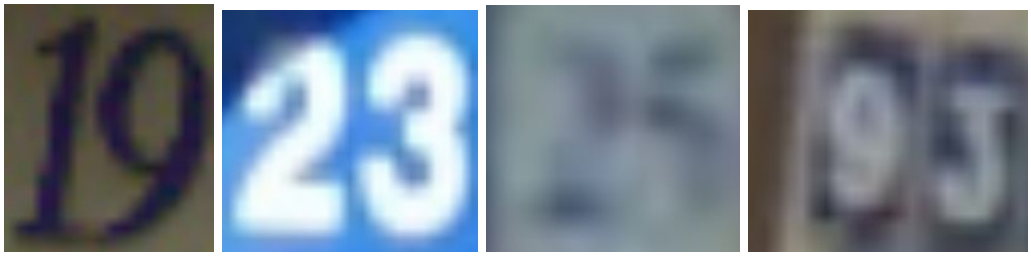
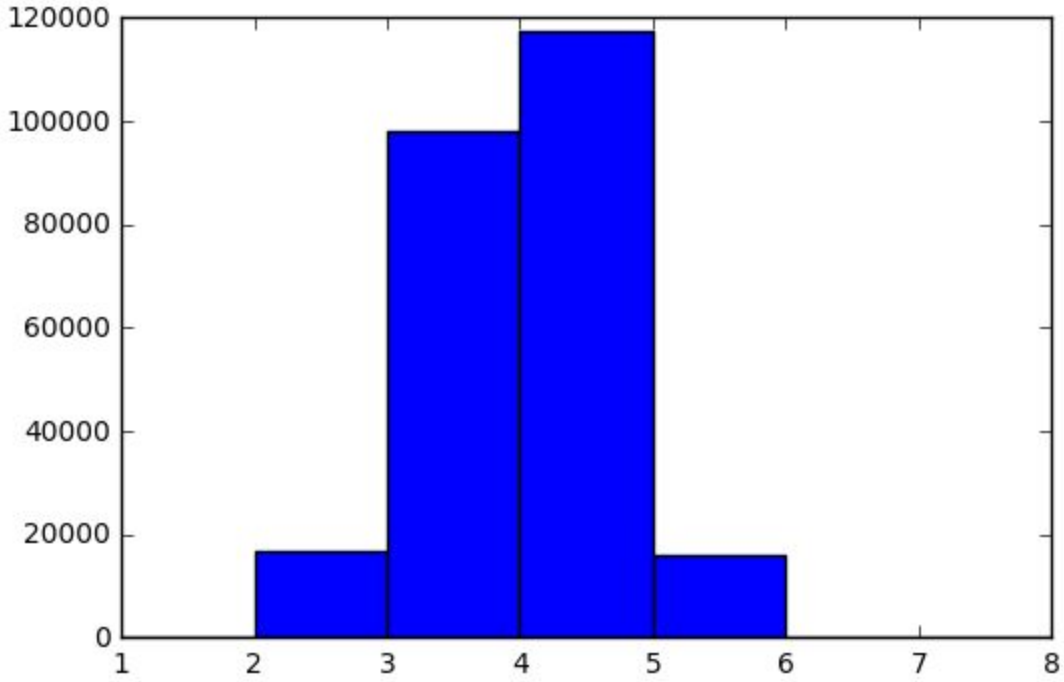
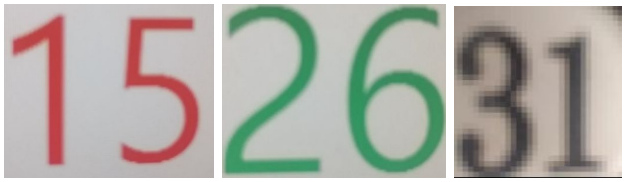


Fig. 1 SVHN digit count histogram. Most images digit length is 3 or 4. Some image has length 6. I'll ignore that images.

```
[     0  17005  98212 117561  15918    126      0] [1 2 3 4 5 6 7 8]
```



Real world numbers.



Algorithms and Techniques

This Project based on Tensorflow CIFAR-10 tutorial model. It use Convolutional Neural Network. And Neural Network has lots of hyperparameters. So we can try various parameter sets.

- Parameter
    - Batch size
    - Number of steps
    - Image Preprocessing param
        - Image Size
        - Image Margin
- Convolutional Neural Network.
    - Convolutional layer
        - Convolutional layer node count
        - Max pooling can be added
        - Normalization can be added
    - Fully connected layer.
        - fully connected layer node count
    - Optimizer
        - Optimizer algorithm
            - GradientDescentOptimizer
            - AdagradOptimizer
        - Learning rate
    - Weight decay

Most parameters are use Tensorflow CIFAR-10 tutorial model's value. Because i think it is fine tuned for image recognition. So, i experiment following parameters.

- Convolutional layer count : 2 to 8
- Each convolutional layer node count : various
- Each fully connected layer node count : various
- Optimizer : GradientDescentOptimizer, AdagradOptimizer
- Learning rate : various
- Image Size : 32, 53
- Image Margin : None, 10%, 30%
- Number of steps : 10001, 100001, 200001
- Batch size : 32, 64, 128

Benchmark

Convert Tensorflow CIFAR-10 tutorial model can accept SVHN input and can predict digits and convert optimizer to AdagradOptimizer, I gain Minibatch accuracy 79.7%. I regard that as base

model. Goodfellow's paper shows that Convolutional Neural Network can predict over 95% accuracy. But due to my experiment machine's performance, I decide my goal as 90% accuracy. It increase about 10% from base model(Tensorflow CIFAR-10 tutorial model).

Methodology

Data Preprocessing

Load SVHN dataset by fuel library. Per each image, i process following steps.

1. Calculate smallest boundary that contains all digits.
2. Add padding for preserve original image's width/height ratio.
    a. If padding area over image's boundary, i expand image's edge pixels.(by numpy's pad function's edge option[6])
3. Add margin(None, 10%, 30%).
4. Crop image by boundary.
5. Resize cropped image to 32x32 or 52x52.
6. Divide datasets to train/validation/test sets.

In Android App, Just resize cropped image to 32x32 or 52x52.

Implementation

First, I implement Convolutional Neural Network.

1. Copy Tensorflow CIFAR-10 tutorial model source code.
2. Load train/validation/test datasets.
3. Convert to custom model
    a. Add more convolutional layer(2 -> 8).
    b. Convert each layer(convolutional/fully-connected)'s node count as input parameter.
    c. Add normalizer on/off option as input parameter.
    d. Convert learning rate parameter as input parameter.
4. Defined parameters.
5. Run train/prediction.
6. Calculate accuracy. If accuracy is not high enough, return to step 4.
7. Save graph as file.

[6] https://docs.scipy.org/doc/numpy/reference/generated/numpy.pad.html

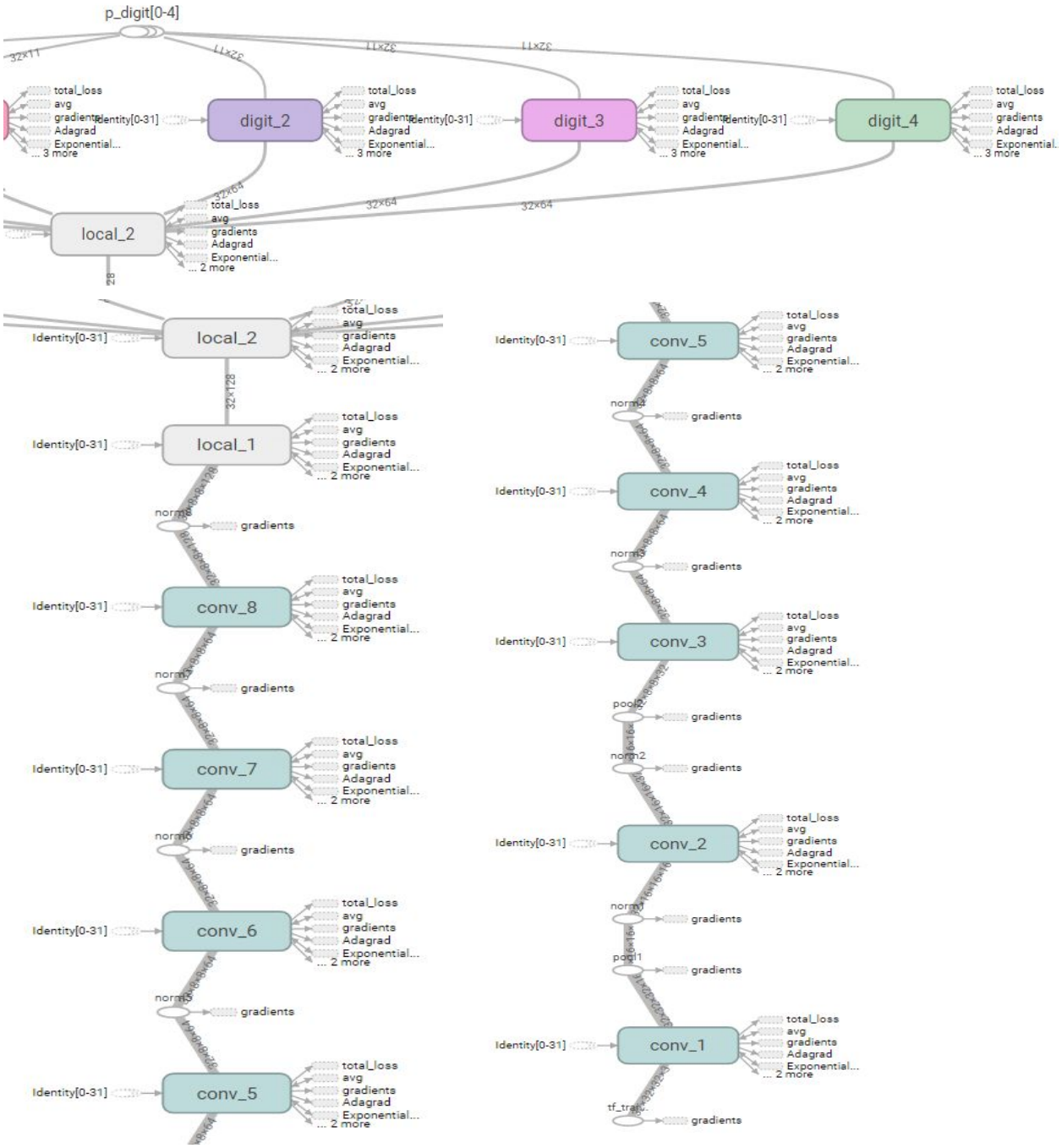


Fig. 2 Final convolutional neural network. Use Adagrad optimizer. The acronyms can be read as follows:

- Conv_x : x'th convolutional layer
- Local_x : x'th fully connected layer
- Digit_x : x'th digit recognize layer

Second, Android application implements by following steps:

1. Git clone Tensorflow source code from github.

2. Compile Tensorflow android camera demo application.
3. Change input & output logic for multi digit recognition.
4. Copy graph files to assets directory & load that graph.
5. Install to android smartphone & test in real world

Refinement

In Benchmark section, I noticed that base model's accuracy is 79.7%. I tune up some parameters following:

1. Convolutional layer count.(2 to 8. Final is 8)
2. Convolutional layer's node count.(8 to 192. Final is [16,32,64,64,64,64,64,128])
3. Fully-connected layer's node count(32 to 3xxx. Final is [128, 64]).

As a result, final model has an accuracy of 89.4%.

Results

Model Evaluation and Validation

Final model's structure & hyperparameters are like this:

- Convolutional layers filter size is 5x5.
- 8 convolutional layer.
- Node count per each layer is [16,32,64,64,64,64,64,128].
- Each convolutional layer does normalize. Normalize option is same as Tensoflow CIFAR-10 tutorial model(tf.nn.lrn(conv, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75)).
- First and second convolutional layer does max pooling with stride 2.
- First fully-connected layer's node count is 128.
- Second fully-connected layer's node count is 64.
- The weights of convolutional layers are initialized by truncated_normal_initializer with standard deviation of 0.05.
- The weights of fully connected layers are initialized by truncated_normal_initializer with standard deviation of 0.04.
- The weights of digit recognize layers are initialized by truncated_normal_initializer with standard deviation of 1/128.
- The training runs for 200,000 iterations.
- The mini batch size is 32.
- The learning rate is start from 0.01.

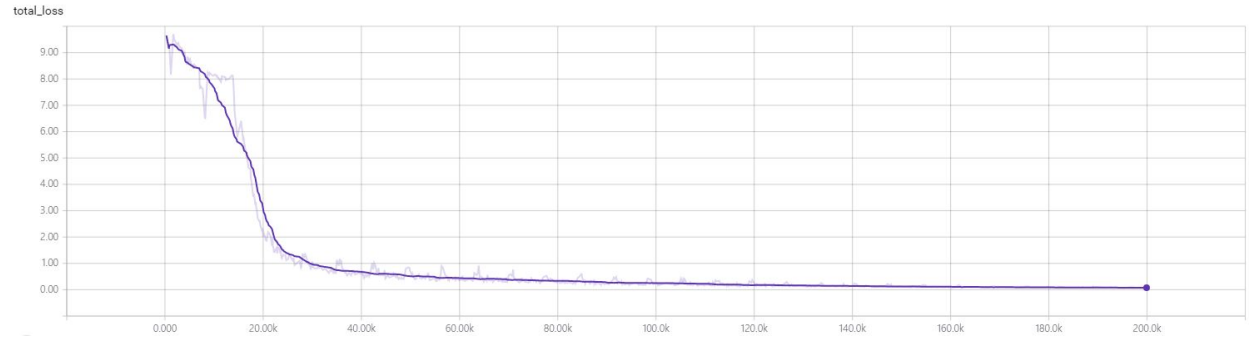


Fig. 3 total loss graph x axis is iteration.

Justification

Final model's accuracy is 89.4%. The base model that mentioned at Benchmark section (Converted Tensorflow CIFAR-10 tutorial model)'s accuray is 79.7%. The final model enhance about 10% accuracy. I think it reasonably good.

And in android application, it can read digits only digits image boundary is exactly same to image cropping section. After that, it can read digits well. But it is hard to matching image boundary to image cropping section.

Conclusion

(approximately 1 - 2 pages)

Free-Form Visualization

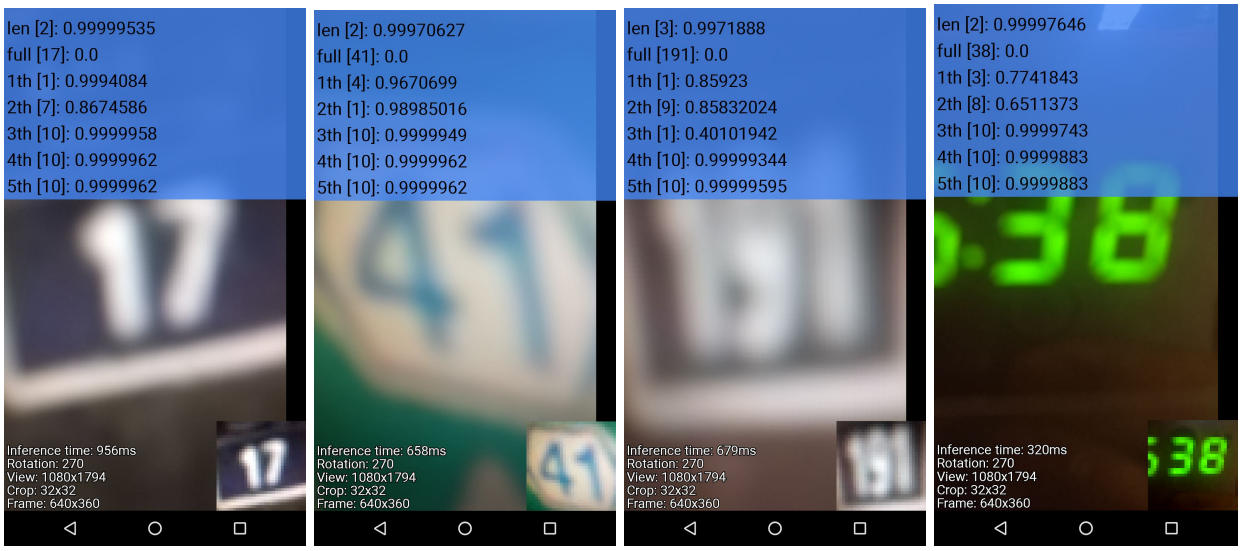


Fig. 4 Android application digits recognition results. Need image cropping boundary matching.

Reflection

This project's process can be summarized as following steps:

1. SVHN data preprocessing.
2. Develop initial model based on Tensorflow CIFAR-10 model and benchmark.
3. Train model and tuning hyperparameters.
4. Develop multi digit recognize Android application based on Tensorflow Android camera application demo.
5. Test Android application in real world.

To me, third step is really hard. Because at first i use Gradient Descent Optimizer. But in that model, total loss is can not converge. After 20~30 times trial, i almost give up. If i don't find AdagradOptimizer is working, i'll be give up this project.
After find that, I test other optimizer also. But in this model, AdagradOptimizer doing best performance.

In Android application, image boundary problem is arise. Because my model trained by boundary cropped images, when using Android application also need feeding cropped image. I think this feature can be added, but need more work.

Improvement

For convolutional network, there are many improvement points. Add drop-out, tune up normalize hyper parameters, learning rate decay, … I try some test for adding that feature to my model but result is not really good. So in this time, i can't add these features. But generally, that features can enhance convolutional network performance. So, If i make these model again, I'll try these features also.

As mentioned in above sections, Android application need detecting digits image boundary feature. It can be added by create more complex neural network.
It can be performed on same convolutional network or other convolutional network.