Machine Learning Engineer Nanodegree

Capstone Proposal

Hai Xiao

January 9th, 2017

Proposal

Use Deep Learning to Build a Traffic Sign Classifier

Domain Background

In the autonomous car industry, computer vision and deep learning have great application use cases in both providing accurate scene or context understanding to an intelligent agent (Self Driving Car) and augmentation or automation to the vehicle control, examples range from lane lines recognition and tracking, pedestrian classification, vehicle detection and tracking or traffic signs classification, etc.

With applications in demand and rapid development of GPU, image processing architecture and algorithm and deep learning infrastructure software have made great advancement in the recent years, popular ones range from Caffe, Torch, Keras, Theano to TensorFlow.

In this capstone project, I will develope one of typical traffic image classifiers using TensorFlow in addition to traditional machine learning library sklearn, with Python 2.7.

Problem Statement

In this project, I will use deep neural networks and specifically convolutional neural networks to classify traffic signs. Specifically I'll design and train a model using German Traffic Sign Dataset.

After I train the model, I will test it on the new traffic sign images from the test dataset provided to assess the quality of the model. Finally I will explore the performance of the model with traffic signs that I find on the web, and provide conclusions with justification.

Datasets and Inputs

In the project, both train & test dataset come from German Traffic Sign Dataset, it is also available from Udacity SDCND (Self-Driving Car Nanodegree) Traffic Sign Classifier project download.

***N.B.** Dataset provided from Udacity SDCND is pickled with Python 3 and incompatible with Python 2. So I have transformed and repickled them to Python 2 version and included directly in submission.*

Dataset briefs:

- There are 43 different classes (labels) of traffic signs in both training dataset ('train.pkl') and testing dataset ('test.pkl')
- Each data sample is an image, and all images are uniformed shape of (32, 32, 3) in RGB color space, where the 3 is color depth representing R, G and B, 32 is image width or height.
- There are 39209 training samples
- There are 12630 testing samples
- Sample counts across different traffic sign classes are not near identical
    - Training set: minimum sample counts class id = 0, count = 210
    - Training set: maximum sample counts class id = 2, count = 2250
    - Testing set: minimum sample counts class id = 0, count = 60
    - Testing set: maximum sample counts class id = 2, count = 750

- In the end I have found 5 images for extra validation of built model.
    - These 5 images are not uniformed in image sizes (that requires image resize)
    - They are same in RGB space but in PNG vs. JPEG format (that requires different normalization)

Solution Statement

This is a supervised learning problem, with input X as an image, output Y as a traffic sign class ID.

To resolve the problem, first we need to find proper representation of each image X, then to find proper classification model and algorithm.

Each image X can be represented as 32x32x3 (3072) bytes scalar. It is possible to build a classifier e.g. fully connected neural network, taking them all as direct input and iterate through several hidden layers to produce a class label.

But that sounds inefficient with lots unnecessary architecture overhead. This approach ignores any 'locality' existing in images applications most of the time, so it is a bad overkill incurred with extra processing overhead and less generalization. So we go with Convolutional Neural Networks (CNN) solution.

Now the detailed solution comes down to CNN architecture and its training. By design, we could experiment multiple CNN network architecture with various layers configuration to find an optimal architecture, but considering the facts that it is timing consuming even to train and tune one deep neural network, and the lack of feasible automation in searching through multiple architectures (it is far more complex in searching space than a GridSearchCV on SVM for example), we normally take trial and error empirical approach to find a good solution.

Benchmark Model

I have not collected project benchmark from Udacity student statistics, but here we have the final competition benchmark (on test dataset) from GTSRB 2011 (http://benchmark.ini.rub.de/):

Rank	Team	Representative	Method	Correct recognition rate
1	IDSIA	Dan Ciresan	Committee of CNNs	99.46 %
2	INI		Human Performance	98.84 %
3	sermanet	Pierre Sermanet	Multi-Scale CNNs	98.31 %
4	CAOR	Fatin Zaklouta	Random Forests	96.14 %

Ideally I expect my result get close enough to those results, but since I am using only generic CPU vs. GPU with a non-extensive deep (for training time) network architecture, so I expect ~95.0% test accuracy a reasonable target to achieve, also considering the Human Performance at 98.84% from above table.

Evaluation Metrics

We will follow standard machine learning validation metrics to evaluate the model at various stages. Specifically validation accuracies are used during training, and test accuracy is used as final report.

By design, we could run cross validation with limited samples, but given the time consuming in training the deep neuron network, I will skip cross validations (e.g. generalize model further w/. KFold) for the time being, and may revisit this if a final testing score (Loss) is not good enough.

Since this is a classification, cross_entropy is used to derive Loss function to drive the training optimizer. And prediction accuracy is used as evaluation metrics.

Project Design

In this final section, summarize a theoretical workflow for approaching a solution given the problem. Provide thorough discussion for what strategies you may consider employing, what analysis of the data might be required before being used, or which algorithms will be considered for your implementation. The workflow and discussion that you provide should align with the qualities of the previous sections. Additionally, you are encouraged to include small visualizations, pseudocode, or diagrams to aid in describing the project design, but it is not required. The discussion should clearly outline your intended workflow of the capstone project.

Data Preprocessing

- Data exploration of training and testing image data set
- Normalization (zero-Mean) input image to network input layer
- Implement One-hot encoding to all class labels of training, validation and testing set
- Training, validation set split (test set as preserved)

Implementation

- Construct classification DNN network with convolutional and fully connected layers.
- Execute TensorFlow Graph defined
- Training and Model Validation with Learning Curve
- Measure the testing performance (N.B. not involve testing set in network training)

Refinement

- Use epoch-by-epoch training approach for continuous model refinement
- Measure the model with 5 new traffic sign images that may not come from German
- Justify the result on 5 new images and the generalization of prior trained model
- Reflection and discussions for future direction