Neural Network Training with Tensorflow

KTH Royal Institute of Technology
Long Zhang (longz@kth.se)

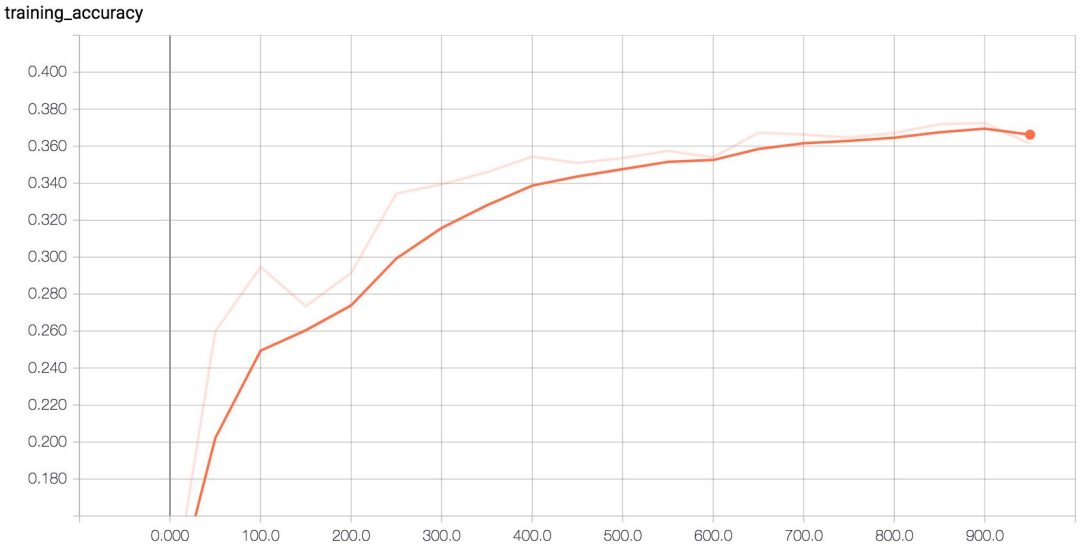
Abstract

- The requirements of this assignment can be read on this page
- Our goal is using TensorFlow to explore neural network and design 3 models to do the classification tasks on CIFAR-10 dataset

- Short introduction to the 3 models:
  - Model 1: only 1-layer network trained with cross-entropy loss, mainly testing different parameters' influence on results, test accuracy is around 0.35
  - Model 2: enhancing Model 1, adding an additional hidden layer and using momentum optimizer, test accuracy 0.3927
  - Model 3: implementation of a convolutional network, test accuracy 0.558

- The experiments codes with sufficient comments can be seen on this GitHub repo

Training the first network



- Learning rate: 0.01
- Batch size: 200
- Iteration: 1000

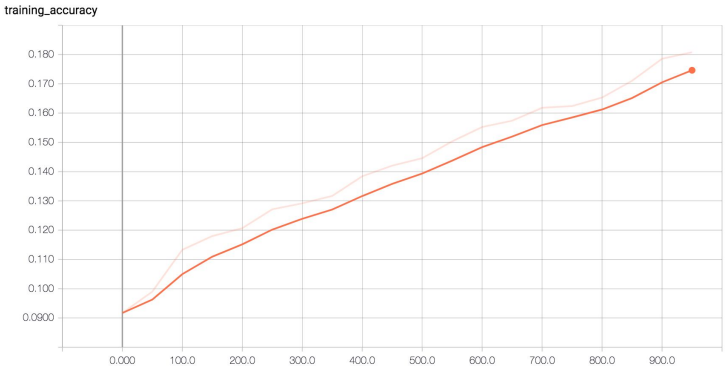
Test accuracy: 0.3737

The basic source codes have been provided, so that we can easily get to understand the Tensorflow class and methods. Above are the basic parameters and outcomes. Next we will change different parameters' values and discuss why they influence the results like that.

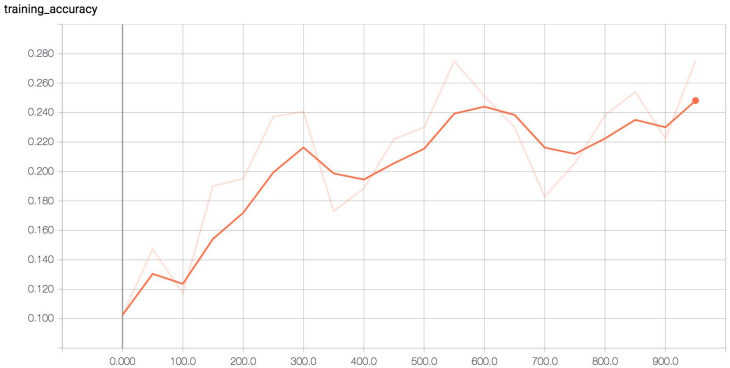
Training the first network

Batch size: 200. Iteration: 1000

Learning rate: 0.0001, test accuracy: 0.1858



Learning rate: 0.1, test accuracy: 0.2024

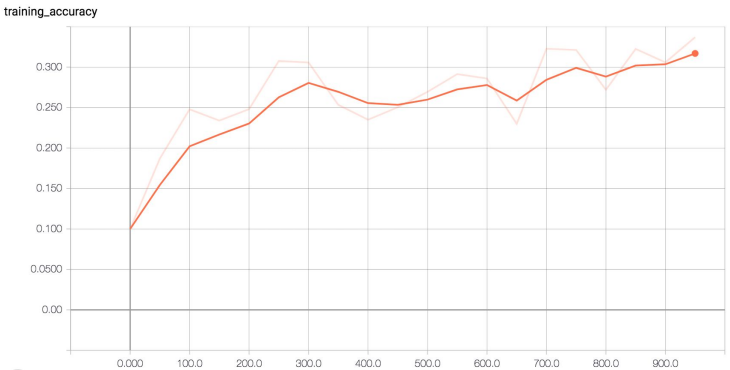


This time, we only change the key parameter “learning rate” into 0.0001 (extremely small) and 0.1 (extremely big).
You can see from the charts that the line is much smoother when learning rate is 0.0001, this is because learning rate determines how fast to change the weights in training. If it’s too big, the changing will have more oscillations, and too small will make it really slow to get the optimization results.

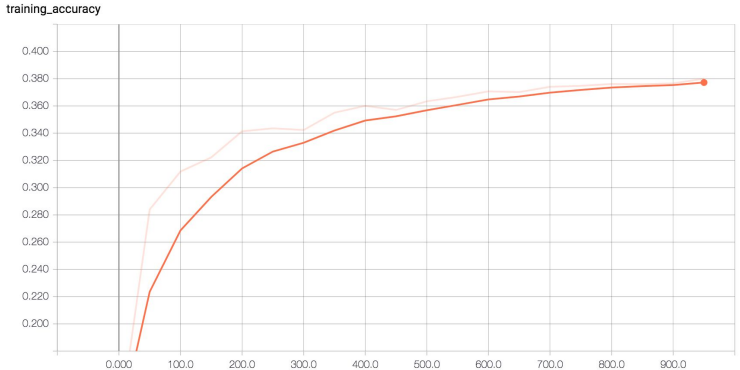
Training the first network

Learning rate: 0.01. Iteration: 1000

Batch size: 20, test accuracy: 0.3198

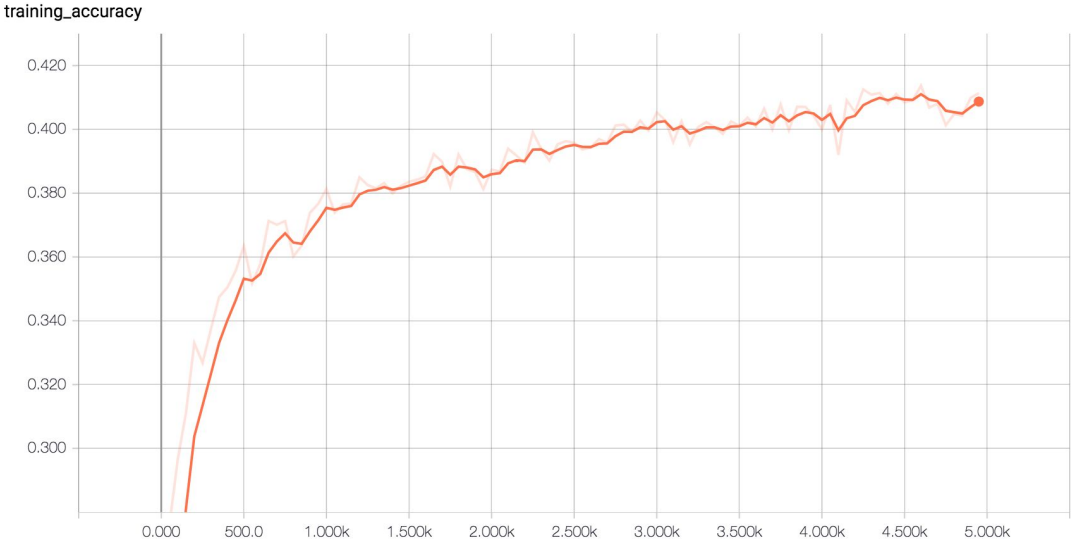


Batch size: 1000, test accuracy: 0.3793



Comparing to the previous experiments, it seems that batch size has less influence on the test accuracy. However, there is still some difference. If we choose a very small batch size, it's better for the randomization, but sometimes it may cause the loss function to oscillate without convergence. If the batch size is too big, besides for consuming more resource, it will also cause "sharp minima". (not be seen from this example)

Training the first network

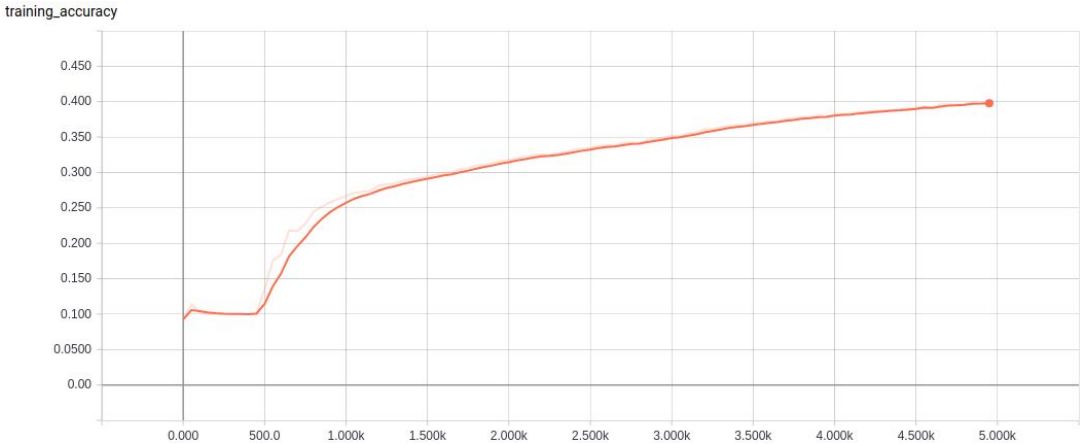


- Learning rate: 0.01
- Batch size: 200
- Iteration: 5000

Test accuracy: 0.4004

Now we only increase the iteration times, from 1000 to 5000. You can see that the accuracy only increased 2 percent from iteration 1000 to 5000, which means that for this model and hyperparameters, the accuracy nearly saturate at 0.40

A 2-layer network with momentum training



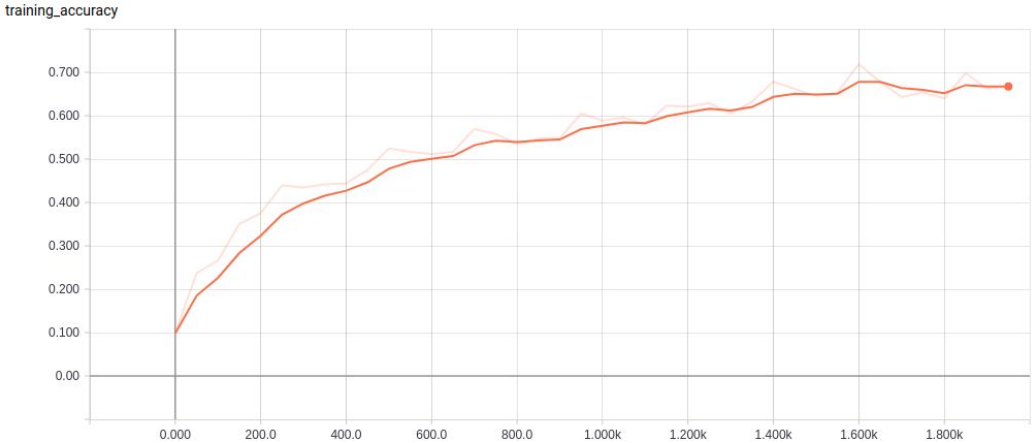
- Learning rate: 0.001
- Batch size: 500
- Iteration: 5000

Test accuracy: 0.3927

Implementation of a convolutional network

- This network model contains the following layers:
  - Layer 1: convolutional network + relu + lrn + max_pool_2x2, patch 5x5x3, output 16x16x64
  - Layer 2: convolutional network + relu + lrn + max_pool_2x2, patch 5x5x64, output 8x8x128
  - Layer 3: fully connected network + relu, output 384
  - Layer 4: fully connected network + relu, output 192
  - Layer 5: prediction
  - cross_entry: softmax_cross_entropy_with_logits + reduce_mean
  - optimizer: AdamOptimizer, learning_rate 0.01

Implementation of a convolutional network



- Learning rate: 0.01
- Batch size: 200
- Iteration: 2000

Test accuracy: 0.558