TEXAS A&M UNIVERSITY

EE649 Project Report

GROUP MEMBER
Zheyong Sun, UIN:228000667
Zeyu Wang, UIN:228001608

December 12, 2018

1 PROJECT TYPE GOAL IN PROJECT PLAN

Our project type is application. We want to use Convolution Neural Network, specifically the VGG16 Net algorithm, to do image classification based on Tensorflow.
Our goal is to train a model that could reach 85% to 90% accuracy rate on test set.

2 INTRODUCTION OF DNN AND CNN

2.1 DEEP NEURAL NETWORK

Deep Learning, as a branch of Machine Learning, has been so hot since ALPHA GO beat Yi Sedol in go in 2016. The concept of "Deep Learning" actually comes from Neural Network, which is an algorithm combining multiply linear or nonlinear computing layers(operations) to study mapping raw date to certain high-level domain so that computer could make decision by themselves in the future.

Scientists first came up with the idea of so called Artificial Neural Network based on the study of the brain of animals in 1960s. In 1980s, the appearance of algorithm of back-propagation made Neural Network become popular at the first time since it solved the problem of how to train each neuron in the network. But due to the bottleneck of computing ability of hardware at that time and the problem of overfitting, researchers stopped their steps from going deeper in the road of Neural Network.

But everything changed in 2006, Geoffery E.Hinton, one of the contributors of back-propagation algorithm, published a paper about "Deep Brief Network"[1], which proved that the model of Neural Network with deep layers could be trained correctly and work much better than other previous way. Since then, Neural Network started to develop in the name of "Deep Learning". The "Deep" here refers to the large number of layers in the Neural Network.

Up to now, the mathematics principle of Neural Network is still a blank sheet and many researchers suffer from that. Neural Network just likes a black box. It seems that only if we give the data and adjust the parameters correctly, we could get the expected result. Sometimes, the way of training network is just like a kind of "art" rather than science, because we have to develop based on experience most of the time.

It is such a funny and mystery algorithm that beats many other excellent traditional machine learning algorithms in almost every area. End-to-end manner may be the key weapon of "Deep Learning". Comparing with traditional machine learning algorithms, Neural Network doesn't rely on so called Feature Engineering, kinds of process of transforming raw data into particular format that represents the underlying connection to the algorithm in a better way, resulting in improving of accuracy. Neural Network learns the features from the input data directly instead of hand-craft feature designed by human. In traditional way, we would like to divide the whole problem into several sub-problems and cause a serious problem: the best solution for one sub-problem may not be the best one for the final problem. But we will never worry about that in Deep Neural Network. We only need to care about how to optimal the process of transforming raw data representation into high-level semantic representation.

In a word, Deep Learning is a kind of methodology, which try to build complex and deep hierarchical model containing many nonlinear computing layers for the raw data and study the rules from that.

2.2 Convolution Neural Network

DNN, as we said, is very powerful in process data, but we have a much better way to process image, the Convolution Neural Network(CNN). In DNN, we have to compress the image to a 1-D vector and lose many information. But in CNN, we use convolution operators to extract a square part of image once a time for computing, which could keep the relationship of the pixel with other nearby pixels.

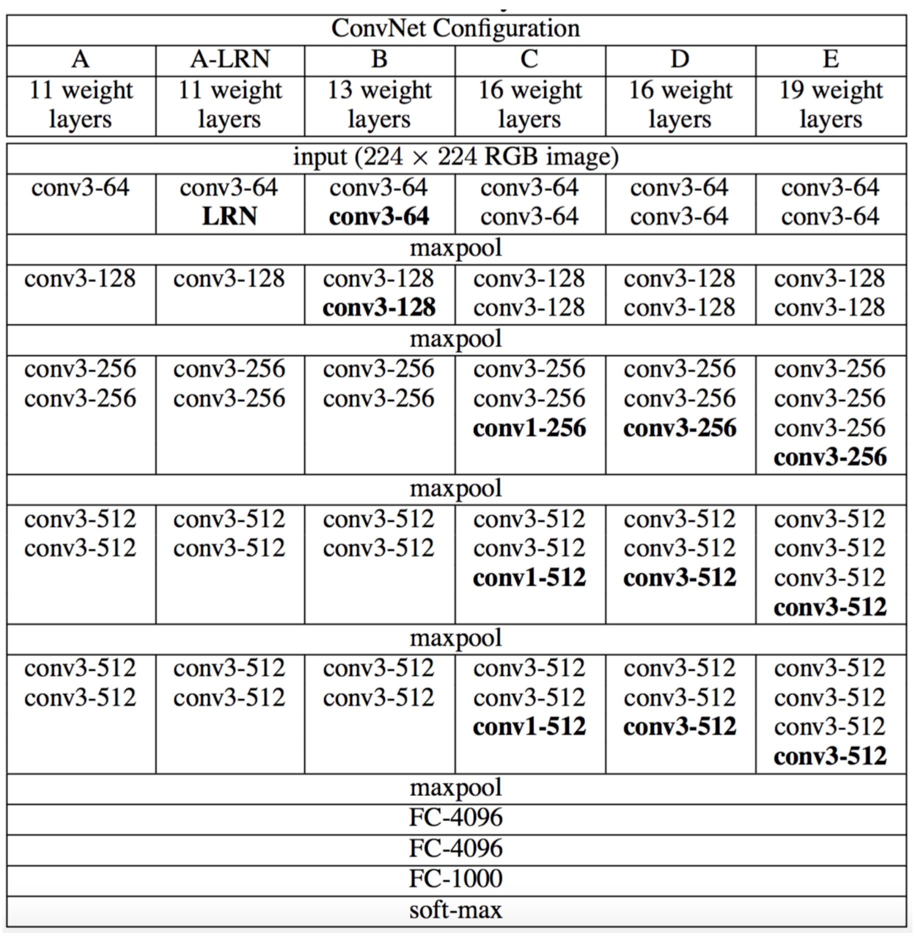
CNN was inspired by the research of animals' brain in 1960s to 1980s, especially the concept of 'receptive field'. Professor LeCun first implemented CNN on hand-writing digits in 1998 and got a great experiments result. But CNN was not so popular until 2012, when a CNN algorithm called AlexNet won the game of ImageNet and gained a high score, which is more than about 12% over the runner-up.
CNN is composed of several kinds of layer and each of them has its only duty. Convolution layer executes the convolution operation. Pooling layer compress the data in height and width. Softmax layer converts the data from matrix to vector and computes the probability for each class. Some researchers use the batch-normalization layer to optimize the speed of SGD and improve the final result.

3 Experiment on VGG16

In this project, we implemented the algorithm of VGG16[2], which was published in 2015 and got a great score at the game of ImageNet at that year. Actually, VGG is the name of a lab in University of Oxford and they named a series of algorithms with the name of their lab. VGG16 is one of them. 16 here just represents the number of weighted layer(Conv and softmax). The following image shows the structure of VGG and we used the structure with the name of D.

In this project, we used the cifar-10 database, which are commonly used for training machine learning and computer vision algorithms to recognize objects. We optimized the reading algorithm and finally visualized the results.

The package we used for this project is TensorFlow, which works well with GPU model on Win10. The dataset we use for training and testing is Cifar10 image set, which contains 60000 images with 10 different labels. Each one is a 32*32*3 colorful image. There are 50000 images in training set and 10000 images in test set. To improve the ability of robust of the network, the input image is distorted.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.1: Structure of VGG16[2]

We choose Relu to replace the Sigmoid to be the activation function since many papers show that the performance of Rule is better since it could solve the problem of gradient vanishing. Maxpooling layer will reduce half the size of height and width of current input data, so we remove the last maxpooling layer to avoid cut 1 to 0 since the size of input data is 32*32. The first 2 convolution layers will modify the input 32*32*3 image to a 32*32*64 matrix and the first maxpooling layer will reduce it to 16*16*64. Maxpooling not only compress the data but extract the representative information from a certain region. The second group of convolution layers and maxpooling layer changes input of 16*16*64 to 8*8*128 and the third group changes it to 4*4*256 and so on. At last Convolution layer, the size of input data will become 1*1*512. So there is no way to use maxpooling anymore. Then we use fully connected layer to convert 1*1*512 matrix to a 1-D vector and the output will be a vector with size of 4096. After the process of three fully connected layer, the final output will be a vector with size of 10, which means that there are 10 labels in total.

We use cross-entropy as the loss function, which is better then simple square loss as well. The formula of cross-entropy is $C = -(1/n)\sum_x yLnx + (1-y)Ln(1-x)$. It is non-negative at first. If the output is close to the target, the value of cross-entropy will be close to zeros and that's our desired result. What's more, the most significant advantage of cross-entropy is its derivation. The bigger the difference, the bigger the derivation(gradient) is, which accelerates the speed of training.

Once we build the whole network according to the paper[2], we need to think about the hyperplane of the parameters. There are so many parameters we need to consider to get a convergent model, such as the parameters of initialization, the number input data in each minibatch, the size of step of study and the how many step we need to make to get a good result. We combined the experience from other CNN user and the practice. We use truncated normal distribution to initialize the weight of each element in data matrix of each layer. The function of truncated normal distribution is sample a random value from the norm distribution but let the value larger then zero. Small value closing to zeros helps network to converge. A small bias closing to zero helps the network to converge as well. The size of each batch determine how many samples we need to consider at the same time, it can't be too small or too large otherwise it may converges to a wrong direction or cost too much time. At last we choose 32 to be the number of samples in each batch. The size of each step is extremely important since it decides whether we could get a out-performance model or not. I tried from a pretty small value about 1e-5 to 1e-3 and found that 1e-4 could bring a stable trend of the decreasing in loss. Finally, to get a high accuracy as we set in plan, we train the model with steps of 140000.
We use TensorBoard to record the changing of loss in training and show the corresponding image below. You could find that the trend of loss is decreasing with the increasing of step of training. Finally, we test our model on test set and gain a precision about 85%.

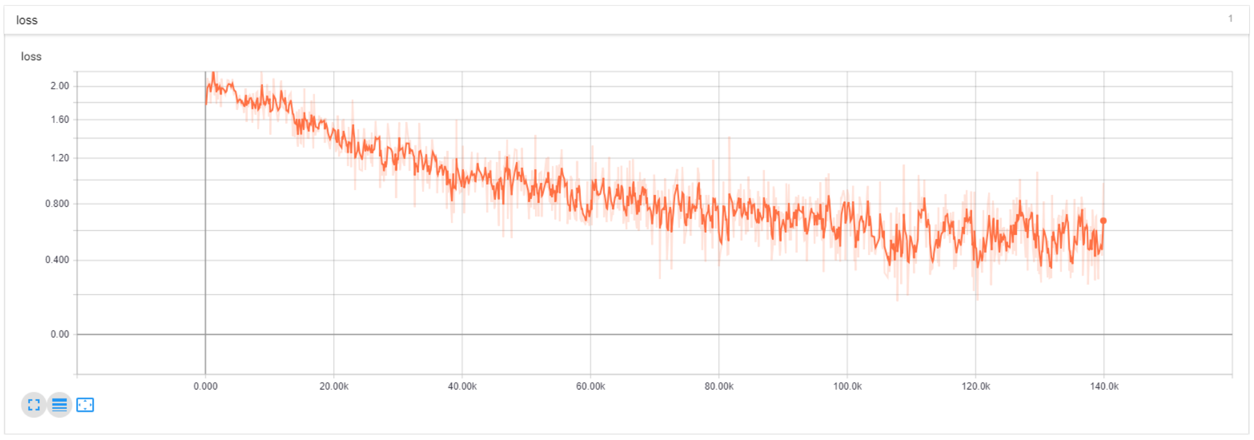
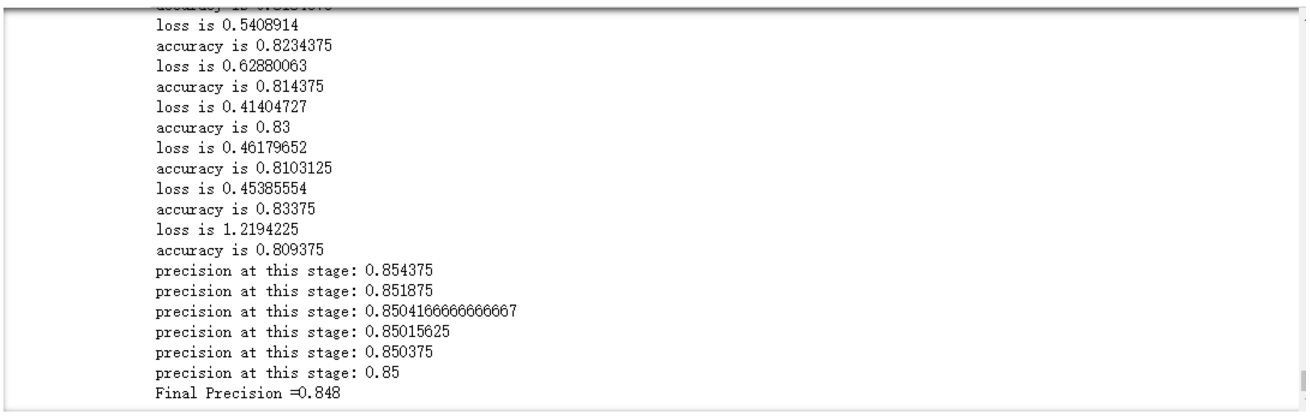


Figure 3.2: The changing of loss in training



```
loss is 0.5408914
accuracy is 0.8234375
loss is 0.62880063
accuracy is 0.814375
loss is 0.41404727
accuracy is 0.83
loss is 0.46179652
accuracy is 0.8103125
loss is 0.45385554
accuracy is 0.83375
loss is 1.2194225
accuracy is 0.809375
precision at this stage: 0.854375
precision at this stage: 0.851875
precision at this stage: 0.8504166666666667
precision at this stage: 0.85015625
precision at this stage: 0.850375
precision at this stage: 0.85
Final Precision =0.848
```

Figure 3.3: printed result

Our code has been upload to Github as follow: https://github.com/liekejiang/EE649-VGG16

REFERENCES

[1] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.