Implementation of ReLayNet

Zhenyao Cai

Abstract: Roy et al. proposed an efficient deep learning approach to solve the retinal layer segmentation of OCT images problem. In the report, I discuss my python implementation of the proposed CNN architecture and loss functions discussed in the paper, the dataset and input values I used for experiment, and other potential applications of this approach.

1. Introduction

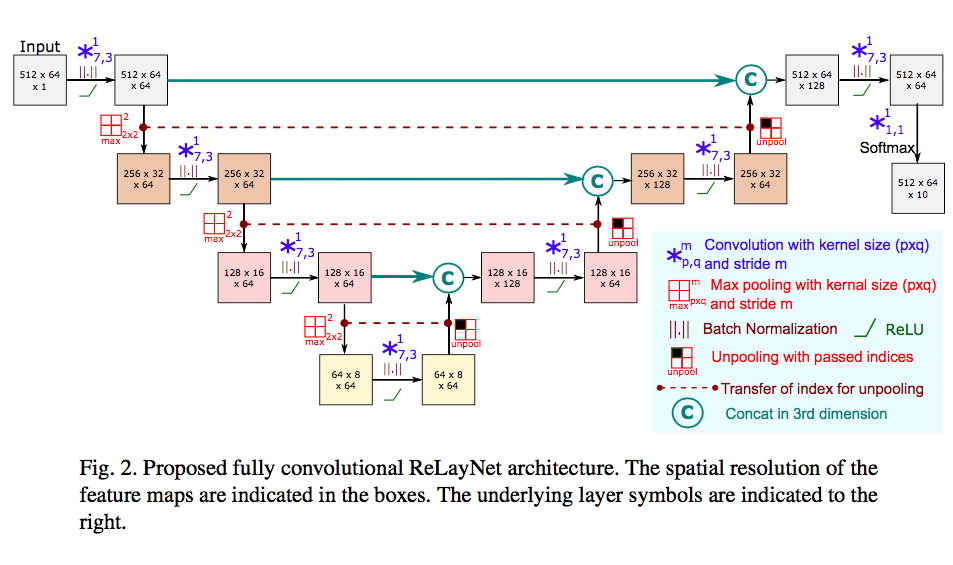
Optical Coherence Tomography (OCT) is used for non-invasive diagnosis of diabetic macular edema assessing the retinal layers. In the paper by Roy et al., they proposed a deep-learning architecture named ReLayNet, which is trained to segment the retinal layers and fluid of macular OCT. The segmentation of multiple retinal layers in eye OCT images is necessary for examining the situation of diabetic patients in order to prevent the occurrence of blindness.

* 1. Proposed Methodology in the paper

The problem is to segment each pixel location x=(r,c) to particular label l in the label space L={1,..,K} for K classes, where K in 2the paper is 10. Another way to say is to classify the pixels in OCT images to 10 different classes.

Network architecture

The CNN architecture consists of 8 convolution Layers. According to the paper, the first layer has a convolution filter with kernel size (7x3) and stride 1, a relu layer, a batch normalization, and a max pool with kernel size 2x2 and stride 2; the second layer and the third layer are the same as the first layer, except the size of the input image. The fourth, fifth, sixth layers are the same as the previous ones, except that the images are unpooled with passed indices. The seventh layer doesn’t have pooling layer, and the eighth layer is a classifier layer which has only a convolution with kernel size (1x1) and stride 1.



Loss function

Cross-entropy provides a probabilistic similarity between the actual label and the predicted value at the current state of the network. Another thing is a penalty function, which penalizes the pixels at the boundary of each layer.

1. Implementation
   1. Dataset

The dataset is a Duke SD-OCT publicly available dataset for DME patients. <http://people.duke.edu/~sf59/Chiu_BOE_2014_dataset.htm>. The data is originally stored in .mat file, so I wrote code to save the image files in order to import them in python.

* 1. Functions and Modules

Three important modules used in the implementation: cv2 is in opencv package, which is used to read images in grayscale; numpy is used to store and process the image information as a matrix; tensorflow is an open library for machine learning, the training steps and optimization are automatically done by using tensorflow built in functions, so it is efficient to build CNN model by using tensorflow.

In functions.py, there are four important functions. The first is cnn\_model function, which is used to build the CNN model by setting up 8 convolutional layers. This is not so complicated, because convolving images with specified kernel size and stride, adding bias, adding relu layer and adding pooling layer are all tensorflow built in functions (eg. tf.conv2d, tf.nn,bias\_add, tf.nn,relu, tf.nn.max\_pool, and tf.nn.lrn).

The second is loss\_function, which is used to calculate the loss of the image in training. The loss is defined by two things indicated in the paper, one is the cross entropy, the other is the penalty of the boundary. The former one is a builtin function of tensorflow, the latter one needs some math to do it correctly.

The last one is the test accuracy rate function, which is done by comparing the prediction image with the target label image by using tf.equal(prediction, target).

In train.py, there are mainly three parts. First is to load data from the dataset, second is to build the CNN model by calling functions in functions.py, and last is to feed the data we loaded into the model in order to train it.

* 1. Experiment

I experiment the program with 100 images in the dataset. The maximum iteration times is 3000, each time feed in 10 test images, and after 50 iterations, test the loss and accuracy rate. For the training step, I set the initial learning rate as 0.002, the decay steps as 25, and the decay rate as 0.9.

2.2 Challenges

Building a deep neural network is extremely slow, so I could not use my laptop to run the code, but using GPU. Packages necessary for running the program are tensorflow1.6, cuda9.0, which is used to install tensorflow, and cuddn7.1. Though using GPU, it still takes around 1 hour to complete training the model.

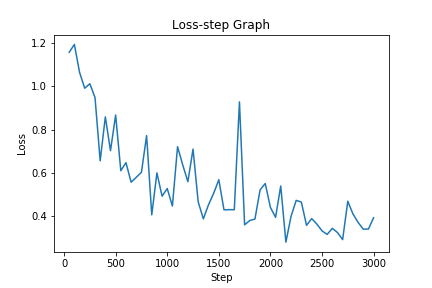
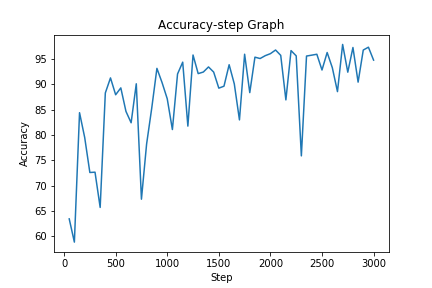
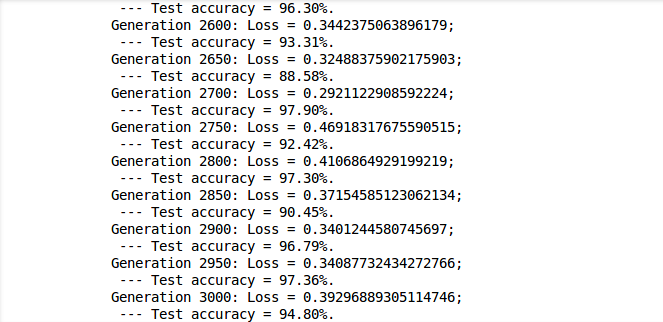
Understanding tensorflow library is also a challenge for me. It took me a while to understand how to build CNN layers by using tensorflow, and how to feed in numpy array data in the CNN model.

Besides, understanding the loss functions is also very challenging, but fortunately, tensorflow has built in cross-entropy function.

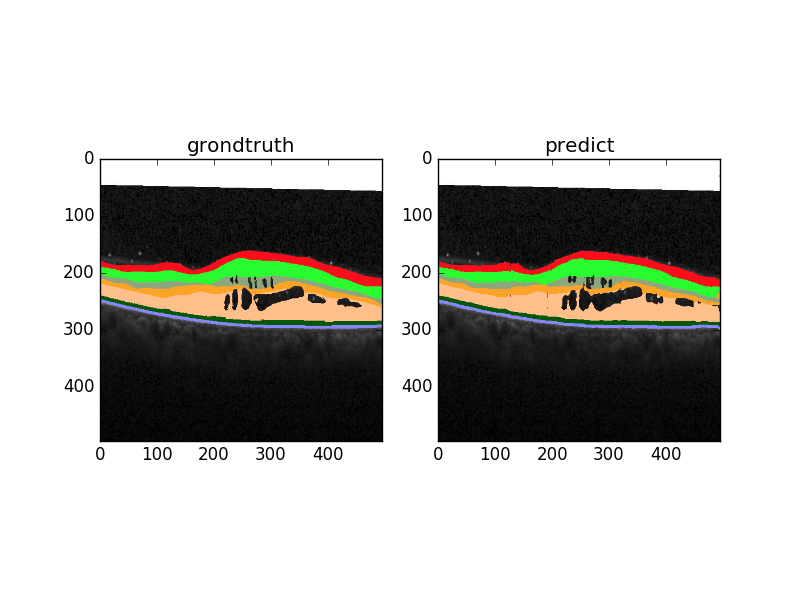
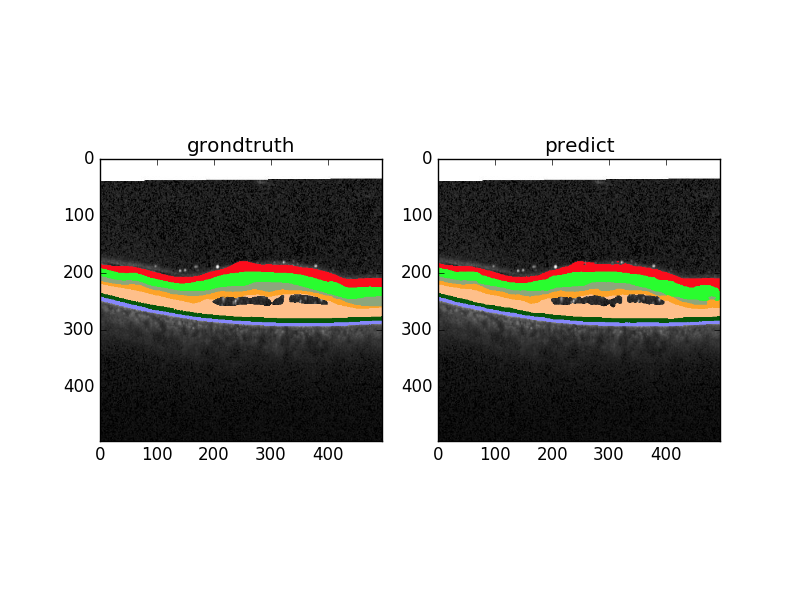
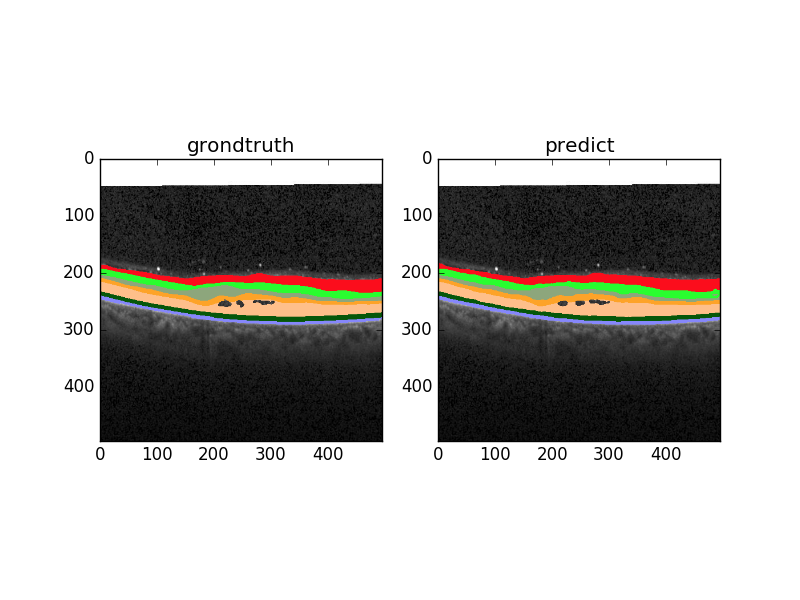
Lastly, the issue I haven’t solved is to import trained model and test it, so I only have seperate functions.py and train.py. However, I merged them together, and used the trained model right away in segment.py, and produced the testing results.

1. Result

The result shows that even though there are some fluctuations, the loss decreases gradually, and the accuracy rate increases gradually. After 3000 training iterations, the accuracy rate approaches 100%.



The coloring of the predicted layer after training also shows the accuracy of the prediction comparing to the target segmentation.



1. Future Work and Conclusion

This convolution neural network might be only suitable in this particular problem, because it only has 8 layers. It may be applied to other simple image segmentation problems that have less than 10 classes of pixels (10 objects). More complicated CNN solvable problems, such as face detection or object detection might need up to hundreds of layers. Thus, this simple architecture, though not hard to implement, has very limited usage.

Works Cited

Roy, Abhijit Guha, et al. “ReLayNet: Retinal Layer and Fluid Segmentation of Macular Optical Coherence Tomography Using Fully Convolutional Networks.” *Biomedical Optics Express*, vol. 8, no. 8, 2017, p. 3627., doi:10.1364/boe.8.003627.