Convolutional Neural Network - cs612 project

Jiho Noh (jiho@cs.uky.edu)
Department of Computer Science, University of Kentucky

1 Overview

Convolutional Neural Network (CNN) is a type of feed-forward neural network which is inspired by and simulating multiple layers of biological perceptrons. Its strength stands out in many sub areas of artificial intelligence such as image classification, object recognition and natural language processing.

Throughout this course, the fundamental concepts of neural network are studied and various applications of CNN is explored.

1.1 Project Goals

This project is to practically apply the neural network mechanisms on real datasets. There are many neural network frameworks available, such as Caffe, Torch, Tensorflow, and more general purpose libraries. For this project, Tensorflow by Google is used to build a network and trained to achieve reasonably accurate predictions.

1.2 Deep Learning Frameworks

A good carpenter must know what tools he can use and its usages. There are many tools for deep learning tasks. Arguably, the most popular frameworks are developed by a few big players in the deep learning field.

Theano and Pylearn2 are developed by Yoshua Bengio's group at the University of Montreal. Theano embraces computation graphs and symbolic computations, so it is relatively more flexible to design complex graph models like RNN (Recurrent Neural Network). It is know by many users that Theano is much slower to compile large models.

Caffe is developed from U.C. Berkeley. It is most widely used for convolutional finetune pretrained models. Pretrained models can be obtained from "Model Zoo" which maintains most of the state-of-the-art models such as AlexNet, VGG, GoogLeNet, ResNet, and others.

Torch is from NYU, mostly developed by Yan LeCun's lab, although LeCun's contribution is minimal. Torch is written in Lua, a high level scripting language. Becuause of this unfamilar language, it requires an additional learning phase to start using this library. Due to the same reason, it is much easier to write your own layer and run on GPUs. Both Caffe and Torch has downsides of implementing RNN.

Tensorflow is recently release by Google. It stands out by its capability of using multiple GPUs and multi-node training. Also it provides a visual interface (TensorBoard) to see the structure of the network and also check the status of training process while running.

In this project, Tensorflow will be used to train a minimal network and learn how to practically use a deep learning library.

1.3 Dataset

SAT-4 and SAT-6 airborne datasets consist of sampled landscape images in 28 by 28 pixels and 4 channels – red, green, blue and near infrared. The satellite images cover the whole state of California, which are categorized into 4 and 6 classes as below:

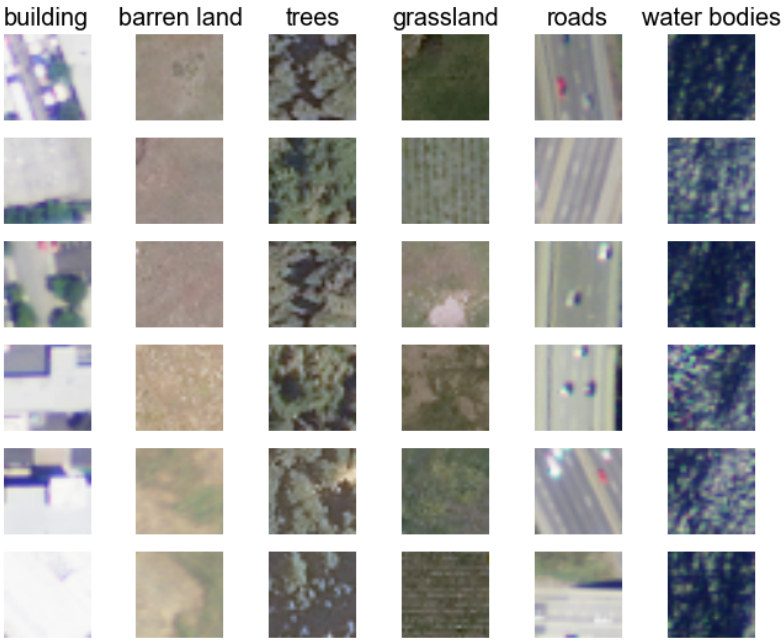
SAT-4

- train_x: $28 \times 28 \times 4 \times 400{,}0000$
- train_x: $400{,}000 \times 4$
- test_x: $28 \times 28 \times 4 \times 100{,}0000$
- test_x: $100{,}000 \times 4$
- classes barren land, trees, grassland, uncategorized

SAT-6

- train_x: $28 \times 28 \times 4 \times 324{,}0000$
- train_x: $324{,}000 \times 6$
- test_x: $28 \times 28 \times 4 \times 81{,}0000$
- test_x: $81{,}000 \times 6$
- classes building, barren land, trees, grassland, roads, water bodies

Figure 1: SAT-6 images categorized by labels



2 Model

The model design and implementation is mostly based on Tensorflow Tutorials, that use popular datasets such as MNIST handwritten digit classification, and CIFAR-10 image classification.

Generally, training a neural network consists of three modules: 1) load a dataset, 2) predict by the given weights and biases of the layers (forward pass), 3) and optimize the variables to minimize loss (back-propagation).

2.1 Inputs

The DEEPSAT dataset images are in 4 bands – red, green, blue, and NIR (near infrared). In this project, only green channel is used as a single channel image by assuming that all four channels are correlated. This will definitely impair the accuracy in some degree, but at the same time it will reduce the number of operations significantly.

Reading data from disk into memory requires significant amount of process, hence it requires an efficient way of loading data. In particular, Tensorflow reads data in 16 separate threads. However in this project, the original dataset is in Matlab .mat file, it is loaded via Python Scipy loadmat method and divided into batches.

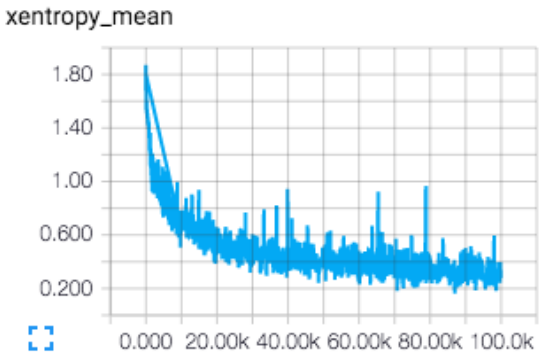
2.2 Prediction

Two different inference models have been tested: 1) dc1-dc2-softmax 2) conv1-pool1-norm-conv2-pool2-norm2-dc3-dc4-softmax. Deeper networks like VGG or Residual network has much more layers, and it would yield better results. However, our two-layers and four-layers network can also generate meaningful results within relatively short running time (< 30 min).

2.3 Training

Once the computaion graph is designed, Tensorflow can start training. It will automatically finds the gradients of the cost by differentiating operations with respect to each of the variables. In each step, it will apply the gradient descent updates trying to minimize the total loss.

Figure 2: Cross Entrophy Loss Function



3 Evaluation and Results

Once the logits are returned, it will be compared to the ground truth. If the prediction matches the true label of the image, it will be considered as a correct prediction. Then we take the mean of correct predictions among the test set, and returns it as the precision value.

A simple two-layers FCN can reach up to 93% accuracy within 30 minutes of training. [Appendix A]

4 Future Works

For this project, only one channel out of four (red, green, blue and NIR) was used as the input data. Presumably color information is important in natural images because it is not uniformly distributed. There is a chance to increase the accuracy by using all the information including NIR.

Additionally, more sophisticated models like ResNet was beyond the scope of this project because of the resource limitation. Training this kind of models on multiple GPUs should improve the result significantly.

5 Conclusion

Throughout this project, I was able to explore many different deep learning frameworks, and how to use Tensorflow for image classification tasks.

A Training Log

```
 --( $:~/612/deepsat )-- python ds2_run.py
- loading images
train: 150000, validation: 50000, test: 50000 loaded
- Training started
Step 0: loss = 1.84 (0.022 sec)
Step 100: loss = 1.57 (0.004 sec)
Step 200: loss = 1.53 (0.003 sec)
Step 300: loss = 1.52 (0.003 sec)
Step 400: loss = 1.49 (0.003 sec)
Step 500: loss = 1.43 (0.003 sec)
Step 600: loss = 1.45 (0.004 sec)
Step 700: loss = 1.41 (0.005 sec)
Step 800: loss = 1.38 (0.003 sec)
Step 900: loss = 1.34 (0.004 sec)
Training Data Eval:
  Precision: 0.5629 [Total: 150000  Correct: 84429]
Validation Data Eval:
  Precision: 0.5593 [Total: 50000  Correct: 27965]
Test Data Eval:
  Precision: 0.5626 [Total: 50000  Correct: 28130]
Step 1000: loss = 1.18 (0.014 sec)
Step 1100: loss = 1.22 (0.003 sec)
Step 1200: loss = 1.16 (0.004 sec)
.
.
.
Step 99100: loss = 0.29 (0.003 sec)
Step 99200: loss = 0.33 (0.004 sec)
Step 99300: loss = 0.21 (0.003 sec)
Step 99400: loss = 0.18 (0.003 sec)
Step 99500: loss = 0.22 (0.003 sec)
Step 99600: loss = 0.25 (0.005 sec)
Step 99700: loss = 0.19 (0.003 sec)
Step 99800: loss = 0.24 (0.003 sec)
Step 99900: loss = 0.21 (0.003 sec)
Training Data Eval:
  Precision: 0.9273 [Total: 150000  Correct: 139095]
Validation Data Eval:
  Precision: 0.9012 [Total: 50000  Correct: 45060]
Test Data Eval:
  Precision: 0.9088 [Total: 50000  Correct: 45440]
```

B Reference

Code (https://github.com/romanegloo/deepsat)
DEEPSAT dataset (http://csc.lsu.edu/~saikat/deepsat/)
Tensorflow (https://www.tensorflow.org/)