Convolution Neural Network based on Embedded System

Li Guan Ting[*]
UEC Student No. 1495005
TamKang University (TKU)
New Taipei City, Taiwan

Takayuki NAGAI
Department of Mechanical Engineering
The University of Electro-Communications
Tokyo, Japan

Abstract

In this paper, we are using Caffe library which is deep learning framework maintained and developed by Berkeley Vision and Learning Center(BLVC). Convolution Neural Network(CNN) is an image learning algorithm for classification objects which is a heavy duty for computer. But in many situations, such as building a small robot which do not have enough space to put a PC or laptop on it to do image processing. In this case, we need a small computer so called embedded system which is powerful enough to do this job.

Keywords: Convolution Neural Network, Machine Learning, Classification, Image Processing, Embedded System

1 Introduction

Machine learning systems automatically learn programs from data. This is often a very attractive alternative to manually constructing them, and in the last decade the use of machine learning has spread rapidly throughout computer science and beyond. Machine learning is used in web search, spam filters, recommender systems, ad placement, credit scoring, fraud detection, stock trading, drug design, and many other applications. Machine learning have many different types and Convolution Neural Network(CNN) is one of them.

In this paper, we go to use CNN which has been used on image recognition and voice analysis field widely. The benefit of CNN is, it use share weights structure which is more likely biology neural network. And can decrease network model complexity and number of weights quantity. The advantage can be more obviously when input is a high dimension images. CNN is a special design of multilayer perceptron(MLP), this structure have a high invariance to shifting, scaling, rotation, or other invariant processing.

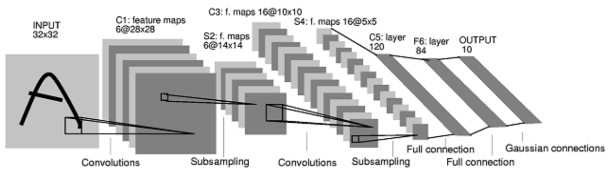


Figure 1: Convolution Neural Network Structure

CNN is a multilayer neural network which is shown in Fig.1. Every layer is made up of many two dimension plane, and every plane is made up of many independent neurons. The concept of CNN is like below: First layer we also called input layer, C_X layer called convotion layer, S_X layer called sub-sampling layer and then full connection to output layer. Convolution is to get the feature from previous layer, using convolution can make feature more obvious and decrease noise. Subsampling or down-sampling, can reducing size of signal and still can remain useful information.
First covolution first layer by 5*5 patch convert C_X layer into six 28*28 feature maps, then use 2*2 patch sum up and add a bias, finally through sigmoid activation function, to scaling down 4 times smaller feature map become S_{X+1} which shown in Fig.2.
Continue loop convolution and sub-sampling to the end, then calculate the error between output and training data. Do backward processing to update all weights and baises.

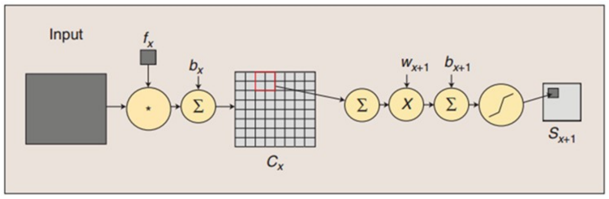


Figure 2: Convolution and Sub-sampling Procedure

[*]The author is supported by UEC Scholarship (Apr to Dec) and JASSO Scholarship (Jan to Mar)

2 Objective

Transplant big and heavy object recognize algorithm structure into efficiency and light which can be implement on embedded system.

3 Method

3.1 Convolution Layers

At a convolution layer, the previous layers feature maps are convolved with learnable kernels and put through the activation function to form the output feature map. Each output map may combine convolutions with multiple input maps. In general, we have that:

$$x_j^l = f(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l) \tag{1}$$

where M_j represents a selection of input maps. Each output map is given an additive bias b, however for a particular output map, the input maps will be convolved with distinct kernels. That is to say, if output map j and map k both sum over input map i, then the kernels applied to map i are different for output maps j and k.

3.2 Sub-sampling Layers

A subsampling layer produces downsampled versions of the input maps. If there are N input maps, then there will be exactly N output maps, although the output maps will be smaller. More formally,

$$x_j^l = f(\beta_j^l down(x_j^{l-1}) + b_j^l) \tag{2}$$

where down(.) represents a sub-sampling function. Typically this function will sum over each distinct n-by-n block in the input image so that the output image is n-times smaller along both spatial dimensions. Each output map is given its own multiplicative bias and an additive bias b. We can also simply throw away every other sample in the image.

3.3 Embedded System

In order to use embedded system to process CNN algorithm heavy duty on computing, which need powerful CPU, and need parallel processor to deal with CNN. And small size and low power Consumption for small robot which don't have too many space for circuit board and battery. Our solution is Parallella board shown in Fig.3.

Table 1: Parallella Specifications

Processor	ARM Dual-core A9 @1GHz
Coprocessor	Epiphany 16-core CPU
Size	90mm*55mm*18mm
Power Consumption	<5 Watt

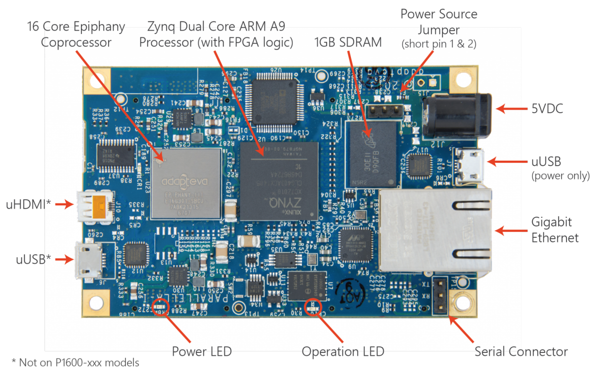

16 Core Epiphany Coprocessor
Zynq Dual Core ARM A9 Processor (with FPGA logic)
1GB SDRAM
Power Source Jumper (short pin 1 & 2)
5VDC
uUSB (power only)
Gigabit Ethernet
uHDMI*
uUSB*
Power LED
Operation LED
Serial Connector
* Not on P1600-xxx models

Figure 3: Parallella Board

3.4 Program

This is program structure shown in Fig.4. First Webcam get image information through USB cable go to Client side which we use Windows as operating system, and convert from image stream into picture using JPG format. Through Internet sent picture to server side which is Parallella board, here we use Ubuntu 14.04 as operating system. CNN algorithm will recognize it and sent result back to client side.

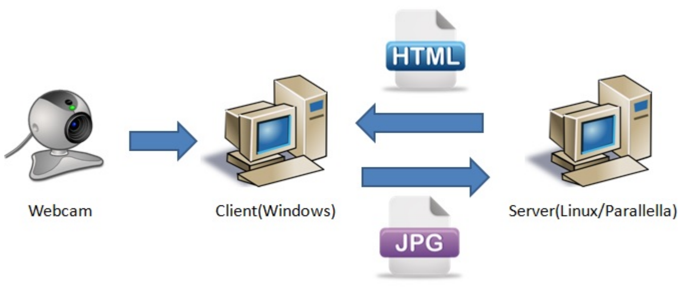
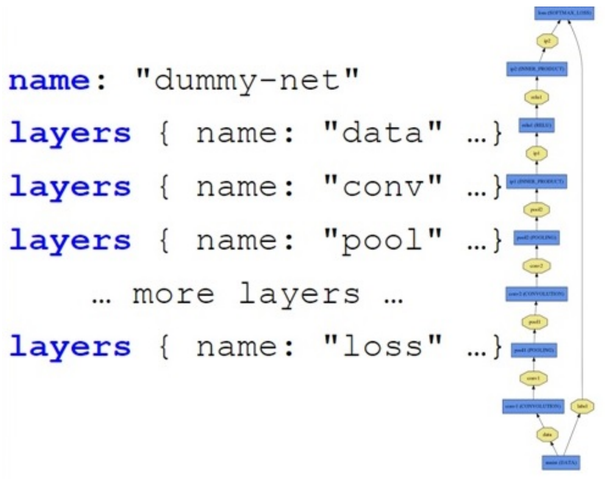


Figure 4: Program Structure

3.5 Caffe Library

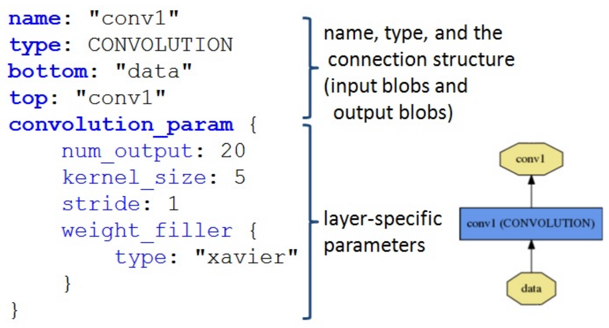
Caffe library provides a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying general-purpose convolution neural network and other deep models efficiently on commodity architectures. Caffe is maintained and developed by the Berkeley Vision and

Learning Center(BVLC). Caffe is easy to build layer structure and setting parameters such as connect structure, type of math to process input data, kernel size, output number to generate output data to next layer shown in Fig.5, Fig.6.



```
name: "dummy-net"
layers { name: "data" …}
layers { name: "conv" …}
layers { name: "pool" …}
    … more layers …
layers { name: "loss" …}
```

Figure 5: Create Layers



```
name: "conv1"
type: CONVOLUTION
bottom: "data"
top: "conv1"
convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
        type: "xavier"
    }
}
```

name, type, and the connection structure (input blobs and output blobs)
layer-specific parameters
conv1
conv1 (CONVOLUTION)
data

Figure 6: Create Layers

4 Results

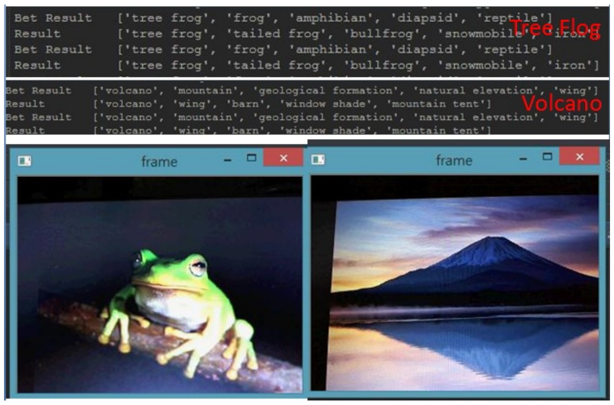
In this paper, we are using ImageNet database (ILSVRC12 challenge) for training CNN which contain category, each category have 1200 images for training. Total 12 milion images, 138GB for classification.

In Table 2. we compare Caffe run on PC in CPU mode and GPU mode with Parallella in CPU mode. Operating sysyem of PC is running Ubuntu 12.04 with Nvidia GTX960 graphic card. As the result, PC running Caffe library on graphic card is 189 times faster and running Caffe Library on PC in CPU mode is 9.7 times faster than transplant Caffe library on Parallella board.

Table 2: Parallella result

Processor	PC(CPU)	PC(GPU)	Parallella(CPU)
Time(sec)	2.5(x9.7)	0.128(x189)	24.2(x1)

We design a classification system to recognize objects which have learned from ImageNet database(ILSVRC12 challenge). According to the possibility that classification system have recognized, sorting into a list from top to low, and we choose top 5 rate catagory into considered list and show result in textbox. As the result, in Fig.7 classification system can recognize flog as tree flog and mountain as volcano.



```
Bet Result   ['tree frog', 'frog', 'amphibian', 'diapsid', 'reptile']
Result       ['tree frog', 'tailed frog', 'bullfrog', 'snowmobile', 'iron']
Bet Result   ['tree frog', 'frog', 'amphibian', 'diapsid', 'reptile']
Result       ['tree frog', 'tailed frog', 'bullfrog', 'snowmobile', 'iron']
```

```
Bet Result   ['volcano', 'mountain', 'geological formation', 'natural elevation', 'wing']
Result       ['volcano', 'wing', 'barn', 'window shade', 'mountain tent']
Bet Result   ['volcano', 'mountain', 'geological formation', 'natural elevation', 'wing']
Result       ['volcano', 'wing', 'barn', 'window shade', 'mountain tent']
```

Figure 7: CNN Result

Regions with Convolutional Neural Network(R-CNN) is a state-of-the-art visual object detection system that combines bottom-up region proposals with rich features computed by a convolutional neural network. With these technique, algorithm can detect where object and classification at the same time. The result is shown in Fig.7.

5 Conclusions

In this paper, we transplant caffe library into Parallella board. We can directly send image we want to recognize through Internet and embedded system will run convolution neural network, do image classification. After all recognition procedure Parallella board will sent result back through Internet and waitting for next image.

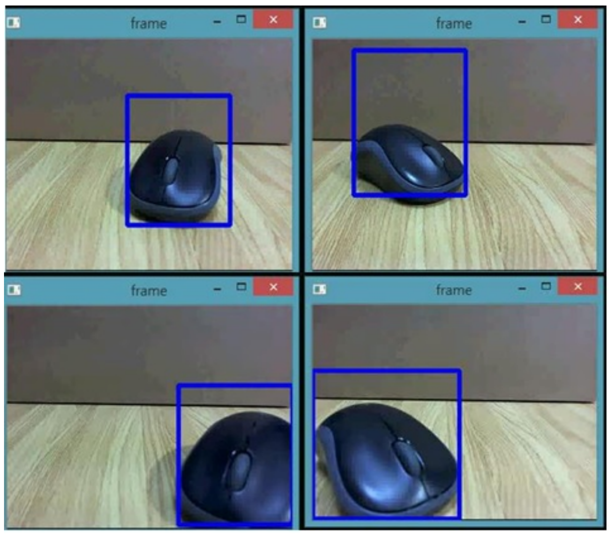


Figure 8: R-CNN Result

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton "ImageNet Classification with Deep Convolutional Neural Network," *Neural Information Processing Systems*, 2012.

[2] R. Girshick, J. Donahue, T. Darrell, J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2013.

[3] Y. Jia, E. Shelhamer, J. Donahue, "Caffe: Convolutional Architecture for Fast Feature Embedding," 2014.

[4] J. Donahue, Y. Jia, O. Vinyals, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," 2013.

[5] Caffe: http://caffe.berkeleyvision.org/