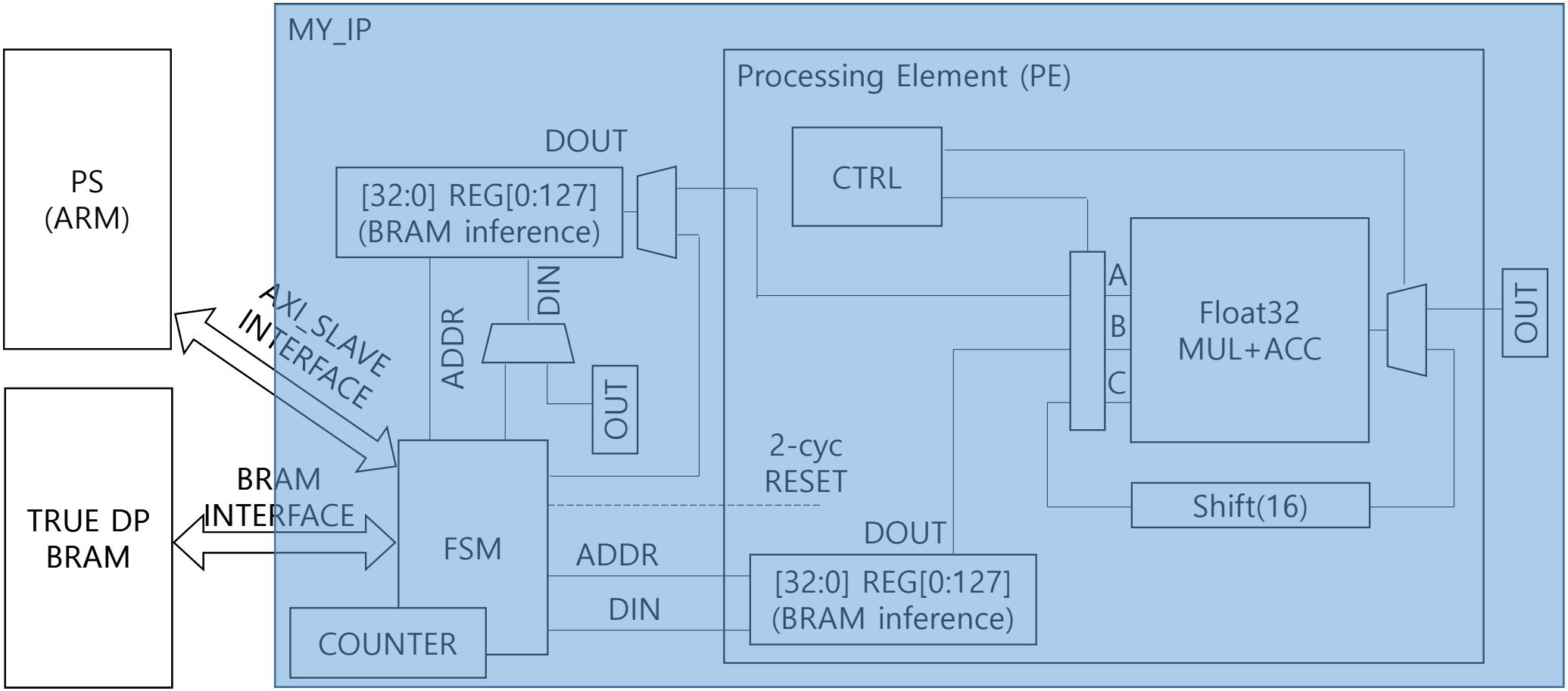
Practice 7

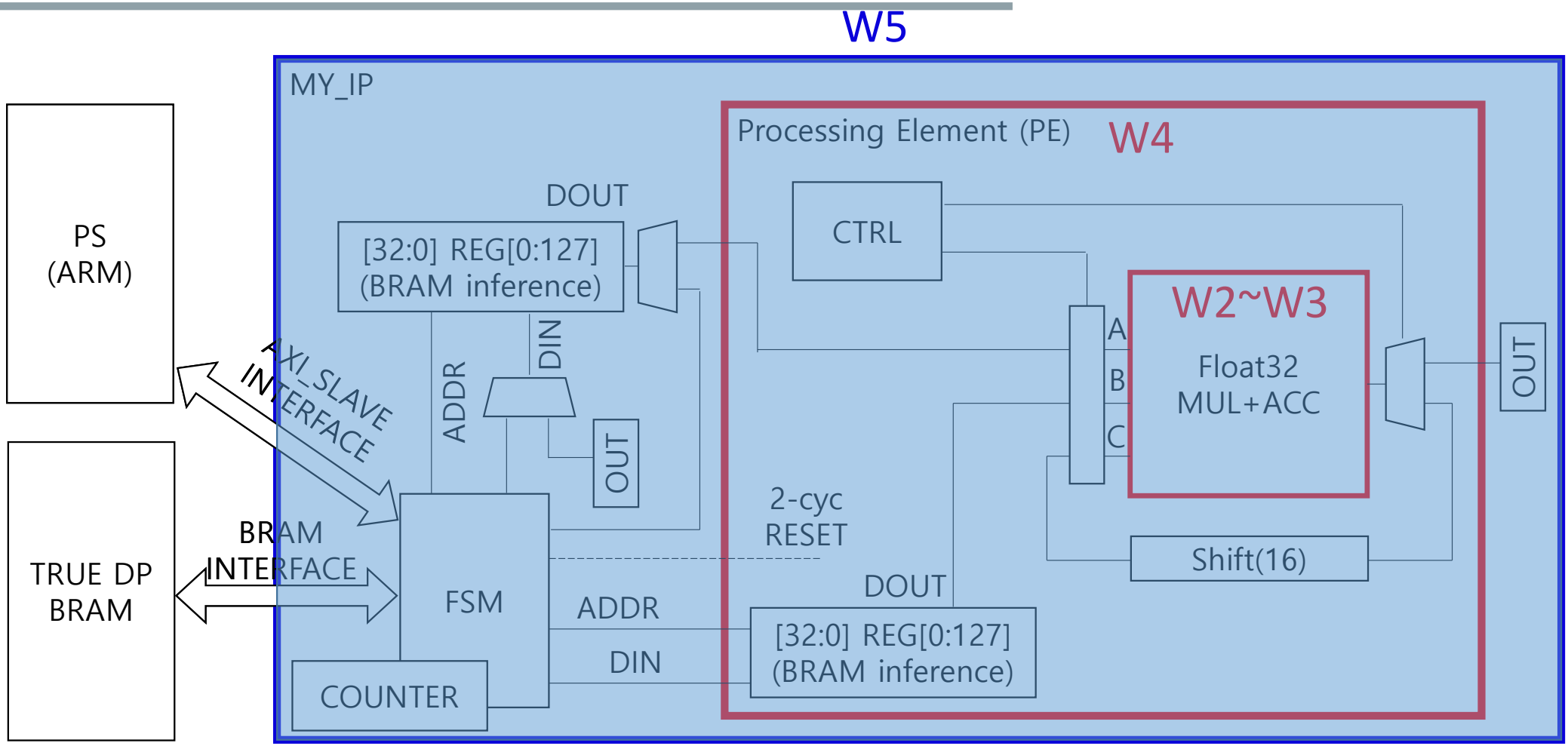
- Convolution Lowering SW

Computing Memory Architecture Lab.

Final Project Overview: Matrix Multiplication IP



Final Project Overview: Matrix Multiplication IP



Final Project Overview: Matrix Multiplication IP

- Our original application

One layer in MLP (Software running on CPU)

```
for(j=0; j<1024; j+=8) {
   for(i=0; i<256; i+=8) {
      Output[i] += Input[j]*W[i,j] + Input[j+1]*W[i,j+1] + ...
      Output[i+1] += Input[j]*W[i+1,j] + Input[j+1]*W[i+1,j+1] + ...
      ...
      Output[i+7] += Input[j]*W[i+7,j] + Input[j+1]*W[i+7,j+1] + ...
   }
}
```

MV function on Hardware

Vector-Vector Multiplication

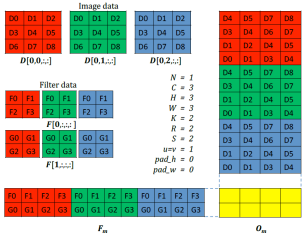
- Advanced application

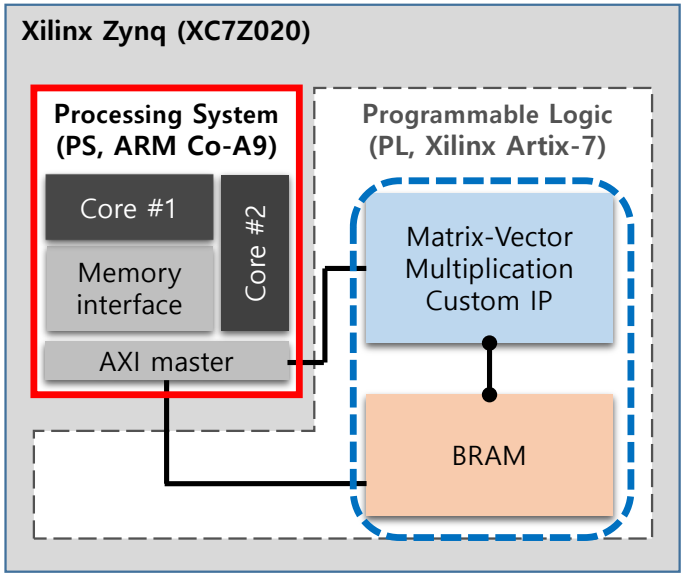
One layer in Convolution with Convolution Lowering

```
for(j=0; j<1024; j+=8) {
   for(i=0; i<256; i+=8) {
      Output[i] += Input[j]*W[i,j] + Input[j+1]*W[i,j+1] + ...
      Output[i+1] += Input[j]*W[i+1,j] + Input[j+1]*W[i+1,j+1] + ...
      ...
      Output[i+7] += Input[j]*W[i+7,j] + Input[j+1]*W[i+7,j+1] + ...
   }
}
```

MV function on Hardware

Vector-Vector Multiplication

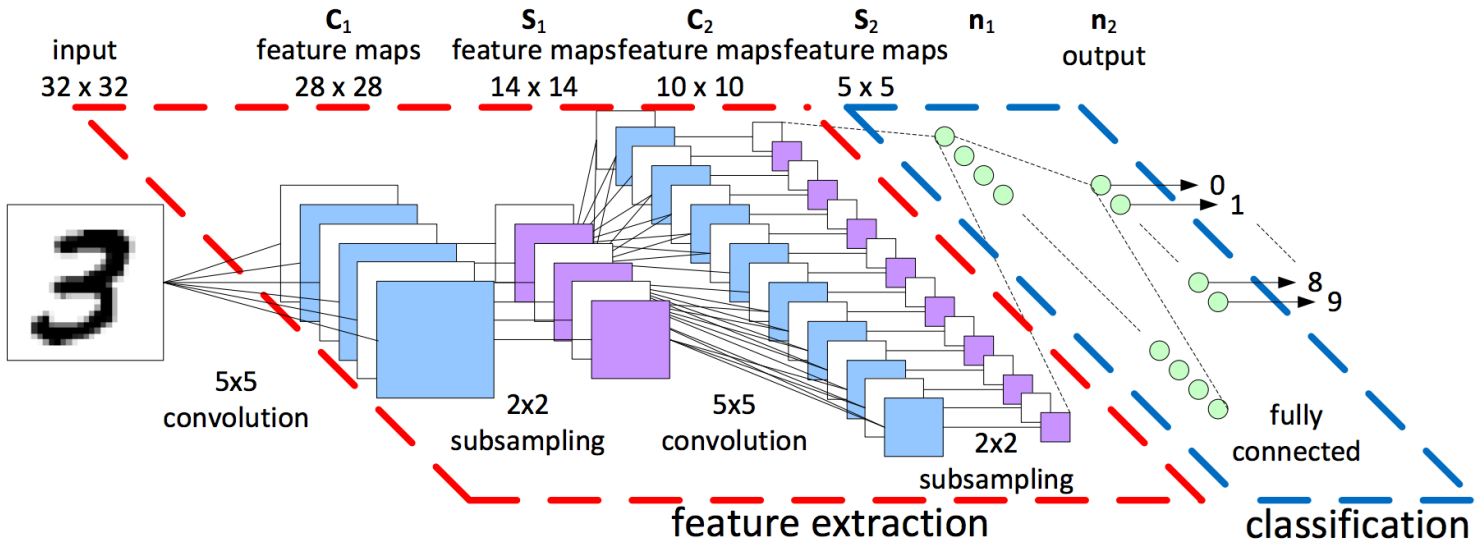




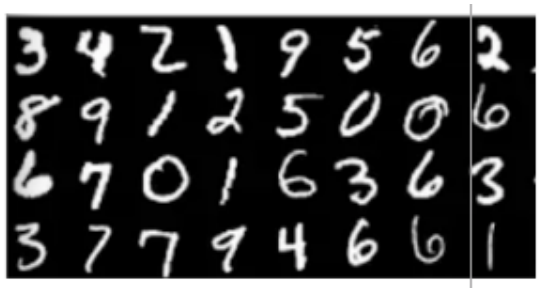
Main Practice

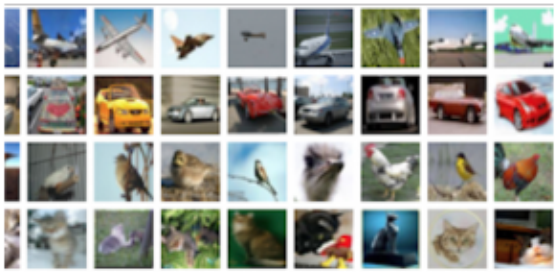
Lab 7: Overview

- Goal
  - Implement convolution lowering in C++
  - Integrate convolution lowering into the pretrained model(CNN)
    - On MNIST
  - Example) Convolutional Neural Network



MNIST, CIFAR 10

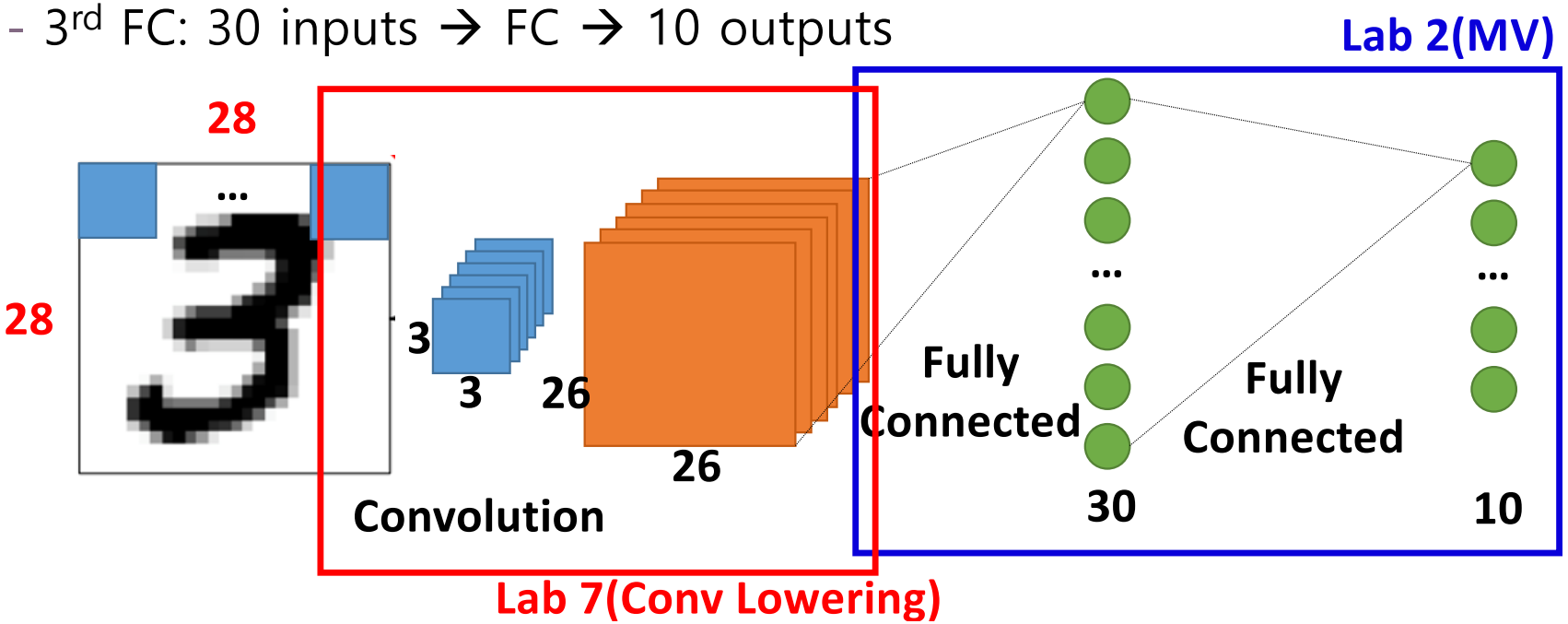




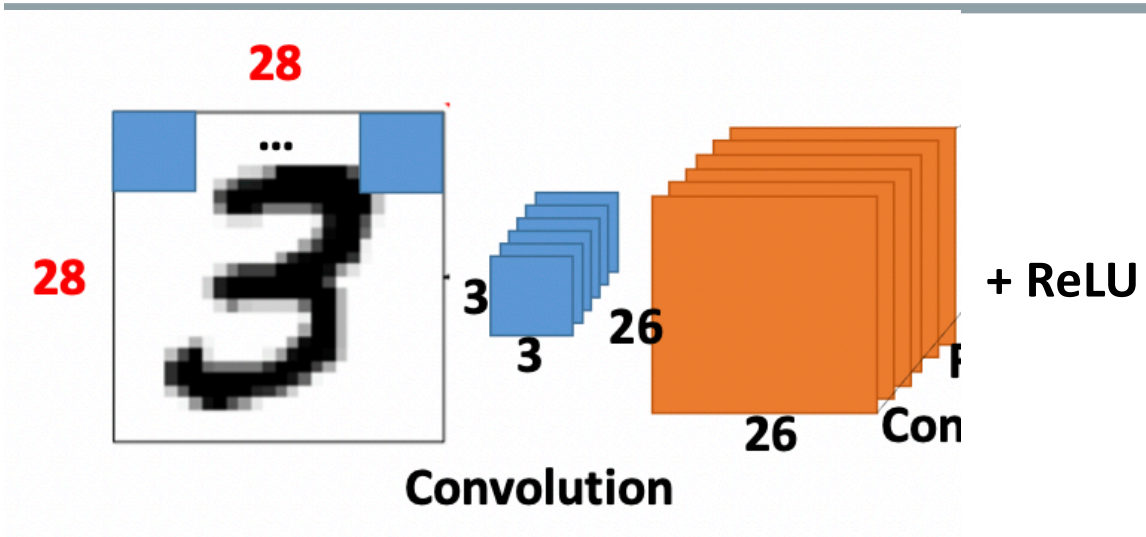
Dataset	MNIST	CIFAR 10
Category	digits(0-9)	airplane, automobile, …, truck
Image size	28x28	32x32
Color	Gray scale	RGB
# images (training/test)	60000/10000	60000/10000

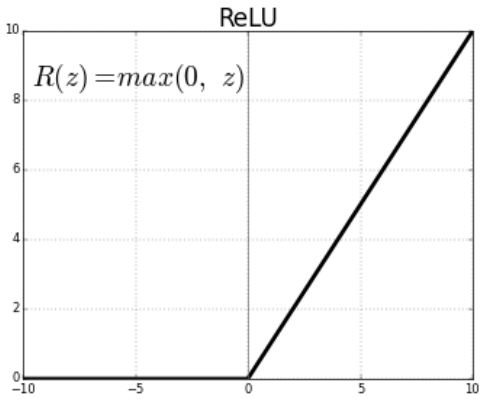
(pretrained) Convolutional Network Network(CNN)

- Input: **28x28** pixels → **6 3x3** Conv → **30** values → **10** values
  - 1st Conv: **28x28** inputs → 6 3x3 Conv → 6 26x26 outputs
  - 2nd FC: 6*26*26(=4056) inputs → FC → 30 outputs
  - 3rd FC: 30 inputs → FC → 10 outputs



Implementation Detail: 1st layer





1) Convolution → Convolution Lowering

Output[0][0, 0] = $\Sigma_{i,j}$ Input[i,j]*W[0][i,j]

Output[0][0, 1] = $\Sigma_{i,j}$ Input[i,j+1]*W[0][i,j]

…

Output[5][25, 25] =

$\Sigma_{i,j}$ Input[i+25,j+25]*W[5][i,j]

2) Activation(ReLU)

Output[0][0, 0] = max(0, Output[0][0, 0])

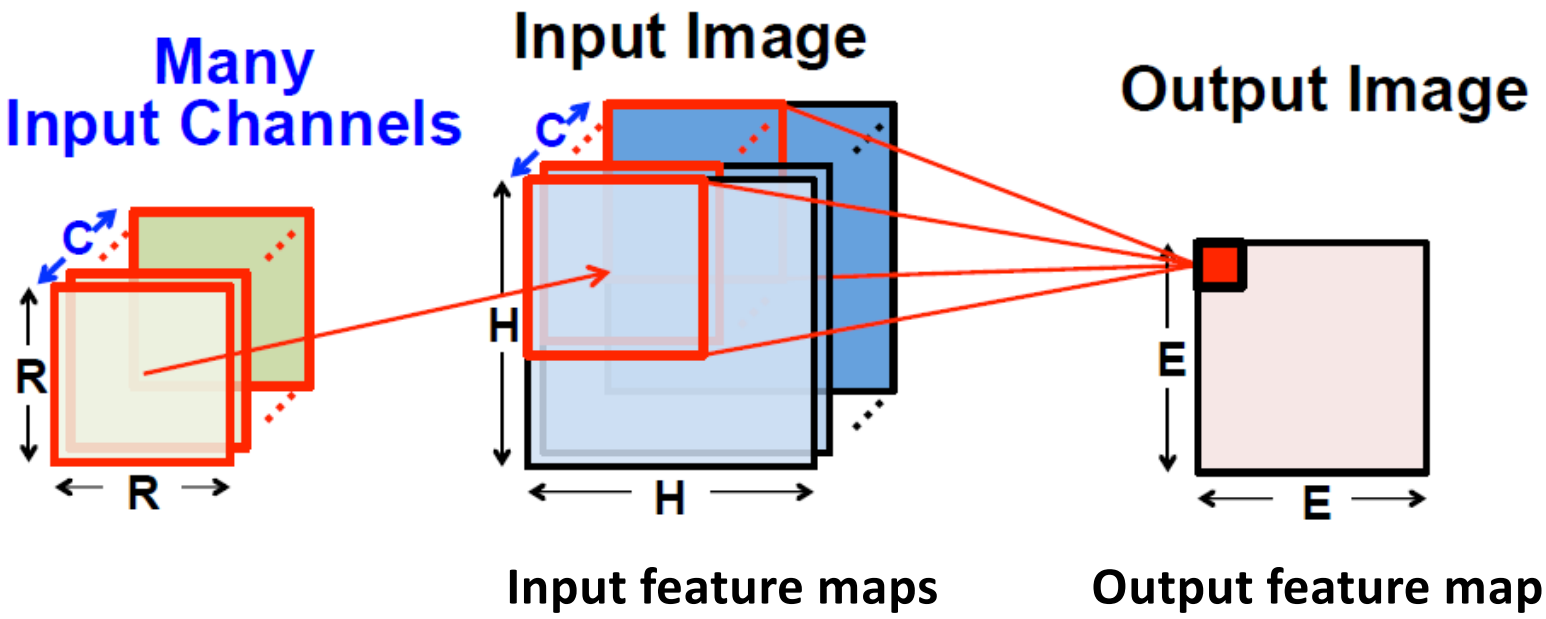
…

Output[5][25, 25] = max(0, Output[5][25, 25])

[Chen, 2016]

Convolution: 3D Input Case

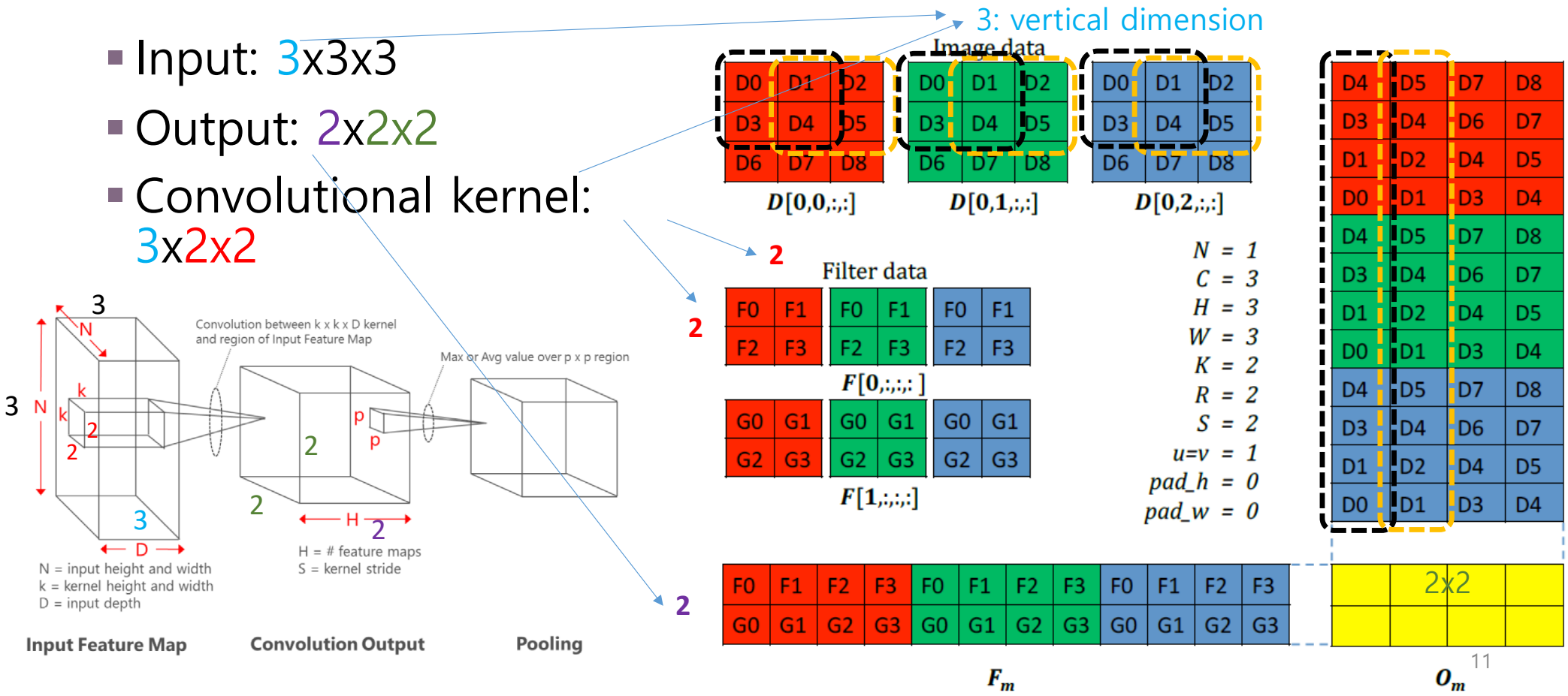
- A receptive field in input feature maps gives an output neuron
- Each output feature map has its own set of kernel weights



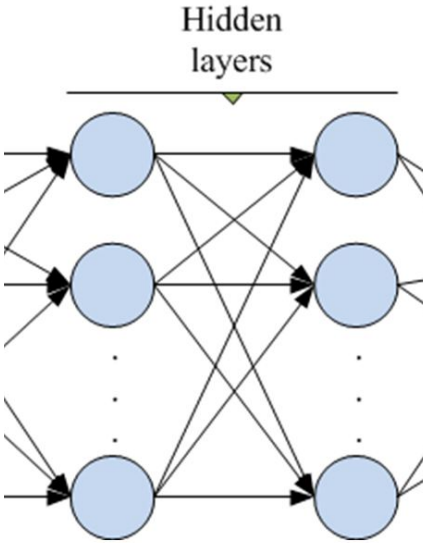
Convolution with Matrix Multiplication (called Convolution Lowering)

[Chetlur 2014]

- Input: 3x3x3
- Output: 2x2x2
- Convolutional kernel: 3x2x2



Implementation Detail: 2nd layer



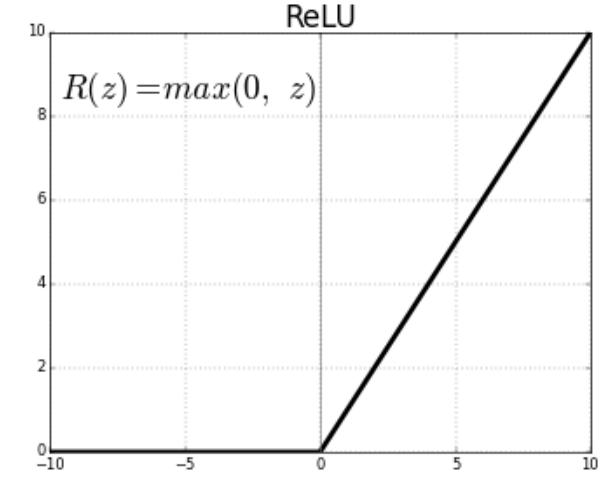
+ ReLU

+ ReLU

+ ReLU

6*26*26 =4056

30



1) (MV) Multiplication

Output[0] = Σ_j Input[j]*W[0,j]

Output[1] = Σ_j Input[j]*W[1,j]

…

Output[29] = Σ_j Input[j]*W[29,j]

2) Activation(ReLU)

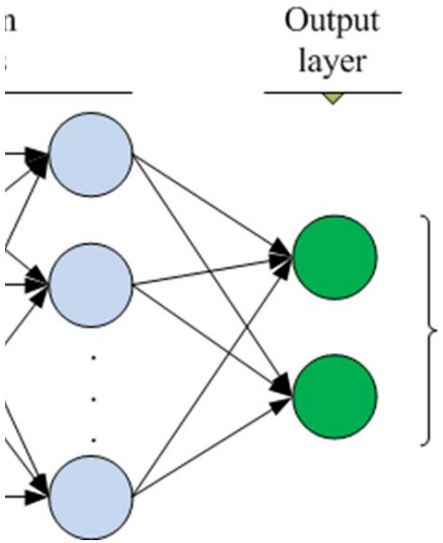
Output[0] = max(0, Output[0])

Output[1] = max(0, Output[1])

…

Output[29] = max(0, Output[29])

Implementation Detail: 3rd layer



+ Softmax

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

1) (MV) Multiplication

Output[0] = Σ_j Input[j]*W[0,j]
Output[1] = Σ_j Input[j]*W[1,j]
…
Output[9] = Σ_j Input[j]*W[9,j]

2) Softmax

Output[0] = softmax(Output[0])
…
Output[9] = softmax(Output[9])

3) Argmax

Prediction = argmax_i (Output[i: 0~9])

Install

- **Connect to the Server we distributed in Lab02**
- Code download
  - $ git clone https://github.com/tahsd/hsd20_lab07
  - $ cd hsd20_lab07
- Dataset download
  - $ bash download.sh

Skeleton Code

```
.
|- Makefile
|- src
| |- fpga_api.cpp         # IMPLEMENT THIS!(only convLowering)
| |- fpga_api_on_cpu.cpp  # IMPLEMENT THIS!(only convLowering)
| |- tf_dnn.cpp     # Lab 07, Note: Don't need to edit
| |- caffe_dnn.cpp  # Lab 02, Note: Don't need to edit
| `- py_lib.cpp     # don't need to edit
|
|- eval.py        # Evaluate the pretrained model
|
|- download.sh
|- pretraiend_weights
| |- cnn_weights.txt            # Lab 07, CNN(28x28x1->6 3x3conv->fc30->fc10)
| `- mlp_iter_10000.caffemodel  # Lab 02, MLP(784-1200-1200-10)
|
|- include
`- data
```

Implement Convolution Lowering

- Edit `./src/fpga_api.cpp`, `./src/fpga_api_on_cpu.cpp`

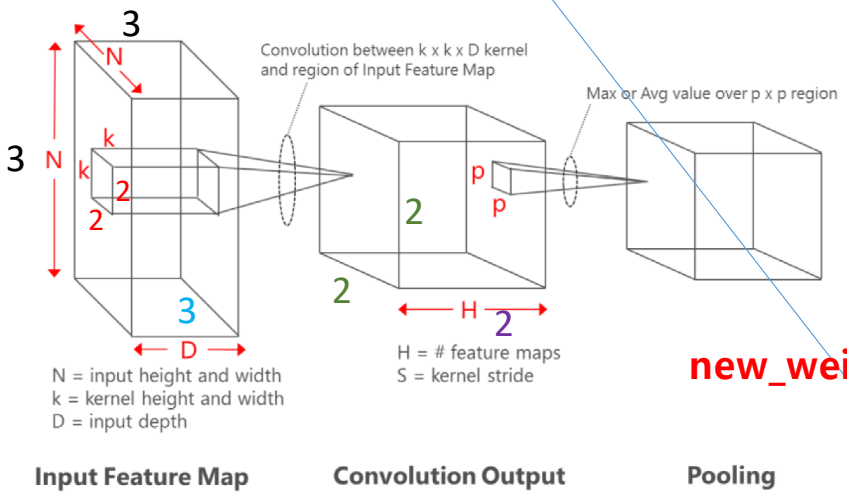
```
// src/fpag_api.cpp, src/fpag_api_on_cpu.cpp
void FPGA::convLowering(const vector<vector<vector<vector<float>>>>& cnn_weights,
                        vector<vector<float>>& new_weights,
                        const vector<vector<vector<float>>>& inputs,
                        vector<vector<float>>& new_inputs) {
  /*
   * Arguments:
   *
   * conv_weights: [conv_channel, input_channel, conv_height, conv_width]
   * new_weights: [?, ?]
   * inputs: [input_channel, input_height, input_width]
   * new_inputs: [?, ?]
   *
   */
   ...
}
```

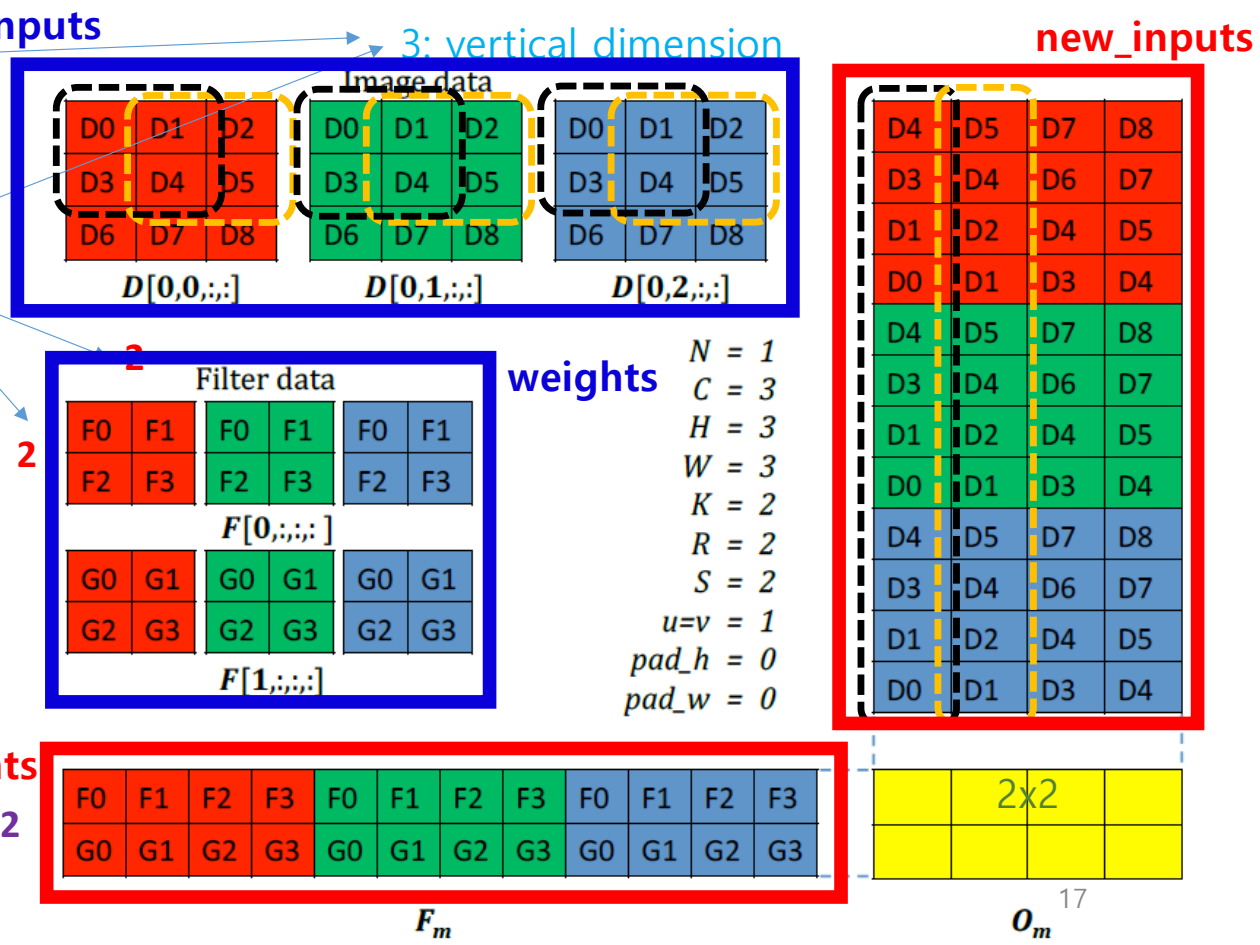
- [Note] Fill the function `largeMV` that you implemented in Lab 02

[Chetlur 2014]

Convolution Lowering

- Input: 3x3x3
- Output: 2x2x2
- Convolutional kernel: 3x2x2





Run

- Edit `./src/fpga_api.cpp`, ./src/fpag_api_on_cpu.cpp`
  - void convLowering(…)
- Build
  - $ make
- Evaluate
  - $ python eval.py --num_test_images 10 --m_size 64 --v_size 64 --network cnn --run_type cpu
- Options
  - num_test_images : 1~10000
  - m_size, v_size : 8, 16, 32, 64, …
  - network: cnn, mlp
    - cnn(lab07), mlp(lab2)
  - run_type: cpu
    - We don't have FPGA now, so you have to write this option when you eval

Example: Download datasets

```
zed@debian-zynq:~/                    :master$ bash download.sh
--2019-05-13 17:56:45--  https://dl.dropbox.com/s/mdwy0kzf57nfl5f/t10k
Resolving dl.dropbox.com (dl.dropbox.com)... 162.125.80.6, 2620:100:60
Connecting to dl.dropbox.com (dl.dropbox.com)|162.125.80.6|:443... con
HTTP request sent, awaiting response... 302 FOUND
Location: https://dl.dropboxusercontent.com/s/mdwy0kzf57nfl5f/t10k-ima
--2019-05-13 17:56:46--  https://dl.dropboxusercontent.com/s/mdwy0kzf5
Resolving dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 162
Connecting to dl.dropboxusercontent.com (dl.dropboxusercontent.com)|16
HTTP request sent, awaiting response... 200 OK
Length: 7840016 (7.5M) [application/octet-stream]
Saving to: ‘data/t10k-images.idx3-ubyte’

data/t10k-images.idx3-ubyte                          100%[=======

2019-05-13 17:56:49 (3.70 MB/s) - ‘data/t10k-images.idx3-ubyte’ saved
```

Example: Build your implementations

```
zed@debian-zynq:~/                    :master$ make
protoc -I=./proto --cpp_out=./proto proto/caffe.proto
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/py_lib.o
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/caffe.pb
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/caffe_dn
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/tf_dnn.o
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/fpga_api
g++ -shared -o build/libpylib_cpu.so  build/py_lib.o build/caffe.
g++ -fPIC -std=c++11 -O3 -I ./include -I./proto -o build/fpga_api
g++ -shared -o build/libpylib_fpga.so  build/py_lib.o build/caffe
```

Example: Evaluation

```
zed@debian-zynq:~/            :master$ sudo python eval.py --num_test_images 10 --m_size 8 --v_size 64 --network mlp --run_type cpu
[*] Arguments: Namespace(m_size=8, network='mlp', num_test_images=10, run_type='cpu', v_size=64)
[*] Read MNIST...
[*] The shape of image: (10, 28, 28)
[*] Load the network...
[*] Run tests...
[*] Statistics...
{'accuracy': 0.9,
 'avg_num_call': 4838,
 'm_size': 8,
 'total_image': 10,
 'total_time': 0.5971291065216064,
 'v_size': 64}
```

- Accuracy(10,000 images)
  - MLP(Lab 2): around 95%
  - CNN(Lab 11): around 98%

Checklists

- MLP(Lab 2)
  - $ sudo python eval.py --num_test_images 10 --m_size 64 --v_size 64 **--network mlp --run_type cpu**
- MV
  - $ sudo python eval.py --num_test_images 10 --m_size 64 --v_size 64 --network mlp **--run_type cpu**
- Convolution Lowering(Lab 7)
  - $ sudo python eval.py --num_test_images 10 --m_size 64 --v_size 64 **--network cnn** --run_type cpu
- Other arguments
  - num_test_images, m_size, v_size, …

Example: Benchmark

- $ bash benchmark.sh

- Please remove ‘sudo’ command in benchmark.sh file

```
zed@debian-zynq:~/            :master$ sudo bash benchmark.sh
[*] Arguments: Namespace(m_size=64, network='mlp', num_test_images=100, run_type='cpu', v_size=
[*] Read MNIST...
[*] The shape of image: (100, 28, 28)
[*] Load the network...
[*] Run tests...
[*] Statistics...
{'accuracy': 0.08,
 'avg_num_call': 627,
 'm_size': 64,
 'total_image': 100,
 'total_time': 3.223191022872925,
 'v_size': 64}

=> Accuracy should be 0.97

[*] Arguments: Namespace(m_size=64, network='mlp', num_test_images=100, run_type='fpga', v_siz
[*] Read MNIST...
[*] The shape of image: (100, 28, 28)
[*] Load the network...
[*] Run tests...
[*] Statistics...
{'accuracy': 0.07,
 'avg_num_call': 627,
 'm_size': 64,
 'total_image': 100,
 'total_time': 6.748210906982422,
 'v_size': 64}

=> Accuracy should be 0.97

[*] Arguments: Namespace(m_size=64, network='cnn', num_test_images=100, run_type='cpu', v_size=
[*] Read MNIST...
[*] The shape of image: (100, 28, 28)
[*] Load the network...
[*] Run tests...
[*] Statistics...
{'accuracy': 0.08,
 'avg_num_call': 741,
 'm_size': 64,
 'total_image': 100,
 'total_time': 3.8482890129089355,
 'v_size': 64}

=> Accuracy should be 1.0
```

Homework

- Requirements
  - Result
    - Attach your codes(e.g., fpga_api.cpp, fpga_api_on_cpu.cpp)
      - Refer Lab02
    - Attach your benchmark.sh results with [student_number, name]
      - Refer Lab02
  - Report
    - Explain Convolution Lowering that you implemented
    - In your own words
    - Either in Korean or in English
    - # of pages does not matter
    - **PDF only!!**
  - **Result + Report to a .zip**
- Upload (.zip) file on ETL
  - Submit one (.zip) file
    - zip file name : [Lab07]name.zip (ex : [Lab07]홍길동.zip)
  - Due: 5/5(TUE) 23:59
    - **No Late Submission**

```
// src/fpga_api.cpp
void FPGA::largeMV(const float* l
                   float* output
{
    // IMPLEMENT HERE
    ...
}
```