Machine Learning Intelligent Chip Design

Homework1 Implementation of AlexNet in SystemC

Description

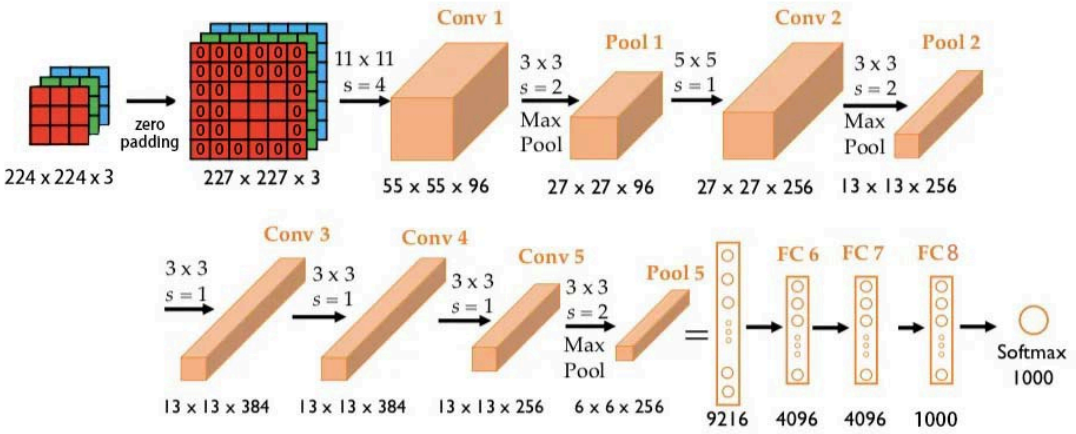
Implement the AlexNet CNN architecture using SystemC. The model consists of convolutional layers, max-pooling layers, and fully connected layers, following its original design for image classification tasks.

Implementation Details

- **Network Architecture**:
  You are provided with the network architecture details, including the number of layers, the size of input and output feature maps, filter sizes, and strides.
- **Input Data**:
  Input data for the network will be provided to you. Each input image will have the appropriate dimensions compatible with the network's input layer.
- **Weights and Biases**:
  The weights and biases for each layer of the network will be provided. These parameters are essential for the convolutional and fully connected layers' computations.
- **Simulation**:
  Once you have implemented the network in SystemC, you should simulate the network to obtain the predicted output. This process involves passing the input data through the network layers, applying convolutional operations, activation functions, and pooling, followed by fully connected layers, until the final output is obtained.

Alexnet Training Model

The AlexNet model is shown in the diagram below. Before the input image enters the first convolutional layer, **zero padding** should be applied: **two rows on the top and left, and one row on the bottom and right**. Additionally, since the results need to be displayed as probabilities, please apply softmax at the final stage.



The Pre-trained AlexNet Model Information

Layer	Type	Description	Output Dimension
0	Input Layer	Zero Padding 224 to 227	227x227x3
1	Convolutional Layer 1 + ReLU	64 kernels of 11x11, stride 4	55x55x64
1b	Max Pooling 1	3x3, stride 2	27x27x64
2	Convolutional Layer 2 + ReLU	192 kernels of 5x5, stride 1, padding 2	27x27x192
2b	Max Pooling 2	3x3, stride 2	13x13x192
3	Convolutional Layer 3 + ReLU	384 kernels of 3x3, stride 1, padding 1	13x13x384
4	Convolutional Layer 4 + ReLU	256 kernels of 3x3, stride 1, padding 1	13x13x256
5	Convolutional Layer 5 + ReLU	256 kernels of 3x3, stride 1, padding 1	13x13x256
5b	Max Pooling 5	3x3, stride 2	6x6x256
6	Fully Connected Layer 6 + ReLU	4096 neurons	4096
7	Fully Connected Layer 7 + ReLU	4096 neurons	4096
8	Fully Connected Layer 8	1000 neurons	1000
9	Softmax Layer	Converts logits to probabilities	1000

Provided Data Description

Values in the pre-train model in Pytorch are floating points with 16 digits after the decimal. We export these values as txt file for you. Values in these txt files are floating point but rounded to the sixth decimal place.

- **Model layer parameters**

  conv1_bias.txt
  conv1_weight.txt
  conv2_bias.txt
  conv2_weight.txt
  conv3_bias.txt
  conv3_weight.txt
  conv4_bias.txt
  conv4_weight.txt
  conv5_bias.txt
  conv5_weight.txt
  fc6_bias.txt
  fc6_weight.txt
  fc7_bias.txt
  fc7_weight.txt
  fc8_bias.txt
  fc8_weight.txt

- **imagenet_classes.txt**
  https://gist.github.com/ageitgey/4e1342c10a71981d0b491e1b8227328b
- **Input Data**

dog.txt

cat.txt

Input image and weight reading format

In this assignment, you need to read the image and weight txt files provided by the TA to perform calculations. The reading method follows the **raster scan order**, as illustrated in the diagram below.

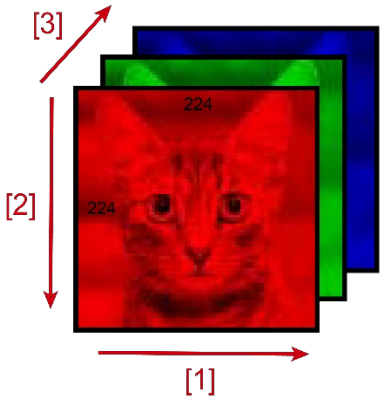
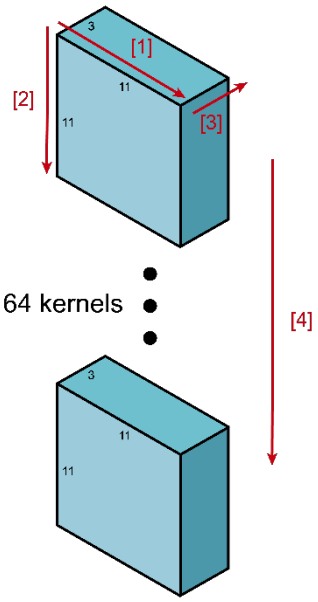


Image reading format



Weight reading format

Output Result Format

- **Simulation results of AlexNet executed in Python**
  This part is for reference only; you do not need to run Python yourself.
  Dog

```
In [20]: runfile('C:/Users/micha/Desktop/use_dataset_alexnet_model_instance.py', wdir='C:/Users/micha/Desktop')
Predicted class: golden retriever
```

  Cat

```
In [9]: runfile('C:/Users/micha/Desktop/use_dataset_alexnet_model_instance.py', wdir='C:/Users/micha/Desktop')
Predicted class: Egyptian cat
```

- **Simulation results of AlexNet executed in SystemC**
  Dog:

```
Top 100 classes:
================================================
  idx |       val | possibility | class name
------------------------------------------------
  207 |     16.59 |       38.63 | golden retriever
  175 |     15.57 |       13.86 | otterhound
  220 |     15.36 |       11.26 | Sussex spaniel
  163 |     15.00 |        7.86 | bloodhound
  219 |     14.59 |        5.22 | cocker spaniel
```

Cat:

```
Top 100 classes:
=================================================
  idx |       val | possibility | class name
-------------------------------------------------
  285 |     20.21 |       96.38 | Egyptian cat
  281 |     16.14 |        1.65 | tabby
  282 |     15.73 |        1.10 | tiger cat
  287 |     14.79 |        0.43 | lynx
  728 |     14.41 |        0.29 | plastic bag
  330 |     12.73 |        0.05 | wood rabbit
```

The example output above only displays the top 5 labels with the highest probability. However, in this assignment, you need to display the **top 100 labels with the highest probability**, and the **output format and layout must be identical to the example provided by the TA**. For example, both output values of the val and possibility should always **display two decimal places, regardless of the number of digits in the integer part.**

Below, we provide formatting code for reference. Please note that this code is for reference only. If you choose to use it, make sure to modify variable names and other details as needed.

```cpp
cout << fixed << setprecision(2);
cout << "Top 100 classes:" << endl;
cout << "=================================================" << endl;
cout << right << setw(5) << "idx"
     << " | " << setw(8) << "val"
     << " | " << setw(11) << "possibility"
     << " | " << "class name" << endl;
cout << "-------------------------------------------------" << endl;

for (int i = 0; i < 100; i++) {
    file.clear(); // Clear any potential error flags
    file.seekg(0, ios::beg); // Seek back to the beginning of the file
    int index = top_5_val[i].second;
    string line;
    for (int j = 0; j <= index; j++) {
        getline(file, line);
    }

    cout << right << setw(5) << index
         << " | " << setw(8) << top_5_val[i].first
         << " | " << setw(11) << (top_5_pos[i].first) // Assuming softmax outputs probabilities
         << " | " << line << endl;

}
cout << "=================================================" << endl;
```

Implement Notes

The purpose of this assignment is just to make students familiar with SystemC, so as long as the execution results are correct and follow some rules, we will not restrict how students implement it.

Here are some tips for your reference:

- You can use one `SC_MODULE` to implement the entire module, or implement each layer with different `SC_MODULE`.
- You can use `sc_signal` to connect different modules.
- We strongly recommend building a monitor module to receive output and print out the execution results.

Submission Guidelines & Grading Policy

- **The grading breakdown for this assignment is as follows:**
  - **Report : 30%**
    - Simulation results demonstrate the predicted output for the provided input data.
    - Your implementation approach, challenges faced, and any observations or insights gained during the implementation and simulation process.
  - **Simulation Result : 70%**
    - Only the two test datasets provided by the TA will be evaluated, **each worth 35 points**. The simulation results must **display the correct output on the terminal, and the formatting must match the provided example exactly.**
- **For the code submission, please use the compression and submission command provided by the TA.**
- **Please submit the report file to the new E3. The name of your report file is report_mlchipXXX.pdf (XXX is your account ID). If the file violates the naming rule and the file format, 10 points will be deducted.**
- **Ensure that your code is well-commented and organized for clarity and understanding.**
- **Plagiarism is forbidden, otherwise you will get 0 point!!!**

Submission & demo command

- **Please use make cat、make dog commands to execute your SystemC code. These commands correspond to running cat.txt and dog.txt, respectively. This means you need to modify how input files are handled by passing "dog" and "cat" as parameters to read the corresponding image files. We will provide a main function example and a parameter-passing example, as below.**

```
int sc_main(int argc, char* argv[]) {
    sc_clock clk("clk", 1, SC_NS);
    sc_signal<bool> reset;

    if (argc != 2) {  // Ensure exactly one filename is provided
        std::cerr << "Usage: " << argv[0] << " <file>" << std::endl;
        return 1;
    }

    std::string file = argv[1];  // Get the filename from the argument
    top->input_layer->load_data_and_pad("./data/" + file);
    return 0;
}
```

main function example

```cpp
void load_data_and_pad(const string& input_file_path){
    ifstream input_file(input_file_path.c_str());
}
```

parameter-passing example

- **Follow following command in 09_SUBMIT folder to submit your code and demo.**

1. **./00_tar**

```
10:05 mlchipTA05@ee21[~/hw1/09_SUBMIT]% ./00_tar
[Info] Directory created: hw1_mlchipTA05
[Info] Copied: ../main.cpp -> hw1_mlchipTA05/main_mlchipTA05.cpp
[Info] Creating tar archive: hw1_mlchipTA05.tar.gz
hw1_mlchipTA05/
hw1_mlchipTA05/main_mlchipTA05.cpp
[Success] hw1_mlchipTA05.tar.gz created successfully.
```

2. **./01_submit**

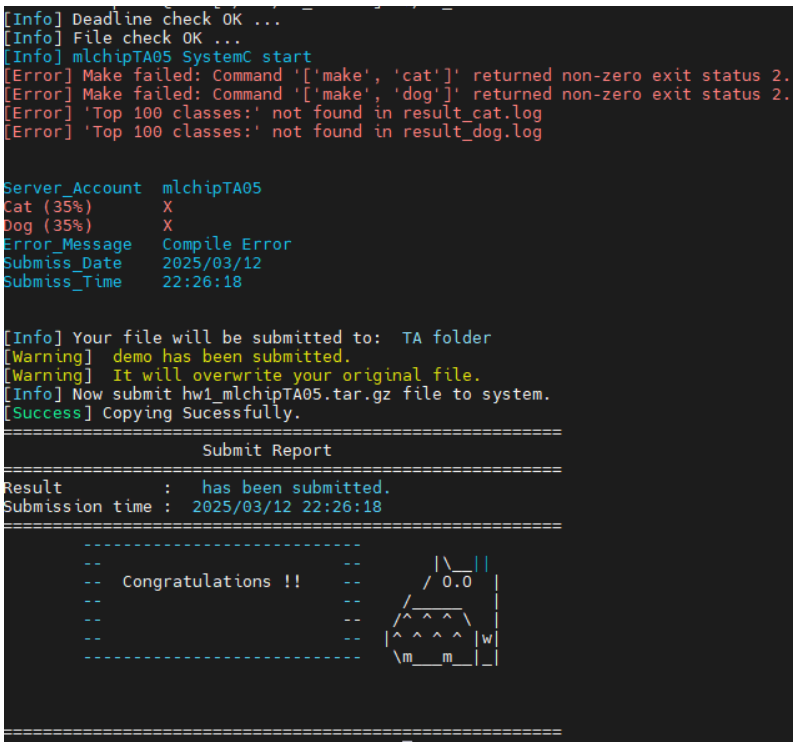
- **We will run make dog、make cat command to verify the correctness of your assignments. Simulation examples of correct and incorrect results are shown below.**

```
[Info] Deadline check OK ...
[Info] File check OK ...
[Info] mlchipTA05 SystemC start
[Info] result_cat.log Match Golden Result
[Info] result_dog.log Match Golden Result


Server_Account  mlchipTA05
Cat (35%)       0
Dog (35%)       0
Error_Message   No_Error
Submiss_Date    2025/03/12
Submiss_Time    22:16:37


[Info] Your file will be submitted to:  TA folder
[Info] Now submit hw1_mlchipTA05.tar.gz file to system.
[Success] Copying Sucessfully.
============================================================
                    Submit Report
============================================================
Result          :   has been submitted.
Submission time :  2025/03/12 22:16:37
============================================================
        ---------------------------
        --                       --       |\__||
        --  Congratulations !!   --      / 0.0  |
        --                       --    /_____   |
        --  Submission Sucessful!!--   /^ ^ ^ \  |
        --                       --  |^ ^ ^ ^ |w|
        ---------------------------   \m___m__|_|
   Please remember to check your submission with ./02_check !!
   Please remember to check your submission with ./02_check !!
   Please remember to check your submission with ./02_check !!
============================================================
```

Correct example



Incorrect example

3. **./02_check**