CS 6886: Systems Engineering for Deep Learning

Assignment 2

Instructions

1. This assignment is released on **9**th Mar and is due on **19**th Mar (midnight).
2. The total marks (in overall weightage for the course) is 10 marks for this assignment.
3. This assignment consists of three parts.
    a. Part A is compulsory.
    b. Part B is compulsory and is worth 10 marks.
    c. Part C is optional and is worth 2 bonus marks.
4. You are required to submit a PDF report with contents as described in each of the questions along with the code in the provided folder structure.
5. Final submission is to be done on Google Classroom.
6. Copying of code or parts of the report is disallowed.
7. You are allowed to discuss with each other and refer to sources online. You are required to clearly state any such collaboration or references in the report.
8. You are **restricted to use only AVX instructions**. Use of AVX2, SSE instructions are disallowed for this assignment. Documentation for AVX instructions can be found here.
9. Plots are preferred over tables for representing execution times. Try to normalise execution times over the baseline reference.

Abbreviations

Symbol	Parameter
N	Number of input feature maps in a batch (Batch Size)
C	Number of channels in a single input feature map
H	Height of input feature map
W	Width of input feature map
R	Height of filter weight

S	Width of filter weight
M	Number of filters
E	Height of output feature map
F	Width of output feature map
Sx	Stride along H dimension
Sy	Stride along W dimension
Px	Padding along H dimension
Py	Padding along W dimension

Part A (0 points)

Step 1 (0 points) - Machine Configuration

State the following parameters of the machine on which you will be measuring execution times for the questions given below.

- L1D Cache Size
- L2 Cache Size
- L3 Cache Size

Step 2 (0 points) - AVX intrinsics

Briefly describe all the AVX instructions you've used throughout the rest of the assignment.

Part B (10 points)

Step 1 (2.5 points) - Correctness

Implement the following building blocks commonly used in DL models using Intel AVX instructions. A separate header file (**util.h**) provides the parameters for each of the operations and each operation is to be implemented in the **util.cpp** file. Each operation, along with parameters and constraints is described below.

1. **2D Convolution**
    a. **Inputs:** Filter [M, C, R, S], Input feature map[N, C, H, W], Stride [Sx, Sy], Padding: [Px, Py]
    b. **Output:** Output feature map [N, M, E, F]

    c. Separate memory for output feature map is allocated inside the function. The function should finally return a pointer to the output feature map.
    d. This implementation is considered a baseline for the following questions to measure performance.
2. **Fully Connected Layer**
    a. **Inputs:** Filter[M, HWC], Input feature maps[N, CHW]
    b. **Output:** Output feature map[N, M]
3. **Rectified Linear Unit (ReLU) operation**
    a. **Inputs:** Input feature map[N, C, H, W] (for convolution) / [N, HWC] (for Linear)
    b. **Output:** Output feature map[N, C, H, W] (for convolution) / [N, HWC] (for Linear)
4. **2D MaxPool operation**
    a. **Inputs:** Input feature map[N, C, H, W], subsample window (Wx, Wy], stride [Sx, Sy]
    b. **Output:** Output feature map [N, C, E, F]

Step 2 (1.5 points) - Bottleneck Analysis

Using the given **alexnet.cpp**, measure the execution time for each CONV and FC layer and report layerwise results. Analytically calculate the operational intensity (ops/byte) for each layer, and support with arguments to correlate with the execution times obtained.

Step 3a (2.5 points) - Implementing different Dataflows

In systolic arrays, we studied different dataflows (weight stationary, etc.) where some data remains stationary within each PE. In this assignment, we load data from the main memory to one of the vector registers and **reuse** them. Depending on which of input or weight or output is reused, for the purpose of this assignment, they are referred to as input-stationary (IS), weight-stationary (WS) and output-stationary (OS) convolutions. Implement all three variants (OS, WS, IS) of convolutions using AVX instructions.

Step 3b (1.5 points) - Design Space Exploration

As an extension to step 2, for each convolution layer of AlexNet taken separately implement all three variants (OS, IS, WS), i.e., a total of (5 layers) x (3 variants) = 15 configurations. Report layerwise execution-times. Identify the best reuse pattern for each layer.

Step 4 (2 points) - More Exploration - Tiling

The order of accessing data from memory plays a huge role in the inference time. One design approach to influence this is *tiling,* wherein a smaller part of the input is read and processed, and then the next part is read. Modify the optimised variant of Layer 2 from Step 3b above to use tiling. Report the tiling configuration that leads to the best performance on this layer.

Part C (Bonus, 2 points) - Memory Layout

All feature and weight maps discussed so far have NCHW memory layout, by default. The task is to explore different orders of storing in memory (eg. NHWC, CNHW, etc.) and find the best memory layout for each of OS, IS and WS. Support your choice with appropriate plots.