BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI, K. K. BIRLA GOA CAMPUS, I SEMESTER 2020-2021
Operating Systems (CS F372)
Component: Online #4 Weightage: 6% [18 Marks]
Date: 08/11/2020, Time: 9:00 P.M to 11:59 P.M

Problem: Multi-threaded Convolutional Neural Network

Note:

➢ Untar the OSOnline4.tar.gz file. Rename the OSOnline4 folder to <YOUR_ID_NO>_OSOnline4. You can find [driver.c, convNet.h, convNet.c, Makefile and README] under it. You are supposed to modify [convNet.h, convNet.c and README]. You need to submit the tar.gz of this folder. Take care of validation and error checking.

EndNote

Introduction to the Problem

Artificial **N**eural **N**etworks (ANNs) are computing systems that are based on a collection of **connected units or nodes called artificial neurons**, which loosely model the neurons in a biological brain. **C**onvolutional **N**eural **N**etworks (ConvNets) are a Deep Learning strategy (derived from ANNs) that can **take in an input image**, assign importance (learnable **weights and biases**) to various aspects / objects in the image, and be able to differentiate one from the other.

Every ConvNet is made up of multiple layers. **The output of the previous layer is the input for the next layer.** The architecture of the model you will be implementing is: **CL ➔ 2D to 1D Layer ➔ FC**

Where, CL is convolutional layer with an input of a 2D square matrix and a single filter (another 2D square matrix). FC is fully-connected layer.

What you will be implementing?

1. A main function calls a function named convNet with input file name [it received through command line argument] as argument [already coded in driver.c. You need not do any modification to it].
2. convNet function [in convNet.c file] calls the getData function [code provided to you in convNet.c file] that takes inputs from the input file [sample input is given in input.txt] and stores it in corresponding data

structures [No global variable is allowed in the program. Your solution will not be considered for major portion of the evaluation if you use global variable even as pointer].

3. The input file is in following format
   <first line> convImageDim <# of Rows in the Square Matrix>
   <Next convImageDim lines> Each line consists of convImageDim Number of elements
   <Next line> convKernelDim <# of Rows in the Square Matrix>
   <Next convKernelDim lines> Each line consists of convKernelDim Number of elements
   <Next line> convStride <Stride used for the input>
   <Next line> fullyConnectedWeights <the size of the array [number of elements] needs to be calculated by using convImageDim, convKernelDim and convStride. Need to read that many elements from the line.
4. The convNet function creates a detachable thread to print the input 2D matrices. The convNet function should create an output array of CL (CLOut) in heap. The convNet function also should create joinable threads for CL. The number of threads depends on the 2D array input and the stride value [will be discussed in detail later]. These all threads should be joinable in nature. Each thread in CL will return a value to convNet function and the convNet function will update it in the corresponding location of the output array (CLOut).
5. After all the CL threads join to convNet function, it will create a joinable thread to print CLOut. During the execution of this print thread, it takes input from the user whether the user wants it to continue as joinable thread or detachable thread. Based on the user input, the thread tries to change its state. The convNet function creates a 1D array in heap (named OneDOut) with number of elements equal to the total number of elements in CLOut. The convNet function creates number of joinable threads equal to the number of rows in CLOut where each thread places the elements in that row of CLOut array to the corresponding location of OneDOut array.
6. After all the threads finish their work, the convNet function creates a thread which finds the output Y as $x_1w_1 + x_2w_2 + \ldots\ldots x_nw_n$. Where x_1, x_2 … x_n are the elements of OneDOut array and w_1, w_2 … w_n are weight [taken as input from user].
7. The convNet function returns Y to the main function. The main function (in driver.c) prints the final value of Y.

Convolutional Layer:

In this question, always assume that the inputs of Convolutional Layer are Square matrices. The stride of the layer corresponds to the size of the steps one would take before arriving at the next sub-matrix of the input. Examples:

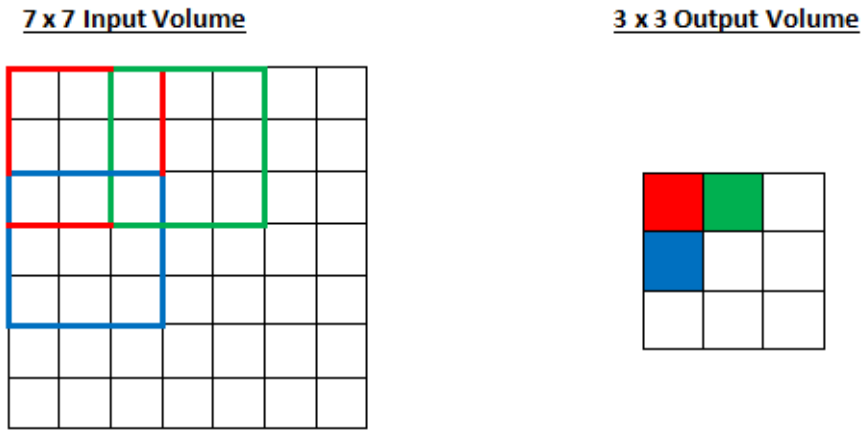


Figure 1: Channel of size 3x3 & stride of 2 to move over an input matrix

Implementation of the ConvNet:

1. Input: Your input will be an image (2D square matrix).
2. Start by placing a channel over the top left corner of the image.
3. Multiply the element on the channel with corresponding element of the image (see figure 2)
4. Add these numbers up and place it in the appropriate cell of the output layer.
5. Move to next stride and repeat till all output cells are filled.

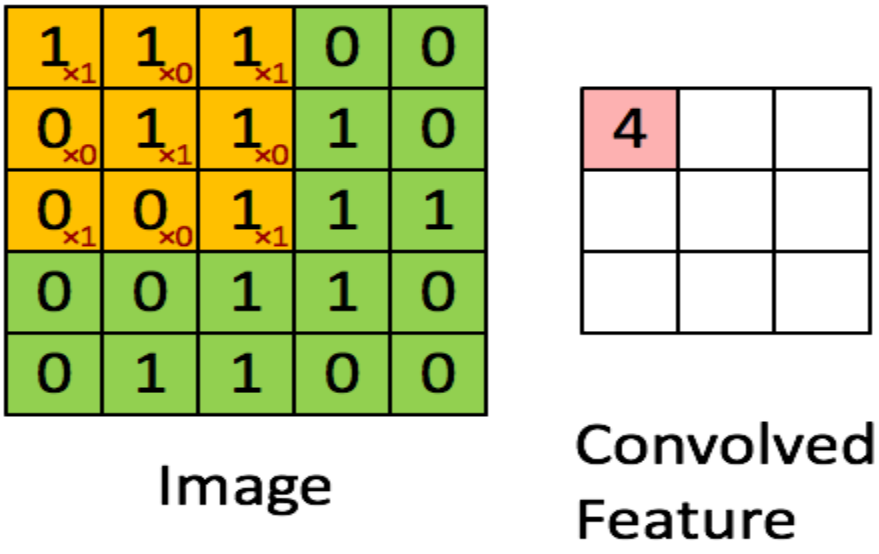
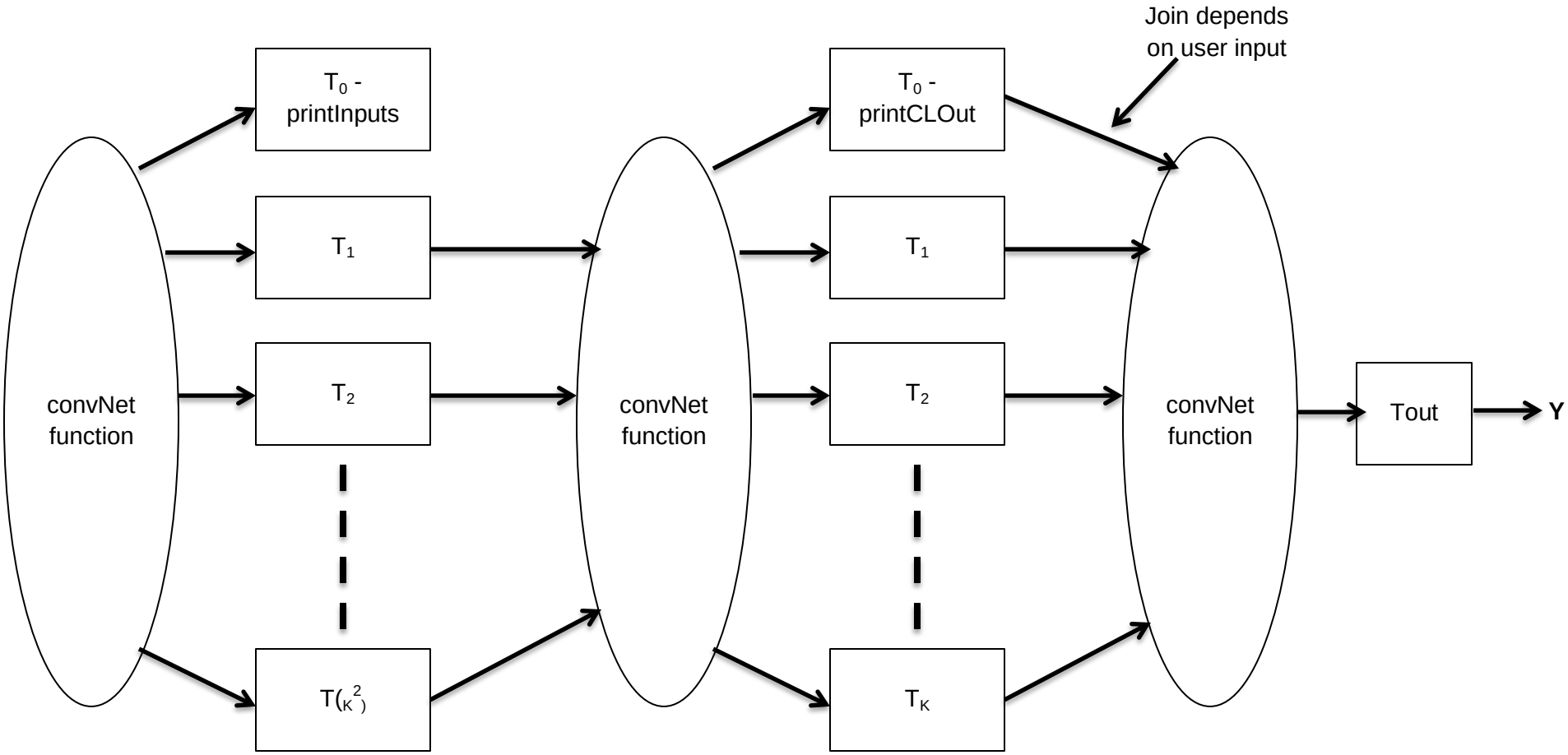


Figure 2: Convolution layer (inputs are on the left and the output on the right)



Inputs:
N X N Double dimensional Array
M X M Double dimensional Array
Stride

Output of previous stage
K X K Double dimensional Array

Output of Previous stage:
Single dimensional Array
[K^2 elements]
Input:
Single dimensional Array
[K^2 elements]

Figure 3: Thread diagram for the problem

.

Note: Maximum Parallelism has to be achieved among the threads. Don't create more threads than specified in the program. Make sure you are updating README with the final status. EndNote