# CS6023: GPU Programming (Jan 2025)

Assignment 2

Due March 2, 23:59

## 1 Problem Statement

Convolution is a crucial operation in image processing, computer vision, and neural networks. It helps extract spatial features by applying filters to input data, allowing for pattern detection, edge recognition, and more complex feature extraction. This assignment focuses on implementing a 2D convolution operation using CUDA to better understand parallel computing concepts.

In this assignment, you will implement a 2D convolution operation on an input image and filter set using CUDA. The task requires handling images and filters in a non-tensor form, specifically as stacked 2D matrices. Your objective is to perform this operation efficiently using GPU parallelism; using **memory coalescing** and **shared memory** is a **must**.

### 1.1 Definitions

- **Input Image:** An image typically has dimensions $H \times W \times C$, where $H$ is the height, $W$ is the width, and $C$ is the number of channels (e.g., RGB has 3 channels).
- **Filter Set:** Filters typically have dimensions $K \times R \times S \times C$, where $K$ is the number of filters, $R$ and $S$ are the spatial dimensions of each filter, and $C$ is the number of channels (which will be the same as the input filter channels).

### 1.2 Input Transformation

To simplify the implementation using 2D matrices, the input image and filters are transformed as follows:

1. The input image of size $H \times W \times C$ is stacked along the channel axis, resulting in a matrix of size $(H \times C) \times W$, refer to figure 1 to know how.
2. Each filter of size $R \times S \times C$ is transformed into a matrix of size $(R \times C) \times S$. The full filter set, with $K$ filters, will be a collection of $K$ such matrices.

### 1.3 Task Description

Your task is to:

- Perform a 2D convolution of the transformed input image with each filter in the set. You have to apply the convolution of the given channel with the corresponding channel in the filter and sum it over channels. You can refer to figure 2.

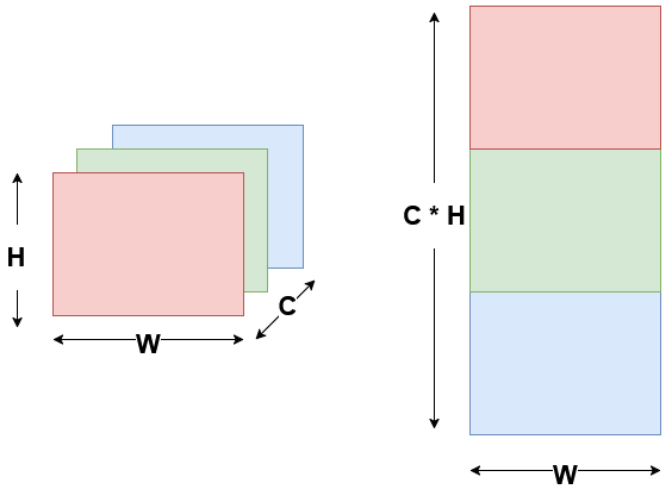


Figure 1: Input representation

- The output for each filter should have dimensions $H \times W$, as appropriate padding is applied.
- Populate the output matrices in the corresponding variables provided in the starter code.

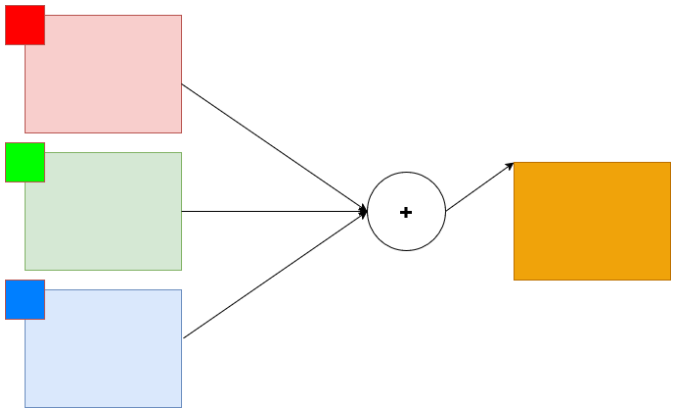


Figure 2: conv2d operation for one filter

- Ensure that your implementation is parallelized using CUDA kernel calls.
- Assume that if any cell required for the convolution operation falls outside the matrix boundaries, its value is **0** (zero-padding).
- The stride length for the convolution is **1**.

- Filters will always have **odd dimensions**, and you are expected to align the center of the filter with each corresponding matrix section during the operation.

# 2 General Guidelines

- Do not use tensor libraries or pre-built convolution functions. You must work with 2D matrices only.
- Do not modify any existing pieces of code provided in the starter template. You are expected to add necessary memory allocation and CUDA memory copy operations.
- We will time your kernel executions to verify parallelism.
- Plagiarism is a serious offense and will result in academic disciplinary actions.

# 3 Input and Output Format

## 3.1 Input Format and Output Format

The input folder consists of test case files. Each file has the following format:

- The input image matrix is stacked vertically along its channels.
- The filter matrix is stacked similarly and applied to the image using 2D convolution.
- The output matrix corresponds to the result of this operation, including padding to maintain the $H \times W$ dimensions for one filter, and in the case of multiple filters, the matrices are stacked vertically.

**Example 1 (Single Channel):**
**Image Dimensions:**

$$3 \quad 3 \quad 1$$

*Height = 3, Width = 3, Channels = 1*
**Image Matrix:**

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

**Filter Dimensions:**

$$1 \quad 3 \quad 3 \quad 1$$

*Filter Channels = 1, Filter Height = 3, Width = 3, Number of Filters = 1*
**Filter Matrix:**

$$3$$

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

**Output Matrix:**

$$\begin{matrix} 12 & 21 & 16 \\ 27 & 45 & 33 \\ 24 & 39 & 28 \end{matrix}$$

**Example 2 (Two Channels, One Output Filter):**
**Image Dimensions:**

$$\begin{matrix} 3 & 3 & 2 \end{matrix}$$

*Height = 3, Width = 3, Channels = 2*
**Image Matrix (Stacked by Channels):**

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{matrix}$$

**Filter Dimensions:**

$$\begin{matrix} 2 & 3 & 3 & 1 \end{matrix}$$

*Filter Channels = 2, Filter Height = 3, Width = 3, Number of Filters = 1*
**Filter Matrix (Stacked by Channels):**

$$\begin{matrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 2 & 0 & -1 \\ 2 & 0 & -1 \\ 2 & 0 & -1 \end{matrix}$$

**Output Matrix:**

$$\begin{matrix} -18 & 33 & 57 \\ -27 & 63 & 99 \\ -18 & 51 & 75 \end{matrix}$$

**Example 3 (Two Channels, Two Output Filters):**
**Image Dimensions:**

$$\begin{matrix} 3 & 3 & 2 \end{matrix}$$

*Height = 3, Width = 3, Channels = 2*

**Image Matrix (Stacked by Channels):**

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{matrix}$$

**Filter Dimensions:**

$$2 \quad 3 \quad 3 \quad 2$$

*Filter Channels = 2, Filter Height = 3, Width = 3, Number of Filters = 2*

**Filter Matrix (Stacked by Channels):**

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \\ 4 & 4 & 4 \\ 4 & 4 & 4 \end{matrix}$$

**Output Matrices:**

$$\begin{matrix} & 108 & 171 & 120 \\ \textit{Filter 1:} & 189 & 297 & 207 \\ & 144 & 225 & 156 \\ & 228 & 363 & 256 \\ \textit{Filter 2:} & 405 & 639 & 447 \\ & 312 & 489 & 340 \end{matrix}$$

### 3.2 Constraints

- $0 \leq$ each cell of the input $\leq 1000$
- $1 \leq H, W \leq 1024$
- $1 \leq K \times R \times S \times C \leq 4096$

# 4 Testing and Submission

- Use the provided tester program to validate your solution.
- **Penalty:** -2 points will be penalized for execution times greater than 2*average.
- Submit a zip file with roll number (e.g., `cs23m065.zip`) which contains a single `.cu` file named with your roll number (e.g., `cs23m065.cu`).
- Ensure the output matches the expected format to avoid penalties.
- **0 Marks if** the code was written sequentially/ not used shared memory.