# CS 6260 Semester Project

Exploration of Convolution

Prelimenary Design Document

Brandon Rodriguez
March 16, 2021

# Part 1

This is the “core” of my project. This is the bare minimum in order to consider the project a “success”.

***Part 1 Languages/Libraries***: Written from scratch, using C (C++?) and CUDA. If possible, I do not intend to use much additional outside code, other than basic C language libraries.

***Part 1 Objectives***: Goal is to have practical experience with implementation of Convolution. Will implement Convolution from scratch, using C (C++?) and CUDA, which will also give hands-on experience writing programs that use the GPU (up until this point, I’ve only learned theory about using the GPU, but never wrote my own code for such).

***Part 1 Expected Implementation***:

Initially, I will write the most basic version of Convolution I can. It will likely be command line, taking in the path to an image as input. Upon a valid image being provided, the program will implement a naive solution to Convolution, which will do basic math to grab the average value from nearby neighbors of each pixel. The expected output will be a new image file with these average values as the output at each pixel. The expected result should be a naive implementation of Gaussian Blur.

Once that works, I will probably also implement a similar solution using only the CPU, simply to test performance and get practical results of just how much the GPU speeds the algorithm up. Due to expected difference in efficiency, I’ll probably only test with very small input files on the CPU version, for the sake of time.

***Part 1 Dataset***: For Part 1, nearly any image will likely be sufficient, as long as the image itself is not already very blurry to begin with. For simplicity, this will likely use the same input as Part 2 (see below).

# Part 2

This is my desired endpoint for this project. This is the minimum in order to recieve a “high grade” for my project.

***Part 2 Languages/Libraries***: Written from scratch, using C (C++?) and CUDA. If possible, I do not intend to use much additional outside code, other than basic C language libraries.

***Part 2 Objectives***: Goal is to have a more in-depth examination of Convolution, via additional implementations and applications. Will likely reuse sections of code from Part 1.

***Part 2 Expected Implementation***:

Using knowledge from Part 1, I’ll create a second program that will expand my Convolution logic. This time, it will use Convolution to examine nearby neighbors of each pixel, noting any pixels that have a large change/difference in value. The expected output should be a new image file, with only these “noteworthy pixels” highlighted, and all other pixels dulled out. The expected result will hopefully be a naive implementation of Line Detection.

Once that is working, I’ll look into Fourier Transform. It’s been noted that Convolution can be more efficiently implemented with Fourier Transform. While I’ve heard Fourier Transform mentioned previously (particularly in Algorithms classes), I’ve never had a reason to look into the details of it, nor implement it myself. I have no idea how complicated it is, but assuming it’s not too elaborate, I’ll then take one or both programs (from Part1 and Part2) and rewrite a version that uses Fourier Transform. Obviously, I’ll compare results between them all.

Should Fourier Transform seem too complicated to implement in the given time constraints, I’ll instead research and document how Fourier Transform could be used to improve the algorithms. But the current hope is that I’ll fully implement it.

## ***Part 2 Dataset***:

For part 2, the input dataset is potentially much more meaningful than in Part 1. Thus for this, I'll specifically tailor images for testing.

Most likely, I'll use images I take myself (such as from a phone). I'll likely try to find a variety of images for testing with. Off the top of my head, I could use, at minimum:

- One or more images with distinct and varied objects. Such as from an indoor or business environment.
- One or more images with many similar/repeating objects. Such as a classroom, computer lab, or hallway environment from WMU.
- One or more images with all objects having large amounts of similarity and generally blending together. Such as a pile of leaves in an outside environment.

Having such variety will let me compare how my code works when detecting in a variety of different image subjects, and with varying levels of line/object distinctness.

To further test, I will likely also resize the images down and test again. Thus having a benchmark for how my code performs among both different subject matter and different resolutions.

# Part 3

This is an optional extension of the project. While the subject matter here is of interest, it is only to be completed if Parts 1 and 2 end up being much easier to implement than I expect; The semester is fast approaching an end, and I'm not confident I can complete all three parts in the remaining time.

***Part 3 Languages/Libraries***: Written in Python, with large help from 3rd party libraries. Most likely Keras/TensorFlow (but I'll need to research further to see if there are better alternatives).

***Part 3 Objectives***: Realistically in a professional/business environment, one is unlikely to implement Convolution from scratch. The goal is to have experience using Convolution in a way that is much more akin to "real world business implementations".

Thus, I'll use Python and 3rd party Neural Net libraries to build a Convolutional Network of some kind. The point here is less "writing something from scratch" and more "learning how to use existing and commonly-used libraries to create a working Convolutional Network."

I'm unsure of the end result here. It will require additional research before I can say one way or another. If I get this far, I may see about creating a more efficient Line Detection program and compare it to my Naive implementation in Part 2. Alternatively, if that seems too trivial to accomplish with third party libraries, I may instead look into creating a basic "image categorization" program.

***Part 2 Dataset***:

The dataset for this will vary based on the end goal. If doing Line Detection, I will probably just use the dataset from part 2 (see above). If creating an image categorization program, then I'll probably search for and use one of the most basic and commonly used "image categorization starter datasets" that I can find online (I know they exist, but I would need to do further research to determine which one to use).