CSEE 4840 Design Document: FPGA based Convolutional Neural Network Acceleration for Real-Time Face Detection

Yanchen Liu (yl4189), Minghui Zhao (mz2866)

Mar 31, 2022

1 Introduction

Convolutional Neural Network (CNN) is widely used in the machine learning task in the computer vision and neural language processing area. However, traditional general processing unit such as CPU can not provide ideal resource and allocate reasonably. GPU provides parallel computing by applying single instruction multiple threads on thousands of stream processor (core) and using Fast Fourier Transform (FFT) to accomplish fast convolution calculation. Although the acceleration of GPU is impressive, the rate is still limited by the instruction decoding, executing and memory sharing defined by the Von Neumann architecture. FPGA as a no instruction and no shared memory architecture device, is not limited by the rules and able to make pipeline and data paralleling at the same time. In this project, we plan to program the FPGA to accelerate a pre-trained CNN-based network for real-time facial detection and display the result on a monitor.

First, a real-time video stream will be captured by a Raspberry Pi + Camera, and sent via Ethernet to the HPS on DE1-SOC. The HPS which will then feed video stream to the CNN-based network via the HPS-FPGA high speed channel. After the FPGA finishes computation, the HPS obtains the output, then display the image frames with the bounding box on a monitor through VGA port.

2 System Block Diagram

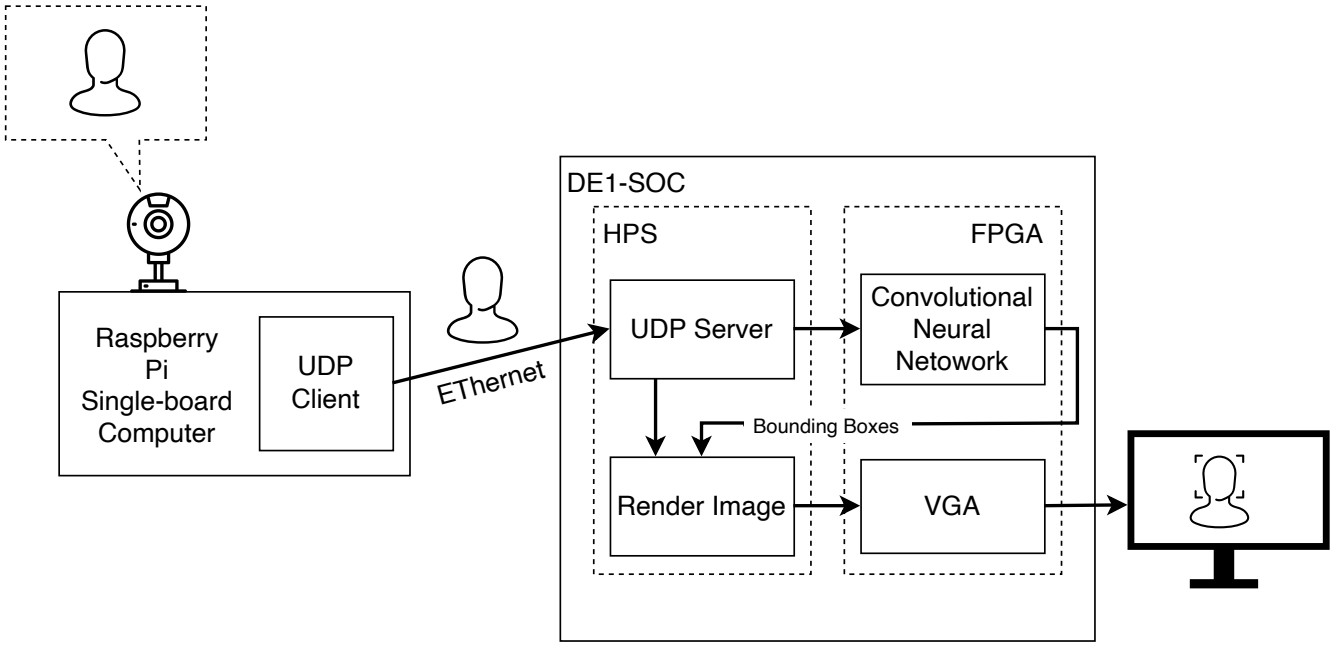

Figure 1: System Block Diagram

3 Algorithm

3.1 Fast Fourier Transform

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. Multiplexing of the same DFT results improve the time complexity from $O(N^2)$ to $O(NlogN)$. The convolution of two matrices in the space domain equals to point-wise multiplication of two matrices in the frequency domain. The pipeline of applying the algorithm is 1) zero padding the input matrix and the kernel; 2) calculate the Fourier transform of the padding matrices; 3) multiply the transferred matrices; 4) calculate the inverse Fourier transform of the result.

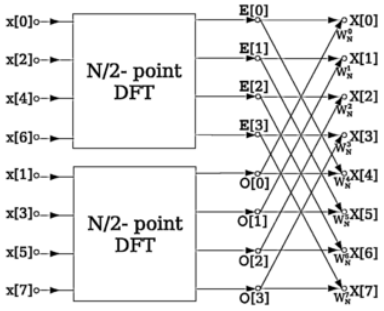

Figure 2: FFT diagram

3.2 Img2col

The core idea of img2col algorithm is first transfer the convolution operation to general matrix multiplication then applying fast matrices algorithm to accelerate the operation. The pipeline of applying the algorithm is 1) transfer input feature to input matrix; 2) transfer convolution kernel to kernel matrix; 3) multiply matrices and get the output features. There are two reasons why img2col is efficient: 1) large scale matrix multiplication is a well researched area and mature existing algorithms make the optimization close to the limit; 2) transfer the convolution kernel to kernel matrix reduce the frequent visiting of fragmented memory.

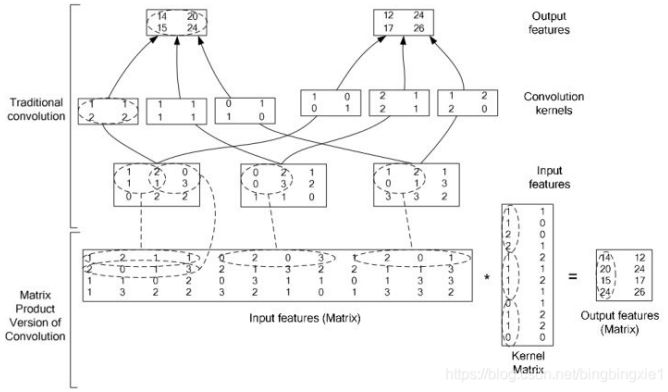

Figure 3: img2col example

4 Resource Budget

Category	Size (bytes)	# per second	# of bytes per second
Camera Image	1280x720x3 (720P)	20	55,296,000
Detection Bounding Box	4x2x10+1 (x1,y1,x2,y2) x (uint16) x (max 10 boxes)	20	1,620
Total			55,297,620

Table 1: Resource Budget

5 The Hardware/Software Interface

5.1 Camera Stream from Raspberry Pi to UDP Server on DE1 HPS

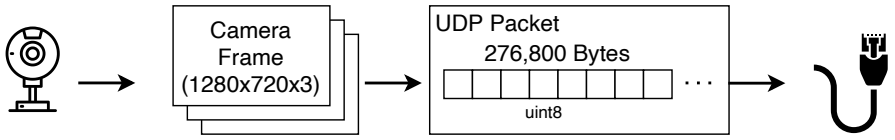

Figure 4: UDP packet containing an image frame

Camera frames are captured in 720p resolution (1280x720) at 20fps. When Raspberry Pi obtained the image, it will be in a 3D array of 1280 (pixel) x 720 (pixel) x 3 (RGB), each pixel is a 8-bit number (a byte). This array will be flattened into a linear array of length 276,800 bytes and put into a UDP packet. Finally, the packet will be sent to a UDP server running on DE1's HPS.

5.2 Camera Frame from HPS to FPGA

As the UDP server running on HPS receive the image frame, it will store into a memory shared with FPGA and tell FPGA to start computing.

5.3 FPGA Output Detected Bounding Box

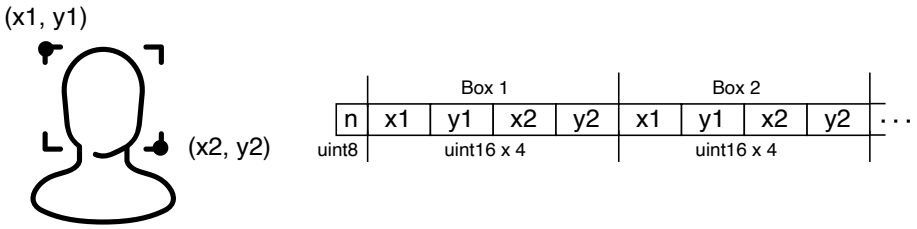

Figure 5: FPGA output bounding boxes

Once the convolutional neural network algorithm finishes the face detection, it will output the detected bounding boxes in the format above. The first byte is number of bounding boxes detected, followed by each bounding box's top left pixel index (x1 and y1), and bottom right pixel index (x2 and y2). Since the image is in 1280x720, we've chosen to use two bytes to store x1, y1, x2 or y2. We will support maximum of 10 bounding boxes.

Once the bounding box information is received, HPS will replace the respective pixels with the color of the bounding box, and send the final image to VGA driver to be displayed on an external monitor.