Pattern Detection Using Convolutional Neural Network (CNN) Kernels

CMPE 650 Spring 2017

Abhijeet Vhotkar
Masters of Science in Computer Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA
abv1@umbc.edu

***Abstract*— Convolution is a key kernel in Convolutional Neural Networks (CNN). It is also a powerful tool for object and pattern detection. In this project, we are going to detect a specific pattern/object from a pool of objects and patterns. We are provided with a pattern and an image where you need to detect that pattern and determine how many times the pattern is being matched. Therefore, we will supply our fpga with two input images and fpga will render an output of how many times a match is found**

Keywords—FPGA; CNN; Convolution Neural Networks; Pattern Detection; Sobel filter; Laplacian Filter

I. INTRODUCTION

Convolution is a key kernel in Convolutional Neural Networks (CNN). It is also a powerful tool for object and pattern detection. In this project, we are going to detect a specific pattern/object from a pool of objects and patterns. We are provided with a pattern and an image where you need to detect that pattern and determine how many times the pattern is being matched. Therefore, we will supply our FPGA with two input images and FPGA will render an output of how many times a match is found. We would be using convolution to extract features (filter) from the sample-image and pattern-image. There are several filters you may choose to apply here:

1. Sobel filter for vertical edge detection

[-1 -2 -1; 0 0 0;1 2 1]

2. Sobel filter for horizontal edge detection

[-1 0 1; -2 0 2; -1 0 1]

3. Laplacian filter [0 -1 0; -1 4 -1; 0 -1 0]

These are 3x3 filters with depth 1 (1D). Since the input is 3D you need to construct 3D filter by replication. After this feature extraction layer; we will convolve the feature extracted version of sample-image and pattern-image. a maxpool layer may be cascaded with the convolutional layer for dimensionality reduction.

At positions of match, high value of convolution result will appear. We will need to devise a scheme to detect how many times match is found. One way of doing it would be to analyze positions of maximum and near maximum spikes. We will then black out matched pattern by covering it with the black rectangle, for instance if you find 2 patterns then we should replace those patterns with three black rectangles and the final image should have only three black rectangles making rest of the image white.

Finally, we will display the image on VGA port.

II. SYSTEM DESIGN

The objective it to design a fully working hardware that can perform patter recognition using the steps explained earlier. The possible modules that need to be designed are shown in Figure II-1 and as follows:

- Module for convolution
- Module for max pooling
- Module for sorting (to detect how many time peaks are happening)
- Module that controls this sequence

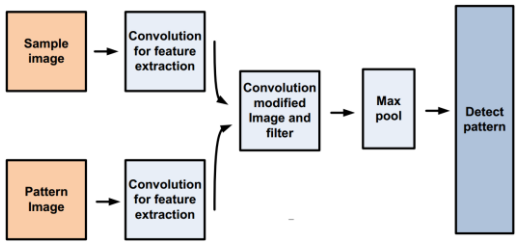

Figure II-1 Block diagram of pattern recognition by filter convolution and max pooling

A. MATLAB integration

The design is implemented in MATLAB as the convolution and maxpool modules are given. The design in MATLAB is implemented on the block diagram as shown in Figure II-1.

There are 3 sample images given for us to test and there is one pattern image which will detect the pattern in the sample image.

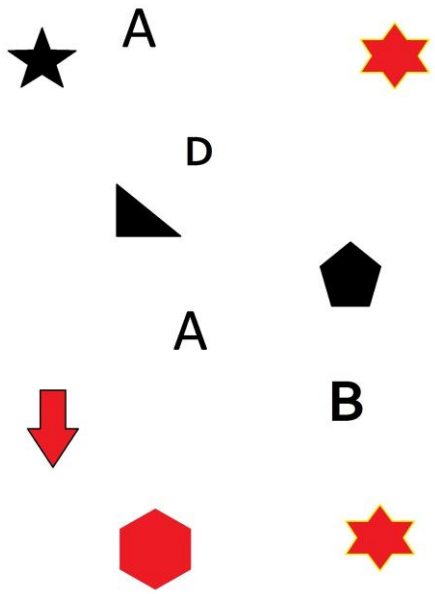

Figure II-A-1 Sample image with 2 A patterns

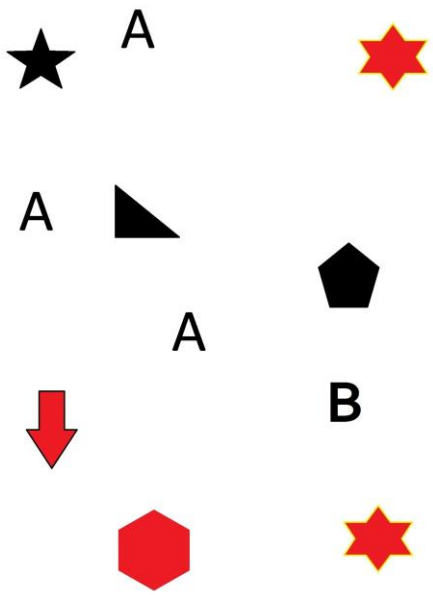

Figure II-A-II-2 Sample image with 3 A patterns

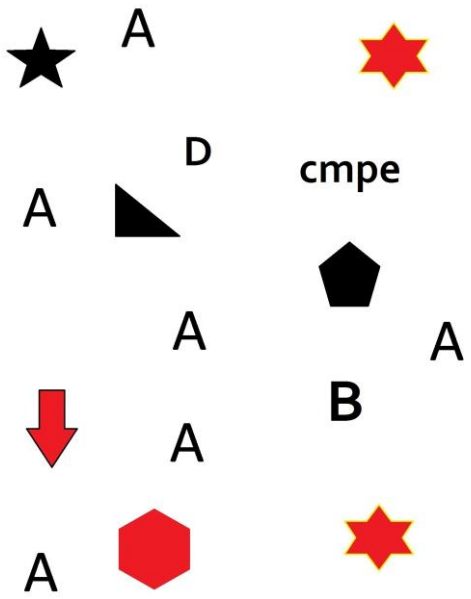

Figure II-A-4 Sample image with 6 A patterns

Figure II-A-5 Pattern image for detection

The MATLAB code structure flows as per the block diagram shown in Figure II-1 and as below:

a) Preparing filter block.
b) Making 3D filter for 3D input (rgb).
c) Convolving 3D filter for image edge detection.
d) Preparing Input image block.
e) Convolution Between Filter and Image processed.
f) MaxPooling layer cascaded with the Convolution.
g) Determining how many pattern detected.

a) Preparing Pattern image block.

The pattern image is read and converted to grayscale in this block.

b) Making 3D filter for 3D input (rgb).

Implement a 3D matrix of the filter which will be used to convolute.

c) Convolving 3D filter for image edge detection.

This block will convolve the filter and the pattern image. The block will be used to convolve the filter and sample image.

d) Preparing Input (Sample) image block.

The Sample image is read and converted to grayscale in this block.

e) Convolution Between Pattern and Image processed.

In this block, the Pattern image and Sample Image which are convoluted in the blocks before will be convoluted with each other again.

f) MaxPooling layer cascaded with the Convolution.

The Maxpool layer will then down-sample and find out the maximum values in the sub-regions binned.

g) Determining how many pattern detected.

This block will contain two loops which will run over the all the values and will keep only the values which will be greater than the threshold. This block will pin point the number of patterns detected in the sample image.

The images below show convolution of the processed image and patterns.

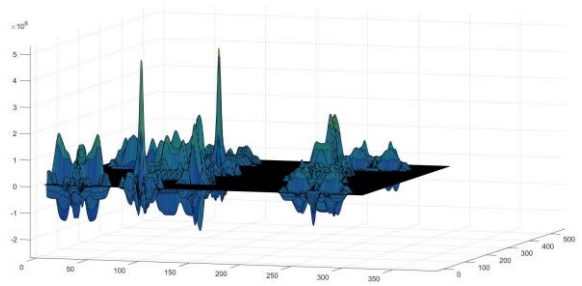

Figure II-A-6 Plot of convolution between Pattern image and Sample image for Figure-II-A-1

The Figure-II-A-6 shows the surf plot which is a function in MATLAB which creates a three-dimensional surface plot. The surf plot is for the image with 2 A patterns and it is visible that there are 2 peaks which shows that the value is maximum at the location where the pattern is detected.

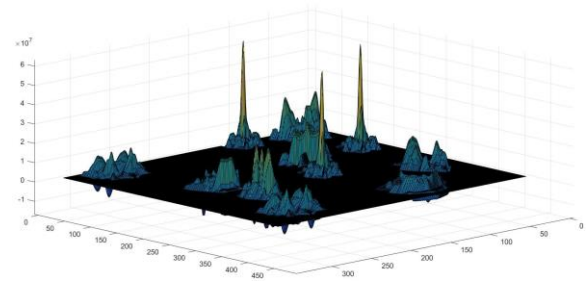

Figure II-A-7 Plot of convolution between Pattern image and Sample image for Figure-II-A-2

The Figure-II-A-7 shows the surf plot which is a function in MATLAB which creates a three-dimensional surface plot. The surf plot is for the image with 3 A patterns and it is visible that there are 3 peaks which shows that the value is maximum at the location where the pattern is detected.

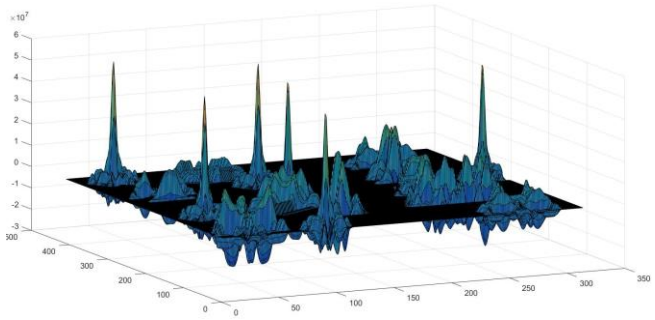

Figure II-A-8 Plot for convolution between Pattern image and Sample image for Figure-II-A-3

The Figure-II-A-8 shows the surf plot which is a function in MATLAB which creates a three-dimensional surface plot. The surf plot is for the image with 6 A patterns and it is visible that there are 6 peaks which shows that the value is maximum at the location where the pattern is detected.

The MATLAB code is for reference and visualization of how the system should work on a FPGA. The next section will discuss about the requirements for the Verilog module.

III. VERILOG IMPLEMENTATION REQUIREMENTS

The FPGA provided to us is Nexys 4 DDR Artix-7 FPGA. The key features of this device are it has a 12-bit VGA output, 128 MiB DDR2 RAM, 240 DSP slices etc.

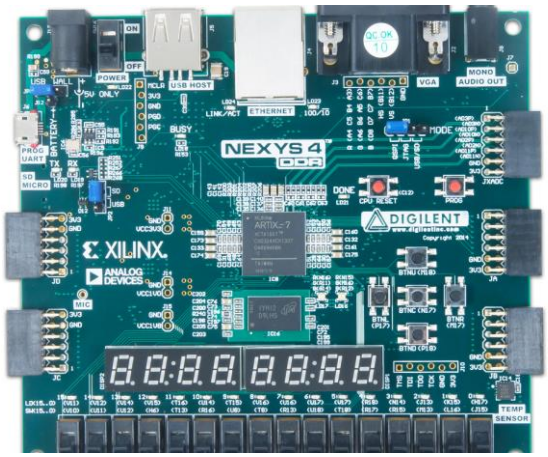

Figure III-1 Nexys 4 Artix-7 FPGA

The Figure III-1 shows the front view of the FPGA. We use the features like the push button for resetting a signal and initializing start to 1 in binary, so that the system starts executing the Verilog module.

It is impossible to count the pixel values of the image in Verilog so we provide the pixel values from MATLAB. The Verilog module uses a feature called IPcore to store the pixel values of all the calculations.

A different module for reading the values from the image is implemented. This module reads the values from the image and converts them to grayscale. The pixel values are then store to a .txt file. This text file is then read and stored in IPcores.

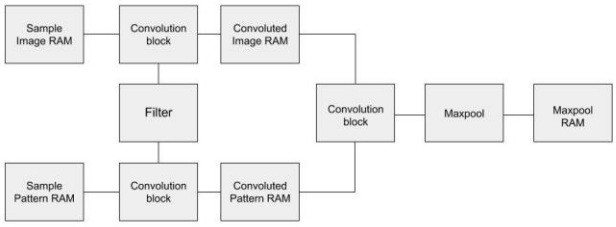

Figure III-2 Block diagram for Verilog design

There are some design constraints required to be kept in mid while designing the IPcores and the whole Verilog module.

TABLE I. DESIGN CONSTRAINTS

Input	Data width	Address width	Memory size	Dimension
Pattern	8	10	810	30 x 27
Image	8	18	149600	440 x 340
Processed pattern	11	10	700	28 x 25
Processed image	11	18	148044	438 x 338
Maxpool	13	21	5084	82 x 62

The design is implemented such that all the computation take up to 1 second.

IV. VERILOG MODULE

The Verilog module is implemented with the use of several modules keeping the vga display module in the top. Below are the modules used for implementing the design.

a) Vga top module.

This module call all the modules below which will also include its own module for calling the clock divider and horizontal and vertical synchronization module.

b) Convolution top

This module calls all the modules below except the vga module which is described above.

c) General Convolution

This module will convolute any input with the filter.

d) Filter

This is the module where which filter to be used is declared. There can be 3 types of filters as discussed above which can be declared in this module.

e) Address pointer

This module will point the address while convolution of sample image with filter, pattern image with filter and processed sample image with processed pattern image.

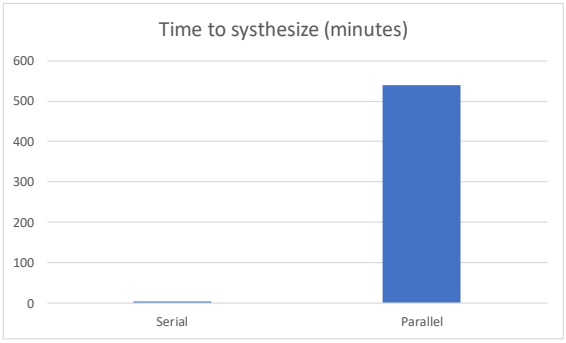

Figure IV-1 Time to synthesize the design

The design can be implemented in two different ways being one as serial or in parallel fashion. But there is a trade-off by using either of the two. The parallel implementation takes about 9 hours, i.e., 540 minutes whereas the serial implementation takes about 5 minutes to synthesize. The power in parallel used is less as compared to serial implementation. Below is the graphical comparison between serial and parallel implementation.

V. RESULTS

The Verilog design is implemented in a serial fashion. There are 3 images as inputs and one pattern image which will detect the number of patterns in the input image. Below are the results which are captured on a screen connected with the FPGA.

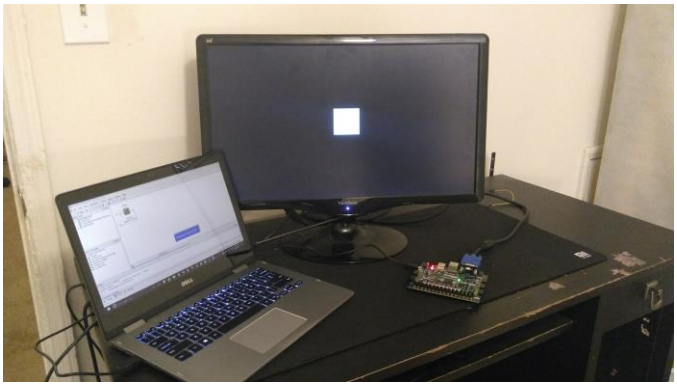

Figure V-2 Setup for testing the design

The Figure V-2 shows the setup showing the laptop connected to the FPGA via USB and the FPGAs output is further connected to the screen which also displays the detected patterns in a 62 x 82 window.

Furthermore, the results for the three images are shown below;

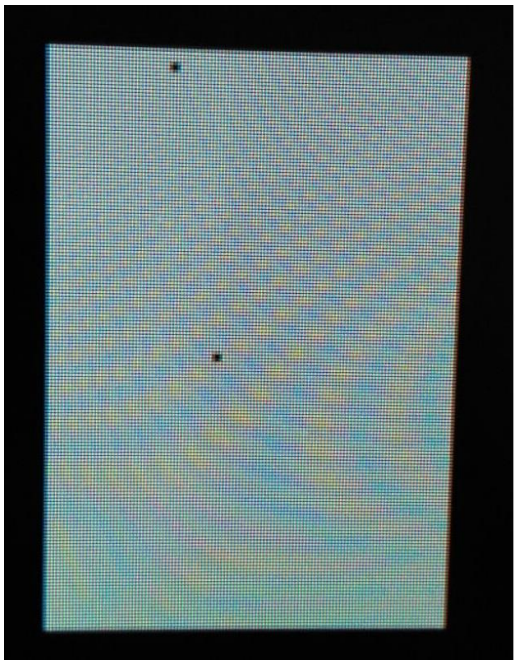

Figure V-3 Picture of image with 2 As

These images are captured through phone as the screen output from the vga cannot be captured by software. There is a little interference and color bleed due to the LCD.

The image displayed on the vga is 62 x 82 pixels. The black dot represents one pixel where the pattern is detected. It is because all other pixel values are initialized to white as the threshold matched only at certain points.

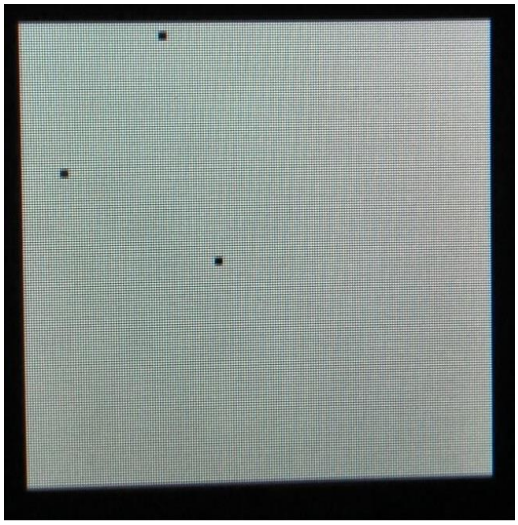

Figure V-4 Picture of image with 3 As

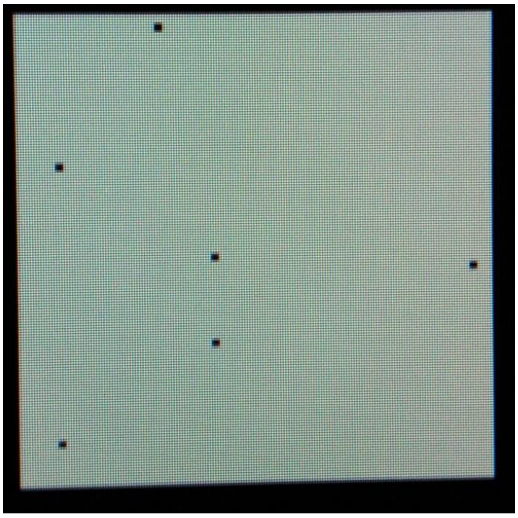

Figure V-5 Picture of image with 6 As

VI. ANAYSIS

We use the in-built features of Xilinx to analyze Timing and Power constraints. Furthermore, we will analyze the device utilization.

a) Timing Analysis

Minimum period is 14.058ns
Setup slack time is 0.942ns
Hold slack time is 0.001ns

b) Power Analysis

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.004	3	---	---
Logic	0.001	995	63400	2
Signals	0.003	1720	---	---
BRAMs	0.006	*	*	*
DSPs	0.000	8	240	3
IOs	0.011	30	210	14
Leakage	0.090			
Total	0.115			

Figure VI-1 Power Analysis for the the device

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.031	0.014	0.018
Vccaux	1.800	0.013	0.000	0.013
Vcco33	3.300	0.007	0.003	0.004
Vccbram	1.000	0.002	0.000	0.001
Vccadc	1.710	0.020	0.000	0.020

Figure VI-2 Voltage & Current requirement

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.5	25.5

Figure VI-3 Thermal properties of the device

	Total	Dynamic	Quiescent
Supply Power (W)	0.115	0.025	0.090

Figure VI-4 Power supply requirement

c) Device Utilization Summary