ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
j.inst.Korean.electr.electron.eng.Vol.21,No.4,388~396,December 2017
논문번호 17-04-09
http://dx.doi.org/10.7471/ikeee.2017.21.4.388
58

실시간 얼굴 검출을 위한 Cascade CNN의 CPU-FPGA 구조 연구

Cascade CNN with CPU-FPGA Architecture for Real-time Face Detection

남광민 *, 정용진*★

Kwang-Min Nam*, Yong-Jin Jeong*★

Abstract

Since there are many variables such as various poses, illuminations and occlusions in a face detection problem, a high performance detection system is required. Although CNN is excellent in image classification, CNN operatioin requires high-performance hardware resources. But low cost low power environments are essential for small and mobile systems. So in this paper, the CPU-FPGA integrated system is designed based on 3-stage cascade CNN architecture using small size FPGA. Adaptive Region of Interest (ROI) is applied to reduce the number of CNN operations using face information of the previous frame. We use a Field Programmable Gate Array(FPGA) to accelerate the CNN computations. The accelerator reads multiple featuremap at once on the FPGA and performs a Multiply-Accumulate (MAC) operation in parallel for convolution operation. The system is implemented on Altera Cyclone V FPGA in which ARM Cortex A-9 and on-chip SRAM are embedded. The system runs at 30FPS with HD resolution input images. The CPU-FPGA integrated system showed 8.5 times of the power efficiency compared to systems using CPU only.

요 약

얼굴 검출에는 다양한 포즈, 빛의 세기, 얼굴이 가려지는 현상 등의 많은 변수가 존재하므로, 높은 성능의 검출 시스템이 요구된다. 이에 영상 분류에 뛰어난 Convolutional Neural Network (CNN)이 적절하나, CNN의 많은 연산은 고성능 하드웨어 자원을 필요로한다. 그러나 얼굴 검출을 위한 소형, 모바일 시스템의 개발에는 저가의 저전력 환경이 필수적이고, 이를 위해 본 논문에서는 소형의 FPGA를 타겟으로, 얼굴 검출에 적절한 3-Stage Cascade CNN 구조를 기반으로하는 CPU-FPGA 통합 시스템을 설계 구현한다. 가속을 위해 알고리즘 단계에서 Adaptive Region of Interest (ROI)를 적용했으며, Adaptive ROI는 이전 프레임에 검출된 얼굴 영역 정보를 활용하여 CNN이 동작해야 할 횟수를 줄인다. CNN 연산 자체를 가속하기 위해서는 FPGA Accelerator를 이용한다. 가속기는 Bottleneck에 해당하는 Convolution 연산의 가속을 위해 FPGA 상에 다수의 FeatureMap을 한번에 읽어오고, Multiply-Accumulate (MAC) 연산을 병렬로 수행한다. 본 시스템은 Terasic사의 DE1-SoC 보드에서 ARM Cortex A-9와 Cyclone V FPGA를 이용하여 구현되었으며, HD (1280x720)급 입력영상에 대해 30FPS로 실시간 동작하였다. CPU-FPGA 통합 시스템은 CPU만을 이용한 시스템 대비 8.5배의 전력 효율성을 보였다.

Key words : Face Detection, Cascade Convolutional Neural Network (CNN), Hybrid FPGA, Acceleration, Energy Efficient

---

* Dept. of Electronics Engineering, Kwangwoon University
★ Corresponding author
E-mail: yjjeong@kw.ac.kr Tel:+82-02-940-5551
※ Acknowledgment
Manuscript received Dec. 15, 2017; revised Dec. 27, 2017 ; accepted Dec. 29, 2017

Ⅰ. 서론

1. 연구 목적

카메라, 영상 산업이 발전함에 따라 영상 내부의 정보를 이용한 연구와 어플리케이션이 증가하고 있으며, 특히 인간을 대상으로 하는 경우 더욱 활발하다. 그 중 얼굴 인식의 경우 활용도가 매우 높아 다양한 연구 방향이 제시되었는데, 현재 딥러닝 기법, 특히 CNN 기법이 우수한 성능을 발휘하고 있다. 그림 1은 딥러닝 사용 이전의 다양한 얼굴 검출 알고리즘과 CNN을 포함한 많은 종류의 딥러닝 알고리즘을 이용한 얼굴 검출의 검출률을 나타낸다[1]. True Positive Rate 최상위 성적은 모두 딥러닝 기법을 활용한 사례이다.

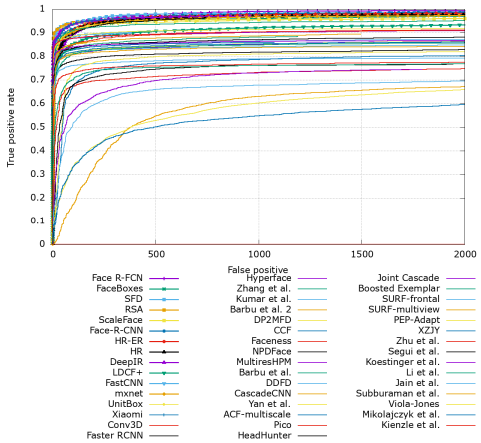


Fig. 1. Performance Cureves for Face Detection Dataset and Benchmark (FDDB)[1]

그림 1. Face Detection Dataset and Benchmark에 적용한 얼굴 검출 성능 결과 그래프[1]

이렇듯 얼굴 검출에 높은 성능을 기대할 수 있으므로 금융 본인 인증 시스템, CCTV를 이용한 용의자 탐색, 스마트 디바이스의 본인 인증, 다양한 엔터테인먼트 어플리케이션에의 활용 등 많은 분야에 접목시킬 수 있다. 다만 CNN 자체가 요구하는 연산량과 메모리 크기다는 단점이 있다. 그림 2는 CNN 연산의 핵심인 Convolutional Layer의 연산 방법을 나타낸다. 입출력 FeatureMap, 영상의 Width, Height, Weight의 크기가 모두 개별적인 인덱스가 되므로 연산량이 기하급수적으로 늘어난다. 많은 연산량의 빠른 처리를 위해 대부분의 경우 PC 환경에서 CNN 구조 연구[3]-[10]가 개발되는데, CCTV, 차량, 모바일 제품 등 실제품은 고가의 연산기를 장착하기에 어려움이 있고, 저가의 저전력 시스템이 요구된다. 본 연구는 위 조건에 입각하여, 임베디드용으로 개발된 소형 FPGA인 Intel FPGA사의 Cyclone V를 타겟으로 개발되었다. 임베디드 환경에서 많은 연산량을 고속 처리하기 위한 첫 번째 방법으로 CNN의 구조 설계를 본론의 2장에서, 두 번째 방법으로 소프트웨어 알고리즘의 측면에서 이용되는 Adaptive ROI의 설명은 3장에서, 마지막으로 하드웨어를 이용한 가속기에 대한 내용은 본론의 4장에서 설명한다.

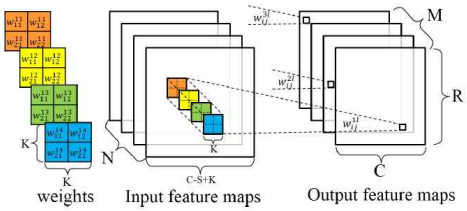


Fig. 2. Example of Convolutional Layer in CNN[2]

그림 2. CNN Convolutional Layer의 구조 예시[2]

2. 관련 연구

본 논문에 관련된 연구는 크게 Cascade CNN 구조에 관한 연구, CNN의 FPGA 가속에 관한 연구로 나뉜다. 먼저 Cascade CNN 구조에 관한 연구로, Cascade 얼굴 검출기는 Viola and Jones [3]가 Haar-Like features와 AdaBoost 분류기를 사용하면서 제안되었다. 이후로 Mathias *et al.* [4]-[6]는 변형가능한 얼굴 검출기 모델을 제안하여 높은 성능을 보였다. 그러나 과도한 연산량과 학습량을 필요로하는 단점이 있었다. 최근 CNN의 적용으로 영상 분류[7], 얼굴 인식[8]에서 큰 발전을 보였다. Li *et al.*[9]은 Cascade CNN 구조를 얼굴 검출에 적용하는 시도를 했지만, Bounding Box Calibration으로 인한 과도한 연산이 필요하다는 단점이 있었다. 개선을 위해 Zhang *et al.*[10]은 Multitask CNN을 이용하여 Multiview 얼굴 검출의 정확도를 높였지만, 초기 검출 window의 한계로 인해 성능에 한계가 있었다.

CNN의 발전과 더불어 CNN의 가속에 관한 연구도 왕성하게 진행되고 있는데, 최근 FPGA, GPU, ASIC 등 다양한 형태의 가속기 설계[11]-[13]가 제안되어 성능을 높이고 있다. 이러한 연구 중 특히 FPGA를 이용한 가속기는 성능의 우수성, 높은 에너지 효율, 빠른 개발 기간, 수정의 용이성 때문에 더욱 주목받고 있다[14]-[17].

본 논문에서는 [18]의 네트워크 구조를 기반으로 얼굴 분류와 Calibration을 통합한 형태의 CNN을 구성하고, Adaptive ROI를 적용해 성능과 속도 문제를 개선한다. 또한, Multiple FeatureMap Load, Local Memory 재사용을 통해 Memory Bandwidth를 최적화하고, 공통적으로 사용되는 커널 크기를 기준으로 Computation Engine을 설계하여 Layer 종류에 관계없이 연산을 수행토록 한다.

II. 본론

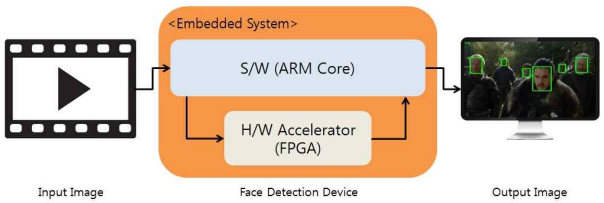


Fig. 3. Overview of Proposed System
그림 3. 제안하는 시스템의 개요

1. 시스템 개요

본 논문의 시스템은 그림 3과 같이 간략화된다. 입력 영상으로 저장된 동영상 혹은 웹캠을 통해 얻어지는 영상 프레임을 사용한다. 시스템 구현에 있어 입력 영상 크기는 HD (1280x720)급 해상도로 지정한다. 입력된 영상은 기본적으로 DE1-SoC의 ARM Core를 사용하는 소프트웨어 알고리즘을 거친다. 영상의 입출력, 다양한 영상의 전, 후처리는 ARM Core를 이용하여 실행되며, Bottleneck에 해당되는 CNN 부분은 FPGA 내에 설계, 구현된 하드웨어 가속기를 거쳐서 빠르게 실행된다. 얼굴 검출이 완료된 영상은 얼굴 검출 영역을 표시한 상태로 특정 메모리 주소에 저장되어 VGA 포트를 통해 모니터로 출력된다.

2. Cascade CNN

전체적인 Pipeline은 그림 4와 같다. 입력된 영상은 다양한 크기로 변형되어 Image Pyramid를 생성한다. Image Pyramid는 총 세 단계로 구성되는 Cascade CNN에 최초 입력되는 영상이다.

첫 번째 Stage인 Top-net은 가장 적은 연산량을 가지는 CNN으로, 개략적이지만 빠르게 입력 영상을 훑어 얼굴의 후보군을 검출한다. Top-net을 거치면서 얼굴 후보군의 대다수를 걸러낼 수 있어 연산량이 많은 Mid-net, Bot-net의 동작 횟수를 줄인다. 이는 실시간 동작에 필수적이다. Top-net을 거치면서 얼굴 후보군의 Bounding Box와 해당 Box의 Confidence, Regression 정보를 얻는다. 얻어진 Bounding Box 중 겹치는 영역이 많을 경우 Non-Maximum Suppression (NMS)을 통해 하나의 Bounding Box에 흡수되고, Regression으로 Box 좌표 위치가 보정된다.

두 번째 Stage인 Mid-net은 Top-net의 결과로 얻어진 얼굴 후보 영역을 입력으로 받아, 다시 한 번 더욱 정교하게 얼굴을 검출한다. Top-net보다 많은 연산량으로 실제 얼굴이 위치하는 영역에 근접한 결과의 Bounding Box를 계산한다. 마찬가지로 NMS를 거쳐 Box의 수를 줄이고, Regression으로 Box 좌표 위치가 보정된다.

마지막 Stage인 Bot-net은 Mid-net보다 더욱 커진 크기의 CNN으로 가장 정교하게 최종 얼굴 검출을 마친다. 다만 수행 시간이 가장 길기 때문에 Bot-net의 동작을 줄이는 것이 실시간 동작에 큰 영향을 끼친다.

CNN의 전체 구조는 그림 5와 같다. 커널의 크기는 빠른 연산을 위해 3x3, 2x2로 설계되었다. Convolutional Layer를 거친 후, ReLU 보다 정교한 PReLU를 활성 함수로 사용한다.

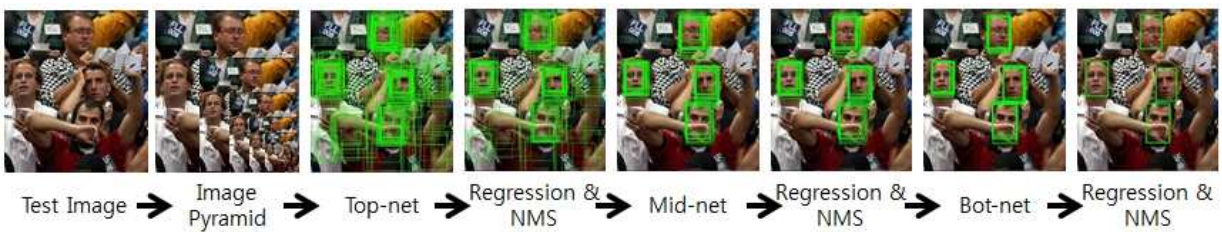


Fig. 4. Pipeline of Our Cascade CNN
그림 4. 본 시스템의 Cascade CNN 파이프라인

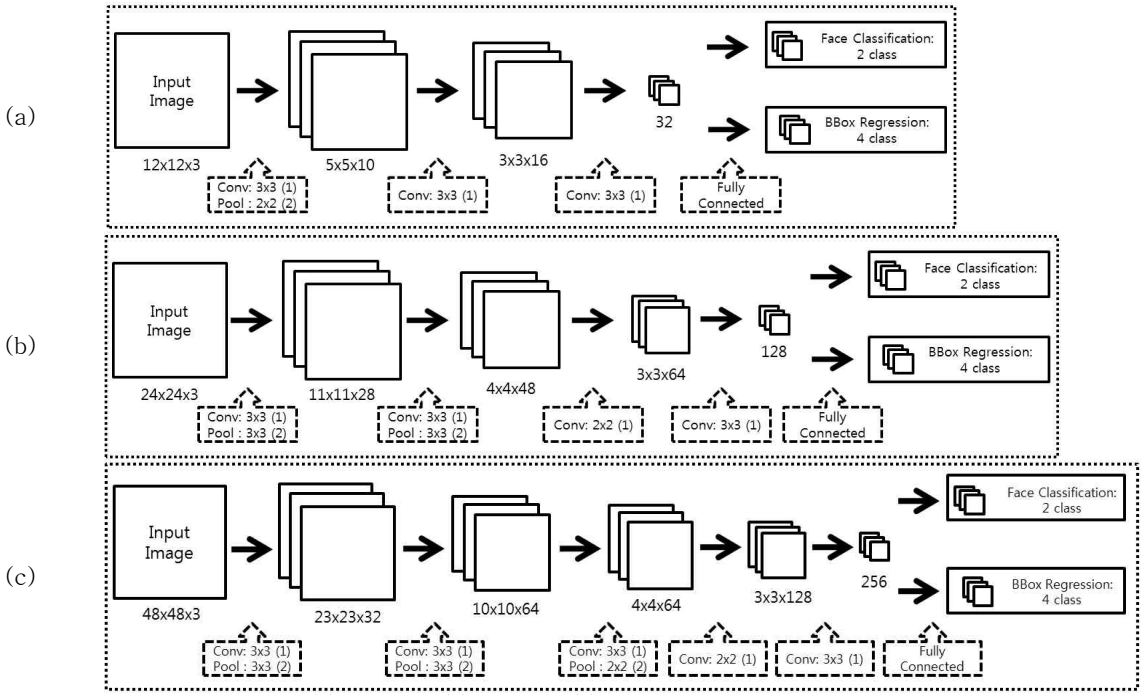


Fig. 5. Cascade CNN Architecture : (a) Top-net, (b) Mid-net, (c) Bot-net
그림 5. Cascade CNN의 구조 : (a) Top-net, (b) Mid-net, (c) Bot-net

3. Adaptive ROI

Adaptive ROI는 이전 프레임에서 검출된 얼굴 위치를 참조하여, 현재 프레임에서 CNN에 입력할 영상 ROI의 수를 감소시키기위해 적용되는 기법이다. 이전 프레임의 얼굴 검출 여부와 현재 상황에 따라 각 경우를 나누고, ROI 영역을 설정하는데 그 규칙은 표 1과 같다.

Table 1. Adaptive ROI Rule
표 1. Adaptive ROI 규칙

	First Frame	Object Detected		Object Not Detected	
Case	0	1	2	3	4
Full-ROI	O	X	X	O	X
Half-ROI	X	X	O	X	X
Obj-ROI	X	O	O	X	X
Period	-	1-4	5	1	2-3

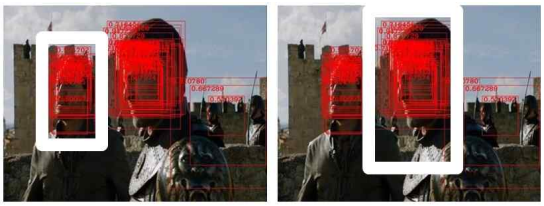
이전 프레임에서 얼굴이 검출됐을 경우, 얼굴 주변 영역만을 검출하고, 새로운 얼굴 객체의 유입이나 오검출에 대비하여 5번의 프레임 입력을 기준으로 입력영상의 절반에 해당하는 Half-ROI로 설정하여 검출한다. Half-ROI의 위치는 x축을 기준으로 3분할된 모양으로 나누어 변경된다. 이전 프레임에서 얼굴이 검출되지 않았을 경우, 입력 영상 전체를 대상으로 얼굴을 검출하지만, 매 프레임마다 얼굴을 검출하는 것은 계산시간을 많이 요구하므로 3번의 프레임 입력을 기준으로 검출을 수행한다. 표 2의 Adaptive ROI 동작 예시는 이전 프레임에서의 얼굴 검출여부와 프레임 입력 기준으로 Adaptive ROI가 어떻게 작동하는지 나타낸다.

Table 2. Adaptive ROI Example
표 2. Adaptive ROI 예시

	0	1	2	3	4	5	6	7	8	9	10	11
Object Detected	-	O	O	O	O	O	X	X	X	X	O	O
ROI	Full	Obj	Obj	Obj	Obj	Half	Full	No	No	Full	Obj	Obj
Case0	0	1	1	1	1	2	3	4	4	3	1	1

Adaptive ROI를 사용할 경우 그림 6의 흰색 네모 박스 내부의 영역 만큼만을 CNN의 입력으로 사용한다. 영상 전체를 입력으로 사용하는 경우보다 매우 적은 양임을 확인할 수 있다. 그 효율성은 그림 7의 그래프에서 비교된다.

Object ROI Size Ratio의 값 1.0은 이전 프레임에서 검출된 얼굴 영역의 Width, Height의 100%를 ROI 영역에 추가로 더해서 ROI 크기를 결정함을 나타낸다. 값이 0.5라면 50%, 1.5라면 150%



(a) (b)

Fig. 6. Sample Image for Applying Adaptive ROI
그림 6. Adaptive ROI가 적용된 영상 예시 :
(a) 얼굴 객체 1, (b) 얼굴 객체 2

의 Width, Height가 ROI 영역에 더해진다. CNN 동작 횟수는 Object ROI Size Ratio가 줄어들수록 증가하지만, 얼굴 검출률이 감소하므로 높은 검출률을 보이면서 동작 효율이 좋은 1.1~1.3 주위의 값이 적절하다.

Fig. 7. Detection Rate & Operation Ratio depends on Adaptive ROI's Parameter
그림 7. Adaptive ROI 인자 변화에 따른 검출률과 연산량 변화

4. FPGA Accelerator

CNN 자체의 연산 속도를 높이기 위해서는 FPGA 상에 하드웨어 가속기가 이용된다. 그림 8은 CNN 하드웨어 가속기의 Block Diagram으로 가속기의 작동 방식를 보여준다. 가속기는 연산을 위해 Global Memory로부터 입력 FeatureMap과 Parameter를 Local Memory로 읽어들인다. 현재 사용되는 CNN, Layer 종류에 알맞은 연산(Convolution, Pooling, Fully Connected)을 하도록 컨트롤러를 통해 제어하며, 가속기 내부의 주소 연산기를 통해서 Global Memory의 어느 주소에서 데이터를 읽어오고, 쓸지 판단한다. 현재 입력 FeatureMap이 모두 연산될 때까지 가속기는 같은 동작을 반복하며, Global Memory에 출력 FeatureMap을 저장한다.

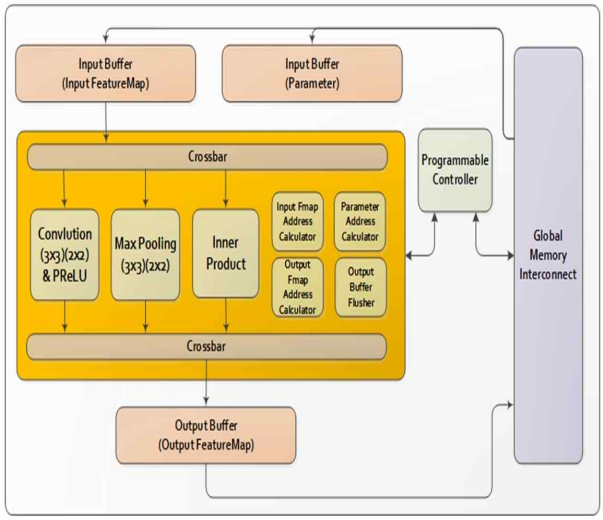


Fig. 8. CNN Accelerator Block Diagram
그림 8. CNN 가속기의 블록 다이어그램

가. Computation & Memory Optimization

하드웨어 가속기의 연산 성능을 증가시키기위해 모든 종류의 CNN 입력 FeatureMap들을 공통적으로 저장할 수 있는 버퍼 크기를 찾는다. 그림 9는 Top, Mid, Bot-net의 입력 FeatureMap이 최대로 입력되도록 3, 6, 12, 24, 48 크기를 기준으로 저장 공간을 나눈 버퍼이다. 3 이하 크기의 입력 FeatureMap은 3x3 버퍼에, 12 이상 24 이사 크기의 입력 FeatureMap은 24x24 버퍼에 저장된다.

Fig. 9. Buffer for Multiple Input FeatureMap Loading
그림 9. 입력 FeatureMap의 다중 읽기를 위한 Buffer

Top-net의 경우 그림 10의 형태로 입력 FeatureMap을 읽어 저장한다. 하드웨어 자원에 적절한 버퍼의 크기를 정했다면,FeatureMap을 읽어 저장한다. 하드웨어 자원에 적절한 버퍼의 크기를 정했다면, 그 버퍼 크기 내에서 최대 개수의 입력 FeatureMap을 읽어와 동시에 연산할 수 있으므로

Memory Bandwidth를 낮춤과 동시에, 오버헤드 감소로 연산 속도 또한 높일 수 있다. CNN, Layer

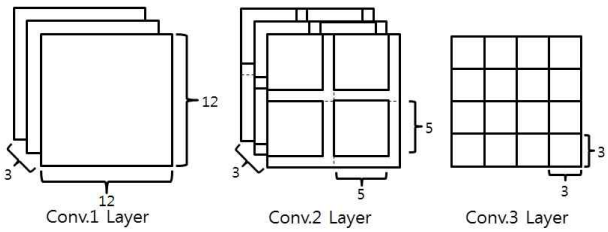


Fig. 10. Input FeatureMap Multiple Loading in Top-net
그림 10. Top-net에서의 다중 Input FeatureMap 로드

에 관계없이 공통적으로 사용가능한 커널의 병렬연산 지점을 Pseudo 코드를 활용하여 파악한다. 그림 11은 기본적인 CNN의 Convolution 반복 연산을 나타내는 코드이다. 그림 2를 참고하여 입출력 FeatureMap의 Depth, Width, Height, 커널 크기를 확인한다.

```
for(row=0; row<R; row++) {
 for(col=0; col<C; col++) {
  for(out_d=0; out_d<M; out_d++) {
   for(in_d=0; in_d<N; in_d++) {
    for(m_row=0; m_row<K; m_row++) {
     for(m_col=0; m_col<K; m_col++) {
        output_fmap[out_d][row][col] +=
          weight[out_d][in_d][m_row][m_col] *
          input_fmap[in_d][S*row+m_row][S*col+m_col];
}}}}}}
```

Fig. 11. Pseudo Code of Basic Convolution Operation
그림 11. 기초적인 Convolution 연산의 슈도 코드

그림 12는 MAC 연산의 병렬처리 구간을 결정한 코드이다. Multiple 입력 FeatureMap을 처리하므로 반복문(in_fm)이 한 단계 추가되었다. 버퍼

```
for(in_d=0; in_d<N; in_d++) {
 for(in_fm=0; in_fm<D; in_fm++) {
  for(out_d=0; out_d<M; out_d++) {
   for(row=0; row<R; row++) {
    for(col=0; col<C; col++) {
     #pragma unroll
     for(m_row=0; m_row<K; m_row++) {
      #pragma unroll
      for(m_col=0; m_col<K; m_col++) {
         output_fmap[out_d][row][col] +=
           weight[in_d][in_fm][out_d][m_row][m_col] *
           input_fmap[in_d][in_fm][m_row][m_col];
}}}}}}}
```

Fig. 12. Proposed Convolution Accelerator Structure
그림 12. 제안하는 Convolution 가속기의 구조

에 입력된 입력 FeatureMap의 수만큼 동작을 반복한다. 입력 FeatureMap을 기준으로 연산을 수행하므로 반복문의 차례 또한 입력 FeatureMap - 출력 FeatureMap이 우선되도록 변경한다. 커널의 크기가 3x3 혹은 2x2로 선택 가능하도록 병렬 처리한다. Local Memory 접근을 줄이기 위해 병렬 연산된 결과 값은 매 경우 Local Memory에 접근하지 않고, 하나의 레지스터에 저장되어 병렬 연산이 끝난뒤, Local Memory에 한번만 접근한다. 레지스터는 그림 13의 conv_result_temp에 해당하고, Local Memory는 output_fmap에 해당된다.

```
for(in_d=0; in_d<N; in_d++) {
 for(in_fm=0; in_fm<D; in_fm++) {
  for(out_d=0; out_d<M; out_d++) {
   for(row=0; row<R; row++) {
    for(col=0; col<C; col++) {
     #pragma unroll
     for(m_row=0; m_row<K; m_row++) {
      #pragma unroll
      for(m_col=0; m_col<K; m_col++) {
         conv_result_temp +=
           weight[in_d][in_fm][out_d][m_row][m_col] *
           input_fmap[in_d][in_fm][m_row][m_col];
     }}
     output_fmap[out_d][row][col] += conv_result_temp
}}}}}
```

Fig. 13. Local Memory Access Reduction Convolution Accelerator Structure
그림 13. Local Memory 접근을 줄이는 Convolution 가속기 구조

나. Implementation Details

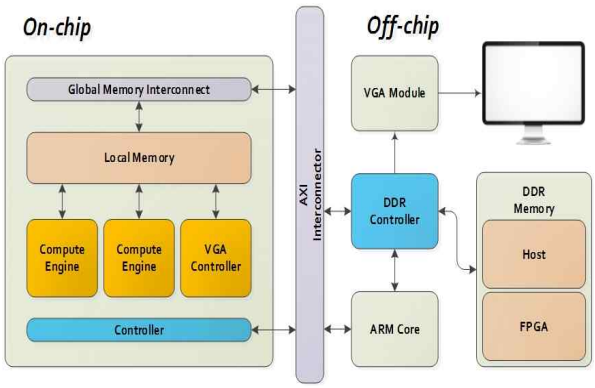


Fig. 14. Implementation Overview
그림 14. 시스템 구현 개요

임베디드 환경에서 구현된 전체 시스템 구조는 그림 14와 같다. On-Chip 영역에 CNN 가속기가 구현되었고, Off-Chip 영역에서 CNN을 제외한 나머지 소프트웨어 알고리즘이 동작하며 VGA 출력 또한 이루어진다. 얼굴 검출 시스템 작동에 앞서

FPGA, ARM Core에서 모두 접근할 수 있는 DDR 영역에 CNN 동작을 위한 다양한 Parameters(Weight, Bias, PReLU)가 저장된다. 또한 얼굴 검출에 사용될 동영상을 불러온다. 얼굴 검출이 시작되면 ARM Core가 영상을 입력받고, 영상 전처리 과정을 진행하며 CNN 하드웨어 가속기에 데이터가 입력되도록 ROI를 결정한다. Global Memory의 특정 주소에 입력된 영상은 AXI 버스를 거쳐 순차적으로 CNN 가속기 내부의 Local Memory에 저장된다. CNN, Layer의 종류에 맞게 그림 14의 Compute Engine이 결정되고, Convolution, Pooling, Fully Connected 과정이 선택적으로 가속기 내부에서 동작한다. 구현된 가속기 내부 변수는 ARM Core에 의해 제어된다.

가속기에 입출력될 FeatureMap과 Parameter는 가속기 내부의 메모리 주소 모듈이 계산한다. 연산이 완료된 데이터는 출력을 위한 Global Memory의 특정 주소에 저장된다. 출력할 영상을 준비하는 과정에서 FPGA에 구현된 VGA 컨트롤러가 VGA 출력 모듈에 영상의 동기 신호를 전달 해준다.

III. 실험

1. 실험 환경

제시한 얼굴 검출 시스템의 검증을 위해 DE1-SoC 임베디드 보드 (Dual-Core ARM Cortex A9, 1GB DDR3 SDRAM, Cyclone V 5CSEMA5F31)가 이용되었다. Intel FPGA for OpenCL 툴을 이용해 C/C++ 언어로 하드웨어를 개발하였다.

CNN의 학습을 위해 Face Detection Dataset and Benchmark (FDDB)[1] (2,845장-5,171얼굴) 데이터셋이 이용되었으며, 검증을 위해 FDDB, HD급 해상도의 동영상이 이용되었다.

2. 실험 결과

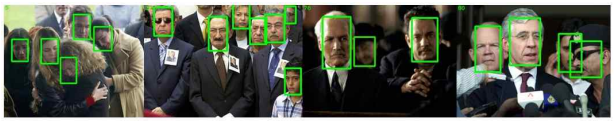


Fig. 16. Our Face Detection Result Images using FDDB Dataset
그림 16. FDDB 데이터를 이용한 얼굴 검출 결과 영상

표 3은 얼굴 검출을 위해 사용된 CPU, 가속기 종류 별 CNN의 동작 시간, FPS, 전력 소모량을 나타낸다. 임베디드 환경에서 구현된 CPU-FPGA 통합 시스템은 30 FPS를 나타내며 실시간 얼굴 검출을 수행한다. 같은 임베디드 환경에서 CPU만을 사용한 경우보다 9배 이상 가속되었다.

전력 소모량은 임베디드 환경에 구현할 경우 PC에 비해 10여 배 이하로 전력 소모량이 줄어든다. 임베디드 환경에서 CPU만을 사용하면 전력 사용량은 적지만, 낮은 동작 속도로 인해 에너지 효율은 높지 못하다. 그러나 CPU-FPGA 통합 시스템은 낮은 전력 소모량과 빠른 동작 속도를 보장하여, 전력 대비 연산 효율성이 PC의 10.5배, CPU만을 이용한 임베디드 시스템의 8.3배로 가장 높다.

Table 3. 동작 시간과 전력 소모
표 3. Process Time and Power Consumption

	CPU	Accelerator	Top-net (12x12)	Mid-net (24x24)	Bot-net (48x48)	FPS	Power	Power/FPS	Efficiency (W/FPS)
Embedded (DE1-SoC)	ARM Cortex A-9	-	0.3 ms	10 ms	54 ms	3.11	7.08 W	2.28 W	x1
PC	Intel-i5 4460	Nvidia GTX 750 GPU	0.02 ms	0.76 ms	5.71 ms	34.81	98.66 W	2.83 W	x0.8
Embedded (DE1-SoC)	ARM Cortex A-9	Cyclone V FPGA	0.027 ms	0.85 ms	7.04 ms	28.56	7.79 W	0.27 W	x8.3

Table 4. FPGA Resource Usage
표 4. FPGA 자원 사용량

Resource	LUT	FF	BRAM	DSP
Usage	26,939	53,886	417	28
Available	32,070	128,300	496	87
Percentage	84%	42%	84%	32%

FPGA 가속기를 구성하는데 사용된 자원의 양을 표 4에서 볼 수 있다. 최대한 많은 양의 데이터를 최소한의 Global Memory 접근으로 읽고, 쓰기 위해 BRAM(Local Memory)의 사용량이 높게 나타난다. Convolution, Pooling, Fully Connected 모두 가속기에 설계, 구현되어서 LUT 사용량이 다소 높은 편이다.

Ⅳ 결론

본 논문에서는 저전력 임베디드 환경에서 실시간 동작하는 얼굴 검출 용 Cascade CNN의

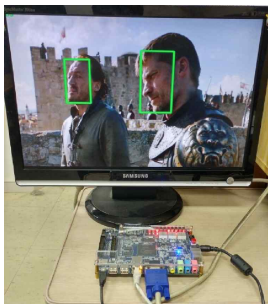


Fig. 18. Prototype of Proposed System
그림 18. 제안하는 시스템의 프로토타입

CPU-FPGA 구조에 대해 분석하였다. 검증 결과 FPGA를 이용한 가속을 통해 HD (1280x720)급 영상을 입력으로 받아 실시간 동작하였으며, GPU를 활용한 PC 대비 1/10 수준의 전력을 소모하며 같은 동작을 수행했다. 따라서 CCTV, 모바일 환경 등 풍부한 전력, 하드웨어 자원을 쓸 수 없는 상황에서 CPU-FPGA 통합 시스템의 사용이 얼굴 검출에 적합함을 확인했다. 다만 영상 내부의 다양한 환경 변수로 인해, 영상의 종류에 따라 동작 시간과 검출률에 변화가 있는데, 검출 대상의 크기와 개수에 변화에 강인한 얼굴 검출 시스템의 추가적인 연구가 필요하다.

References

[1] V. Jain and E. G. Learned-Miller, “FDDB: A benchmark for face detection in unconstrained settings,” Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. MCS-2010-009, 2010.

[2] C. Zhang, “Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks,” *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161-170, 2015.
DOI:10.1145/2684746.2689060

[3] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, May. 2004.
DOI:10.1023/B:VISI.0000013087.49260.fb

[4] M.Mathias, R. Benenson, M. Pedersoli and L. Van Gool, “Face detection without bells and whistles,” *European Conference on Computer Vision*, 2014, pp. 720-735.
DOI:10.1007/978-3-319-10593-2_47

[5] J. Yan, Z. Lei, L. Wen, and S. Li, “The fastest deformable part model for object detection,” *in IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2497-2504.

[6] X. Zhu and D. Ramanan, “Face detection, pose estimation, and landmark localization in the wild,” *in IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2879-2886.
DOI: 10.1109/CVPR.2012.6248014

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *in Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097-1105.

[8] Y. Sun, Y. Chen, X.Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” *in Adv. Neural Inf. Process. Syst.*, 2014, pp. 1988-1996.

[9] H. Li, Z. Lin, X. Shen, J. Brandt, and G.

Hua, "A convolutional neural network cascade for face detection," *in IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5325-5334.

[10] C. Zhang and Z. Zhang, "Improving multiview face detection with multitask deep convolutional neural networks," *in IEEE Winter Conf. Appl. Comput. Vis.*, 2014, pp. 1036-1041.
DOI: 10.1109/WACV.2014.6835990

[11] S. Chakradhar, M. Sankaradas, V. Jakkula and S. Cadambi, "A dynamically congurable coprocessor for convolutional neural networks," *In ACM SIGARCH Computer Architecture News*, vol.38, no.3, pp.247-257, ACM, Jun.2010.
DOI:10.1145/1816038.1815993

[12] T. Chen and O. Temam. Diannao, "A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGPLAN Not.*, vol,49. no.4, pp.269-284, Feb. 2014.
DOI:10.1145/2644865.2541967

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classication with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 25 pp.1097-1105. Curran Associates, Inc., 2012.

[14] D. Aysegul, J. Jonghoon, G. Vinayak, K. Bharadwaj, C. Alfredo, M. Berin and C. Eugenio, "Accelerating deep neural networks on mobile processor with embedded programmable logic," *In NIPS 2013. IEEE*, 2013.

[15] S. Cadambi, A. Majumdar, M. Becchi, S. Chakradhar and H. P. Graf, "A programmable parallel accelerator for learning and classication," *In Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pp.273-284. ACM, 2010.
DOI:10.1145/1854273.1854309

[16] C. Farabet, C. Poulet, J. Y. Han and Y. LeCun, "Anfpga-based processor for onvolutional networks," *In Field Programmable Logic and Applications FPL 2009. International Conference*, IEEE, 2009. pp. 32-37.
DOI: 10.1109/FPL.2009.5272559

[17] M. Peemen, A. A. Setio, B. Mesman and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," *In Computer Design (ICCD), 2013 IEEE 31st International Conference*, IEEE, 2013. pp.13-19.
DOI: 10.1109/ICCD.2013.6657019

[18] Kaipeng Zhang, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters, 2016*, vol.23, no.10, pp. 1499 - 1503, Oct. 2016.
DOI: 10.1109/LSP.2016.2603342

BIOGRAPHY

Kwang-Min Nam (Student Member)

2016 : BS degree in Electronics and Communications Engineering, Kwangwoon University.
2016~ : Course of MS in Electronics and Communications Engineering, Kwangwoon University.

Yong-Jin Jeong (Member)

1983 : BS degree in Control and Instrumentation Engineering, Seoul National University.
1995 : MS, PhD degree in Electronics and Computer Engineering, University of Massachusetts, Amherst
1983~1989 : Research Engineer, ETRI
1995~1999 : Chief Researcher, Samsung Electronics
1999~current : Professor, Dept. of Electronics and Communications Engineering, Kwangwoon Univ.