Final Project: Deep Learning

Tae-Hyun Oh
Associate Professor
Dept. Electrical Engineering
POSTECH, Korea

Slides by Youngjoo Lee

EPIC LAB

Background

Background

딥러닝 연산 설명

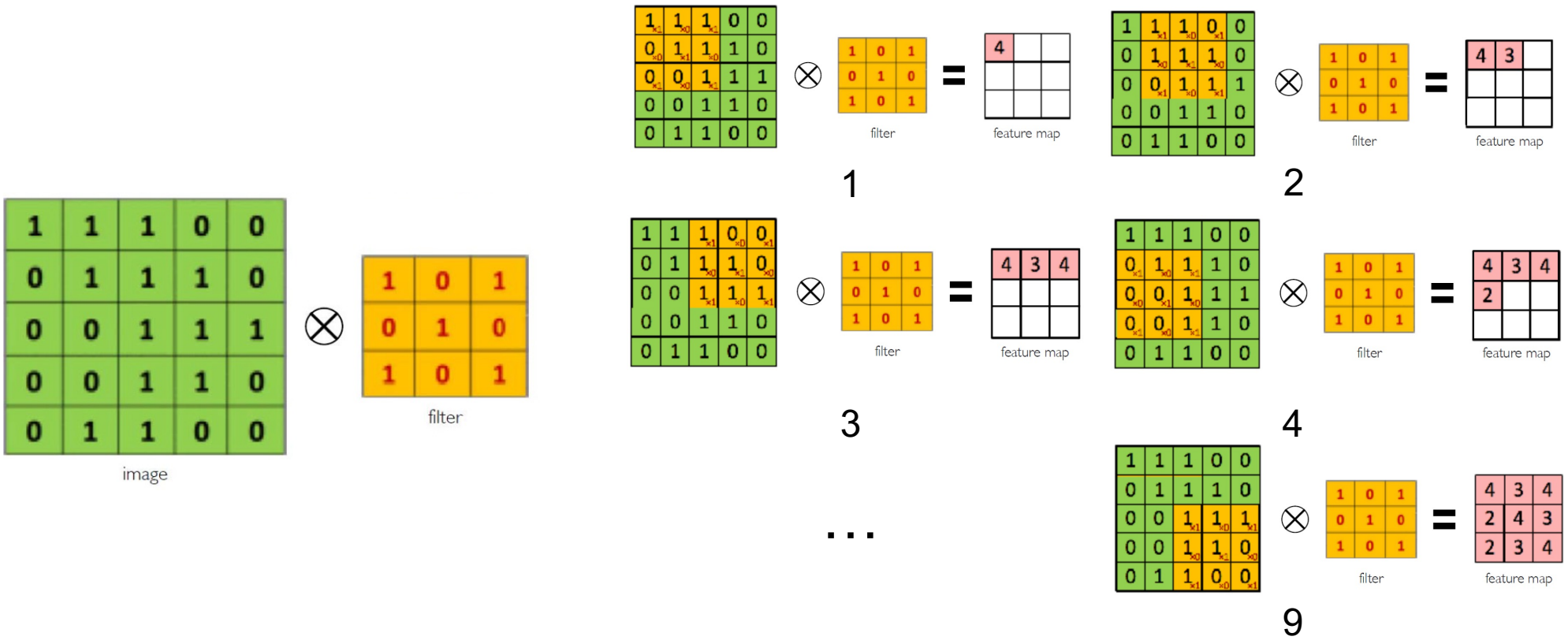
- ✓ Convolution(Conv)
- ✓ Zero padding
- ✓ ReLU
- ✓ Fully connected (FC)
- ✓ Flatten to 1D array
- ✓ CAM (Class Activation Map)

Background

2D Convolution (in Neural Networks, same with 2D Correlation)

✓ 특정 크기의 filter가 image를 slide하면서 연산을 진행

- 5×5 이미지와, 3×3 필터에 대한 연산 예시 (연산자 ⊗ 은 내적을 의미)

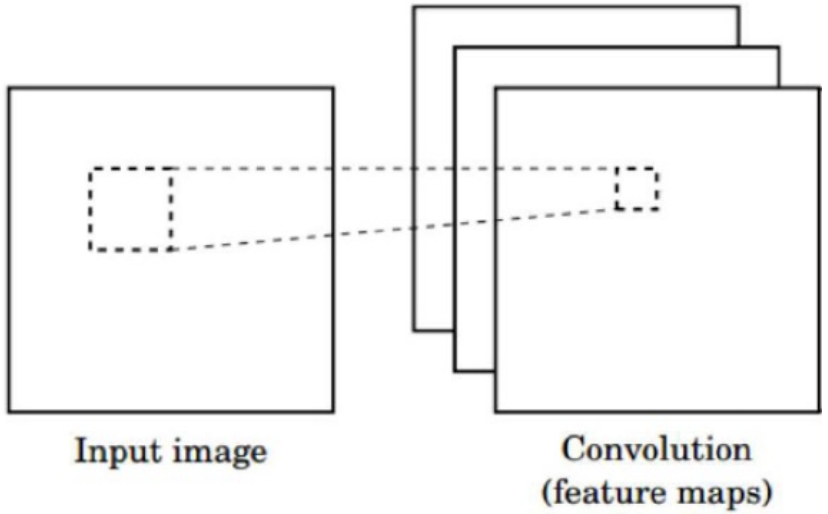


Background

Convolution

✓ 특정 크기의 filter가 image를 slide하면서 연산을 진행

- Filter가 여러 개인 경우 output 의 채널의 수가 늘어남
  - N개의 filter로 연산을 하는 경우 output의 차원은 $W \times H \times N$이 됨.
  - 앞에서는 1개의 filter로 연산을 하여 output은 $W \times H \times 1$이 되었음.

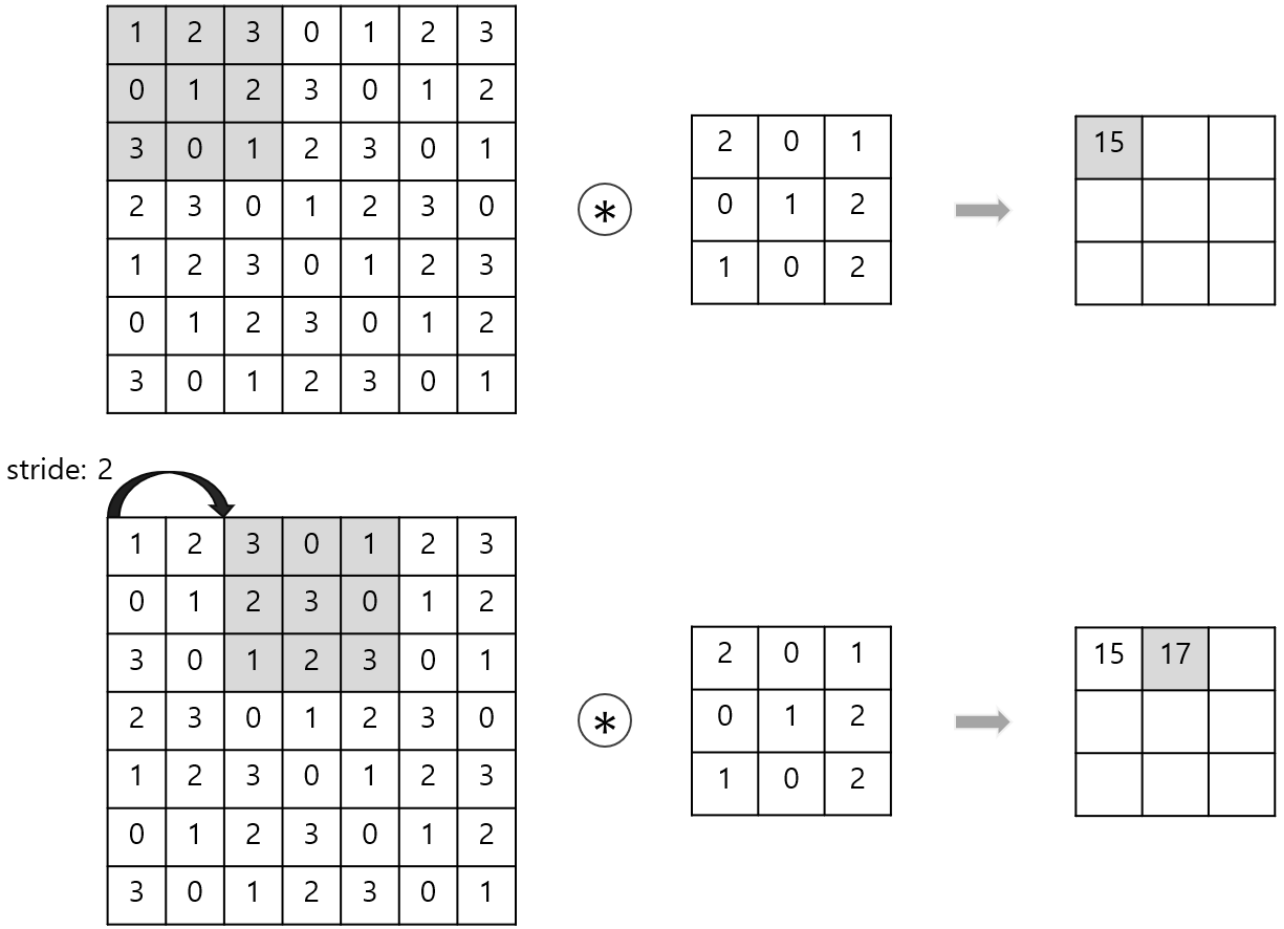


Background

Stride

✓ Filter를 image에 적용하는 간격

- Stride = 2인 경우, Filter는 2 Pixel 씩 이동하여 convolution을 계산



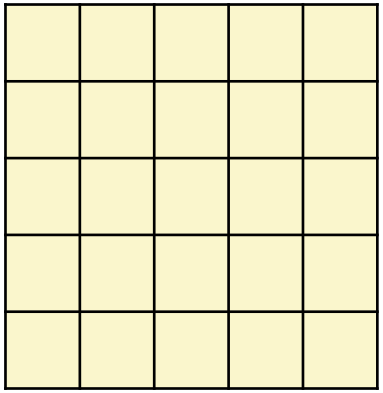
Background

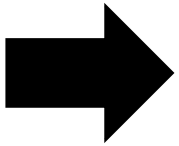
Zero padding

✓ 입력 image의 주변에 값이 0인 pixel을 붙여 입력의 크기를 늘림

- Padding size=1인 경우 아래 그림과 같이 값이 0인 pixel을 입력의 4방향으로 1pixel씩 추가
- 각 channel마다 동일하게 padding을 하여, Conv. 연산이 Border에서도 골고루 적용될 수 있도록 입력 사이즈를 맞춰줌.

5x5





7x7

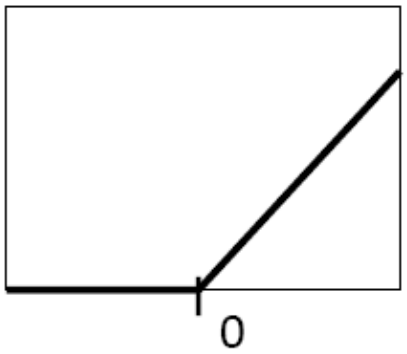
0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

Background

ReLU

✓ 아래 그림의 수식을 만족하는 “non-linear” activation function

- Ex) $ReLU(2) = 2, ReLU(-1) = 0$



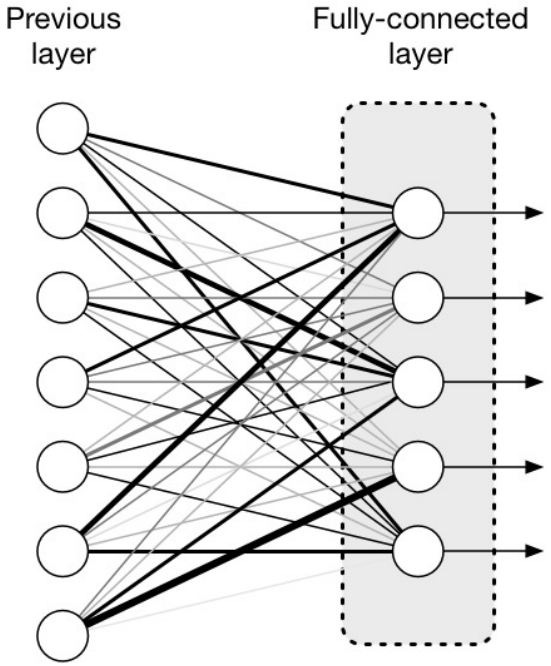
$$y = \begin{cases} z \text{ if } z > 0 \\ 0 \text{ otherwise} \end{cases}$$

Background

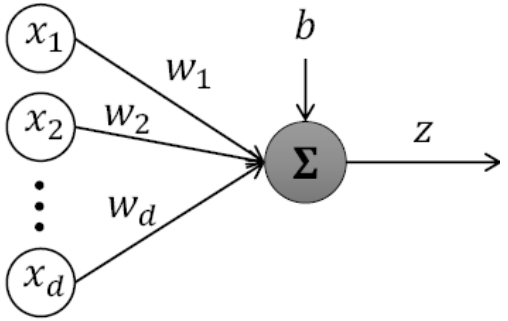
Fully connected layer

✓ 입력으로 받은 모든 pixel이 출력의 모든 pixel과 연결된 layer

- 전체 layer의 구조는 아래 왼쪽 그림과 같으며 각 output 은 오른쪽 그림처럼 연산된다.



$$z = \sum_i w_i x_i + b$$

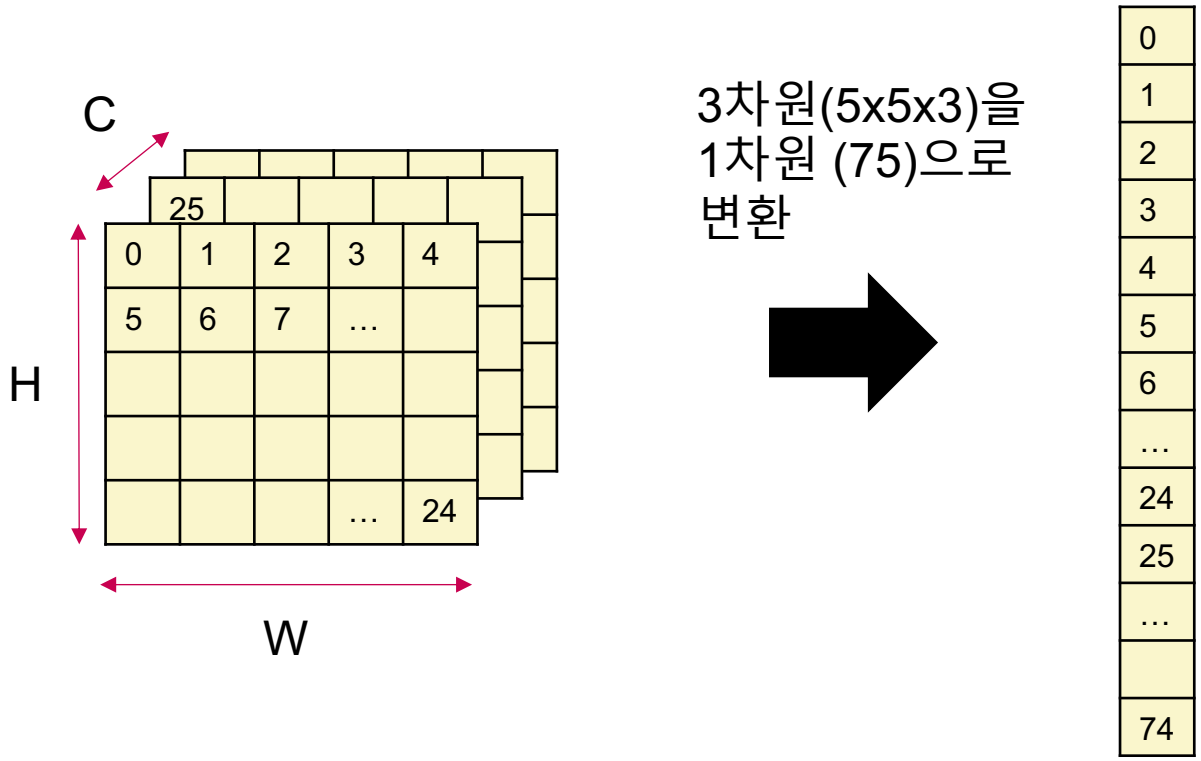


Background

Flatten to 1D array

✓ 3D (W×H×C) Activation에 대해 fully connected 연산을 하기 위해서는 activation을 1D로 만들어 줘야 함

- 1D로 만든 결과는 fully connected layer의 입력이 됨.



Background

Class Activation Map

✓ CNN에서 특정 Class를 예측할 때 이미지의 어떤 부분이 가장 중요한지를 식별할 수 있게 하는 기법

- 이번 과제에서는 간단화 하여, Convolution output과 예측한 class에 해당하는 fc layer의 weight를 element-wise product 하여 진행.

Image with Label

Class Activation Mapping(CAM)

Problem Definition

Problem Definition

Overview

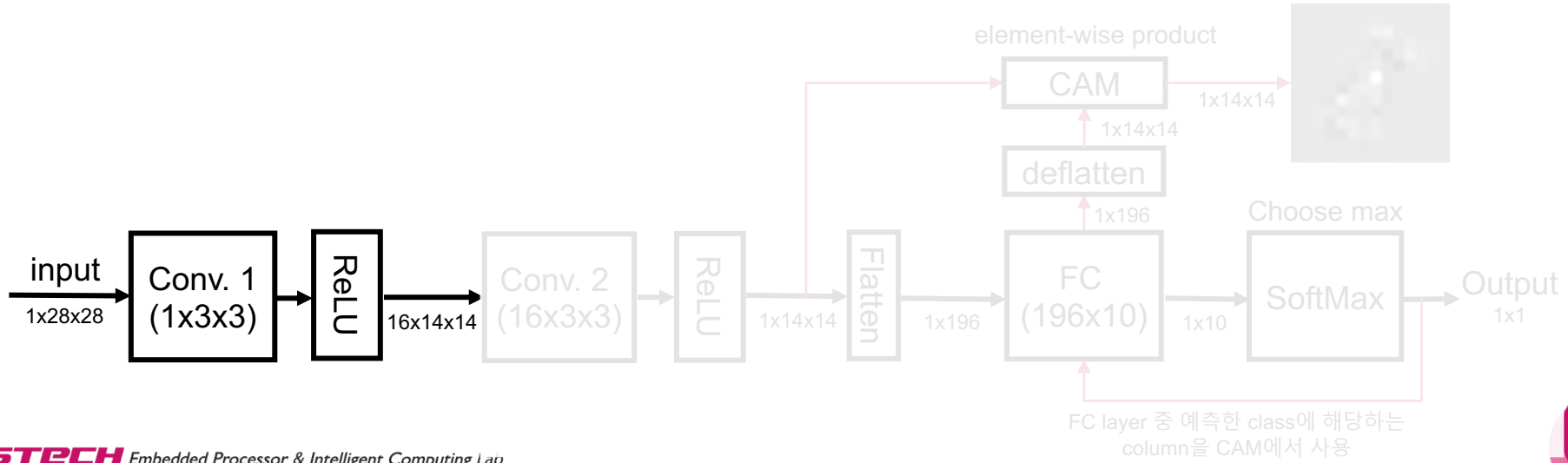
1. Input argument에 따라 모드 설정
   1-1. (Camera mode) Host PC와의 serial 통신으로 카메라 촬영 명령 전달 및 촬영
   1-2. (Example mode) serial 통신 없이 example 이미지 사용
2. 이미지 파일(.bmp)을 네트워크 입력으로 하여 미리 학습된 모델을 통해 추론(inference)
3. 추론된 결과를 7-segment에 출력, Activation Map은 output.bmp로 저장
4. 수행 시간과 추론 결과를 출력

- 가속 전/후 소스 코드 파일을 분리하기
- ex) 학번.c (가속 전 파일), 학번_opt.c (가속 후 파일)

Problem Definition

Step by step – Convolution layer 1

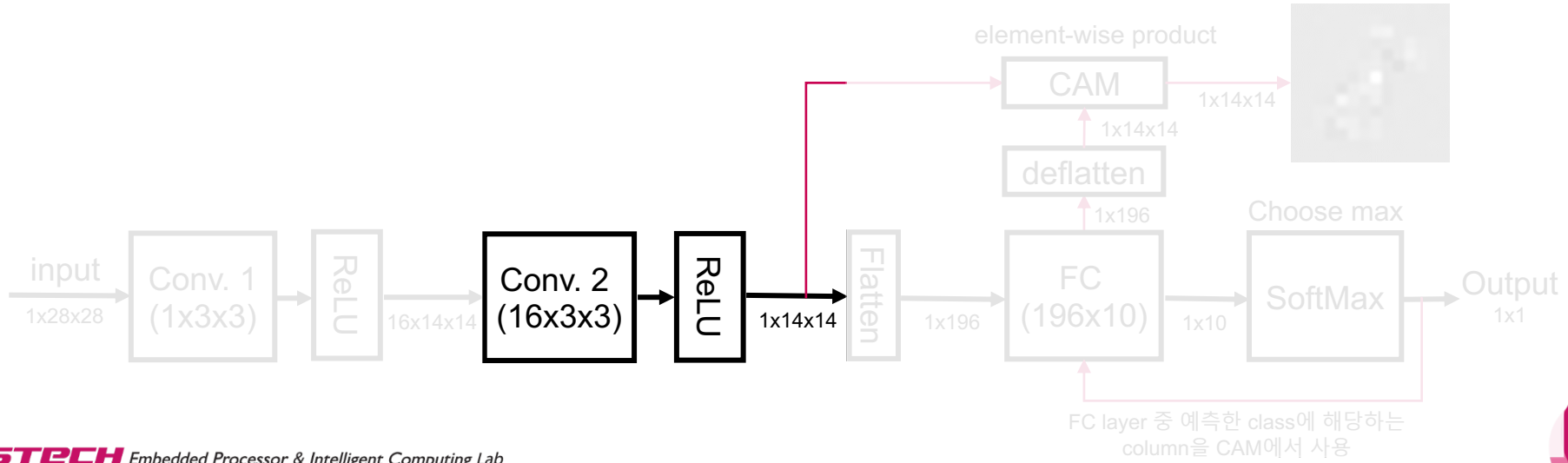
- ✓ Input: 1x28x28 dimension의 이미지
- ✓ zero padding 적용 -> 1x30×30의 이미지로 만들기
- ✓ 1x3×3 dimension인 16개의 filter로 convolution을 수행 stride: 2
- ✓ Output:16×14×14
- ✓ 이 Output에 ReLU function 적용 후 최종 16x14x14 output 만들기



Problem Definition

Step by step – Convolution layer 2

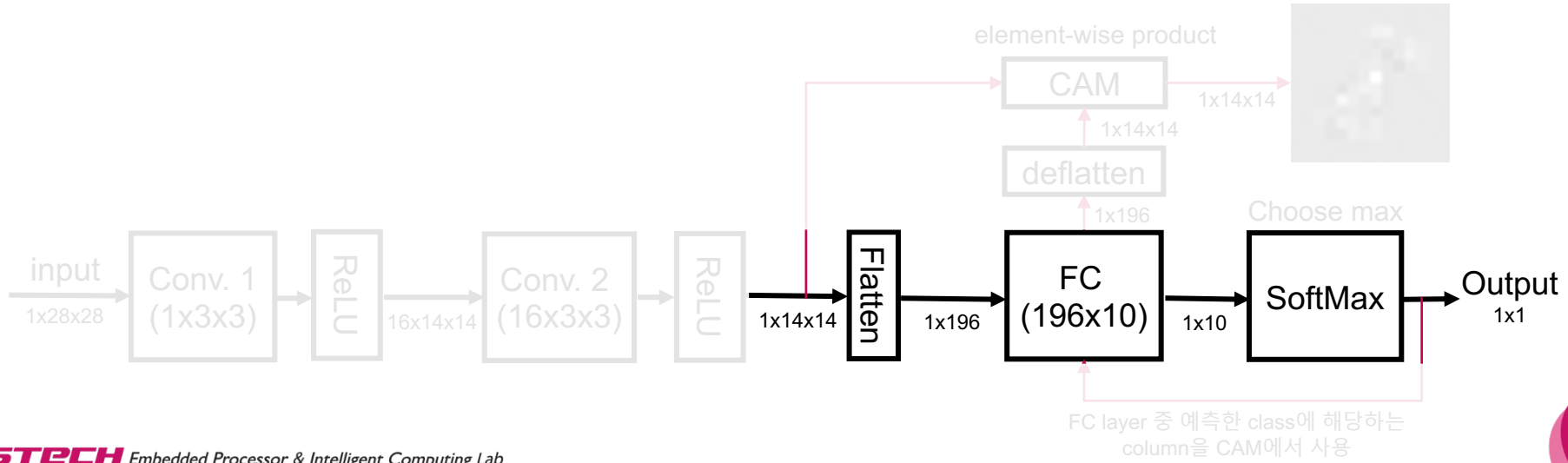
- ✓ Input: 16x14x14 dimension의 conv.1 output
- ✓ zero padding 적용 -> 16x16x16 dimension으로 만들기
- ✓ 16x3×3 dimension인 1개의 filter로 convolution을 수행 stride: 1
- ✓ Output:1×14×14
- ✓ 이 Output에 ReLU function 적용 후 최종 1x14x14 output 만들기



Problem Definition

Step by step – Fully Connect layer (FC) & SoftMax

- ✓ Input: 1x14x14 dimension의 conv.2 output
- ✓ Flatten 적용 -> 1x196 dimension으로 만들기
- ✓ 196x10 dimension인 FC layer와 Matrix Multiplication
- ✓ Output:1x10
- ✓ 이 Output에 SoftMax를 취해 그중 Maximum에 해당하는 class가 최종 output

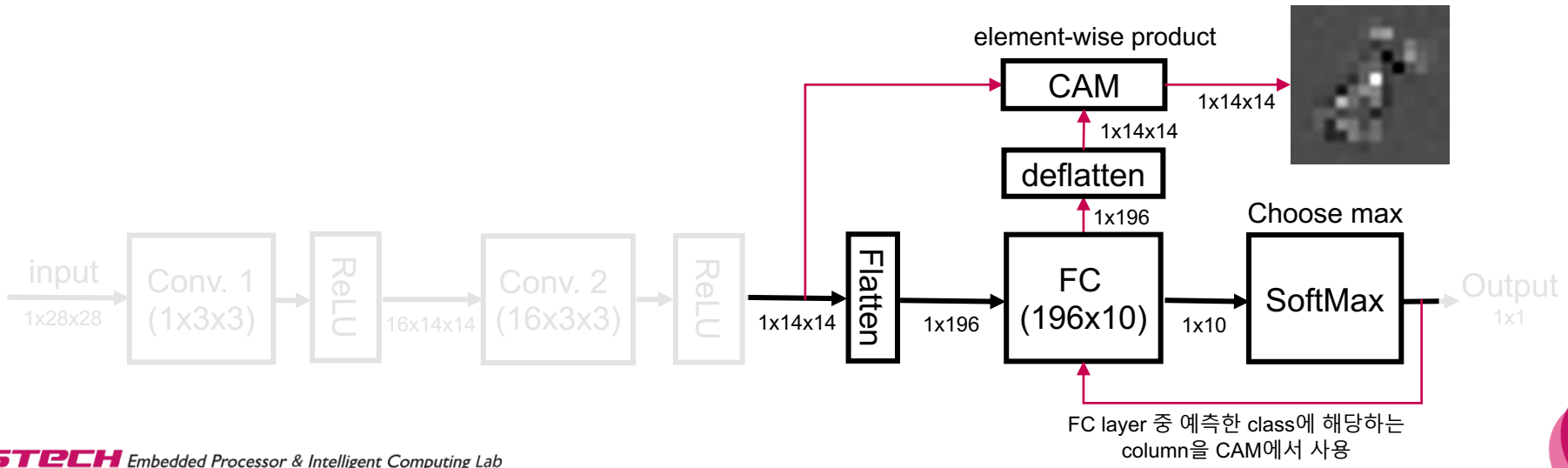


Problem Definition

Step by step – CAM

- ✓ Input 1: conv.2 output
- ✓ Input 2: FC layer 중 예측한 class에 해당하는 column 선택

  (e.g., 8번째 class로 predict 했다면 FC layer 중 8번째 column 선택)
- ✓ Input 2의 dimension을 input1의 dimension과 맞춰주기 위해서 deflatten 진행
- ✓ 이후, input 1,2를 element wise하게 곱하여 최종 activation map을 얻을 수 있다



Problem Definition

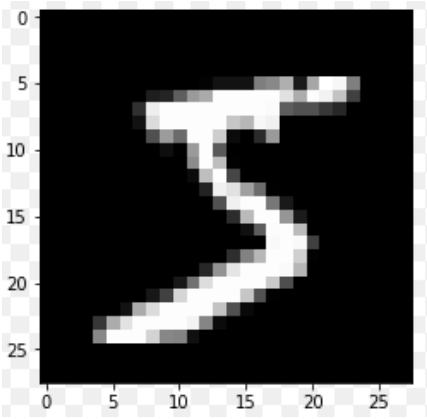
1. Input argument에 따라 모드 설정

- ✓ Input argument로 camera mode / example mode 설정
  - ‘0’ (camera mode) , ‘1’, ‘2’ (example mode)
- ✓ Camera mode: host PC에서 serial 통신을 통해 ‘c’ 또는 ‘C’가 입력되면 카메라를 촬영하고, 생성된 bmp 파일을 network의 입력으로 함.
  - $ sudo ./exec 0 실행 후 hterm으로 ‘c’ 또는 ‘C’ 입력하여 촬영
- ✓ Example mode: serial 통신 없이 바로 example 사진 파일을 network의 입력으로 함.
  - $ sudo ./exec 1 : example_1.bmp로 inference
  - $ sudo ./exec 2 : example_2.bmp로 inference

Problem Definition

1-1. (Camera mode) Host PC와의 serial 통신으로 카메라 촬영 명령 전달 및 촬영

- ✓ HW4, HW5에서 진행했던 내용을 참고
- ✓ Host PC에서 serial 통신을 통해 'c' 또는 'C'가 입력되면 카메라를 촬영하고, 생성된 bmp 파일을 network의 입력으로 함.
  - $ sudo ./exec 0 실행 후 hterm으로 'c' 또는 'C' 입력
- ✓ 이미지 해상도 : 280 x 280
- ✓ 학습에 사용된 MNIST는 아래 그림처럼 이미지가 저장되어 있음.
  - 촬영할 때 숫자가 이미지의 대부분을 차지하도록 촬영 권장
  - 숫자를 정중앙에 위치하도록 촬영을 권장



Problem Definition

1-2. (Example mode) serial 통신 없이 example 파일 사용

- ✓ HW5에서 진행했던 내용을 참고
- ✓ Serial 통신 없이 바로 example 사진 파일을 network의 입력으로 함.
  - $ sudo ./exec 1 : example_1.bmp로 inference
  - $ sudo ./exec 2 : example_2.bmp로 inference

example_1.bmp

example_2.bmp

Problem Definition

2. 이미지 파일(.bmp)을 네트워크 입력으로 하여 미리 학습된 모델을 통해 추론(inference)

- ✓ 제공된 모델(weights.bin) 데이터를 network에 저장하여 weight로 사용
- ✓ 입력 이미지 사이즈를 28×28로 변환하고, RGB 색 채널을 그레이스케일(grayscale)로 변환하여 새로운 배열에 저장 (Data type : unsigned char)
  - 숫자 부분의 pixel값이 255가 되도록 변경
- ✓ 각 픽셀 데이터를 0~1로 scaling하여 새로운 배열에 저장 (Data type : float)
- ✓ 위 내용은 스켈레톤 코드에 이미 구현되어 있음.

Activation_map_1.bmp

Activation_map_2.bmp

Problem Definition

2. 이미지 파일(.bmp)을 네트워크 입력으로 하여 미리 학습된 모델을 통해 추론(inference)

✓ 아래 함수들은 직접 구현해야함.

- Padding
- Conv_2d
- ReLU
- Linear (fully connected layer)
- Get_CAM (class activation map)

✓ Convolution 결과를 fully connected layer에서 연산하기 위해 1D로 flatten 할 때 순서는 width, height, channel

- model 안의 fc_weight의 구조 참고

Problem Definition

2. 이미지 파일(.bmp)을 네트워크 입력으로 하여 미리 학습된 모델을 통해 추론 (inference)

shape: CxHxW

✓ Convolution layer 1

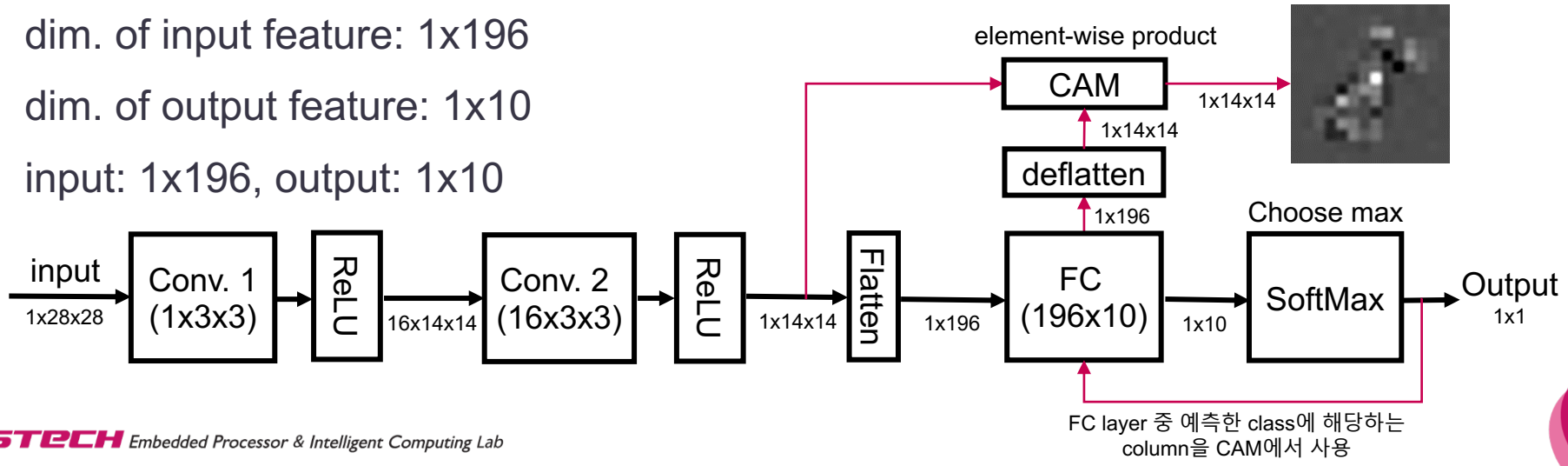
- Input channel 1
- Output channel 16
- Filter size 3×3
- Stride 2
- input: 1x28x28, output: 16x14x14

✓ Convolution layer 2

- Input channel 16
- Output channel 1
- Filter size 3×3
- Stride 1
- input: 16x14x14, output: 1x14x14

✓ Fully connected layer

- dim. of input feature: 1x196
- dim. of output feature: 1x10
- input: 1x196, output: 1x10



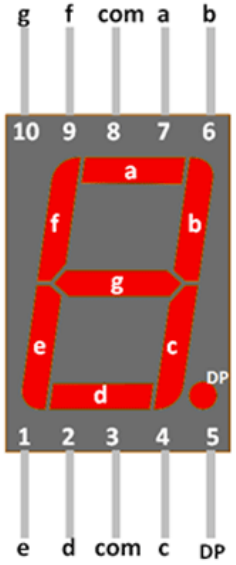
Problem Definition

3. 추론 결과를 7-segment에 출력

✓ HW4에서 진행했던 내용을 참고

✓ 다음과 같이 7-segment에 GPIO pin을 할당하여 코드를 작성

- a – 0
- b – 1
- c – 2
- d – 3
- e – 4
- f – 5
- g – 6
- dp - 7




```
pi@raspberrypi ~/code/wiringPi $ gpio readall
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 0 |  7 || 8  | 1 | ALT0 | TxD     |  15 |  14 |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     |  16 |  15 |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 |   1 |  18 |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 |   4 |  23 |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 |   5 |  24 |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 |   6 |  25 |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 0 | IN   | CE0     |  10 |   8 |
 |     |     |      0v |      |   | 25 || 26 | 0 | IN   | CE1     |  11 |   7 |
 |   0 |  30 |   SDA.0 |   IN | 0 | 27 || 28 | 0 | IN   | SCL.0   |  31 |   1 |
 |   5 |  21 | GPIO.21 |   IN | 0 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 0 | 31 || 32 | 0 | IN   | GPIO.26 |  26 |  12 |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 |  27 |  16 |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 |  28 |  20 |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 |  29 |  21 |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
```

4. CAM을 라즈베리파이에 bmp로 저장

✓ 구현 되어 있음

Problem Definition

5. 수행 시간과 추론 결과를 출력

- ✓ 수행 시간과 추론 결과 값 출력
- ✓ 컴파일 시 –lm 옵션 추가
- ✓ 수행 시간 측정은 구현되어 있음
- ✓ 수행 시간은 Softmax를 제외한 연산에 걸린 시간을 측정함.
  - 수행 시간 1: 추론 시에 zero padding, convolution, relu, fc 연산 수행 시간
  - 수행 시간 2: convolution 2 output과 de-flatten한 fc layer weight의 element-wise multiplication 연산 수행시간 (CAM 수행시간)
  - 수행 시간1,2의 합 측정
  - 수행 시간 측정 코드는 skeleton 코드에 이미 구현되어 있음

- ✓ 추론 결과 값은 softmax 값과 추론된 숫자를 출력함.

Evaluation

Evaluation

조교가 검사할 수 있는 source code와 결과보고서 PDF를 제출

- ✓ Due date: 6/8 (토) 23:59
- ✓ 실험 물품 반납 due: 6/8 (토), 추후 상세 안내 예정
- ✓ 과제에 관한 질문은 Q&A 게시판 활용
- ✓ 제출 방식: **학번.zip** 파일의 형식으로 PLMS에 제출
  - 제출 소스코드는 주어진 project_skeleton.c 파일을 아래와 같이 파일 이름 수정 후 제출
  - 제출 예시)

```
20240123.zip
  ㄴ 20240123.c        // 가속 전 파일
  ㄴ 20240123_opt.c    // 가속 후 파일
  ㄴ 20240123.pdf      // 보고서
```

Evaluation

평가 항목

✓ 시스템 구현

- 카메라로 촬영한 이미지에 대한 추론 여부

✓ 가속 구현

- 가속 전/후의 수행시간, softmax 값, 추론 결과 출력
- 시간은 zero padding, convolution, ReLU, fully connected layer, weight element-wise multiplication 만 측정
  - 측정 수행시간 1 : zero padding, convolution, ReLU, fully connected layer 시간만 고려된 추론 수행
  - 측정 수행시간 2 : fc layer weight를 de-flatten하고, convolution 2+ReLU output과 element-wise multiplication 연산 수행시간 (CAM 수행시간)
  - 측정 수행시간 1,2의 합: 시간의 단순 합이 아닌, 1,2에 해당하는 코드 구간의 수행시간
- 가속 구현 방법은 수업시간에 배운 내용 모두 사용가능
- 동일한 이미지에 대한 추론을 위해 test image 제공

Evaluation

100점 만점으로 채점하며 다음 사안을 고려

✓ 실행 시간 (80)

- 시스템 구현 기본 점수 (40)
- **가속 후의 실행 시간에 따라 차등 배점** (만점 40)
  - 동일한 라즈베리파이 환경에서 실행 시간 측정 예정
  - 동일한 컴파일 옵션을 적용하여 테스트 예정

✓ 보고서 (20)

- PDF 형식으로 제출
- A4 3장 이하
- Background 생략하고 가속 방법에 대해서만 작성
- 프로젝트를 수행하며 학습한 핵심 내용만을 작성

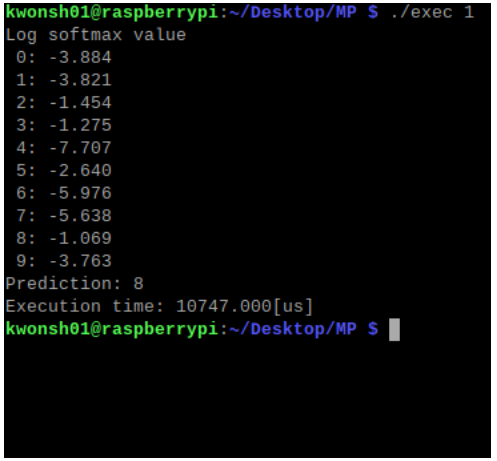
✓ **부정행위 적발 시 0점**

Evaluation

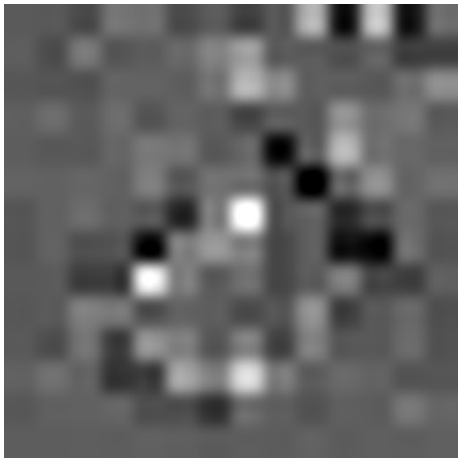
보고서 작성 시 고려 사항

- ✓ PDF 형식으로 제출
- ✓ A4 3장 이하
- ✓ Background 내용 생략하고 가속 방법에 대해서만 작성
- ✓ 프로젝트를 수행하며 학습한 핵심 내용만을 작성
- ✓ 직접 촬영한 이미지와 inference 결과 이미지 첨부
  - 예시

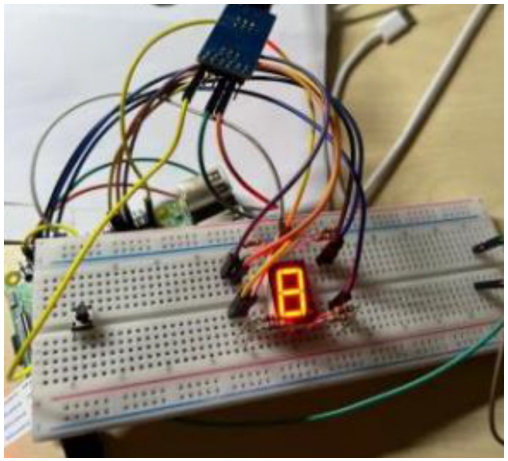
Captured image



softmax output



Activation map



7-segment output