REPORT

전자공학도의 윤리 강령 (IEEE Code of Ethics)

(출처: http://www.ieee.org)

나는 전자공학도로서, 전자공학이 전 세계 인류의 삶에 끼치는 심대한 영향을 인식하여 우리의 직업, 동료와 사회에 대한 나의 의무를 짐에 있어 최고의 윤리적, 전문적 행위를 수행할 것을 다짐하면서, 다음에 동의한다.

1. **공중의 안전, 건강 복리에 대한 책임**: 공중의 안전, 건강, 복리에 부합하는 결정을 할 책임을 질 것이며, 공중 또는 환경을 위협할 수 있는 요인을 신속히 공개한다.

2. **지위 남용 배제**: 실존하거나 예기되는 이해 상충을 가능한 한 피하며, 실제로 이해가 상충할 때에는 이를 이해 관련 당사자에게 알린다. (이해 상충: conflicts of interest, 공적인 지위를 사적 이익에 남용할 가능성)

3. **정직성**: 청구 또는 견적을 함에 있어 입수 가능한 자료에 근거하여 정직하고 현실적으로 한다.

4. **뇌물 수수 금지**: 어떠한 형태의 뇌물도 거절한다.

5. **기술의 영향력 이해**: 기술과 기술의 적절한 응용 및 잠재적 영향에 대한 이해를 높인다.

6. **자기계발 및 책무성**: 기술적 능력을 유지, 증진하며, 훈련 또는 경험을 통하여 자격이 있는 경우이거나 관련 한계를 전부 밝힌 뒤에만 타인을 위한 기술 업무를 수행한다.

7. **엔지니어로서의 자세**: 기술상의 업무에 대한 솔직한 비평을 구하고, 수용하고, 제공하며, 오류를 인정하고 수정하며, 타인의 기여를 적절히 인정한다.

8. **차별 안하기**: 인종, 종교, 성별, 장애, 연령, 출신국 등의 요인에 관계없이 모든 사람을 공평하게 대한다.

9. **도덕성**: 허위 또는 악의적인 행위로 타인, 타인의 재산, 명예, 또는 취업에 해를 끼치지 않는다.

10. **동료애**: 동료와 협력자가 전문분야에서 발전하도록 도우며, 이 윤리 헌장을 준수하도록 지원한다.

위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학부
마감일: 2021. 03. 31.
과목명: 전자공학프로그래밍
교수명: 이정원 교수님
학 번: 201620837
성 명: 안민규

1. 문제 분석

1) Convolution

convolution이란 하나의 함수와 또 다른 함수를 반전 이동하여 곱하고, 구간에 대해 적분하여 새로운 값을 만드는 연산이다. 이를 사용하는 이유로는 여러 가지가 있지만, 이렇게 프로그램상에서 이루어지는 convolution은 신호처리 분야에서 가장 많이 사용되곤 한다. convolution은

$(f * g)(m) = \sum_n f(n)g(m-n)$으로 정의된다.

이렇게 프로그램상에서 convolution에는 두 가지 방식이 존재하는데, 배열의 크기를 일정하게 유지하는 Same방법과 convolution을 진행하면 크기가 줄어드는 Valid방법이 존재한다.

2) Same Convolution 방법

주어진 입력과 같은 크기의 배열을 가지는 결과 배열을 만들어내는 방법이다. 특이적으로 zero padding을 사용한다. zero padding이란 주어진 배열을 크기를 일정하게 유지하는 방법으로, 배열의 외곽에 1픽셀씩 더 추가시켜 이를 0으로 채우는 방법이다. 주어진 입력을 반복해서 convolution 할 때 zero padding을 하지 않으면 입력의 크기가 점점 줄어들기 때문에, 이를 방지하기 위해 크기를 같게 (Same) 유지하는 방법이다.

이를 위해선 주어진 입력보다 가로, 세로로 2픽셀씩 추가한 배열을 새로 만들고 [1,1]~[1,11]~[11,11]까지를 주어진 입력의 값을 받아 새로운 입력을 만드는 방법을 사용하였다. 입력이 3x3 배열이라면, 아래 표와 같이 구성한다.

0	0	0	0	0
0	주어진 입력 [0.0]	[0.1]	[0.2]	0
0	[1.0]	[1.1]	[1.2]	0
0	[2.0]	[2.1]	[2.2]	0
0	0	0	0	0

2) Valid Convolution 방법

Valid 방법이란 앞선 same방법과는 다르게 zero padding을 하지 않고 convolution을 진행하는 방법이다. 입력 배열과 출력 배열의 크기가 다른 방법이고, 이는 앞의 결과와 다른 특징을 추출하려고할 때 사용하는 방법이다.

이 방법은 따로 zero padding을 하지 않으므로 새로운 배열을 만들지 않고 곧바로 입력값과 필터값을 convolution하여 결과값에 넣었다.

2. 함수의 기능 및 코드 분석

2-1) makeinput 함수

```
void makeInput(int input_value[][WIDTH])
{
    int type = 0;
    int max_value = 0;
    printf("어떤 방식으로 입력값을 생성 하시겠습니까?( 1. 랜덤, 2. 모든 값1)₩n");
    scanf("%d", &type);

    switch (type)
    {
    case 1:
        printf("랜덤 값의 최대 값은? ");
```

```
        scanf("%d", &max_value);
        for (int i = 0; i < HEIGHT; i++)
            for (int j = 0; j < WIDTH; j++)
                input_value[i][j] = rand() % (max_value + 1);
        break;

    case 2:
        for (int i = 0; i < HEIGHT; i++)
            for (int j = 0; j < WIDTH; j++)
                input_value[i][j] = 1;
        break;
    default:
        break;
    }
}
```

위 함수는 makeinput 함수로써 인풋 배열을 만들어주는 역할을 한다. 전역변수로 선언된 input_value를 변수로 받아 작동한다. 사용자에게 인풋 배열을 어떻게 만들 것인지를 확인한다.

1번의 경우 랜덤으로 배열 내부가 채워지며, 랜덤 값의 최대 값을 입력하면 그 이상의 값은 채워지지 않는다. 2번의 경우 배열의 모든 값을 1로 채우는 방법이다. 모두 1의 값을 가지므로 convolution이 올바르게 작동하는지 확인하기 용이하다. 1, 2번이 아닌 다른 숫자를 입력 시 바로 스위치 문을 나오게끔 설정되어있다.

주어진 HEIGHT와 WIDTH만큼의 반복문을 진행하여 배열을 채운다.

2-2) Makefilter 함수

```
void makeFilter(int filter_value[][FILTER_WIDTH])
{
    int type = 0;
    int max_value = 0;
    printf("어떤 방식으로 입력값을 생성 하시겠습니까?( 1. 랜덤, 2. 모든 값 1, 3. ppt filter)₩n");
    scanf("%d", &type);

    switch (type)
    {
    case 1:
        printf("랜덤 값의 최대 값은? ");
        scanf("%d", &max_value);
        for (int i = 0; i < FILTER_HEIGHT; i++)
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = rand() % (max_value + 1);
        break;

    case 2:
        for (int i = 0; i < FILTER_HEIGHT; i++)
```

```
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = 1;
            break;
    case 3:
        for (int i = 0; i < FILTER_HEIGHT; i++)
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = ppt_filter[i][j];
        break;
    default:
        break;
    }
}
```

위 함수는 makefilter 함수로써 makeinput 함수와 같이 전역변수로 선언된 filter_value를 변수로 받아 작동한다. makeinput 함수와 같이 필터를 만들어주는 함수인데, 사용자에게 입력을 받아서 1, 2, 3중 어떤 방법으로 사용할지 확인한다. 1번의 경우 필터에 들어가는 숫자의 최댓값을 받아 그 이하의 숫자로 필터를 무작위 적으로 채우는 방법이고, 2번의 경우 필터의 모든 값을 1로, 3번의 경우 주어진 필터를 사용하는 것이다. 미리 선언된 filter_height, width만큼 반복문을 진행하여 필터를 만든다.

2-3) printinput 함수

```
void printInput(int input_value[][WIDTH])
{
    printf("₩n");
    printf("입력값₩n");
    printf("----------------------------------------₩n");
    for (int i = 0; i < HEIGHT; i++)
    {
        printf("|");
        for (int j = 0; j < WIDTH; j++)
        {
            printf("%2d |", input_value[i][j]);
        }

        printf("₩n");
    }
    printf("----------------------------------------₩n");
}
```

위 함수는 printinput 함수로 앞서 makeinput 함수를 통해 생성된 인풋 배열을 보여주는 함수이다. 주어진 높이와 너비 만큼 반복문을 진행하여 배열의 값을 출력한다.

2-4) printfilter 함수

```
void printFilter(int filter_value[][FILTER_WIDTH])
{
    printf("필터값₩n");
```

```
    printf("------------₩n");
    for (int i = 0; i < FILTER_HEIGHT; i++)
    {
        printf("|");
        for (int j = 0; j < FILTER_WIDTH; j++)
        {
            printf("%2d |", filter_value[i][j]);
        }

        printf("₩n");
    }
    printf("------------₩n");
}
```

위 함수는 printfilter 함수로 printinput 함수와 마찬가지로 makefilter 함수로 생성된 필터를 출력해준다.

2-5) ConvSame 함수

```
void ConvSame()
{
    int padded_signal[12][12] = { 0, };
    for (int n = 0; n < 12; n++)
    {
        padded_signal[n][0] = 0;
        padded_signal[n][11] = 0;
        padded_signal[0][n] = 0;
        padded_signal[11][n] = 0;
    } //padding 작업

    for (int i = 1; i < 11; i++)
        for (int j = 1; j < 11; j++)
            padded_signal[i][j] = input_value[i - 1][j - 1]; //input을 padded_signal에 넣어 새로운 표본 만들기

    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 10; j++)
            result[i][j] = padded_signal[i][j] * filter_value[2][2] + padded_signal[i][j + 1] * filter_value[2][1] + padded_signal[i][j + 2] * filter_value[2][0]
            + padded_signal[i + 1][j] * filter_value[1][2] + padded_signal[i + 1][j + 1] * filter_value[1][1] + padded_signal[i + 1][j + 2] * filter_value[1][0]
            + padded_signal[i + 2][j] * filter_value[0][2] + padded_signal[i + 2][j + 1] * filter_value[0][1] + padded_signal[i + 2][j + 2] * filter_value[0][0]; //컨벌루션값 결과에 저장

    result_height = 10;
    result_width = 10; //main 함수에서 반복문을 진행하기 위해 높이와 너비 정함
```

```
}
```

위 함수는 주어진 게 아닌 직접 프로그래밍한 함수이다. convolution의 경우 총 2가지의 방법을 사용할 수 있으므로 그 중 SAME 방법으로 계산하는 함수이다. 앞서 문제 분석에서 설명한 것처럼 SAME 방법은 zero padding을 이용하므로 새로운 배열을 만들어주어야 한다고 생각하였다. 따라서 padded_signal 배열을 만든다. 이때, 주어진 인풋의 크기 10x10보다 양옆으로 1픽셀씩 키운 12x12의 크기로 만든다. 반복문을 사용해 새롭게 추가된 부분은 0으로 채워 zero padding을 완성한다.

padding이 완료된 배열에 주어진 인풋 배열을 입력하여 새로운 배열을 완성한다. 그 후 반복문을 진행해 convolution을 계산한다. 이때, 필터를 그대로 계산하지 않고 cross해 계산을 해야하므로 코드를 위와 같이 작성하였다. 마지막으로 후에 main 함수에서 결괏값을 출력할 때 반복문의 조건을 정하기 위해 결과 배열의 height, width를 정해주었다.

2-6) ConvValid 함수

```
void ConvValid()
{
    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 10; j++)
        {
            result[i][j] = input_value[i][j] * filter_value[2][2] + input_value[i][j + 1] * filter_value[2][1] + input_value[i][j + 2] * filter_value[2][0]
                + input_value[i + 1][j] * filter_value[1][2] + input_value[i + 1][j + 1] * filter_value[1][1] + input_value[i + 1][j + 2] * filter_value[1][0]
                + input_value[i + 2][j] * filter_value[0][2] + input_value[i + 2][j + 1] * filter_value[0][1] + input_value[i + 2][j + 2] * filter_value[0][0];
        } //컨벌루션값 결과에 저장

    result_height = 8;
    result_width = 8; //main 함수에서 반복문을 진행하기 위해 높이와 너비 정함
}
```

위 함수는 Valid방법으로 convolution을 계산하는 함수를 프로그래밍하였다. Same방법과 다르게 zero padding을 하지 않으므로 새로 배열을 만들지 않고, 주어진 인풋과 필터를 convolution한 값을 결과 배열에 대입한다. valid 방법의 경우 10x10배열이 8x8배열로 크기가 줄어들기 때문에 결과 배열의 height, width를 8로 정해주었다.

2-7) main 함수

```
int main()
{
    makeInput(input_value);
    makeFilter(filter_value);

    int mode;

    printInput(input_value);
```

```
    printFilter(filter_value);

    printf("진행할 convolution을 정해주세요. (1. SAME 2. VALID) : ");
    scanf("%d", &mode);

    switch (mode)
    {
    case 1:
        ConvSame();
        break;
    case 2:
        ConvValid();
        break;
    default:
        printf("잘못된 값을 입력하였습니다. 프로그램이 종료됩니다.");
        return -1;
    }
    printf("결과값₩n");
    printf("----------------------------------------₩n");
    for (int i = 0; i < result_height; i++)
    {
        printf("|");
        for (int j = 0; j < result_width; j++)
        {
            printf("%2d |", result[i][j]);
        }

        printf("₩n");
    }
    printf("----------------------------------------₩n");

    return 0;
}
```

main함수이다. makeinput함수와 makefilter함수를 먼저 작동시켜 인풋 배열과 필터 배열을 만든다. 그 후 만들어진 input과 filter를 출력한다. 그 후 진행할 convolution 방법을 선택한다. 선택한 값에 대한 값은 따로 mode라는 변수를 만들어 이를 가지고 switch를 진행한다. 1을 입력할 시 Same방법이 진행되고 2를 입력할 시 Valid방법이 진행된다. 이 때 1, 2가 아닌 다른 값을 입력할 시 경고문구가 뜨고 프로그램이 종료된다.

올바르게 convolution이 진행된 후에는 결과값이 출력되게 설정하였다. 그 후 프로그램이 종료된다.

3. 프로그램 코드

```
/*
학과 : 전자공학과
```

```
학번 : 201620837
이름 : 안민규
과목 : 전자공학 프로그래밍
교수님 : 이정원 교수님
*/
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define WIDTH           10
#define HEIGHT          10
#define FILTER_WIDTH   3
#define FILTER_HEIGHT 3
#define SAME            0
#define VALID           1  //변하지 않는 값으로 지정해놓음

int ppt_filter[3][3] = { {1,2,1}, {2,3,2}, {1,2,1} }; //강의노트에 제시된 필터(필터 종류 3번)

int input_value[HEIGHT][WIDTH];
int filter_value[FILTER_HEIGHT][FILTER_WIDTH];
void printInput(int input_value[][WIDTH]);
void makeInput(int input_value[][WIDTH]);
void makeFilter(int filter_value[][FILTER_WIDTH]);
void printFilter(int filter_value[][FILTER_WIDTH]); //주어진 함수 및 변수 선언
int result_height=0;
int result_width=0;
int result[HEIGHT][WIDTH]; //새로 설정한 결과 배열을 위한 변수 선언

void ConvSame() //SAME 방법 함수
{
   int padded_signal[12][12] = { 0, };
   for (int n = 0; n < 12; n++)
   {
      padded_signal[n][0] = 0;
      padded_signal[n][11] = 0;
      padded_signal[0][n] = 0;
      padded_signal[11][n] = 0;
   } //padding 작업

   for (int i = 1; i < 11; i++)
      for (int j = 1; j < 11; j++)
         padded_signal[i][j] = input_value[i - 1][j - 1]; //input을 padded_signal에 넣어 새로운
```

표본 만들기

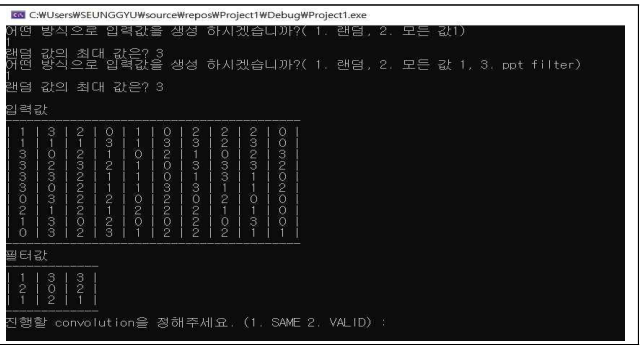
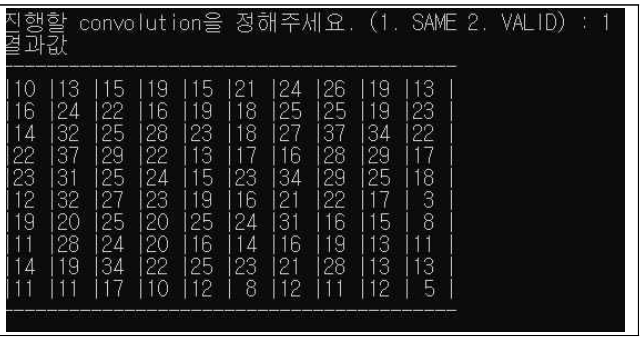
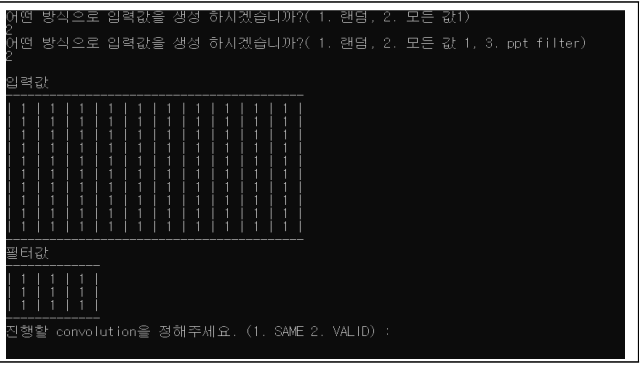
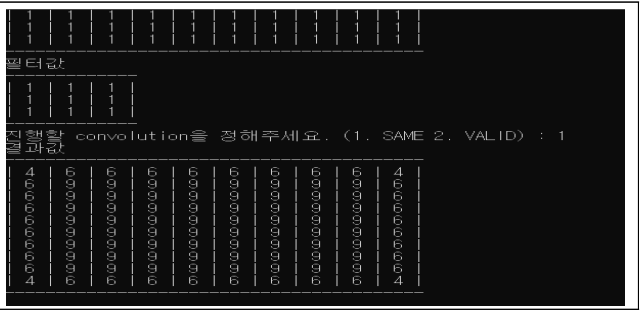
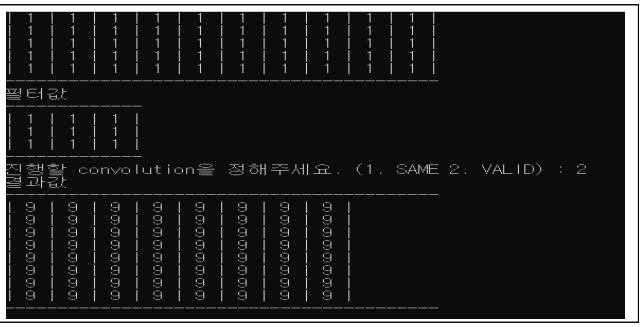
```
   for (int i = 0; i < 10; i++)
      for (int j = 0; j < 10; j++)
         result[i][j] = padded_signal[i][j] * filter_value[2][2] + padded_signal[i][j + 1] * filter_value[2][1] + padded_signal[i][j + 2] * filter_value[2][0]
         + padded_signal[i + 1][j] * filter_value[1][2] + padded_signal[i + 1][j + 1] * filter_value[1][1] + padded_signal[i + 1][j + 2] * filter_value[1][0]
         + padded_signal[i + 2][j] * filter_value[0][2] + padded_signal[i + 2][j + 1] * filter_value[0][1] + padded_signal[i + 2][j + 2] * filter_value[0][0]; //컨벌루션값 결과에 저장

   result_height = 10;
   result_width = 10; //main 함수에서 반복문을 진행하기 위해 높이와 너비 정함
}

void ConvValid() //VALID 방법 함수
{
   for (int i = 0; i < 10; i++)
      for (int j = 0; j < 10; j++)
      {
         result[i][j] = input_value[i][j] * filter_value[2][2] + input_value[i][j + 1] * filter_value[2][1] + input_value[i][j + 2] * filter_value[2][0]
            + input_value[i + 1][j] * filter_value[1][2] + input_value[i + 1][j + 1] * filter_value[1][1] + input_value[i + 1][j + 2] * filter_value[1][0]
            + input_value[i + 2][j] * filter_value[0][2] + input_value[i + 2][j + 1] * filter_value[0][1] + input_value[i + 2][j + 2] * filter_value[0][0];
      } //컨벌루션값 결과에 저장

   result_height = 8;
   result_width = 8; //main 함수에서 반복문을 진행하기 위해 높이와 너비 정함
}

int main()
{
   makeInput(input_value);
   makeFilter(filter_value); // 입력과 필터를 만들어주는 함수 구동

   int mode; // 어떤 방법으로 convolution 진행할지 저장할 변수

   printInput(input_value);
   printFilter(filter_value); // 생성된 입력과 필터를 출력해주는 함수

   printf("진행할 convolution을 정해주세요. (1. SAME 2. VALID) : ");
   scanf("%d", &mode);
```

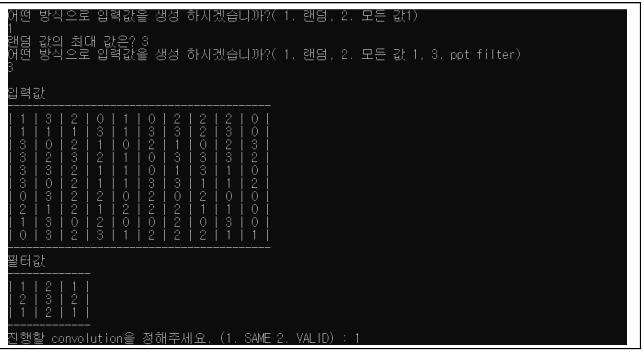
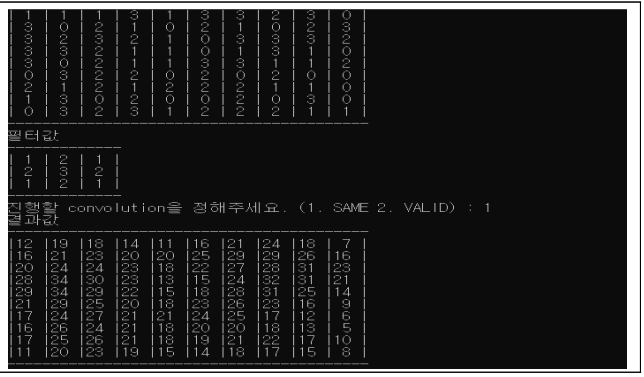
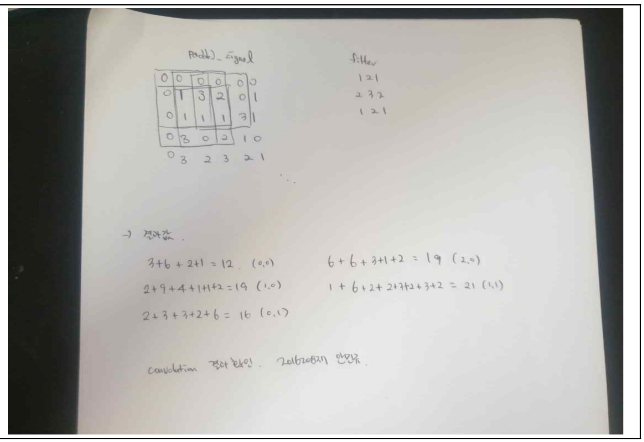
```
    switch (mode) //mode에 입력된 값에 따라 방법 선택하여 convolution 진행
    {
    case 1:
        ConvSame();
        break;
    case 2:
        ConvValid();
        break;
    default:
        printf("잘못된 값을 입력하였습니다. 프로그램이 종료됩니다.");
        return -1; //1과 2가 아닌 다른 숫자 입력시 진행
    }
    printf("결과값\n");
    printf("----------------------------------------\n");
    for (int i = 0; i < result_height; i++)
    {
        printf("|");
        for (int j = 0; j < result_width; j++)
        {
            printf("%2d |", result[i][j]);
        }

        printf("\n");
    } //same함수와 valid함수에서 미리 설정해놓은 height, width에 따라 반복문 진행
    printf("----------------------------------------\n");

    return 0;
}

void printInput(int input_value[][WIDTH]) //인풋 출력 함수
{
    printf("\n");
    printf("입력값\n");
    printf("----------------------------------------\n");
    for (int i = 0; i < HEIGHT; i++)
    {
        printf("|");
        for (int j = 0; j < WIDTH; j++)
        {
            printf("%2d |", input_value[i][j]);
        }

        printf("\n");
```

```
    } //주어진 height, width 만큼 반복문 진행하여 생성된 입력 출력
    printf("----------------------------------------\n");
}

void printFilter(int filter_value[][FILTER_WIDTH]) //필터 출력 함수
{
    printf("필터값\n");
    printf("-------------\n");
    for (int i = 0; i < FILTER_HEIGHT; i++)
    {
        printf("|");
        for (int j = 0; j < FILTER_WIDTH; j++)
        {
            printf("%2d |", filter_value[i][j]);
        }

        printf("\n");
    } //주어진 height, width 만큼 반복문 진행하여 생성된 필터 출력
    printf("-------------\n");
}

void makeInput(int input_value[][WIDTH]) //인풋 생성 함수
{
    int type = 0;
    int max_value = 0;
    printf("어떤 방식으로 입력값을 생성 하시겠습니까?( 1. 랜덤, 2. 모든 값1)\n");
    scanf("%d", &type); //어떤 방법으로 진행할지 입력받음

    switch (type)
    {
    case 1: //랜덤으로 생성
        printf("랜덤 값의 최대 값은? "); //배열에 들어갈 최대 숫자값 입력 받음
        scanf("%d", &max_value);
        for (int i = 0; i < HEIGHT; i++)
            for (int j = 0; j < WIDTH; j++)
                input_value[i][j] = rand() % (max_value + 1); //랜덤으로 숫자를 생성하는 구문
        break;

    case 2: //1로 배열 채움
        for (int i = 0; i < HEIGHT; i++)
            for (int j = 0; j < WIDTH; j++)
                input_value[i][j] = 1;
        break;
    default:
```

```
        break;
    }
}

void makeFilter(int filter_value[][FILTER_WIDTH]) //필터 생성 함수
{
    int type = 0;
    int max_value = 0;
    printf("어떤 방식으로 입력값을 생성 하시겠습니까?( 1. 랜덤, 2. 모든 값 1, 3. ppt filter)\n");
    scanf("%d", &type); //어떤 방법으로 진행할지 입력받음

    switch (type)
    {
    case 1: //랜덤 생성
        printf("랜덤 값의 최대 값은? "); //배열에 들어갈 최대 숫자값 입력 받음
        scanf("%d", &max_value);
        for (int i = 0; i < FILTER_HEIGHT; i++)
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = rand() % (max_value + 1);
        break;

    case 2: //1로 배열 채움
        for (int i = 0; i < FILTER_HEIGHT; i++)
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = 1;
            break;

    case 3:
        for (int i = 0; i < FILTER_HEIGHT; i++)
            for (int j = 0; j < FILTER_WIDTH; j++)
                filter_value[i][j] = ppt_filter[i][j];
        break;
    default:
        break;
    }
}
```

4. 결과화면 분석

랜덤 방법으로 입력 생성 최댓값 3 설정 랜덤 방법으로 필터 생성 최댓값 3 설정 생성된 인풋과 필터 출력 랜덤 생성이 잘 일어남을 확인할 수 있고, 최댓값 또한 잘 적용됨을 확인할 수 있다.	
앞의 사진 1을 입력받아 SAME 방법으로 convolution 진행 SAME 방법으로 진행하였으므로 convolution한 결과 배열이 크기가 인풋 배열과 같음을 알 수 있다.	
입력 배열 1로 채움 필터 배열 1로 채움 생성된 인풋과 필터 출력 모든 값이 전부 다 1로 잘 나오는 것을 확인할 수 있다.	
앞의 사진 1을 입력받아 SAME 방법으로 convolution 진행 이 또한 크기가 인풋 배열과 같음을 알 수 있다.	
앞의 사진 2을 입력받아 valid 방법으로 convolution 진행 same 방법과의 차이를 보기 위해 조건을 똑같이 하고 valid 방법으로 진행하였다. same 방법과 다르게 결과 배열의 크기가 인풋 배열보다 작음을 알 수 있다. 또한 결과값이 서로 다름을 알 수 있다.	

랜덤 방법으로 입력 생성 최댓값 3 설정 주어진 필터 사용 생성된 인풋과 필터 출력	
앞의 사진에 1을 입력받아 same 방법으로 convolution을 진행한 결과 올바르게 작동함을 알 수 있다.	
앞의 결과를 직접 손으로 계산하여 확인한 결과, convolution이 잘 이뤄짐을 알 수 있다.	

5. 참고문헌

https://deepestdocs.readthedocs.io/en/latest/004_deep_learning_part_2/0043/
전자공학 프로그래밍 과제 강의노트