Digital Signal Processing Final Project

Hand-Gesture classification with IR-UWB Radar data using Deep Learning model

학과 : 전기생체공학부 전기공학전공

이름 : 홍종혁

1. 서론

1.1 프로젝트의 주제

본 실습에서는 IR-UWB Radar를 이용하여 “Air-HandWriting”을 수행하고, 이를 통해 생성한 이미지들을 딥러닝을 이용해서 판별할 것이다.

IR-UWB Radar란, 극히 짧은 주기를 가지는 Impulse 신호를 의미하는 Impulse-Radio와 매우 넓은 주파수 대역을 의미하는 Ultra-WideBand의 특성을 지닌 전파를 사용하는 Radar이다. IR-UWB Radar는 높은 분해능을 지니고 있어 모션 인식에 활용할 수 있는데[1], 이러한 특성을 바탕으로 하면 IR-UWB Radar를 이용하여 “Air-HandWriting”을 수행할 수 있다.

“Air-HandWriting”이란, 손가락이나 손을 움직여 허공에 문자나 단어, 기호 등을 쓰는 것을 의미한다. Air-HandWriting은 보통 손의 움직임의 6개의 자유도(Six degrees of freedom) 기반 데이터를 기반으로 수행되는데, 해당 데이터는 X축 중심의 좌우회전, Y축 중심의 앞뒤 회전, Z축 중심의 상하 회전에 더해 앞뒤, 좌우, 상하로의 병진 운동까지 포함된 데이터이다. 모델링과 인식에는 Hidden Markov Model이 사용되며[2], 이를 통해 얻은 데이터를 딥러닝을 이용해서 판별할 것이다.

딥러닝은 인공지능 기술의 한 종류이다. 인공지능 기술이란, 주어진 데이터들을 가지고 스스로 학습하여 적응해 나갈 수 있는 기술을 의미하는데, 머신러닝(ML)은 이러한 인공지능 모델 중, 데이터를 많이 학습할수록 성능이 좋아지는 모델을 의미하고, 딥러닝은 이러한 머신러닝의 한 종류이다. 딥러닝(DL)에서는 인공 신경망(Neural Network)를 사용하여 학습을 진행한다. 인공 신경망이란, 사람의 뇌가 외부 정보를 감지하고 학습하여 판단하는 과정을 본떠 만든 인공적인 신경망이다. 주어진 데이터가 여러 신경망의 층을 거치면서 데이터의 특징 추출 및 학습이 진행된다. 학습이 끝난 결과가 실제 답과 같아질 때까지, 학습을 진행하면서 모델을 완성한다. 대표적인 인공 신경망의 종류는 Fully-Connected Neural Network(FNN), Convolution Neural Network(CNN), Recurrent Neural Network(RNN)등이 있다. FNN은 입력으로 숫자가 나열된 배열을 사용한다. CNN은 이미지 분류 등에 사용되는 인공 신경망으로 입력으로 2차원 정사각 배열을 사용한다. RNN은 연속적인 시간 데이터(Serial Data)의 미래 예측에 사용되는 인공 신경망으로, 입력으로 연속적인 과거 데이터들을 사용한다. 이번 프로젝트에서는 CNN 방식을 이용한 모델을 설계할 것이다[3].

1.2 프로젝트의 학습적 의의 및 실용적 의의

디지털신호처리는 자연의 신호들을 이산신호로 변환하고, 변환한 데이터를 컴퓨터의 계산을 이용하여, 해당 신호들이 실제로 가지는 의미가 무엇인지 분석하여 파악하는 데에 의의를 둔다. 본 실습에서 IR-UWB Radar를 활용하여 “거리”에 관한 정보를 이산신호로 변환하고 이를 딥러닝을 이용하여 분석하는 과정을 거쳐, 신호가 현실에서 가지는 의미가 무엇인지를 파악하여 디지털신

호처리에 관해 학습한 내용을 적용하는 방법을 학습할 수 있다. 또한 이러한 실용적 학습을 통해 얻은 지식을 실생활에 적용해 볼 수 있다. 본 프로젝트에서는 “Air-HandWriting” 데이터를 분석하였지만, 사람의 생체 신호를 분석할 수도 있을 것이다. 예를 들어, 사람이 호흡할 때 폐의 팽창과 수축 작용에 의해서 IR-UWB Radar와의 거리가 변화함을 이용하여, 폐의 미세한 상하 운동을 감지하여 데이터를 수집하고 호흡 패턴을 RNN 방식을 이용하여 분석하여 각종 폐질환 발견과 예측에 적용할 수 있다[4]. 또한 암 조직과 정상 조직에 대한 Radar Data를 수집하고 CNN 모델을 학습 시켜 비파괴, 비침습, 비접촉 방식으로 인체 내부의 악성종양을 탐색하는 방안도 생각해 볼 수 있다[5]. 혹은, 출입구 등에 IR-UWB Radar를 부착해 놓고, 거리의 변화 감지를 이용하여 사람들의 출입을 관리하거나, 차의 후방에 부착하여 장애물과의 거리를 측정하여 주차에 도움을 줄 수도 있을 것이다.

2. 데이터 측정과 이미지 생성

2.1 IR-UWB Radar를 이용한 데이터 측정 방식

IR-UWB Radar를 책상에 올려놓고, 케이블로 인해 비뚤어지지 않도록 케이블은 노트북으로 눌러서 고정해 주고, 모든 모양에 대해 위에서 시작해서 아래로 50회, 아래에서 시작해서 위로 50회씩 수행하였다. 자세한 방법은 그림과 같다.

그림 1-1,2

그림 2-1,2

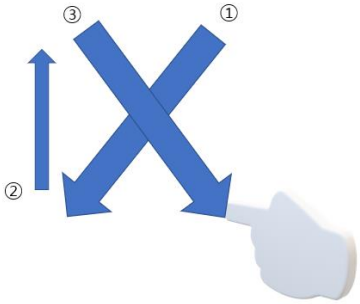

그림 3-1,2

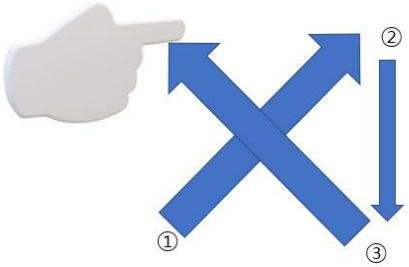

그림 1은 Circle, 그림 2는 Triangle, 그림 3은 X를 그리기 위한 각각의 방법이다.

진동이 없는 환경에서 진행되었으며, “Air-HandWriting”을 수행하는 중 Static Object와 손 외에 레이더에 감지된 물체는 없었다. Hilbert_History.mat 파일은 카테고리당 10개씩 저장되었으며, 한 개의 mat 파일은 10회의 “Air-HandWriting” 데이터를 포함한다. 따라서 카테고리당 총 100개의 “Air-HandWriting” 데이터를 얻을 수 있으며, 하나의 이미지 파일당 48,64, 혹은 80 frame이 사용된다. 메인 모델에서는 64 frame을 사용한다.

이렇게 IR-UWB Radar를 통해 얻은 미가공 데이터(Raw Data)를 저장한 후, 해당 신호의 DC 성분을 제거해 주고 추가로 Convolution을 이용하여 Band Pass Filtering을 수행해 준다. 이때 size는 입력값과 동일하게 나오도록 설정한다. 이렇게 Band Pass Filtering을 거친 Data에는 배경 등의 Static Object로 인한 반사파의 신호가 나타난다. 이를 Clutter라고 하는데, Threshold 설정을 통해 제거해 준다. 신호는 정현파의 합으로 이루어져 있기 때문에 여러 가지 peak가 나타난다. 따라서 신호의 Envelope만 얻기 위해 Filtered Signal을 Hilbert Transform 해준 후 크기 정보만 Hilbert_History에 저장해 준다. 그리고 DataCursor값이 데이터의 최대길이보다 커지면, 현재시간 정보를 저장해 놓은 변수를 이용하여 “Hilbert_History” 행렬을 Data-현재시간정보.mat로 저장한다.

2.2 IR-UWB Radar를 이용해 얻은 데이터 기반 이미지 생성 방식

```
FPS=20;
one_image_fr=3.2*FPS;
numofimage=10;
directory={"Circle","Triangle","X_Shape"};
shape={"O","T","X"};
```

다음과 같이 변수들을 설정해 준다. 이때 directory와 shape에 저장되는 string들은 실제 디렉토리 내의 폴더명과 동일해야 한다.

```
for shapenum=1:3
    PATH_from=strcat("C:\Users\DSP_Final_Project\Final_Project\Data\",directory{shapenum},"\");
    PATH_to=strcat("C:\Users\DSP_Final_Project\Final_Project\image\",shape{shapenum},"\");
    filelist=dir(PATH_from);
    filelist(1:2)=[];
    imgnum=1;
    imgnamehead=directory(shapenum);
```

Project 폴더 내부의 Data 폴더로 끝나는 디렉토리 내에서 Hilbert_History.mat 파일을 불러온다. 이때 디렉토리 내부 파일들을 filelist에 저장할 때, 첫 번째와 두 번째에는 각각 “.”과 “..”이 저장되기 때문에, filelist(1:2)=[];와 같이 설정하여 해당 값들을 지워준다. imgnum은 생성되는 이미지의 파일명을 다르게 저장하기 위한 변수이고, imgnamehead는 생성되는 이미지의 모양에 따라 이미지 파일 맨 앞에 directory와 동일한 문자열을 붙여주기 위한 변수이다. for문을 이용해 3회 반복하는데, 이를 통해서 한 번의 실행으로 “O”, ”T”, ”X” 모두의 이미지를 생성할 수 있다.

```
for i=1:size(filelist,1)
    load(strcat(PATH_from,filelist(i).name));
    for k=1:numofimage
```

```
        tempX=Hilbert_History((k-1)*one_image_fr+1:k*one_image_fr,1:one_image_fr);
        for a=1:one_image_fr
            tempX(a,:)=normalize(tempX(a,:),'range').*255;
        end
        tempX=uint8(tempX);
```

먼저 for문을 이용하여 i=1에서 i=10까지 반복하도록 한다. filelist의 i번째 행의 “name” 필드를 이용해 해당하는 Hilbert_History.mat를 불러온다. 변수 tempX에 해당 Hilbert_History의 값들을 one_image_fr의 값과 동일한 정사각행렬 size로 슬라이싱하여 저장해주는데, 이 때 k의 값에 따라 선택하는 행의 index가 달라진다. 예를 들어 k가 1이면 처음 Air-HandWriting을 했을 때의 데이터가, k가 5이면 5번째로 Air-HandWriting을 했을 때의 데이터가 슬라이싱되어 저장된다. 이렇게 얻은 데이터의 값들은 너무 작아서 이미지로 표현하기 어렵다. 따라서 normalize 함수를 사용하여 그 후 tempX의 각 행의 모든 열을 [0,1] 범위 내로 정규화 시켜주고, 255를 곱하는 과정을 for문을 사용하여 모든 행에 대해서 반복해 준다. 그 후 tempX를 uint8 함수를 사용하여 8bit unsigned integer 형태로 변환하여 주면, 행렬의 모든 값들이 [0,255] 범위 내의 정수들로 이루어진 흑백 이미지 데이터가 완성된다.

```
if imgnum<10
    fname=strcat(imgnamehead{1},"000",num2str(imgnum),".png");
elseif imgnum<100
    fname=strcat(imgnamehead{1},"00",num2str(imgnum),".png");
elseif imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
```

이후 변수 imgnum의 값에 따라 파일명을 지정해 준다. 파일명은 imgnameheadxxxx.png와 같은 식으로 저장될 것이므로, imgnum의 자릿수가 바뀔 때마다 저장하는 방식을 다르게 해준다. imshow를 이용해 tempX를 이미지화해서 보여주고, disp를 이용해서 해당 파일의 이름을 보여준다. 빠른 실행이 필요할 경우 주석으로 처리하여도 이미지 저장에는 지장이 없다. imwrite 함수를 활용하여 tempX를 PATH_to에 저장한 디렉토리로 fname라는 이름의 png 파일로 저장한다. 저장된 이미지의 예시는 다음과 같다.


그림 4-1


그림 4-2


그림 5-1


그림 5-2


그림 6-1


그림 6-2

그림 4-1,2는 그림 1-1,2의 방식, 그림 5-1,2는 그림 2-1,2의 방식, 그림 6-1,2는 그림 3-1,2의 방식으로 얻은 이미지이다. 바로 알아보기는 어려우나, 데이터 측정방식을 생각해보면 이미지와 모양을 매치할 수 있다. 이미지에 좌표축을 대입해보면 다음과 같이 나타낼 수 있다.

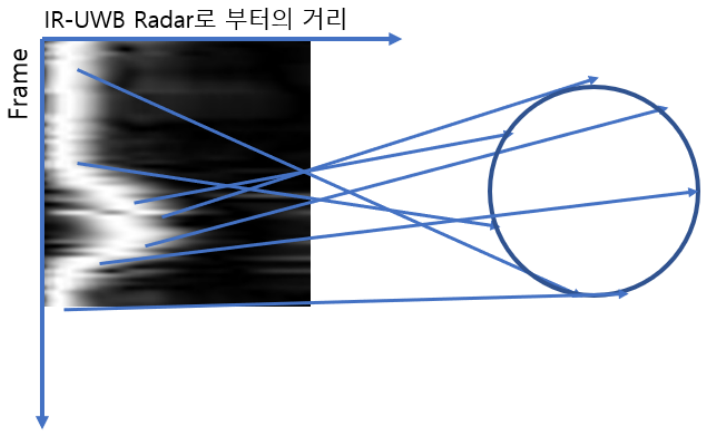

그림 7-1

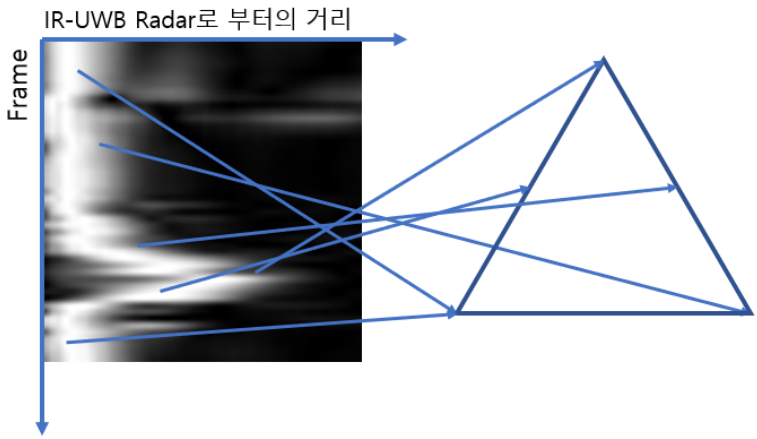

그림 7-2

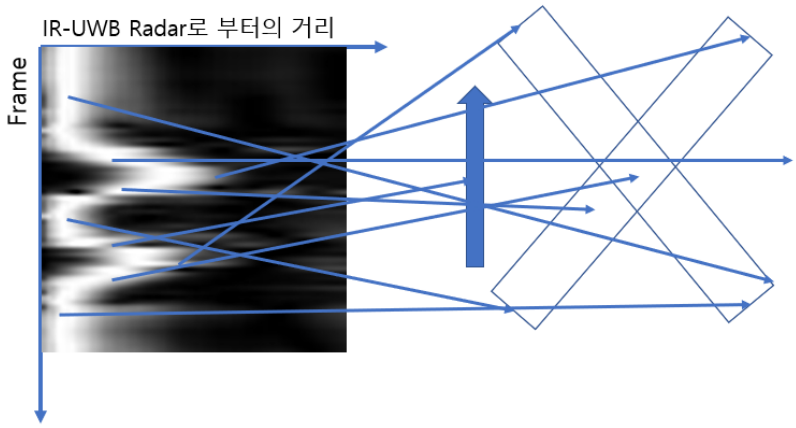

그림 7-3

그림 1-2, 2-2, 3-2와 같은 방법으로 “Air-HandWriting”을 수행한 경우에도 동일하게 이미지의 원형을 쉽게 유추할 수 있다.

이렇게 원본 이미지를 저장한 후, 추가적인 Image Processing을 통해 각 디렉토리 별로 500개

의 추가 이미지를 확보하였다. 2 Level Quantization, Brighten, Darken, Gaussian Noise, Salt and Pepper Noise를 적용하였다. 각각의 알고리즘은 다음과 같다.

2 Level Quantization

```
frame=64;
for quant=1:frame
    for ization=1:frame
        if tempX(quant,ization)<128
            tempX(quant,ization)=0;
        else
            tempX(quant,ization)=255;
        end
    end
end
```

2 Level Quantization을 수행하는 알고리즘이다. tempX의 각 행과 열의 성분에 대해서, 성분의 값이 128보다 작은 경우 0으로, 크거나 같은 경우 255로 변경하여 저장한다. 이를 반복한다.

Brighten & Darken

```
tempX=tempX.*2;   %Brighten
tempX=tempX.*0.5; %Darken
```

Brighten 혹은 Darken을 수행하는 알고리즘이다. tempX에 1보다 큰 양의 정수를 곱하면 밝게, 1보다 작은 양의 정수를 곱하면 어둡게 변한다.

Gaussian-Noise

```
frame=64;
for add=1:frame
    for gaussian=1:frame
        tempX(add,gaussian)=tempX(add,gaussian)+50*randn(1);
    end
end
```

Gaussian Noise를 첨가하는 알고리즘이다. tempX의 각 행과 열의 성분에 대해서 표준정규분포로부터 얻은 random 난수에 대해서 50을 곱한 값을 더해준다. 이는 적절한 Noise를 얻기 위해 임의로 지정한 것이다. 이를 반복한다.

Salt and Pepper Noise

```
frame=64;
for salt=1:frame
    for pepper=1:frame
        determine=rand(1);
        if (0<determine) && (determine<=1/8)
            tempX(salt,pepper)=0;
        elseif (1/8<determine) && (determine<=1/4)
            tempX(salt,pepper)=255;
        end
    end
end
```

Salt and Pepper Noise를 첨가하는 알고리즘이다. Uniform 분포로부터 random 난수를 얻어, 이 값의 크기에 따라 tempX(salt,pepper)는 원본값을 유지하거나, 흰색 또는 검정색으로 변환한다. for문을 이용하여 반복한다.

이 알고리즘들을 함수로 만들어, 원본 이미지를 저장한 후에 바로 추가적인 Processing을 수행

할 수 있도록 코드를 작성해 주었다. 저장된 이미지의 예시는 다음과 같다.

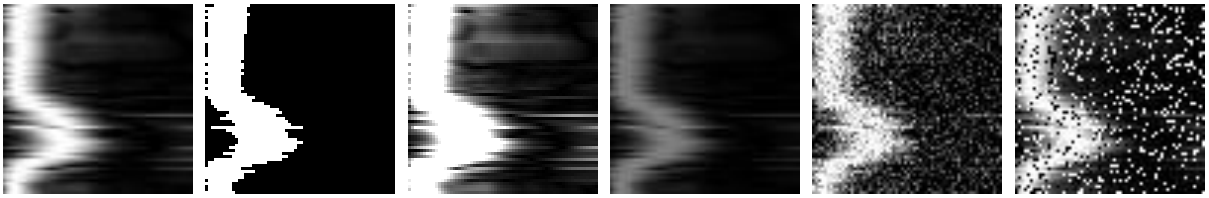

그림 8 그림 9 그림 10 그림 11 그림 12 그림 13

그림 8은 원본, 그림 9는 2 Level Quantization, 그림 10은 Brighten, 그림 11은 Darken, 그림 12는 Gaussian Noise, 그림 13은 Salt and Pepper Noise 알고리즘이 각각 적용된 모습이다. 추가적인 Image Processing을 거친 이미지를 저장할 때 주의할 점은, 해당 이미지들의 파일명이 중복되지 않도록 이미지를 저장해야 한다는 것이다. 원본 이미지가 0~100번 대를 부여받았으므로, Quantization을 거친 이미지를 101~200, Brighten을 201~300, Darken을 301~400, Gaussian Noise를 401~500, Salt and Pepper Noise를 501~600번대로 fname에 저장해주면 된다.

```
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(100+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(100+imgnum),".png");
end
```

위의 예시는 2 Level Quantization을 수행한 image의 파일명을 정하는 코드이다. 위와 같이 num2str 내부에서 대역을 잘 설정해 주면 중복을 피할 수 있다.

```
%% Data preparation
dirpath="C:\Users\DSP_Final_Project\Final_Project\image\";
imgdatas=imageDatastore(dirpath,"IncludeSubfolders",true,"FileExtensions" ...
    ,".png","LabelSource","foldernames");
%% Divide into Train Set and Validation Set
numTrainFiles=450;
[imdsTrain,imdsValidation]=splitEachLabel(imgdatas,numTrainFiles,'randomized');
```

이미지를 생성한 후에는 이미지를 학습 데이터와 테스트 데이터로 분류해야 한다. 먼저, 불러올 이미지가 있는 디렉토리를 dirpath 변수에 저장한다. 그 후 ImageDataStore 함수를 사용하여 dirpath에 저장된 디렉토리, 즉 프로젝트 폴더 내의 image 폴더의 하위 폴더를 범위에 포함하고, 폴더 내에서 png파일을 불러오고, 라벨 데이터를 폴더명으로 할당하여 imgdatas에 저장한다. 그 후 훈련에 사용할 이미지 개수를 변수에 저장해 준 후, splitEachLabel함수를 사용하여 imgdatas를 무작위로 Training Set 혹은 Validation Set로 나눈다. 예를 들어, 디렉토리 별로 600개의 파일이 있는데 변수에 저장한 Training Set의 개수가 450이라면 Training Set은 450개, Validation Set은 150개가 된다.

3. Image Classification을 위한 CNN 모델 설계

3.1 CNN의 Feature Learning Layer의 동작 원리

데이터가 입력되면, 제일 먼저 convolution2dLayer 함수를 사용하여 Convolution Filter를 적용하게 된다. 본 모델은 3 x 3 size의 필터를 3개 적용하였다. convolution2dLayer 함수의 stride는 기본적으로 1로 설정되어 있다. 이 때 'Padding' 명령어로 인해 Input 배열의 네 모서리에 0으로 이루어진 행과 열이 붙게 되는데, 'same' 명령어를 사용하였으므로 Padding은 입력과 출력의 크기가 같도록 행해진다. 따라서 결과 배열은 Input 배열과 같은 크기로 출력될 것이다. 예시를 통해 확인해보자. Input 배열의 크기를 3 x 3, 필터의 크기를 3 x 3으로 가정했을 때, Padding을 수행한 배열에 대해 Convolution을 수행하는 알고리즘은 다음과 같다.

```
2D Convolution when Input Layer had padded
m= row size before padding, n=column size before padding, k=filtersize, result= k by k size
for i=1 : m
    for j=1 : n
        temp = InputLayer(i : i+k-1, j : j+k-1)
        Result(i , j) = sum(sum(temp.*filter))
    end for
end for
```

이를 적용하여 보면, 다음의 그림과 같은 결과를 얻을 수 있다.

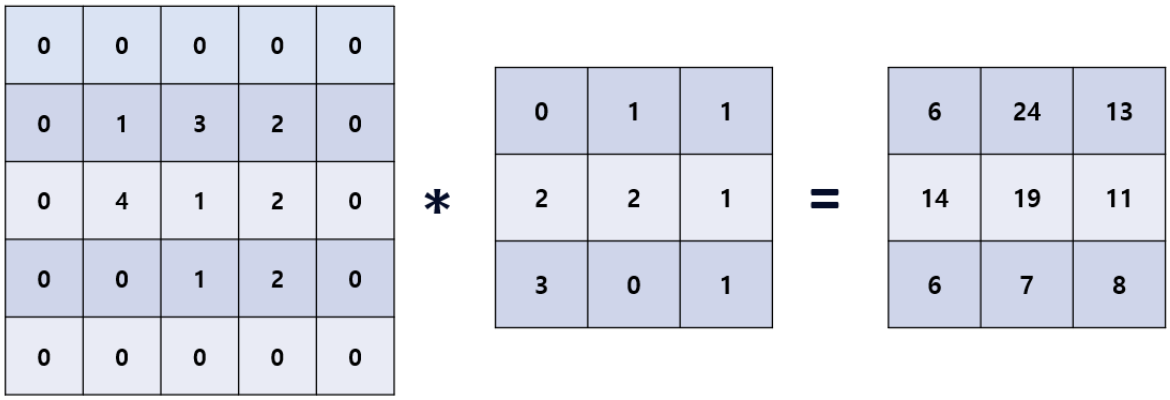

그림 14 – Padding을 거친 2D Convolution의 예시

그림 14의 Input과 Filter를 이용하여 알고리즘을 실제로 실행하여 보면 동일한 결과를 얻을 수 있다. 만약 Input 배열에 Padding을 하지 않고 Convolution을 하고자 한다면 알고리즘은 다음과 같이 수정해야 한다.

```
2D Convolution when Input Layer had not padded
m= row size, n=column size, k=filtersize, result= m-k+1 by n-k+1 size
for i=1 : m-k+1
    for j=1 : n-k+1
        temp = InputLayer(i : i+k-1, j : j+k-1)
        Result(i , j) = sum(sum(temp.*filter))
    end for
end for
```

알고리즘을 실행하여 보면, 다음의 그림과 같은 결과를 얻는다.

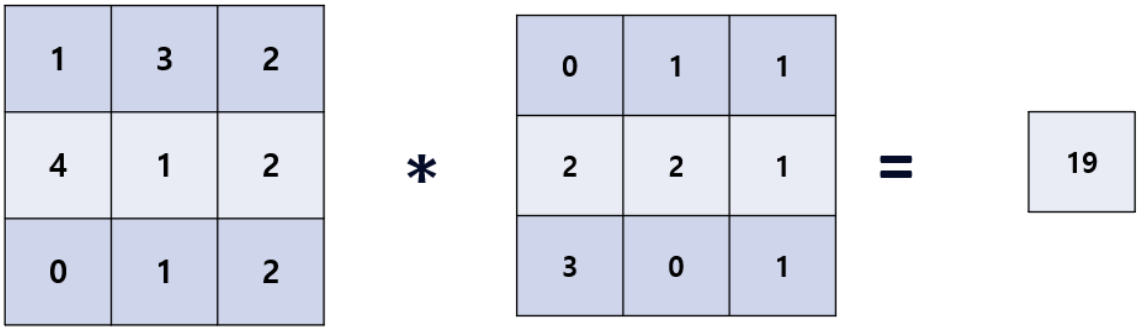

그림 15 – Padding 처리를 하지 않은 2D Convolution의 예시

Padding을 하지 않으면 그림 15와 같이 Input 배열과 Output 배열의 크기가 달라진다. 따라서 원하는 결과를 얻기 위해서는 Padding을 적용해야 한다.

Convolution을 거친 각 채널에 대해서 Batch normalization을 적용했다. Batch normalization은 모든 관측값에서 학습 데이터의 부분집합을 독립적으로 정규화하는 것인데, 이는 CNN 모델의 훈련 속도를 높이고, 역전파를 통한 신경망의 초기화에 대한 민감도를 줄이는 효과를 준다[6].

Batch normalization을 마친 데이터에 대해서 ReLU activation function을 적용했다. ReLU Layer는 입력값의 각 성분에 대해 0보다 작은 값은 모두 0으로 설정하고, 0보다 큰 값이어야만 전달되게 하는 역할을 한다. ReLU activate function은 입력값이 음수인 경우 기울기가 0이 되는 문제가 있긴 하지만, 학습 속도가 빠르고 기울기 소실 문제를 방지하는 장점에 주목해 적용한다[7].

Activation layer를 거친 데이터에 대해서 Max Pooling을 수행한다. Pooling은 입력받은 Data의 크기를 줄여 불필요한 정보를 버리고 손실에 의해 정의된 Global Feautre를 남기는 역할을 한다. 이는 메모리 절약, 모델 파라미터 수 감소에 따른 연산량 감소와 학습 속도 증가라는 장점을 가지고 있다. 또한 파라미터 수의 감소는 Overfitting을 방지한다[8]. Max Pooling은 이러한 Pooling 기법의 하나이며, 부분 영역에서 최댓값을 뽑는 Pooling 방식을 의미한다. 위와 같은 다양한 이유로 Pooling을 적용한다. Max Pooling을 적용하는 방법을 4 by 4 size의 행렬을 예시로 들어 확인해 보았다. 알고리즘은 다음과 같다.

	Maxpooling
1	L = length(InputLayer) , fsz = filtersize , std = stride, Result = (L – std) x (L – std) size
2	**for** i = 1 : L – std
3	**for** j = 1 : L – std
4	temp = InputLayer((i-1)*std+1 : i*std+fsz-std , (j-1)*std+1 : j*std+fsz-std)
5	tempmax = max(max(temp))
6	Result(i , j) = tempmax
7	end **for**
8	end **for**

이 알고리즘을 각각 stride=1일 때와 stride=2일 때에 대해 적용한 결과는 다음 그림과 같다.

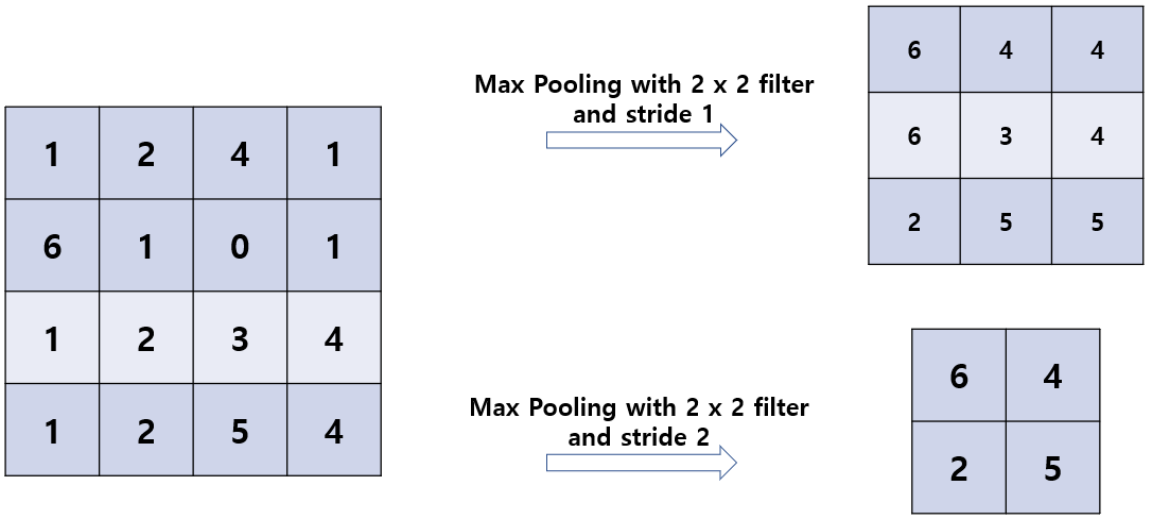

그림 16 – Max Pooling의 예시

stride는 Max Pooling을 수행할 때 filter가 한 번 이동할 때 몇 칸을 이동하는지를 의미한다. stride 값을 설정할 때는 stride 값을 키우거나 줄였을 때 정보의 손실 정도와 메모리의 절약 정도 등의 따라오는 장단점을 잘 고려하여 stride 값을 설정해야 한다. 설계한 CNN 모델에서는 stride값을 2로 설정하였다.

3.2 CNN의 Classification Layer의 동작 원리

Featuring Layer를 여러 개 거친 후, 데이터는 Classification Layer에 진입한다. 이 때 fullyConnectedLayer 함수는 3D 배열을 평탄화 하여 1차원 배열로 만들고, 입력해준 output size에 맞춰 배열을 재배열한 후,softmaxLayer함수를 사용하여 Softmax Classification을 수행한다. Softmax Classification을 사용하는 이유는, 모양에 1, 2, 3과 같이 index를 부여하여 판단하는 경우에는 결과값이 소수점이 존재하는 경우에는 어느 카테고리로 판단해야 하는지 명확하지 않은 경우가 발생하는데, 이를 방지하기 위해서이다. Softmax 연산을 하면 Layer의 출력 값은 합이 1인 양수로 정규화 된다[9]. 각각의 출력 값은 카테고리 별로 얼마나 일치하는지에 대한 비율을 확률로 표현한 것이다. 이 때 가장 높은 값을 가진 카테고리가 최종 결과로 선정된다.

3.3 학습을 위해 설계한 CNN 모델

앞서 서술한 원리를 이용해 이미지 데이터 학습을 하기 위해 설계한 CNN 모델의 이미지는 다음과 같다.

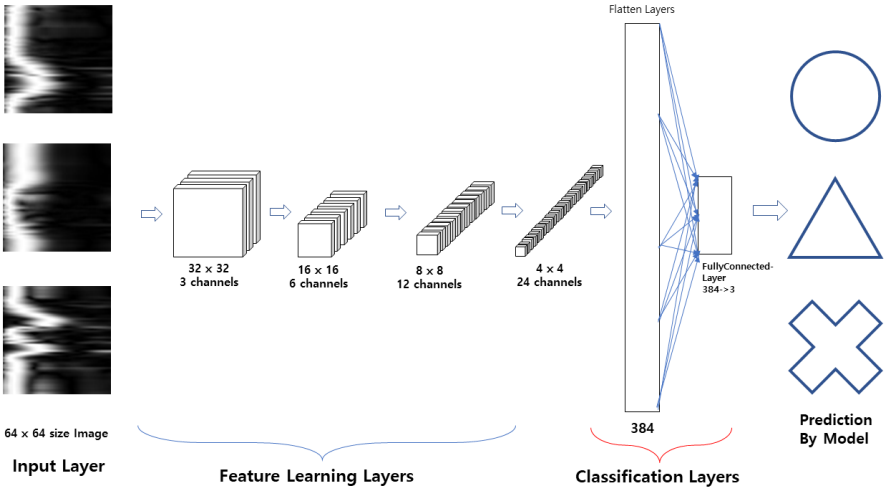

그림 17 - 입력 데이터의 크기가 64 x 64일 때의 CNN 모델

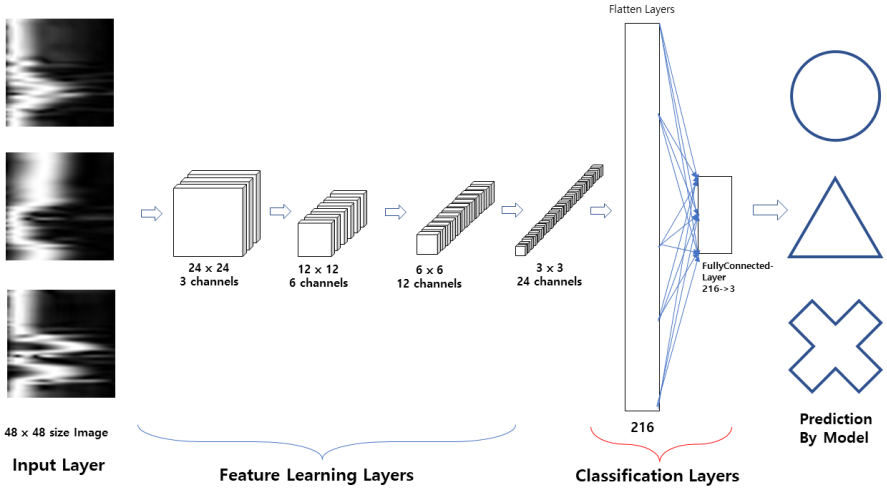

그림 18 - 입력 데이터의 크기가 48 x 48일 때의 CNN 모델

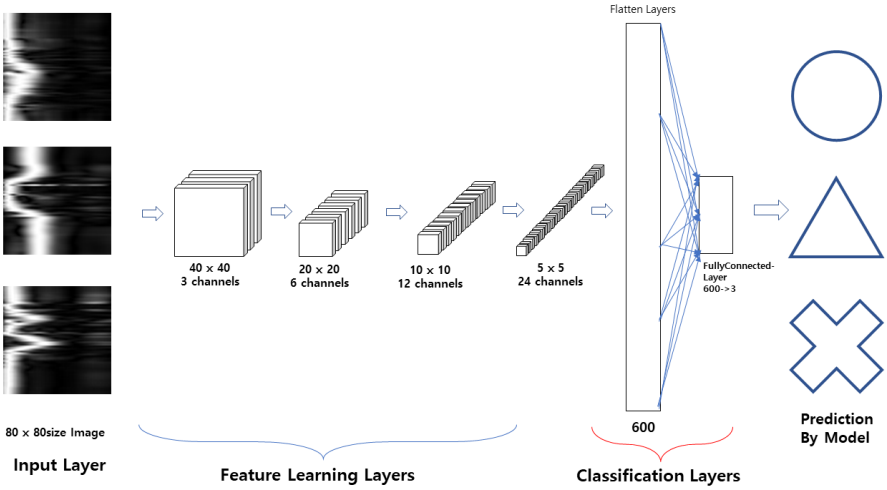

그림 19 - 입력 데이터의 크기가 80 x 80일 때의 CNN 모델

4. 결론

4.1 학습 진행 방식

딥러닝 모델의 학습을 진행하기 위해서 먼저 Training Option을 다음과 같이 설정해준다.

최적화 함수	SGDM 함수 사용
초기 학습률	0.01
Epoch 최댓값	20
Shuffle 방식	각 Epoch 마다
검증에 사용할 데이터셋	imdsValidation
검증 빈도	10
진행 상황을 명령 창에 표시	false
Plots	training-progress (훈련 진행 상황 Plot O)

이때 주의할 점은, 초기 학습률을 너무 낮게 설정하면 훈련 시간이 오래 걸릴 수 있고, 너무 높으면 최적의 결과보다 못한 값에 도달하거나 발산할 수 있다. 따라서 Deep Learning Toolbox에서 제공하는 최적화 함수별 디폴트 값을 사용하는 게 가장 안정적이다. SGDM 함수의 디폴트 값은 0.01이므로, 문제없이 설정되었다고 볼 수 있다. 또한, MaxEpochs 값을 너무 작게 설정하면 underfitting이 일어나고 학습이 제대로 이루어지지 않을 수 있다. MaxEpochs 값을 크게 설정하면 일반적으로 정확도가 높아지지만, 너무 크게 설정하면 overfitting이 일어날 수 있으므로 적정한 값을 설정해 주어야 한다[10]. 그 후 trainNetwork 함수를 사용하여, 이미지 데이터 셋을 입력 받아, CNN 모델을 Option 설정값에 따라 훈련한다. 이 결과는 변수 net에 저장한다.

4.2 CNN 모델의 학습 정확도 계산

학습 데이터의 정확도는 다음과 같은 알고리즘으로도 계산해 볼 수 있다.

	Accuracy
1	YPred = Prediction, YVal = Validation, Totnum= Total number of Data, cnt = count
2	**for** n = 1 : totnum
3	**if** YPred(n) == YVal(n)
4	cnt = cnt + 1
5	end **if**
6	end **for**
7	$Accuracy = \frac{cnt}{totnum}$

코드상에서는 $Accuracy = sum(\text{YPred} == \text{YValidation})/\text{numel(YValidation)}$ 이라는 코드를 입력하는데, YPred == YValidation은 0 또는 1로 이루어진 훈련 파일 개수와 동일한 길이의 벡터이며 0은 불일치, 1은 일치를 의미한다. 따라서 sum 함수는 일치한 데이터의 개수를 반환하고, numel(YValidation)은 YValidation의 길이를 반환하는데, 이는 총 데이터의 개수와 동일하다. 따라서 둘을 나누어 주면 모델의 정확도를 얻을 수 있다.

4.3 각 모델의 Epoch별 학습 진행도 및 정확도

각 CNN 모델에 대해 학습을 진행한 결과는 다음과 같다.

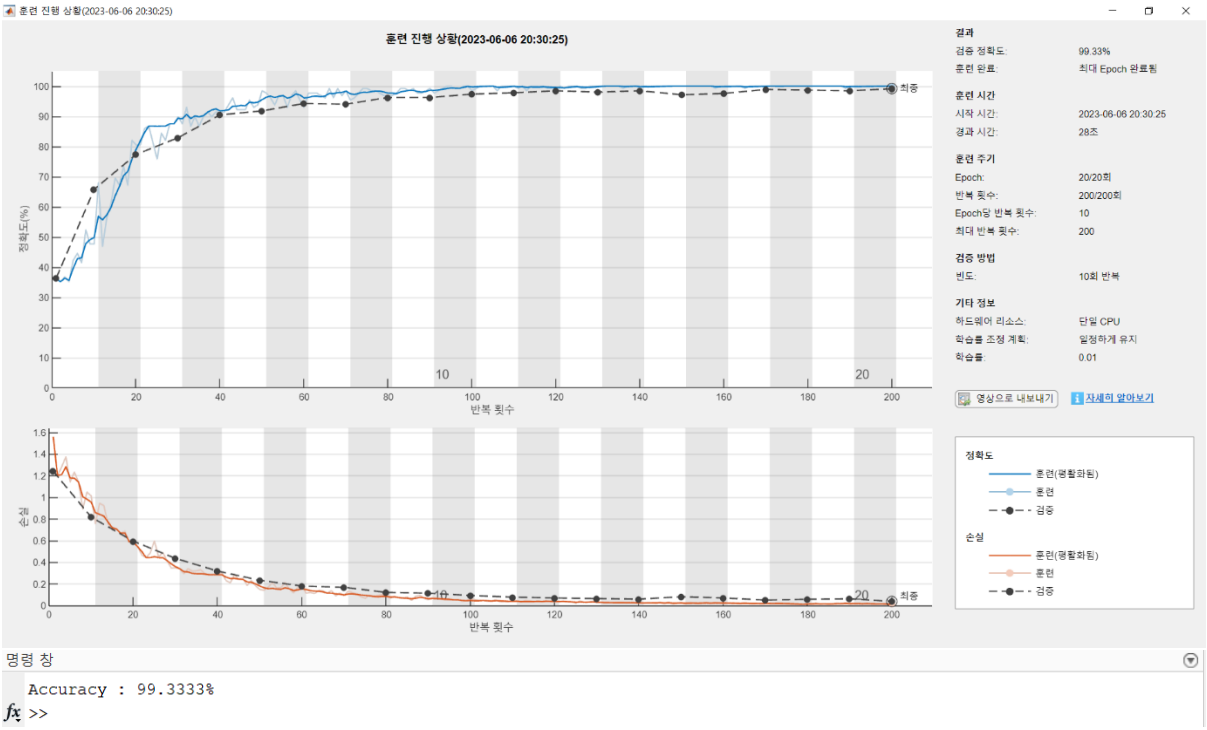

그림 20 - 입력 데이터의 크기가 64 x 64일 때의 학습 결과

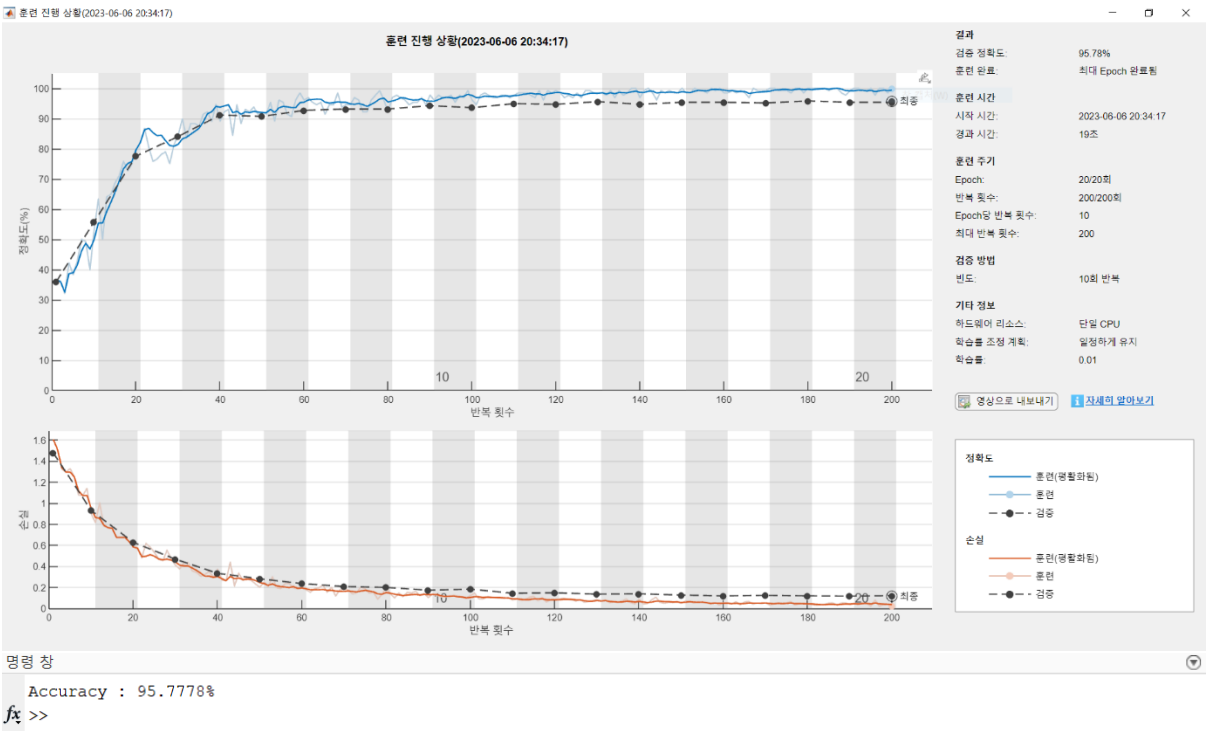

그림 21 - 입력 데이터의 크기가 48 x 48일 때의 학습 결과

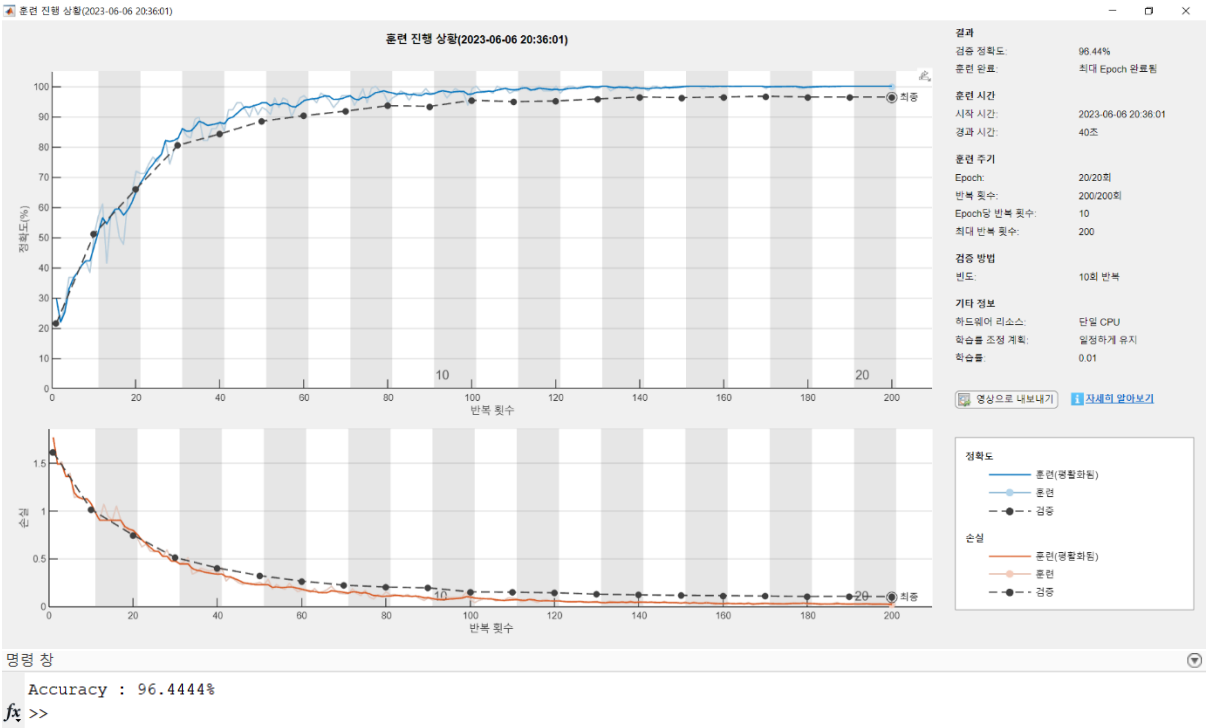

그림 22 - 입력 데이터의 크기가 80 x 80일 때의 학습 결과

5. 부록

(1) make_image.m

```
clc;clear;close all;

FPS=20;
one_image_fr=3.2*FPS;
numofimage=10;
directory={"Circle","Triangle","X_Shape"};
shape={"O","T","X"};

for shapenum=1:3
    PATH_from=strcat("C:\Users\DSP_Final_Project\Final_Project\Data\",directory{shapenum},"\");
    PATH_to=strcat("C:\Users\DSP_Final_Project\Final_Project\image\",shape{shapenum},"\");
    filelist=dir(PATH_from);
    filelist(1:2)=[];
    imgnum=1;
    imgnamehead=directory(shapenum);
    for i=1:size(filelist,1)
        load(strcat(PATH_from,filelist(i).name));
        for k=1:numofimage
            tempX=Hilbert_History((k-1)*one_image_fr+1:k*one_image_fr, ...
                1:one_image_fr);
            for a=1:one_image_fr
                tempX(a,:)=normalize(tempX(a,:),'range').*255;
            end
            tempX=uint8(tempX);
            if imgnum<10
```

```
                fname=strcat(imgnamehead{1},"000",num2str(imgnum),".png");
            elseif imgnum<100
                fname=strcat(imgnamehead{1},"00",num2str(imgnum),".png");
            elseif imgnum<1000
                fname=strcat(imgnamehead{1},"0",num2str(imgnum),".png");
            else
                fname=strcat(imgnamehead{1},num2str(imgnum),".png");
            end
            imshow(tempX);
            disp(fname);
            imwrite(tempX,strcat(PATH_to,fname),'png');
            quantization(tempX,imgnum,imgnamehead,PATH_to,one_image_fr)
            brighten(tempX,imgnum,imgnamehead,PATH_to)
            darken(tempX,imgnum,imgnamehead,PATH_to)
            gaussian_noise(tempX,imgnum,imgnamehead,PATH_to,one_image_fr)
            salt_and_pepper(tempX,imgnum,imgnamehead,PATH_to,one_image_fr)
            imgnum=imgnum+1;
        end
    end
end
```

(2) quantization.m

```
function quantization(tempX,imgnum,imgnamehead,PATH_to,frame)
for quant=1:frame
    for ization=1:frame
        if tempX(quant,ization)<128
            tempX(quant,ization)=0;
        else
            tempX(quant,ization)=255;
        end
    end
end
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(100+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(100+imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
end
```

(3) brighten.m

```
function brighten(tempX,imgnum,imgnamehead,PATH_to)
tempX=tempX.*2;
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(200+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(200+imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
end
```

(4) darken.m

```
function darken(tempX,imgnum,imgnamehead,PATH_to)
tempX=tempX.*0.5;
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(300+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(300+imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
end
```

(5) gaussian_noise.m

```
function gaussian_noise(tempX,imgnum,imgnamehead,PATH_to,frame)
for add=1:frame
    for gaussian=1:frame
        tempX(add,gaussian)=tempX(add,gaussian)+50*randn(1);
    end
end
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(400+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(400+imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
end
```

(6) salt_and_pepper.m

```
function salt_and_pepper(tempX,imgnum,imgnamehead,PATH_to,frame)
for salt=1:frame
    for pepper=1:frame
        determine=rand(1);
        if (0<determine) && (determine<=1/8)
            tempX(salt,pepper)=0;
        elseif (1/8<determine) && (determine<=1/4)
            tempX(salt,pepper)=255;
        end
    end
end
if imgnum<1000
    fname=strcat(imgnamehead{1},"0",num2str(500+imgnum),".png");
else
    fname=strcat(imgnamehead{1},num2str(500+imgnum),".png");
end
imshow(tempX);
disp(fname);
imwrite(tempX,strcat(PATH_to,fname),'png');
end
```

(7) Classification.m

```
clear;clc;close all;
```

```
%% Data preparation
dirpath="C:\Users\DSP_Final_Project\Final_Project\image\";
imgdatas=imageDatastore(dirpath,"IncludeSubfolders",true,"FileExtensions" ...
    ,".png","LabelSource","foldernames");

%% Divide into Train Set and Validation Set
numTrainFiles=450;
[imdsTrain,imdsValidation]=splitEachLabel(imgdatas,numTrainFiles,'randomized');

%% Modeling
layers=[
    % Use 64 by 64 by 1 Layer
    imageInputLayer([64 64 1]);
    convolution2dLayer(3,3,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    % Use 32 by 32 by 3 Layer
    convolution2dLayer(3,6,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    % Use 16 by 16 by 6 Layer
    convolution2dLayer(3,12,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    % Use 8 by 8 by 12 Layer
    convolution2dLayer(3,24,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    % Use 4 by 4 by 24 Layer
    fullyConnectedLayer(3)
    softmaxLayer
    classificationLayer];

options=trainingOptions('sgdm','InitialLearnRate',0.01, ...
    'MaxEpochs',20,'Shuffle','every-epoch','ValidationData', ...
    imdsValidation,'ValidationFrequency', ...
    10,'Verbose',false,'Plots','training-progress');

net=trainNetwork(imdsTrain,layers,options);

YPred=classify(net,imdsValidation);
YValidation=imdsValidation.Labels;

accuracy=sum(YPred==YValidation)/numel(YValidation);
disp("Accuracy : "+num2str(accuracy*100)+"%")
```

6. 참고문헌

[1]이진섭, 윤정원.(2019).IR-UWB 레이다를 이용한 모션 인식에 관한 연구.한국전자파학회논문지,30(3),236-242.

[2] M. Chen, G. AlRegib and B. -H. Juang, "Air-Writing Recognition—Part I: Modeling and Recognition of Characters, Words, and Connecting Motions," in IEEE Transactions on Human-Machine Systems, vol. 46, no. 3, pp. 403-413, June 2016, doi: 10.1109/THMS.2015.2492598.

[3]수업자료 참조, Data Preparation Guide.

[4] 최정우, 이유나, 조석현, 임영효, 조성호.(2017).IR-UWB 레이더 센서 기반 수면 효율 측정 알고리즘.한국통신학회논문지,42(1),214-217.

[5] 김지훈, 이지훈, 정신기, 손경식, 김유성.(2022).IR-UWB Radar를 통한 비파괴 · 비침습 · 비접촉 악성 종양 탐지.한국정보과학회 학술발표논문집,(),2054-2056.

[6]https://kr.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html

[7] Mastromichalakis, Stamatis. "ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance." arXiv preprint arXiv:2012.07564 (2020).

[8] 이성주, 조남익.(2020).Spectral Pooling: DFT 기반 풀링 계층이 보여주는 여러 가능성에 대한 연구.한국방송미디어공학회 학술발표대회 논문집,(),87-90.

[9] https://kr.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html

[10] Sinha, Shivam, et al. "Epoch determination for neural network by self-organized map (SOM)." Computational Geosciences 14 (2010): 199-206.