

Lab 10. Team Project

- Sign Language Recognition

20150688 정연수

20160272 박지윤

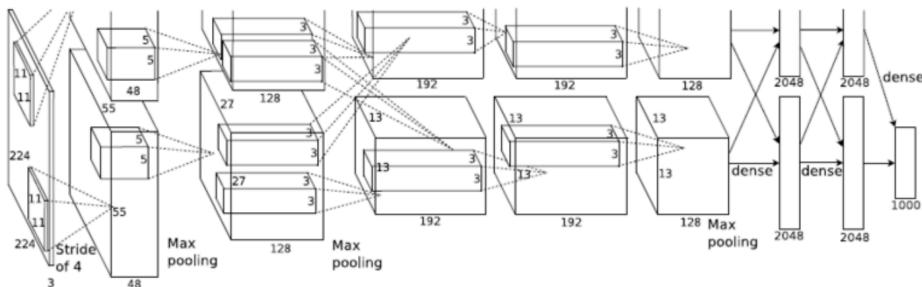
1. Objective

수화는 소리가 아닌 손짓을 이용해서 뜻을 전달하는 언어의 일종이다. 대부분 청각 장애인들이 사용한다. 하지만, 비장애인들에게 수화는 매우 낯선 영역 중 하나이고, 사전 지식도 거의 없는 편이다. 우리 조는 이처럼 수화에 익숙하지 않은 사람과 수화를 사용하는 사람이 서로 불편함 없이 의사소통을 할 수 있게 도와주고 싶다는 점에서 착안하여 이 프로젝트를 구상하게 되었다. 우리가 수업 시간에 배운 딥러닝을 통한 image recognition 기술을 발전시켜 사용하면 실시간으로 수화 동작을 인식하여 단어를 띄워주고, 그 단어를 통해 완성된 문장을 최종적으로 보여주는 것이 프로젝트의 목표이다. 한국 수화를 사용하였지만, 표현의 통일성과 용이성을 위해 영어로 번역하였다. (매트랩에서 한국어를 지원하지 않는다.) 프로젝트의 단계는 다음과 같다.

- ① 딥러닝을 이용하여 움직이는 동작이 포함된 수화를 올바르게 인식한다.
- ② 수화 단어들을 알맞은 문법 순서에 따라 배열하여 문장을 완성한다.
- ③ 완성된 문장을 소리나게 읽는다.

2. Theoretical Background

① Alexnet



[그림 2-1] Alexnet 의 구조

Alexnet은 2012년 ILVRC(ImageNet Large Scale Visual Recognition Challenge, 이미지 분류 경진 대회)에서 15.4%를 기록해 2위(26.2%)를 큰 폭으로 따돌리고 1위를 차지하였다. 총 8개의 레이어로 구성되어 있으며, 처음 5개는 convolutional layer, 다음 3개는 fully-connected layer이다. 이러한 레이어들은 하나의 이미지에 대해 각각 독립적으로 특징을 추출하며 필터들을 학습시킨다. GPU 연산을 고려하여 두 개의 병렬 네트워크로 구성하였다는 점, 활성함수로는 ReLU를 사용하였고, Overlapping

pooling 을 사용, 그리고 활성함수를 적용하기 전에 normalization 을 적용해 일반화 능력을 높인 점이 특징이다. 1~2 번째 fully-connected layer 에 50% 확률의 dropout 을 사용하였는데, dropout 이란 신경망 간의 연결을 랜덤하게 끊어 빠른 학습을 가능하게 하고 overfitting 을 줄여준다.

② English sentence structure

한국에서는 영어를 가르칠 때 영어 문장을 5 형식으로 나눈다. S 를 주어, V 를 동사, C 를 보어, 그리고 O 를 목적어라고 하자.

1 형식 : S + V

2 형식 : S + V + C

3 형식 : S + V + O

4 형식 : S + V + I.O (간접 목적어) + D.O (직접 목적어)

5 형식 : S + V + O + O.C (목적격 보어)

이 중 우리는 1 형식부터 3 형식 까지의 문장을 완성하는 것을 목표로 하였다.

3. Experimental Method

① Data acquisition

A. 많이 쓰는 수화 동작을 찾아본 후 연습한다. 이 때 문장의 구성 요소 중 우리가 다룰 1~3 형식 문장을 보다 많이 만들기 위해 명사, 동사, 형용사의 개수를 적절하게 (동사와 형용사는 비슷한 개수로, 명사는 주어와 목적어로 쓰이므로 동사와 형용사보다는 많게 준비하기) 맞춘다.

B. Webcam 으로 수화 동작을 찍으며 dataset 을 모은다.

② Train & Detection

A. 얻은 dataset 을 바탕으로 Alexnet 을 이용하여 training 시킨다.

B. Webcam 으로 실시간으로 수화 동작을 해보며 우리가 training 시킨 수화 동작을 잘 인식하는지 확인한다.

③ Make sentence

A. 문장은 blank 상태(아무 동작 안하고 있을 때)가 아닐 때 시작하며, 그 후 다시 blank 상태가 되면 문장이 끝나는 것으로 인식한다.

B. 인식한 word 들을 바탕으로 1, 2, 3 형식 문장을 구분하여 문장을 만든 후 화면에 띄운다.

④ Sentence to speech

A. 문장을 speaker 를 통해 소리 나게 읽는다. 이 때, 매트랩 내장함수인 tts 함수를 이용한다.

4. Result

① Data acquisition

우리는 수화동작을 직접 해보고 이를 webcam 으로 촬영하여 dataset 을 만들었다. Dataset 을 모으기 전, detection 이 잘 되도록 background 를 흰색의 품보드로 설정하였다.

A. Code 설명

```

for k=1:100
    img = snapshot(cam);
    image(img);
    imwrite(img, sprintf('sign_language\\you\\image_%04d.jpeg', k), 'jpeg');
    pause(0.25);
end

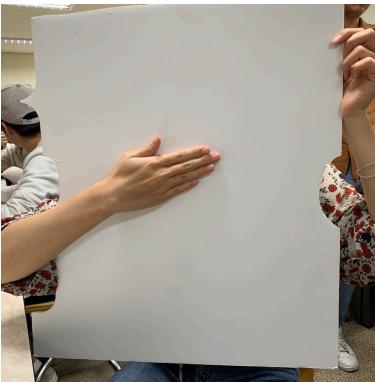
```

위 코드는 dataset 을 sign_language 폴더 안에 저장하는 과정을 보여준다. sign_language 폴더 안에 you (너, 당신) 을 뜻하는 하위 폴더를 생성한 후, 그 폴더 안에 100 개의 캡처된 수화 이미지를 저장한다. 최대한 다양한 위치, 그리고 각도에서 동작의 캡처된 데이터를 수집할 수 있도록 한다.

B. Dataset



[그림 4-1] webcam



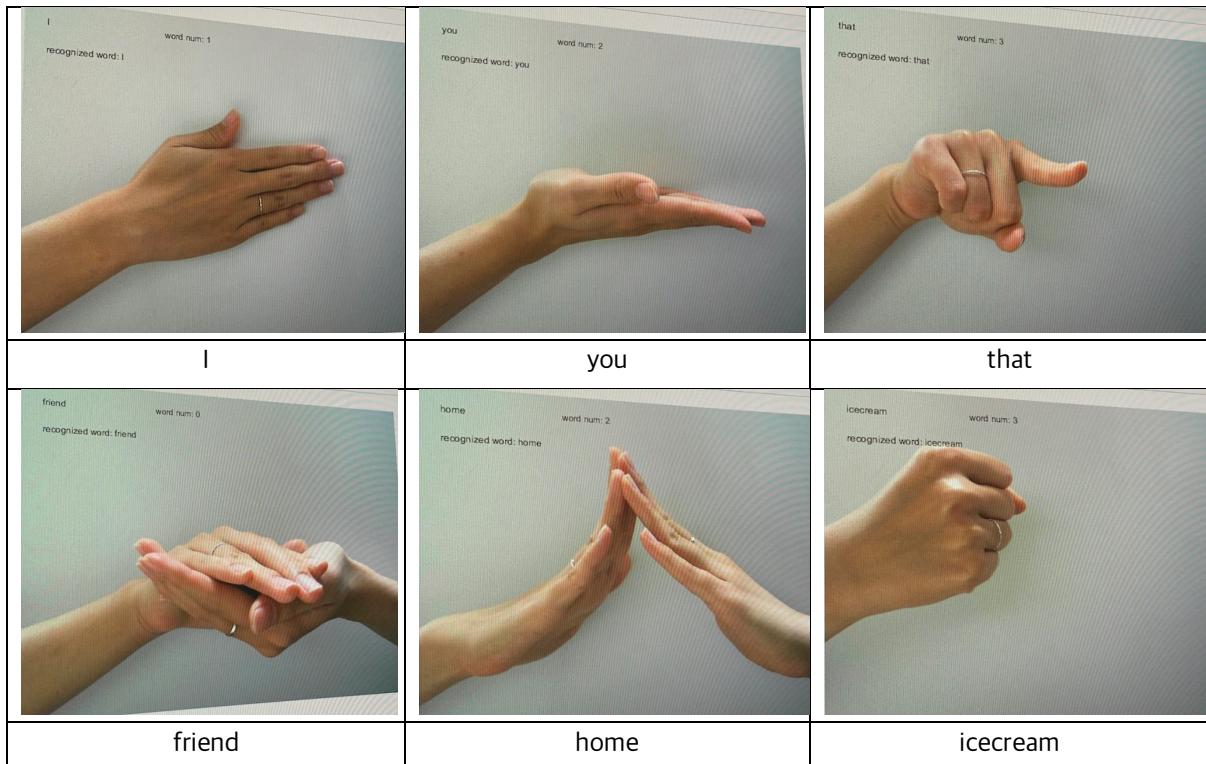
[그림 4-2] background

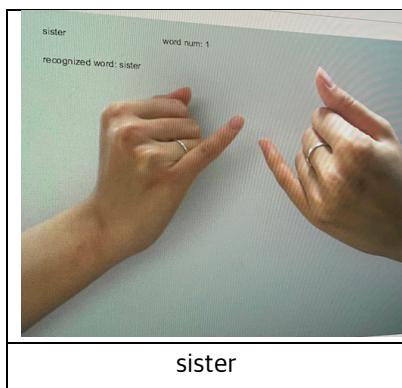


[그림 4-3] dataset acquisition

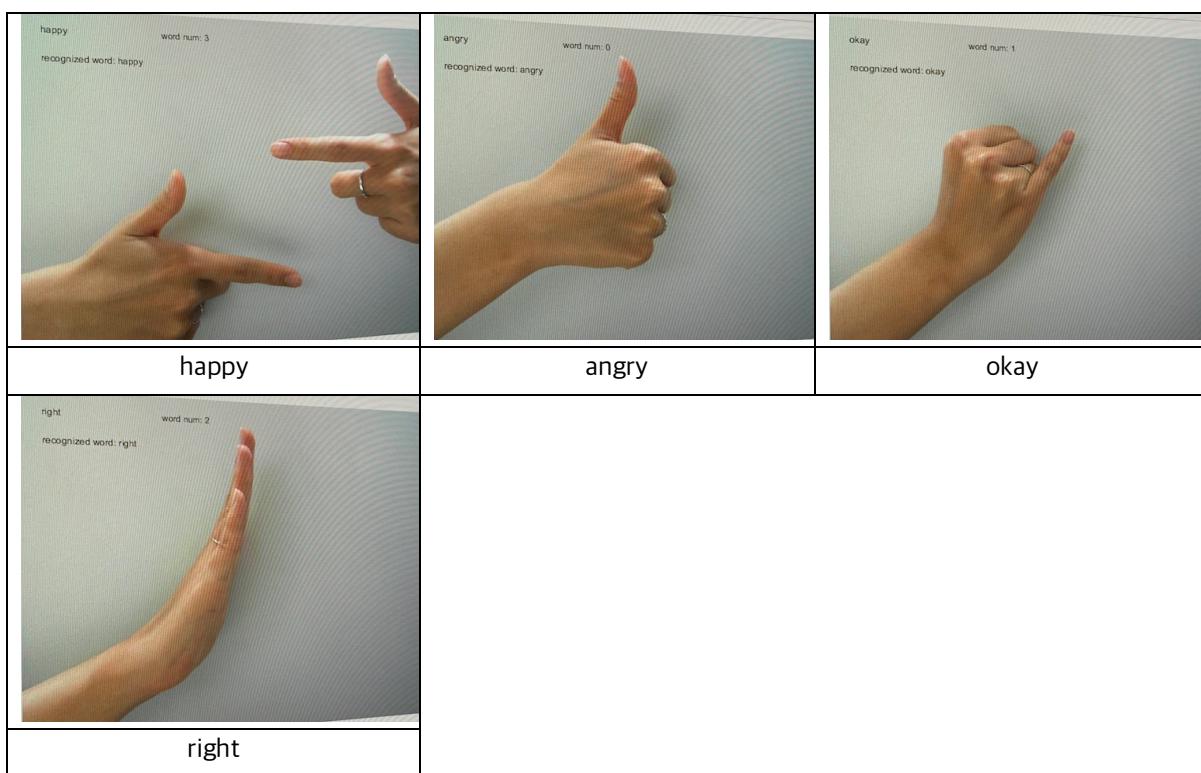
우리는 문장 안에서 흔히 쓰이는 명사 7 개, 형용사 4 개, 동사 6 개로 dataset 을 구성하였고, 각 단어들은 다음과 같다. 움직이는 동작의 경우 그 동작 중 한 순간이 사진으로 저장되었다.

i. 명사





ii. 형용사



iii. 동사





② Train & Detection

A. train.m Code 설명

```
% trained neural network to use
net=alexnet();

categories={'come', 'love', 'happy', 'blank', 'eat', 'you', 'angry', 'like',
'I', 'sister', 'that', 'home', 'friend', 'promise', 'icecream', 'okay','right',
'dislike'};
imds=imageDatastore(fullfile('sign_language',categories),           'LabelSource',
'foldernames');
tbl=countEachLabel(imds)
imds.ReadFcn          = @(filename)readAndPreprocessImage(filename,
net.Layers(1).InputSize(1:2));
[trainingSet, testSet] = splitEachLabel(imds, 0.7, 'randomize');

featureLayer = 'fc7';
trainingFeatures = activations(net, trainingSet, featureLayer, 'MiniBatchSize',
32, 'OutputAs', 'columns');

% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;

% Train multiclass SVM classifier using a fast linear solver, and set
% 'ObservationsIn' to 'columns' to match the arrangement used for training
% features.
classifier = fitcecoc(trainingFeatures, trainingLabels, ...
    'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'columns');

% Extract test features using the CNN
testFeatures = activations(net, testSet, featureLayer, 'MiniBatchSize', 32,
'OutputAs', 'columns');

% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures, 'ObservationsIn',
'columns');

% Get the known labels
testLabels = testSet.Labels;

% Tabulate the results using a confusion matrix.
confMat = confusionmat(testLabels, predictedLabels);

% Convert confusion matrix into percentage form
confMat = bsxfun(@rdivide,confMat,sum(confMat,2))

% Display the mean accuracy
mean(diag(confMat))
```

위 코드는 alexnet 을 이용하여 우리가 만든 dataset 을 학습시키는 역할을 한다. 먼저 ‘fc7’ layer 에 18 가지 output (categories 안의 단어들 집합)을 가진 SVM(support vector machine) 를 추가한다. 그 후 SVM classifier 도 train 시킨다. CNN 을 거친 이미지들을 훈련된 classifier 에 보내 testlabels 를 얻어낸다. 마지막으로, 추측 결과값을 confusion matrix 에 저장하고, 이를 퍼센트 단위로 변형한 후 평균 정확도를 얻어 낸다.

B. train.m Code 실행 결과

```

>> train

tbl =

18x2 table

    Label    Count
    -----  -----
    I        100
    angry    100
    blank    100
    come     100
    dislike  100
    eat      200
    friend   100
    happy    100
    home     100
    icecream 100
    like     100
    love     100
    okay     200
    promise   100
    right    100
    sister   100
    that     200
    you      200

confMat =


1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

mean accuracy 가 1인 것을 보아 제대로 학습이 되었음을 알 수 있다.

C. detect.m Code 중 detection 부분 설명

```

if ~exist('cam', 'var')
    cam=webcam; % use one of webcam(1), webcam(2), and so on if there are
multiple webcams
    cam.Resolution='640x480';
end
k = 0;
i = 0;
feature('DefaultCharacterSet', 'UTF8');
slCharacterEncoding('UTF-8');

label_set = zeros(1, 40);
word_array = strings(1, 300);
% noun
word_array(1) = "I";
word_array(2) = "you";
word_array(3) = "that";
word_array(4) = "friend";
word_array(5) = "sister";
word_array(6) = "home";
word_array(7) = "icecream";
% adjective
word_array(101) = "angry";
word_array(102) = "okay";
word_array(103) = "right";
word_array(104) = "happy";
% verb
word_array(201) = "dislike";
word_array(202) = "love";
word_array(203) = "promise";
word_array(204) = "come";
word_array(205) = "like";
word_array(206) = "eat";

prev_most_common = 0;
% for sentence
word_count = 0;
word_num = zeros(1, 10); % array of word label number
word_set = strings(1, 10);

```

(detect.m code 의 첫번째 부분)

우리가 훈련시킨 단어들로 구성된 word_array 를 선언하였다. Word_array 의 크기는 300 으로, 1부터 99 까지 index 에는 명사를, 100 부터 199 까지의 index 에는 형용사를, 그리고 200 부터 299 까지의 index 에는 동사를 넣었다. 그러므로 index 의 범위를 보고도 단어가 동사인지, 형용사인지, 명사인지 알 수 있을 것이다.

(detect.m code의 두번째 부분)

```

while 1
    im=snapshot(cam);
    % Pre-process the images as required for the CNN

    img = imresize(im, net.Layers(1).InputSize(1:2));

    % Extract image features using the CNN
    imageFeatures = activations(net, img, featureLayer, 'OutputAs', 'columns');
    % Make a prediction using the classifier
    [label,scores] = predict(classifier, imageFeatures, 'ObservationsIn',
    'columns');

    if(label == 'blank')
        index = 0;
    elseif(label == 'I')
        index = 1;
    elseif(label == 'you')
        index = 2;
    elseif(label == 'that')
        index = 3;
    elseif(label == 'friend')
        index = 4;
    elseif(label == 'sister')
        index = 5;
    elseif(label == 'home')
        index = 6;
    elseif(label == 'icecream')
        index = 7;
    elseif(label == 'angry')
        index = 101;
    elseif(label == 'okay')
        index = 102;
    elseif(label == 'right')
        index = 103;
    elseif(label == 'happy')
        index = 104;
    elseif(label == 'dislike')
        index = 201;
    elseif(label == 'love')
        index = 202;
    elseif(label == 'promise')
        index = 203;
    elseif(label == 'come')
        index = 204;
    elseif(label == 'like')
        index = 205;
    elseif(label == 'eat')
        index = 206;
    end

    label_set(mod(k,40)+1) = index;
    imshow(im)
    text(30,30,sprintf('%s', label))
    % text(70,100,sprintf('%s', string(label_set(mod(k,40)+1))))
    % text(220,100,sprintf('test num: %d', k))
    text(200,30,sprintf('word num: %d', word_count))
    % text(30,60,sprintf('%f ',scores))

    [most_common, times] = mode(label_set(:));

```

while 문을 돌리면서 실시간으로 webcam 을 통해 이미지를 받아온다. 이 while 문은 0.05 초마다 돌아가게 설계되었다. 0.05 초 단위로 이미지를 받아 들여 그 이미지의 feature 를 CNN 을 통해 추출하고, 이를 이용하여 어떤

동작인지 classifier 로 판별한다. 그 결과인 label 을 앞서 train 시킨 단어들 중 어떤 것과 일치하는지 알아낸 다음 index 를 정한 후, 크기가 40 인 array 인 label_set 에 이를 순서대로 넣어준다. Label_Set 는 크기가 40 이므로 총 $0.05 * 40 = 2$ 초 동안의 결과를 저장할 수 있는 용량을 가지고 있다. 이 2 초 동안 수행하는 수화의 동작이 바뀔 수도 있고, 동작 중 일부만을 이미지로 인식하여 판별할 수도 있지만, 40 개 중 20 개 이상의 결과가 같은 label 을 가리킨다면, (threshold 를 0.5 로 설정) 그것을 옳은 동작으로 인식하기로 하였다. 이는 우리가 train 시킬 때 정지된 하나의 동작이 아닌 여러 개의 label_set 에서 가장 많이 등장하는 index 를 알아내기 위해서 매트랩 내장함수인 mode 함수를 이용하였다. Label_set 을 mode 함수의 input 으로 넣어주면, 두가지 output 을 얻을 수 있는데, 가장 많이 등장하는 단어가 첫번째 output, 그 단어가 등장한 횟수인 times 를 두번째 output 으로 얻을 수 있다.

(detect.m code 의 다섯번째 부분)

```

else
    most_common_word = word_array(most_common);
    word_percent = times / numel(label_set);
    threshold = 0.5;
    if (word_percent > threshold)
        text(30,70, sprintf('recognized word: %s', (most_common_word)))
        if (prev_most_common ~= most_common)
            word_set(word_count+1) = most_common_word;
            word_num(word_count+1) = find(word_array == most_common_word);
            word_count = word_count + 1;
        end
        if (word_count > 3)
            word_set = strings(1, 3);
            word_count = 0;
        end
        prev_most_common = most_common;
    end
end

drawnow
pause(0.05);
k = k + 1;
end

```

2 초 동안 가장 많이 인식된 단어의 index 를 most_common 이라고 한다. 그리고 이것이 label_set 에서 몇 회 등장하였는지를 times 라고 하였다. 만약 most_common 이 blank 라면 한 문장이 끝나는 것을 의미하므로 인식된 word 가 word_set 에 저장되지 않는다. 따라서 위 코드의 맨 위의 else 는 인식된 word 가 blank 가 아닐 때를 뜻한다.(앞의 If 문은 바로 아래의 3)Make sentence 에서 다를 예정이다.) Most_common 을 알면, word_array 에서 설정해 놓은 index 를 통해 most_common_word 가 무엇인지 알 수 있다. 또한, times 를 알기 때문에 most_common_word 가 우리가 설정해 놓은 threshold (0.5) 이상의 확률로 등장했는지 여부도 판단할 수 있다. 40 칸의 label_set 중에서 20 개 이상의 칸에 같은 label 이 저장되어 있다면, 그 단어로 인식된 것이 맞다고 가정한다. 그러나 한 동작을 오랫동안 하고 있을 수도 있으므로 previous_most_common_word 와 다를 경우에만 word_count 수를 1 만큼 증가시키고 새롭게 그 단어가 인식되었다고 생각한다. 그리고 이 때 word_set 과 word_num 에 각 word 에 해당하는 string 과 index 를 넣어준다. Word_count 는 word_set 과 word_num 에 있는 word 의 갯수인데, word 의 갯수가 3 개가 넘어가면 다시 word_count 를 0 으로 초기화 해주어 3 개의 단어로만 이루어진 문장을 인식하도록 하였다.

D. detect.m Code 실행 결과

Result 1) Data acquisition 의 dataset 사진을 보면 왼쪽 상단에 실시간으로 찍고 있는 동작을 detection 한 word 를 나타내는데, 모두 제대로 인식한 것을 확인할 수 있다.

③ Make sentence

A. code 설명

이제는 인식된 단어들을 가지고 문장을 완성해야 한다. 이 문장은 영어 문법적으로 오류가 없는 1형식부터 3형식까지의 문장이다. 앞서 말한 바와 같이, 직전까지는 단어가 나오다가 어느 순간부터 blank 가 2초 동안 가장 많이 나온 단어로 인식되면 한 문장이 끝났음을 의미한다. 코딩에서 EOF 와 같은 역할을 blank 가 대신 하는 것이다. 따라서 most_common 이 0 이 되면 즉, blank 가 나오면 문장 하나를 인식할 수 있는 상태가 되고, 이전까지 인식된 단어들은 word_set(string 형태)과 word_num(index 형태)로 각각 최대 3개씩 저장된다.

```

if (most_common == 0)
    if (word_count > 0)
        % type 1 sentence
        if (word_count == 1)
            % show the word
            text(250,400,sprintf('%s', string(word_set(1))), 'FontSize',20)
            pause(1);
            word_count = 0; % initialization
        % type 2 sentence
    elseif (word_count == 2)
        if (word_num(2) > 100 && word_num(2) < 200)
            word_set(3) = word_set(2);
            if (word_num(1) == 1) % detected word : I
                word_set(2) = 'am'
            elseif (word_num(1) == 2) % detected word : you
                word_set(2) = 'are'
            else % detected word : others
                word_set(2) = 'is'
            end
            text(180, 400, sprintf('%s %s %s', string(word_set(1)),
string(word_set(2)), string(word_set(3))), 'FontSize',20)
            pause(1);
            sentence = strcat(word_set(1), " ",word_set(2), " ", word_set(3));
            if (sentence == 'I am happy')
                tts('I am happy');
            elseif (sentence == 'sister is angry')
                tts('sister is angry');
            end
        else
            text(210, 400, sprintf('%s %s', string(word_set(1)),
string(word_set(2))), 'FontSize',20)
            pause(1);
        end
        word_count = 0; % initialization
    
```

(detect.m code 의 세 번째 부분)

위 코드에 대해 설명하자면, word_count 가 1이면(word_set 과 word_num 에 word 가 하나 저장되어있으면) 그냥 단순히 단어 한 개로 인식한다. 인식된 단어 하나가 스크린상에 프린트되고

word_count 를 초기화한다. word_count 가 2 이면 1 형식, 또는 2 형식 문장이 가능하다. 두 번째로 오는 word 가 형용사인 경우는 명사와 형용사 사이에 be 동사가 필요하기 때문에 주어에 맞게 be 동사를 word_set(2)에 넣어준다. 이 때 word_set(2)에 있는 word 는 미리 word_set(3)에 넣어둔다. 그리고 난 후 word_set(1), word_set(2), word_set(3)은 있는 경우만)를 프린트하여 2 형식 문장을 완성한다. 두 번째 word 가 동사이면 그냥 명사와 동사를 순서대로 배열하여 문장을 완성시키면 된다.

```
% type 3 sentence
elseif (word_count == 3)
    % if 2nd word is noun and 3rd word is not noun → change the order of them
    if (word_num(2) < 100 && word_num(3) > 100)
        if(word_num(2) ==1)
            word_set(2) = "me";      % I → me (objective version)

        end
        text(180,400, sprintf('%s %s %s', string(word_set(1)),
string(word_set(3)), string(word_set(2))), 'FontSize',20)
    else
        if(word_num(3) ==1)          % I → me (objective version)
            word_set(3) = "me";
        end
        text(180, 400, sprintf('%s %s %s', string(word_set(1)),
string(word_set(2)), string(word_set(3))), 'FontSize',20)
    end
    pause(1);
    sentence = strcat(word_set(1), " ",word_set(2), " ", word_set(3));
    if (sentence == 'you come home')
        tts('you come home');
    elseif (sentence == 'I like icecream')
        tts('I like icecream');
    elseif (sentence == 'you promise me')
        tts('you promise me');
    end
    word_count = 0;    % initialization

end
prev_most_common = most_common;
end
end

drawnow
pause(0.05);
k = k + 1;
end
```

(detect.m code 의 네 번째 부분)

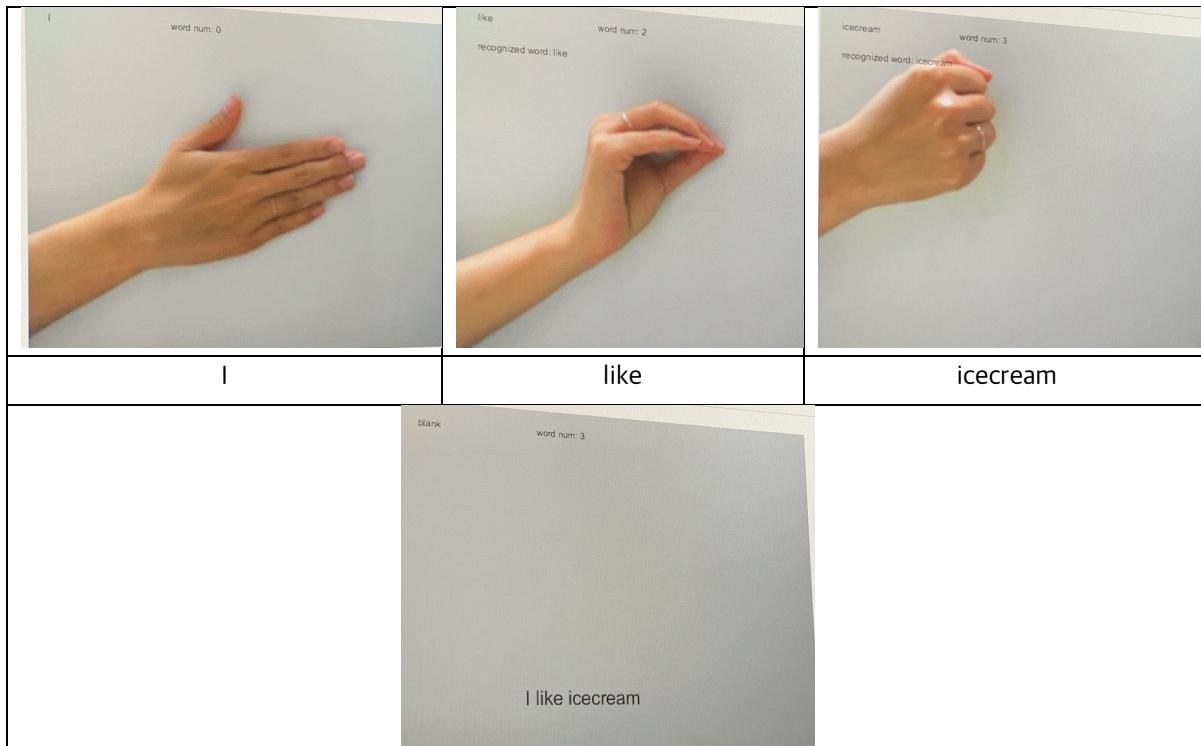
이어서, word_count 가 3 이면 3 형식 문장이므로 S+V+O 로 순서로 구성되어 있어야 한다. 이 때 목적어와 동사의 순서가 바뀐 경우 목적어와 동사의 순서를 바꿔준다. 또한, I 가 목적어로 인식된 경우는 I 를 me 로 바꿔주어 목적격 형태를 만들어준다. 그리고 난 후 word_set(1), word_set(2), word_set(3)을 프린트하여 3 형식 문장을 완성한다.

마지막으로 prev_most_common 을 현재 most_common 으로 업데이트 시켜준다.

B. result

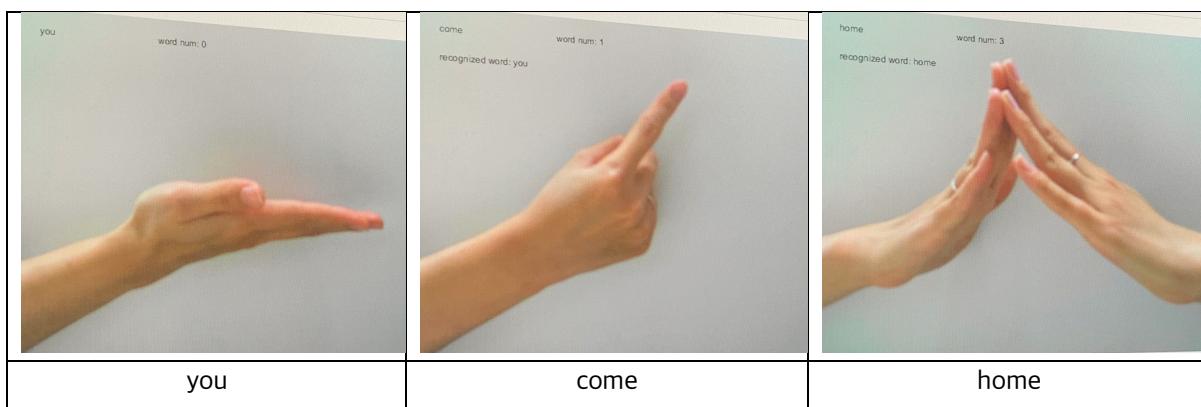
원쪽 상단은 현재 detect 되는 word 를 나타내고, word_num 은 code 에서 언급한 word_count 를, 그리고 recognized word 는 threshold 를 넘어 word_set 에 들어간 word 를 나타낸다.

i. I like icecream



‘나는 아이스크림을 좋아한다’라는 문장을 말하기 위해 영어식으로 ‘I like icecream’ 이라고 할 수도 있지만 한국식으로 이야기하면, ‘나는 아이스크림 좋아’라고 할 수도 있다. 즉 동사와 목적어 순서가 바뀐 이 두 가지 경우를 모두 같은 문장으로 인식함을 확인하였다.

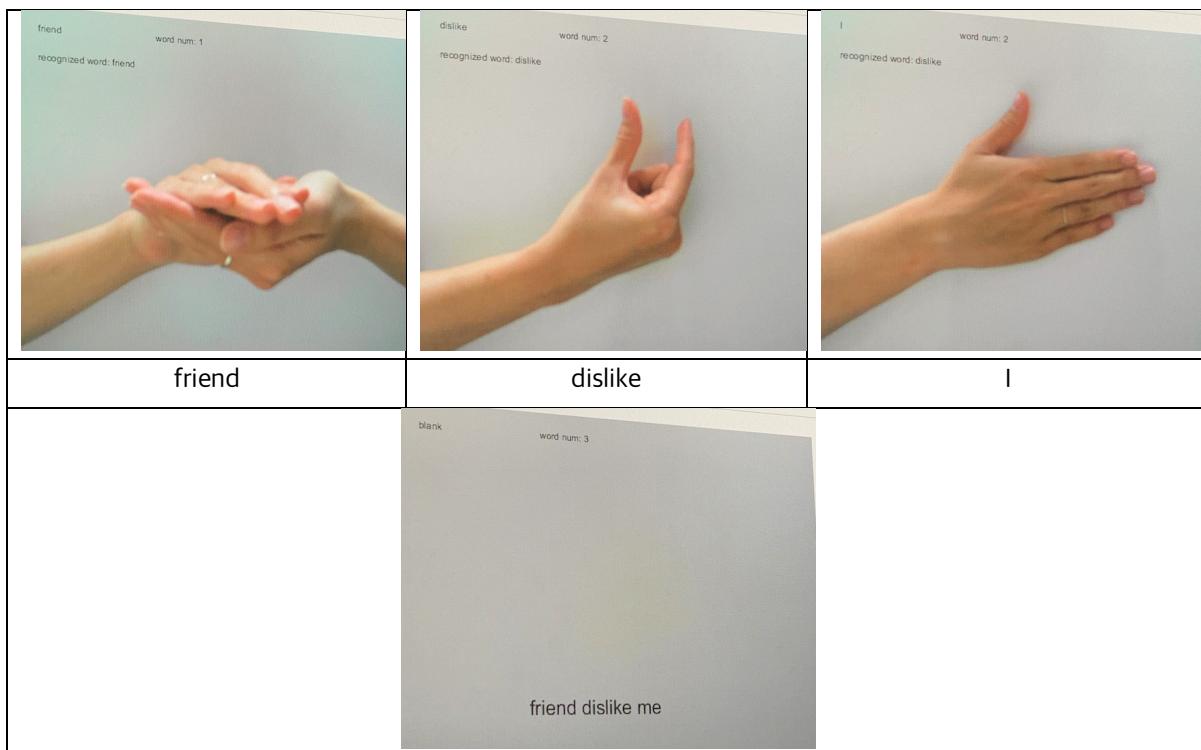
ii. you come home





you come home

iii. friend dislike me



I(나)로 저장된 3 번째 word 가 문장을 만들 시에는 목적격인 me 로 변형되어 나타난 것을 확인할 수 있다.

④ Sentence to speech

완성된 문장을 음성으로 변환하여 스피커를 통해 소리가 날 수 있도록 매트랩 내장함수인 tts 를 사용하였다. 이를 나타낸 부분을 make sentence 의 코드(즉, detect.m code 의 세 번째와 네 번째 부분)에 볼드체로 표시해놓았다. tts 함수는 tts('string')일 때만 정상적으로 작동하고 a라는 변수가 있을 때, tts(a)는 작동하지 않는다. 따라서 변수를 이용하여 tts()를 일반화하여 작동시킬 수 없기 때문에, 우리는 일부 문장만을 위와 같이 tts()함수를 이용하여 음성으로 변환하였다. 먼저 word_set 에 있는 word 들을 strcat 을 이용하여 하나의 문장 string 으로 만들어준 후 if 문으로 해당하는 문장을 수화동작으로 인식하였을 때 소리 나게 읽을 수 있도록 구현하였다. 이 결과는 demo 로 대체한다. 아래는 우리가 실험에서 사용한 스피커이다.



[그림 4-4] 스피커

5. Discussion

① Train 방법

동작의 길이가 길거나 인식이 잘 안되는 동작의 경우는 dataset 의 양을 200 개로 늘려서 학습시켜주었고 그 결과 mean accuracy 가 1 이 된 것을 확인할 수 있었다.

② Sentence to speech

매트랩 내장함수 tts 의 한계로 string formatting 을 하여 모든 문장을 읽을 수 없어 일부의 문장만 소리나게 읽을 수 있었다. tts()와 같은 역할을 하고, input 으로 변수를 인식하는 함수를 찾는다면 일반화할 수 있게 되어 모든 문장을 소리나게 읽을 수 있을 것이다.

③ 동작을 인식하는 새로운 방식

정지된 이미지가 아닌 움직이는 동작을 detect 하기 위하여 처음에는 영상을 찍어 여러 frame 을 하나로 이어 training 을 시키는 방법을 생각하였으나 training 시키는 시간이 오래 걸리고 convolution 이 원하는 대로 적용되지 않을 수도 있다는 조교님의 조언을 듣고 또 다른 새로운 방법을 고안해보았다. 그동안 이미지 training 을 할 때처럼 같은 모양을 반복하는 것이 아니라, 수화 동작을 하는 과정을 모두 이미지로 캡쳐하여 dataset 을 만드는 것이다. 이렇게 되면 동작과 동작 사이의 transition, 혹은 움직이는 동작의 경우, 순간적으로 다른 word 와 특징이 불분명하여 틀리게 recognize 할 수도 있지만 우리는 threshold 를 이용하여 이 문제를 해결하기로 하였고, label_set 40 개 중 threshold 인 20 개 넘게 같은 동작을 했느냐 안 했느냐를 통해 이러한 소수의 에러 가능성을 효과적으로 줄일 수 있었다.

④ Sentence making

Sentence 코드는 2 형식과 3 형식의 경우 word_set(1)은 명사가 저장되어 있다는 것을 가정으로 짜여 있는데 이는 우리는 문장구조를 아예 모르는 사람들을 이 프로젝트의 target 으로 삼은 것이 아니고 문장구조를 어느정도 아는 사람들을 target 으로 삼았기 때문이다. 2 형식 문장의 경우 수화에서 be 동사는 생략되어 있었기 때문에 문법적으로 맞게 하기 위해 be 동사를 추가해주는 작업을 하였고, 3 형식 문장의 경우 한국말로 했을 때 S+V+O 뿐만 아니라 S+O+V 도 말이 되지만 영어 문장에서는 S+V+O 만 문법적으로 맞기 때문에 이 순서를 유지하기 위해 word 의 순서를 바꿔주는 코드를 추가하였다.

⑤ Limitations

- i. 일일이 word_array 에 index 를 추가하는 방법 대신 string 자체를 word_array 에 넣으려 했으나 매트랩 행렬과 array 와의 호환이 자유롭지 않은 함수들 때문에 그렇게 할 수 없었다.

데이터 셋이 이보다 더 늘어난다면 하나하나 index 를 추가하는 방식이 매우 비효율적일 수도 있기 때문에 이를 해결할 방안을 모색한다면 더 발전된 프로그램을 만들 수 있을 것이다.

- ii. 머신러닝 및 딥러닝을 이용해서 학습한 결과를 바탕으로 영어 문법에 맞는 문장을 만들고 싶었으나 결국 manually 해준 점이 아쉬웠다. 원래의 계획은 한 권의 책 정도 분량의 영어 문장을 train data 로 하여 단어의 품사, 즉, 명사, 동사, 형용사, 부사, 조사 사이의 상관 관계를 강화학습으로 학습하는 것이었으나 가 training 시킬 dataset 의 크기의 한계의 문제도 있고, 시간적으로나 기술적으로나 구현하기 힘들어 보였다. 보통 영어의 문장이 어떠한 순서로 구성되는지 스스로 학습한 다음 들어온 단어를 품사에 맞게 label 해주면 더 발전된 프로그램이 될 것이라고 생각한다.

6. Conclusion

우리는 이번 프로젝트에서 한국 수화를 기준으로 수화 동작을 인식하고, 인식한 단어들을 이용하여 문장을 만들어 최종적으로 문장을 소리내어 읽는 시스템을 구현하였다. 음성언어와 마찬가지로 수화도 국가별로 차이가 있으나 음성언어의 공통점보다는 수화 사이의 공통점을 찾기가 더욱 쉽다고 하기에 발전 가능성이 높다고 생각하고, 우리는 3 형식 문장까지만 구현하였지만, 4 형식, 5 형식 문장까지도 만들 수 있을 것으로 생각된다. 또한, 데이터 셋이 많이 쌓이면 더 풍부하고 복잡한 문장도 만들어 낼 수 있고 제대로 인식되는 확률도 높아질 것으로 예상한다. 지금은 배경이 제거된 경우에만 인식 가능하지만 추가적인 과정을 통해 배경이 제거되지 않았을 때도 인식하도록 발전할 수 있을 것이다.

7. Reference

EE405(C), Lecture 7 & Lab 7