Sequential 한 Data 수집 방식과 RNN 모델을 활용한 제스처의 실시간 분류

2018741035 한준호

Junho Han

Abstract

청각장애인과 같이 언어장애를 가지고 있는 사람과 수화로 대화를 하거나 소리를 낼 수 없는 군 전쟁 지역이나, 작전 지역에서는 제스처만을 이용해 소통을 해야 하기 때문에 원활한 소통이 어려울 수 있다. 따라서 카메라를 통해 손 모양을 인식하고 각 손 모양의 관절에 따른 랜드마크들 간에 관계를 활용해, 해당 제스처 Data를 수집하고 이를 분류를 하는 연구를 진행한다. 카메라를 통해 받아온 영상을 'Mediapipe'를 이용해 손 모양, 즉 특정 수화나 제스처를 인식하고, 인식한 손 모양의 관절에 따른 21개의 랜드마크들의 (x, y, z) 좌표를 통해 손가락 마디의 벡터 좌표를 구하고, 최종적으로 인접한 손마디 간에 각도를 구한다. 움직이는 제스처까지 분류하기 위해, 구한 해당 각도 Data를 각 시간에 따른 Sequential한 Data로 저장하여 수집한다. 수집한 Data를 이용해 RNN 모델을 직접 구성하고 학습하여 모델의 성능을 확인하고, 실제로 해당 RNN 학습 모델이 제스처를 정확히 분류하는지 확인한다.

1. 서 론

청각장애인과 같이 언어장애를 가지고 있는 사람과 수화로 대화를 하게 되거나, 소리를 함부로 낼 수 없는 전쟁 및 작전 지역에서는 제스처만을 이용해 소통을 해야 하기 때문에 소통이 어려울 수 있다.

따라서 카메라를 통해 받아온 영상으로부터 'Mediapipe'를 이용해 손 모양을 인식하고 각 손가락 관절에 따른 21개의 랜드마크들의 좌표들을 저장한다. 인접한 손가락 관절에 따른 랜드마크들의 (x, y, z) 좌표들을 이용해 손가락 마디의 벡터 좌표를 찾는다. 찾은 손가락 마디의 벡터 좌표를 이용해 마디 간에 관계, 즉 각도를 계산하여 이를 Sequential한 Data로 저장한다.

손 모양 이미지 자체를 학습 Data로 입력 받고, 이를 CNN으로 학습한다면 멈춰 있는 제스처만 학습하고 분류할 수 있지만, 실제 일상 속에서는 움직이는 제스처를 사용한다. 따라서 움직이는 제스처까지 분류하기 위해서는 해당 제스처의 미래나 과거 정보까지 이용해야 하기 때문에 저장된 Data들 간에 Sequential한 정보도 사용해야 한다. 따라서 이미지 자체를 Data로 저장하는 것이 아닌 인접한 손마디 간에 각도를 Sequential한 Data로 저장한다. 저장한 시간에 따라 움직이는 Sequential한 손 모양 Data를 RNN으로 학습을 진행하여, 움직이고 변화하는 특정 제스처까지도 분류할 수 있도록 한다.

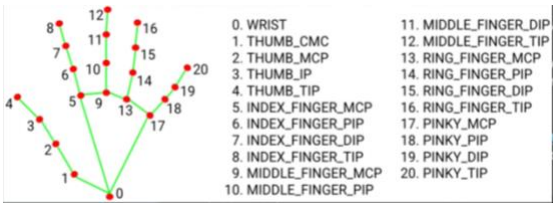
RNN 학습 과정에서는 'Tensorflow'를 이용해 각 레이어를 직접 쌓아보고 학습 파라미터를 적절히 설정하여 모델의 성능을 높이고 해당 모델을 저장해 실시간으로 영상의 제스처를 효과적으로 분리하는지 확인한다.

2. 본 론

2.1 Dataset 수집

2.1.1 손 모양 및 손가락 관절의 랜드마크 인식

'OpenCV'를 이용해 실시간으로 영상을 받아온다.

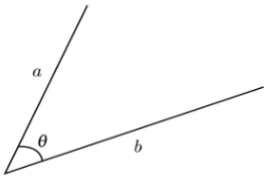


[Fig 1] 'Mediapipe'에 저장된 손가락 관절에 따른 랜드마크 정보

받아온 영상으로부터 'Mediapipe'를 이용해 미리 정해준 일정 시간동안 손 모양 및 손가락을 인식하고 [Fig 1]과 같이 'Mediapipe'에 저장된 21개의 랜드마크 정보들을 이용해, 인식된 손가락 각 관절에 따른 랜드마크들의 (x, y, z) 좌표와 해당 랜드마크가 보이는지에 대한 정보인 Visibility 정보를 저장한다.

2.1.2 손가락 마디간 각도

손가락 각 관절에 따른 랜드마크의 (x, y, z) 좌표를 이용해, 인접한 두 랜드마크 간에 (x, y, z) 좌표를 뺀다면 인접한 손가락 마디의 벡터 좌표를 알 수 있다. 따라서 손가락 각 관절의 21개의 랜드마크로 20개의 손가락 마디 벡터 좌표를 구한다.
구한 인접한 손가락 마디의 벡터 좌표를 벡터의 크기로 나눠줘 크기가 1인 벡터로 만드는 Normalization를 진행한다.



[Fig 2] 두 벡터와 두 벡터 사이의 각도(사이각) 예시

[Fig 2]에서와 같이 두 벡터 사이의 사이각은 0~180도 사이의 값으로 정의되고, 두 벡터 $\vec{a}, \vec{b}$ 의 사이각을 θ라 할 때, 사이각 $\theta(0 \le \theta \le \pi)$ 는 내적과 역삼각함수로 구할 수 있다.

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}|cos\theta \quad (1)$$

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \quad (2)$$

$$\theta = cos^{-1}\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right) = acos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right) \quad (3)$$

찾은 Normalize된 손가락 마디의 벡터 좌표를 수식 (3)을 이용해 인접한 손가락 마디 사이에 각도를 구할 수 있다.

2.1.3 Data 저장

정해진 시간동안 시간에 따라 일정 간격으로 인식되는 움직이는 손 모양마다 인접한 손마디 사이의 각도 Data 15개, 손가락 각 관절의 21개의 랜드마크 (x, y, z) 좌표 Data 63개, 각 랜드마크의 Visibility Data 21개, 정답 라벨 Data 1개로 총 100개 짜리의 행렬을 만든다. 정해진 시간 동안의 움직이는 손 모양에 따른 100개 짜리의 행렬을 이어 붙여 Data로 저장한다.
RNN 학습에는 Sequential한 Data가 필요하기 때문에, 정해준 시간동안 저장한 Date를 미리 정해준 'seq_length' 만큼씩 차례대로 윈도우를 이동해가며 저장해, 최종적으로 출력 윈도우 사이즈가 'seq_length' 인 Numpy Array 형태로 변환하여 저장한다.

본 연구에서는 10개의 제스처('OKAY', 'NO', 'COME', 'AWAY', 'STAY', 'GOOD', 'VICTORY', 'HELLO', 'PROMISE', 'ROCK')를 한 제스처 당 45초 동안 'seq_length'=45로 설정하여 Data를 수집하였고, 결과는 다음 [Fig 3]과 같다.

```
actions = ['OKAY', 'NO', 'COME', 'AWAY', 'STAY',
           'GOOD', 'VICTORY', 'HELLO', 'PROMISE', 'ROCK']

data = np.concatenate([
    np.load('./DeeplearningProject/dataset/seq_Okay.npy'),
    np.load('./DeeplearningProject/dataset/seq_No.npy'),
    np.load('./DeeplearningProject/dataset/seq_Come.npy'),
    np.load('./DeeplearningProject/dataset/seq_Away.npy'),
    np.load('./DeeplearningProject/dataset/seq_Stay.npy'),
    np.load('./DeeplearningProject/dataset/seq_Good.npy'),
    np.load('./DeeplearningProject/dataset/seq_Victory.npy'),
    np.load('./DeeplearningProject/dataset/seq_Hello.npy'),
    np.load('./DeeplearningProject/dataset/seq_Promise.npy'),
    np.load('./DeeplearningProject/dataset/seq_ROCK.npy')
], axis=0)

print(data.shape)
(6317, 45, 100)
```

[Fig 3] 10개의 제스처 Data를 이어 붙인 최종 'data'

2.2 RNN 모델 구성 및 학습

2.2.1 Data 전처리

[Fig 3]과 같이 (6317, 45, 100) 형태의 수집한 'data'에 가장 끝에는 정답 라벨 Data가 들어있기 때문에 가장 끝의 Data는 'labels', 나머지는 'x_data'에 저장한다. 10개의 제스처에 대한 정답 라벨인 'labels'을 이용하여 'y_data'에 저장한다.
최종적으로, 'x_data'는 (6317, 45, 99), 'y_data'는 (6317,10)의 형태가 된다.

2.2.2 Validation Data 생성

효과적인 모델 학습과, 학습된 모델의 성능을 검증하기 위해 Validation Data를 생성한다. 'train_test_split()' 함수를 이용해 'x_data'와 'y_data'를 9:1 비율로 Train Data와 Validation Data으로 나눠준다.

2.2.3 RNN Layer

움직이는 제스처까지 분류가 가능한 학습 모델을 만들기 위해 Data들 간에 Sequential한 정보도 사용해야하기 때문에, 시계열 Data를 이용하는 인공 신경망 유형인 RNN을 구성하였다.

```
model = Sequential()

model.add(LSTM(64, input_shape=x_train.shape[1:3], return_sequences=True))
model.add(Bidirectional(LSTM(64, input_shape=x_train.shape[1:3], return_sequences=True)))
model.add(Bidirectional(LSTM(64, input_shape=x_train.shape[1:3], return_sequences=True)))
model.add(LSTM(64, input_shape=x_train.shape[1:3], return_sequences=False))

model.add(Dense(32))
model.add(Dense(len(actions), activation='softmax'))
```

[Fig 4] 'TensorFlow'를 이용해 구성한 RNN Layer

[Fig 4]와 같이 RNN 모델을 만들기 위해서 LSTM을 사용했다. 층을 깊게 쌓지 않아도 비교적 분류가 잘 되었기 때문에 층을 깊게 쌓지 않았다.

수집한 제스처 Data가 일정 시간동안 같은 제스처가 반복되는 Data이므로, 제스처의 움직이는 순서가 반대로 입력될 수도 있기 때문에, 현재 Data의 미래와 과거 정보까지 모두 사용할 수 있도록 2개의 RNN 구조가 쌓여 있는 상태라고 볼 수 있는 양방향(Bidirectional) RNN을 사용하였다.

사용하는 제스처 Data가 일정 시간 간격으로 순차적으로 배치된 시계열 Data이므로 Data의 입력 순서가 중요하기 때문에, Drop-out 층을 추가하지 않았다.

안정적인 학습을 위해 Dense를 한개의 층으로 진행하지 않고 Dense 층을 두번으로 나누어 Layer를 구성하였다.

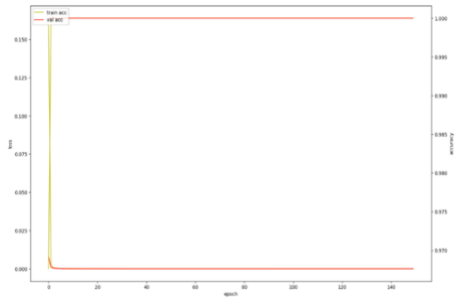
여러 개의 제스처를 분류하는 모델이므로, 마지막 층의 활성화 함수로 다중 분류에 좋은 성능을 보여주는 활성화 함수인 'Softmax' 함수를 사용하였다.

2.2.4 학습 파라미터

옵티마이저(optimizer) 함수로는 'Adam', 손실 함수(Loss Function)으로는 다중 분류 문제에 주로 사용하는 'categorical_crossentropy' 함수를 사용하였다.

'batch_size = 32', 'epochs = 150'으로 설정하였고, 'save_best_only = True' 로 설정하여 가장 정확도가 높은 모델이 저장될 수 있도록 하였다.

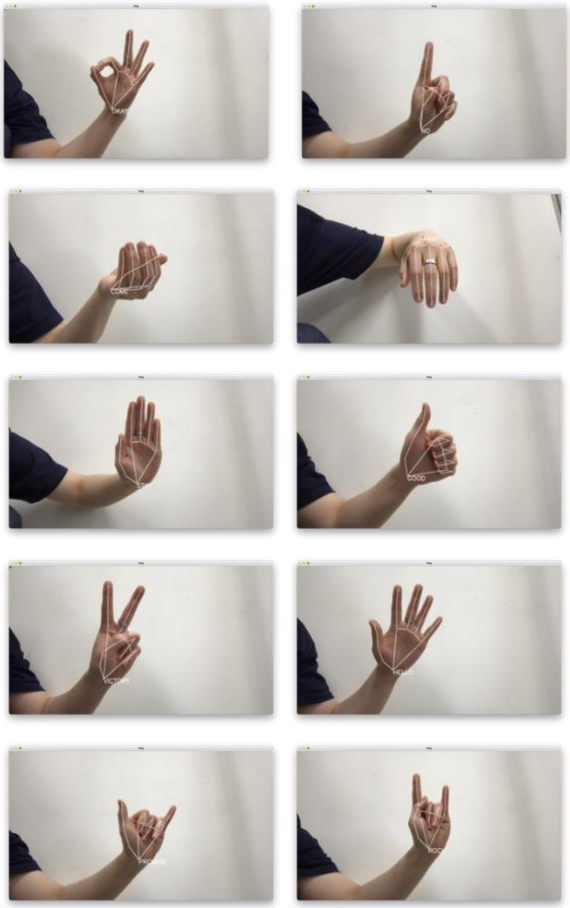
3. 실험 결과



[Fig 5] 학습 'epoch'에 따른 Training, Validation Data의 Accuracy와 Loss 결과 그래프

[Fig 5]와 같이 직접 구성한 RNN 모델이 Training Data와 Validation Data에 대해 좋은 학습 성능을 보여주었다.

학습된 모델을 불러와 각 제스처를 실제로 잘 분류하는지 확인해본 결과, [Fig 6]과 같이 웹 캠으로부터 인식된 손 모양을 학습한 10개의 제스처에 대해 좋은 성능으로 분류하고 이와 맞게 Labeling된 것을 확인하였다.



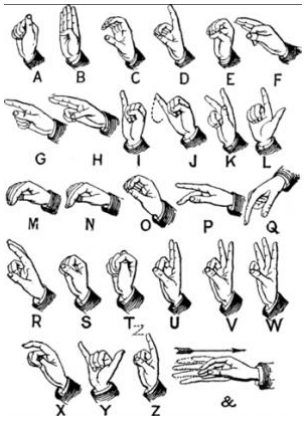
[Fig 6] 학습된 모델이 실제로 인식된 손 모양을 분류한 결과 이미지

4. 결 론

멈춰 있는 손 모양 이미지 자체를 CNN을 통해 학습하지 않고, 손 관절에 따른 21개의 랜드마크 좌표를 이용해 손가락 각 마디의 벡터를 구하고 이를 이용해 손가락 마디 사이의 각도를 구하여 구한 각도 값을 Sequential하게 저장하고, 이를 RNN으로 학습하여 제스처 움직임에 대한 미래와 과거의 정보까지 사용하여 움직이는 제스처까지 분류할 수 있었다.

학습 모델의 목표에 맞는 RNN Layer 의 구조를 설정하고, 적절한 학습 파라미터의 설정을 통해 RNN 모델을 개선하여 모델의 성능을 높일 수 있었다.

본 연구에서는 10 개의 제스처만을 분류했지만, 해당 연구의 Sequential 한 Data 수집 방법과 RNN 학습 방법을 이용해, [Fig 7]과 같은 수화나 군에서 사용하는 제스처까지 여러 제스처를 Data 로 수집한다면, 보다 넓은 분야에서 사용이 가능할 것으로 보인다.



[Fig 7] 수화 예시

References

[1] Jungpil Shin et al, “American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation”, 2021

[2] George Sung et al, “On-device Real-time Hand Gesture Recognition”, 2021

[3] Fan Zhang et al, “On-device Real-time Hand Tracking”, 2020