

[모션인식]을 통한 수부재활 프로그램



전진하, 김수연, 이은경, 유혜지

Table of contents

01

배경 및 필요성

02

데이터 수집 / 전처리

- keypoint logging
- data preprocessing

03

Model

- DNN
- CNN
- LSTM

04

Conclusion

- 프로젝트 한계
- 개선 방안

01

배경 및 필요성



수부재활 프로그램의 필요성



수부부상


과제지향적 활동이
1)집기 근력, 조작능력, 동작수행에 유의미하게
긍정적 영향



재활의 어려움

혼자서 과제지향적 활동을 하기 어려움
병원의 경우 시간과 돈을 수반

¹⁾노동희, 한승협, 조은주, 안성호, 김훈주, 감경윤.(2015).과제 지향적 활동이 수부손상환자의 손 기능에 미치는 효과.한국산학기술학회 논문지,16(2),1153-1163.



손의 움직임을 **시각적**으로 확인
피드백 제공 / 올바른 동작으로 개선
맞춤형 치료



02

데이터 수집 / 전처리

- keypoint logging
- data processing

Keypoint Logging - 데이터 수집

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("fail")
        break
    res = hand.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    if number == -1:
        key = cv2.waitKey(10)
        number, mode = select_mode(key, mode)

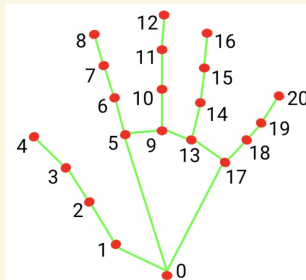
    image_width, image_height = frame.shape[1], frame.shape[0]
    # image = copy.deepcopy(frame)
    # image = cv2.flip(image, 1)
    # image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

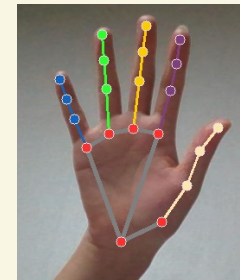
            landmark_list = calc_landmark_list(image_width, image_height, landmarks)
            pre_processed_landmark_list = pre_process_landmark(landmark_list)
            print(number, frame_cnt)
            if number == 1:
                logging_csv(number, mode, pre_processed_landmark_list, count)
                frame_cnt += 1
                if frame_cnt == 56:
                    number = -1
                    frame_cnt = 0
                    count += 1

    cv2.imshow("MediaPipe Hands", cv2.flip(frame, 1))
    if cv2.waitKey(5) == ord('q'):
        break

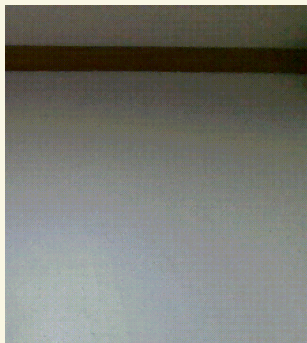
cap.release()
cv2.destroyAllWindows()
```



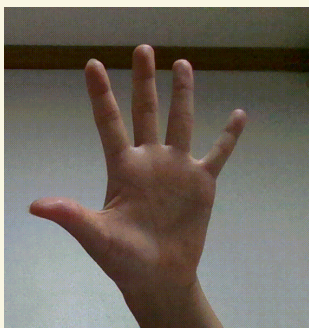
- | | |
|-----------------------|-----------------------|
| 0. WRIST | 11. MIDDLE_FINGER_DIP |
| 1. THUMB_CMC | 12. MIDDLE_FINGER_TIP |
| 2. THUMB_MCP | 13. RING_FINGER_MCP |
| 3. THUMB_IP | 14. RING_FINGER_PIP |
| 4. THUMB_TIP | 15. RING_FINGER_DIP |
| 5. INDEX_FINGER_MCP | 16. RING_FINGER_TIP |
| 6. INDEX_FINGER_PIP | 17. PINKY_MCP |
| 7. INDEX_FINGER_DIP | 18. PINKY_PIP |
| 8. INDEX_FINGER_TIP | 19. PINKY_DIP |
| 9. MIDDLE_FINGER_MCP | 20. PINKY_TIP |
| 10. MIDDLE_FINGER_PIP | |



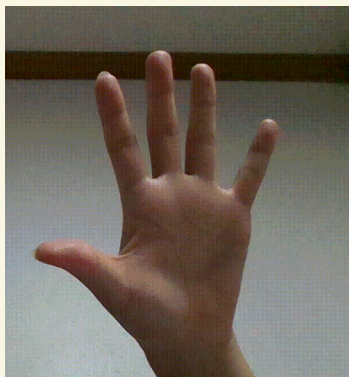
Keypoint Logging - 데이터 수집



손가락벌림 모음굴곡운동



손끝 대립 운동



충양근 운동

Labeling

- 0_손가락벌림 모음굴곡운동 왼손
- 1_손가락벌림 모음굴곡운동 오른손
- 2_엄지 검지 손끝 대립 왼손
- 3_엄지 검지 손끝 대립 오른손
- 4_엄지 중지 손끝 대립 왼손
- 5_엄지 중지 손끝 대립 오른손
- 6_엄지 약지 손끝 대립 왼손
- 7_엄지 약지 손끝 대립 오른손
- 8_엄지 소지 손끝 대립 왼손
- 9_엄지 소지 손끝 대립 오른손
- 10_충양근운동 왼손
- 11_충양근운동 오른손

Left_hand_label = [0, 2, 4, 6, 8, 10]

right_hand_label = [1, 3, 5, 7, 9, 11]

출처: <https://www.youtube.com/watch?v=a-sTkn15dE8>

Data Preprocessing

Frame > 70 제거

70 프레임 이상인 label과 count로 구성된 list를 반환

```
def get_drop_index_list(df):
    drop_label = pd.DataFrame(data[['count', 'label']].value_counts() > 70)
    drop_label.reset_index(inplace=True)
    drop_label_df = drop_label[drop_label[0] == True][['count', 'label']]
    drop_index_list = []
    for idx, row in drop_label_df.iterrows():
        drop_index_list.append([row['label'], row['count']])
    return drop_index_list
```

특정 라벨에 해당하는 동작의 프레임 전체 반환

```
def label_dataframe(label_idx, df):
    result_df = pd.DataFrame(columns = data.columns)
    for idx, row in df.iterrows():
        if row['label'] == label_idx:
            result_df.append(row)
    return result_df
```

특정 라벨값에 대하여 프레임 수가 70 초과인 행의 인덱스 반환

```
def drop_row(label_idx, df):
    result_df = label_dataframe(label_idx, df)
    drop_index_list = get_drop_index_list(data)
    for i in range(len(drop_index_list)):
        if drop_index_list[i][0] == label_idx:
            drop_idx = result_df[(result_df['label'] == label_idx) & (result_df['count'] == drop_index_list[i][1])].index
            result_df.drop(drop_idx, inplace=True)
    return result_df
```

프레임 수 70 초과인 label, count에 대하여 행 제거

```
def drop_morethan70(label_idx, label_list, label_df):
    preprocessed_ = label_dataframe(label_idx, label_df)
    for idx, row in preprocessed_.iterrows():
        if row['count'] not in label_list:
            preprocessed_.drop(idx, inplace=True)
    return preprocessed_
```

Data Preprocessing

Frame = 56 통일

56frame 미만 → 편 동작 앞뒤 삽입

56frame 초과 → 동작 앞뒤 프레임 제거

```
# 각 count에 대한 프레임 수를 데이터프레임으로 반환
def get_frame_count(df):
    frame_count = pd.DataFrame(df.groupby('count')['label'].count())
    frame_count.reset_index(inplace=True)
    frame_count.rename(columns={'label': 'frame_count'}, inplace=True)
    return frame_count
```

손 펴고 있는 동작 랜드마크 로그

```
hands_default = {"wrist_x": [0], "wrist_y": [0], "thumb_mcp_x": -0.175258, "thumb_mcp_y": -0.087629, "thumb_mcp_x": -0.329897, "thumb_mcp_y": -0.206186, "thumb_ip_x": -0.469072, "thumb_ip_y": -0.273196,
"thumb_tip_x": -0.592784, "thumb_tip_y": -0.278351, "index_finger_mcp_x": -0.149485, "index_finger_mcp_y": -0.489691, "index_finger_pip_x": -0.195876, "index_finger_pip_y": -0.701031,
"index_finger_dip_x": -0.221649, "index_finger_dip_y": -0.829897, "index_finger_tip_x": -0.231959, "index_finger_tip_y": -0.938144, "middle_finger_mcp_x": -0.025773, "middle_finger_mcp_y": -0.515464,
"middle_finger_pip_x": -0.025773, "middle_finger_pip_y": -0.742268, "middle_finger_dip_x": -0.025773, "middle_finger_dip_y": -0.881443, "middle_finger_tip_x": -0.025773, "middle_finger_tip_y": -1.0,
"ring_finger_mcp_x": 0.092784, "ring_finger_mcp_y": -0.489691, "ring_finger_pip_x": 0.118557, "ring_finger_pip_y": -0.695876, "ring_finger_dip_x": 0.134021, "ring_finger_dip_y": -0.829897,
"ring_finger_tip_x": 0.144330, "ring_finger_tip_y": -0.938144, "pinky_mcp_x": 0.201031, "pinky_mcp_y": -0.422680, "pinky_pip_x": 0.283505, "pinky_pip_y": -0.551546, "pinky_dip_x": 0.340206,
"pinky_dip_y": -0.639175, "pinky_tip_x": 0.381443, "pinky_tip_y": -0.721649}
```

```
def get_56frames(df, count_df, label):
    remain = []
    for idx, row in count_df.iterrows():
        remain.append([row['count'], row['frame_count']])

    insert_df = pd.DataFrame(columns=df.columns)
    for i in range(len(remain)):
        if remain[i][1] == 56:
            continue:
        # 프레임수 부족
        elif remain[i][1] < 56:
            num = 0
            default_hand = pd.DataFrame(hands_default)
            default_hand.insert(0, 'label', label)
            default_hand.insert(0, 'count', remain[i][0])
            for w in range(0, 56 - int(remain[i][1])):
                insert_df = pd.concat([insert_df, default_hand, df[df['count'] == remain[i][0]], ignore_index=True)
            df.drop(df[df['count'] == remain[i][0]].index, axis=0, inplace=True)

        # 프레임수 초과
        elif remain[i][1] > 56:
            half = (int(remain[i][1]) - 56) // 2

            for k in range(0, half):
                del_idx1 = df[df['count'] == remain[i][0]].index[0]
                df.drop(del_idx1, axis=0, inplace=True)
            for j in range(0, int(remain[i][1]) - (56 + half)):
                del_idx2 = df[df['count'] == remain[i][0]].index[-1]
                df.drop(del_idx2, axis=0, inplace=True)
            return pd.concat([insert_df, df], ignore_index=True)
```

Keypoint Logging - 데이터 수집

0_ 89

1_ 90

2_ 90

3_ 92

4_ 90

5_ 90

6_ 89

7_ 90

8_ 90

9_ 90

10_ 84

11_ 80

}

1,064개

03

Model

- DNN
- CNN
- LSTM



DNN

Input_shape = ((42 * 56,))

Model: "sequential"

Layer (type)	Output Shape	Param #
dropout (Dropout)	(None, 2352)	0
dense (Dense)	(None, 20)	47060
dropout_1 (Dropout)	(None, 20)	0
dense_1 (Dense)	(None, 10)	210
dense_2 (Dense)	(None, 12)	132

Total params: 47,402
Trainable params: 47,402
Non-trainable params: 0

```
model.compile(  
    optimizer='adam',  
    loss='sparse_categorical_crossentropy',  
    metrics=['accuracy']  
)
```

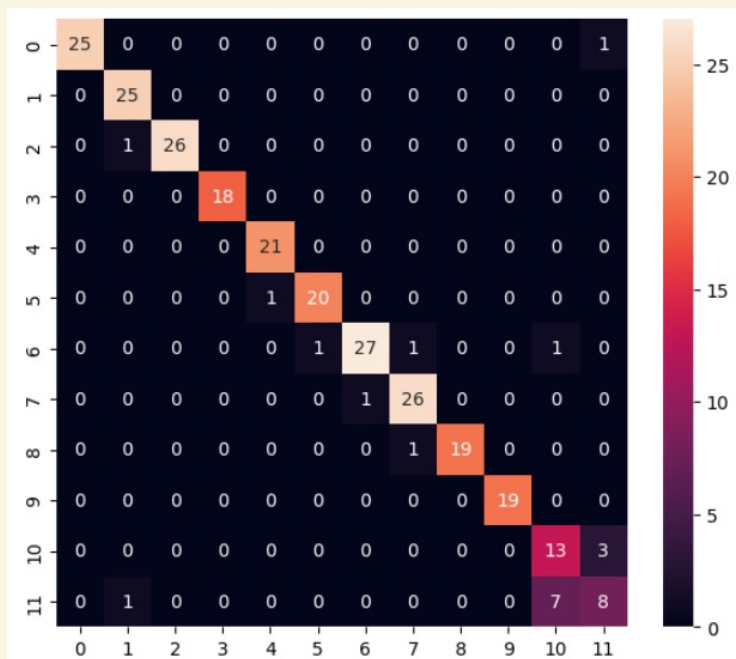
```
model.fit(  
    X_train,  
    y_train,  
    epochs=1000,  
    batch_size=128,  
    validation_data=(X_test, y_test),  
    callbacks=[cp_callback, es_callback]  
)
```

ETA: 0s - loss: 0.5490 - accuracy: 0.7969

ve/MyDrive/KHUDA3rd 3차/0724/keypoint_classifier_0725_lstm.hdf5

0s 21ms/step - loss: 0.7254 - accuracy: 0.7368 - val_loss: 0.4291 - val_accuracy: 0.9286

DNN



Classification Report				
	precision	recall	f1-score	support
0	1.00	0.96	0.98	26
1	0.93	1.00	0.96	25
2	1.00	0.96	0.98	27
3	1.00	1.00	1.00	18
4	0.95	1.00	0.98	21
5	0.95	0.95	0.95	21
6	0.96	0.90	0.93	30
7	0.93	0.96	0.95	27
8	1.00	0.95	0.97	20
9	1.00	1.00	1.00	19
10	0.62	0.81	0.70	16
11	0.67	0.50	0.57	16
accuracy			0.93	266
macro avg	0.92	0.92	0.91	266
weighted avg	0.93	0.93	0.93	266

LSTM

Input_shape = ((56, 42,))

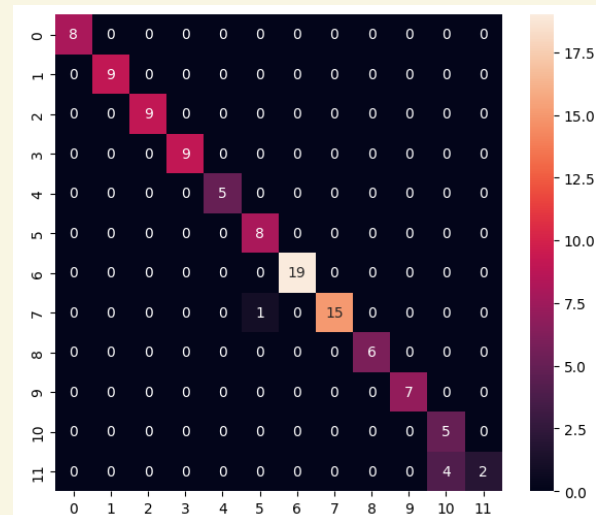
Model: "sequential_13"

Layer (type)	Output Shape	Param #
lstm_10 (LSTM)	(None, 64)	27392
dropout_20 (Dropout)	(None, 64)	0
dense_30 (Dense)	(None, 32)	2080
dense_31 (Dense)	(None, 12)	396

Total params: 29,868

Trainable params: 29,868

Non-trainable params: 0



ETA: 0s - loss: 0.2334 - accuracy: 0.8924

ive/MyDrive/khuda_3rd/final_keypoint_classifier.hdf5

1s 131ms/step - loss: 0.2334 - accuracy: 0.8924 - val_loss: 0.0699 - val_accuracy: 0.9626

CNN

Input_shape = ((56, 42, 1,))

Model: "sequential_4"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 40, 16)	160
max_pooling2d (MaxPooling2D)	(None, 27, 20, 16)	0
conv2d_1 (Conv2D)	(None, 25, 18, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 12, 9, 32)	0
conv2d_2 (Conv2D)	(None, 10, 7, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 5, 3, 32)	0
flatten (Flatten)	(None, 480)	0
dense_9 (Dense)	(None, 512)	246272
dense_10 (Dense)	(None, 1)	513
Total params: 260,833		
Trainable params: 260,833		
Non-trainable params: 0		

ETA: 0s - loss: nan - accuracy: 0.0703
e/MyDrive/KHUDA3rd 3차/0724/keypoint_classifier_0725_lstm.hdf5
0s 23ms/step - loss: nan - accuracy: 0.0789 - val_loss: nan - val_accuracy: 0.0977

04

Conclusion

- 프로젝트 의의 및 한계
- 개선 방안



프로젝트 의의 및 한계

의의

타 재활운동 프로그램같은 경우 단순히 동작의 시간만 재주는 식이었다면, 해당 재활운동 프로그램은 동작의 횟수, 정확도 등의 정량적 평가가 포함되어 재활 환자로 하여금 더욱 효율적인 과제를 수행하도록 할 수 있음.

프로젝트 의의 및 한계

한계

- 동작을 느리게 또는 빠르게 수행하면 모델의 동작 인식 정확도가 떨어짐.
- 동작을 모두 56프레임으로 맞춰기 때문에 수행 시간이 긴 동작은 구별하기 어려움.
- 그래프의 peak를 찾는 시스템이 부정확할 경우 존재
- 카메라 정면을 바라보고 동작을 수행해야 함. → 손의 특정 방향, 각도만 인식 가능. 유연성X

개선 방안

- 동작 횟수를 카운트하는 방식을 더욱 정확하게
ex) 동작의 변곡점을 체크한 후 일정값 이하면 해당 동작을 무시
- 다양한 방향, 각도에서 동작하는 데이터셋을 추가로 수집해 유연한 분류가 가능하도록
- unity 등을 활용하여 더욱 편리한 UI 개발 및 모바일 환경 고려

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

전진하, 김수연, 이은경, 유혜지