SIB

Summary

of previous presentation

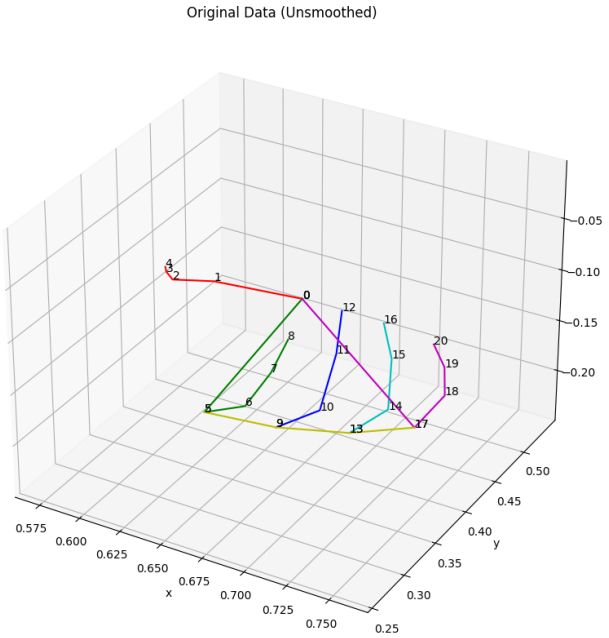
- Dataset Gerneration
  - 데이터셋 생성 코드 완성
- Hand 3D Visualization
  - Data smoothing 및 3D 시각화 작업
- Preprocessing
  - 데이터 전처리 작업 완료

Data Generation

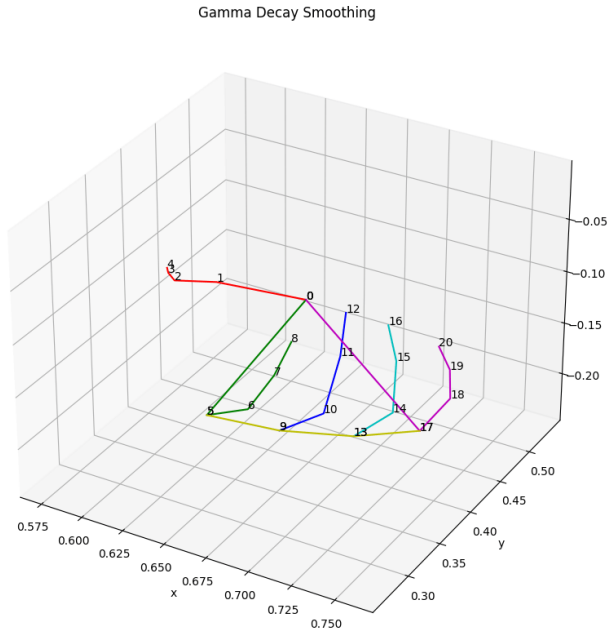
L0-x	~	L21-z	a	s	d	f
0.7227		-0.2402	0	0	0	0
0.7224		-0.2064	0	1	0	0
0.7237		-0.2054	0	0	0	0
0.7272		-0.2061	1	0	0	0
0.7292		-0.2051	0	0	0	1

Hand 3D Visualization

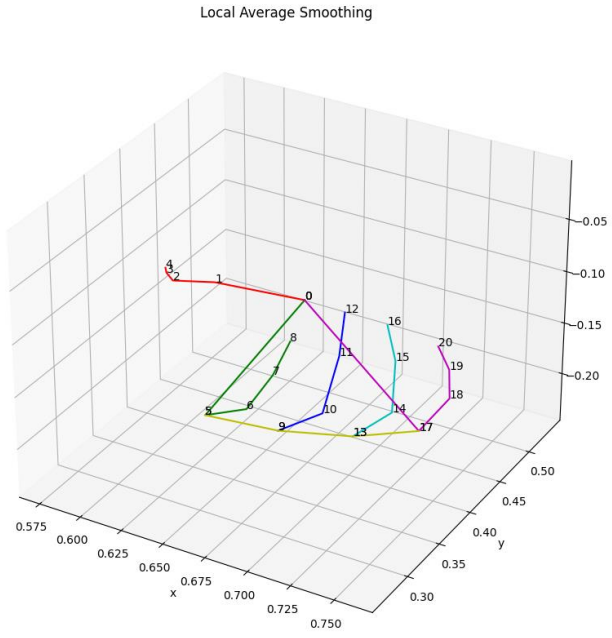
Data Smoothing



Original data



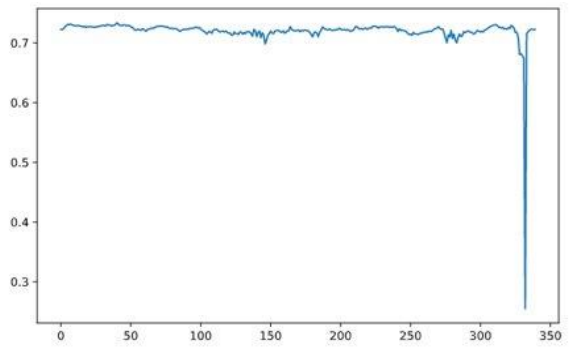
Gamma Decay

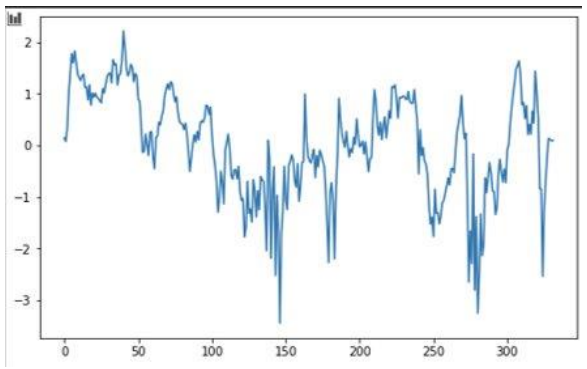


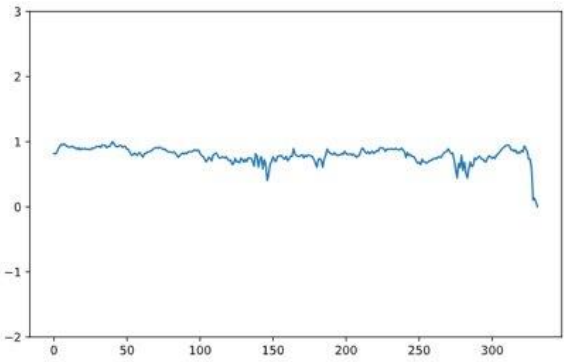
Local Average

Preprocessing

Outlier removal → Standardization → Minmax Norm





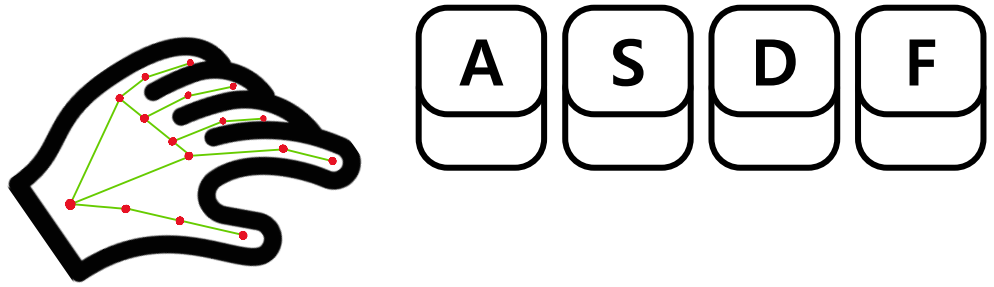


Updated

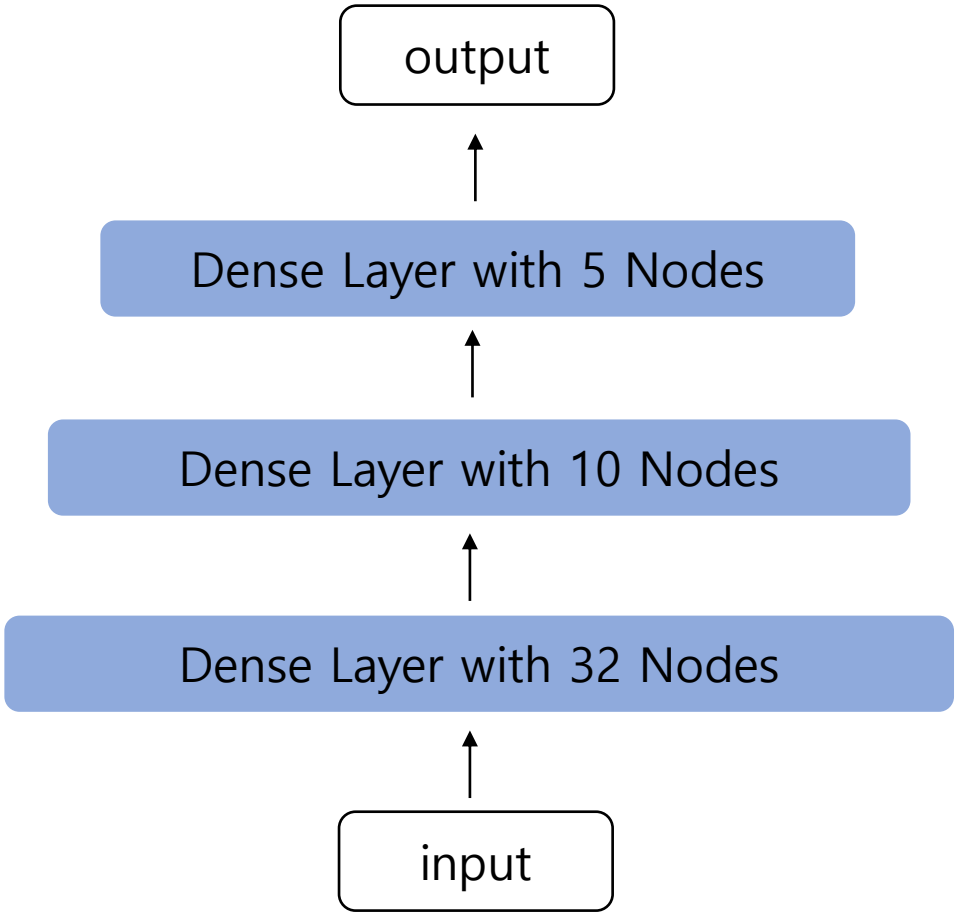
- Modeling

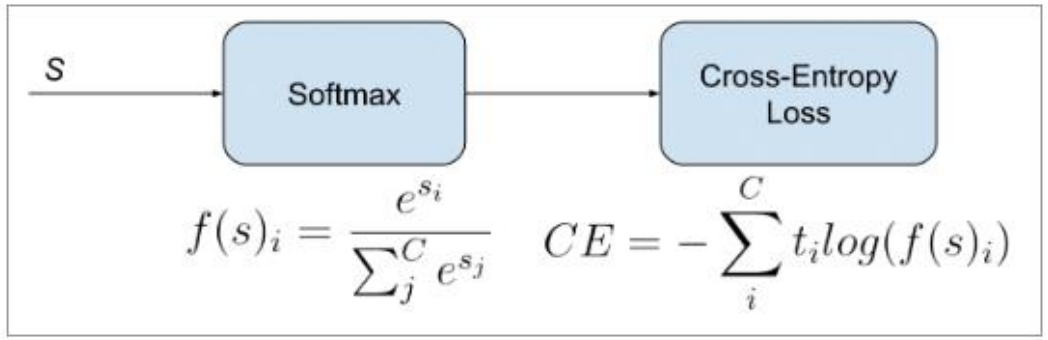
MLP　　RNN　　LSTM

- ‘A’, ‘S’, ‘D’, ‘F’ 입력을 LIVE로 받아서 처리 하기



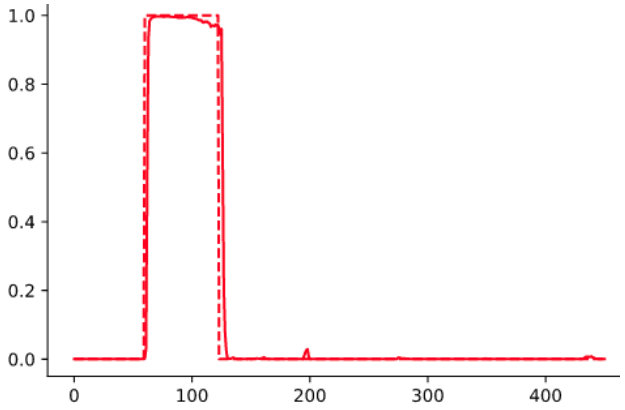
Model(MLP)



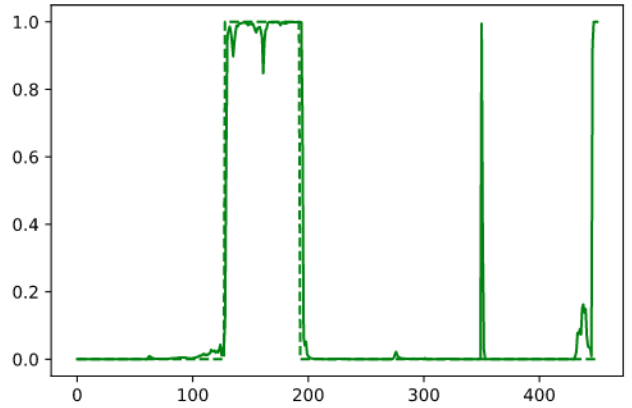



```
model = tf.keras.models.Sequential([
    keras.layers.Dense(32, activation='relu', input_shape=(63,)),
    # keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(5, activation='softmax')
  ])

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

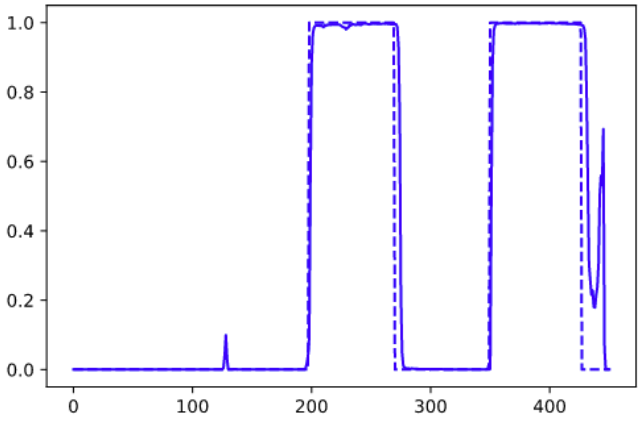
Model(MLP)



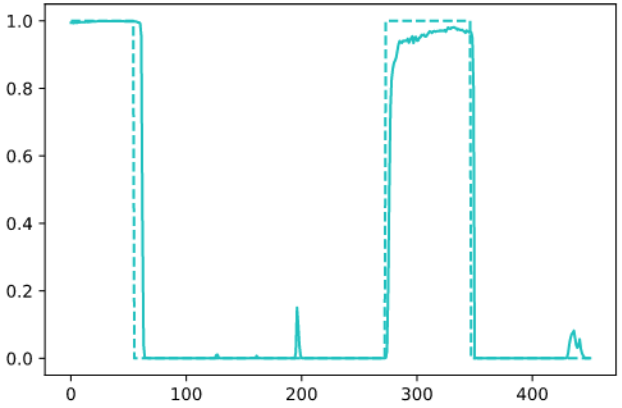
a



s

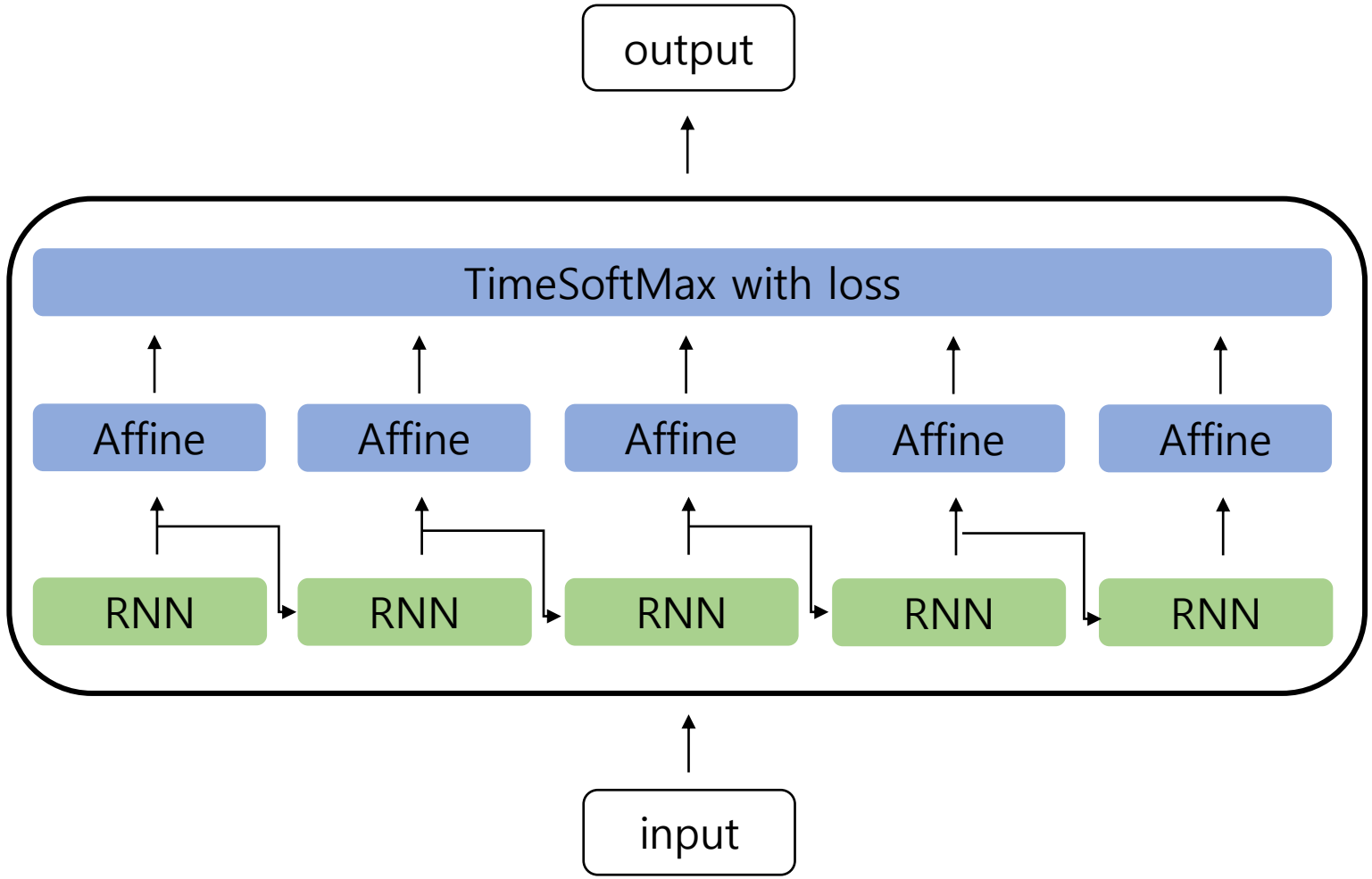


d

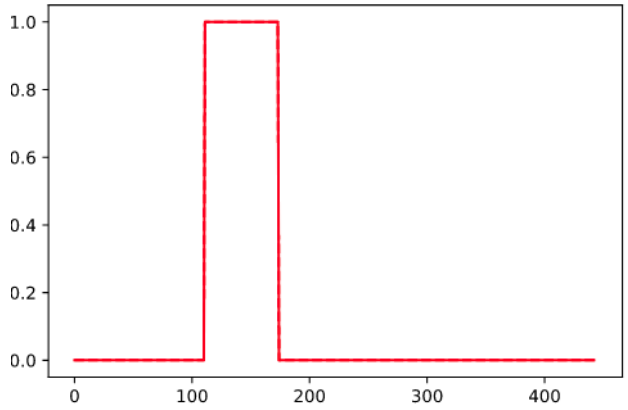


f

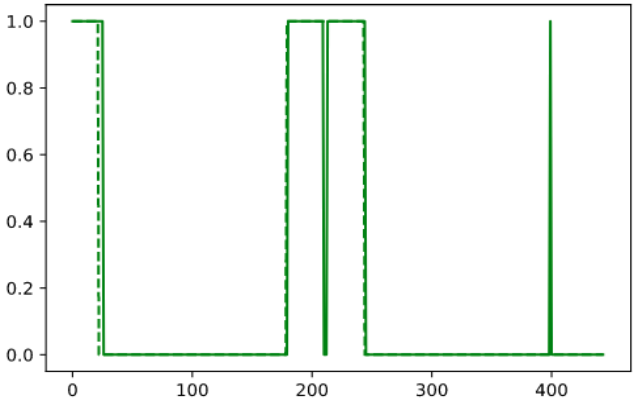
Model(RNN)


output
TimeSoftMax with loss
Affine
Affine
Affine
Affine
Affine
RNN
RNN
RNN
RNN
RNN
input

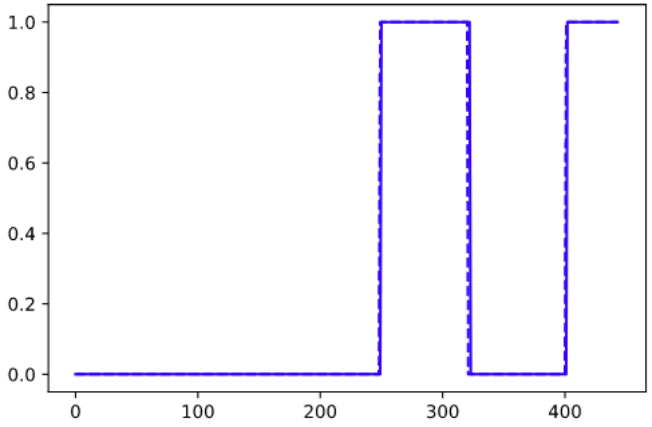
Model(RNN)



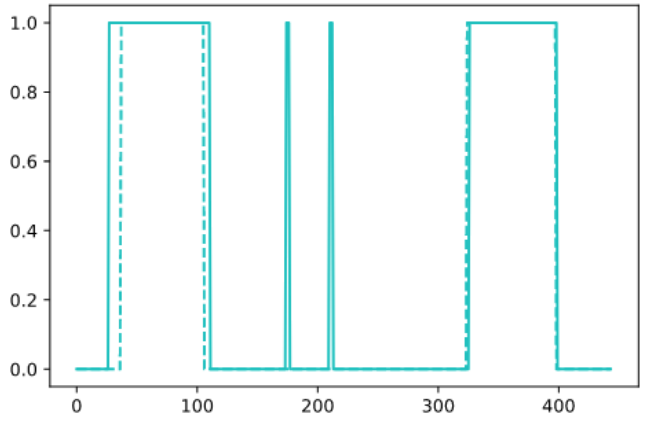
a



s

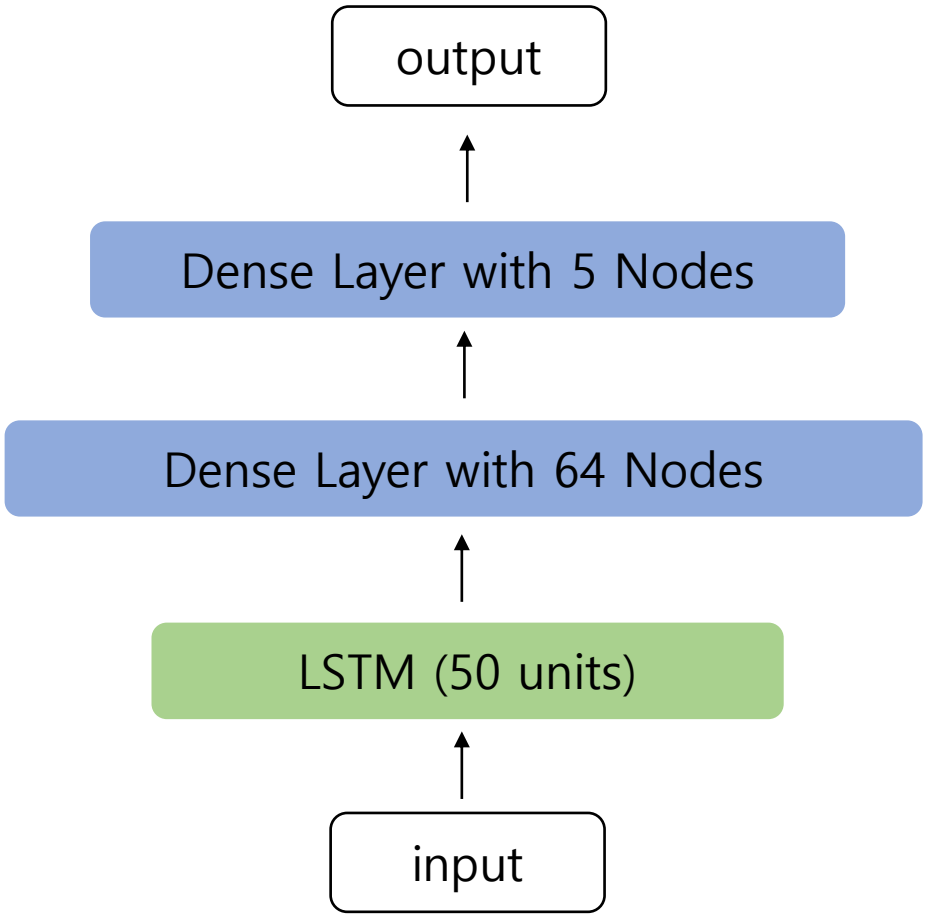


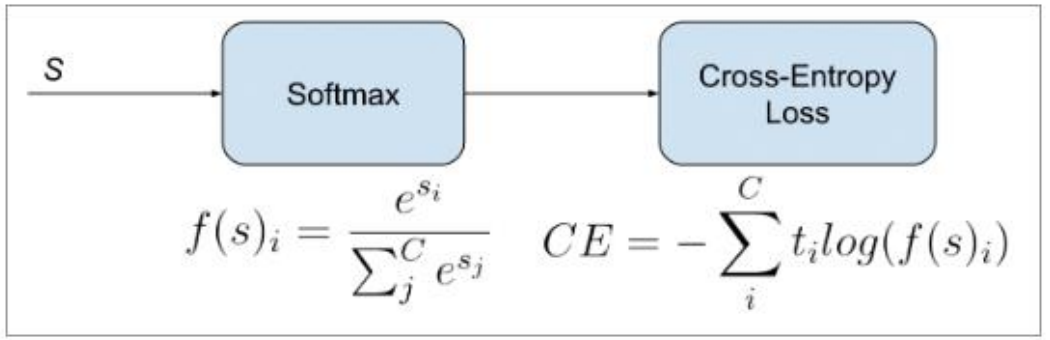
d



f

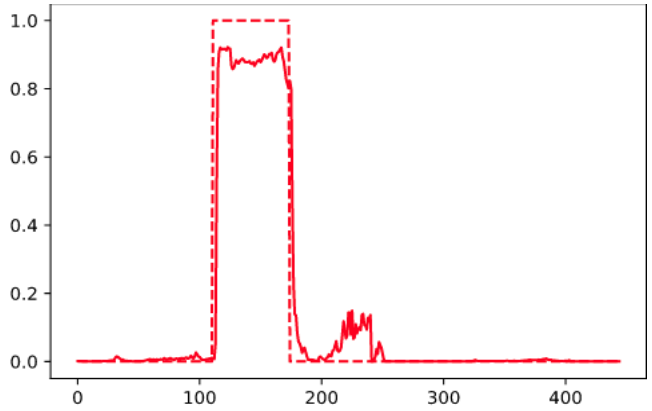
Model(LSTM)



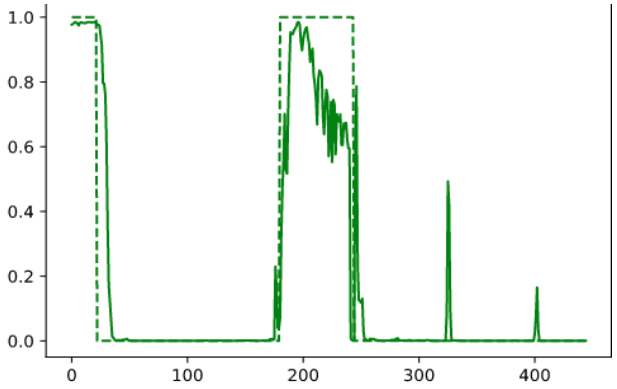



```
model = Sequential()
model.add(LSTM(50, input_shape=(63,1)))
model.add(Dense(64, activation='relu'))
model.add(Dense(5, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

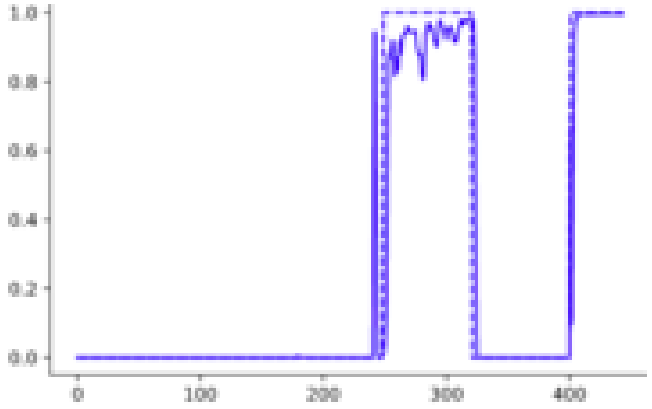
Model(LSTM)



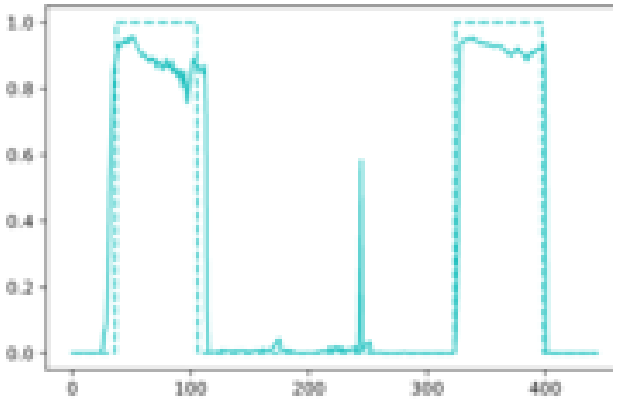
a



s

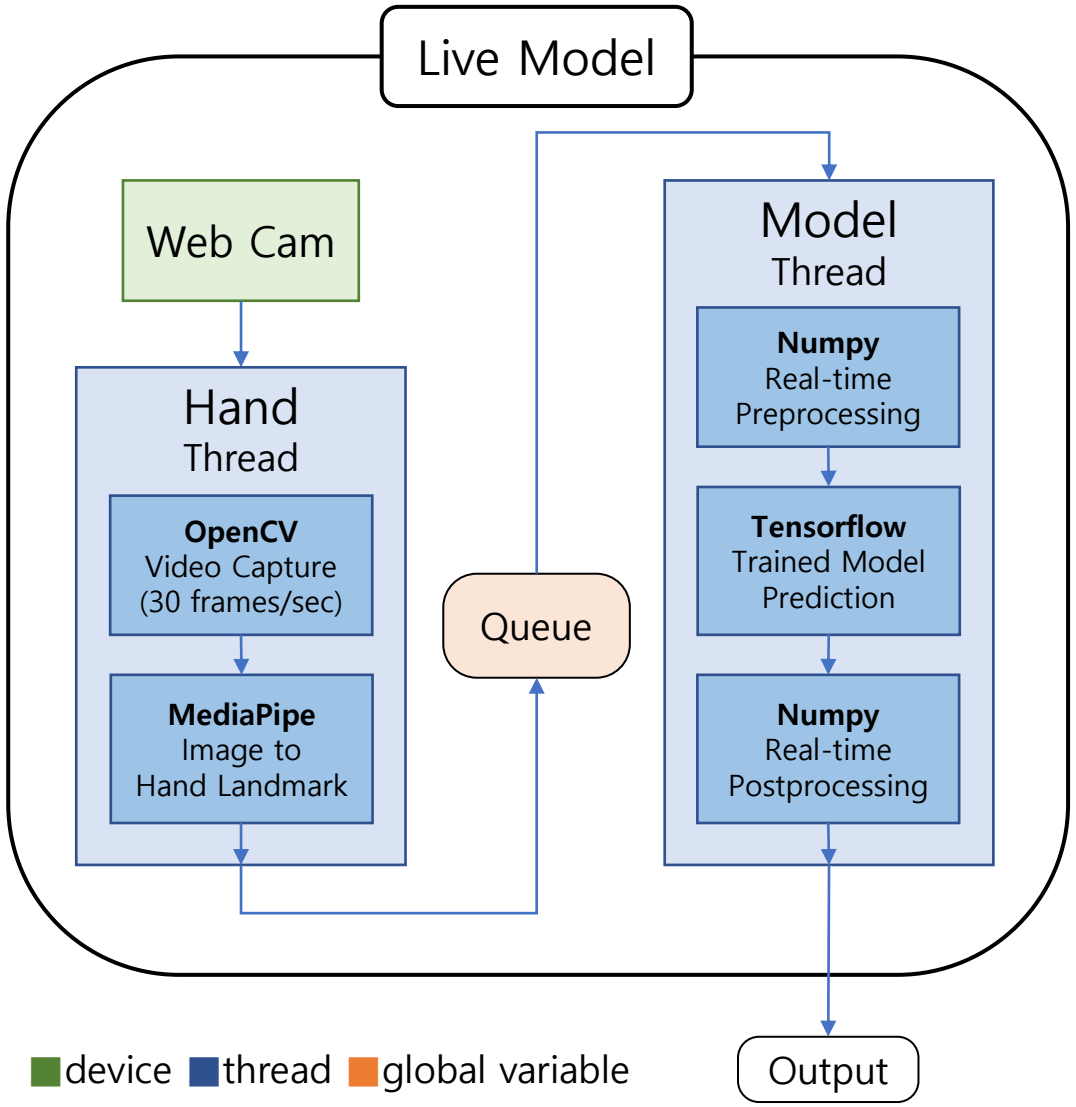


d

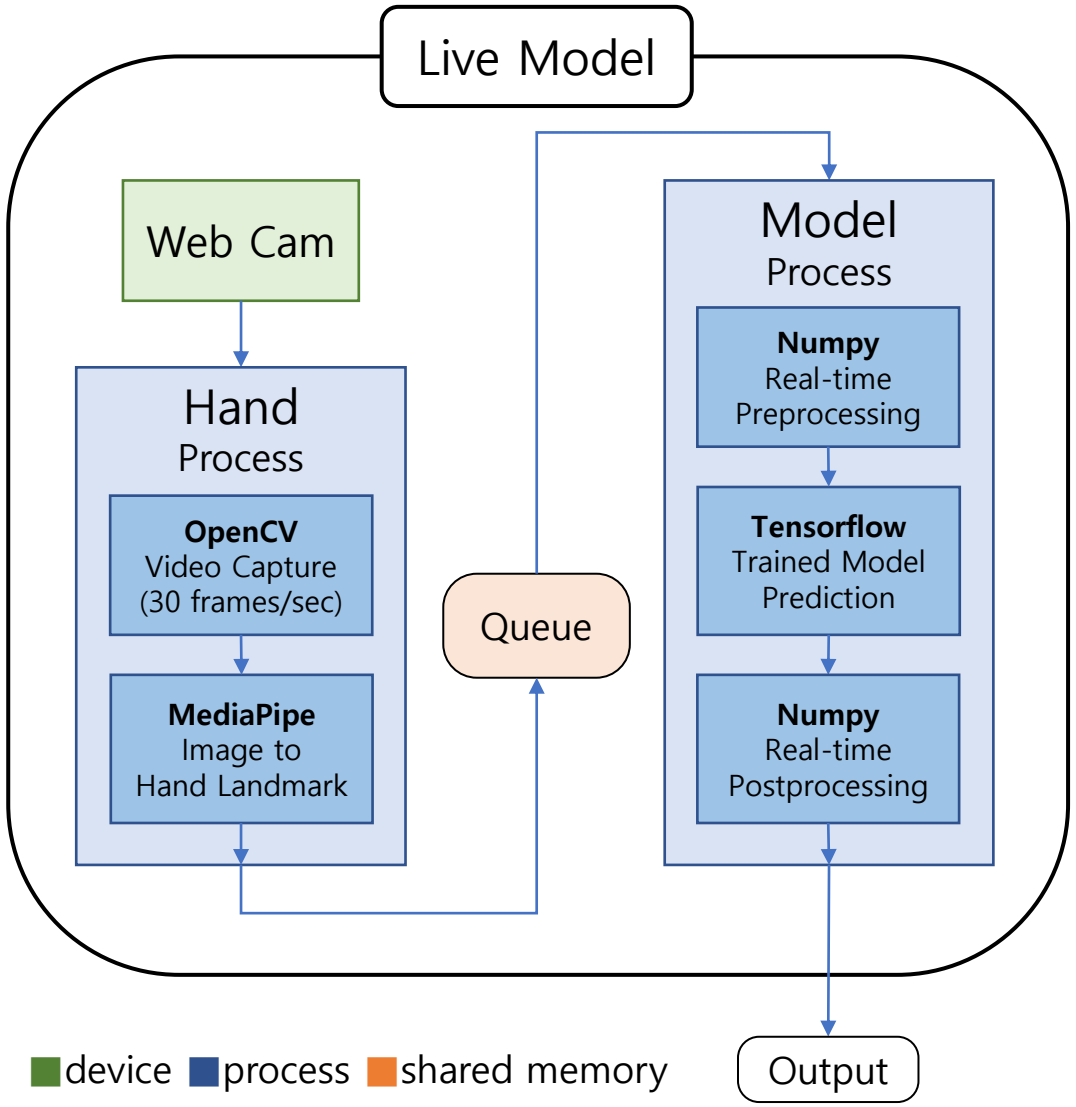


f

Live Model


Live Model
Web Cam
Hand
Thread
OpenCV
Video Capture
(30 frames/sec)
MediaPipe
Image to
Hand Landmark
Queue
Model
Thread
Numpy
Real-time
Preprocessing
Tensorflow
Trained Model
Prediction
Numpy
Real-time
Postprocessing
device
thread
global variable
Output

Live Model


Live Model
Web Cam
Hand
Process
OpenCV
Video Capture
(30 frames/sec)
MediaPipe
Image to
Hand Landmark
Queue
Model
Process
Numpy
Real-time
Preprocessing
Tensorflow
Trained Model
Prediction
Numpy
Real-time
Postprocessing
device
process
shared memory
Output

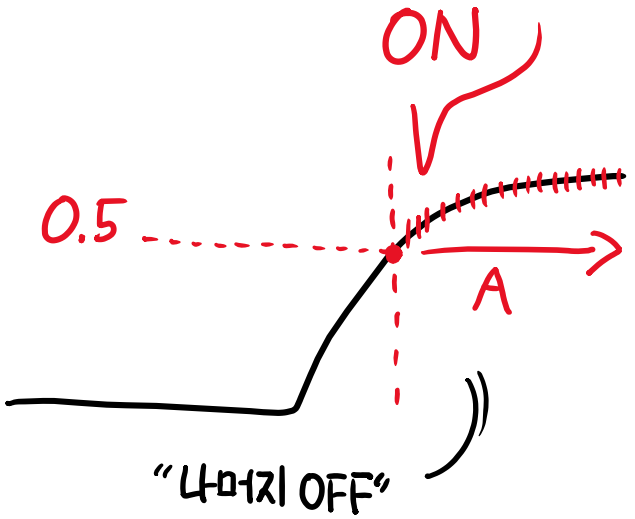
Real-time Preprocessing

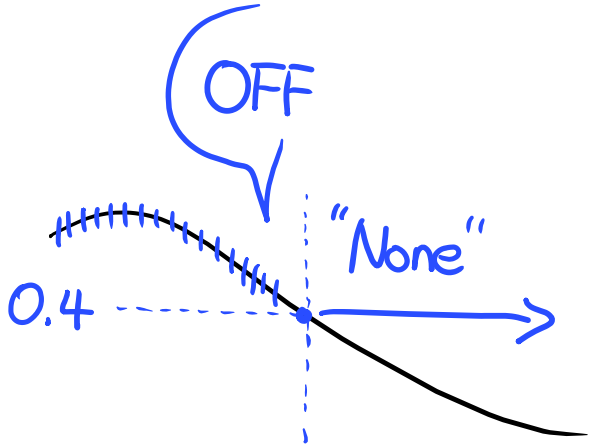
```python
class GammaSmoothing:
    def __init__(self, gamma=0.4):
        self.gamma = gamma
        self.prev  = None

    def process(self, x: np.ndarray) -> np.ndarray:
        if self.prev is None:
            self.prev = x
            return x

        result = (1-self.gamma)*self.prev + self.gamma*x
        self.prev = result
        return result

    def reset(self):
        self.prev = None
```

```python
class FixedMinMax:
    def __init__(self, fixed_minmax_filepath):
        import pandas as pd
        x_names = ['L%d%c' % (i, c) for i in range(21) for c in ['x', 'y', 'z']]
        y_names = ['a', 's', 'd', 'f']
        col_names = x_names + y_names

        df = pd.read_csv(fixed_minmax_filepath, names=col_names)
        df = df[x_names]

        self.fix_min = df.min().values
        self.fix_max = df.max().values

    def process(self, x: np.ndarray) -> np.ndarray:
        value = (x - self.fix_min) / (self.fix_max - self.fix_min)
        # value[np.where(value < 0)] = 0
        # value[np.where(value > 1)] = 1

        return value

    def reset(self):
        pass
```

Real-time Postprocessing

```python
class LetterTrigger:
    def __init__(self, up_threashold=0.5, down_threashold=0.4):
        self.up_threashold = up_threashold
        self.down_threashold = down_threashold
        self.activate_list = np.array([0,0,0,0,1])
        self.key_mapping = ['A','S','D','F','~']

    def process(self, x: np.ndarray):
        max_index = x.argmax()
        min_index = x.argmin()
        before_max_index = self.activate_list.argmax()
        x = x[0]

        if x[max_index] > self.up_threashold:
            if self.activate_list[max_index] == 0:
                self.activate_list[before_max_index] = 0
                self.activate_list[max_index] = 1
                print(self.key_mapping[max_index])

        elif self.activate_list[4] != 1 and x[before_max_index] < self.down_threashold:
                self.activate_list[before_max_index] = 0
                self.activate_list[4] = 1
                print("~")

    def reset(self):
        self.activate_list = np.ndarray([0,0,0,0,1])
```

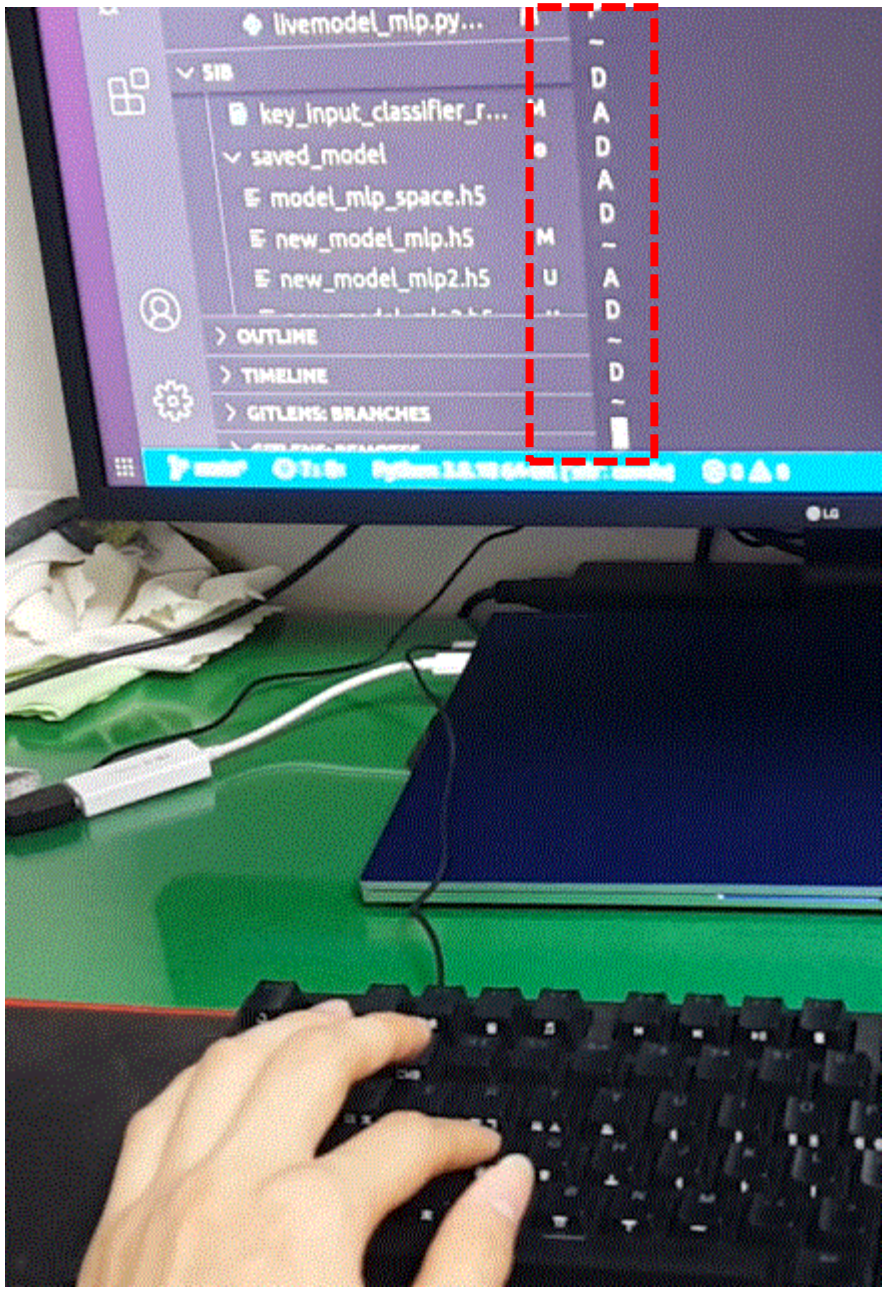




Result

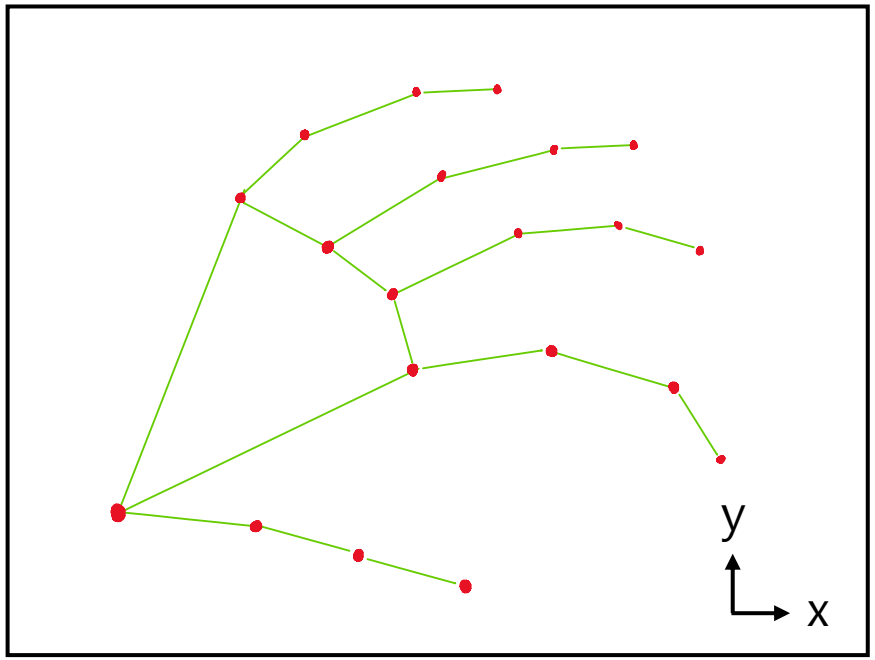
- Live model 검증
- A / S / D / F / none(~)
- 다섯가지 상태를 정확히 예측

- 한계점
  - 손의 위치, 카메라의 각도 등
    데이터셋 생성 당시의 환경과 달라지면
    정확도가 급격히 떨어짐

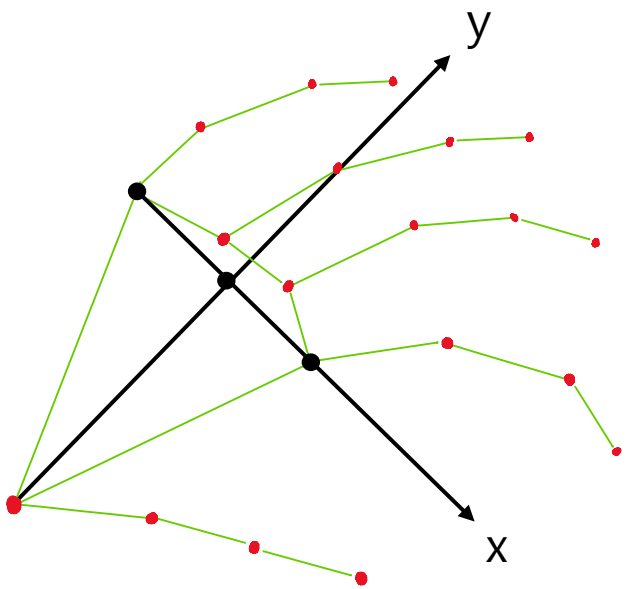


Idea

Coordinate Transformation



Camera Coordinate



Hand Coordinate

Achievement

목표 완성도 : 100%

달성 완성도 : 90%

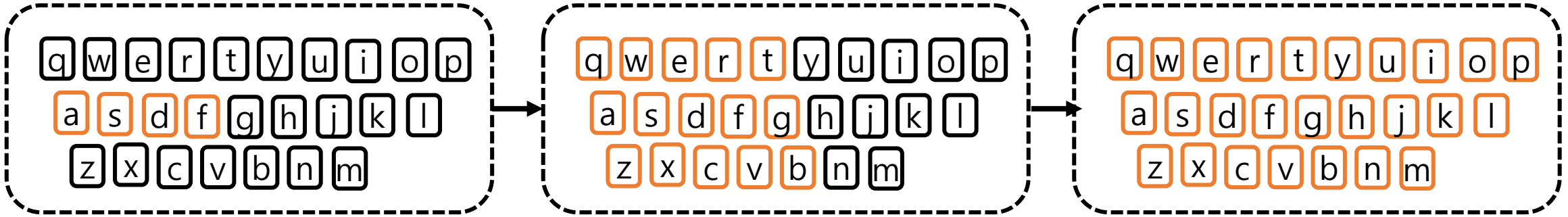
Future Works

- Better pre-/post-processing method
  - Coordinate transformation을 통한 좌표 값의 일관성 향상
  - 더 정교한 후처리 기법 고안

- Higher Accuracy
  - 더 많은 양질의 데이터셋 생성
  - Live model에 LSTM 등 다양한 모델 테스트

Future Works

asdf -> qwertasdfgzxcvb -> qwertyuiopasdfghjklzxcvbnm



Thank you