Gesture Detection

December 12, 2021

0.1 Instrukcja przedstawiająca proces uczenia modelu rozpoznającego dowolne gesty dłoni.

1 1. Rysowanie Dłoni

Rozpoznawanie dłoni polega na wyznaczeniu pozycji elementów charakterystycznych dłoni. W sumie można ich wyznaczyć 21. Są to między innymi stawy, nadgarstek lub końcówki palców. Współrzędne są obliczne względem lewego górnego rogu obrazu kamery.

Rozpoczynamy od zaimportowania odpowiednich bibliotek.

OpenCV pozowli na przeprowadzenie wstępnych przekształceń obrazu, w taki sposób, aby biblioteka MediaPipe mogła poprawnie rozpoznać dłoń oraz jej elementy charakterystyczne.

[15]:
```
import mediapipe as mp
import cv2
import numpy as np
```

Wybieramy dwa obiekty klasy mp.solutions:

1. mp_drawing - pozowli na naniesienie punktów na elementy charakterystyczne dłoni oraz linii ich łączących.
2. mp_hands - zostanie wykorzystany do rozpoznania dłoni z wybraną dokładnością.

[16]:
```
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
```

Wstępne ropoznanie dłoni i naniesienie grafiki na obraz pobrany z kamery.

2 2. Zapis pozycji elementów charakterystycznych do pliku CSV

[17]:
```
import csv
import os
import numpy as np
```

Tworzymy oznaczenia kolumn (klasy, współrzędne)

[18]:
```
landmarks = ['class']
for val in range(1, 21):
```

```
    landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val)]
```

[6]:
```
landmarks
```

[6]:
```
['class',
 'x0',
 'y0',
 'z0',
 'x1',
 'y1',
 'z1',
 'x2',
 'y2',
 'z2',
 'x3',
 'y3',
 'z3',
 'x4',
 'y4',
 'z4',
 'x5',
 'y5',
 'z5',
 'x6',
 'y6',
 'z6',
 'x7',
 'y7',
 'z7',
 'x8',
 'y8',
 'z8',
 'x9',
 'y9',
 'z9',
 'x10',
 'y10',
 'z10',
 'x11',
 'y11',
 'z11',
 'x12',
 'y12',
 'z12',
 'x13',
 'y13',
 'z13',
```

```
 'x14',
 'y14',
 'z14',
 'x15',
 'y15',
 'z15',
 'x16',
 'y16',
 'z16',
 'x17',
 'y17',
 'z17',
 'x18',
 'y18',
 'z18',
 'x19',
 'y19',
 'z19',
 'x20',
 'y20',
 'z20']
```

Tworzymy plik CSV i zapisujemy do niego oznaczenia kolumn.

[19]:
```
FILE_NAME='wrist_normalized_positions.csv'
```

[20]:
```
with open(FILE_NAME, mode='w', newline='') as f:
    csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.
 ↪QUOTE_MINIMAL)
    csv_writer.writerow(landmarks)
```

Tworzymy zmienną **class_name**, która będzie przechowywała informację o aktualnie przechwytywanym geście. W momencie rozpoczęcia tej części programu, będziemy zapisywać wszystkie współrzędne elementów charakterystycznych dla wybranego gestu.

[23]:
```
class_name="fist"
```

[24]:
```
cap = cv2.VideoCapture(0)

detections = 0
with mp_hands.Hands(min_detection_confidence=0.8, min_tracking_confidence=0.5)␣
 ↪as hands:
    while cap.isOpened():
        ret, frame = cap.read()

        #BGR to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
        #Flip horizontal
        image = cv2.flip(image, 1)

        #Set flag
        image.flags.writeable = False

        #Detections
        results = hands.process(image)

        #Set flag back to True
        image.flags.writeable = True

        #RGB to BGR
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        #print(results)

        #Rendering results
        if results.multi_hand_landmarks:
            for num, hand in enumerate(results.multi_hand_landmarks):
                mp_drawing.draw_landmarks(image, hand, mp_hands.HAND_CONNECTIONS,
                                          mp_drawing.DrawingSpec(color=(0,255,0),␣
↪thickness=2, circle_radius=4),
                                          mp_drawing.DrawingSpec(color=(0,0,255),␣
↪thickness=2, circle_radius=4))

        try:
            hand_landmarks = results.multi_hand_landmarks[0].landmark
            wrist = hand_landmarks[0]

            #Układ współrzędny ustawiamy względem pozycji nadgarstka
            hand_landmarks_row = np.zeros((20,3))
            for i in range(1, len(hand_landmarks)):
                hand_landmarks_row[i-1]=[hand_landmarks[i].x-wrist.x,␣
↪hand_landmarks[i].y-wrist.y, hand_landmarks[i].z-wrist.z]

            # Zebraną macierz danych przekształcamy w wektor i normalizujemy␣
↪względem największej co do wartości bezwzględnej liczby.
            hand_landmarks_row = hand_landmarks_row.flatten()
            hand_landmarks_row = list(hand_landmarks_row/np.max(np.
↪absolute(hand_landmarks_row)))

            hand_landmarks_row.insert(0, class_name)


            with open(FILE_NAME, mode='a', newline='') as f:
```

```
                csv_writer = csv.writer(f, delimiter=',', quotechar='"',␣
↪quoting=csv.QUOTE_MINIMAL)
                csv_writer.writerow(hand_landmarks_row)
                detections += 1
        except:
            pass

        if detections == 500:
            break

        cv2.imshow("Gesture Training", image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

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

[13]:
```
len(hand_landmarks_row)
```

[13]: 61

3 3. Trening modeli z wykorzystaniem Scikit Learn

[25]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[26]:
```
df = pd.read_csv(FILE_NAME)
```

[27]:
```
df.head()
```

[27]:

	class	x1	y1	z1	x2	y2	z2	x3	\
0	open	-0.145129	-0.031657	-0.049891	-0.286565	-0.135506	-0.097751	-0.407040	
1	open	-0.151966	-0.068719	-0.026542	-0.290023	-0.175108	-0.063248	-0.402538	
2	open	-0.154023	-0.072011	-0.026425	-0.297388	-0.179660	-0.059356	-0.415849	
3	open	-0.150740	-0.071340	-0.024960	-0.290231	-0.179649	-0.054728	-0.404190	
4	open	-0.150375	-0.071224	-0.023871	-0.290663	-0.183199	-0.050535	-0.404244	

	y3	z3	...	z17	x18	y18	z18	x19	\
0	-0.201005	-0.149839	...	-0.205701	0.159795	-0.580274	-0.275263	0.189066	
1	-0.247342	-0.109925	...	-0.159945	0.108089	-0.620367	-0.205653	0.132412	
2	-0.253713	-0.101513	...	-0.151207	0.098926	-0.627229	-0.193536	0.121643	
3	-0.255424	-0.093624	...	-0.144194	0.091754	-0.626092	-0.185198	0.116435	
4	-0.263047	-0.086115	...	-0.135458	0.088946	-0.630362	-0.174885	0.112889	

	y19	z19	x20	y20	z20

```
0 -0.703311 -0.308286  0.210130 -0.816663 -0.331339
1 -0.729752 -0.228523  0.146252 -0.832854 -0.245732
2 -0.740013 -0.214478  0.133588 -0.843611 -0.229912
3 -0.734410 -0.205761  0.132213 -0.836915 -0.220996
4 -0.737980 -0.195269  0.128409 -0.839018 -0.210178

[5 rows x 61 columns]
```

[28]:
```
df.tail()
```

[28]:
```
     class        x1        y1        z1        x2        y2        z2  \
995   fist  0.417932  0.088436 -0.096360  0.754135  0.081900 -0.278739
996   fist  0.410915  0.088109 -0.095302  0.750889  0.084176 -0.285430
997   fist  0.399111  0.052180 -0.072884  0.725338  0.011546 -0.232225
998   fist  0.354725  0.042709 -0.087128  0.651962 -0.010755 -0.238476
999   fist  0.299697  0.037428 -0.101292  0.582615 -0.038480 -0.222069

            x3        y3        z3  ...       z17       x18       y18  \
995  0.810148  0.095274 -0.468271  ... -0.652607  0.035596 -0.159826
996  0.798059  0.098386 -0.482713  ... -0.672379 -0.013507 -0.137151
997  0.794225  0.001789 -0.402805  ... -0.589324  0.070717 -0.197640
998  0.700747 -0.045856 -0.395376  ... -0.528639  0.029000 -0.307909
999  0.612825 -0.135910 -0.341066  ... -0.383005 -0.023773 -0.561991

          z18       x19       y19       z19       x20       y20       z20
995 -0.710277 -0.008996 -0.091264 -0.649022 -0.029721 -0.301096 -0.617818
996 -0.725778 -0.055338 -0.076488 -0.662481 -0.080464 -0.289736 -0.633149
997 -0.637126  0.018903 -0.127941 -0.574709 -0.011059 -0.321808 -0.541439
998 -0.575988  0.002703 -0.200245 -0.523472 -0.014184 -0.351851 -0.495351
999 -0.403550 -0.014028 -0.378816 -0.363341 -0.017701 -0.445063 -0.339712

[5 rows x 61 columns]
```

[29]:
```
x = df.drop('class', axis=1)
y = df['class']
```

[20]:
```
x
```

[20]:
```
           x0        y0        z0        x1        y1        z1        x2  \
0   -0.355147 -0.032142 -0.059415 -0.711971 -0.267873 -0.109290 -0.874771
1   -0.348118 -0.111412 -0.044334 -0.660333 -0.349064 -0.082020 -0.800372
2   -0.329805 -0.072463 -0.088667 -0.663210 -0.325190 -0.132609 -0.803425
3   -0.325644 -0.042529 -0.107411 -0.676567 -0.321084 -0.150288 -0.814076
4   -0.313416  0.001278 -0.142923 -0.684182 -0.310132 -0.206950 -0.829213
..        ...       ...       ...       ...       ...       ...       ...
995 -0.108496 -0.131669  0.002777 -0.168349 -0.302872 -0.015401 -0.222697
996 -0.109571 -0.134762  0.002793 -0.166234 -0.300893 -0.014064 -0.215698
```

```
997 -0.105325 -0.140384  0.010819 -0.158385 -0.308314 -0.002116 -0.205401
998 -0.104438 -0.145411  0.015083 -0.157658 -0.315324  0.005441 -0.202828
999 -0.107509 -0.147371  0.014984 -0.166582 -0.319206  0.005130 -0.210805

           y2        z2        x3  ...       z17       x18       y18  \
0   -0.585776 -0.176378 -0.785738  ... -0.393786 -0.043493 -0.688421
1   -0.661933 -0.143653 -0.747354  ... -0.255892 -0.075983 -0.732788
2   -0.668499 -0.187836 -0.712904  ... -0.199188 -0.045779 -0.747948
3   -0.679980 -0.198461 -0.704761  ... -0.184859 -0.029472 -0.726842
4   -0.674107 -0.266548 -0.701651  ... -0.159237 -0.020853 -0.702601
..        ...       ...       ...  ...       ...       ...       ...
995 -0.451654 -0.037590 -0.285287  ... -0.167540  0.353134 -0.512154
996 -0.450004 -0.033814 -0.275328  ... -0.158413  0.352931 -0.524204
997 -0.458247 -0.018558 -0.264543  ... -0.151300  0.360903 -0.518527
998 -0.469660 -0.008486 -0.254065  ... -0.143496  0.355707 -0.519447
999 -0.479064 -0.008576 -0.255600  ... -0.139967  0.345722 -0.522866

          z18       x19       y19       z19  x20  y20  z20
0   -0.343707  0.005121 -0.648451 -0.273109  NaN  NaN  NaN
1   -0.215074 -0.006075 -0.691734 -0.159024  NaN  NaN  NaN
2   -0.160437  0.007321 -0.682202 -0.100647  NaN  NaN  NaN
3   -0.146908  0.026971 -0.658891 -0.085273  NaN  NaN  NaN
4   -0.111857  0.025162 -0.645926 -0.039078  NaN  NaN  NaN
..        ...       ...       ...       ...  ...  ...  ...
995 -0.182570  0.402230 -0.577371 -0.193260  NaN  NaN  NaN
996 -0.171414  0.397086 -0.590246 -0.180486  NaN  NaN  NaN
997 -0.162096  0.404405 -0.581302 -0.170278  NaN  NaN  NaN
998 -0.152268  0.400174 -0.579451 -0.158444  NaN  NaN  NaN
999 -0.145458  0.382965 -0.580134 -0.147861  NaN  NaN  NaN

[1000 rows x 63 columns]
```

[21]:
```
y
```

[21]:
```
0      fist
1      fist
2      fist
3      fist
4      fist
       ...
995    open
996    open
997    open
998    open
999    open
Name: class, Length: 1000, dtype: object
```

Wszystkie pobrane dane dzielimy na dwie części, pierwsza posłuży do trenowania, druga do testwowania.

[30]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
↪random_state=3451)
```

[23]:
```
type(y_train)
```

[23]:
```
pandas.core.series.Series
```

[11]:
```
x_train
```

[11]:
```
            x0        y0        z0        x1        y1        z1        x2  \
295 -0.292044 -0.045307 -0.120753 -0.601194 -0.340314 -0.181280 -0.689188
296 -0.266880 -0.011237 -0.156898 -0.577938 -0.334095 -0.237343 -0.670003
117  0.136355 -0.344878  0.064587  0.131343 -0.682229  0.061997  0.023484
422  0.293701 -0.244369  0.132324  0.350801 -0.523015  0.171205  0.223739
569 -0.091890 -0.184264  0.010634 -0.118926 -0.414225  0.008861 -0.106391
..        ...       ...       ...       ...       ...       ...       ...
456  0.275047 -0.163213 -0.015345  0.462741 -0.415370 -0.042566  0.502899
97  -0.350041 -0.080015  0.042218 -0.576590 -0.272303  0.052587 -0.666904
150  0.240303  0.165247 -0.388521  0.543414  0.099620 -0.583327  0.797213
654 -0.128641 -0.119461 -0.018614 -0.230274 -0.287478 -0.038325 -0.305148
95  -0.310046  0.055653 -0.150592 -0.695671 -0.121819 -0.202166 -0.907515

            y2        z2        x3  ...       z17       x18       y18  \
295 -0.715878 -0.249059 -0.514983  ... -0.156124  0.086355 -0.725097
296 -0.685700 -0.313390 -0.500107  ... -0.244575  0.114456 -0.661679
117 -0.888441  0.044076 -0.080605  ... -0.144917 -0.392129 -0.418296
422 -0.682709  0.183194  0.075970  ... -0.063056 -0.292071 -0.498660
569 -0.574150 -0.006241 -0.102249  ... -0.112552 -0.024978 -0.535783
..        ...       ...       ...  ...       ...       ...       ...
456 -0.664704 -0.094929  0.381717  ... -0.233755  0.109718 -0.758664
97  -0.540706  0.013732 -0.616325  ... -0.135026 -0.169884 -0.804405
150 -0.105692 -0.665089  1.000000  ... -0.257261  0.607581 -0.632273
654 -0.425181 -0.068415 -0.376470  ... -0.149520  0.266449 -0.638252
95  -0.463460 -0.249841 -0.818008  ... -0.075658 -0.133305 -0.760121

          z18       x19       y19       z19  x20  y20  z20
295 -0.114633  0.102864 -0.587833 -0.052388  NaN  NaN  NaN
296 -0.201599  0.128727 -0.552241 -0.131925  NaN  NaN  NaN
117 -0.109947 -0.337065 -0.337305 -0.087502  NaN  NaN  NaN
422 -0.027513 -0.253113 -0.399806 -0.017275  NaN  NaN  NaN
569 -0.117283 -0.086578 -0.572035 -0.116888  NaN  NaN  NaN
..        ...       ...       ...       ...  ...  ...  ...
456 -0.196599  0.075892 -0.656934 -0.149515  NaN  NaN  NaN
97  -0.099632 -0.076870 -0.698475 -0.049948  NaN  NaN  NaN
150 -0.262033  0.504659 -0.592551 -0.255958  NaN  NaN  NaN
```

```
654 -0.170190  0.300784 -0.705749 -0.184726  NaN  NaN  NaN
95  -0.040917 -0.074379 -0.629640  0.022101  NaN  NaN  NaN

[700 rows x 63 columns]
```

4 4. Trenowanie Klasyfikujących Modeli Uczenia Maszynowego

[52]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression, RidgeClassifier,␣
 ↪SGDClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import NearestCentroid
from sklearn.tree import DecisionTreeClassifier
```

Tworzymy słownik przechowywujący 4 metody uczenie maszynowego wraz z metodą normalizacji.

[53]:
```
pipelines = {
    'lr':make_pipeline(StandardScaler(), LogisticRegression()),
    'nc':make_pipeline(StandardScaler(), NearestCentroid()),
    'dt':make_pipeline(StandardScaler(), DecisionTreeClassifier()),
    'rd':make_pipeline(StandardScaler(), RidgeClassifier()),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gd':make_pipeline(StandardScaler(), SGDClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
    'nn':make_pipeline(StandardScaler(), MLPClassifier()),
}
```

[33]:
```
y_train
```

[33]:
```
295    fist
296    fist
117    fist
422    fist
569    open
       ...
456    fist
97     fist
150    fist
654    open
95     fist
Name: class, Length: 700, dtype: object
```

Trenujemy 4 różne modele jednocześnie.

!!! PRZETESTOWAĆ INNE METODY !!!

[62]:
```
fit_models = {}

for algorithm, pipeline in pipelines.items():
    model = pipeline.fit(x_train, y_train)
    fit_models[algorithm] = model
```

[64]:
```
fit_models['rf'].predict(x_test)
```

[64]:
```
array(['open', 'open', 'open', 'fist', 'open', 'open', 'open', 'fist',
       'open', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'fist',
       'fist', 'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist',
       'open', 'fist', 'fist', 'fist', 'fist', 'fist', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist', 'open',
       'open', 'open', 'fist', 'fist', 'fist', 'fist', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'open', 'fist', 'fist', 'open',
       'fist', 'fist', 'open', 'open', 'fist', 'fist', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'open', 'open', 'open', 'open', 'open',
       'open', 'fist', 'open', 'fist', 'open', 'fist', 'fist', 'fist',
       'fist', 'open', 'fist', 'open', 'open', 'open', 'open', 'fist',
       'open', 'open', 'open', 'open', 'open', 'open', 'open', 'open',
       'open', 'fist', 'open', 'open', 'open', 'open', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'open', 'open', 'fist', 'open',
       'open', 'fist', 'open', 'fist', 'open', 'open', 'open', 'fist',
       'fist', 'open', 'fist', 'fist', 'open', 'fist', 'open', 'open',
       'open', 'open', 'fist', 'fist', 'open', 'fist', 'fist', 'fist',
       'open', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'open',
       'open', 'fist', 'fist', 'fist', 'fist', 'open', 'fist', 'fist',
       'open', 'fist', 'open', 'fist', 'fist', 'fist', 'fist', 'open',
       'open', 'fist', 'open', 'open', 'open', 'fist', 'fist', 'open',
       'open', 'open', 'open', 'open', 'fist', 'open', 'fist', 'fist',
       'fist', 'open', 'open', 'open', 'fist', 'fist', 'open', 'open',
       'fist', 'fist', 'open', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'fist', 'fist', 'fist', 'open', 'open',
       'fist', 'open', 'open', 'fist', 'fist', 'open', 'fist', 'open',
       'open', 'open', 'open', 'fist', 'fist', 'fist', 'fist', 'open',
       'fist', 'open', 'open', 'fist', 'open', 'open', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist', 'fist',
       'open', 'fist', 'fist', 'fist', 'open', 'fist', 'fist', 'fist',
       'open', 'fist', 'fist', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'fist', 'fist', 'fist', 'fist', 'fist', 'fist', 'open',
```

```
       'fist', 'fist', 'open', 'open', 'open', 'open', 'open', 'fist',
       'open', 'fist', 'fist', 'fist'], dtype=object)
```

5 5. Ewaluacja Modelu

[64]:
```
from sklearn.metrics import accuracy_score
import pickle
```

Porównujemy dokładność każdego modelu wykorzystując funkcję accuracy_score

[63]:
```
for algorithm, model in fit_models.items():
    yhat = model.predict(x_test)
    print(f'{algorithm}, {round(accuracy_score(y_test, yhat)*100,2)}%')
```

```
lr, 99.0%
nc, 88.0%
dt, 99.67%
rd, 98.67%
rf, 99.67%
gd, 96.67%
gb, 99.0%
nn, 99.33%
```

[65]:
```
with open('gesture_recognition.pkl', 'wb') as f:
    pickle.dump(fit_models['rf'], f)
```

6 5. Detekcje

Powtórnie ładujemy model.

[66]:
```
with open('gesture_recognition.pkl', 'rb') as f:
    model = pickle.load(f)
```

[49]:
```
model
```

[49]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

[50]:
```
model.predict(x_test)
```

[50]:
```
array(['open', 'open', 'open', 'fist', 'open', 'open', 'open', 'fist',
       'open', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'fist',
       'fist', 'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist',
       'open', 'fist', 'fist', 'fist', 'fist', 'fist', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist', 'open',
```

```
       'open', 'fist', 'fist', 'fist', 'fist', 'fist', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'open', 'fist', 'fist', 'open',
       'fist', 'fist', 'open', 'open', 'fist', 'fist', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'open', 'open', 'open', 'open', 'open',
       'open', 'fist', 'open', 'fist', 'open', 'fist', 'fist', 'fist',
       'fist', 'open', 'fist', 'open', 'open', 'open', 'open', 'fist',
       'open', 'open', 'open', 'open', 'open', 'open', 'open', 'open',
       'open', 'fist', 'open', 'open', 'open', 'open', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'open', 'fist',
       'fist', 'fist', 'open', 'open', 'open', 'open', 'fist', 'open',
       'open', 'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist',
       'fist', 'open', 'fist', 'fist', 'open', 'fist', 'open', 'open',
       'open', 'fist', 'fist', 'fist', 'open', 'fist', 'fist', 'fist',
       'open', 'fist', 'open', 'open', 'fist', 'open', 'fist', 'open',
       'open', 'fist', 'fist', 'fist', 'fist', 'open', 'fist', 'fist',
       'open', 'fist', 'open', 'fist', 'fist', 'fist', 'fist', 'open',
       'open', 'fist', 'open', 'open', 'open', 'fist', 'fist', 'open',
       'open', 'open', 'open', 'open', 'fist', 'open', 'fist', 'fist',
       'fist', 'open', 'open', 'open', 'fist', 'fist', 'open', 'open',
       'fist', 'fist', 'open', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'open', 'open', 'open', 'fist', 'fist', 'open', 'open',
       'fist', 'open', 'open', 'fist', 'fist', 'open', 'fist', 'open',
       'open', 'open', 'open', 'fist', 'fist', 'fist', 'fist', 'open',
       'fist', 'open', 'open', 'fist', 'open', 'open', 'open', 'fist',
       'fist', 'open', 'fist', 'open', 'fist', 'open', 'fist', 'fist',
       'open', 'fist', 'open', 'fist', 'open', 'fist', 'fist', 'fist',
       'open', 'fist', 'fist', 'open', 'open', 'open', 'fist', 'open',
       'fist', 'fist', 'fist', 'fist', 'fist', 'fist', 'fist', 'open',
       'fist', 'fist', 'open', 'open', 'open', 'open', 'open', 'fist',
       'open', 'fist', 'fist', 'fist'], dtype=object)
```

[51]:
```
pd.DataFrame(x_test)
```

[51]:

	x1	y1	z1	x2	y2	z2	x3
311	-0.122366	0.154571	-0.031917	-0.199848	0.348932	-0.042279	-0.261205
472	-0.120036	-0.118734	-0.029220	-0.216168	-0.285095	-0.058979	-0.286563
387	-0.018149	-0.325881	-0.009420	0.062272	-0.614014	-0.007430	0.150967
560	-0.289820	0.108541	-0.205375	-0.734906	-0.080346	-0.264466	-0.946463
423	-0.117827	-0.138752	-0.033318	-0.211735	-0.323980	-0.066010	-0.290154
..	...	...	...	...	...	...	...
766	0.106971	-0.360454	0.092596	0.081788	-0.660642	0.103109	-0.037583
181	0.210523	0.083584	-0.035197	0.428733	0.052988	-0.064152	0.603579
896	-0.284715	-0.232632	0.005019	-0.449822	-0.521568	-0.017280	-0.466985
637	-0.300201	0.012477	-0.129525	-0.658218	-0.200261	-0.237174	-0.849936

```
765  0.032872 -0.372588  0.102737 -0.045548 -0.657220  0.114579 -0.167819

           y3        z3        x4  ...       z17       x18       y18  \
311  0.517757 -0.050960 -0.327744  ... -0.003933  0.232572  0.484272
472 -0.416182 -0.097176 -0.357530  ... -0.132790  0.215612 -0.546907
387 -0.824606 -0.017904  0.192865  ... -0.034878  0.672843 -0.164905
560 -0.400162 -0.294812 -0.861793  ...  0.026708 -0.162364 -1.000000
423 -0.462984 -0.106554 -0.373721  ... -0.126488  0.336963 -0.495040
..        ...       ...       ...  ...       ...       ...       ...
766 -0.805570  0.092885 -0.151875  ... -0.126584 -0.328395 -0.709786
181  0.009858 -0.095681  0.747452  ... -0.142963  0.289274 -0.735975
896 -0.781971 -0.079262 -0.379222  ... -0.252696  0.082159 -0.875625
637 -0.453128 -0.338899 -0.803452  ... -0.243938 -0.012868 -0.759016
765 -0.788998  0.099934 -0.262368  ... -0.145308 -0.441985 -0.628679

          z18       x19       y19       z19       x20       y20       z20
311 -0.016964  0.259329  0.536242 -0.026011  0.283341  0.584262 -0.032395
472 -0.171076  0.242087 -0.638599 -0.188892  0.254576 -0.727021 -0.201592
387 -0.062415  0.775462 -0.187457 -0.075748  0.870415 -0.223513 -0.079016
560 -0.044242 -0.080222 -0.733392 -0.014111 -0.096771 -0.623113  0.046288
423 -0.167951  0.411554 -0.564842 -0.191383  0.474222 -0.633846 -0.208356
..        ...       ...       ...       ...       ...       ...       ...
766 -0.071641 -0.302990 -0.609321 -0.037710 -0.262762 -0.541080 -0.025406
181 -0.197838  0.332259 -0.869786 -0.231096  0.372779 -0.984128 -0.254844
896 -0.313326  0.076459 -0.670760 -0.265768  0.146113 -0.561699 -0.207085
637 -0.292509  0.006543 -0.533807 -0.265404 -0.017603 -0.560098 -0.232358
765 -0.095732 -0.405659 -0.526059 -0.059475 -0.349275 -0.466140 -0.044703

[300 rows x 60 columns]
```

[68]:
```
import warnings
```

[74]:
```
cap = cv2.VideoCapture(0)

detections = 0
with mp_hands.Hands(min_detection_confidence=0.8, min_tracking_confidence=0.5)␣
 ↪as hands:
    while cap.isOpened():
        ret, frame = cap.read()

        #BGR to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        #Flip horizontal
        image = cv2.flip(image, 1)

        #Set flag
```

```
        image.flags.writeable = False

        #Detections
        results = hands.process(image)

        #Set flag back to True
        image.flags.writeable = True

        #RGB to BGR
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        #Rendering results
        if results.multi_hand_landmarks:
            for num, hand in enumerate(results.multi_hand_landmarks):
                mp_drawing.draw_landmarks(image, hand, mp_hands.HAND_CONNECTIONS,
                                          mp_drawing.DrawingSpec(color=(0,255,0),␣
↪thickness=2, circle_radius=4),
                                          mp_drawing.DrawingSpec(color=(0,0,255),␣
↪thickness=2, circle_radius=4))

        try:
            hand_landmarks = results.multi_hand_landmarks[0].landmark
            wrist = hand_landmarks[0]

            hand_landmarks_row = np.zeros((20,3))
            for i in range(1, len(hand_landmarks)):
                hand_landmarks_row[i-1]=[hand_landmarks[i].x-wrist.x,␣
↪hand_landmarks[i].y-wrist.y, hand_landmarks[i].z-wrist.z]

            # print(hand_landmarks_row)
            hand_landmarks_row = hand_landmarks_row.flatten()
            hand_landmarks_row = list(hand_landmarks_row/np.max(np.
↪absolute(hand_landmarks_row)))

            #Make Detections
            x = pd.DataFrame([hand_landmarks_row])

            with warnings.catch_warnings():
                warnings.filterwarnings("ignore")
                gesture = str(model.predict(x)[0])
                cv2.putText(image, gesture, (10,20), cv2.FONT_HERSHEY_SIMPLEX,␣
↪1, (255,255,255), 2, cv2.LINE_AA)

        except:
            pass
```

```
        #image = cv2.flip(image, 0)
        cv2.imshow("Hand Tracking", image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    print(image.shape)

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

```
(480, 640, 3)
```

[ ]:
```
hand_landmarks_row
```