[Japanese/English]

hand-gesture-recognition-using-mediapipe

Estimate hand pose using MediaPipe(Python version).
This is a sample program that recognizes hand signs and finger gestures with a simple MLP using the detected key points.

This repository contains the following contents.

- Sample program
- Hand sign recognition model(TFLite)
- Finger gesture recognition model(TFLite)
- Learning data for hand sign recognition and notebook for learning
- Learning data for finger gesture recognition and notebook for learning

Requirements

- mediapipe 0.8.1
- OpenCV 3.4.2 or Later
- Tensorflow 2.3.0 or Later
  tf-nightly 2.5.0.dev or later (Only when creating a TFLite for an LSTM model)
- scikit-learn 0.23.2 or Later (Only if you want to display the confusion matrix)
- matplotlib 3.3.2 or Later (Only if you want to display the confusion matrix)

Demo

Here's how to run the demo using your webcam.

```
python app.py
```

The following options can be specified when running the demo.

- --device
  Specifying the camera device number (Default： 0)
- --width
  Width at the time of camera capture (Default： 960)
- --height
  Height at the time of camera capture (Default： 540)
- --use_static_image_mode
  Whether to use static_image_mode option for MediaPipe inference (Default： Unspecified)
- --min_detection_confidence

  Detection confidence threshold (Default： 0.5)
- --min_tracking_confidence

  Tracking confidence threshold (Default： 0.5)

Directory

```
│  app.py
│  keypoint_classification.ipynb
│  point_history_classification.ipynb
│
├─model
│  ├─keypoint_classifier
│  │  │  keypoint.csv
│  │  │  keypoint_classifier.hdf5
│  │  │  keypoint_classifier.py
│  │  │  keypoint_classifier.tflite
│  │  └─ keypoint_classifier_label.csv
│  │
│  └─point_history_classifier
│      │  point_history.csv
│      │  point_history_classifier.hdf5
│      │  point_history_classifier.py
│      │  point_history_classifier.tflite
│      └─ point_history_classifier_label.csv
│
└─utils
    └─cvfpscalc.py
```

app.py

This is a sample program for inference.

In addition, learning data (key points) for hand sign recognition,

You can also collect training data (index finger coordinate history) for finger gesture recognition.

keypoint_classification.ipynb

This is a model training script for hand sign recognition.

point_history_classification.ipynb

This is a model training script for finger gesture recognition.

model/keypoint_classifier

This directory stores files related to hand sign recognition.

The following files are stored.

- Training data(keypoint.csv)
- Trained model(keypoint_classifier.tflite)
- Label data(keypoint_classifier_label.csv)
- Inference module(keypoint_classifier.py)

model/point_history_classifier

This directory stores files related to finger gesture recognition.

The following files are stored.

- Training data(point_history.csv)
- Trained model(point_history_classifier.tflite)
- Label data(point_history_classifier_label.csv)
- Inference module(point_history_classifier.py)

utils/cvfpscalc.py

This is a module for FPS measurement.

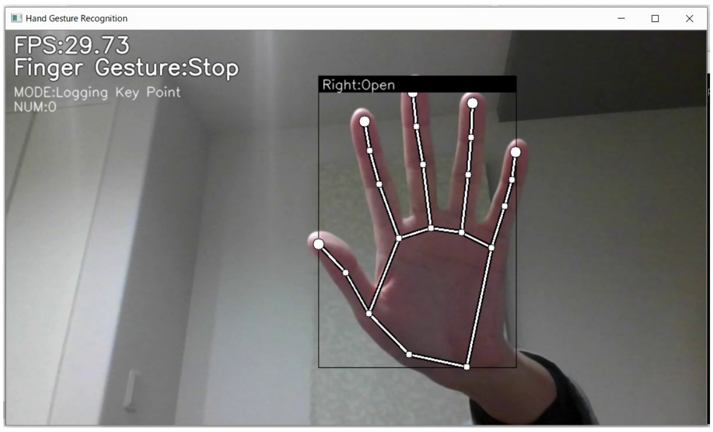
Training

Hand sign recognition and finger gesture recognition can add and change training data and retrain the model.

Hand sign recognition training

1.Learning data collection

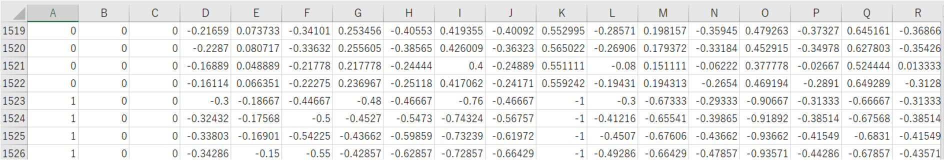
Press "k" to enter the mode to save key points（displayed as 「MODE:Logging Key Point」）



FPS:29.73
Finger Gesture:Stop
MODE:Logging Key Point
NUM:0
Right:Open

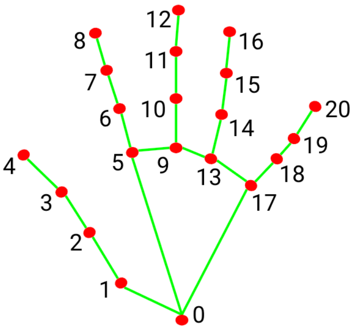
If you press "0" to "9", the key points will be added to "model/keypoint_classifier/keypoint.csv" as shown below.

1st column: Pressed number (used as class ID), 2nd and subsequent columns: Key point coordinates

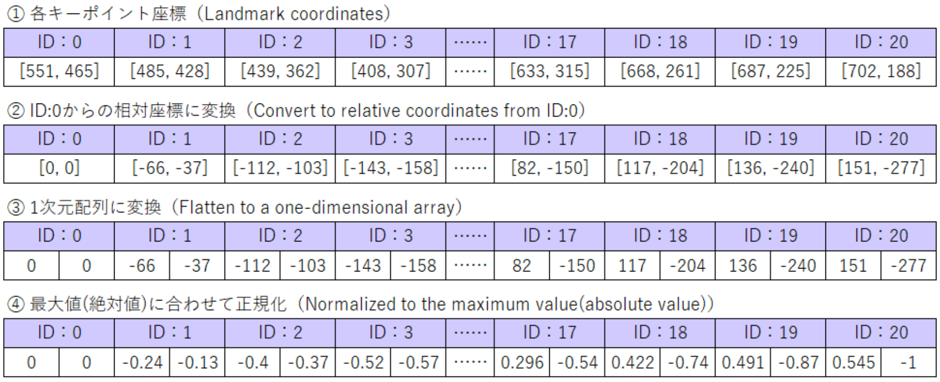


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1519	0	0	0	-0.21659	0.073733	-0.34101	0.253456	-0.40553	0.419355	-0.40092	0.552995	-0.28571	0.198157	-0.35945	0.479263	-0.37327	0.645161	-0.36866
1520	0	0	0	-0.2287	0.080717	-0.33632	0.255605	-0.38565	0.426009	-0.36323	0.565022	-0.26906	0.179372	-0.33184	0.452915	-0.34978	0.627803	-0.35426
1521	0	0	0	-0.16889	0.048889	-0.21778	0.217778	-0.24444	0.4	-0.24889	0.551111	-0.08	0.151111	-0.06222	0.377778	-0.02667	0.524444	0.013333
1522	0	0	0	-0.16114	0.066351	-0.22275	0.236967	-0.25118	0.417062	-0.24171	0.559242	-0.19431	0.194313	-0.2654	0.469194	-0.2891	0.649289	-0.3128
1523	1	0	0	-0.3	-0.18667	-0.44667	-0.48	-0.46667	-0.76	-0.46667	-1	-0.3	-0.67333	-0.29333	-0.90667	-0.31333	-0.66667	-0.31333
1524	1	0	0	-0.32432	-0.17568	-0.5	-0.4527	-0.5473	-0.74324	-0.56757	-1	-0.41216	-0.65541	-0.39865	-0.91892	-0.38514	-0.67568	-0.38514
1525	1	0	0	-0.33803	-0.16901	-0.54225	-0.43662	-0.59859	-0.73239	-0.61972	-1	-0.4507	-0.67606	-0.43662	-0.93662	-0.41549	-0.6831	-0.41549
1526	1	0	0	-0.34286	-0.15	-0.55	-0.42857	-0.62857	-0.72857	-0.66429	-1	-0.49286	-0.66429	-0.47857	-0.93571	-0.44286	-0.67857	-0.43571

The key point coordinates are the ones that have undergone the following preprocessing up to ④.



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



① 各キーポイント座標（Landmark coordinates）

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[551, 465]	[485, 428]	[439, 362]	[408, 307]	……	[633, 315]	[668, 261]	[687, 225]	[702, 188]

② ID:0からの相対座標に変換（Convert to relative coordinates from ID:0）

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[0, 0]	[-66, -37]	[-112, -103]	[-143, -158]	……	[82, -150]	[117, -204]	[136, -240]	[151, -277]

③ 1次元配列に変換（Flatten to a one-dimensional array）

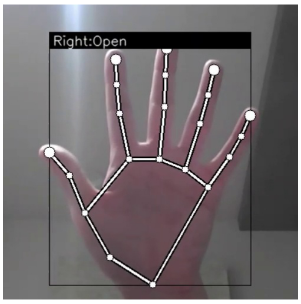
ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-66	-37	-112	-103	-143	-158	……	82	-150	117	-204	136	-240	151	-277

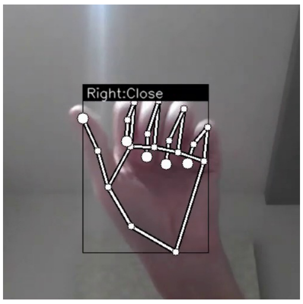
④ 最大値(絶対値)に合わせて正規化（Normalized to the maximum value(absolute value)）

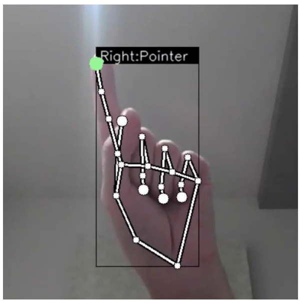
ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-0.24	-0.13	-0.4	-0.37	-0.52	-0.57	……	0.296	-0.54	0.422	-0.74	0.491	-0.87	0.545	-1

In the initial state, three types of learning data are included: open hand (class ID: 0), close hand (class ID: 1), and pointing (class ID: 2).

If necessary, add 3 or later, or delete the existing data of csv to prepare the training data.







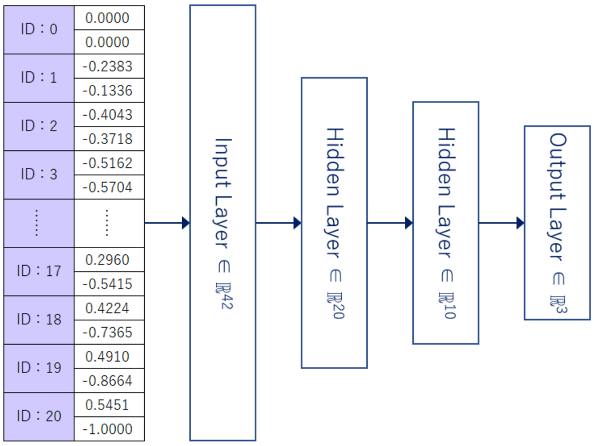
2.Model training

Open "keypoint_classification.ipynb" in Jupyter Notebook and execute from top to bottom.

To change the number of training data classes, change the value of "NUM_CLASSES = 3"
and modify the label of "model/keypoint_classifier/keypoint_classifier_label.csv" as appropriate.

X.Model structure

The image of the model prepared in "keypoint_classification.ipynb" is as follows.

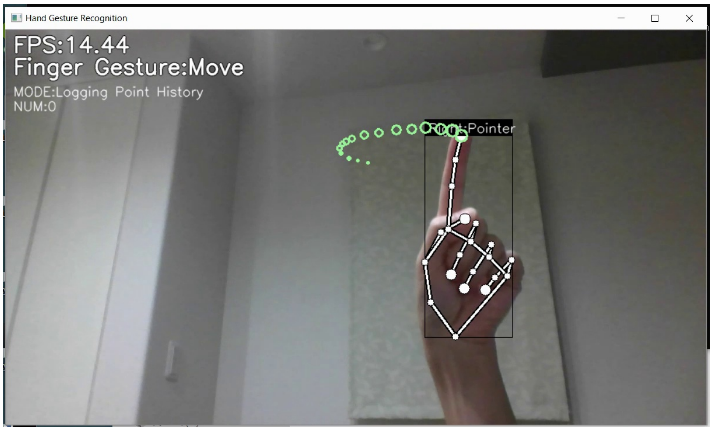


ID : 0	0.0000
	0.0000
ID : 1	-0.2383
	-0.1336
ID : 2	-0.4043
	-0.3718
ID : 3	-0.5162
	-0.5704
⋮	⋮
ID : 17	0.2960
	-0.5415
ID : 18	0.4224
	-0.7365
ID : 19	0.4910
	-0.8664
ID : 20	0.5451
	-1.0000

Finger gesture recognition training

1.Learning data collection

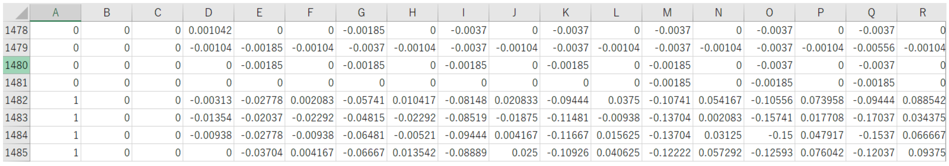
Press "h" to enter the mode to save the history of fingertip coordinates (displayed as "MODE:Logging Point History").



FPS:14.44
Finger Gesture:Move
MODE:Logging Point History
NUM:0

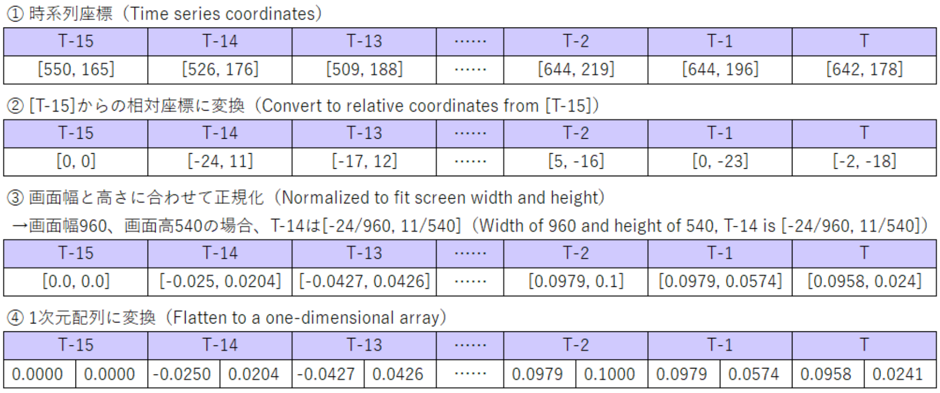
If you press "0" to "9", the key points will be added to "model/point_history_classifier/point_history.csv" as shown below.

1st column: Pressed number (used as class ID), 2nd and subsequent columns: Coordinate history



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1478	0	0	0	0.001042	0	0	-0.00185	0	-0.0037	0	-0.0037	0	-0.0037	0	-0.0037	0	-0.0037	0
1479	0	0	0	-0.00104	-0.00185	-0.00104	-0.0037	-0.00104	-0.0037	-0.00104	-0.0037	-0.00104	-0.0037	-0.00104	-0.0037	-0.00104	-0.00556	-0.00104
1480	0	0	0	0	-0.00185	0	-0.00185	0	-0.00185	0	-0.00185	0	-0.00185	0	-0.0037	0	-0.0037	0
1481	0	0	0	0	0	0	0	0	0	0	0	0	-0.00185	0	-0.00185	0	-0.00185	0
1482	1	0	0	-0.00313	-0.02778	0.002083	-0.05741	0.010417	-0.08148	0.020833	-0.09444	0.0375	-0.10741	0.054167	-0.10556	0.073958	-0.09444	0.088542
1483	1	0	0	-0.01354	-0.02037	-0.02292	-0.04815	-0.02292	-0.08519	-0.01875	-0.11481	-0.00938	-0.13704	0.002083	-0.15741	0.017708	-0.17037	0.034375
1484	1	0	0	-0.00938	-0.02778	-0.00938	-0.06481	-0.00521	-0.09444	0.004167	-0.11667	0.015625	-0.13704	0.03125	-0.15	0.047917	-0.1537	0.066667
1485	1	0	0	0	-0.03704	0.004167	-0.06667	0.013542	-0.08889	0.025	-0.10926	0.040625	-0.12222	0.057292	-0.12593	0.076042	-0.12037	0.09375

The key point coordinates are the ones that have undergone the following preprocessing up to ④.



① 時系列座標（Time series coordinates）

T-15	T-14	T-13	……	T-2	T-1	T
[550, 165]	[526, 176]	[509, 188]	……	[644, 219]	[644, 196]	[642, 178]

② [T-15]からの相対座標に変換（Convert to relative coordinates from [T-15]）

T-15	T-14	T-13	……	T-2	T-1	T
[0, 0]	[-24, 11]	[-17, 12]	……	[5, -16]	[0, -23]	[-2, -18]

③ 画面幅と高さに合わせて正規化（Normalized to fit screen width and height）
→画面幅960、画面高540の場合、T-14は[-24/960, 11/540]（Width of 960 and height of 540, T-14 is [-24/960, 11/540]）

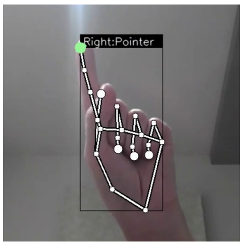
T-15	T-14	T-13	……	T-2	T-1	T
[0.0, 0.0]	[-0.025, 0.0204]	[-0.0427, 0.0426]	……	[0.0979, 0.1]	[0.0979, 0.0574]	[0.0958, 0.024]

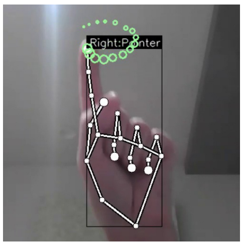
④ 1次元配列に変換（Flatten to a one-dimensional array）

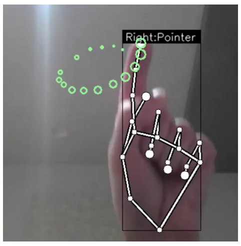
T-15		T-14		T-13		……	T-2		T-1		T	
0.0000	0.0000	-0.0250	0.0204	-0.0427	0.0426	……	0.0979	0.1000	0.0979	0.0574	0.0958	0.0241

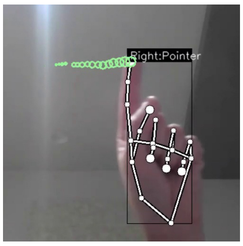
In the initial state, 4 types of learning data are included: stationary (class ID: 0), clockwise (class ID: 1), counterclockwise (class ID: 2), and moving (class ID: 4).

If necessary, add 5 or later, or delete the existing data of csv to prepare the training data.









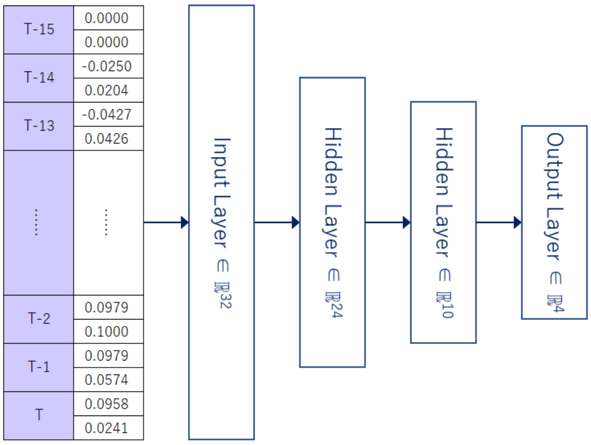
2.Model training

Open "point_history_classification.ipynb" in Jupyter Notebook and execute from top to bottom.

To change the number of training data classes, change the value of "NUM_CLASSES = 4" and modify the label of "model/point_history_classifier/point_history_classifier_label.csv" as appropriate.

X.Model structure

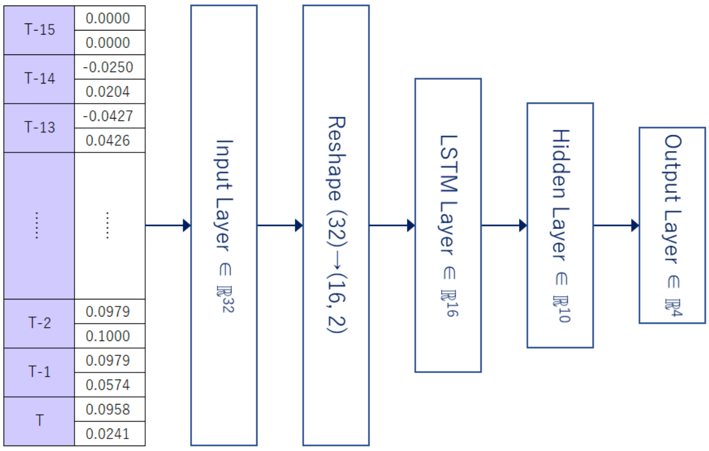
The image of the model prepared in "point_history_classification.ipynb" is as follows.



T-15	0.0000
	0.0000
T-14	-0.0250
	0.0204
T-13	-0.0427
	0.0426
⋮	⋮
T-2	0.0979
	0.1000
T-1	0.0979
	0.0574
T	0.0958
	0.0241

Input Layer ∈ $\mathbb{R}^{32}$
Hidden Layer ∈ $\mathbb{R}^{24}$
Hidden Layer ∈ $\mathbb{R}^{10}$
Output Layer ∈ $\mathbb{R}^{4}$

The model using "LSTM" is as follows.
Please change "use_lstm = False" to "True" when using (tf-nightly required (as of 2020/12/16))



T-15	0.0000
	0.0000
T-14	-0.0250
	0.0204
T-13	-0.0427
	0.0426
⋮	⋮
T-2	0.0979
	0.1000
T-1	0.0979
	0.0574
T	0.0958
	0.0241

Input Layer ∈ $\mathbb{R}^{32}$
Reshape (32)→(16, 2)
LSTM Layer ∈ $\mathbb{R}^{16}$
Hidden Layer ∈ $\mathbb{R}^{10}$
Output Layer ∈ $\mathbb{R}^{4}$