20K0008 井手海翔

ここでは、コードの動かし方やファイルについて説明する。詳しい手法やデータセットについての説明は抄録や本論文を読んでほしい。処理の全体像は次の図１のようになる。

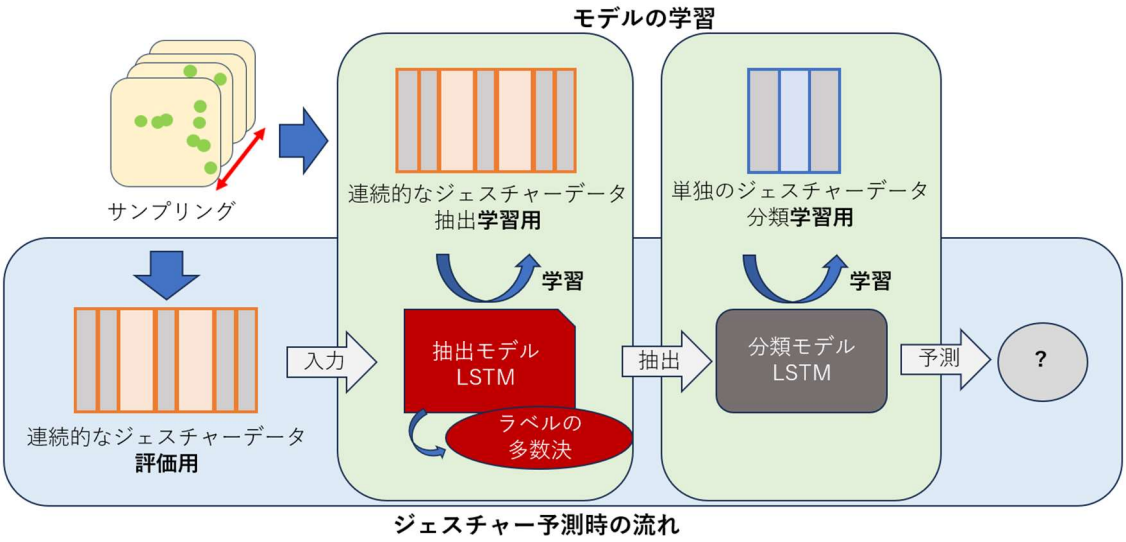
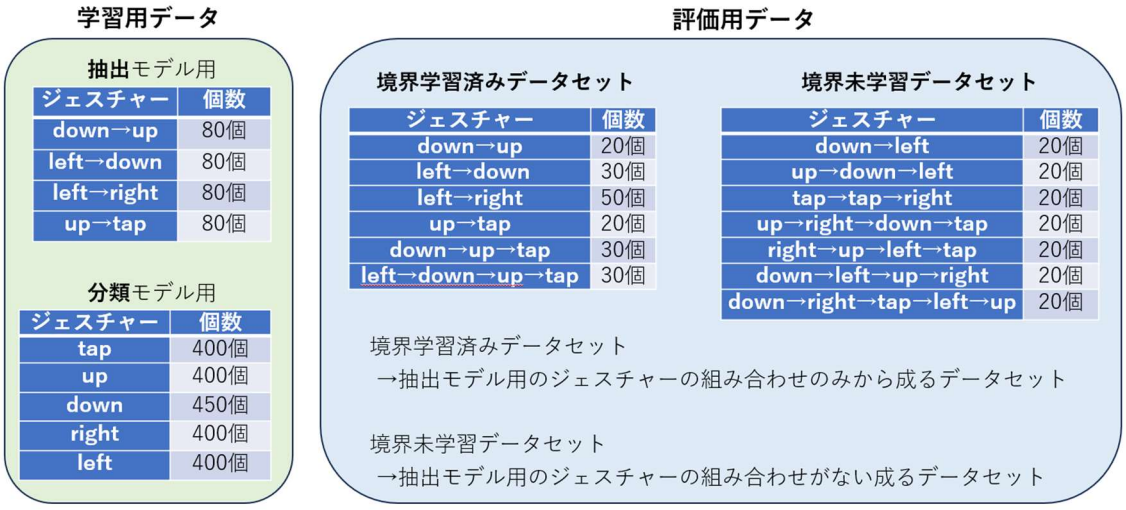


図１　全体像

データセット



学習用データ

抽出モデル用

ジェスチャー	個数
down→up	80個
left→down	80個
left→right	80個
up→tap	80個

分類モデル用

ジェスチャー	個数
tap	400個
up	400個
down	450個
right	400個
left	400個

評価用データ

境界学習済みデータセット

ジェスチャー	個数
down→up	20個
left→down	30個
left→right	50個
up→tap	20個
down→up→tap	30個
left→down→up→tap	30個

境界未学習データセット

ジェスチャー	個数
down→left	20個
up→down→left	20個
tap→tap→right	20個
up→right→down→tap	20個
right→up→left→tap	20個
down→left→up→right	20個
down→right→tap→left→up	20個

境界学習済みデータセット
→抽出モデル用のジェスチャーの組み合わせのみから成るデータセット

境界未学習データセット
→抽出モデル用のジェスチャーの組み合わせがない成るデータセット

図２　データセットについて

データセットは、学習用データと評価用データの２つに分けられる。これらのデータは Auto_cut_data_collection_from_mmradar/cut_frame_save_path 内にある。

学習用データは抽出モデル用と分類モデル用の 2 つあり、評価用データは 2 種類ある。分類用モデルの学習用データは only_ges_last_all であり、抽出用モデルの学習用データは origin_data である。また、評価用データについて、境界学習済みデータセットは countinuous_eval_kizon であり、境界未学習データセットは continuous_eval2 である。

ジェスチャー抽出と分類

図 1 のように連続的なジェスチャー認識のためには、抽出用モデルと、分類用モデルが必要である。それらのモデルを学習させるためのコードは learn ファイルにある。抽出用モデルは my_model.py であり、分類用モデルは gesture_model.py である。両コード共にコードは似ている。モデルは両方ともコード内の LSTM_LIN を用いた。また、path は Auto_cut_data_collection_from_mmradar/cut_frame_save_path/…/のように設定する。…の部分はデータセットのファイル名に設定する。例えば、分類用モデルの学習用データセットであれば only_ges_last_all とする。

続いてモデルに学習させる際の変数名の説明をする。LSTM に学習させる際にはデータの点群数とフレーム数を固定しなければならない。そのため、分類モデルではフレーム数を 50 とし、抽出モデルではウィンドーサイズ分の数を設定する。(5,10,15,20,25,30,35,40)
また、点群数は 1150 としてある
Framelength: 入力するフレーム数
Pointlength: 点群数の最大値
Class_number: 出力するクラス数（分類モデルでは 5、抽出モデルでは 3）

また、コードの最下部にエポック数や学習率を決める変数がある。ここでモデルの層構造やノードの数を決められる。
Epochs: エポック数
Optimizer_step_value: 学習率
Test_percent: テストデータの割合
Parameters: 特徴量の種類の数(x,y,z,dp)
Layer: 層の数（LSTM）

学習済みのモデルはファイルの model_list 内にある。

前処理は図 3 のように行う。x,y,z 座標において距離によってノイズ除去を行う。

```
for i in range(0, int(len(x_pos))):
    if (x_pos[i] <= 1) and (x_pos[i] >= -1):
        if (z_pos[i] <= 1.5) and (z_pos[i] >= -1):
            if (y_pos[i] <= 2.0):
                x[i] = x_pos[i]
                y[i] = y_pos[i]
                z[i] = z_pos[i]
                doppler[i] = doppler_pos[i]

framedata = np.array([x, y, z, doppler])
gesturedata[FrameNumber] = framedata
FrameNumber += 1
```

図３　ノイズ除去

評価用コードについて

評価用コードは 2 つある。1 つは手動抽出を評価するための only_ges_eval であり、2 つ目は、自動抽出を評価するための eval4_denoise である。まず、only_ges_eval について説明する。

変数

GES : データセットに該当するジェスチャー（eval_kizon: up_tap, left_down, down_up 等）

File_path: 評価するためのデータセットがあるパス

続いて、eval4_denoise について説明する。

変数

Framelength_spot: 抽出モデルのウィンドーサイズ(5,10,15,…40)

Framelength_calsi: 分類モデルのウィンドーサイズ(50)

Pointlength: 学習で用いた点群の最大数(1150)

Node_number: 学習で用いたモデルのノード数

Num_layer: 学習で用いたモデルの LSTM 層の数

Class_number: 抽出モデルのクラス分類数(3:　静止、ジェスチャー、境界の 3 つ)

Class_number_clasi: 分類モデルのクラス分類数（5: tap, right, left, up, down）

データの収集と可視化について

　データ収集は Auto_cut_data_collection_from_mmradar の gui_main を実行すると出来る。詳しいやり方は大学院生に聞いてほしい。

　データの可視化は mmwave_final の vis_new に入っている。この中の config ファイルを自分が可視化したいデータのパスに書き換えて、main_new3 を実行すれば出来る。これはノ

イズ除去後の点群が表示される。

以上で説明は終わりである。分からないことがあれば” kaito.ide.2t@stu.hosei.ac.jp”までメールしてください。