

# 原点位置・軸長読み取りの機械学習プログラム (Learning\_OrgAxl.py) 説明書

## 目次

1. 機能
2. 動作環境
3. 概要
  - 3.1 構造
  - 3.2 プログラムリストの概要
  - 3.3 入出力
    - 3.3.1 入力 と実行
    - 3.3.2 出力
4. パラメータ
5. プログラムリスト

## 1. 機能

このプログラム(Learning\_OrgAxl.py)は機械学習を行い、グラフ画像からグラフの原点とX,Y軸長を読み取る学習モデルを生成する。この学習モデルはグラフ数値化プログラム (WaveForm Digitizer.py) で用いる。

## 2. 動作環境

1) このプログラムはWindows10, Ubuntu18-04での動作が確認されている

2) このプログラムを実行するには、以下のモジュールがあらかじめ準備された環境が必要である。また、記載されている以外のVersionでの動作確認は行っていないので注意が必要である

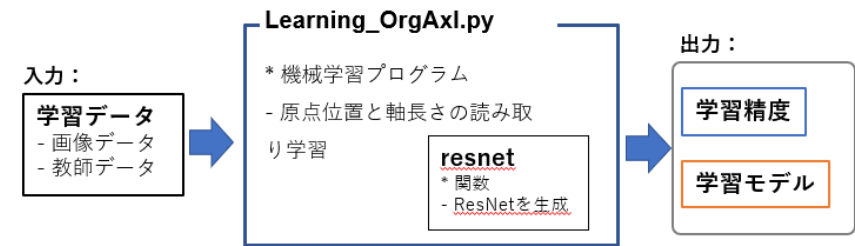
- Python 3.6
- Tensorflow-gpu 1.4.0
- Keras 2.2.4
- sklearn 0.19.1, 0.21.2
- matplotlib 2.2.2

2) このプログラムの存在するディレクトリ下に'data'および'model'の2つのサブディレクトリが必要である

## 3. 構造

### 3.1 概要

このプログラムは、学習用データ（画像データと教師データからなる）を入力とし、内部の関数 resntで生成するモデルに従って機械学習を行い、学習精度を出力し学習済モデルを生成する。



### 3.2 プログラムリストの概要

プログラムは「学習モデルを生成する関数」と「機械学習の実行」の2つのモジュールに分けられる

全リストは、5. プログラムリストに示す

1. 1-8 行: 開始コメント
2. 10-220 行: 学習モデルを生成する関数
  - 11-31 行: モジュールのインポート
  - 41-85 行: モデルパラメータの記述
  - 87-220 行: ResNetの生成
3. 222 - 524行: 機械学習の実行
  - 226-228 行: モジュールのインポート
  - 230-242 行: 開始準備
  - 244-284 行: データの読み込みと振り分け
  - 284-294 行: パラメータの設定
  - 296-318 行: モデル構築
  - 320-330 行: モデルコンパイル
  - 332-341 行: モデルフィット
  - 343-498 行: 結果の評価と出力
  - 500-524 行: モデルの保存
  - 526 行: END

### 3.3 入出力

#### 3.3.1 入力と実行

プログラムの実行は、まず、下記の様にプログラムの247行目に学習データの名前を記述して(例: ./data/xxx.pik)保存する  
次に、<2. 動作環境>を満たした動作環境下でPython上で実行する

```
244 # --- Data import -----
245 # data
246 # data_n = input("Data Name --> ")
247 data_n = './data/xxx.pik'
248 print("\n Data: ",data_n)
...
```

#### 学習用データの要件

学習用データは以下の要件を満たしている必要がある

- pickle形式で保存された辞書型のデータであること。 <データ名.pik >
- 3つのキー'in', 'out', 'para'を持ち、'in'に画像データ、'out'に画像データと1対1に対応した教師データが保存されていること

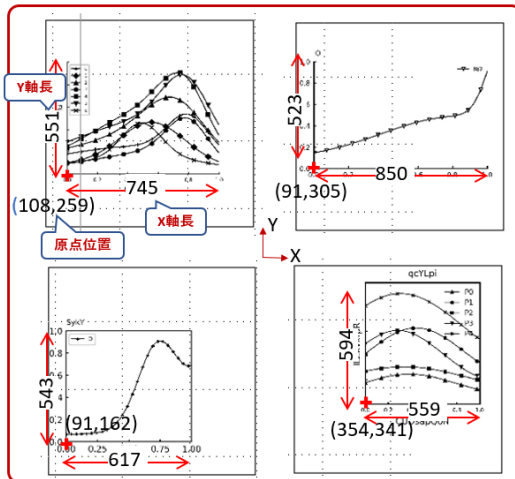
画像データ:

0-255の整数の数値をとり、そのサイズが1024x1024で1チャンネル(グレイタイプ)の画像配列(1024,1024,1)

教師データ:

0-1023の4個の整数値をとり、X原点座標位置、 y 原点座標位置、 X軸長さ、 Y 軸長さをその値とする整数配列(4)

画像データ:



教師データ:

[X-原点位置, Y-原点位置, X-軸長, Y-軸長]

[108, 259, 745, 551]  
[ 91, 305, 850, 523]  
[ 91, 162, 617, 543]  
[354, 341, 559, 594]

- キー'para'には7個の要素を持ち、2、4、5、6、7番目の要素には、下記の値が入っていること

key item

data 画像データ

out 教師データ

para -  
dpi,  
-,  
原点位置の最小値,  
原点位置の最大値,  
軸長の最小値,  
軸長の最大値

学習済みモデルおよびそのモデル形状は計算終了後、保存の可否を選ぶことができる。

以下に実際の出力例をしめす

### (1) 入力パラメータの確認

< RUN: Learning of Origin-position and Axis-length\_ 200206-0523 >

Data: ./data/OrgAxl\_Data\_r1\_1024\_10000\_20200116-0544.pik

Data dimension = 1024 , dpi = 360  
Origin\_range = 85.2 - 313.9  
X-axis\_length\_range = 702.1 - 949.1

Train\_n = 9000 , Test\_n = 1000

Process Start...

This is ResNet-18+2+2 with l2-regularizer  
2-branch Outputs

L2-Regularizer = 5e-06

Optimizer : adam  
first\_filter = 16  
batch = 60  
max epoch = 400  
patience = 100

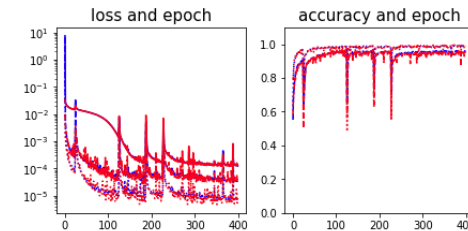
### (2) 精度と正答率

lossとaccの履歴と正答率の表示

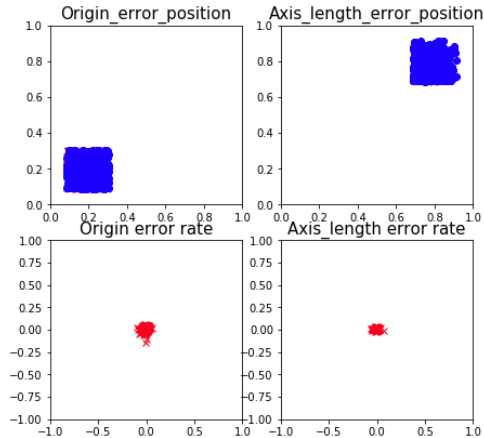
...Finished. Exe time (s) = 65874.07 , n\_ep = 400

< loss and accuracy >

loss = 1.24e-04 , org\_acc = 0.988 , axl\_acc = 0.961



< Pass\_Rate > (SI; org = +/- 0.030 , axl = +/- 0.030 )  
Org\_NG = 116 / 1000 , Pass\_Rate= 0.884  
Axs\_NG = 12 / 1000 , Pass\_Rate= 0.988

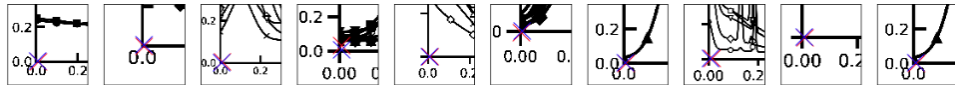


### (3) エラー

1) 原点がうまく予測できなかったものの中でランダムに10個を取り出して表示

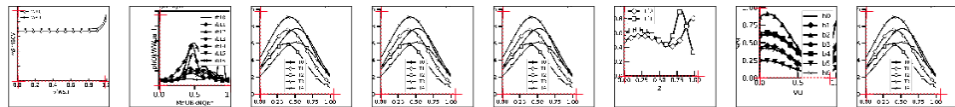
< NG figure sample >

- Org NG -  
369 , Error [-0.03554363 0.0408144 ]  
428 , Error [0.01052159 0.03203281]  
667 , Error [-0.04526536 -0.01755117]  
78 , Error [-0.00632197 -0.13845763]  
540 , Error [-0.03095355 -0.00403046]  
854 , Error [0.03181799 0.02743695]  
91 , Error [-0.02272581 0.03096249]  
792 , Error [-0.07308563 0.01131068]  
43 , Error [-0.04432924 0.01539852]  
91 , Error [-0.02272581 0.03096249]



2) 軸長がうまく予測できなかったものの中でランダムに10個を取り出して表示

79 , Error [0.00952129 0.03768249]  
71 , Error [-0.01612407 0.03173038]  
950 , Error [-0.06741782 0.01436839]  
950 , Error [-0.06741782 0.01436839]  
950 , Error [-0.06741782 0.01436839]  
132 , Error [-0.05058273 0.00189707]  
210 , Error [0.07402948 -0.01042222]  
950 , Error [-0.06741782 0.01436839]



### (4) モデル出力

モデルおよびモデル形状は計算終了後、下の様に入力待ち状態になるので、キーボードから y をタイプすることで保存できる。y 以外では保存されない

< Model Save and Print >  
Print the model shape? y or n --> y  
Model Shape is printed --> ./model/OrgAxl\_2-StepFit\_1024\_RN26\_bt60ft16ep400\_200212-0441.png

Save the learned model? y or n --> y  
Model is saved --> ./model/OrgAxl\_2-StepFit\_1024\_RN26\_bt60ft16ep400\_200212-0441.h5

## 4. パラメータ

このプログラムのパラメータは1024x1024の大きさの画像選択に最適化されている。しかし、学習データおよびパラメータは下記の様に変更可能であり、このプログラムの全体あるいは部分を利用して他の用途に転用することができる。ただし計算環境によってはGPUやCPUのメモリ不足などのため、計算可能な組み合わせには限界が生じる場合がある

変数名	説明	デフォルト値	可能な値
test_size ,train_size	データの訓練数とテスト数の割合	0.9,0.1	合計して 1
dim	画像の大きさ	1024	256,360,1024は検証済
m_n	ResNetの種類	RN26	RN18,RN34,RN50,RN26は検証済
reg_f	L2-Regularizer	5e-6	0-1
n_epoch	最大エポック数	400	1-
n_batch	バッチ数	60	1-
first_fil_n	開始フィルタ数	16	8,16,24,32は検証済
n_pat	EarlyStopで計算を打ち切る変動の最大数	100	1-
flag_ver	エポック毎の表示の選択	0	0:表示しない、1:表示する

## 5. プログラムリスト

プログラムは「学習モデルを生成する関数」と「機械学習の実行」の2つのモジュールに分けられる

全リストは、5. プログラムリストに示す

- 1-8 行: 開始コメント
- 10-220 行: 学習モデルを生成する関数  
11-31 行: モジュールのインポート  
41-85 行: モデルパラメータの記述  
87-220 行: ResNetの生成
- 222 - 524行: 機械学習の実行  
226-228 行: モジュールのインポート  
230-242 行: 開始準備  
244-284 行: データの読み込みと振り分け

284-294 行: パラメータの設定  
296-318 行: モデル構築  
320-330 行: モデルコンパイル  
332-341 行: モデルフィット  
343-498 行: 結果の評価と出力  
500-524 行: モデルの保存  
526 行: END

```
1  # -*- coding: utf-8 -*-
2  ""
3  Created on Jan.17,2020
4
5  < グラフ画像からグラフの原点位置とX,Y軸長を読み取る機械学習 >
6
7  @author: T.KONOで
8  ""
9
10 # ----- Model Build Function -----
11
12 """Residual Net を作成するための関数
13 """
14
15 import numpy as np
16 import matplotlib.pyplot as plt
17 from keras.callbacks import EarlyStopping
18 from keras.layers import (Input,
19                             Conv2D,
20                             Add,
21                             Activation,
22                             Dense,
23                             GlobalAveragePooling2D,
24                             MaxPooling2D
25                             )
26 from keras.models import (Model)
27 from keras.regularizers import l2
28
29 def resnet(input_shape,
30           num_outputs,
31           m_name,
32           n_fil,
33           l_f
34           ):
35
36     com = ' with l2-regularizer\n 2-branch Outputs'
37
38     n_filter = n_fil
39     reg_f = l_f
40
41     # --- Model parameter define -----
42     if m_name == 'RN18':
43         print('\n This is ResNet-18'+com)
44         nb_blocks = [2,2,2,2]
45         wide = 2
46         block = 'plane'
47
48     elif m_name == 'RN34':
49         print('\n This is ResNet-34'+com)
50         nb_blocks = [3,4,6,3]
51         wide = 2
```

```
52     block = 'plane'
53
54     elif m_name == 'RN50':
55         print('\n This is ResNet-50'+com)
56         nb_blocks = [3,4,6,3]
57         wide = 2
58         block = 'bottleneck'
59
60     elif m_name == 'RN101':
61         print('\n This is ResNet-101'+com)
62         nb_blocks = [3,4,23,3]
63         wide = 2
64         block = 'bottleneck'
65
66     elif m_name == 'RN152':
67         print('\n This is ResNet-152'+com)
68         nb_blocks = [3,8,36,3]
69         wide = 2
70         block = 'bottleneck'
71
72     elif m_name == 'RN26':
73         print('\n This is ResNet-26'+com)
74         nb_blocks = [2,2,2,2,2]
75         wide = 2
76         block = 'plane'
77
78     elif m_name == 'RN46':
79         print('\n This is ResNet-46'+com)
80         nb_blocks = [3,4,6,3,3]
81         wide = 2
82         block = 'plane'
83
84     else:
85         print('\n ???')
86
87 # --- Model Building -----
88 print('\n L2-Regularizer =' ,l_f)
89
90 input = Input(shape=input_shape)
91 X = input
92
93 # Initial convolution
94 X = Conv2D(filters=n_fil,
95            kernel_size=(3,3),
96            strides=(2,2),
97            padding="same",
98            kernel_regularizer=l2(reg_f)
99            )(X)
100
101 X = Activation("relu")(X)
102
103 X = MaxPooling2D(pool_size=(3,3),
104                 strides=(2,2),
105                 padding='same'
106                 )(X)
107
108 shortcut = X
109
110 # Iteration of resnet blocks
111 if block == 'plane':
112     for i, repete in enumerate(nb_blocks):
```

```

113     for j in range(repete):
114         if i>0 and j == 0:
115             shortcut = Conv2D(n_filter,
116                               kernel_size=(1, 1),
117                               strides=(2, 2),
118                               kernel_regularizer=l2(reg_f)
119                               )(shortcut)
120
121             X = Activation("relu")(X)
122
123             X = Conv2D(n_filter,
124                       kernel_size=(3,3),
125                       strides= (2,2),
126                       padding="same",
127                       kernel_regularizer=l2(reg_f)
128                       )(X)
129
130         else:
131             X = Activation("relu")(X)
132
133             X = Conv2D(n_filter,
134                       kernel_size=(3,3),
135                       padding="same",
136                       kernel_regularizer=l2(reg_f)
137                       )(X)
138
139             X = Activation("relu")(X)
140
141             X = Conv2D(n_filter,
142                       kernel_size=(3,3),
143                       padding="same",
144                       kernel_regularizer=l2(reg_f)
145                       )(X)
146
147             # ショートカットとマージ
148             X = Add()([X, shortcut])
149             shortcut = X
150
151         n_filter *= wide
152
153     if block == 'bottleneck':
154         shortcut = Conv2D(n_filter * 4,
155                           kernel_size=(1, 1) ,
156                           kernel_regularizer=l2(reg_f)
157                           )(shortcut)
158
159     for i, repete in enumerate(nb_blocks):
160         for j in range(repete):
161             if i>0 and j == 0:
162                 shortcut = Conv2D(n_filter * 4,
163                                   kernel_size=(1, 1),
164                                   strides=(2, 2),
165                                   kernel_regularizer=l2(reg_f)
166                                   )(shortcut)
167
168                 X = Activation("relu")(X)
169
170                 X = Conv2D(n_filter,
171                           kernel_size=(1,1),
172                           strides= (2,2),
173                           padding="same",

```

```

174                           kernel_regularizer=l2(reg_f)
175                           )(X)
176
177         else:
178             X = Activation("relu")(X)
179
180             X = Conv2D(n_filter,
181                       kernel_size=(1,1),
182                       padding="same",
183                       kernel_regularizer=l2(reg_f)
184                       )(X)
185
186             X = Activation("relu")(X)
187
188             X = Conv2D(n_filter,
189                       kernel_size=(3,3),
190                       padding="same",
191                       kernel_regularizer=l2(reg_f)
192                       )(X)
193
194             X = Activation("relu")(X)
195
196             X = Conv2D(n_filter * 4,
197                       kernel_size=(1,1),
198                       padding="same",
199                       kernel_regularizer=l2(reg_f)
200                       )(X)
201
202             # ショートカットとマージ
203             X = Add()([X, shortcut])
204             shortcut = X
205
206         n_filter *= wide
207
208         X = Activation("relu")(X)
209
210         # Branch to 2-Output
211
212         X1 = GlobalAveragePooling2D()(X)
213         y1 = Dense(2,name='org_out')(X1)
214
215         X2 = GlobalAveragePooling2D()(X)
216         y2 = Dense(2,name='axl_out')(X2)
217
218         model = Model(inputs=[input], outputs=[y1,y2])
219
220         return model
221
222     # -----
223     #                               Machine Learning Process
224     # -----
225
226     from sklearn.model_selection import train_test_split
227     import pickle, time
228     from datetime import datetime
229
230     # --- Opening -----
231     # get date
232     today = datetime.today().strftime('%y%m%d-%H%M')
233
234     # opening print

```

```

235 print('\n< RUN: Learning of Origin-position and Axis-length_',today,'>')
236
237 # get start time
238 s_time = time.time()
239
240 # Pass/Ng slice level
241 or_sl = 0.03
242 ax_sl = 0.03
243
244 # --- Data import -----
245 # data
246 # data_n = input(' Data Name --> \n ')
247 data_n = './data/xxx.pik'
248 print('\n Data: ',data_n)
249
250 with open(data_n, mode='rb') as f:
251     test_graf = pickle.load(f)
252
253 # X-normalize and black-white reversed
254 dim = 1024
255 X = test_graf['data']
256 if np.max(X[0,:]) == 255:
257     nor = 255 # case of gray data
258 else:
259     nor = 1 # case of monochro data
260 X = 1-X/nor
261
262 # Y-normalize
263 Y = test_graf['out']/dim
264
265 # input/output size
266 n_in = len(X[0])
267 n_out = len(Y[0])
268
269 # Data detail print
270 print('\n Data dimension    =',dim,
271       ', dpi =', test_graf['para'][1])
272 print(' Origin_range    =',{3.1f}.format(test_graf['para'][3]),
273       '-',{3.1f}.format(test_graf['para'][4]),
274       '\n X-axis_length_range =',{3.1f}.format(test_graf['para'][5]),
275       '-',{3.1f}.format(test_graf['para'][6]))
276
277 # --- Data split -----
278 x_train, x_test, y_train, y_test = train_test_split(X, Y,
279                                                    test_size = 0.1,
280                                                    train_size=0.9)
281
282 print('\n Train_n =',len(x_train),'Test_n =',len(x_test))
283
284 print('\n>> Learning Process Start...')
285
286 # --- Parameter input -----
287 m_n = 'RN34'
288
289 reg_f = 1e-6
290 n_epoch = 200
291 n_batch = 30
292 first_fil_n = 16
293 n_pat = 100
294 flag_ver = 0
295

```

```

296 # --- Model Built -----
297 input_ = x_train.shape[1:]
298 model = resnet(input_,
299               n_out,
300               m_n,
301               first_fil_n,
302               reg_f
303               )
304
305 # Optimizer setting
306 optimizer = 'adam'
307 print('\n Optimizer : adam')
308
309 '''Optimizer
310 # Optimizer adam
311 optimizer = 'adam'
312 print('\n Optimizer : adam')
313
314 # Optimizer SGD + Momentum
315 from keras.optimizers import SGD
316 optimizer = SGD(decay=1e-6, momentum=0.9, nesterov=True)
317 print('\n Optimizer : SGD + Momentum')
318 '''
319
320 # --- Model compile -----
321 model.compile(loss = 'mean_squared_error',
322              optimizer = optimizer,
323              metrics = ['accuracy']
324              )
325
326 # iteration parameter print
327 print('\n first_filter = ',first_fil_n)
328 print(' batch      = ',n_batch)
329 print(' max epoch   = ',n_epoch)
330 print(' patience    = ',n_pat)
331
332 # --- Model fit -----
333 early_stopping = EarlyStopping(monitor='val_loss', patience=n_pat)
334
335 hist = model.fit(x_train, [y_train[:,0:2],y_train[:,2:4]],
336                 epochs = n_epoch,
337                 batch_size = n_batch,
338                 validation_data = (x_test, [y_test[:,0:2],y_test[:,2:4]]),
339                 verbose = flag_ver,
340                 callbacks = [early_stopping]
341                 )
342
343 # --- Predict and evaluation -----
344 predict_ = model.predict(x_test, batch_size=n_batch)
345 predict = np.concatenate([predict_[0], predict_[1]],1)
346
347 # difference between predict and test
348 dif = (y_test - predict)
349
350 # error count
351 error_or = dif/(y_test)
352 error_ax = error_or
353
354 org_pass_list = []; org_ng_list=[]
355 axl_pass_list = []; axl_ng_list=[]
356 for i in range(0,len(y_test)):

```

```

357
358 # origin error
359 if -1*or_sl < error_or[i,0] < or_sl and -1*or_sl < error_or[i,1] < or_sl:
360     org_pass_list.append(i)
361 else:
362     org_ng_list.append(i)
363
364 # axis error
365 if -1*ax_sl < error_ax[i,2] < ax_sl and -1*ax_sl < error_ax[i,3] < ax_sl:
366     axl_pass_list.append(i)
367 else:
368     axl_ng_list.append(i)
369
370 org_nonzero = len(org_ng_list)
371 org_accuracy = 1 - org_nonzero/(len(dif))
372 axl_nonzero = len(axl_ng_list)
373 axl_accuracy = 1 - axl_nonzero/(len(dif))
374
375 # Execution time
376 e_time = time.time() # end time
377 n_ep = len(hist.history['loss'])
378 print('\n ...Finished. Exe time (s) = ',
379       '{:6.2f}'.format(e_time - s_time),
380       ', n_ep = ', n_ep)
381
382 # print("\n* model_summary"); model.summary()
383
384 # --- Print out of results -----
385 # loss and accuracy print
386 print('\n < loss and accuracy > \n loss = ',
387       '{:4.2e}'.format(hist.history['loss'][n_ep-1]),
388       ', org_accuracy = ', '{:4.3f}'.format(hist.history['org_out_accuracy'][n_ep-1]),
389       ', axl_accuracy = ', '{:4.3f}'.format(hist.history['axl_out_accuracy'][n_ep-1]),
390       )
391
392 # loss and accuracy history
393 fig = plt.figure(figsize=(7, 7))
394 plt.subplot(2, 2, 1)
395 x_ep = range(len(hist.history['loss']))
396 plt.plot(x_ep, hist.history['loss'], 'b')
397 plt.plot(x_ep, hist.history['org_out_loss'], 'b:')
398 plt.plot(x_ep, hist.history['axl_out_loss'], 'b--')
399 plt.plot(x_ep, hist.history['val_loss'], 'r')
400 plt.plot(x_ep, hist.history['val_org_out_loss'], 'r:')
401 plt.plot(x_ep, hist.history['val_axl_out_loss'], 'r--')
402 plt.yscale('log')
403 plt.title("loss and epoch", fontsize=15)
404
405 plt.subplot(2, 2, 2)
406 plt.plot(x_ep, hist.history['org_out_accuracy'], 'b:')
407 plt.plot(x_ep, hist.history['axl_out_accuracy'], 'b--')
408 plt.plot(x_ep, hist.history['val_org_out_accuracy'], 'r:')
409 plt.plot(x_ep, hist.history['val_axl_out_accuracy'], 'r--')
410 plt.title("accuracy and epoch", fontsize=15)
411 plt.ylim(0, 1.1)
412 plt.show()
413
414 print("\n < Pass_Rate > (Sl; org = +/-, '{:3.3f}'.format(or_sl),
415       ', axl = +/-, '{:3.3f}'.format(ax_sl), ')')
416 print('Org_NG = ', org_nonzero, '/', len(dif),
417       ', Pass_Rate = ', '{:4.3f}'.format(org_accuracy))

```

```

418 print('Axs_NG = ', axl_nonzero, '/', len(dif),
419       ', Pass_Rate = ', '{:4.3f}'.format(axl_accuracy))
420
421 # result evaluation
422 # _error rate
423 fig = plt.figure(figsize=(7, 14))
424
425 # _origin error rate
426 plt.subplot(4, 2, 3)
427 plt.plot(error_or[:,0], error_or[:,1], 'x', color='red') # x-y origin
428 plt.title("Origin error rate", fontsize=15)
429 plt.ylim(-1, 1); plt.xlim(-1, 1)
430
431 # _axis error rate
432 plt.subplot(4, 2, 4)
433 plt.plot(error_ax[:,2], error_ax[:,3], 'x', color='red') # x-y axis length
434 plt.title("Axis_length error rate", fontsize=15)
435 plt.ylim(-1, 1); plt.xlim(-1, 1)
436
437 # _Print error position
438 # _origin error position
439 plt.subplot(4, 2, 1)
440 for j in org_ng_list:
441     # print('No. ', j, y_test[j,:], dif[j,:])
442     plt.plot(y_test[j,0], y_test[j,1], 'x', color='red')
443 for k in org_pass_list:
444     plt.plot(y_test[k,0], y_test[k,1], 'o', color='b')
445 plt.title("Origin_error_position", fontsize=15)
446 plt.ylim(0, 1); plt.xlim(0, 1)
447
448 # _axis error position
449 plt.subplot(4, 2, 2)
450 for j in axl_ng_list:
451     # print('No. ', j, y_test[j,:], dif[j,:])
452     plt.plot(y_test[j,2], y_test[j,3], 'x', color='red')
453 for k in axl_pass_list:
454     plt.plot(y_test[k,2], y_test[k,3], 'o', color='b')
455 plt.title("Axis_length_error_position", fontsize=15)
456 plt.ylim(0, 1); plt.xlim(0, 1)
457 plt.show()
458
459 # _NG sample drawing
460 print("\n < NG figure sample > ")
461 fig = plt.figure(figsize=(20, 20))
462 n = 10
463 n_list = np.random.choice(org_ng_list, n)
464 print(' - Org NG - ')
465 for i in range(n):
466     nn = n_list[i]
467     print(nn, ' Error', error_or[nn][0:2])
468     ax = plt.subplot(4, n, i+1)
469     plt.imshow(np.reshape(1-x_test[nn,:], (dim, -1)))
470     plt.plot((predict[nn,0])*dim, (1-predict[nn,1])*dim,
471            'rx', ms=20)
472     plt.plot((y_test[nn,0])*dim, (1-y_test[nn,1])*dim,
473            'bx', ms=20)
474     plt.gray()
475     ax.get_xaxis().set_visible(False)
476     ax.get_yaxis().set_visible(False)
477     plt.ylim(dim, dim*2/3); plt.xlim(0, dim/3)
478 plt.show()

```

```

479 fig=plt.figure(figsize=(20,20))
480 n = 8
481 n_list=scale = np.random.choice(axl_ng_list, n)
482 for i in range(n):
483     nn = n_list[i]
484     print(nn,' Error',error_or[nn][2:4])
485     ax = plt.subplot(4, n, i+1)
486     plt.imshow(np.reshape(1-x_test[nn,:],(dim,- 1)))
487     plt.plot([y_test[nn,0]*dim, (y_test[nn,0]+predict[nn,2])*dim],
488             [(1-y_test[nn,1])*dim, (1-y_test[nn,1])*dim],
489             'r+:',ms=15) # X-axis
490     plt.plot([y_test[nn,0]*dim, y_test[nn,0]*dim],
491             [(1-y_test[nn,1])*dim, (1-y_test[nn,1]-predict[nn,3])*dim],
492             'r+:',ms=15) # Y-axis
493     plt.gray()
494     ax.get_xaxis().set_visible(False)
495     ax.get_yaxis().set_visible(False)
496     plt.ylim(dim+10,0) ; plt.xlim(0,dim+10)
497 plt.show()
498
499 # --- Model Print and Save -----
500 print('< Model Save and Print >')
501 mp_flag = 'n' ; ms_flag = 'n'
502
503 m_para = str(dim)+'_'+m_n+'_bt'+str(n_batch)+'ft'+str(first_fil_n)+'\
504         'ep'+str(n_ep)
505 m_name = 'OrgAxl_'+m_para+'_'+str(today)
506
507 # model print
508 mp_flag = input(' Print the model shape? y or n --> ')
509 if mp_flag == 'y':
510     from keras.utils import plot_model
511     plot_model(model, to_file = './model/'+m_name+'.png',
512               show_shapes=True)
513     print(' Model Shape is printed --> ./model/'+m_name+'.png')
514 else:
515     print(' ! Model is NOT printed. ')
516
517 # model save
518 ms_flag = input('\n Save the learned model? y or n --> ')
519 if ms_flag == 'y':
520     model.save('./model/'+m_name+'.h5')
521     print(' Model is saved --> ./model/'+m_name+'.h5')
522 else:
523     print(' ! Model is NOT saved. ')
524
525 # _____ END OF CODE _____
526

```

END