

画像整形の機械学習プログラム（Learning_Shaping.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
 - 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_Shaping.py)は機械学習を行い、グラフ画像から数値化に不要な部分を削除（整形）す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

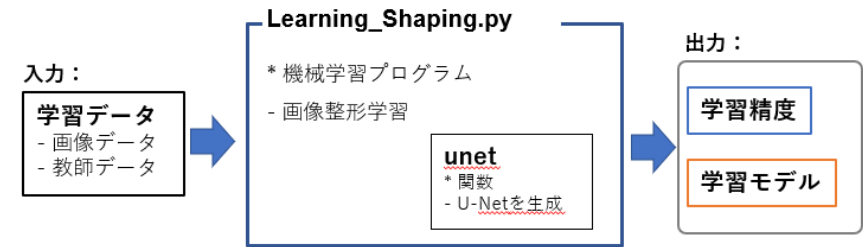
- Python 3.6
- Tensorflow-gpu 1.4.0
- Keras 2.2.4
- sklearn 0.19.1, 0.21.2
- matplotlib 2.2.2

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）を入力とし、内部の関数で生成したU-Net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. 9-254 行: 学習モデルを生成する関数
 - 10-21 行: モジュールのインポート
 - 23-254 行: U-Netの生成
3. 256-535 行: 機械学習の実行
 - 260-263 行: モジュールのインポート
 - 265-275 行: 開始準備
 - 277-308 行: データの読み込みと振り分け
 - 310-319 行: パラメータの設定
 - 321-330 行: モデル構築
 - 332-370 行: モデルコンパイル
 - 372-390 行: モデルフィット
 - 392-508 行: 結果の評価と出力
 - 510-534 行: モデルの保存
 - 536 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279行目に学習データの名前を記述して（例:WaveShaping_Data_1024_5000_191217-0229.pik）保存する次に、<2. 動作環境>を満たした動作環境下でPytnon上で実行する

```
277 # --- Data Reading -----
278 # data_n = input("\n Data Name --> ")
279 data_n = 'WaveShaping_Data_1024_5000_191217-0229.pik'
280 print("\n < Data >\n",data_n)
281 with open(data_n,mode='rb') as f:
282     test_graf=pickle.load(f)
283 ~~~
```

学習用データの要件

- 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 >
- 2つのキー'in','out'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

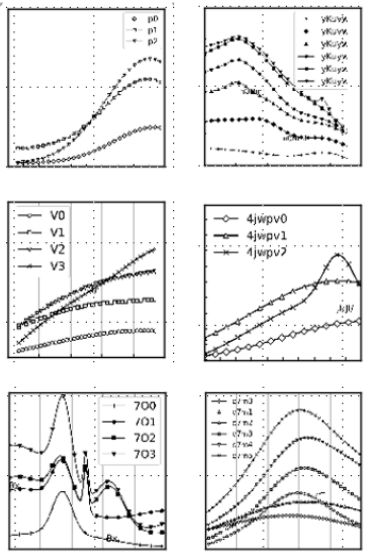
画像データ:

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)

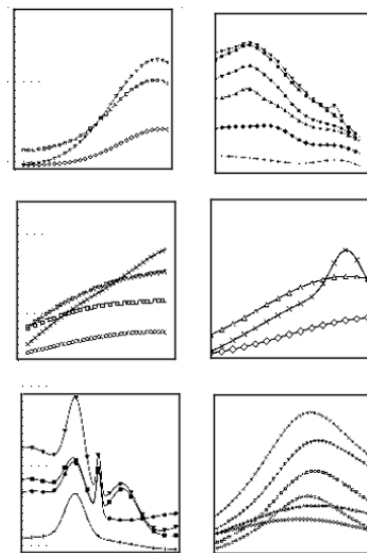
教師データ:

画像データと形式、大きさ、共に同じ。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)

画像データ:



教師データ:



3.3.2 出力

学習済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

== Start: MACHINE LEARNING FOR WAVEFORM SHAPING_ 191219-0803 ==

< Data >

./data/WaveShaping_Data_1024_5000_191217-0229.pik

Learning Process Start ...

This is U-NET 1x2x4x8x16x8x4x2x1

l2_reg = 1e-05

Learn_n = 4500, Test_n = 500

max epoch = 300

filter = 32

batch size = 20

e-stop = 30

Optimizer : adam

loss : mean_squared_error

2-GPU RUNNING

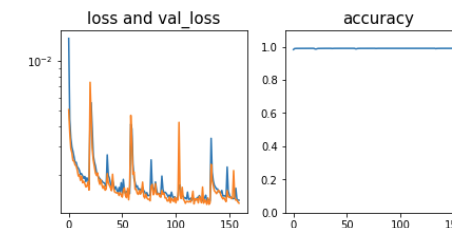
(2) 精度と正答率

lossとaccの履歴と正答率の表示

... Finished. Exe_time (s) = 25804.90, n_ep = 160

< loss and accuracy >

loss = 1.41e-03, acc = 0.990



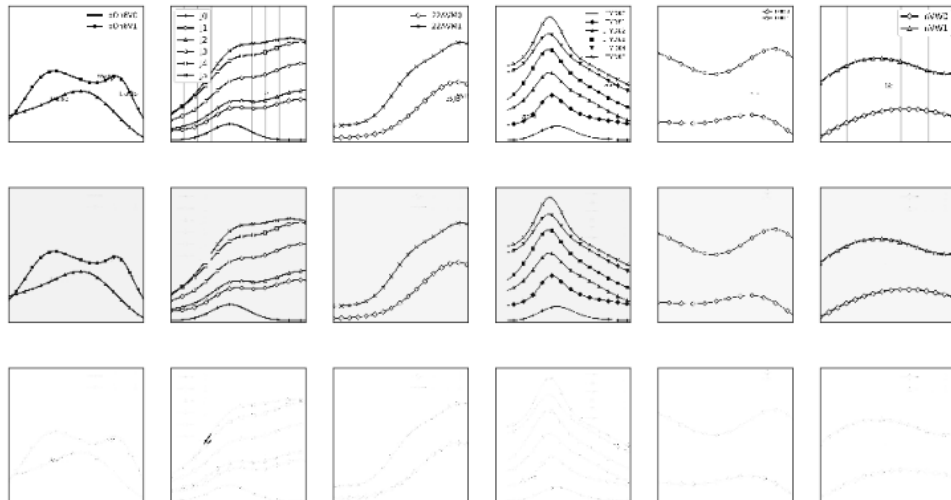
< Correct predict rate > * slice-level < 0.1

Correct rate = 0.7660, (383 / 500)

(3) Pass and NG

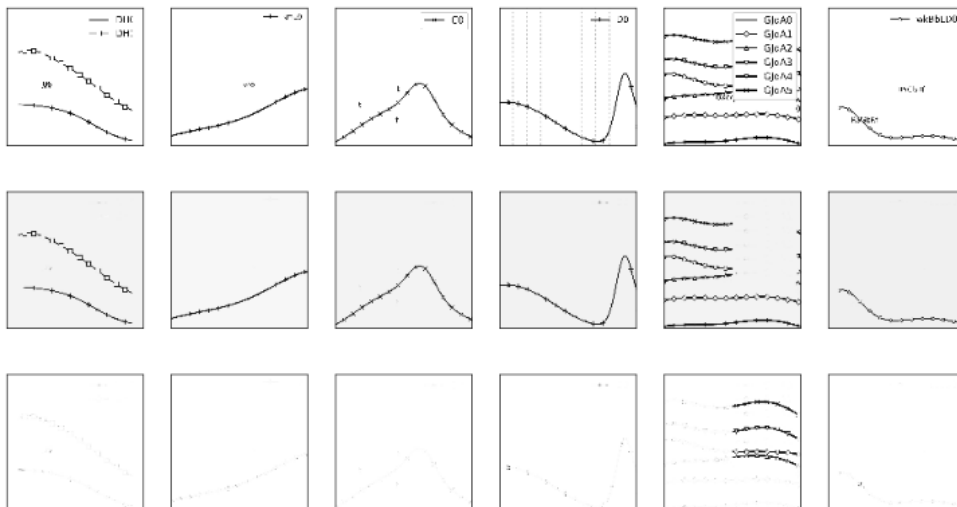
1) うまく整形できたものの中からランダムに6個を取り出して表示

```
< Pass Figure >
1st = input, 2nd = predict, 3th = target - predict
Fig 186, error 0.072910346
Fig 308, error 0.07484072
Fig 181, error 0.09141519
Fig 337, error 0.056499112
Fig 64, error 0.096929334
Fig 419, error 0.0762513
```



2) うまく整形できなかったものの中でランダムに6個を取り出して表示

```
< NG Figure >
1st = input, 2nd = predict, 3th = target - predict
Fig 443, error 0.12829527
Fig 254, error 0.111452356
Fig 282, error 0.11420366
Fig 266, error 0.12873219
Fig 287, error 0.36278743
Fig 46, error 0.1537545
```



(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 output >
Print the model shape? y or n --> y
Printed --> Shaping_1024_5000_Unt_bt20ft32ep160_191219-0803.png
Save the model? y or n --> y
Saved --> Shaping_1024_5000_Unt_bt20ft32ep160_191219-0803.h5
```

4. パラメータ

このプログラムのパラメータは1024x1024の大きさの画像整形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
reg_f	L2-Regularizer	1e-5	0-1
n_epoch	最大エポック数	300	1-
n_batch	バッチ数	20	1-
first_fil_n	開始フィルタ数	32	8, 16, 24, 32は検証済
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

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

- 1-8行: 開始コメント
- 9-254 行: **学習モデルを生成する関数**
 - 10-21 行: モジュールのインポート
 - 23-254 行: U-Netの生成
- 256-535 行: **機械学習の実行**
 - 260-263 行: モジュールのインポート
 - 265-275 行: 開始準備
 - 277-308 行: データの読み込みと振り分け
 - 310-319 行: パラメータの設定

321-330 行: モデル構築
332-370 行: モデルコンパイル
372-390 行: モデルフィット
392-508 行: 結果の評価と出力
510-534 行: モデルの保存
536 行: END

全リスト

In []:

```
1  # -*- coding: utf-8 -*-
2  ...
3  Created on Nov.04,2019
4
5  < 画像から数値化に不要な部分を取り除く 整形を行うための機械学習 >
6
7  @author: T.KONO
8  ...
9  # ----- Model Build Function -----
10 import numpy as np
11 from keras.layers import (Input,
12                             Conv2D,
13                             MaxPooling2D,
14                             Conv2DTranspose,
15                             concatenate,
16                             UpSampling2D
17                             )
18
19 from keras.models import Model
20 from keras.callbacks import EarlyStopping
21 from keras.regularizers import l2
22
23 def unet(input_shap,nb_filt,reg_f):
24     ...
25     <U-NET モデルを作成 >
26
27     input_shap : 入力の形
28     nb_filt : 最初のフィルタ数 各層毎にx1x2x4x8x16x8x4x2x1
29     reg_f : L2-regularizer の大きさ
30     ...
31
32     n_conv = 3
33     k_size = 3
34     n_strid = 2
35
36     print('\n This is U-NET 1x2x4x8x16x8x4x2x1')
37     print(' l2_reg = ',reg_f)
38
39     input = Input(shape = input_shap)
40     X = input
41
42     # Pre Conv
43     X = Conv2D(nb_filt,
44                 kernel_size=(21,21),
45                 strides=(2,2),
46                 activation='relu',
47                 padding='same',
48                 kernel_regularizer=l2(reg_f)
49                 )(X)
50
51     X = MaxPooling2D(pool_size=(3,3),
52                      strides=(2,2),
53                      padding='same'
54                      )(X)
55
56     # U-Net
57     conv1 = Conv2D(nb_filt,
58                    (n_conv,n_conv),
59                    activation='relu',
```

```

60         padding='same',
61         kernel_regularizer=l2(reg_f)
62     )(X)
63
64     conv1 = Conv2D(nb_filt,
65                   (n_conv, n_conv),
66                   activation='relu',
67                   padding='same',
68                   kernel_regularizer=l2(reg_f)
69                   )(conv1)
70
71     pool1 = MaxPooling2D(pool_size=(3,3),
72                          strides=(2,2),
73                          padding='same'
74                          )(conv1)
75
76     conv2 = Conv2D(nb_filt*2,
77                   (n_conv,n_conv),
78                   activation='relu',
79                   padding='same',
80                   kernel_regularizer=l2(reg_f)
81                   )(pool1)
82
83     conv2 = Conv2D(nb_filt*2,
84                   (n_conv,n_conv),
85                   activation='relu',
86                   padding='same',
87                   kernel_regularizer=l2(reg_f)
88                   )(conv2)
89
90     pool2 = MaxPooling2D(pool_size=(3,3),
91                          strides=(2,2),
92                          padding='same'
93                          )(conv2)
94
95     conv3 = Conv2D(nb_filt*4,
96                   (n_conv,n_conv),
97                   activation='relu',
98                   padding='same',
99                   kernel_regularizer=l2(reg_f)
100                  )(pool2)
101
102     conv3 = Conv2D(nb_filt*4,
103                   (n_conv,n_conv),
104                   activation='relu',
105                   padding='same',
106                   kernel_regularizer=l2(reg_f)
107                  )(conv3)
108
109     pool3 = MaxPooling2D(pool_size=(3,3),
110                          strides=(2,2),
111                          padding='same'
112                          )(conv3)
113
114     conv4 = Conv2D(nb_filt*8,
115                   (n_conv,n_conv),
116                   activation='relu',
117                   padding='same',
118                   kernel_regularizer=l2(reg_f)
119                  )(pool3)
120

```

```

121     conv4 = Conv2D(nb_filt*8,
122                   (n_conv, n_conv),
123                   activation='relu',
124                   padding='same',
125                   kernel_regularizer=l2(reg_f)
126                   )(conv4)
127
128     pool4 = MaxPooling2D(pool_size=(3,3),
129                          strides=(2,2),
130                          padding='same'
131                          )(conv4)
132
133     conv5 = Conv2D(nb_filt*16,
134                   (n_conv,n_conv),
135                   activation='relu',
136                   padding='same',
137                   kernel_regularizer=l2(reg_f)
138                   )(pool4)
139
140     conv5 = Conv2D(nb_filt*16,
141                   (n_conv,n_conv),
142                   activation='relu',
143                   padding='same',
144                   kernel_regularizer=l2(reg_f)
145                   )(conv5)
146
147     up6 = concatenate([Conv2DTranspose(nb_filt*8,
148                                       (k_size, k_size),
149                                       strides=(n_strid,n_strid),
150                                       padding='same',
151                                       kernel_regularizer=l2(reg_f)
152                                       )(conv5),conv4
153                       ],axis=3)
154
155     conv6 = Conv2D(nb_filt*8,
156                   (n_conv,n_conv),
157                   activation='relu',
158                   padding='same',
159                   kernel_regularizer=l2(reg_f)
160                   )(up6)
161
162     conv6 = Conv2D(nb_filt*8,
163                   (n_conv,n_conv),
164                   activation='relu',
165                   padding='same',
166                   kernel_regularizer=l2(reg_f)
167                   )(conv6)
168
169     up7 = concatenate([Conv2DTranspose(nb_filt*4,
170                                       (k_size, k_size),
171                                       strides=(n_strid, n_strid),
172                                       padding='same',
173                                       kernel_regularizer=l2(reg_f)
174                                       )(conv6),conv3
175                       ], axis=3)
176
177     conv7 = Conv2D(nb_filt*4,
178                   (n_conv,n_conv),
179                   activation='relu',
180                   padding='same',
181                   kernel_regularizer=l2(reg_f)

```

```

182         )(up7)
183
184     conv7 = Conv2D(nb_filt*4,
185                   (n_conv,n_conv),
186                   activation='relu',
187                   padding='same',
188                   kernel_regularizer=l2(reg_f)
189                   )(conv7)
190
191     up8 = concatenate([Conv2DTranspose(nb_filt*2,
192                                     (k_size, k_size),
193                                     strides=(n_strid, n_strid),
194                                     padding='same',
195                                     kernel_regularizer=l2(reg_f))
196                       (conv7),conv2
197                       ],axis=3)
198
199     conv8 = Conv2D(nb_filt*2,
200                   (n_conv,n_conv),
201                   activation='relu',
202                   padding='same',
203                   kernel_regularizer=l2(reg_f)
204                   )(up8)
205
206     conv8 = Conv2D(nb_filt*2,
207                   (n_conv,n_conv),
208                   activation='relu',
209                   padding='same',
210                   kernel_regularizer=l2(reg_f)
211                   )(conv8)
212
213     X = concatenate([Conv2DTranspose(nb_filt,
214                                     (k_size, k_size),
215                                     strides=(n_strid, n_strid),
216                                     padding='same',
217                                     kernel_regularizer=l2(reg_f))
218                   (conv8),conv1
219                   ],axis=3)
220
221     # Post Conv
222     X = Conv2D(nb_filt,
223               (n_conv,n_conv),
224               activation='relu',
225               padding='same',
226               kernel_regularizer=l2(reg_f)
227               )(X)
228
229     X = UpSampling2D(size=(2,2))(X)
230
231     X = Conv2D(nb_filt,(n_conv,n_conv),
232               activation='relu',
233               padding='same',
234               kernel_regularizer=l2(reg_f)
235               )(X)
236
237     X = UpSampling2D(size=(2,2))(X)
238
239     X = Conv2D(nb_filt,
240               (n_conv,n_conv),
241               activation='relu',
242               padding='same',

```

```

243         kernel_regularizer=l2(reg_f)
244         )(X)
245
246     Y = Conv2D(1,
247               (n_conv,n_conv),
248               padding='same',
249               kernel_regularizer=l2(reg_f)
250               )(X)
251
252     model = Model(inputs=[input], outputs=[Y])
253
254     return model
255
256     #-----
257     #           MACHINE LEARNING PROCESS
258     #-----
259
260     import matplotlib.pyplot as plt
261     from sklearn.model_selection import train_test_split
262     import pickle, time
263     from datetime import datetime
264
265     # --- Opening -----
266     now = datetime.today().strftime('%y%m%d-%H%M')
267     s_time = time.time()
268
269     # gpu selection
270     gpu_n = 2
271
272     # start print
273     print('\n== Start: MACHINE LEARNING FOR WAVEFORM SHAPING_',
274           now,'==')
275     s_time = time.time()
276
277     # --- Data Reading -----
278     # data_n = input('\n Data Name --> ')
279     data_n = 'WaveShaping_Data_1024_5000_191217-0229.pik'
280     print('\n < Data >\n ',data_n)
281     with open(data_n, mode='rb') as f:
282         test_graf=pickle.load(f)
283
284     # data import
285     dim = 1024 # test_graf['para'][0]
286     d_nu = len(test_graf['out'])
287
288     if np.max(test_graf['data'][0,:]) == 255:
289         nor = 255 # case of gray data
290     else:
291         nor = 1 # case of monochro data
292
293     X = (1-test_graf['data']/nor).astype(np.float16) # black-white reverse
294     Y = (1-test_graf['out']/nor).astype(np.float16)
295
296     del test_graf
297
298     # data split
299     x_train, x_test, y_train, y_test = train_test_split(
300         X, Y,
301         test_size = 0.1,
302         train_size = 0.9,
303         random_state = 0

```

```

304     )
305
306 del X; del Y
307
308 print("\nLearning Process Start ...")
309
310 # --- Model Built -----
311 # itaration parameter
312 n_epoch = 300
313 n_batch = 20
314 n_stop = 30
315 n_filter = 32
316
317 leg_f = 1e-5
318
319 flag_ver = 0
320
321 # model built
322 input_ = x_train.shape[1:]
323 model = unet(input_, n_filter, leg_f)
324
325 # parameter print
326 print("\n Learn_n = ',len(x_train)',Test_n =',len(x_test))
327 print(' max epoch = ',n_epoch)
328 print(' filter = ',n_filter)
329 print(' batch size = ',n_batch)
330 print(' e-stop = ',n_stop)
331
332 # ---- Model Compile -----
333 ## Optimizer adam
334 optimizer = 'adam'
335 print("\n OpimIZER : adam')
336
337 ...
338 ## Optimizer adam
339 optimizer = 'adam'
340 print("\n OpimIZER : adam')
341
342 ## Optimizer SGD + Momentum
343 from keras.optimizers import SGD
344 optimizer = SGD(decay=1e-6, momentum=0.9, nesterov=True)
345 print("\n OpimIZER : SGD + Momentum')
346
347 ...
348 # loss setting
349 loss = 'mean_squared_error'
350 print(' loss :',loss)
351
352 # Compile
353 if not gpu_n == 1:
354     # Multi GPU
355     print("\n',str(gpu_n)+'-GPU RUNNING '")
356     from keras.utils import multi_gpu_model
357     parallel_model = multi_gpu_model(model,gpus=gpu_n)
358     parallel_model.compile(loss = loss,
359                           optimizer = optimizer,
360                           metrics = ['accuracy']
361                           )
362 else:
363     # 1-GPU
364     print("\n 1-GPU RUNNING '")

```

```

365     model.compile(loss = loss,
366                   optimizer = optimizer,
367                   metrics = ['accuracy']
368                   )
369
370 early_stopping = EarlyStopping(monitor = 'loss', patience = n_stop)
371
372 # ---- Model Fit -----
373 if not gpu_n == 1:
374     # Multi GPU
375     hist = parallel_model.fit(x_train, y_train,
376                              epochs = n_epoch,
377                              batch_size = n_batch,
378                              validation_data = (x_test, y_test),
379                              verbose = flag_ver,
380                              callbacks = [early_stopping]
381                              )
382 else:
383     # 1-GPU
384     hist = model.fit(x_train, y_train,
385                     epochs = n_epoch,
386                     batch_size = n_batch,
387                     validation_data = (x_test, y_test),
388                     verbose = flag_ver,
389                     callbacks = [early_stopping]
390                     )
391
392 # ---- Predict and evaluation -----
393 # predict
394 predict_ = (model.predict(x_test,batch_size=n_batch))
395 # error
396 dif = np.absolute(y_test-predict_)
397
398 diff = np.reshape(dif,(-1,dim*dim))
399 y_t = np.reshape(y_test,(-1,dim*dim))
400
401 '''y_test と Predict の差分の総和 を\
402 y_test の総和で割った値を誤差の評価値とする
403 *float16のままだと範囲を超えてしまうので32に'''
404 err = np.sum(diff,axis=1)/(np.sum(y_t.astype(np.float32),axis=1))
405
406 sl_level =0.1
407
408 pass_list = []
409 ng_list = []
410 for i in range(0,len(y_test)):
411     if err[i] < sl_level :
412         pass_list.append(i)
413     else:
414         ng_list.append(i)
415
416 # exe.time
417 n_ep = len(hist.history['loss'])
418 e_time = time.time()
419 print("\n... Finished. Exe_time (s) = ',
420       '{:6.2f}'.format(e_time - s_time),
421       ', n_ep = ',n_ep )
422
423 # --- Results out-----
424 # loss and accuracy
425 print("\n < loss and accuracy >")

```



```

426 n_ep = len(hist.history['loss'])
427 print(' loss = ', '{:4.2e}'.format(hist.history['loss'][n_ep-1]),
428       ', acc = ', '{:4.3f}'.format(hist.history['acc'][n_ep-1]))
429 fig = plt.figure(figsize=(7,7))
430 plt.subplot(2,2,1)
431 x_ep = range(len(hist.history['loss']))
432 plt.plot(x_ep, hist.history['loss'])
433 plt.plot(x_ep, hist.history['val_loss'])
434 plt.yscale('log')
435 plt.title("loss and val_loss", fontsize=15)
436 # plt.ylim(0.0001,1)
437 plt.subplot(2,2,2)
438 x_ep = range(len(hist.history['acc']))
439 plt.plot(x_ep, hist.history['acc'])
440 # plt.yscale('log')
441 plt.title("accuracy", fontsize=15)
442 plt.ylim(0,1.1)
443 plt.show()
444
445 # correc rate
446 print("\n < Correct predict rate > ")
447 correct_rate = len(pass_list)/len(y_test)
448 print(' cpr = ', '{:6.4f}'.format(correct_rate),
449       '(', len(pass_list), '/', len(y_test), '), sl = ', sl_level)
450
451 # comparison of figure
452 print("\n < Pass Figure > ")
453 fig=plt.figure(figsize=(16,12))
454 n = 6
455 n_list=np.random.choice(pass_list, n)
456 # print(' figure number = ',n_list)
457 print(' 1st = input, 2nd = predict, 3th = target - predict')
458 for i in range(n):
459     nn = n_list[i]
460     print(' Fig ',nn,', error ',err[nn])
461     ax = plt.subplot(4,n,i+1)
462     plt.imshow(np.reshape(1-x_test[nn,:].astype(np.float32),
463                          (dim,-1)),cmap='gray')
464     ax.get_xaxis().set_visible(False)
465     ax.get_yaxis().set_visible(False)
466
467     ax = plt.subplot(4,n,i+1+n)
468     plt.imshow(np.reshape(1-predict_[nn,:].astype(np.float32),
469                          (dim,-1)),cmap='gray')
470     ax.get_xaxis().set_visible(False)
471     ax.get_yaxis().set_visible(False)
472
473     ax = plt.subplot(4, n, i+1+n+n)
474     plt.imshow(np.reshape(1-dif[nn,:].astype(np.float32),
475                          (dim,-1)),cmap='gray')
476     ax.get_xaxis().set_visible(False)
477     ax.get_yaxis().set_visible(False)
478
479 plt.show()
480
481 print("\n < NG Figure > ")
482 fig=plt.figure(figsize=(16,12))
483 n = 6
484 n_list=np.random.choice(ng_list, n)
485 # print(' figure number = ',n_list)
486 print(' 1st = input, 2nd = predict, 3th = target - predict')

```

```

487 for i in range(n):
488     nn = n_list[i]
489     print(' Fig ',nn,', error ',err[nn])
490     ax = plt.subplot(4,n,i+1)
491     plt.imshow(np.reshape(1-x_test[nn,:].astype(np.float32),
492                          (dim,-1)),cmap='gray')
493     ax.get_xaxis().set_visible(False)
494     ax.get_yaxis().set_visible(False)
495
496     ax = plt.subplot(4,n,i+1+n)
497     plt.imshow(np.reshape(1-predict_[nn,:].astype(np.float32),
498                          (dim,-1)),cmap='gray')
499     ax.get_xaxis().set_visible(False)
500     ax.get_yaxis().set_visible(False)
501
502     ax = plt.subplot(4, n, i+1+n+n)
503     plt.imshow(np.reshape(1-dif[nn,:].astype(np.float32),
504                          (dim,-1)),cmap='gray')
505     ax.get_xaxis().set_visible(False)
506     ax.get_yaxis().set_visible(False)
507
508 plt.show()
509
510 # --- Model Print and Save -----
511 print("\n < model output > ")
512 mp_flag = 'n'; ms_flag = 'n'
513
514 m_para = 'Unt_bt'+str(n_batch)+'ft'+str(n_filter)+'ep'+str(n_ep)
515 model_n = 'Shaping_'+str(dim)+'_'+str(d_nu)+'_'+m_para+'_'+str(now)
516
517 # model shape print
518 mp_flag = input(' Print the model shape? y or n --> ')
519 if mp_flag == 'y':
520     from keras.utils import plot_model
521     plot_model(model,
522               to_file = model_n+'.png',
523               show_shapes=True)
524     print(' --> '+model_n+'.png')
525 else:
526     print(' ... Not Printed ')
527
528 # model save
529 ms_flag = input(' Save the model? y or n --> ')
530 if ms_flag == 'y':
531     model.save(model_n+'.h5')
532     print(' --> '+model_n+'.h5')
533 else:
534     print(' ... Not Saved ')
535
536 # _____ END of Code _____

```

END

