

画像整形の機械学習用データ作成プログラム (Data_Shaping.py) 使用説明書

目次

1. 機能
2. 動作環境
3. 内容
 - 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 入力及びプログラムの実行
 - 3.3.2 出力
 - 3.3.3 データの出力形式
4. プログラムリスト

1. 機能

このプログラム(Data_Shaping.py)は、機械学習用の学習データを作成するものであり、作成されたデータは、グラフの整形処理を行う機械学習(Learning_Shaping.py)に用いられる

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

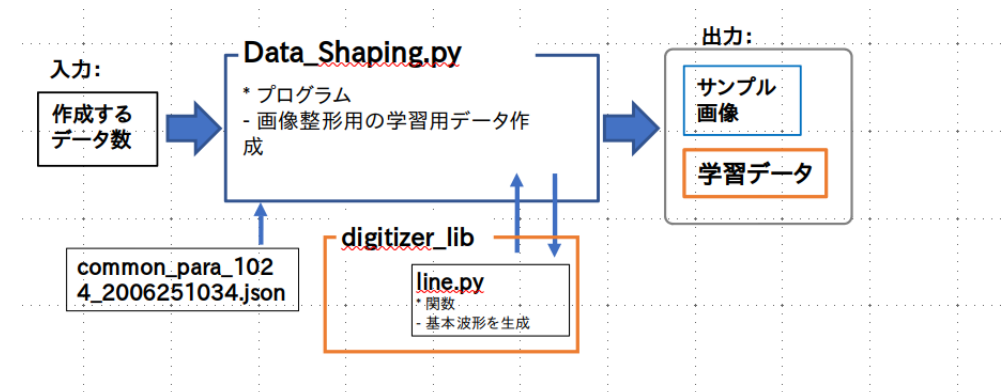
2) このプログラムを実行するには、以下のモジュールがあらかじめ準備された環境が必要である。また、記載されている以外のVersionでの動作確認は行っていないので注意が必要である

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

3. 内容

3.1 構造

このプログラムは、入力として、作成するデータ数を使用者からのインプットとして受け取り、出力として学習用データとデータのサンプル画像を自動的に出力する
このプログラムは、画像データを生成するため、画像パラメータファイル(common_para_1024_2006251034.json) から画像パラメータを読み込む
このプログラムは基本波形を生成するため、関数ライブラリdigitizer_lib に収納されている波形生成関数 line.py を呼び出して使用する



3.2 プログラムリストの概要

このプログラムはデータ作成本体 (Data_Shaping.py) と関数ライブラリ (digitizer_lib) に収められている波形生成関数 (Line.py) の2つのモジュールに分けられる
全リストは、4. プログラムリストに示す

1. データ作成本体 (Data_WavsShaping.py)

1-8 行: 開始コメント
10-20 行: モジュールのインポート
28-34 行: 開始準備
36-79 行: データパラメータの設定
81-99 行: パラメータの確認
101-313 行: データの生成
314-349 行: チェック用の画像出力
351-363 行: モデルの保存
365 行: END

2. 波形生成関数 (Line.py)

1-7 行: 開始コメント
9-9 行: モジュールのインポート
15-38 行: 開始準備
40-69 行: 波形の生成
71 行: return

3.3 入出力

以下、Jupyter Notebook上で実行された例を示して説明する

3.3.1 入力およびプログラムの実行

このプログラムの入力は、作成するデータ数のみである
プログラムを実行すると、下の図の様にコンソール上でデータ数の入力待ち==>になるので、使用者は必要なデータ数を入力して実行する
注) データ数が100個以下が入力された場合は、味見としてデータの保存は実行されず、サンプル画像の表示のみが行われる

< Data Creating of Machine Learning for WaveShaping. 200627-1138 >

Data Dimension: 1024 x 1024

Input data number. if <= 99, datas are not saved

==>

次に、下の様な画像パラメータが表示され、確認の入力待ち==>になる

このパラメータを確認してOKであれば、'y'もしくは'n'以外を入力して実行するとデータ作成が始まる

< Data Creating of Machine Learning for WaveShaping. 200627-1138 >

Data Dimension: 1024 x 1024

Input data number. if <= 99, datas are not saved

==> 500

< Data parameter >

common parameter = common_para_1024_2006251034.json

Data number = 500

line number = [1, 2, 3]

line peak = [2, 3, 4]

dim = 1024, dpi = 360, fig size = 2.84 x 2.84 inch

line width = 2.0 p

font size = 8.0 p

marker num = 7 ['+', 'D', '^', 'o', 's', 'v', 'x']

comment : WaveShaping_Data_

Confirm data parameter, process start OK? y or n -->

3.3.2 出力

データ数が100個以上の場合、データは自動的に保存される

データ数が100個以下の場合は、味見としてデータの保存は実行されず、サンプル画像の表示のみが行われる

以下に500個のデータを作成した実際の出力例をしめす

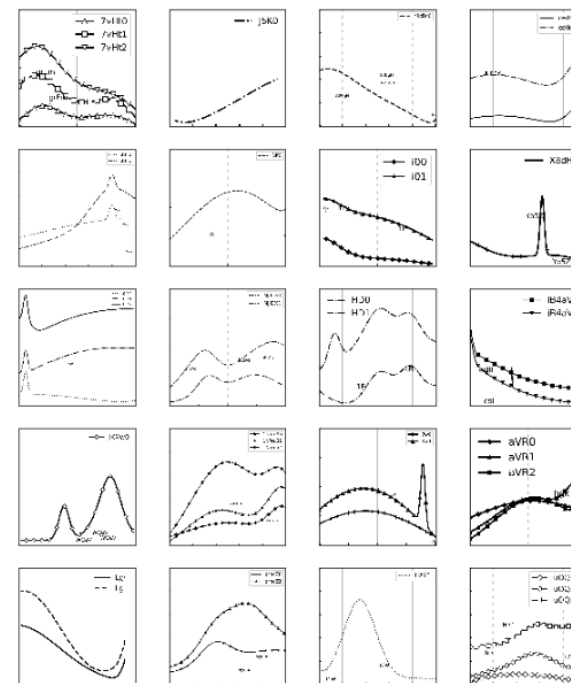
- 実行時間に続いてデータの中からランダムに選ばれた20個の学習データと教師データのサンプルが表示される

Data creating was started...

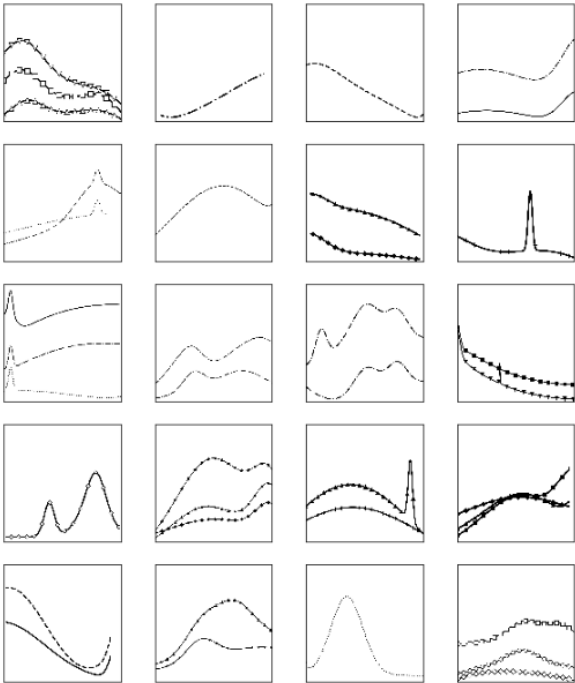
... Finished. exe_time (s) = 46.6, exe/n = 0.09

< Learning Data >

selected number = [388 285 442 151 101 34 234 193 122 230 263 40 453 188 383 180 336 460 475 28]



< Teaching Data >



- 次に、最下段に自動的に保存されたデータ名が示されプログラムは終了する（この例では ./data/OrgAxl_Data_1024_500_200626-1617.pik にデータが保存された）

Date is saved : WaveShaping_Data_1024_500_200627-1138.pik

3.3.3 データの出力形式

以下に出力されるデータの形式について説明する

参照:4.1 データ作成本体(Data_OrgAxl.py)プログラムリスト,339-357 行:モデルの保存 及び A3_Learning_OrgAxl_Manyual、3.3.1 入力

このデータは:

- pickle形式で保存された辞書型のデータ、<データ名.pik>
- 'data','out','para'の3つのキーを持ち、下表のような要素で構成されている

key	item
data	画像データ
out	教師データ
para	画像の大きさ, dpi, コメント,

画像データと教師データの形式

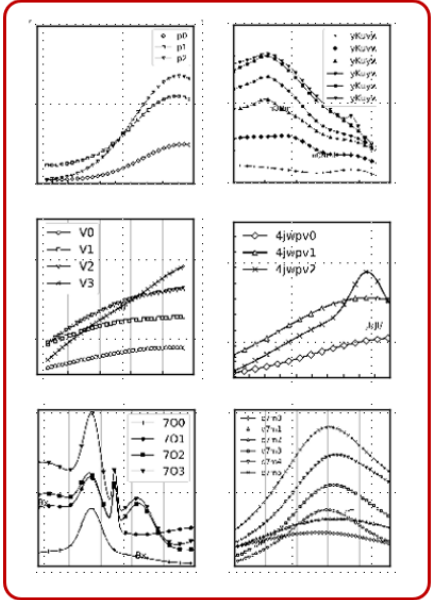
画像データ:

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)

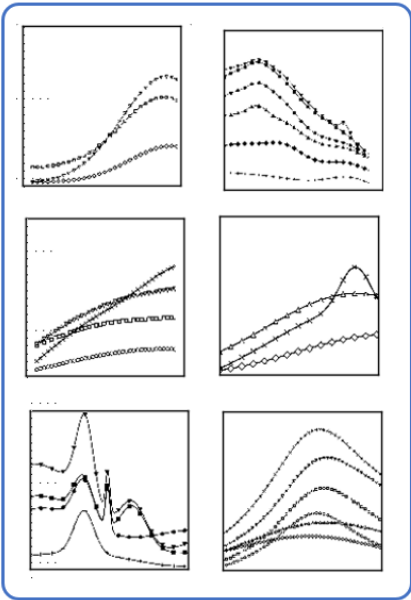
教師データ:

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:



4. プログラムリスト

以下に、データ作成本体（Data_OrgAxl.py）と関数ライブラリ（digitizer_lib）に収められている 波形成関数（Line.py）の全プログラムリストを示す

4.1 データ作成本体（Data_WavsShaping.py）

概要

```

1-8 行: 開始コメント
10-20 行: モジュールのインポート
28-34 行: 開始準備
36-79 行: データパラメータの設定
81-99 行: パラメータの確認
101-313 行: データの生成
314-349 行: チェック用の画像出力
351-363 行: モデルの保存
365 行: END
<br>

```

全リスト

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Mar.14,2019 / Fixed on Jun.30,2020
4
5  < 画像整形処理のための学習データを生成 >
6
7  @author: T.Kono
8  """
9
10 import numpy as np
11 import matplotlib.pyplot as plt
12 import cv2
13 import sys, string, random
14 import pickle, time
15 from datetime import datetime
16 import json
17
18 from keras.preprocessing.image import img_to_array
19
20 import kn_lib.line as line
21
22 # -----
23 #
24 #       Data Creating of Machine Learning for WaveShaping
25 #
26 # -----
27
28 # --- Opning -----
29 # Date and time
30 now = datetime.today().strftime('%y%m%d-%H%M')
31
32 # start comment
33 print('\n< Data Creating of Machine Learning for WaveShaping.',now,'>')
34 comment = 'WaveShaping_Data_'
35
36 # --- Parameter input and define -----
37 # data dimension
38 dim =1024 # fixed
39 print('\n Data Dimension:',dim,'x',dim)
40
41 # data number input
42 print('\nInput data number. if <= 99, datas are not saved')
43 loop = int(input(' ==> '))
44
45 # common parameter

```

```

46 '''同じdimの学習データは画像のパラメータを共有させる
47 '''
48 com_n = 'common_para_1024_2006251034.json'
49 with open(com_n, mode='r') as f:
50     com_para = json.load(f)
51
52 # parameter import
53 ndpi = com_para['para'][1]
54 lw = com_para['para'][2]
55 f_s = com_para['para'][3]
56 weigt = com_para['para'][4]
57 g_val = com_para['para'][5]
58 '''g_val[0][1] はマーカー間隔の最小値と最大値(pixel)
59 g_val[2]はマーカーの大きさ
60 '''
61 pix_poit = round(ndpi/72) # pixel number of point
62 ww = dim/ndpi ; hh = ww # graph width (inchs)
63
64 # marker
65 marker = ['+', 'D', '^', 'o', 's', 'v', 'x']
66 '''marker sample
67 ['o', 's', 'v']
68 ['+', 'D', '^', 'o', 's', 'v', 'x']
69 ['*', '+', ',', ':', '8', '<', '>', 'D', 'H', '^', '_', 'd', 'h', 'o', 'p', 's', 'v', 'x', '|']
70 '''
71 mak_l=len(marker)
72
73 # Line number
74 cho_l = [1,2,3]
75
76 # Peak of line number
77 '''1つの波形を作るために重ね合わせるガウス関数の数
78 '''
79 cho_p = [2,3,4]
80
81 # --- Parameter comfirmation -----
82 print('\n < Data parameter >')
83 print(' common parameter =',com_n)
84 print('\n Data number =',loop )
85 print(' line number =',cho_l)
86 print(' line peak =', cho_p)
87 print(' dim =',dim, ', dpi =',ndpi,
88       ', fig size =',{3.3}'.format(ww),
89       'x',{3.3}'.format(ww),'inch')
90 print(' line width =',lw,'p','\n font size =',f_s,'p')
91 print(' marker num =',len(marker),marker)
92 print('\n comment :',comment)
93
94 yn = input(' Confirm data parameter, process start OK? y or n --> ')
95 if yn == 'n':
96     print('Parameter is wrong') ;sys.exit()
97
98 print('\nData creating was started...')
99 s_time = time.time()
100
101 # --- Data creating -----
102 # initial arry set
103 ima_data = []
104 out_data = []
105 arrayimage = np.zeros([dim,dim,3],dtype=np.int8)
106 xxyy = np.zeros([2])

```



```

225 ax.spines["top" ].set_color('none')
226 ax.spines["left" ].set_color('none')
227 ax.spines["bottom"].set_color('none')
228
229 plt.ylim(0,1) ; plt.xlim(0,1)
230
231 # out data
232 """波形だけで、レジェンドなどを何も書いていない状態. これを教師データとする
233 ...
234 fig.canvas.draw()
235 buf = np.frombuffer (fig.canvas.tostring_rgb(),
236                     dtype=np.uint8
237                     )
238 imar = np.reshape(buf,(dim,dim,-1))
239
240 """3個に1個の割合でデータの前後にランダムな空白を挿入
241 ...
242 flg = l%3
243 if flg == 0 :
244     of_x = np.random.randint(0,dim*0.10,2)
245     cv2.rectangle(imar,(0,0),(of_x[0],dim),
246                  (255,255,255),-1) # left
247     cv2.rectangle(imar,(dim-of_x[1],0),(dim,dim),
248                  (255,255,255),-1) # right
249
250 gray_image = cv2.cvtColor(imar,cv2.COLOR_BGR2GRAY)
251 arrayimage = img_to_array(gray_image).astype(np.uint8)
252
253 out_data.append(arrayimage)
254
255 # legend draw
256 loc_n = 'best'
257 fr_ch = np.random.choice([1,0])
258 ax.legend(loc = loc_n,
259          fontsize = f_s*scale[2]*1.5,
260          framealpha = fr_ch)
261
262 # random text draw
263 for ii in range(0,len(pf)):
264     ax.text(pf[ii], pk[ii],
265            d_n2,
266            fontsize=f_s*scale[2]*1.2)
267
268 # tick draw
269 sel_d = np.random.choice(['in','out'])
270 sel_s = np.random.choice(['-',':',',',' ',' ',' '])
271
272 ax.tick_params(width = lw*scale[0],
273               length = 3*scale[1],
274               direction = sel_d)
275
276 ## 目盛り線の数の最大値をランダムに選ぶ
277 plt.locator_params(nbins = np.random.randint(1,12))
278
279 ## 縦のグリッド線を描く
280 ax.grid(axis='x',linestyle = sel_s)
281
282 # Buffer からデータを取得して学習データとする
283 fig.canvas.draw()
284 buf = np.frombuffer (fig.canvas.tostring_rgb(),
285                     dtype=np.uint8

```

```

286         )
287 imar = np.reshape(buf,(dim,dim,-1))
288
289 """3個に1個の割合でデータの前後にランダムな空白を挿入
290 ...
291 if flg == 0 :
292     cv2.rectangle(imar,(0,0),(of_x[0],dim),
293                  (255,255,255),-1) # left
294     cv2.rectangle(imar,(dim-of_x[1],0),(dim,dim),
295                  (255,255,255),-1) # right
296
297 gray_image = cv2.cvtColor(imar,cv2.COLOR_BGR2GRAY)
298 arrayimage = img_to_array(gray_image).astype(np.uint8)
299
300 ima_data.append(arrayimage)
301
302 # Learning Data
303 ima_data = np.asarray(ima_data)
304
305 # Teaching Data
306 out_data = np.asarray(out_data)
307
308 # Execution time
309 e_time = time.time()
310 ext_p_n = (e_time-s_time)/loop
311 print(' ... Finished. exe_time (s) = ',
312       '{:4.1f}'.format(e_time - s_time),
313       ', exe/n = ', '{:4.2f}'.format(ext_p_n))
314
315 # ---- Figure check -----
316 print('\n < Learning Data > ')
317 n = 20
318 n_list = np.random.randint(0,len(ima_data),n)
319 plt.close()
320 fig = plt.figure(figsize=(10,12))
321 ax1 = fig.add_subplot(111)
322 print(' selected number = ',n_list)
323 for i in range(n):
324     nn = n_list[i]
325     ax1 = plt.subplot(5,4,i+1)
326     ax1.imshow(np.reshape(ima_data[nn,:],(dim,-1)),cmap='gray')
327     # plt.gray()
328     ax1.get_xaxis().set_visible(False)
329     ax1.get_yaxis().set_visible(False)
330     plt.gray()
331 plt.show()
332
333 print('\n < Teaching Data > ')
334 fig = plt.figure(figsize=(10,12))
335 ax = fig.add_subplot(111)
336 # print(' * sample number = ',n_list)
337 for i in range(n):
338     nn = n_list[i]
339     ax = plt.subplot(5,4,i+1)
340     ax.imshow(np.reshape(out_data[nn,:],(dim,-1)),cmap='gray')
341     # plt.gray()
342     ax.get_xaxis().set_visible(False)
343     ax.get_yaxis().set_visible(False)
344 plt.show()
345
346 # Special exit

```

```

347 if loop <= 99:
348     print("\n /// Data were not saved, because loop was < 100. ///")
349     sys.exit()
350
351 # --- Data save -----
352 ml_data = {'data':ima_data,
353            'out':out_data,
354            'para':(dim,ndpi,com_n)}
355 d_name = 'WaveShaping_Data_'+str(dim)+'_'+str(loop)+'\
356          '+str(now)+'_pik'
357 with open(d_name,'wb') as f1:
358     pickle.dump(ml_data,
359                 f1,
360                 protocol=4
361                 )
362
363 print("\n Date is saved :",d_name)
364
365 # _____ END of code _____
366

```

4.2 波形生成関数（Line.py）

概要

1-7 行: 開始コメント
 9-9 行: モジュールのインポート
 15-38 行: 開始準備
 40-69 行: 波形の生成
 71 行: return

全リスト

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Jun.22,2020 / Fixed on Jun.30,2020
4
5  @author: T.Kono
6
7  """
8
9  import numpy as np
10
11  # ..... Line Create Function .....
12
13  def line_n(nn,p_n):
14
15      """p_n個のガウス関数を重ね合わせて波形を生成する関数
16
17      入力
18          nn : data number
19          p_n : peak number
20
21      出力
22          x : xの値
23          y : yの値
24          p_f : ピークの中心値

```

```

25     p_k : ピークの尖度
26     g_ : 全体のゲイン
27     ofs : オフセット
28
29     ""
30     # Xの値の生成
31     x = np.linspace(0,1,nn)
32     # yの領域生成
33     y = np.zeros([nn])
34
35     p_f = []
36     p_k = []
37     g_ = []
38     ofs = []
39
40     for i in range(0,p_n):
41
42         "" ガウス関数で波形を発生させる
43         下のパラメータをランダムに変えて、p_n本の波形を生成して重ね合わせる
44
45         fo : ピークの中心周波数
46         pk : ピークの尖度
47         h : 全体のゲイン
48         ofse : オフセット量
49
50         ""
51
52         fc = (np.random.randint(-30,130))/100
53         pk = (np.random.randint(10,500))/1000
54         ga = (np.random.randint(100,150))/100
55         ofse = (np.random.randint(10,100))/1000
56
57         # ガウス関数の生成
58         y_add = ofse + ga*np.exp(
59             -((x-fc)**2)/
60             (2*(pk)**2)
61             )
62         # ガウス関数の重ね合わせ
63         y = y + y_add
64
65         p_f.append(fc)
66         p_k.append(pk)
67         g_.append(ga)
68         ofs.append(ofse)
69
70     return x, y, p_f, p_k, g_, ofs
71
72

```

End