

# 原点位置・軸長読み取りの機械学習用データ作成プログラム（Data\_OrgAxl.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 データ形式
4. プログラムリスト

## 1. 機能

このプログラム(Data\_OrgAxl.py)は、機械学習用の学習データを作成するものである。作成されたデータは、グラフの原点位置と軸の長さを読み取る機械学習(Learning\_OrgAxl.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

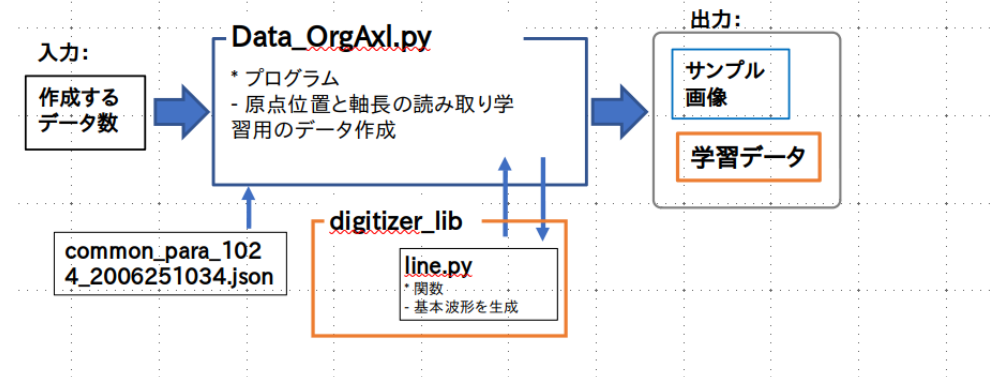
2) このプログラムを実行するには、以下のモジュールがあらかじめ準備された環境が必要である。また、記載されている以外のVersionでの動作確認は行っていないので注意が必要である

- 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\_OrgAxl.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\_OrgAxl.py）

1-9 行: 開始コメント  
11-21 行: モジュールのインポート  
29-35 行: 開始準備  
37-105 行: データパラメータの設定  
107-126 行: パラメータの確認  
128-314 行: データの生成  
316-337 行: チェック用の画像出力  
339-357 行: モデルの保存  
359 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

< Data Creating of ML for Origin Position and Axis Length 200626-1617 >

Input data number. \*if n <= 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 ML for Origin Position and Axis Length 200626-1617 >

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

==> 500

< Data parameter >

```
common parameter = common_para_1024_2006251034.json
dim               = 1024
dpi               = 360
data number       = 500
Graph size        = 2.84 x 2.84 inch
Line Number       = [1, 2, 3, 4, 5, 6]
Line width(normal) = 2.0 point( 10.0 pixel)
Font size(normal) = 8.0
Marker number     = 7 ['+', 'D', '^', 'o', 's', 'v', 'x']
Marker size(normal) = 6
Pitch of marker   = 50 - 80
scale factor      = [0.8, 0.9, 1.0, 1.13, 1.25]
```

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

\*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\*

### 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

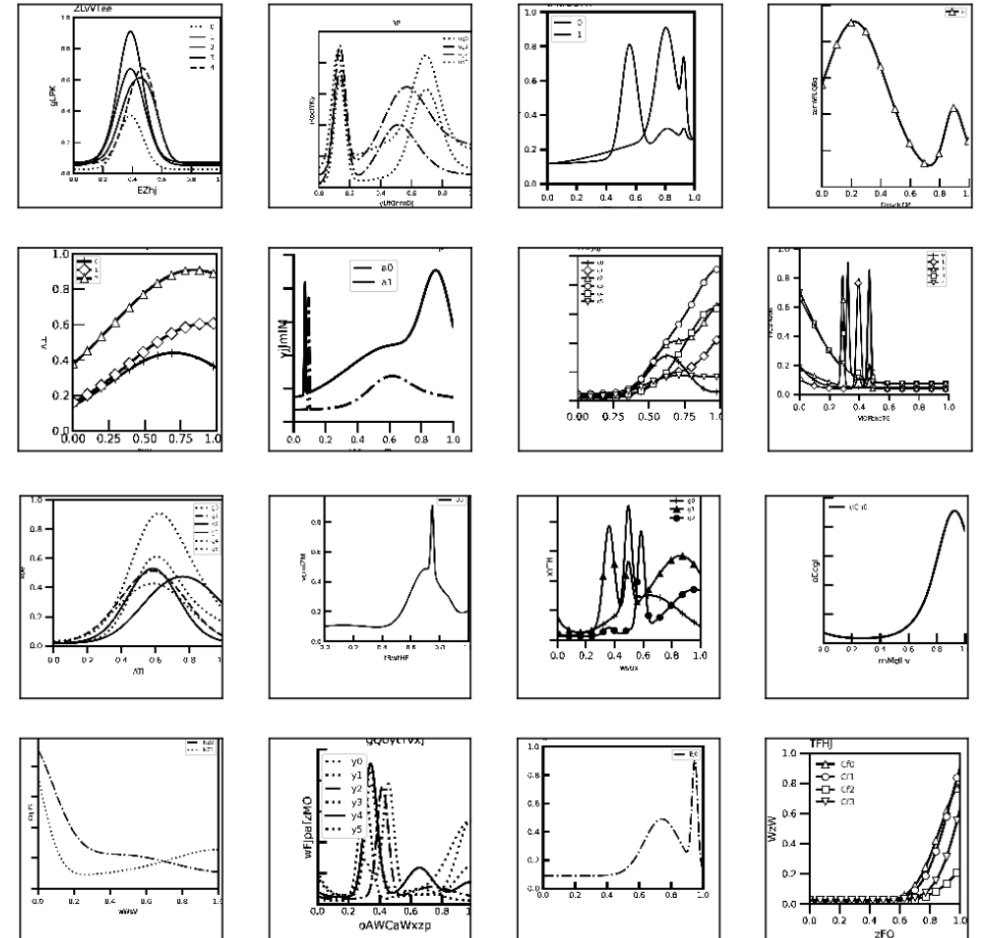
- 実行時間に続いてデータの中からランダムに選ばれた16個のサンプル表示される
- 次に、最下段に自動的に保存されたデータ名が示され（この例では ./data/OrgAxl\_Data\_1024\_500\_200626-1617.pikにデータが保存された）プログラムは終了する

Creating Process Start ...

... Finished. Execution time (s) = 38.1, exe\_t/n = 0.08

< Data figure >

sample number = [424 244 97 143 92 293 499 270 65 54 434 436 29 60 393 159]



Data were saved : ./data/OrgAxl\_Data\_1024\_500\_200626-1617.pik

### 3.3.3 データ形式

以下に、出力されるデータ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

- 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

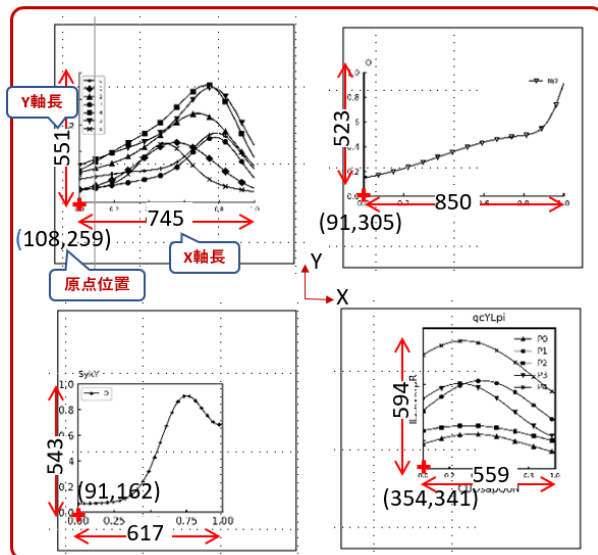
key	item
data	画像データ
out	教師データ
para	画像の大きさ, dpi, コメント, 原点位置の最小値, 原点位置の最大値, 軸長の最小値, 軸長の最大値, マーカー, 少ない波形ピークの数選択数, 大きな波形ピークの数選択数

### 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:  
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]
```

## 4. プログラムリスト

以下に、データ作成本体 (Data\_OrgAxI.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) の全プログラムリストを示す

### 4.1 データ作成本体 (Data\_OrgAxI.py)

#### 概要:

1-9 行: 開始コメント  
11-21 行: モジュールのインポート  
29-35 行: 開始準備  
37-105 行: データパラメータの設定  
107-126 行: パラメータの確認  
128-314 行: データの生成  
316-337 行: チェック用の画像出力  
339-357 行: モデルの保存  
359 行: END

#### 全リスト:

```
1  # -*- coding: utf-8 -*-
2  ""
3  Created on Nov.01,2018 / Fixed on Jun.30,2020
4
5  < 原点と軸長さの読み取りのための機械学習データ作成 >
6
7  @author: T.Kono
8
9  ""
10
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import cv2
14 import sys, string, random
15 import pickle, time
16 from datetime import datetime
17 import json
18
19 from keras.preprocessing.image import img_to_array
20
21 import kn_lib.line as line
22
23 # -----
24 #
25 #  Data Creating of Machine Learning for Origin Positin and Axis Length
26 #
27 # -----
28
29 # --- Opening -----
30 # Date and time
31 today = datetime.today().strftime('%y%m%d-%H%M')
32
33 # start comment
34 print('\n< Data Creating of ML for Origin Position and Axis Length',
35       today,'>')
```

```

36
37 # --- Parameter input and define -----
38 # data dimension
39 dim = 1024 # fixed
40
41 # data number
42 print('\n Input data number. *if n <= 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 axis_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 # image type
65 im_type = 'g'
66 '''gray に固定
67 '''
68 if im_type == 'g':
69     imt = 'L';i_t='gray'
70 elif im_type == 'm':
71     imt = '1';i_t='mono'
72 else :
73     print('Image type is wrong') ;sys.exit()
74
75 # line type
76 line_type = 'c'
77 '''cross に固定
78 '''
79 if line_type == 'c':
80     of_weight = [0.2,0.5,0.8];l_t='cross'
81 elif line_type == 'p':
82     of_weight = [1,1.2,1.5,2];l_t='para'
83 else :
84     print('Line type is wrong') ;sys.exit()
85
86 # marker
87 marker = ['+', 'D', '^', 'o', 's', 'v', 'x']
88 '''marker sample
89 ['o','s','v']
90 ['+', 'D', '^', 'o', 's', 'v', 'x']
91 ['*', '+', ',', ',', '8', '<', '>', 'D', 'H', '^', '_', 'd', 'h', 'o', 'p', 's', 'v', 'x', '|']
92 '''
93 mak_l=len(marker)
94
95 # p_weightは各ラインのピークの変化させる
96 p_weight = [0.8, 0.9, 1.1, 1.2, 1.3]

```

```

97
98 # Line number
99 cho_l = [1,2,3,4,5,6]
100
101 # Peak of line number
102 '''1つの波形を作るために重ね合わせるガウス関数の数
103 ...
104 cho_p_l = [1,2,3]
105 cho_p_h = [3,4,5]
106
107 # --- Parameter confirmation
108 print('\n < Data parameter >')
109 print(' common parameter = ',com_n)
110 print(' dim = ',dim, '\n dpi = ',ndpi,
111       '\n data number = ',loop )
112 print(' Graph size = ', '{:3.3}'.format(ww), 'x', '{:3.3}'.format(ww), 'inch')
113 print(' Line Number = ',cho_l)
114 print(' Line width(normal) = ',axis_lw, 'point', 'axis_lw*ndpi/72, 'pixel')
115 print(' Font size(normal) = ',f_s)
116 print(' Marker number = ',len(marker), marker)
117 print(' Marker size(normal) = ',g_val[2], '\n Pitch of marker = ',
118       g_val[0], '-', g_val[1],
119       '\n scale factor = ',weigt)
120 # print('\n comment = ',comment)
121
122 yn = input(' Confirm data parameter, process start OK? y or n --> ')
123 if yn == 'n' :
124     print('Parameter is wrong') ;sys.exit()
125
126 print('\n Creating Process Start ...'); s_time = time.time()
127
128 # --- Data create -----
129 # arry initial set
130 ima_data = []
131 out_data = np.zeros([0, 4])
132 arrayimage = np.zeros([dim,dim,3], dtype=np.int8)
133 xxyy = np.zeros([2])
134
135 # Create loop
136 for l in range(0,loop):
137
138     # marker on/off
139     mk_on_off = random.choice([1,0])
140
141     # lune number select
142     li_n=np.random.choice(cho_l)
143
144     # peak number select
145     '''マーカーの有無とラインの本数により波形のピーク数(ガンマ関数の重ね合わせ)を変え
る。
146     マーカーを描く場合もしくはラインの本数が3本以上では少なく、 マーカーを描かないか
ラインの本数
147     が3本以下の場合には多くする
148     ...
149     cho_p = cho_p_h
150     if mk_on_off == 1 or li_n > 3:
151         cho_p = cho_p_l
152     p_n = np.random.choice(cho_p)
153
154     # figure plot
155     plt.close()

```

```

156 fig = plt.figure(figsize = (ww, hh), dpi=ndpi)
157
158 # Origine position and x-y axis length
159 xy = np.random.randint(80,300,2)/1000 # origin position
160 xxyy[0] = np.random.randint(700,1020 - xy[0]*1000)/1000 # x length
161 xxyy[1] = np.random.randint(700,1020 - xy[1]*1000)/1000 # y length
162
163 # Base line
164 nn = int(xxyy[0]*dim-4)
165 ff = line.line_n(nn, p_n)
166
167 fu = ff[0]
168 yn = ff[1]
169 pf = ff[2]
170 pp = ff[3]
171 ph = ff[4]
172 ofs = ff[5]
173
174
175 weigt_ = []
176 y = np.zeros([nn])
177 yy = []
178
179 for j in range(0,li_n):
180     for i in range(0,p_n):
181         of_scl = np.random.choice(of_weight,1)
182         p_scl = np.random.choice(p_weight,2)
183         y_add = ofs[i] + ph[i]*np.exp(-((fu-pf[i]*p_scl[0])**2)/
184             (2*(pp[i]*p_scl[1]/(i+1))**2))
185         y = y*of_scl + y_add
186         yy.append(y)
187
188 yy = np.asarray(yy)
189 ymax = np.max(yy)
190 yn = yy/(ymax*1.1)
191
192 out = (dim - (yn*(dim-1))).astype(np.int32)
193
194 # figure drawing
195 '''line width scale randame choice
196 '''
197 scale = np.random.choice(weigt,4)/(np.random.randint(800,1250)/1000)
198
199 plt.rcParams['axes.linewidth'] = axis_lw*scale[0] # axis linewidth
200 ax = fig.add_axes([xy[0], xy[1], xxyy[0], xxyy[1]]) # figure position
201
202 # marker style
203 if mk_on_off == 1 :
204     mak_ = np.sort(np.random.choice(marker,
205         li_n+1,
206         replace=False
207     ))
208 else:
209     mak_ = [",",",",",",",",",","]
210
211 # marker full color
212 fc_ = np.random.choice(['w','k'])
213
214 dif = np.random.randint(g_val[0],g_val[1]) # マーカー間隔をランダムに選択
215 idx_d = []
216 moj = np.random.randint(0,4) # データ名、軸ラベルの文字数をランダムに選択

```

```

217 t_moj = np.random.randint(0,10,2) # タイトルの文字数をランダムに選択
218
219 d_n="".join(random.choices(string.ascii_letters, k=moj)) # データ名の文字列の生成
220
221 # line plot
222 for k in range(0,li_n):
223     if mk_on_off == 1 :
224         '''マーカーがある場合は線種は実線で固定、マーカーがない場合は線種はランダムに選
225         択する
226         '''
227         ls_ = 'solid'
228         else :
229             ls_ = np.random.choice(['-', '-.', ':', '--'])
230
231 # line plot
232 ax.plot(fu, yn[k,:],
233     ls = ls_,
234     color = 'k',
235     linewidth = axis_lw*scale[1],
236     label = d_n+str(k),
237     markevery = dif,
238     marker = mak_[k],
239     mfc = fc_,
240     mec = 'k',
241     ms = g_val[2]*scale[1]
242 )
243
244 ax_n="".join(random.choices(string.ascii_letters, k=t_moj[0])) # x軸ラベル文字列の生成
245 ay_n="".join(random.choices(string.ascii_letters, k=t_moj[0])) # y 軸ラベル文字列の生成
246 ax.set_xlabel(ax_n, fontsize = f_s*scale[2])
247 ax.set_ylabel(ay_n, fontsize = f_s*scale[2])
248
249 # draw title
250 tp_ = np.random.choice(['right','left','center'])
251 ty_n = "".join(random.choices(string.ascii_letters, k=t_moj[1]))
252 ax.set_title(ty_n, loc = tp_, fontsize = f_s*scale[2])
253
254 # draw legend
255 loc_n = 'best'
256 fr_ch = np.random.choice([1,0])
257 ax.legend(loc = loc_n,
258     fontsize = f_s*scale[2]*0.8,
259     framealpha = fr_ch
260 )
261
262 plt.ylim(0,1) ; plt.xlim(0,1)
263
264 # draw tic
265 sel = np.random.choice( ['in','out'] ) # 目盛線の方向
266 sel2 = np.random.choice( [True, False] ) # Y軸の目盛り数値のON/OFF
267 plt.tick_params(width = axis_lw*scale[0],
268     length = 8*scale[1],
269     labelsize = f_s*scale[3],
270     direction = sel,
271     labelleft = sel2
272 )
273
274 # frame shape (L and Box) randam choice
275 cho = ['k','none']
276 select = np.random.choice(cho)
277 ax.spines["right"].set_color(select)

```

```

277 ax.spines["top"].set_color(select)
278
279 # data create
280 '''Buffer からデータを取得して学習データとする
281 ...
282 fig.canvas.draw()
283 buf = np.frombuffer (fig.canvas.tostring_rgb(),
284                     dtype=np.uint8
285                     )
286
287 imar = np.reshape(buf,(dim,dim,-1))
288 gray_image = cv2.cvtColor(imar,cv2.COLOR_BGR2GRAY)
289 arrayimage = img_to_array(gray_image).astype(np.uint8)
290 ima_data.append(arrayimage)
291
292 l_offset = axis_lw*scale[0]*pix_poit # origin position offset
293 o_offset=l_offset/2 # axis length offset
294
295 out = [xy[0]*dim+o_offset, xy[1]*dim+o_offset,
296        xxyy[0]*dim-l_offset, xxyy[1]*dim-l_offset]
297 ...
298 原点位置と軸長さを教師データとして取得
299 ...
300
301 out_data = np.append(out_data,np.array([out[:]],axis=0)
302
303 tn = 'auto'
304 # Learning data
305 ima_data = np.asarray(ima_data)
306 # Teaching data
307 out_data = np.asarray(out_data)
308
309 # Execution time
310 e_time = time.time()
311 ext_p_n = (e_time-s_time)/loop
312
313 print(' ... Finished. Execution time (s) = ', '{:4.1f}'.format(e_time - s_time),
314       ', exe_t/n = ', '{:4.2f}'.format(ext_p_n))
315
316 # ---- Data check -----
317 plt.close()
318 print('\n < Data figure >')
319 n = 16
320 fig=plt.figure(figsize=(ww*6,ww*6))
321 n_list=np.random.randint(0,len(ima_data),n)
322 print(' sample number = ',n_list)
323 # print(' * upper = input, lower = output')
324 for i in range(n):
325     nn=n_list[i]
326     ax=plt.subplot(4,4,i+1)
327     plt.imshow(np.reshape(ima_data[nn,:],(dim,-1)))
328     plt.gray()
329     ax.get_xaxis().set_visible(False)
330     ax.get_yaxis().set_visible(False)
331
332 plt.show()
333
334 # special exit
335 if loop <= 99:
336     print("\n !!! Data were not saved, because Data number was < 100. ")
337     sys.exit()

```

```

338
339 # --- Data save -----
340 axis_test_graf = {'data':ima_data,
341                  'out':out_data,
342                  'para':(dim,
343                          ndpi,
344                          com_n,
345                          min(out_data[:,0]), max(out_data[:,0]),
346                          min(out_data[:,2]), max(out_data[:,2]),
347                          marker,
348                          cho_l,
349                          cho_p)
350                  }
351 sf_name = './data/OrgAxl_Data_'+str(dim)+'_'+str(loop)+'_'+str(today)
352 with open(sf_name+'.pik','wb') as f1:
353     pickle.dump(axis_test_graf,
354                 f1,
355                 protocol=4
356                 )
357 print('\n Data were saved : ',sf_name+'.pik')
358
359 # _____ End of Code _____
360
361

```

## 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
58         # ガウス関数の生成
59         y_add = ofse + ga*np.exp(
60             -((x-fc)**2)/
61             (2*(pk)**2)
62         )
63         # ガウス関数の重ね合わせ
64         y = y + y_add
65
66         p_f.append(fc)
67         p_k.append(pk)
68         g_.append(ga)
69         ofs.append(ofse)
70
71     return x, y, p_f, p_k, g_, ofs
72

```

End