

波形本数判別の機械学習プログラム (Learning_LineNumber.py) 使用説明書

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_LineNumber.py)は機械学習を行い、グラフからその波形本数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

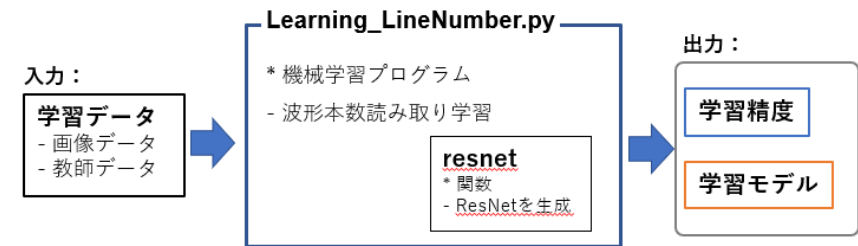
- Python 3.6
- Tensorflow-gpu 1.4.0
- Keras 2.2.4
- sklearn 0.19.1, 0.21.2
- matplotlib 2.2.2

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「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. 10-212 行: 学習モデルを生成する関数
 - 13-24 行: モジュールのインポート
 - 26-68 行: モデルパラメータの記述
 - 70-212 行: ResNetの生成
3. 214-429 行: 機械学習の実行
 - 218-224 行: モジュールのインポート
 - 226-234 行: 開始準備
 - 236-271 行: データの読み込みと振り分け
 - 273-282 行: パラメータの設定
 - 284-292 行: モデル構築
 - 294-325 行: モデルコンパイル
 - 327-337 行: モデルフィット
 - 339-405 行: 結果の評価と出力
 - 407-427 行: モデルの保存
 - 429 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、下記のようにプログラムの239行目に学習データの名前を記述して（例: ./data/LineNumber_1-6l_concat_360_6000_20191203-1155.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

```
236 # --- Data import -----
237 # data read
238 # data_n = input(' Input Learning Data Name --> \n ')
239 data_n = './data/LineNumber_1-6l_concat_360_6000_20191203-1155.pik\'
240 ; print('\n Data : ',data_n)
241 with open(data_n, mode='rb') as f:
242     test_graf = pickle.load(f)
```

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 >
- 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

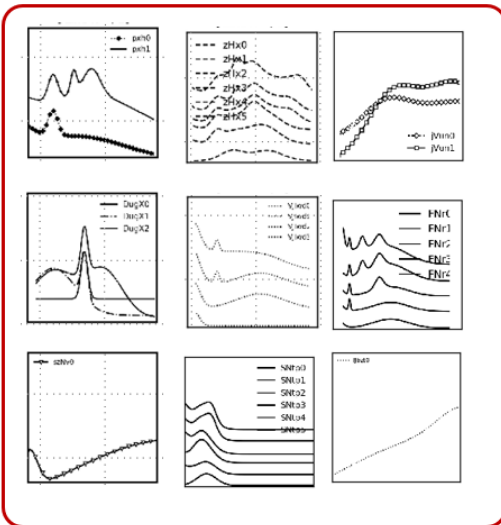
画像データ:

0-255の整数の数値をとり、そのサイズが360x360で1チャンネル(グレイスケール)の画像配列(360,360,1)

教師データ:

1~6の整数値をとる整数配列(1)

画像データ



教師データ

[2] [6] [2]
[3] [4] [5]
[1] [6] [1]

- キー'para'には1個の要素を持ち、1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	画像の大きさ

3.3.2 出力

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

< MACHINE LEARNING FOR LINE NUMBER READING _ 20191203-1318 >
Data : ./data/LineNumber_1-6l_concate_360_6000_20191203-1155.pik

--- Learning Process Start ...

This is ResNet-34
for Classification
with Dropout and l2-regularizer

Dropout = 0.0
L2-Regularizer = 1e-06

Optimizer : adam
loss : categorical_crossentropy

Learn_n = 5400 , Test_n = 600

first_filter = 24
batch = 30
patience = 20

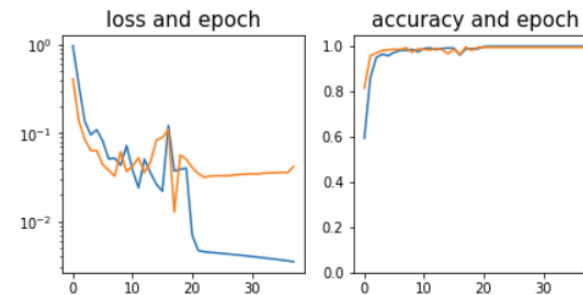
(2) 精度と正答率

lossとaccの履歴と正答率の表示

... Finished.
exe time(s) = 1092.94 , n_ep = 38

< Loss and accuracy >

loss = 3.51e-03 , val_loss = 4.20e-02 , delta = 3.85e-02
accuracy = 1.000



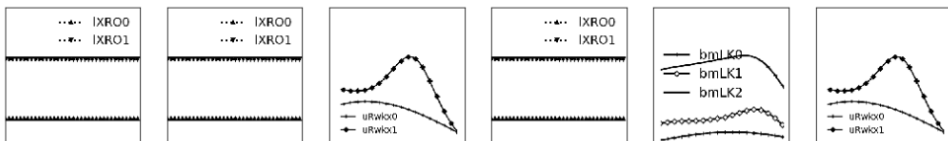
< Error and correct rate >

error_n = 3 / 600 , correct predict rate = 0.995

(3) エラー

うまく予測できなかったデータの中でランダムに6個を取り出して表示

```
< Error sample >
figure number = [138 138 247 138 389 247]
138 pred = [3]
138 pred = [3]
247 pred = [3]
138 pred = [3]
389 pred = [4]
247 pred = [3]
```



(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

```
Print the model shape? y or n --> y
Model is printed --> ./model/LineNum_I6_360_20191203-1318.png
Save the learned model? y or n --> y
Model is saved --> ./model/LineNum_I6_360_20191203-1318.h5
```

4. パラメータ

このプログラムのパラメータは256 x 256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可能である

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	360	256,360,1024
m_n	ResNetの種類	RN34	RN18,RN34,RN50,RN101,RN152
drp_n	ドロップアウト	0	0-1
reg_f	L2-Regularizer	1e-6	0-1
n_epoch	最大エポック数	200	1-
n_batch	バッチ数	30	1-
first_fil_n	開始フィルタ数	24	8,16,24,32,64
n_pat	EarlyStopで計算を打ち切る変動の最大数	20	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

5.1 概要

- 1-8 行: 開始コメント
- 10-212 行: 学習モデルを生成する関数**
13-24 行: モジュールのインポート
26-68 行: モデルパラメータの記述
70-212 行: ResNetの生成
- 214-429 行: 機械学習の実行**
218-224 行: モジュールのインポート
226-234 行: 開始準備
236-271 行: データの読み込みと振り分け
273-282 行: パラメータの設定
284-292 行: モデル構築
294-325 行: モデルコンパイル
327-337 行: モデルフィット
339-405 行: 結果の評価と出力
407-427 行: モデルの保存
429 行: END

5.2 全リスト

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Nov.28,2019
4
5 < グラフの本数を読み取るための機械学習 >
6
7 @author: T.KONO
8 """
9
10 # ----- Model Build Function -----
11 """Residual Net を作成するための関数
12 """
13 from keras.layers import (Input,
14                             Conv2D,
15                             Add,
16                             Activation,
17                             Dense,
18                             Dropout,
19                             BatchNormalization,
20                             GlobalAveragePooling2D,
21                             MaxPooling2D
22 )
23 from keras.models import (Model)
24 from keras.regularizers import l2
25
26 def resnet(input_shape,
27            num_outputs,
28            m_name,
29            n_fil,
30            drp_n,
31            l_f
```



```

154         )(shortcut)
155
156     X = Activation("relu")(X)
157     X = Dropout(drp_n)(X)
158     X = Conv2D(n_filter,
159               kernel_size=(1,1),
160               strides=(2,2),
161               padding="same",
162               kernel_regularizer=l2(reg_f)
163             )(X)
164
165     # X = BatchNormalization()(X)
166
167 else:
168     X = Activation("relu")(X)
169     X = Dropout(drp_n)(X)
170     X = Conv2D(n_filter,
171               kernel_size=(1,1),
172               padding="same",
173               kernel_regularizer=l2(reg_f)
174             )(X)
175
176     # X = BatchNormalization()(X)
177
178     X = Activation("relu")(X)
179     X = Dropout(drp_n)(X)
180     X = Conv2D(n_filter,
181               kernel_size=(3,3),
182               padding="same",
183               kernel_regularizer=l2(reg_f)
184             )(X)
185
186     # X = BatchNormalization()(X)
187     X = Activation("relu")(X)
188     X = Dropout(drp_n)(X)
189     X = Conv2D(n_filter * 4,
190               kernel_size=(1,1),
191               padding="same",
192               kernel_regularizer=l2(reg_f)
193             )(X)
194
195     # ショートカットとマージ
196     X = Add()(X, shortcut)
197     shortcut = X
198
199     # X = BatchNormalization()(X)
200
201     n_filter *= wide
202
203     X = Activation("relu")(X)
204     X = GlobalAveragePooling2D()(X)
205     X = Dropout(drp_n)(X)
206     y = Dense(num_outputs,
207              activation="softmax"
208            )(X)
209
210     model = Model(inputs=[input], outputs=[y])
211
212     return model
213
214 # -----

```

```

215 # Machine Learning Process
216 # -----
217
218 import numpy as np
219 import matplotlib.pyplot as plt
220 from keras.callbacks import EarlyStopping
221 from keras.utils import np_utils
222 from sklearn.model_selection import train_test_split
223 import pickle, time
224 from datetime import datetime
225
226 # --- Opening -----
227 # date
228 today = datetime.today().strftime('%y%m%d-%H%M')
229
230 # Opening print
231 print('\n < MACHINE LEARNING FOR LINE NUMBER READING __, today,> ')
232
233 # start time
234 s_time = time.time()
235
236 # --- Data import -----
237 # data read
238 # data_n = input(' Input Learning Data Name --> \n ')
239 data_n = './data/LineNumber_1-6l_concate_360_6000_20191203-1155.pik\'
240 ; print('\n Data : ',data_n)
241 with open(data_n, mode='rb') as f:
242     test_graf = pickle.load(f)
243
244 # X-normalize and black-white reverse
245 dim = test_graf['para'][0]
246 X = test_graf['data']
247 if np.max(X[0,:]) == 255:
248     nor = 255 # case of gray data
249 else:
250     nor = 1 # case of monochro data
251 X = 1-X/nor # black-white reverse
252
253 # line number
254 l_n = 6
255
256 print('\nLeraning Process Start ... ')
257
258 # Y-categorize
259 Yo = test_graf['out']
260 Y = np_utils.to_categorical(Yo) # 教師データをカテゴリ型に変換
261
262 # input/output size
263 n_in = len(X[0])
264 n_out = len(Y[0])
265
266 # Data split
267 x_train, x_test, y_train, y_test = train_test_split(X, Y,
268                                                    test_size = 0.1,
269                                                    train_size = 0.9
270                                                    )
271 del X,Y
272
273 # --- Iteration parameter -----
274 m_n = 'RN34'
275

```

```

276 n_epoch = 200
277 n_batch = 30
278 first_fil_n = 24
279 drp_n = 0.0
280 l2_f = 1e-6
281 n_pat = 20
282 flag_ver = 0
283
284 # --- Model build -----
285 input_ = x_train.shape[1:]
286 model = resnet(input_,
287               n_out,
288               m_n,
289               first_fil_n,
290               drp_n,
291               l2_f
292             )
293
294 # --- Model compile -----
295 # Optimizer
296 # adam
297 print('\n Optimizer : adam')
298 optimizer = 'adam'
299
300 ''' Optimizer selection
301 # adam
302 print('\n Optimizer : adam')
303 optimizer = 'adam'
304
305 # SGD + momentum
306 from keras.optimizers import SGD
307 print('\n Optimizer : SDG + Momentum')
308 optimizer = SGD(decay=1e-6, momentum=0.9, nesterov=True)
309 '''
310
311 # loss
312 loss = 'categorical_crossentropy'
313 print(' loss :',loss)
314
315 # compile
316 model.compile(loss = loss,
317              optimizer = optimizer,
318              metrics = ['accuracy']
319             )
320
321 # estimator parameter print
322 print('\n Learn_n =',len(x_train),'Test_n =',len(x_test))
323 print('\n first_filter = ',first_fil_n)
324 print(' batch = ',n_batch)
325 print(' patience = ',n_pat)
326
327 # --- Model fit -----
328 early_stopping = EarlyStopping(monitor='val_loss',
329                               patience=n_pat)
330
331 hist = model.fit(x_train, y_train,
332                 epochs = n_epoch,
333                 batch_size = n_batch,
334                 validation_data = (x_test, y_test),
335                 verbose = flag_ver,
336                 callbacks = [early_stopping])

```

```

337 )
338
339 # --- Predict and evaluation -----
340 predict_ = model.predict(x_test, batch_size=n_batch)
341 dif = np.round(y_test - predict_)
342
343 # error list
344 dif_ = abs(dif)
345 pass_list = []; ng_list=[]
346 for i in range(0,len(y_test)):
347     if max(dif_[i]) == 0:
348         pass_list.append(i)
349     else:
350         ng_list.append(i)
351
352 # correct predict rate
353 cpr = 1 - len(ng_list)/(len(dif)) # correct predict rate
354
355 # Execution time
356 e_time = time.time() # end time
357 n_ep = len(hist.history['loss'])
358 print('\n... Finished.\n exe time(s) = ',
359       '{:6.2f}'.format(e_time - s_time),
360       ', n_ep =',n_ep)
361
362 # accuracy print
363 de_loss = hist.history['val_loss'][n_ep-1]-hist.history['loss'][n_ep-1]
364 print('\n< Loss and accuracy >\n loss = ',
365       '{:4.2e}'.format(hist.history['loss'][n_ep-1]),
366       ', val_loss = ', '{:4.2e}'.format(hist.history['val_loss'][n_ep-1]),
367       ', delta = ', '{:4.2e}'.format(de_loss),
368       '\n accuracy = ', '{:4.3f}'.format(hist.history['acc'][n_ep-1]))
369
370 # loss and accuracy history
371 fig = plt.figure(figsize=(7, 7))
372 plt.subplot(2, 2, 1)
373 x_ep = range(len(hist.history['loss']))
374 plt.plot(x_ep,hist.history['loss'])
375 plt.plot(x_ep,hist.history['val_loss'])
376 plt.yscale('log')
377 plt.title("loss and epoch",fontsize=15)
378
379 plt.subplot(2, 2, 2)
380 x_ep = range(len(hist.history['acc']))
381 plt.plot(x_ep,hist.history['acc'])
382 plt.plot(x_ep,hist.history['val_acc'])
383 plt.title("accuracy and epoch",fontsize=15)
384 plt.ylim(0, 1.05)
385 plt.show()
386
387 print('< Error and correct rate >')
388 print(' error_n = ',len(ng_list),'/',len(dif),
389       ', correct predict rate = ', '{:4.3f}'.format(cpr))
390
391 # Error sample drawing
392 print('\n< Error sample >')
393 fig = plt.figure(figsize=(18,18))
394 n = 6
395 n_list = np.random.choice(ng_list, n)
396 print(' figure number = ', n_list)
397 for i in range(n):

```

```

398     nn = n_list[i]
399     print(nn,'pred = ',np.where(np.round(predict_[nn])>0)[0])
400     ax = plt.subplot(4,n,i+1)
401     plt.imshow(np.reshape(1-x_test[nn,:], (dim, -1)),cmap='gray')
402     ax.get_xaxis().set_visible(False)
403     ax.get_yaxis().set_visible(False)
404     plt.ylim(dim,0) ; plt.xlim(0,dim)
405     plt.show()
406
407     # --- Model print and save -----
408     mp_flag = 'n' ; ms_flag = 'n'
409     model_n = './model/LineNum_'+str(l_n)+'_'+str(dim)+'_'+str(today)
410
411     # model print
412     mp_flag = input(' Print the model shape? y or n --> ')
413     if mp_flag == 'y':
414         from keras.utils import plot_model
415         plot_model(model, to_file= model_n+'.png',
416                   show_shapes=True)
417         print(' Model is printed --> ',model_n+'.png')
418     else :
419         print(' Model is not printed. ')
420
421     # model save
422     ms_flag = input(' Save the learned model? y or n --> ')
423     if ms_flag == 'y':
424         model.save(model_n+'.h5')
425         print(' Model is saved --> '+model_n+'.h5')
426     else :
427         print(' Model is not saved. ')
428
429     # _____ END OF CODE _____

```

END