

test

April 26, 2022

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import shutil
import random
import cv2
import glob
from PIL import Image
import PIL.ImageOps
from collections import Counter
from imblearn.over_sampling import SMOTE
from tqdm import tqdm
from imblearn.combine import *
from imblearn.under_sampling import TomekLinks

import time

from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

```
[2]: def count_and_plot(y):
    counter = Counter(y)
    print(counter)
    for k,v in counter.items():
        print('Class=%s, n=%d (%.3f%%)' % (k, v, v / len(y) * 100))
    plt.bar(counter.keys(), counter.values())
    plt.show()
```

```
[3]: def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except OSError:
        print ('Error: Creating directory. ' + directory)
```

```
[4]: def center_crop(img, set_size):

    h, w, c = img.shape

    if set_size > min(h, w):
        return img

    crop_width = set_size
    crop_height = set_size

    mid_x, mid_y = w//2, h//2
    offset_x, offset_y = crop_width//2, crop_height//2

    crop_img = img[mid_y - offset_y:mid_y + offset_y, mid_x - offset_x:mid_x +
    ↪offset_x]
    return crop_img
```

```
[5]: def data_augmentation(path):
    # data load
    train_df = pd.read_csv(path)
    labelList = train_df['label'].unique()
    label = train_df['label'].to_list()

    for i in range(len(labelList)):
        folderpath = './train_data/' + labelList[i]
        createFolder(folderpath)
        # ./train_data/transistor-good
        # ./train_data/capsule-good
        # ./train_data/wood-good

    # label
    for i in range(len(train_df)):
        src_path = './data/train/'
        dst_path = './train_data/'
        filenum = i + 10000
        filename = str(filenum)+'.png'

        src_path += filename
        dst_path += label[i]

        shutil.copy(src_path, dst_path)
        # ./data/train/10000.png ./train_data/transistor-good
        # ./data/train/10001.png ./train_data/capsule-good

    # preprocessing
    labelCount = train_df[['class', 'label']].groupby('label').count().
    ↪rename(columns={'class': 'count'})
```

```

#
origin_datanum = labelCount['count'].tolist()
# 10, 11, 11, 209, 7, 6, 6, 7, 5, 224,...

data = train_df.values
X, y = data[:, 1], data[:, -1]
for i in range(len(X)):
    X[i] = X[i][:5]

X = np.array(X, dtype = np.float64)
X = X.reshape((4277, 1))

count_and_plot(y)

# oversampling
X_resampled, y_resampled = SMOTETomek(random_state=0, smote = SMOTE(k_neighbors=3)).fit_resample(X, y)
count_and_plot(y_resampled)
#print(y_sap)

# oversampled label file_name dataframe
y_resampled2 = y_resampled.reshape((y_resampled.size, 1))
Xy = np.concatenate((X_resampled, y_resampled2), axis =1)

df = pd.DataFrame(Xy)
df.to_csv('smotetomek_result.csv', index=False)

train_df2 = pd.DataFrame(Xy)
augmented_filenames = train_df2[0].tolist()

labelCount2 = train_df2.groupby(1).count().rename(columns={'label':'count'})

# augmentation
oversampled_datanum = labelCount2[0].tolist()
# [391, 391, 391, 391, 391, 391, 391, 391,...

#oversampling
oversampling_num = [x-y for x, y in zip(oversampled_datanum,origin_datanum)]
print("oversampling : ", oversampling_num)
# [381, 380, 380, 182, 384, 385, 385, 384,...

# img augmentation
labelList = np.sort(labelList)
print("total num : ", len(labelList))

```

```

for i in range(len(labelList)):#    dir        label        augmentation

    augmented_num = oversampling_num[i]
    # augmentation
    origin_file_path = './train_data/' + labelList[i] + '/'

    # augmented image
    save_file_path = './data/train/'
    file_names = os.listdir(origin_file_path)
    # ['10000.png', '10002.png', '10009.png', '10042.png', '10049.png',

    # label img
    before = train_df[train_df['label'] == labelList[i]]
    # sampling label img
    after = train_df2[train_df2[1] == labelList[i]]

    # file_name
    before = (before['file_name'].tolist())
    for j in range(len(before)):
        before[j] = before[j][:5]

    before = np.array(before, dtype = np.int64)
    after = (after[0].tolist())

    # before    after
    sample1 = list(set(before) - set(after))
    #print(sample1, len(sample1))

    # after    before
    sample2 = list(set(after) - set(before))
    #print(sample2, len(sample2))

    for j in tqdm(range(len(sample1)), desc = "%d : %s label's deleting_
→process"%(i+1, labelList[i])): #    img
        del_file_name = save_file_path + str(sample1[j]) + '.png'
        #file_names = os.listdir(origin_file_path)
        #print("now files: ", len(file_names))

        #print("del: ", del_file_name)
        os.remove(del_file_name)

    for j in tqdm(range(len(sample2)), desc = "%d : %s label's augmentation_
→process"%(i+1, labelList[i])): #    img

```

```

aug_file_name = sample2[j]
# augmentation
random_file_num = random.randrange(0, len(file_names))
origin_file_name = file_names[random_file_num]

image = Image.open(origin_file_path+origin_file_name)
random_augment = random.randrange(1,4)

if(random_augment == 1):
    #
    convert_color_img = image.convert('L')
    convert_color_img.save(save_file_path +str(aug_file_name)+ '.
→png')

elif(random_augment == 2):
    # center_crop

    img = cv2.imread(origin_file_path+origin_file_name)
    img_cvt = center_crop(img, 1000)
    cv2.imwrite(save_file_path +str(aug_file_name)+ '.png', img_cvt)

elif(random_augment == 3):
    # color_change

    color = [cv2.COLOR_BGR2RGB, cv2.COLOR_BGR2GRAY,
              cv2.COLOR_BGR2XYZ, cv2.COLOR_BGR2YCrCb, cv2.
→COLOR_BGR2HSV,
              cv2.COLOR_BGR2Lab, cv2.COLOR_BGR2Luv, cv2.
→COLOR_BGR2HLS, cv2.COLOR_BGR2YUV]
    img = cv2.imread(origin_file_path+origin_file_name)
    r = random.randrange(0, len(color))

    img_cvt = cv2.cvtColor(img, color[r])
    cv2.imwrite(save_file_path +str(aug_file_name)+ '.png', img_cvt)

return [X_resampled, y_resampled]

```

[]: