توضیحات پیاده سازی مقاله درس بینایی ماشین

عنوان مقاله:

Weakly Supervised Deep Learning for COVID-19 Infection Detection and Classification From CT Images

عنوان مقاله به فارسی:

یادگیری عمیق با نظارت ضعیف برای تشخیص و طبقه بندی عفونت کووید-۱۹ از تصاویر CT

نگارنده:

امیرپویا گمرکچی

بهار 1403

فصل1:

1-1:دیتاست ها و داده های مقاله:

دیتاست های این مقاله به طورعمومی در دسترس نبود.

به همین دلیل من از چندین دیتا ست متفاوت برای بررسی و پیاده سازی این مقاله استفاده کردم.

1-2:رفرنس ها یا مراجع

با توجه به رفرنس های مقاله دیتاهایی که استفاده شده ترکیبی از چند مقاله است.

بنابراین دیتاست مورد استفاده به شکل یکسان مورد استفاده نیست.

1-3:روش جست و جوی دیتاست ها در گیت هاب:

1\_ابتدا کلمات متناسب با مقاله و عنوان مقاله در گیت هاب سرچ شد.

2\_در مرحله دوم کلمات کلیدی مقاله در گیت هاب سرچ شد.

قسمت اول

import os

import cv2

import glob

import numpy as np

from tqdm import tqdm

from .utils import load\_nifti\_volume, CLASS\_MAP, LESION\_THRESHOLD

\_CT\_FNAME\_GLOB = 'volume-covid19-A-\*\_ct.nii.gz'

\_SEG\_FNAME\_GLOB = 'volume-covid19-A-\*\_seg.nii.gz'

def process\_covid\_19\_20\_data(root\_dir, output\_dir):

"""Process slices for COVID-19-20 challenge studies"""

filenames = []

classes = []

ct\_files = sorted(glob.glob(os.path.join(root\_dir, \_CT\_FNAME\_GLOB)))

seg\_files = sorted(glob.glob(os.path.join(root\_dir, \_SEG\_FNAME\_GLOB)))

for ct\_file, seg\_file in tqdm(zip(ct\_files, seg\_files), total=len(ct\_files)):

volume = load\_nifti\_volume(ct\_file)

volume = np.swapaxes(volume, 1, 2) # transpose to natural orientation

seg = load\_nifti\_volume(seg\_file)

seg = np.swapaxes(seg, 1, 2) # transpose to natural orientation

lesion\_frac = np.sum(seg, axis=(1, 2))/(volume.shape[1]\*volume.shape[2])

lesion\_slices = np.where(lesion\_frac > LESION\_THRESHOLD)[0]

for i in lesion\_slices:

fname = os.path.basename(ct\_file).split('.')[0] + '-{:04d}.png'.format(i)

out\_file = os.path.join(output\_dir, fname)

filenames.append(fname)

classes.append(CLASS\_MAP['COVID-19'])

if not os.path.exists(out\_file):

slc = volume[i]

cv2.imwrite(out\_file, slc)

return filenames, classes

قسمت دوم

import os

import cv2

import glob

import numpy as np

from tqdm import tqdm

from .utils import load\_nifti\_volume, CLASS\_MAP, LESION\_THRESHOLD

\_CT\_FNAME\_GLOB = 'volume-covid19-A-\*\_ct.nii.gz'

\_SEG\_FNAME\_GLOB = 'volume-covid19-A-\*\_seg.nii.gz'

def process\_covid\_19\_20\_data(root\_dir, output\_dir):

"""Process slices for COVID-19-20 challenge studies"""

filenames = []

classes = []

ct\_files = sorted(glob.glob(os.path.join(root\_dir, \_CT\_FNAME\_GLOB)))

seg\_files = sorted(glob.glob(os.path.join(root\_dir, \_SEG\_FNAME\_GLOB)))

for ct\_file, seg\_file in tqdm(zip(ct\_files, seg\_files), total=len(ct\_files)):

volume = load\_nifti\_volume(ct\_file)

volume = np.swapaxes(volume, 1, 2) # transpose to natural orientation

seg = load\_nifti\_volume(seg\_file)

seg = np.swapaxes(seg, 1, 2) # transpose to natural orientation

lesion\_frac = np.sum(seg, axis=(1, 2))/(volume.shape[1]\*volume.shape[2])

lesion\_slices = np.where(lesion\_frac > LESION\_THRESHOLD)[0]

for i in lesion\_slices:

fname = os.path.basename(ct\_file).split('.')[0] + '-{:04d}.png'.format(i)

out\_file = os.path.join(output\_dir, fname)

filenames.append(fname)

classes.append(CLASS\_MAP['COVID-19'])

if not os.path.exists(out\_file):

slc = volume[i]

cv2.imwrite(out\_file, slc)

return filenames, classes

قسمت سوم

import os

import cv2

import csv

import glob

import numpy as np

from tqdm import tqdm

from .utils import CLASS\_MAP

from data\_utils import ensure\_uint8

\_RESCALE\_SLOPE = 1

\_RESCALE\_INTERCEPT = -1024

\_CLASS\_MAP = {'Positive': CLASS\_MAP['COVID-19'], 'Negative': CLASS\_MAP['Normal']}

\_FNAME\_FMT\_GLOB = '\*\_{}\_\*.tif'

def \_uint16\_hu\_to\_uint8(data):

data = data.astype(np.float)\*\_RESCALE\_SLOPE + \_RESCALE\_INTERCEPT

return ensure\_uint8(data)

def process\_covid\_ctset\_data(meta\_csv, root\_dir, output\_dir):

"""Processes slices for all patients in the given COVID-CTSet CSV file"""

filenames = []

classes = []

with open(meta\_csv, 'r') as f:

reader = list(csv.DictReader(f, delimiter=',', quotechar='|'))

for row in tqdm(reader):

fnames = \_process\_covid\_ctset\_patient(row['Patient ID'], root\_dir, output\_dir)

cls = \_CLASS\_MAP[row['COVID-19 Infection']]

filenames.extend(fnames)

classes.extend([cls]\*len(fnames))

return filenames, classes

def \_process\_covid\_ctset\_patient(pid, img\_dir, output\_dir):

"""Processes images for a COVID-CTSet patient"""

filenames = []

image\_files = glob.glob(os.path.join(img\_dir, \_FNAME\_FMT\_GLOB.format(pid)))

for imf in image\_files:

fname = os.path.basename(imf).replace('.tif', '.png')

filenames.append(fname)

out\_file = os.path.join(output\_dir, fname)

if not os.path.exists(out\_file):

image = cv2.imread(imf, cv2.IMREAD\_UNCHANGED)

image = \_uint16\_hu\_to\_uint8(image)

cv2.imwrite(out\_file, image)

return filenames