توضیحات پیاده سازی مقاله درس بینایی ماشین

عنوان مقاله:

AI-based wavelet and stacked deep learning architecture for detecting coronavirus (COVID-19) from chest X-ray images

نگارنده:

پوریا دهقان چناری

بهار 1403

فصل1:

1-1:دیتاست ها و داده های مقاله:

دیتاست های این مقاله به طورعمومی در دسترس بود.

1-2:رفرنس ها یا مراجع

[1] Zheng YY, Ma YT, Zhang JY, Xie X. COVID-19 and the cardiovascular system. Nat Rev Cardiol 2020;17(5):259–60.

[2] Worldometers. Worldometers information. 2023, URL https://www.worldometers.info/coronavirus/, [Accessed 09-01-2023].

[3] Wang W, Xu Y, Gao R, Lu R, Han K, Wu G, et al. Detection of SARS-CoV-2 in different types of clinical specimens. JAMA 2020;323(18):1843–4.

[4] Zhang J, Xie Y, Li Y, Shen C, Xia Y. Covid-19 screening on chest x-ray images using deep learning based anomaly detection. 2020, arXiv preprint arXiv:2003.12338.

[5] Salman FM, Abu-Naser SS, Alajrami E, Abu-Nasser BS, Alashqar BA. Covid-19 detection using artificial intelligence 2020.

[6] Abbas A, Abdelsamea MM, Gaber MM. Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. Appl Intell 2021;51(2):854–64.

[7] Wang L, Lin ZQ, Wong A. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. Sci Rep 2020;10(1):1–12.

[8] Sethy PK, Behera SK. Detection of coronavirus disease (covid-19) based on deep features. Preprints 2020.

[9] Apostolopoulos ID, Mpesiana TA. Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. Phys Eng Sci Med 2020;43(2):635–40.

[10] Marques G, Agarwal D, de la Torre Díez I. Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. Appl Soft Comput 2020;96:106691.

[11] Gupta A, Gupta S, Katarya R. InstaCovNet-19: A deep learning classification model for the detection of COVID-19 patients using Chest X-ray. Appl Soft Comput 2021;99:106859.

[12] Mahmud T, Rahman MA, Fattah SA. CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization. Comput Biol Med 2020;122:103869.

[13] Ozturk T, Talo M, Yildirim EA, Baloglu UB, Yildirim O, Acharya UR. Automated detection of COVID-19 cases using deep neural networks with X-ray images. Comput Biol Med 2020;121:103792.

[14] Pereira RM, Bertolini D, Teixeira LO, Silla Jr CN, Costa YM. COVID-19 identification in chest X-ray images on flat and hierarchical classification scenarios. Comput Methods Programs Biomed 2020;194:105532.

[15] Khan AI, Shah JL, Bhat MM. CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. Comput Methods Programs Biomed 2020;196:105581. Computers and Electrical Engineering 108 (2023) 108711 18 R. Soundrapandiyan et al.

[16] Ismael AM, Sengur A. Deep learning approaches for COVID-19 detection based on chest X-ray images. Expert Syst Appl 2021;164:114054.

[17] Minaee S, Kafieh R, Sonka M, Yazdani S, Soufi GJ. Deep-COVID: Predicting COVID-19 from chest X-ray images using deep transfer learning. Med Image Anal 2020;65:101794.

[18] Gour M, Jain S. Automated COVID-19 detection from X-ray and CT images with stacked ensemble convolutional neural network. Biocybern Biomed Eng 2022;42(1):27–41.

[19] Bargshady G, Zhou X, Barua PD, Gururajan R, Li Y, Acharya UR. Application of CycleGAN and transfer learning techniques for automated detection of COVID-19 using X-ray images. Pattern Recognit Lett 2022;153:67–74.

[20] Al-Monsur A, Kabir MR, Ar-Rafi AM, Nishat MM, Faisal F. Covid-EnsembleNet: an ensemble based approach for detecting Covid-19 by utilising chest X-ray images. In: 2022 IEEE world AI IoT congress. 2022, p. 351–6.

[21] Xu X, Jiang X, Ma C, Du P, Li X, Lv S, et al. A deep learning system to screen novel coronavirus disease 2019 pneumonia. Engineering 2020;6(10):1122–9.

[22] Amyar A, Modzelewski R, Li H, Ruan S. Multi-task deep learning based CT imaging analysis for COVID-19 pneumonia: Classification and segmentation. Comput Biol Med 2020;126:104037.

[23] Wang SH, Govindaraj VV, Górriz JM, Zhang X, Zhang YD. Covid-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. Inf Fusion 2021;67:208–29.

[24] Pathak Y, Shukla PK, Tiwari A, Stalin S, Singh S. Deep transfer learning based classification model for COVID-19 disease. Irbm 2020.

[25] Ardakani AA, Kanafi AR, Acharya UR, Khadem N, Mohammadi A. Application of deep learning technique to manage COVID-19 in routine clinical practice using CT images: Results of 10 convolutional neural networks. Comput Biol Med 2020;121:103795.

[26] Gozes O, Frid-Adar M, Greenspan H, Browning PD, Zhang H, Ji W, et al. Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection & patient monitoring using deep learning ct image analysis. 2020, arXiv preprint arXiv:2003.05037.

[27] Kassania SH, Kassanib PH, Wesolowskic MJ, Schneidera KA, Detersa R. Automatic detection of coronavirus disease (COVID-19) in X-ray and CT images: a machine learning based approach. Biocybern Biomed Eng 2021;41(3):867–79.

[28] Loey M, Smarandache F, M. Khalifa NE. Within the lack of chest COVID-19 X-ray dataset: a novel detection model based on GAN and deep transfer learning. Symmetry 2020;12(4):651.

[29] Patrizi N, Tsiropoulou EE, Papavassiliou S. Health data acquisition from wearable devices during a pandemic: A techno-economics approach. In: ICC 2021-IEEE international conference on communications. 2021, pp. 1–6.

[30] ChestXrayImages. Chest X-ray images (pneumonia). 2022, URL https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia, [Accessed 26-12-2022].

قسمت اول:

data\_path = '<>'

gen\_path = '<>'

covid\_path = '<>'

covid\_std\_path = '<>'

covid\_dwt\_path = '<>'

normal\_path = '<>'

normal\_std\_path = '<>'

normal\_dwt\_path = '<>'

bacterial\_path = '<>'

bacterial\_std\_path = '<>'

bacterial\_dwt\_path = '<>'

viral\_path = '<>'

viral\_std\_path = '<>'

viral\_dwt\_path = '<>'

dir\_path = '<>'

train\_path = '<>'

test\_path = '<>'

covid\_path\_name = 'COVID'

covid\_std\_path\_name = 'COVID\_STD'

covid\_dwt\_path\_name = 'COVID\_DWT'

covid\_dwt\_path\_name\_2 = 'COVID\_DWT\_2'

normal\_path\_name = 'NORMAL'

normal\_std\_path\_name = 'NORMAL\_STD'

normal\_dwt\_path\_name = 'NORMAL\_DWT'

bacterial\_path\_name = 'BACTERIAL'

bacterial\_std\_path\_name = 'BACTERIAL\_STD'

bacterial\_dwt\_path\_name = 'BACTERIAL\_DWT'

bacterial\_std\_extra\_path\_name = 'BACTERIAL\_STD\_EXTRA'

viral\_path\_name = 'VIRAL'

viral\_std\_path\_name = 'VIRAL\_STD'

viral\_dwt\_path\_name = 'VIRAL\_DWT'

viral\_std\_extra\_path\_name = 'VIRAL\_STD\_EXTRA'

قسمت دوم

#ImageDataGenerator is used to create more images from the existing COVID covid dataset images

import os

import shutil

#import cv2

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from PIL import Image, ImageOps

from config import covid\_std\_path, gen\_path

if \_\_name\_\_ == '\_\_main\_\_':

datagen = ImageDataGenerator(rotation\_range=30, width\_shift\_range=0.3, height\_shift\_range=0.3, shear\_range=0.3, zoom\_range=0.2, horizontal\_flip=True, fill\_mode='constant')

data\_files = os.listdir(covid\_std\_path)

data = np.zeros(shape=(len(data\_files), 299, 299, 3))

data\_index = 0

for filename in data\_files:

image\_path = os.path.join(covid\_std\_path, filename)

img = Image.open(image\_path)

#print(img\_arr.shape)

# if img.mode != 'L':

# img = ImageOps.grayscale(img)

# img = img.resize((299, 299, 3))

img\_arr = np.asarray(img)

#2D to 3D

# img\_arr = np.reshape(img\_arr, newshape=img\_arr.shape+(1,))

data[data\_index] = img\_arr

data\_index = data\_index+1

print(data.shape)

#import random

#for i in range(10):

# print(data[random.randint(0, data.shape[0])])

count = data.shape[0]

i = 0

for batch in datagen.flow(data, batch\_size=150, save\_to\_dir=gen\_path, save\_prefix='covid', save\_format='png'):

i = i+1

print(f"Batch {i} Done")

if (i>=9):

break

print('Done')

قسمت سوم

import os

import math

import pywt

import numpy as np

from PIL import Image, ImageOps

from config import train\_path, test\_path

from config import covid\_path\_name, covid\_std\_path\_name, covid\_dwt\_path\_name, covid\_dwt\_path\_name\_2

from config import normal\_path\_name, normal\_std\_path\_name, normal\_dwt\_path\_name

from config import bacterial\_path\_name, bacterial\_std\_path\_name, bacterial\_dwt\_path\_name

from config import viral\_path\_name, viral\_std\_path\_name, viral\_dwt\_path\_name

#Main function for creating DWT sets from datasets

def gen\_DWTs(root\_path, src\_path, dst\_path):

dir\_path = os.path.join(root\_path, src\_path)

data\_files = os.listdir(dir\_path)

for file in data\_files:

image\_path = os.path.join(root\_path, src\_path, file)

img\_norm\_cA = get\_dwt2(image\_path)

img\_norm\_cA.save(os.path.join(root\_path, dst\_path, 'dmey\_dwt\_' + file))

print(file)

#Creates DWT of single images

def get\_dwt2(image\_path):

img\_bgr = Image.open(image\_path)

print(img\_bgr.size)

np\_bgr = np.asarray(img\_bgr)

#cA = pywt.wavedec2(np\_bgr, 'db2', mode='periodization', level=2)[0]

#Single-Level decomposition

cA, coEffs = pywt.dwt2(np\_bgr, 'dmey', axes=(0, 1))

print(cA.shape)

norm\_cA = normalize(cA)

print(norm\_cA.shape)

norm\_cA = norm\_cA.astype(np.uint8)

#print(norm\_cA)

img\_norm\_cA = Image.fromarray(norm\_cA)

img\_norm\_cA = img\_norm\_cA.resize((224,224))

return img\_norm\_cA

# cH = coEffs[0]

# norm\_cH = normalize(cH)

# norm\_cH = norm\_cH.astype(np.uint8)

# img\_norm\_cH = Image.fromarray(norm\_cH)#.convert('L')

# # return img\_norm\_cH

# cV = coEffs[1]

# norm\_cV = normalize(cV)

# norm\_cV = norm\_cV.astype(np.uint8)

# img\_norm\_cV = Image.fromarray(norm\_cV)#.convert('L')

# cD = coEffs[2]

# norm\_cD = normalize(cD)

# norm\_cD = norm\_cD.astype(np.uint8)

# img\_norm\_cD = Image.fromarray(norm\_cD)#.convert('L')

# img\_hor1 = get\_concat\_h(img\_norm\_cA, img\_norm\_cH)

# img\_hor2 = get\_concat\_h(img\_norm\_cV, img\_norm\_cD)

# img\_ver = get\_concat\_v(img\_hor1, img\_hor2)

# img\_ver.save(os.path.join(train\_path, 'all.png'))

#Normalize images whose pixel values range goes beyond (0, 255)

def normalize(array):

#img = Image.fromarray(img)

#array = np.asarray(img)

max = -math.inf

min = math.inf

for i in range(array.shape[0]):

for j in range(array.shape[1]):

for k in range(array.shape[2]):

if array[i][j][k]>max:

max = array[i][j][k]

if array[i][j][k]<min:

min = array[i][j][k]

print(min, max)

array1 = (array-min)\*255/(max-min)

return array1

def get\_concat\_h(im1, im2):

dst = Image.new('L', (im1.width + im2.width, im1.height))

dst.paste(im1, (0, 0))

dst.paste(im2, (im1.width, 0))

return dst

def get\_concat\_v(im1, im2):

dst = Image.new('L', (im1.width, im1.height + im2.height))

dst.paste(im1, (0, 0))

dst.paste(im2, (0, im1.height))

return dst

if \_\_name\_\_ == '\_\_main\_\_':

gen\_DWTs(train\_path, covid\_std\_path\_name, covid\_dwt\_path\_name)

# gen\_DWTs(test\_path, normal\_std\_path\_name, normal\_dwt\_path\_name)

# gen\_DWTs(test\_path, bacterial\_std\_path\_name, bacterial\_dwt\_path\_name)

# gen\_DWTs(test\_path, viral\_std\_path\_name, viral\_dwt\_path\_name)

#gen\_DWTs(train\_path, normal\_std\_path\_name, normal\_dwt\_path\_name)

#get\_dwt2('E:\\Research\\5. Covid DWT\\Train\\COVID\_STD\\0a7faa2a.png')

قسمت چهارم

#Extensions are checked to assess the composition of the images in the datasets

import os

import shutil

#import cv2

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from PIL import Image, ImageOps

#from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from config import data\_path

if \_\_name\_\_ == '\_\_main\_\_':

data\_files = os.listdir(data\_path)

jpg\_checks, jpeg\_checks, png\_checks = [], [], []

#print(data\_files)

for i in data\_files:

filename, file\_ext = os.path.splitext(i)

if file\_ext=='.jpg':

jpg\_checks.append(i)

elif file\_ext=='.jpeg':

jpeg\_checks.append(i)

elif file\_ext=='.png':

png\_checks.append(i)

else:

print(file\_ext)

print(len(jpg\_checks), len(jpeg\_checks), len(png\_checks))

#Dimension check

min\_height, min\_width = 10000, 10000

avg\_height, avg\_width = 0, 0

count = 0

for filename in data\_files:

image\_path = os.path.join(data\_path, filename)

img = Image.open(image\_path)

#print(img\_arr.shape)

if img.mode != 'L':

img = ImageOps.grayscale(img)

img\_arr = np.asarray(img)

(height, width) = img\_arr.shape

if height<300 or width<300:

print(height, width)

if min\_height>height:

min\_height = height

if min\_width>width:

min\_width = width

avg\_height = avg\_height+height

avg\_width = avg\_width+width

count = count+1

print(min\_height, min\_width)

print(avg\_height/count, avg\_width/count)

قسمت پنجم

#The metadata of the Cohen JP COVID dataset is checked to see the characteristics of the dataset

import os

import shutil

from PIL import Image, ImageOps

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from config import dir\_path, data\_path

os.chdir(dir\_path)

def get\_image(df):

folder = df['folder']

filename = df['filename']

image\_path = os.path.join(folder, filename)

img = Image.open(image\_path)

img.show()

def copy\_image(df):

folder = df['folder']

filename = df['filename']

src\_path = os.path.join(folder, filename)

dst\_path = os.path.join(data\_path, filename)

shutil.copyfile(src=src\_path, dst=dst\_path)

print(src\_path)

def traverse\_df(df):

for i in range(len(df)):

#get\_image(df.iloc[i])

copy\_image(df.iloc[i])

if \_\_name\_\_ == '\_\_main\_\_':

metadata = pd.read\_csv('metadata.csv')

print(len(metadata))

covid\_chest\_pa = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'PA') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

covid\_chest\_ap = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'AP') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

covid\_chest\_ap\_supine = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'AP Supine') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

covid\_chest\_axial = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'Axial') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

covid\_chest\_coronal = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'Coronal') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

covid\_chest\_l = metadata.loc[(metadata['folder'] == 'images') & (metadata['view'] == 'L') & (metadata['finding'] == 'Pneumonia/Viral/COVID-19')]

#Mostly use only PA, AP and AP Supine

#print(len(covid\_chest\_ap), len(covid\_chest\_ap\_supine), len(covid\_chest\_axial), len(covid\_chest\_coronal), len(covid\_chest\_l), len(covid\_chest\_pa))

print(len(covid\_chest\_ap)+len(covid\_chest\_ap\_supine)+len(covid\_chest\_pa))

#Execute the following to copy the AP, AP\_Supine and PA images to the training directory

traverse\_df(covid\_chest\_ap)

traverse\_df(covid\_chest\_ap\_supine)

traverse\_df(covid\_chest\_pa)

قسمت اول تمرین

from PIL import Image, ImageOps

import pywt

print(pywt.wavelist('dmey'))

# img\_bgr = Image.open('E:\\Research\\5. Covid DWT\\Test\\COVID\_STD\\00870a9c.png')

# print(img\_bgr.size)

قسمت ششم

#Split the datasets into the required train and test sets

import os

import shutil

#import cv2

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

#from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from sklearn.model\_selection import train\_test\_split

from config import train\_path, test\_path, covid\_path\_name

main\_read\_path = train\_path

main\_write\_path = train\_path

def splitFileData(arr\_x, arr\_y):

x\_train, x\_test, y\_train, y\_test = train\_test\_split(arr\_x, arr\_y, test\_size=0.40, random\_state=42)

print(len(x\_train))

print(len(x\_test))

return x\_train, x\_test

def writeFileData(path, data):

with open(path, 'w') as f:

for item in data:

f.write("%s\n" % item)

def moveFiles(filenames, src\_path, dst\_path):

index = 0

file = ''

for filename in filenames:

source = os.path.join(src\_path, filename)

shutil.move(source, dst\_path)

file = source

index = index+1

#if index>0:

# break

print(index)

return file

if \_\_name\_\_ == '\_\_main\_\_':

#Get all the COVID data in train directory and split

#data\_files = os.listdir(data\_path)

#data\_y = np.zeros(len(data\_files))

#data\_file\_train, data\_file\_test = splitFileData(data\_files, data\_y)

#Write the split data to separate text files

#writeFileData(file\_train\_path, data\_file\_train)

#writeFileData(file\_test\_path, data\_file\_test)

#Get all COVID data and split

data\_files = os.listdir(os.path.join(train\_path, covid\_path\_name))

data\_y = np.zeros(len(data\_files))

data\_file\_train, data\_file\_test = splitFileData(data\_files, data\_y)

moveFiles(data\_file\_test, os.path.join(train\_path, covid\_path\_name), os.path.join(test\_path, covid\_path\_name))

قسمت ترنسفرم

#Move chunks of dataset as per requirement

import os

import shutil

#import cv2

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

#from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from PIL import Image, ImageOps

from config import train\_path, test\_path

from config import covid\_path\_name, covid\_std\_path\_name

from config import normal\_path\_name, normal\_std\_path\_name

from config import bacterial\_path\_name, bacterial\_std\_path\_name

from config import viral\_path\_name, viral\_std\_path\_name

def normal\_transfer(filenames, read\_path, write\_path):

count = 0

for filename in filenames:

src\_path = os.path.join(read\_path, filename)

dst\_path = os.path.join(write\_path, 'covid\_'+filename)

#img = Image.open(image\_path)

shutil.copyfile(src=src\_path, dst=dst\_path)

count = count+1

return count

def std\_transfer(filenames, read\_path, write\_path):

count = 0

for filename in filenames:

src\_path = os.path.join(read\_path, filename)

#Destination image must be stored in png

new\_filename, file\_extension = os.path.splitext(filename)

new\_filename = new\_filename+'.png'

dst\_path = os.path.join(write\_path, new\_filename)

img = Image.open(src\_path).convert("RGB")

#if img.mode != 'L':

# img = ImageOps.grayscale(img)

#img = img.resize((224, 224))

print(img.size, img.mode)

img.save(dst\_path, "PNG")

count = count+1

print(dst\_path, img.size, img.mode)

return count

if \_\_name\_\_ == '\_\_main\_\_':

#COVID to Std Directory

covid\_files = os.listdir(os.path.join(test\_path, covid\_path\_name))

result = std\_transfer(covid\_files, os.path.join(test\_path, covid\_path\_name), os.path.join(test\_path, covid\_std\_path\_name))

print(result)

#Normal to Std Directory

normal\_files = os.listdir(os.path.join(test\_path, normal\_path\_name))

result = std\_transfer(normal\_files, os.path.join(test\_path, normal\_path\_name), os.path.join(test\_path, normal\_std\_path\_name))

print(result)

#Bacterial to Std Directory

bacterial\_files = os.listdir(os.path.join(test\_path, bacterial\_path\_name))

result = std\_transfer(bacterial\_files, os.path.join(test\_path, bacterial\_path\_name), os.path.join(test\_path, bacterial\_std\_path\_name))

print(result)

#Viral to Std Directory

viral\_files = os.listdir(os.path.join(test\_path, viral\_path\_name))

result = std\_transfer(viral\_files, os.path.join(test\_path, viral\_path\_name), os.path.join(test\_path, viral\_std\_path\_name))

print(result)