import matplotlib

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk

from matplotlib.figure import Figure

import cv2

import tkinter as tk

#from utils import \*

#from models import \*

from tkinter import \*

import matplotlib.pyplot as plt

import numpy as np

global canvas

global canvas2

global Textbox

global Textbox2

global i

path = ""

numpyimage = 0

i=0

canvas = 0

canvas2 = 0

Textbox=0

Textbox2=0

slicenum =20

indexnodule = 0

indexslice = 20

nodules=[]

malig = []

import SimpleITK as sitk

import numpy as np

def load\_itk\_image(filename):

itkimage = sitk.ReadImage(filename)

numpyImage = sitk.GetArrayFromImage(itkimage)

numpyOrigin = np.array(list(reversed(itkimage.GetOrigin())))

numpySpacing = np.array(list(reversed(itkimage.GetSpacing())))

return numpyImage, numpyOrigin, numpySpacing

def readCSV(filename):

lines = []

with open(filename, "rb") as f:

csvreader = csv.reader(f)

for line in csvreader:

lines.append(line)

return lines

def worldToVoxelCoord(worldCoord, origin, spacing):

stretchedVoxelCoord = np.absolute(worldCoord - origin)

voxelCoord = stretchedVoxelCoord / spacing

return voxelCoord

def normalizePlanes(npzarray):

maxHU = 400.

minHU = -1000.

npzarray = (npzarray - minHU) / (maxHU - minHU)

npzarray[npzarray>1] = 1.

npzarray[npzarray<0] = 0.

return npzarray

def getcrops(numpyImage):

frame=64

crops = []

for d in range(int(numpyImage.shape[0]/64)):

for r in range(int(numpyImage.shape[1]/64)):

for c in range(int(numpyImage.shape[2]/64)):

dd = d \*frame

rr= r\*frame

cc= c\*frame

crop = numpyImage[dd:dd+frame,rr:rr+frame,cc:cc+frame]

crops.append(crop)

for r in range(int(numpyImage.shape[1]/64)):

for c in range(int(numpyImage.shape[2]/64)):

dd = d \*frame

rr= r\*frame

cc= c\*frame

crop = numpyImage[-64:,rr:rr+frame,cc:cc+frame]

crops.append(crop)

return crops

import tensorflow as tf

import keras as k

from keras.models import Sequential ,Model

from keras.layers import Conv2D, MaxPooling2D,AveragePooling3D,AveragePooling2D,MaxPooling3D,Conv3D

from keras.layers import Activation, Dropout, Flatten, Dense ,Input

from keras.callbacks import ModelCheckpoint, LearningRateScheduler

from keras.preprocessing import image

from keras import applications

def inceptionlayer(prev):

tower\_1 = Conv3D(64, (1,1,1), padding='same', activation='relu')(prev)

tower\_1 = Conv3D(64, (3,3,3), padding='same', activation='relu')(tower\_1)

tower\_2 = Conv3D(64, (1,1,1), padding='same', activation='relu')(prev)

tower\_2 = Conv3D(64, (5,5,5), padding='same', activation='relu')(tower\_2)

tower\_3 = MaxPooling3D((3,3,3), strides=(1,1,1), padding='same')(prev)

tower\_3 = Conv3D(64, (1,1,1), padding='same', activation='relu')(tower\_3)

output = k.layers.concatenate([tower\_1, tower\_2, tower\_3], axis = 3)

return output

#GOOGLENET

def makegooglenet():

x = Input(shape=(64,64,64,1))

conv1= Conv3D(32,kernel\_size=(7,7,7),activation='relu')(x)

conv1= Dropout(0.3)(conv1)

max1 = MaxPooling3D(pool\_size=(2,2,2))(conv1)

conv2= Conv3D(32,kernel\_size=(3,3,3),activation='relu')(max1)

conv2= Dropout(0.3)(conv2)

max2 = MaxPooling3D(pool\_size=(2,2,2))(conv2)

incp1= inceptionlayer(max2)

incp1= Dropout(0.3)(incp1)

incp2= inceptionlayer(incp1)

incp2= Dropout(0.3)(incp2)

max3 = MaxPooling3D(pool\_size=(2,2,2))(incp2)

incp3= inceptionlayer(max3)

incp3= Dropout(0.3)(incp3)

incp4= inceptionlayer(incp3)

incp4= Dropout(0.3)(incp4)

max4 = MaxPooling3D(pool\_size=(2,2,2))(incp4)

incp5= inceptionlayer(max4)

incp5= Dropout(0.3)(incp5)

incp6= inceptionlayer(incp5)

incp6= Dropout(0.3)(incp6)

avg1= AveragePooling3D(pool\_size=(2,2,2))(incp4)

flat = Flatten()(avg1)

flat= Dropout(0.3)(flat)

dense= Dense(2,activation="softmax")(flat)

googlenet = Model(inputs=x, outputs=dense)

googlenet.compile(loss=k.losses.categorical\_crossentropy,

optimizer=k.optimizers.Adam(lr=0.001),

metrics=['accuracy'])

googlenet.load\_weights('C:\\Users\\hpr\\Downloads\\weights3d\_googlenet.hdf5')

return googlenet

def makelenet():

model = Sequential()

model.add(Conv2D(32, kernel\_size=(5, 5), strides=(1, 1),

activation='relu',

input\_shape=[64,64,64]))

model.add(Dropout(0.4))

model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(64, (5, 5), activation='relu'))

model.add(Dropout(0.4))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dropout(0.4))

model.add(Dense(1000, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(6, activation='softmax'))

model.compile(loss=k.losses.categorical\_crossentropy,

optimizer=k.optimizers.SGD(lr=0.01),

metrics=['accuracy'])

model\_checkpoint = ModelCheckpoint('C:\\Users\\hpr\\Downloads\\malignancy\_crop.hdf5', monitor='loss', save\_best\_only=True)

model.load\_weights("C:\\Users\\hpr\\Downloads\\malignancy\_crop.hdf5")

return model

googlenet= makegooglenet()

lenet = makelenet()

def getnodules(crops,googlenet):

count1=0

count0=0

global nodules

for crop in crops:

cc = np.expand\_dims(crop,0)

cc = np.expand\_dims(cc,-1)

if np.argmax(googlenet.predict(cc))==1:

nodules.append(crop)

count1= count1+1

else:

count0=count0+1

return nodules

def next():

global i

global canvas

if i< numpyimage.shape[0]:

i= i+1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(numpyimage[i],cmap='gray')

canvas = FigureCanvasTkAgg(f,gui,)

canvas.get\_tk\_widget().grid(row=1,column=0)

def prev():

global i

global canvas

if i >0:

i= i-1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(numpyimage[i],cmap='gray')

canvas = FigureCanvasTkAgg(f,gui,)

canvas.get\_tk\_widget().grid(row=1,column=0)

def nextnod():

global indexnodule

global canvas2

global slicenum

if indexnodule< len(nodules)-1:

indexnodule= indexnodule+1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(nodules[indexnodule][slicenum],cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

Textbox = tk.Text(gui, height=2, width=20)

Textbox.grid(row=0,column=2)

Textbox.insert(tk.END, "malignancy : "+ str(malig[indexnodule]))

def prevnod():

global indexnodule

global canvas2

global slicenum

if indexnodule>0:

indexnodule= indexnodule-1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(nodules[indexnodule][slicenum],cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

Textbox = tk.Text(gui, height=2, width=20)

Textbox.grid(row=0,column=2)

Textbox.insert(tk.END, "malignancy : "+ str(malig[indexnodule]))

def nextslice():

global indexnodule

global indexslice

global canvas2

if indexslice< 63:

indexslice= indexslice+1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(nodules[indexnodule][indexslice],cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

def prevslice():

global indexnodule

global indexslice

global canvas2

if indexslice>0:

indexslice= indexslice-1

f = Figure(figsize=(4,4), dpi=100)

a= f.add\_subplot(1,1,1)

a.imshow(nodules[indexnodule][indexslice],cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

def browsefunc():

filename = filedialog.askopenfilename()

global path

global numpyimage

path = filename

numpyimage,b,x = load\_itk\_image(path)

f = Figure(figsize=(40,40), dpi=10)

a= f.add\_subplot(1,1,1)

a.imshow(numpyimage[0] ,cmap='gray')

canvas = FigureCanvasTkAgg(f,gui,)

canvas.get\_tk\_widget().grid(row=1,column=0)

def analyze():

global numpyimage

global googlenet

global nodules

global indexslice

global malig

nodules =0

crops = getcrops(numpyimage)

nodules = getnodules(crops,googlenet)

for nodule in nodules:

cc = np.expand\_dims(nodule,0)

mnum = np.argmax(lenet.predict(cc))

malig.append(mnum)

f = Figure(figsize=(40,40), dpi=10)

a= f.add\_subplot(1,1,1)

a.imshow(nodules[0][indexslice] ,cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

Textbox = tk.Text(gui, height=2, width=20)

Textbox.grid(row=0,column=2)

Textbox.insert(tk.END, "malignancy : "+ str(malig[indexnodule]))

Textbox2 = tk.Text(gui, height=2, width=20)

Textbox2.grid(row=2,column=2)

Textbox2.insert(tk.END, "nodule count : "+ str(len(nodules)))

gui = tk.Tk()

gui.minsize(800,500)

gui.configure(background='gray')

f = Figure(figsize=(40,40), dpi=10)

a= f.add\_subplot(1,1,1)

a.imshow(np.zeros((5,5)) ,cmap='gray')

canvas = FigureCanvasTkAgg(f,gui,)

canvas.get\_tk\_widget().grid(row=1,column=0)

Textbox = tk.Text(gui, height=2, width=20)

Textbox.grid(row=0,column=2)

Textbox.insert(tk.END, "malignancy : ")

Textbox2 = tk.Text(gui, height=2, width=20)

Textbox2.grid(row=2,column=2)

Textbox2.insert(tk.END, "nodule count : ")

f = Figure(figsize=(40,40), dpi=10)

a= f.add\_subplot(1,1,1)

a.imshow(np.zeros((5,5)) ,cmap='gray')

canvas2 = FigureCanvasTkAgg(f,gui,)

canvas2.get\_tk\_widget().grid(row=1,column=1)

bbrowse = tk.Button(gui , text="Browse" ,command=browsefunc )

bbrowse.grid(row=0,column=0)

anayliz = tk.Button(gui , text="Analyze" ,command=analyze )

anayliz.grid(row=0,column=1)

bnextnod = tk.Button(gui , text="Next Nodule" ,command=nextnod )

bnextnod.grid(row=2,column=1)

bprevnod = tk.Button(gui , text="Previous Nodule" ,command=prevnod )

bprevnod.grid(row=3,column=1)

bnextslice = tk.Button(gui , text="Next slice" ,command=nextslice )

bnextslice.grid(row=1,column=2)

bprevslice = tk.Button(gui , text="Previous slice" ,command=prevslice )

bprevslice.grid(row=1,column=3)

bnext = tk.Button(gui , text="Next" ,command=next )

bnext.grid(row=2,column=0)

bprev = tk.Button(gui , text="Previous" ,command=prev )

bprev.grid(row=3,column=0)

gui.mainloop()