**TRAINING CNN AND LSTM**

from fastai import \*

from fastai.vision import \*

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import auc,roc\_curve import os

print (os.listdir("/content/drive/My Drive/Colab Notebooks"))

% Matplotlib inline

class\_names = {1: "Tumor", 2: "Stroma", 3: "Complex", 4: "Lympho",

5: "Debris", 6: "Mucosa", 7: "Adipose", 8: "Empty"} class\_numbers = {"Tumor": 1, "Stroma": 2, "Complex": 3, "Lympho": 4,

"Debris": 5, "Mucosa": 6, "Adipose": 7, "Empty": 8} class\_colors = {1: "Red", 2: "Orange", 3: "Gold", 4: "Limegreen",

5: "Mediumseagreen", 6: "Darkturquoise", 7: "Steelblue", 8: "Purple"}

label\_percentage = df.label.value\_counts() / df.shape[0]

class\_index = [class\_names[idx] for idx in label\_percentage.index.values] plt.figure(figsize=(20,5))

sns.barplot(x=class\_index, y=label\_percentage.values, palette="Set3"); plt.ylabel("% in data");

plt.xlabel("Target cancer class");

plt.title("How is cancer distributed in this data?"); tfms=get\_transforms(flip\_vert=True, max\_warp=0.)

tfms=get\_transforms(flip\_vert=True, max\_warp=0.) data = (ImageList.from\_folder(path)

.split\_by\_rand\_pct()

.label\_from\_folder() .transform(tfms, size=150) .databunch(num\_workers=2, bs=32))

learner= cnn\_learner(data, models.resnet34, metrics=[accuracy], model\_dir='/content/drive/My Drive/Colab Notebooks')

# Train the model on 4 epochs of data at the default learning rate #learner.fit\_one\_cycle(4)

## Fit the model over 8 epochs

lr=5e-3 ## uncomment this

learner.fit\_one\_cycle(8, lr) ## uncomment this

#save the model

learner.save('/content/drive/My Drive/Colab Notebooks/level-1') #print(os.listdir("./drive/My Drive/Colab Notebooks"))

#load the model

#learner.load('level-1')

#save the model

learner.save('level-2') ## uncomment this

#load the model

#learner.load('level-1')

# intrepting most confused

interp.most\_confused()

# ROC curve

fpr, tpr, thresholds = roc\_curve(lb.numpy(), preds.numpy()[:,1], pos\_label=1) # ROC area

pred\_score = auc(fpr, tpr)

print(f'ROC area is {pred\_score}')

plt.figure()

plt.plot(fpr, tpr, color='green', label='ROC curve (area = %0.2f)' % pred\_score) plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.xlim([-0.01, 1.0])

plt.ylim([0.0, 1.01])

plt.xlabel('False\_Positive\_Rate')

plt.ylabel('True\_Positive\_Rate')

plt.title('Receiver\_Operating\_Characteristic')

plt.legend(loc="lower right")

####/\*\*\*\*\*Testing and prediction (load level-2)\*\*\*\*\*/### #learner.load("level-2")

learner.load("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/old\_level-1") ####/\*\*\*\*\*Testing and prediction \*\*\*\*\*/### #learner.load("level-2")

# lets save our model with two formats: pkl and pth #learner.export('pkl\_colorectal\_CNN\_model.pkl') #learner.save('pth\_colorectal\_CNN\_model') imageC1=random.choice(os.listdir("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/04\_LYMPHO/"))

#read = cv2.imread("/content/drive/My Drive/Colab Notebooks/test.jpeg") #test\_image=cv2.imwrite("/content/drive/My Drive/Colab Notebooks/test\_tif.tif",read)

**EXPORT MODEL AND PERFORM UNIT TESTING**

print(imageC1)

# test case 1:

#159A9\_CRC-Prim-HE-07\_022.tif\_Row\_901\_Col\_151.tif ; 1EAE\_CRC-Prim-HE- 10\_029.tif\_Row\_1\_Col\_451.tif [tumor or debris]

#4B46\_CRC-Prim-HE-07.tif\_Row\_301\_Col\_601.tif [tumor debris adipose] #test\_image=plt.imread("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/01\_TUMOR/"+imageC1) # test case 2:

test\_image=plt.imread("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/04\_LYMPHO/"+imageC1)

#test case 3:

#test\_image=plt.imread("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/03\_COMPLEX/17D73\_CRC-Prim- HE-01\_034.tif\_Row\_451\_Col\_301.tif")

#print(os.listdir("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/02\_STROMA")) #test\_image=plt.imread("/content/drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000/02\_STROMA/11385\_CRC-Prim- HE-06\_003.tif\_Row\_601\_Col\_151.tif")

plt.imshow(test\_image)

file\_name=[]

predictions=[]

from PIL import Image as PImage

import cv2

#from fastai.vision import \*

#-----check for lympho---

lympholist=os.listdir("/content/drive/My Drive/Colab Notebooks/test\_images/") print(len(lympholist))

for i in range(0,len(lympholist)):

if(lympholist[i].endswith(".jpeg")): test\_image=plt.imread("/content/drive/My Drive/Colab Notebooks/test\_images/"+lympholist[i])

#------check end for lympho-----

frame = cv2.cvtColor(test\_image,cv2.COLOR\_BGR2RGB) pil\_im = PImage.fromarray(frame)

x = pil2tensor(pil\_im ,np.float32)

preds\_num = learner.predict(Image(x))[2].numpy()

#print(preds\_num)

if True:# preds\_num[4]!=0 and preds\_num[4]==max([preds\_num[0],preds\_num[1],preds\_num[2],preds\_num[3],preds\_n um[4],preds\_num[5],preds\_num[6],preds\_num[7]]) :

#print(l,"\n",class\_names) file\_name.append(lympholist[i]) predictions.append(preds\_num) #print(lympholist[i]) #print(preds\_num)

#break

#from sklearn.externals import joblib #print(os.listdir("./drive/My Drive/Colab

Notebooks/Kather\_texture\_2016\_image\_tiles\_5000"))

#classifer = joblib.load("./drive/My Drive/Colab Notebooks/Kather\_texture\_2016\_image\_tiles\_5000//drive/My Drive/Colab Notebooks/pkl\_colorectal\_CNN\_model.pkl")

for fn,pre in zip(file\_name,predictions):

print(fn,"\t",pre)

count=0

pridict=[]

for i in range(len(preds\_num)):

if(preds\_num[i]!=0): count+=1

#print(class\_names[i+1]," ---> ",preds\_num[i])

38

pridict.append(class\_names[i+1]) #print(class\_names[i+1],"\t")

print("According to our dataset the scan matches with:\n"," and ".join(pridict),"type of colorectal cancer")

**IMPLEMENT MODEL TO PREDICT OVER WEB**

from flask import Flask,render\_template,request,flash,url\_for,redirect from werkzeug.utils import secure\_filename

import json

import random

import tablib l=learner.load\_learner("./models/level1.pth") app=Flask(\_name\_)

app.secret\_key = 'h432hi5ohi3h5i5hi3o2hi' #create a route

@app.route('/')

def home():

return render\_template('index.html') @app.route('/prediction',methods=['GET','POST']) def result():

if request.method == 'POST': #flash(" ".join(request.form.keys())) f=request.form['img\_file'].split("/") #-----------------------------------# #result=jsonify(l.predict(f)) # #json.dump(result,"testfile.json") #

#-----------------------------------# with open("testfile.json") as jfile: dicl=json.load(jfile) ifile=f[len(f)-1]

if ifile in dicl.keys(): result=dicl[ifile] furl="/test\_images/"+f[len(f)-1] stage="Can't identify"

if 'Mucosa' in result: stage="S0"

if 'Lympho' in result:

stage="S1" elif 'Debris' in result:

stage="S2"

if 'Stroma' in result or 'Complex' in result: stage="S3I"

elif 'Lympho' in result:

stage="S3"

if 'Stroma' in result or 'Complex' in result:

stage="S3I" #print(cell)

#print(result1) return

render\_template('index.html',isindex=True,imagef=str(url\_for("static",filename=furl)),re sult=result,stage=stage)

else:

return redirect(url\_for('home'))

@app.route('/model') def model():

return render\_template('model.html')

**PIP lock file:**

{

"\_meta": {

"hash": {

"sha256": "d5e270fef618e43f481bda408d839a4499afea70db57b11d01d3c414e4b94b4f"

},

"pipfile-spec": 6, "requires": { "python\_version": "3.7" },

"sources": [

{

"name": "pypi",

"url": "https://pypi.org/simple", "verify\_ssl": true

}

]

},

"default": { "beautifulsoup4": { "hashes": [

"sha256:4c98143716ef1cb40bf7f39a8e3eec8f8b009509e74904ba3a7b315431577 e35",

"sha256:84729e322ad1d5b4d25f805bfa05b902dd96450f43842c4e99067d5e1369e b25",

"sha256:fff47e031e34ec82bf17e00da8f592fe7de69aeea38be00523c04623c04fb66 6"

],

"index": "pypi", "version": "==4.9.3" },

"click": {

"hashes": [

"sha256:d2b5255c7c6349bc1bd1e59e08cd12acbbd63ce649f2588755783aa94dfb6 b1a",

"sha256:dacca89f4bfadd5de3d7489b7c8a566eee0d3676333fbb50030263894c38c 0dc"

],

"markers": "python\_version >= '2.7' and python\_version not in '3.0, 3.1, 3.2, 3.3, 3.4'",

"version": "==7.1.2" },

"flask": {

"hashes": [

"sha256:4efa1ae2d7c9865af48986de8aeb8504bf32c7f3d6fdc9353d34b21f4b1270 60",

"sha256:8a4fdd8936eba2512e9c85df320a37e694c93945b33ef33c89946a340a238 557"

],

"index": "pypi", "version": "==1.1.2" },

"itsdangerous": { "hashes": [

"sha256:321b033d07f2a4136d3ec762eac9f16a10ccd60f53c0c91af90217ace7ba1f 19",

"sha256:b12271b2047cb23eeb98c8b5622e2e5c5e9abd9784a153e9d8ef9cb4dd09d749" ],

"markers": "python\_version >= '2.7' and python\_version not in '3.0, 3.1, 3.2, 3.3'", "version": "==1.1.0"

},

"jinja2": {

"hashes": [

"sha256:03e47ad063331dd6a3f04a43eddca8a966a26ba0c5b7207a9a9e4e08f1b29 419",

"sha256:a6d58433de0ae800347cab1fa3043cebbabe8baa9d29e668f1c768cb87a33 3c6"

],

"markers": "python\_version >= '2.7' and python\_version not in '3.0, 3.1, 3.2, 3.3, 3.4'", "version": "==2.11.3"

}, "markupsafe": { "hashes": [

"sha256:cd5df75523866410809ca100dc9681e301e3c27567cf498077e8551b6d20e 42f",

"sha256:3b8a6499709d29c2e2399569d96719a1b21dcd94410a586a18526b143ec8 470f",

"sha256:9add70b36c5666a2ed02b43b335fe19002ee5235efd4b8a89bfcf9005bebac 0d",

"sha256:b1282f8c00509d99fef04d8ba936b156d419be841854fe901d8ae224c59f0 be5",

"sha256:ade5e387d2ad0d7ebf59146cc00c8044acbd863725f887353a10df825fc8ae 21",

"sha256:b2051432115498d3562c084a49bba65d97cf251f5a331c64a12ee7e04dacc 51b",

"sha256:09c4b7f37d6c648cb13f9230d847adf22f8171b1ccc4d5682398e77f40309 235",

"sha256:b1dba4527182c95a0db8b6060cc98ac49b9e2f5e64320e2b56e47cb283197 8c7",

"sha256:596510de112c685489095da617b5bcbbac7dd6384aeebeda4df6025d0256a 81b",

"sha256:e8313f01ba26fbbe36c7be1966a7b74249…