**Binary Classification (VGG-19)**

!pip install numpy

!pip install pandas

!pip install matplotlib

!pip install seaborn

!pip install opencv-python

!pip install scikit-learn

!pip install tensorflow==2.15.0 –user

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import cv2

import glob

import random

from os import listdir

from sklearn.metrics import classification\_report

import tensorflow as tf

import keras.utils as image

# Breast Histopathology Images

# 198,738 IDC(-) image patches; 78,786 IDC(+) image patches

# Invasive Ductal Carcinoma (IDC)

# <https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images>

breast\_img = glob.glob('C:/Users/DELL/Desktop/breast cancer classification CNN/breast-histopathology-images/\*\*/\*.png', recursive = True)

for imgname in breast\_img[:3]:

print(imgname)

N\_IDC = []

P\_IDC = []

for img in breast\_img:

if img[-5] == '0' :

N\_IDC.append(img)

elif img[-5] == '1' :

P\_IDC.append(img)

plt.figure(figsize = (15, 15))

some\_non = np.random.randint(0, len(N\_IDC), 18)

some\_can = np.random.randint(0, len(P\_IDC), 18)

s = 0

for num in some\_non:

img = image.load\_img((N\_IDC[num]), target\_size=(100, 100))

img = image.img\_to\_array(img)

plt.subplot(6, 6, 2\*s+1)

plt.axis('off')

plt.title('no cancer')

plt.imshow(img.astype('uint8'))

s += 1

s = 1

for num in some\_can:

img = image.load\_img((P\_IDC[num]), target\_size=(100, 100))

img = image.img\_to\_array(img)

plt.subplot(6, 6, 2\*s)

plt.axis('off')

plt.title('cancer')

plt.imshow(img.astype('uint8'))

s += 1

NewN\_IDC=N\_IDC[:78786]

print(len(NewN\_IDC))

print(len(P\_IDC))

import cv2

non\_img\_arr = []

can\_img\_arr = []

for img\_path in NewN\_IDC:

n\_img = cv2.imread(img\_path, cv2.IMREAD\_COLOR)

if n\_img is not None:

n\_img\_size = cv2.resize(n\_img, (50, 50), interpolation=cv2.INTER\_LINEAR)

non\_img\_arr.append([n\_img\_size, 0])

for img\_path in P\_IDC:

c\_img = cv2.imread(img\_path, cv2.IMREAD\_COLOR)

if c\_img is not None:

c\_img\_size = cv2.resize(c\_img, (50, 50), interpolation=cv2.INTER\_LINEAR)

can\_img\_arr.append([c\_img\_size, 1])

print(len(non\_img\_arr))

print(len(can\_img\_arr))

print(len(non\_img\_arr)+len(can\_img\_arr))

X = []

y = []

breast\_img\_arr = non\_img\_arr[:12389] + can\_img\_arr[:12389]

random.shuffle(breast\_img\_arr)

for feature, label in breast\_img\_arr:

X.append(feature)

y.append(label)

X = np.array(X)

y = np.array(y)

def describeData(a,b):

print('Total number of images: {}'.format(len(a)))

print('Number of IDC(-) Images: {}'.format(np.sum(b==0)))

print('Number of IDC(+) Images: {}'.format(np.sum(b==1)))

print('Image shape (Width, Height, Channels): {}'.format(a[0].shape))

describeData(X,y)

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=0.3)

from tensorflow.keras.utils import to\_categorical

Y\_train = to\_categorical(Y\_train, num\_classes = 2)

Y\_test = to\_categorical(Y\_test, num\_classes = 2)

print("Training Data Shape:", X\_train.shape)

print("Testing Data Shape:", X\_test.shape)

from sklearn.metrics import classification\_report

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D,MaxPooling2D, Flatten, Dropout, BatchNormalization

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.optimizers import Adam, SGD

from keras.metrics import binary\_crossentropy

from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import confusion\_matrix

import itertools

from tensorflow.keras.applications.vgg19 import VGG19

from keras.models import Model

vgg\_model=VGG19(input\_shape=(50,50,3),include\_top=False)

x=Flatten()(vgg\_model.output)

prediction=Dense(2,activation='softmax')(x)

model=Model(inputs=vgg\_model.input,outputs=prediction)

model.summary()

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X\_train, Y\_train, validation\_data = (X\_test, Y\_test), epochs = 40, batch\_size = 500)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model Accuracy')

plt.xlabel('epoch')

plt.ylabel('accuracy')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model Loss')

plt.xlabel('epoch')

plt.ylabel('loss')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

from sklearn.metrics import accuracy\_score

Y\_pred = model.predict(X\_test)

Y\_pred\_classes = np.argmax(Y\_pred,axis = 1)

Y\_true = np.argmax(Y\_test,axis = 1)

#accuracy=accuracy\_score(y\_true=Y\_true, y\_pred=Y\_pred)

#print(accuracy)

confusion\_mtx = confusion\_matrix(Y\_true, Y\_pred\_classes)

f,ax = plt.subplots(figsize=(8,5))

sns.heatmap(confusion\_mtx, annot=True, linewidths=0.01,cmap="BuPu",linecolor="gray", fmt= '.1f',ax=ax)

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

# Assuming Y\_true and Y\_pred are already defined

Y\_pred\_classes = np.argmax(Y\_pred, axis=1)

Y\_true\_classes = np.argmax(Y\_test, axis=1)

# Calculate the metrics

accuracy = accuracy\_score(Y\_true\_classes, Y\_pred\_classes)

precision = precision\_score(Y\_true\_classes, Y\_pred\_classes,average='weighted')

recall = recall\_score(Y\_true\_classes, Y\_pred\_classes, average='weighted')

f1 = f1\_score(Y\_true\_classes, Y\_pred\_classes, average='weighted')

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

print("F1 Score:", f1)

# Plotting the bar chart

metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score']

values = [accuracy, precision, recall, f1]

plt.figure(figsize=(10, 6))

bars = plt.bar(metrics, values, color=['blue', 'green', 'orange', 'red'])

# Adding the value labels on top of each bar

for bar in bars:

yval = bar.get\_height()

plt.text(bar.get\_x() + bar.get\_width()/2, yval + 0.01, round(yval, 5), ha='center', va='bottom')

plt.ylim(0, 1.1) # Set y-axis limit to [0, 1] for better visualization

plt.ylabel('Score')

plt.title('Classification Metrics')

plt.show()

model.evaluate(X\_test,Y\_test)

def img\_plot(arr,index=0):

plt.figure(figsize = (2, 15))

plt.title('Test Image')

plt.imshow(arr[index])

index = 1000

img\_plot(X\_test, index)

def img\_plot(arr,index=0):

plt.title('Test Image')

plt.imshow(arr[index])

index = 1000

input = X\_test[index:index+1]

pred = model.predict(input)[0].argmax()

label = Y\_test[index].argmax()

print('Predicted Value using cnn model',pred)

print("True Value",label)

model.save('C:/Users/HP/Desktop/breast cancer classification CNN/modelvgg19.h5')

from sklearn.metrics import roc\_curve, auc

import matplotlib.pyplot as plt

# Assuming Y\_true and Y\_pred are already defined for a binary classification problem

fpr, tpr, \_ = roc\_curve(Y\_true\_classes, Y\_pred\_classes)

roc\_auc = auc(fpr, tpr)

# Plotting the ROC curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc="lower right")

plt.show()

from tensorflow.keras.applications import VGG19

model = VGG19(weights='imagenet', include\_top=True)

model.save('C:/Users/HP/Desktop/breast cancer classification CNN/modelvgg19.h5')

import tensorflow as tf

from PIL import ImageFont

import visualkeras

model = tf.keras.models.load\_model("C:/Users/HP/Desktop/breast cancer classification CNN/modelvgg19.h5")

font = ImageFont.truetype("arial.ttf", 32) # using comic sans is strictly prohibited!

visualkeras.layered\_view(model)

import tensorflow as tf

from PIL import ImageFont

import visualkeras

model = tf.keras.models.load\_model("C:/Users/HP/Desktop/breast cancer classification CNN/modelvgg19.h5")

font = ImageFont.truetype("arial.ttf", 32) # using comic sans is strictly prohibited!

visualkeras.layered\_view(model, legend=True, font=font) # font is optional!

visualkeras.layered\_view(model, draw\_volume=False)

!pip install pydot==1.4.2

!pip install graphviz==0.17

from keras.utils import plot\_model

import os

os.environ["PATH"] += os.pathsep + 'C:/Users/HP/Downloads/windows\_10\_cmake\_Release\_Graphviz-10.0.1-win64/Graphviz-10.0.1-win64/bin' # Replace with your actual path

model = tf.keras.models.load\_model("C:/Users/HP/Desktop/breast cancer classification CNN/modelvgg19.h5")

plot\_model(model, to\_file='model\_plot.png', show\_shapes=True, show\_layer\_names=True)