In [1]:

```
import os
import shutil

INPUT_DATASET = "datasets/original"
BASE_PATH = "datasets/idc"
TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])
TRAIN_SPLIT = 0.8
VAL_SPLIT = 0.1
from imutils import paths
import random

originalPaths = list(paths.list_images(INPUT_DATASET))
random.seed(7)
random.shuffle(originalPaths)
index = int(len(originalPaths) * TRAIN_SPLIT)
trainPaths = originalPaths[:index]
testPaths = originalPaths[index:]
index = int(len(trainPaths) * VAL_SPLIT)
valPaths = trainPaths[:index]
trainPaths = trainPaths[index:]
datasets = [("training", trainPaths, TRAIN_PATH),
            ("validation", valPaths, VAL_PATH),
            ("testing", testPaths, TEST_PATH)
            ]
for (setType, originalPaths, basePath) in datasets:
    print(f'Building {setType} set')
    if not os.path.exists(basePath):
        print(f'Building directory {basePath}')
        os.makedirs(basePath)
    for path in originalPaths:
        file = path.split(os.path.sep)[-1]
        label = file[-5:- 4]
        labelPath = os.path.sep.join([basePath, label])
        if not os.path.exists(labelPath):
            print(f'Building directory {labelPath}')
            os.makedirs(labelPath)
        newPath = os.path.sep.join([labelPath, file])
        shutil.copy2(path, newPath)
import tensorflow as tf
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K


class CancerNet:
    @staticmethod
    def build(width, height, depth, classes):
        model = keras.models.Sequential()
        shape = (height, width, depth)
        channelDim = -1
        if K.image_data_format() == "channels_first":
            input_shape = (depth, height, width)
            channelDim = 1
        model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation
        model.add(tf.keras.layers.BatchNormalization(axis=channelDim))
        model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2
        model.add(tf.keras.layers.Dropout(0.25))

        model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation
        model.add(tf.keras.layers.BatchNormalization(axis=channelDim))
        model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation
        model.add(tf.keras.layers.BatchNormalization(axis=channelDim))
        model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2
        model.add(tf.keras.layers.Dropout(0.25))

        model.add(tf.keras.layers.Flatten())
        model.add(tf.keras.layers.Dense(units=256, activation='relu'))
        model.add(tf.keras.layers.BatchNormalization(axis=channelDim))
        model.add(tf.keras.layers.Dropout(0.5))

        model.add(tf.keras.layers.Dense(units=classes, activation='softmax'))

        return model


import matplotlib

matplotlib.use("Agg")

train_datagen = ImageDataGenerator(rescale=1. / 255, shear_range=0.2, zoom_ran
training_set = train_datagen.flow_from_directory('datasets/idc/training', targ
                                                 class_mode='binary')
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.utils import to_categorical
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

NUM_EPOCHS = 2
INIT_LR = 1e-2
BS = 32

trainPaths = list(paths.list_images(TRAIN_PATH))
lenTrain = len(trainPaths)
lenVal = len(list(paths.list_images(VAL_PATH)))
lenTest = len(list(paths.list_images(TEST_PATH)))

trainLabels = [int(p.split(os.path.sep)[-2]) for p in trainPaths]
trainLabels = to_categorical(trainLabels)
classTotals = trainLabels.sum(axis=0)
classWeight = classTotals.max() / classTotals

trainAug = ImageDataGenerator(rescale=1 / 255.0, rotation_range=20, zoom_range
                              height_shift_range=0.1, shear_range=0.05, horizo
                              fill_mode="nearest")

valAug = ImageDataGenerator(rescale=1 / 255.0)
```

```
trainGen = trainAug.flow_from_directory(TRAIN_PATH, class_mode="categorical",
                                        shuffle=True, batch_size=BS)
valGen = valAug.flow_from_directory(
    VAL_PATH,
    class_mode="categorical",
    target_size=(48, 48),
    color_mode="rgb",
    shuffle=False,
    batch_size=BS)
testGen = valAug.flow_from_directory(
    TEST_PATH,
    class_mode="categorical",
    target_size=(48, 48),
    color_mode="rgb",
    shuffle=False,
    batch_size=BS)
model = CancerNet.build(width=48, height=48, depth=3, classes=2)
model.compile(loss="binary_crossentropy", optimizer='adam', metrics=["accuracy
M = model.fit(x=trainGen, validation_data=valGen, epochs=2)

print("Now evaluating the model")
testGen.reset()
pred_indices = model.predict_generator(testGen, steps=(lenTest // BS) + 1)

pred_indices = np.argmax(pred_indices, axis=1)

print(classification_report(testGen.classes, pred_indices, target_names=testGe

cm = confusion_matrix(testGen.classes, pred_indices)
total = sum(sum(cm))
accuracy = (cm[0, 0] + cm[1, 1]) / total
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
print(cm)
print(f'Accuracy: {accuracy}')
print(f'Specificity: {specificity}')
print(f'Sensitivity: {sensitivity}')

N = NUM_EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), M.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), M.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), M.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), M.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on the IDC Dataset")
plt.xlabel("Epoch No.")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('plot.png')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 11
      9 TRAIN_SPLIT = 0.8
     10 VAL_SPLIT = 0.1
---> 11 from imutils import paths
     12 import random
     14 originalPaths = list(paths.list_images(INPUT_DATASET))

ModuleNotFoundError: No module named 'imutils'
```

```
In [ ]:
```