Intermediate Results



Image\_Index True\_Label Predicted\_Label Prediction\_Confidence

0 0 1 1 1.000000e+00

1 1 1 1 1.000000e+00

2 2 1 1 1.000000e+00

3 3 1 1 1.000000e+00

4 4 1 1 1.000000e+00

5 5 0 0 0.000000e+00

6 6 1 1 1.000000e+00

7 7 1 1 1.000000e+00

8 8 1 1 1.000000e+00

9 9 1 1 1.000000e+00

10 10 1 1 1.000000e+00

11 11 1 1 1.000000e+00

12 12 0 0 3.706528e-32

13 13 0 0 0.000000e+00

14 14 0 0 0.000000e+00

15 15 1 1 1.000000e+00

16 16 0 0 3.706528e-32

17 17 0 0 0.000000e+00

18 18 0 0 0.000000e+00

19 19 0 0 0.000000e+00

20 20 0 0 1.306600e-10

21 21 0 0 3.706528e-32

22 22 0 0 1.525824e-38

23 23 1 1 1.000000e+00

24 24 0 0 1.306600e-10

25 25 0 0 0.000000e+00

26 26 1 0 1.798595e-04

27 27 0 0 1.525824e-38

28 28 1 1 1.000000e+00

29 29 0 0 0.000000e+00

30 30 0 0 1.525824e-38

31 31 1 0 7.638519e-04

32 32 0 1 9.999944e-01

33 33 1 1 1.000000e+00

34 34 1 1 1.000000e+00

35 35 0 1 1.000000e+00

36 36 1 1 1.000000e+00

37 37 1 1 1.000000e+00

38 38 1 1 1.000000e+00

39 39 1 1 1.000000e+00

39 39 1 1 1.000000e+00

Data Preprocessing

import os

import cv2

import numpy as np

def load\_and\_process\_dataset(folder\_path):

"""Loads and processes images using alternative while loop structures."""

dataset = []

labels = []

class\_folders = ['non-cancer', 'cancer']

class\_index = 0

while class\_index < len(class\_folders):

class\_folder = class\_folders[class\_index]

images\_path = os.path.join(folder\_path, class\_folder)

image\_index = 0

while True: # Loop infinitely until a "break" occurs

try:

image\_name = os.listdir(images\_path)[image\_index] # Access by index

image\_path = os.path.join(images\_path, image\_name)

image = cv2.imread(image\_path)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB) # Convert to RGB

image = cv2.resize(image, (240, 240))

dataset.append(image)

labels.append(class\_index) # Use class\_index directly for labels

image\_index += 1

except IndexError: # Handle end of image list

break # Exit the inner loop

class\_index += 1

return np.array(dataset), np.array(labels)

# Assuming `folder\_path` is already a string

dataset, labels = load\_and\_process\_dataset(folder\_path)

# Convert to NumPy arrays

dataset = np.array(dataset)

lab = np.array(labels)

# Print shapes

print(dataset.shape, labels.shape)

output

(131, 240, 240, 3) (131)