|  |  |  |
| --- | --- | --- |
|  |  | **RSIP Career Basic AI 088** |
|  |  | **By Tirumalasetty Sai Chaitanya Kumar** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**PROJECT REPORT ON DEEP LEARNING TECHNIQUES**

**FOR BREAST CANCER PREDICTION USING PYTHON**

1. **INTRODUCTION :**

Breast cancer is one of the main causes of cancer deaths worldwide and is the second most common cancer in women and men worldwide. In 2010, it represented about 12 percent of all new cancer cases and 25 percent of all cancers in women while a decade later, the figures have augmented to more than 75% and 89%, respectively.

* 1. **OVERVIEW :**

Computer-aided diagnosis provides a most important option for image diagnosis, which can improve the reliability of experts’ decision-making. Automatic and precision classification for breast cancer histopathological image is of great importance in clinical application for identifying the malignant stage from histopathological images.

* 1. **PURPOSE** :

Computer-aided diagnosis systems showed potential for improving diagnostic accuracy. Since early detection and prevention can significantly reduce the chances of death, it is important to detect breast cancer as early as possible. This project is developed to predict the cancer at the earliest stage using Convolutional Neural Networks from the concept of Deep-Learning.

1. **LITERATURE SURVEY :**

To look at how we have come across this project overcoming the existing problem, we look at these aspects with the proposed solution.

* 1. **EXISTING PROBLEM :**

Breast cancer is typically detected either during screening, before symptoms have developed, or after a woman notices a lump. There are various ways to detect breast cancer including Mammography, Magnetic Resonance Imaging (MRI) Scans, Computed Tomography (CT) Scans, Ultrasound, and Nuclear Imaging. When cancer is suspected, tissue for microscopic analysis is usually obtained from a needle and is stained with H&E, Hematoxylin and Eosin. Selection of the biopsy type is based on multiple factors, including the size, location of the mass, patient factors, preferences and resources. But all of them are very expensive, less accurate and time-taking. Sometimes, it even takes months together to get the final reports and all the aforementioned technologies have their own shortcomings.

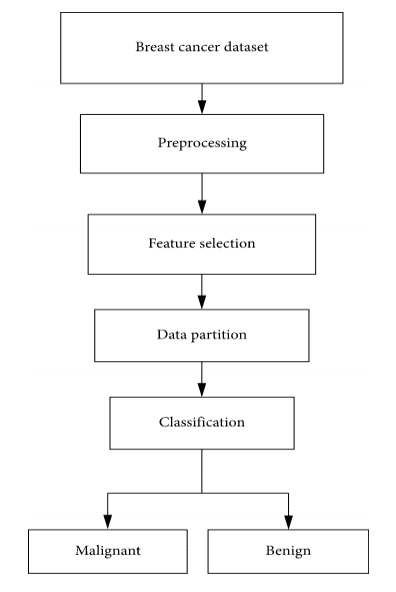
**2.2PROPOSED SOLUTION :**

I have proposed a solution where we build an algorithm to automatically identify whether a patient is suffering from breast cancer or not by looking at biopsy images in less than a minute. This algorithm had to be extremely accurate because lives of people are at stake and time was a real concern. This project results the output in two categories, which are benign and malignant, which mean non cancerous and cancerous, by displaying benign and malignant.

**3. TECHNICAL ANALYSIS :**

**3.1 BLOCK DIAGRAM :**

In the below attached diagram, we see images of biopsy taken as the inputs. We then process them using packages like numpy, ImageDataGenerator, matplotlib, keras etc and classify them from many layers so that the test data gives us accurate results based on training data.



**4.EXPERIMENTAL INVESTIGATIONS :**

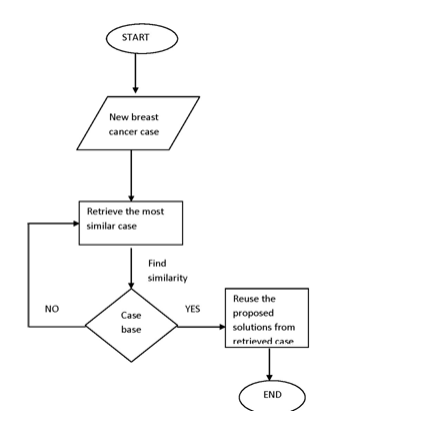
1. I have proposed an automatic breast cancer detection technique that gives prediction accuracy of around 90% for the true class.

2. I have used histopathology images from biopsy, in which tissues affected by the tumor are extracted and stained with H & E.

3. CNN is used for feature extraction, and classification is done using all the various packages to get accurate and quick results.

4. The result is in the form of Binary Classification between two classes of cancer. Benign is class 0 and malignant is class 1.

1. **FLOWCHART :**



1. **RESULT** :

In this project, we designed a new Convolutional Neural Network, the Breast Cancer Histopathology Image Classification Network (BHCNet), for the classification of breast cancer histopathology images. We built a classifier which uses 80% as training dataset, of which 10% is for validation dataset and the rest 20% for test dataset. We designed a small SE-ResNet(Squeeze and Excitation) module with fewer parameters to reduce the training parameters of the model, and to reduce the risk of model over-fitting.

**7. ADVANTAGES AND DISADVANTAGES :**

**ADVANTAGES :**

1. This project helps patients get to know their cancer status really quick.

2. This project will help people to no longer spend thousands and thousands of bugs for getting their reports. This is very economical.

3. This project uses CNN techniques and fetches the results without any physical risk or stain identifications. There are also no or ignorable shortcomings in the project developed so far.

4.Data input can be from various resources.

**DISADVANTAGES :**

1. We have designed this project in Python and is executable only in a properly fixed environment, fulfilling all the hardware and software requirements.

2. It requires high speed data connectivity in real-life situations. Therefore, since the dataset used is large, we need to make sure that the internet connectivity is high and the run-time environment is user-friendly.

**8. APPLICATIONS** :

The developed system can be used in hospitals,diagnostic centres, implantation centers, laboratories etc where people can go and find out if they are affected with breast cancers or for any other purpose. This is a very practical implementation of identifying breast cancer in patients and is a very feasible and economical method.

**9. CONCLUSION :**

I have developed this project where we can identify if the cancer developed is benign or malignant. We do this by taking the images of biopsies and processing them through various layers of CNN and obtaining the results in either of the two cases(0 or 1). Atlast, we also use a confusion matrix to derive the performance of the model.

**10. FUTURE SCOPE :**

1. This project can be used in all the hospitals to identify the situation of patients’ body and recover the malignant patients at the early stages.

2. The project can be used to further add-on other features to improvise the project and get higher efficiency with multiple facilities.

3. The use of Deep-Learning techniques is better than using density based models to get accurate results.

**Source codes:**

**1.config.py:**

1. import os

2.

3. INPUT\_DATASET = "datasets/original"

4.

5. BASE\_PATH = "datasets/idc"

6. TRAIN\_PATH = os.path.sep.join([BASE\_PATH, "training"])

7. VAL\_PATH = os.path.sep.join([BASE\_PATH, "validaon"])

8. TEST\_PATH = os.path.sep.join([BASE\_PATH, "tesng"])

9.

10. TRAIN\_SPLIT = 0.8 11. VAL\_SPLIT = 0.1

**2.build\_dataset.py:**

1. from casncernet import config

2. from imuls import paths

3. import random, shul, os

4.

5. originalPaths=list(paths.list\_images(config.INPUT\_DATASET))

6. random.seed(7)

7. random.shuffle(originalPaths)

8.

9. index=int(len(originalPaths)\*config.TRAIN\_SPLIT)

10. trainPaths=originalPaths[:index]

11. testPaths=originalPaths[index:]

12.

13. index=int(len(trainPaths)\*config.VAL\_SPLIT)

14. valPaths=trainPaths[:index]

15. trainPaths=trainPaths[index:]

16.

17. datasets=[("training", trainPaths, config.TRAIN\_PATH),

18. ("validaon", valPaths, config.VAL\_PATH),

19. ("tesng", testPaths, config.TEST\_PATH)

20. ]

21.

22. for (setType, originalPaths, basePath) in datasets:

23. print(f'Building {setType} set')

24.

25. if not os.path.exists(basePath):

26. print(f'Building directory {base\_path}')

27. os.makedirs(basePath)

28.

29. for path in originalPaths:

30. file=path.split(os.path.sep)[-1]

31. label=file[-5:-4]

32.

33. labelPath=os.path.sep.join([basePath,label])

34. if not os.path.exists(labelPath):

35. print(f'Building directory {labelPath}')

36. os.makedirs(labelPath)

37.

38. newPath=os.path.sep.join([labelPath, file])

39. shul.copy2(inputPath, newPath)

**3.cancernet.py:**

1. from keras.models import Sequenal

2. from keras.layers.normalizaon import BatchNormalizaon

3. from keras.layers.convoluonal import SeparableConv2D

4. from keras.layers.convoluonal import MaxPooling2D

5. from keras.layers.core import Acvaon

6. from keras.layers.core import Flaen

7. from keras.layers.core import Dropout

8. from keras.layers.core import Dense

9. from keras import backend as K

10.

11. class CancerNet:

12. @stacmethod

13. def build(width,height,depth,classes):

14. model=Sequenal()

15. shape=(height,width,depth)

16. channelDim=-1

17.

18. if K.image\_data\_format()=="channels\_first":

19. shape=(depth,height,width)

20. channelDim=1

21.

22. model.add(SeparableConv2D(32, (3,3), padding="same",input\_shape=shape))

23. model.add(Acvaon("relu"))

24. model.add(BatchNormalizaon(axis=channelDim))

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

26. model.add(Dropout(0.25))

27.

28. model.add(SeparableConv2D(64, (3,3), padding="same"))

29. model.add(Acvaon("relu"))

30. model.add(BatchNormalizaon(axis=channelDim))

31. model.add(SeparableConv2D(64, (3,3), padding="same"))

32. model.add(Acvaon("relu"))

33. model.add(BatchNormalizaon(axis=channelDim))

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

35. model.add(Dropout(0.25))

36.

37. model.add(SeparableConv2D(128, (3,3), padding="same"))

38. model.add(Acvaon("relu"))

39. model.add(BatchNormalizaon(axis=channelDim))

40. model.add(SeparableConv2D(128, (3,3), padding="same"))

41. model.add(Acvaon("relu"))

42. model.add(BatchNormalizaon(axis=channelDim))

43. model.add(SeparableConv2D(128, (3,3), padding="same"))

44. model.add(Acvaon("relu"))

45. model.add(BatchNormalizaon(axis=channelDim))

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

47. model.add(Dropout(0.25))

48.

49. model.add(Flaen())

50. model.add(Dense(256))

51. model.add(Acvaon("relu"))

52. model.add(BatchNormalizaon())

53. model.add(Dropout(0.5))

54.

55. model.add(Dense(classes))

56. model.add(Acvaon("somax"))

57.

58. return model

**4.train\_model.py:**

1. import matplotlib

2. matplotlib.use("Agg")

3.

4. from keras.preprocessing.image import ImageDataGenerator

5. from keras.callbacks import LearningRateScheduler

6. from keras.opmizers import Adagrad

7. from keras.uls import np\_uls

8. from sklearn.metrics import classificaon\_report

9. from sklearn.metrics import confusion\_matrix

10. from cancernet.cancernet import CancerNet

11. from cancernet import config

12. from imuls import paths

13. import matplotlib.pyplot as plt

14. import numpy as np

15. import os

16.

17. NUM\_EPOCHS=40; INIT\_LR=1e-2; BS=32

18.

19. trainPaths=list(paths.list\_images(config.TRAIN\_PATH))

20. lenTrain=len(trainPaths)

21. lenVal=len(list(paths.list\_images(config.VAL\_PATH)))

22. lenTest=len(list(paths.list\_images(config.TEST\_PATH)))

23.

24. trainLabels=[int(p.split(os.path.sep)[-2]) for p in trainPaths]

25. trainLabels=np\_uls.to\_categorical(trainLabels)

26. classTotals=trainLabels.sum(axis=0)

27. classWeight=classTotals.max()/classTotals

28.

29. trainAug = ImageDataGenerator(

30. rescale=1/255.0,

31. rotaon\_range=20,

32. zoom\_range=0.05,

33. width\_shi\_range=0.1,

34. height\_shi\_range=0.1,

35. shear\_range=0.05,

36. horizontal\_flip=True,

37. vercal\_flip=True,

38. fill\_mode="nearest")

39.

40. valAug=ImageDataGenerator(rescale=1 / 255.0)

41.

42. trainGen = trainAug.flow\_from\_directory(

43. config.TRAIN\_PATH,

44. class\_mode="categorical",

45. target\_size=(48,48),

46. color\_mode="rgb",

47. shuffle=True,

48. batch\_size=BS)

49. valGen = valAug.flow\_from\_directory(

50. config.VAL\_PATH,

51. class\_mode="categorical",

52. target\_size=(48,48),

53. color\_mode="rgb",

54. shuffle=False,

55. batch\_size=BS)

56. testGen = valAug.flow\_from\_directory(

57. config.TEST\_PATH,

58. class\_mode="categorical",

59. target\_size=(48,48),

60. color\_mode="rgb",

61. shuffle=False,

62. batch\_size=BS)

63.

64. model=CancerNet.build(width=48,height=48,depth=3,classes=2)

65. opt=Adagrad(lr=INIT\_LR,decay=INIT\_LR/NUM\_EPOCHS)

66. model.compile(loss="binary\_crossentropy",opmizer=opt,metrics=["accuracy"])

67.

68.

69. M=model.fit\_generator(

70. trainGen,

71. steps\_per\_epoch=lenTrain//BS,

72. validaon\_data=valGen,

73. validaon\_steps=lenVal//BS,

74. class\_weight=classWeight,

75. epochs=NUM\_EPOCHS)

76.

77. print("Now evaluang the model")

78. testGen.reset()

79. pred\_indices=model.predict\_generator(testGen,steps=(lenTest//BS)+1)

80.

81. pred\_indices=np.argmax(pred\_indices,axis=1)

82.

83. print(classificaon\_report(testGen.classes, pred\_indices, target\_names=testGen.class\_indices.keys()))

84.

85. cm=confusion\_matrix(testGen.classes,pred\_indices)

86. total=sum(sum(cm))

87. accuracy=(cm[0,0]+cm[1,1])/total

88. specificity=cm[1,1]/(cm[1,0]+cm[1,1])

89. sensivity=cm[0,0]/(cm[0,0]+cm[0,1])

90. print(cm)

91. print(f'Accuracy: {accuracy}')

92. print(f'Specificity: {specificity}')

93. print(f'Sensivity: {sensivity}')

94.

95. N = NUM\_EPOCHS

96. plt.style.use("ggplot")

97. plt.figure()

98. plt.plot(np.arange(0,N), M.history["loss"], label="train\_loss")

99. plt.plot(np.arange(0,N), M.history["val\_loss"], label="val\_loss")

100. plt.plot(np.arange(0,N), M.history["acc"], label="train\_acc")

101. plt.plot(np.arange(0,N), M.history["val\_acc"], label="val\_acc")

102. plt.tle("Training Loss and Accuracy on the IDC Dataset")

103. plt.xlabel("Epoch No.")

104. plt.ylabel("Loss/Accuracy")

105. plt.legend(loc="lower le")

106. plt.savefig('plot.png')

5 .App.py :

1. from \_\_future\_\_ import division, print\_funcon

2. # coding=u-8

3. import sys

4. import os

5. import glob

6. import numpy as np

7. from keras.preprocessing import image

8. from keras.applicaons.imagenet\_uls import preprocess\_input, decode\_predicons

9. from keras.models import load\_model

10. from keras import backend

11. from tensorflow.keras import backend

12.

13. import tensorflow as

14.

15. global graph

16. graph=.compat.v1.get\_default\_graph()

17.

18. #global graph

19. #graph = .get\_default\_graph()

20.

21.

22. from skimage.transform import resize

23.

24. # Flask uls

25. from flask import Flask, redirect, url\_for, request, render\_template

26. from werkzeug.uls import secure\_filename

27. from gevent.pywsgi import WSGIServer

28.

29. # Define a flask app

30. app = Flask(\_\_name\_\_)

31.

32. # Model saved with Keras model.save()

33. MODEL\_PATH = 'Breast\_cancer.h5'

34.

35. # Load your trained model

36. model = load\_model(MODEL\_PATH)

37. # Necessary

38. # print('Model loaded. Start serving...')

39. 40. # You can also use pretrained model from Keras

41. # Check hps://keras.io/applicaons/

42. #from keras.applicaons.resnet50 import ResNet50

43. #model = ResNet50(weights='imagenet')

44. #model.save('')

45. print('Model loaded. Check hp://127.0.0.1:5000/')

46. @app.route('/', methods=['GET'])

47. def index():

48. # Main page

49. return render\_template('index.html')

50. @app.route('/predict', methods=['GET', 'POST'])

51. def upload():

52. if request.method == 'POST':

53. # Get the file from post request

54. f = request.files['file']

55.

56. # Save the file to ./uploads

57. basepath = os.path.dirname(\_\_file\_\_)

58. file\_path = os.path.join(

59. basepath, 'uploads', secure\_filename(f.filename))

60. f.save(file\_path)

61. img = image.load\_img(file\_path, target\_size=(48, 48))

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

63. x = np.expand\_dims(x, axis=0)

64.

65. with graph.as\_default():

66. preds = model.predict\_classes(x)

67. print("predicon ",preds)

68. index = ['BENIGN','MALIGNANT']

69. text = "Cancer Type: "+index[preds[0]]

70. # ImageNet Decode

71. return text

72. if \_\_name\_\_ == '\_\_main\_\_':

73. app.run(debug=False,threaded = False)

**Screenshots:**

