SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER APPLICATIONS

A Project Report on

CANCER DETECTION USING DEEP LEARNING

Submitted for partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

BY

Mr. P. Karthikeyan (9923151067)
Mr. R. Marieswaran (9923151050)
Mr. R. Ramakrishnan (9923151132)
Mr. A. Dhetchana moorthy (9923151019)

Under the guidance of

Dr. K. Indhumathi

Assistant Professor
Department of Computer Applications
Kalasalingam Academy of Research and Education

DEPARTMENT OF COMPUTER APPLICATIONS

KALASALINGAM ACADEMY OF RESEARCH AND EDUCATION

DEEMED TO BE UNIVERSITY

(ACCREDITED BY NAAC WITH "A++" GRADE)

ANAND NAGAR- KRISHNANKOIL- 626126

MAY - 2024

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A++" Grade
Anand Nagar, Krishnankoil - 626126. Srivilliputtur (Via), Virudhunagar (Dt), Tamil Nadu | info@kalasalingam.ac.in | www.kalasalingam.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project work entitled **"CANCER DETECTION USING DEEP LEARNING"** is a bonafide work carried out by P. Karthikeyan (9923151067), R. Marieswaran (9923151050), R. Ramakrishnan (9923151132) and A. Dhetchana moorthy (9923151019) in partial fulfilment of the requirements for the award of degree of MASTER OF COMPUTER APPLICATIONS by KALASALINGAM ACADEMY OF RESEARCH AND EDUCATION, Krishnankoil, under our guidance and supervision during the year 2024.

PROJECT GUIDE

Dr. K. INDHUMATHI,

Assistant Professor

Department of Computer Applications

KARE

HEAD OF THE DEPARTMENT

Dr. K. BALASUBRAMANIAN,

Professor & Head

Department of Computer Applications

KARE

Submitted for the project Viva-Voce examination held on _____

---------- ----------

Internal Examiner External Examiner

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A++" Grade
Anand Nagar, Krishnankoil - 626126. Srivilliputtur (Via), Virudhunagar (Dt), Tamil Nadu | info@kalasalingam.ac.in | www.kalasalingam.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER APPLICATIONS

DECLARATION

This is to certify that the work reported in the present project entitled **"CANCER DETECTION USING DEEP LEARNING"** is a record of work done by us in the Department of Computer Applications, Kalasalingam Academy of Research and Education. The reports are based on the project work done entirely by us and not copied from any other source.

Place: Krishnankoil

Date:

Signature of the Students,

P. Karthikeyan (9923151067),

R. Marieswaran (9923151050),

R. Ramakrishnan (9923151132),

A. Dhetchana moorthy (9923151019).

ABSTRACT

Develop a deep learning model for early cancer detection using CNNs to analyze medical imaging data. Preprocess data to enhance model performance. Benefits include enhanced accuracy in cancer detection, automation of diagnostic processes, and potential for personalized treatment plans. Challenges include ensuring data quality, addressing biases, interpreting complex model decisions, managing computational resources, and data privacy. Despite challenges, leveraging deep learning for cancer detection holds immense promise in revolutionizing healthcare, improving patient care, and outcomes. Cancer detection using Convolutional Neural Networks (CNN) is a method that uses deep learning algorithms to automatically detect cancer in medical images. CNNs are a type of neural network that are particularly well-suited for image analysis tasks, as they can automatically learn and extract features from images. In the context of cancer detection, CNNs can be trained on large datasets of medical images to learn the visual patterns that are indicative of cancer.

Once trained, the CNN can then be used to analyze new images and provide a diagnosis. There are several advantages to using CNNs for cancer detection. First, they can provide a more objective and consistent diagnosis than human experts, who may be subject to fatigue, bias, or other factors that can affect their judgement. Second, CNNs can analyze large volumes of images quickly and efficiently, which can help to speed up the diagnostic process and improve patient outcomes. Finally, CNNs can be used to detect cancer at an early stage, when it is more treatable and has a better prognosis. There are also some challenges to using CNNs for cancer detection. One challenge is the need for large and diverse datasets of medical images to train the CNN. Another challenge is the need for high-quality images, as poor image quality can affect the accuracy of the diagnosis. Finally, there is a need for ongoing research and development to improve the accuracy and reliability of CNNs for cancer detection. In summary, cancer detection using CNNs is a promising method that has the potential to improve the accuracy and efficiency of cancer diagnosis. However, there are also some challenges that need to be addressed in order to fully realize the benefits of this technology.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW:

Cancer detection using Convolutional Neural Networks (CNN) is a method that uses deep learning algorithms to automatically detect cancer in medical images. CNNs are a type of neural network that are particularly well-suited for image analysis tasks, as they can automatically learn and extract features from images. In the context of cancer detection, CNNs can be trained on large datasets of medical images to learn the visual patterns that are indicative of cancer. Once trained, the CNN can then be used to analyze new images and provide a diagnosis. TensorFlow is an open-source deep learning framework developed by Google. It provides a wide range of tools and libraries for building and training deep learning models, including support for medical image analysis. TensorFlow has a large and active community of developers, and is widely used in both academia and industry. PyTorch is an open-source deep learning framework developed by Facebook. It is known for its ease of use and flexibility, and provides a dynamic computational graph that allows for efficient and intuitive model building. PyTorch has a growing community of developers, and is increasingly being used for medical image analysis.

Develop a deep learning model for early cancer detection using CNNs to analyze medical imaging data. Preprocess data to enhance model performance. Benefits include enhanced accuracy in cancer detection, automation of diagnostic processes, and potential for personalized treatment plans. Challenges include ensuring data quality, addressing biases, interpreting complex model decisions, managing computational resources, and data privacy. Despite challenges, leveraging deep learning for cancer detection holds immense promise in revolutionizing healthcare, improving patient care, and outcomes. All of these frameworks provide support for medical image analysis, and the choice of fcramework will depend on the specific requirements of the project. TensorFlow and PyTorch are the most widely used frameworks, and have the largest and most active communities of developers. Keras is a high-level framework that provides a simple and intuitive API for building and training models, and is particularly well-suited for medical image analysis. Caffe is known for its speed and efficiency, and is widely used in medical image analysis.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

The existing system may rely on traditional methods of cancer detection, which could involve manual examination of medical images by radiologists or pathologists. These methods can be time-consuming and subjective, leading to variations in diagnosis. While some existing systems might utilize basic automation techniques, such as thresholding or simple machine learning algorithms, they may lack the sophistication and accuracy of CNN-based approaches. Traditional methods may struggle to scale with the increasing volume of medical imaging data generated in healthcare settings. They may also face challenges in handling the complexity of multi-modal imaging data. The decision-making process in traditional systems may lack transparency, making it difficult to understand the reasoning behind a particular diagnosis. This can hinder trust and acceptance among medical practitioners.

2.2 PROPOSED SYSTEM:

The proposed system may employ advanced deep learning techniques, such as transfer learning, ensemble methods, or attention mechanisms, to enhance the performance of the CNN model. Develop a deep learning model for early cancer detection using CNNs to analyze medical imaging data. Preprocess data to enhance model performance. Benefits include enhanced accuracy in cancer detection, automation of diagnostic processes, and potential for personalized treatment plans. Challenges include ensuring data quality, addressing biases, interpreting complex model decisions, managing computational resources, and data privacy. Despite challenges, leveraging deep learning for cancer detection holds immense promise in revolutionizing healthcare, improving patient care, and outcomes. Cancer detection using Convolutional Neural Networks (CNN) is a method that uses deep learning algorithms to automatically detect cancer in medical images.

CNNs are a type of neural network that are particularly well-suited for image analysis tasks, as they can automatically learn and extract features from images. In the context of cancer detection, CNNs can be trained on large datasets of medical images to learn the visual patterns that are indicative of cancer. Once trained, the CNN can then be used to analyze new images and provide a diagnosis. There are several advantages to using CNNs for cancer detection.

First, they can provide a more objective and consistent diagnosis than human experts, who may be subject to fatigue, bias, or other factors that can affect their judgement. Second, CNNs can analyze large volumes of images quickly and efficiently, which can help to speed up the diagnostic process and improve patient outcomes. Finally, CNNs can be used to detect cancer at an early stage, when it is more treatable and has a better prognosis. There are also some challenges to using CNNs for cancer detection. One challenge is the need for large and diverse datasets of medical images to train the CNN. Efforts are made to enhance the interpretability of the CNN model's decisions. Techniques such as attention maps, saliency maps, or model explanations are incorporated to provide insights into the features driving the predictions, thereby increasing trust and understanding among healthcare professionals.

2.2.1 ADVANTAGES:

1. **High Accuracy**: CNNs can learn intricate patterns and features directly from raw medical imaging data, leading to high accuracy in cancer detection. Their ability to capture complex relationships in images often results in more reliable diagnoses compared to traditional methods.

2. **Automation**: CNN-based systems enable automation of the cancer detection process, reducing the need for manual intervention by radiologists or pathologists. This automation can significantly speed up the diagnosis process, leading to quicker treatment decisions and better patient outcomes.

3. **Scalability**: CNNs are highly scalable and can efficiently process large volumes of medical imaging data. As the amount of medical data continues to grow, CNN-based systems can handle this increasing workload without sacrificing performance.

4. **Consistency**: CNNs provide consistent results, minimizing variations in diagnoses that can occur due to differences in human interpretation. This consistency ensures that patients receive standardized and reliable care, regardless of who is interpreting the medical images.

5. **Early Detection**: CNN-based systems can detect subtle signs of cancer that may be missed by human observers, enabling earlier detection and treatment. Early detection is crucial for improving patient outcomes and increasing survival rates for various types of cancer.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS:

GRAPHICS PROCESSING UNITS (GPUS):

GPUs are essential for training deep learning models like CNNs efficiently. They excel at performing the matrix multiplications and parallel computations involved in training neural networks. High-performance GPUs with CUDA-enabled cores from NVIDIA or similar architectures from other vendors are commonly used for deep learning tasks.

CENTRAL PROCESSING UNITS (CPUS):

While GPUs handle the heavy computational workload during training, CPUs are still important for managing overall system operations and handling tasks that are not parallelizable. Modern multi-core CPUs with sufficient memory bandwidth are recommended to support the training process.

MEMORY (RAM):

Deep learning models, especially large CNNs, require significant amounts of memory during both training and inference. Having an ample amount of RAM ensures that the entire dataset can be loaded efficiently into memory, reducing I/O bottlenecks during training.

STORAGE:

Large datasets of medical images can occupy substantial storage space. High-speed solid-state drives (SSDs) or large-capacity hard disk drives (HDDs) are needed to store and access these datasets efficiently. Additionally, SSDs are preferred for faster data loading during training.

CLUSTER OR CLOUD COMPUTING RESOURCES:

For large-scale training tasks or when dealing with massive datasets, it may be necessary to use clusters of computers or cloud computing resources. Cloud-based platforms

such as AWS, Google Cloud, or Microsoft Azure offer scalable GPU instances specifically optimized for deep learning tasks.

NETWORKING:

When using distributed computing resources or accessing cloud-based services, a reliable and high-speed network connection is crucial for transferring data between nodes and accessing remote computing resources efficiently.

3.2 SOFTWERE REQUIREMENTS:

DEEP LEARNING FRAMEWORKS:

Software libraries such as TensorFlow, PyTorch, or Keras provide tools and APIs for building, training, and deploying deep learning models, including CNNs.

PYTHON:

A programming language commonly used for machine learning and deep learning tasks. Many deep learning frameworks are built on top of Python.

DEVELOPMENT ENVIRONMENTS:

Integrated Development Environments (IDEs) such as Jupyter Notebook, PyCharm, or Visual Studio Code are often used for coding, debugging, and experimentation with deep learning models.

IMAGE PROCESSING LIBRARIES:

Libraries like OpenCV or scikit-image may be used for preprocessing and manipulation of medical images before feeding them into the CNN model.

DATA MANAGEMENT TOOLS:

Software for managing and organizing large datasets, such as pandas for data manipulation and TensorFlow Datasets for handling preprocessed datasets.

DEPLOYMENT TOOLS:

Tools and frameworks for deploying trained models into production environments, such as TensorFlow Serving, ONNX Runtime, or Flask for building web applications.

VISUALIZATION TOOLS:

Libraries like Matplotlib, Seaborn, or TensorBoard for visualizing training metrics, model performance, and intermediate outputs during the training process.

DOCUMENTATION AND REPORTING:

Tools for documenting code, experiments, and results, such as Markdown for writing documentation and Jupyter Notebooks for interactive reporting.

OPERATING SYSTEM:

Most deep learning frameworks and tools are compatible with major operating systems like Windows, macOS, and Linux.

CHAPTER 4

SYSTEM DESCRIPTION

4.1 PYTHON 3.12.3:

Python 3.12.3 is indeed the latest maintenance release of Python 3.12, which was released on **April 9, 2024**. This version includes over **300 bugfixes, build improvements, and documentation changes** since Python 3.12.2. Here are some of the major new features and changes introduced in the Python 3.12 series compared to Python 3.11.

NEW FEATURES:

1. More flexible f-string parsing (PEP 701). Support for the buffer protocol in Python code (PEP 688). A new debugging/profiling API (PEP 669).
2. Support for isolated sub interpreters with separate Global Interpreter Locks (GIL). Improved error messages and typo suggestions.
3. Support for the Linux perf profiler to report Python function names in traces.
4. Performance improvements delivering an estimated 5% overall performance improvement.

TYPE ANNOTATIONS:

New type annotation syntax for **generic classes** (PEP 695). New **override decorator** for methods (PEP 698).

DEPRECATIONS AND REMOVALS:

Removal of deprecated wstr and wstr_length members of the C implementation of unicode objects (PEP 623). Removal of long deprecated methods and classes in the unittest module. Removal of the deprecated smtpd and distutils modules (PEP 594 and PEP 632).

SYNTAX CHANGES:

Invalid backslash escape sequences in strings now warn with Syntax Warning instead of Deprecation Warning.

INTERNAL CHANGES:

The internal representation of integers has changed in preparation for performance enhancements.

4.2 PACKAGE:

Python packages are bundles of code that extend the functionality of Python by providing additional modules, classes, functions, and variables for specific tasks or applications. These packages are created by developers and community members and are typically distributed and installed using package managers like pip or conda. Python libraries make it very easy for us to handle the data and perform typical and complex tasks with a single line of code.

4.2.1 PANDAS :

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

4.2.2 NUMPY:

Numpy arrays are very fast and can perform large computations in a very short time. Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

4.2.3 MATPLOTLIB:

Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc. In this article, to gain a comprehensive understanding of the diverse range of plots and charts supported by Matplotlib, empowering to create compelling and informative visualizations for data analysis tasks.

4.2.4 SKLEARN:

Scikit-learn, often abbreviated as sklearn, is a widely-used machine learning library in Python. It provides simple and efficient tools for data mining and data analysis, built on top of other popular libraries like NumPy, SciPy, and matplotlib. Sklearn offers a wide range of

machine learning algorithms for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

4.2.5 OPENCV:

Opencv is a huge open-source library for computer vision, machine learning, and image processing. Opencv supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with opencv. This is an open-source library mainly focused on image processing and handling.

4.2.6 TENSORFLOW:

TensorFlow is an open-source machine learning framework developed by Google that provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models. It is designed to be flexible, scalable, and efficient, making it suitable for a wide range of applications, from research prototyping to large-scale production deployments. This is an open-source library that is used for Machine Learning and Artificial intelligence and provides a range of functions to achieve complex functionalities with single lines of code.

CHAPTER 5

SYSTEM DESIGN AND DEVELOPMENT

5.1 APPLICATION DESIGN:

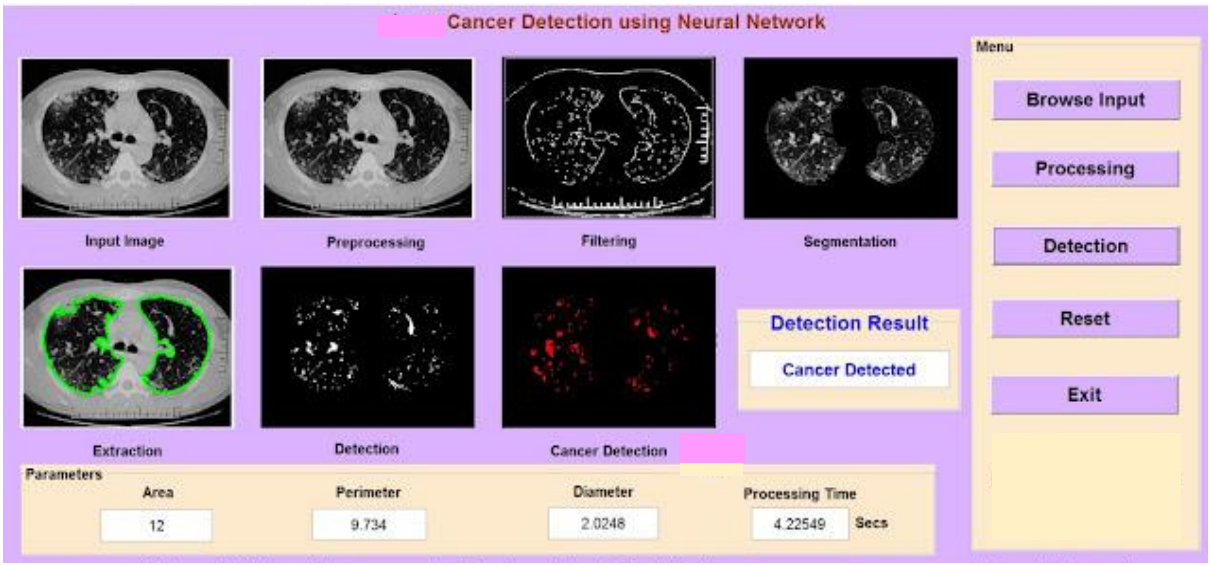

Fig 5.1.1 Application Design for Cancer Detection using Deep learning

5.2 DATA FLOW DIAGRAM:

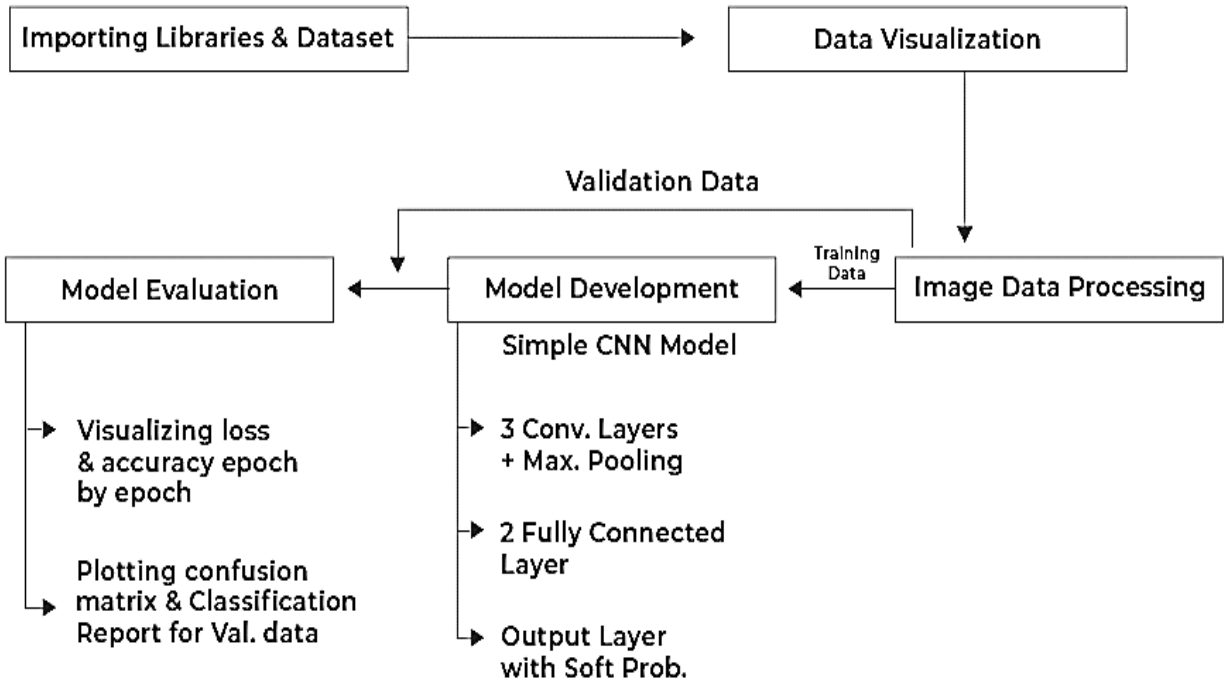

Fig 5.2.1 Data Flow Diagram for Cancer Detection using Deep learning

5.2.1 IMPORTING DATASET:

The dataset which we will use here has been taken from realtime dataset histopathological-images. This dataset includes 5000 images for three classes of cancers conditions:

1. Normal Class
2. Adenocarcinomas
3. Squamous cell carcinomas

These images for each class have been developed from 250 images by performing data augmentation on them. That is why we won't be using data augmentation further on these images.

5.2.2 DATA VISUALIZATION:

Data visualization is the graphical representation of information and data in a pictorial or graphical format (Visualization of Data could be: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, and Dashboards. In this section, we will try to understand visualize some images which have been provided to us to build the classifier for each class.

5.2.3 DATA PREPARATION FOR TRAINING:

In this section, to convert the given images into NumPy arrays of their pixels after resizing them because training a **Deep Neural Network** on large-size images is highly inefficient in terms of computational cost and time. For this purpose, to use the OpenCV library and Numpy library of python to serve the purpose. Also, after all the images are converted into the desired format, we will split them into training and validation data so, that evaluate the performance of our model.

5.2.4 MODEL DEVELOPMENT:

From this step onward to use the TensorFlow library to build our CNN model. Keras framework of the tensor flow library contains all the functionalities that one may need to define the architecture of a Convolutional Neural Network and train it on the data.

5.2.5 MODEL ARCHITECTURE:

To implement a Sequential model which will contain the following parts:

- Three Convolutional Layers followed by MaxPooling Layers.
- The Flatten layer to flatten the output of the convolutional layer. Then we will have two fully connected layers followed by the output of the flattened layer.
- To include some BatchNormalization layers to enable stable and fast training and a Dropout layer before the final layer to avoid any possibility of overfitting.
- The final layer is the output layer which outputs soft probabilities for the three classes.

5.2.6 CALLBACK:

Callbacks are used to check whether the model is improving with each epoch or not. If not then what are the necessary steps to be taken like ReduceLROnPlateau decreases learning rate further. Even then if model performance is not improving then training will be stopped by EarlyStopping. We can also define some custom callbacks to stop training in between if the desired results have been obtained early.

5.2.7 MODEL EVALUATION:

Now as we have our model ready let's evaluate its performance on the validation data using different metrics. For this purpose, we will first predict the class for the validation data using this model and then compare the output with the true labels.

CHAPTER 6

APPENDIX

6.1 SAMPLE CODING:

MODULES USED:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from PIL import Image

from glob import glob

from sklearn.model_selection import train_test_split

from sklearn import metrics

import cv2

import gc

import os

import tensorflow as tf

from tensorflow import keras

from keras import layers

import warnings

warnings.filterwarnings('ignore')
```

IMPORTING DATASET:

```
from zipfile import ZipFile

data_path = 'lung-and-colon-cancer-histopathological-images.zip'

with ZipFile(data_path,'r') as zip:

zip.extractall()

print('The data set has been extracted.')
```

DATA VISUALIZATION:

```
path = 'lung_colon_image_set/lung_image_sets'

classes = os.listdir(path)

path = '/lung_colon_image_set/lung_image_sets'

for cat in classes:

        image_dir = f'{path}/{cat}'

        images = os.listdir(image_dir)

        fig, ax = plt.subplots(1, 3, figsize=(15, 5))

        fig.suptitle(f'Images for {cat} category . . . .', fontsize=20)

        for i in range(3):

                k = np.random.randint(0, len(images))

                img = np.array(Image.open(f'{path}/{cat}/{images[k]}'))

                ax[i].imshow(img)

                ax[i].axis('off')

        plt.show()
```

DATA PREPARATION FOR TRAINING:

```
IMG_SIZE = 256

SPLIT = 0.2

EPOCHS = 10

BATCH_SIZE = 64

X = []

Y = []

for i, cat in enumerate(classes):

images = glob(f'{path}/{cat}/*.jpeg')

for image in images:

        img = cv2.imread(image)

        X.append(cv2.resize(img, (IMG_SIZE, IMG_SIZE)))

        Y.append(i)

X = np.asarray(X)

one_hot_encoded_Y = pd.get_dummies(Y).values

X_train, X_val, Y_train, Y_val = train_test_split(X, one_hot_encoded_Y, test_size = SPLIT,
                                                  random_state = 2022)

print(X_train.shape, X_val.shape)
```

MODEL ARCHITECTURE:

```
model = keras.models.Sequential([
        layers.Conv2D(filters=32,
                      kernel_size=(5, 5),
                      activation='relu',
                      input_shape=(IMG_SIZE,
                                   IMG_SIZE,
                                   3),
                      padding='same'),
        layers.MaxPooling2D(2, 2),
        layers.Conv2D(filters=64,
                      kernel_size=(3, 3),
                      activation='relu',
                      padding='same'),
        layers.MaxPooling2D(2, 2),
        layers.Conv2D(filters=128,
                      kernel_size=(3, 3),
                      activation='relu',
                      padding='same'),
        layers.MaxPooling2D(2, 2),
        layers.Flatten(),
```

```
        layers.Dense(256, activation='relu'),
        layers.BatchNormalization(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.BatchNormalization(),
        layers.Dense(3, activation='softmax')
])
model.summary()
keras.utils.plot_model(
        model,
        show_shapes = True,
        show_dtype = True,
        show_layer_activations = True
)
model.compile(
        optimizer = 'adam',
        loss = 'categorical_crossentropy',
        metrics = ['accuracy']
)
```

CALLBACK:

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
                if logs.get('val_accuracy') > 0.90:
                        print('\n Validation accuracy has reached upto \
                                        90% so, stopping further training.')
                        self.model.stop_training = True
es = EarlyStopping(patience=3,
                                monitor='val_accuracy',
                                restore_best_weights=True)
lr = ReduceLROnPlateau(monitor='val_loss',
                                        patience=2,
                                        factor=0.5,
                                        verbose=1)
history = model.fit(X_train, Y_train,
                                        validation_data = (X_val, Y_val),
                                        batch_size = BATCH_SIZE,
                                        epochs = EPOCHS,
                                        verbose = 1,
                                        callbacks = [es, lr, myCallback()])
```

```
history_df = pd.DataFrame(history.history)

history_df.loc[:,['loss','val_loss']].plot()

history_df.loc[:,['accuracy','val_accuracy']].plot()

plt.show()
```

MODEL EVALUATION:

```
Y_pred = model.predict(X_val)

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

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

metrics.confusion_matrix(Y_val, Y_pred)

print(metrics.classification_report(Y_val, Y_pred, target_names=classes))
```

6.2 SAMPLE SCREEN SHOT:

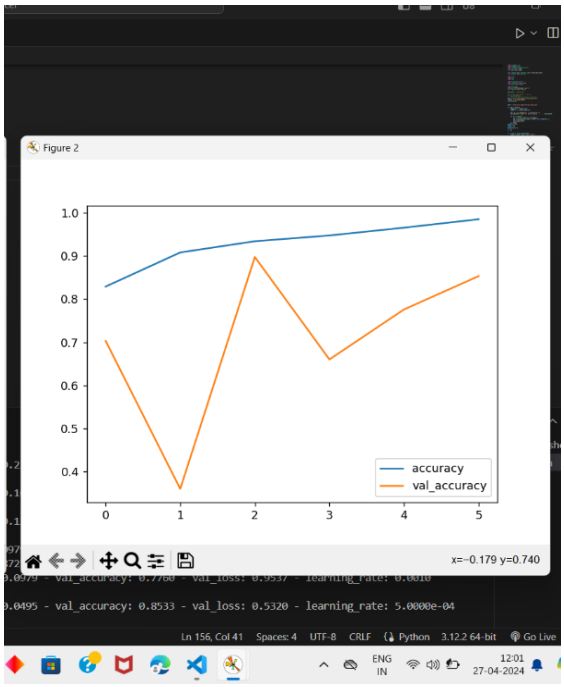

Fig 6.2.1 validation accuracy

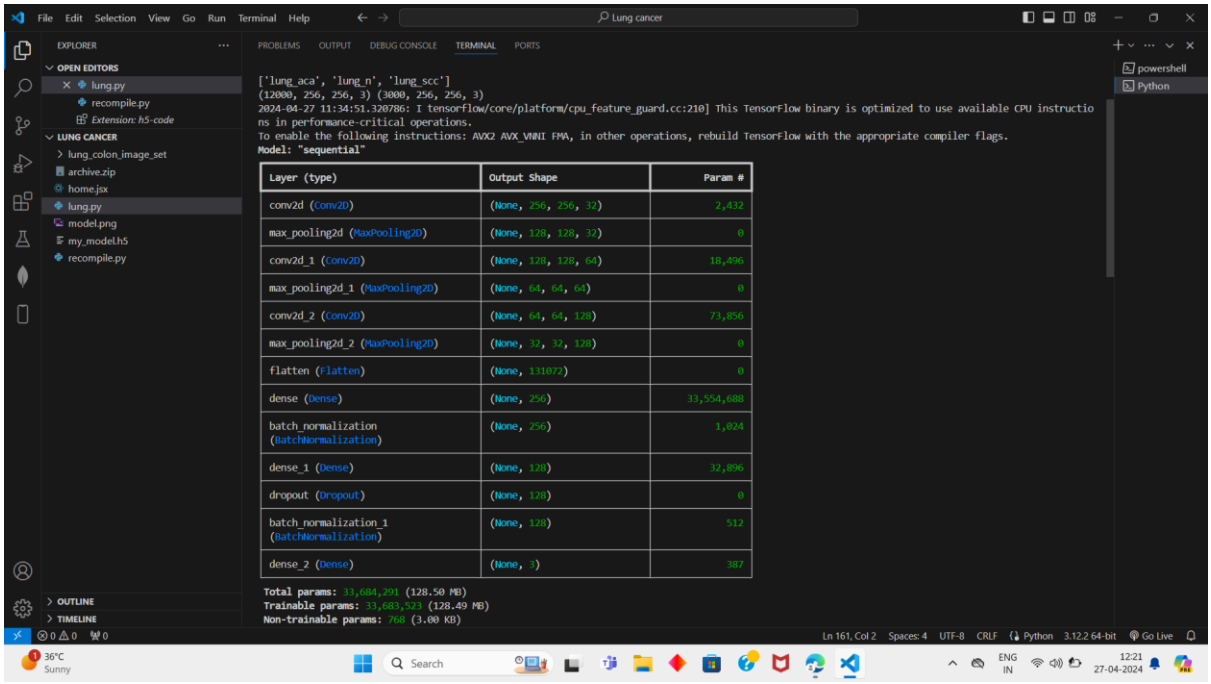

Fig 6.2.2 CNN model layers

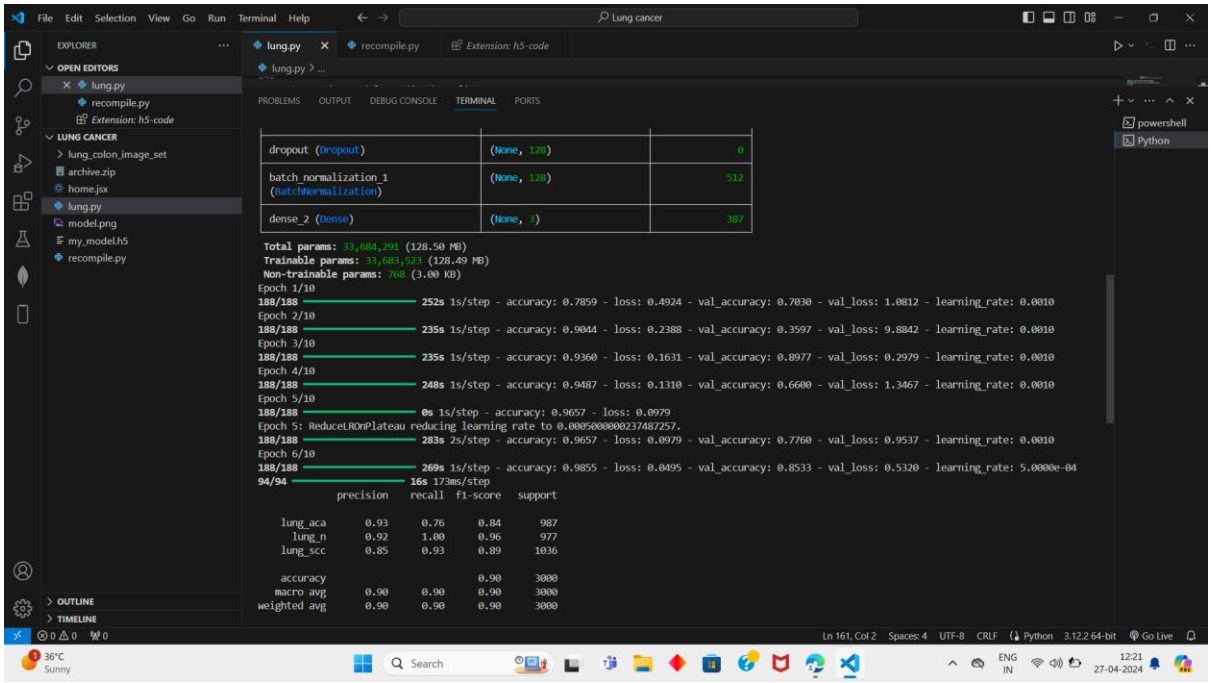

Fig 6.2.3 Performance evaluation

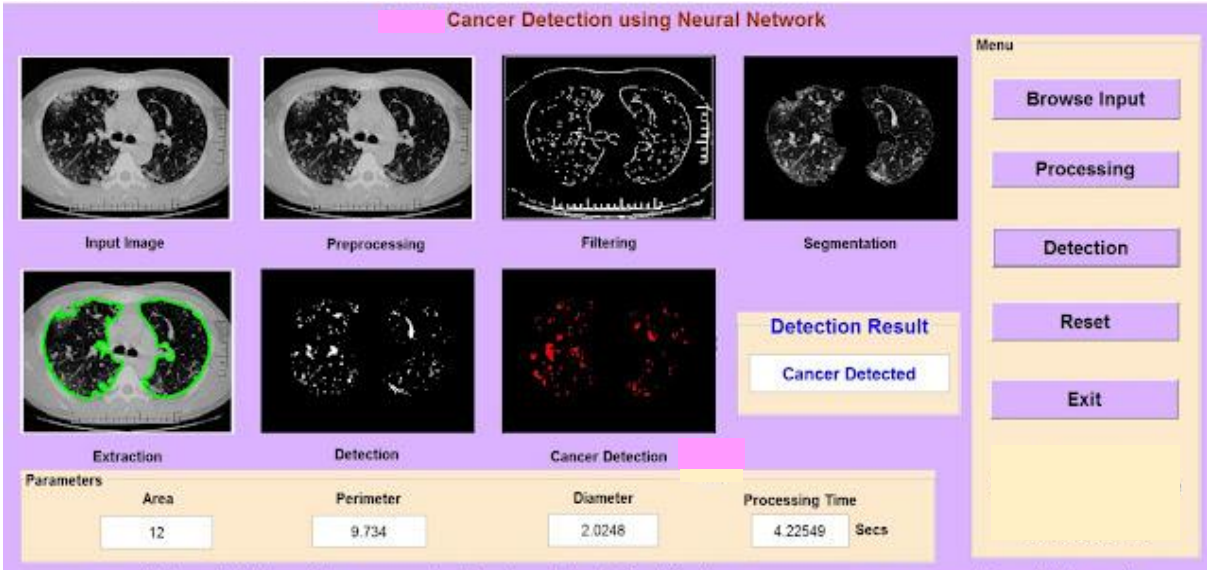

Fig 6.2.4 Application Design for cancer detection

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENT

7.1 CONCLUSION:

Deep learning presents significant potential in cancer detection, offering improved diagnostic accuracy, workflow automation, and advancements in personalized medicine. Advantages include enhanced accuracy, scalability, and the potential for tailored treatments. Challenges such as data quality, interpretability, and ethical considerations require careful attention. Despite hurdles, continuous model improvement underscores deep learning's importance in revolutionizing cancer diagnosis and patient care. Addressing these challenges is essential for realizing the full potential of deep learning in healthcare. Indeed the performance of our simple CNN model is very good as the **f1-score** for each class is above 0.90 which means our model's prediction is correct 90% of the time. This is what we have achieved with a simple CNN model what if we use the Transfer Learning Technique to leverage the pre-trained parameters which have been trained on millions of datasets and for weeks using multiple GPUs? It is highly likely to achieve even better performance on this dataset. Overall, deep learning holds immense promise in transforming cancer detection and improving patient outcomes.

7.2 FUTURE ENHANCEMENT:

In the real of CNN-based cancer detection systems, future enhancements are poised to elevate the field to new heights. Research endeavors may concentrate on evolving model architectures, honing them to better analyze medical images with heightened accuracy and efficiency. The fusion of data from multiple imaging modalities, such as MRI, CT, and histopathology images, holds promise for bolstering the robustness and precision of these systems. Leveraging pre-trained models and domain adaptation methods could prove instrumental, particularly in scenarios where labeled data is limited or in transferring knowledge across diverse medical imaging domains. Moreover, advancements in uncertainty estimation techniques would furnish models with the ability to gauge the reliability of their predictions, thereby enhancing interpretability. Expanding the repertoire of interpretation methods, including attention maps and saliency maps, could fortify trust and comprehension among healthcare professionals. Furthermore, exploring sophisticated data augmentation and

synthesis techniques could mitigate challenges stemming from data scarcity and class imbalance. Initiatives in privacy-preserving techniques, such as federated learning or differential privacy, are paramount to facilitating collaborative model training while upholding patient privacy. Additionally, optimizing models for real-time inference and deployment on edge devices holds the potential to enable point-of-care cancer detection in clinical settings. Lastly, ensuring robust clinical validation and regulatory approval processes are imperative steps toward the real-world implementation of CNN-based cancer detection systems, underscoring their safety, efficacy, and ethical compliance. Through concerted efforts in these realms, CNN-based cancer detection systems are poised to revolutionize early diagnosis, treatment planning, and patient care in oncology.

CHAPTER 8

REFERENCES

8.1 TEXT BOOK:

[1] Convolutional neural network: an overview and application in radiology and it was published in the journal Insights into Imaging in 2018.

[2] Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions Published: 18 August 2021.

[3] A review on deep learning in medical image analysis Published online 2021 Sep 4.

[4] Cancer Biology, Epidemiology, and Treatment in the 21st Century: Current Status and Future Challenges From a Biomedical Perspective Published online 2021 Sep 27.

8.2 WEBSITES:

https://www.springernature.com/gp/open-research/about/the-fundamentals-of-open-access-and-open-research

https://doi.org/10.1007/s42979-021-00815-1

A review on deep learning in medical image analysis - PMC (nih.gov)

https://journals.sagepub.com/doi/10.1177/10732748211038735