1.PROJECT DESCRIPTION

INTRODUCTION

Lung cancer originates when cells in the lung mutate or change abnormally, leading to uncontrolled growth and the formation of tumors. These tumors can interfere with the normal function of the lungs, impairing breathing and potentially spreading to other parts of the body.

Types of Lung Cancer:

Non-small cell lung cancer (NSCLC): This category includes several subtypes, such as adenocarcinoma, squamous cell carcinoma, and large cell carcinoma. Each subtype has unique characteristics and may respond differently to treatment.

Small cell lung cancer (SCLC): This aggressive type of lung cancer tends to grow quickly and is often associated with smoking.

Causes and Risk Factors:

Smoking tobacco, including cigarettes, cigars, and pipes, is the primary cause of lung cancer. Tobacco smoke contains numerous carcinogens that damage lung cells and trigger cancerous changes. Exposure to environmental factors such as radon gas, asbestos fibers, air pollution, and certain industrial chemicals can increase the risk of developing lung cancer.

Individuals with a family history of lung cancer or certain genetic mutations may also have an elevated risk.

Symptoms:

Early-stage lung cancer may not produce noticeable symptoms. As the disease progresses, common symptoms may include a persistent cough, chest pain, difficulty breathing, hoarseness, recurrent respiratory infections, fatigue, and unintended weight loss.

Symptoms can vary depending on the location and size of the tumor, as well as whether the cancer has spread to other organs.

Diagnosis:

Diagnosing lung cancer typically involves a combination of medical history evaluation, physical examination, imaging tests (such as chest X-rays, CT scans, and PET scans), and tissue biopsy.A biopsy is the definitive method for confirming lung cancer, where a sample of suspicious tissue is extracted and examined under a microscope by a pathologist.

Staging:

Staging determines the extent of cancer spread and helps healthcare providers plan appropriate treatment strategies. The most commonly used staging system for lung cancer is the TNM system, which evaluates the tumor size, lymph node involvement, and presence of distant metastases.

Stages range from 0 (localized cancer) to IV (advanced cancer with distant metastases).

Treatment Options:

The treatment approach for lung cancer depends on several factors, including the type and stage of cancer, the patient's overall health, and their treatment preferences.

Common treatment modalities include surgery, chemotherapy, radiation therapy, targeted therapy, immunotherapy, and palliative care.

Surgery:

Surgical resection aims to remove the cancerous tissue from the lung. The type of surgery performed depends on the size and location of the tumor, as well as the patient's lung function and overall health.

Lobectomy, where an entire lobe of the lung is removed, is the most common surgical procedure for early-stage lung cancer.

Chemotherapy and Radiation Therapy:

Chemotherapy involves the use of potent drugs to kill cancer cells or slow their growth. It may be administered orally, intravenously, or directly into the lungs (intracavitary chemotherapy).

Radiation therapy uses high-energy radiation beams to target and destroy cancer cells. It can be delivered externally (external beam radiation therapy) or internally (brachytherapy).

Targeted Therapy and Immunotherapy:

Targeted therapy drugs are designed to selectively inhibit specific molecular pathways or proteins involved in cancer growth and progression. They are particularly effective against certain types of NSCLC that have specific genetic mutations or alterations.

Immunotherapy harnesses the body's immune system to recognize and attack cancer cells. Immune checkpoint inhibitors, such as PD-1/PD-L1 inhibitors, have shown promising results in treating advanced lung cancer by blocking the immune evasion mechanisms employed by cancer cells.

Prognosis and Survival Rate:

The prognosis for lung cancer varies widely depending on factors such as the stage at diagnosis, the type of lung cancer, the patient's overall health, and their response to treatment.
Overall, the survival rate for lung cancer remains relatively low compared to many other types of cancer, particularly when diagnosed at advanced stages. However, advances in treatment options and early detection techniques have led to improved outcomes for some patients.

Prevention:

Smoking cessation is the single most effective way to prevent lung cancer and reduce the risk of numerous other health problems. Quitting smoking at any age can significantly lower the risk of developing lung cancer.

Minimizing exposure to environmental carcinogens, such as radon gas and asbestos, can also help reduce the risk of lung cancer.

Education and public awareness campaigns play a crucial role in promoting healthy lifestyle choices and encouraging regular screening for those at high risk.

Support and Resources:

Lung cancer patients and their families can benefit from a range of support services, including counseling, support groups, and educational materials.

Healthcare providers, advocacy organizations, and community resources can offer valuable information and assistance to individuals navigating the challenges of lung cancer diagnosis and treatment.

Early detection initiatives, smoking cessation programs, and research funding are essential components of comprehensive efforts to combat lung cancer and improve patient outcomes

2. SYSTEM STUDY

2.1 EXISTING SYSTEM:

Loading and Preprocessing Images:

The existing system loads grayscale images of lung scans and preprocesses them using Keras' image module. Images are resized to 150x150 pixels and converted to arrays. The pixel values are normalized to the range [0, 1].

Model Loading and Prediction:

The existing system loads a pre-trained lung cancer detection model (lung_cancer_detection_model.h5) using TensorFlow Keras. It then performs predictions on individual images using the loaded model.

Data Augmentation and Model Training:

The existing system defines paths to image folders for training, validation, and testing.

It uses Keras' ImageDataGenerator to perform real-time data augmentation and normalization.

A CNN model is defined with four convolutional layers followed by max-pooling layers and dense layers for classification.

The model is compiled with binary cross-entropy loss and RMSprop optimizer.

The model is trained on the training data with validation on the validation set for 20 epochs.

Model Evaluation:

After training, the model is evaluated on the test dataset to measure its accuracy.

2.2 PROPOSED SYSTEM:

Model Architecture:

Experiment with different CNN architectures, including deeper networks or using pre-trained models like VGG, ResNet, or DenseNet. These architectures may capture more complex features and improve classification accuracy.

Data Augmentation:

Introduce more data augmentation techniques like rotation, shearing, and zooming to increase the diversity of the training data and improve model generalization.

Hyperparameter Tuning:

Perform hyperparameter tuning to optimize the learning rate, batch size, and other parameters for better model performance.

Transfer Learning:

Fine-tune pre-trained models on larger datasets related to medical imaging, such as the ChestX-ray14 dataset. Transfer learning can leverage features learned from a larger dataset and adapt them to the specific task of lung cancer detection.

Class Imbalance Handling:

If there's a class imbalance in the dataset (e.g., more unaffected cases than affected cases), consider techniques like class weighting or oversampling the minority class to address this issue and prevent bias towards the majority class.

Regularization:

Apply techniques like dropout or L2 regularization to prevent overfitting and improve model generalization.

Model Interpretability:

Incorporate methods for model interpretability, such as gradient-weighted class activation mapping (Grad-CAM), to visualize which parts of the lung scans are most important for the model's predictions.

Deployment and Integration:

Once the model is optimized, consider deploying it as a web or mobile application for real-time lung cancer detection. Integration with healthcare systems or telemedicine platforms could facilitate its use by healthcare professionals.

2.3 SYSTEM CONFIGURATION

2.2.1 HARDWARE SPECIFICATION:

- Processor - I5
- Speed - 3 GHz
- RAM - 8 GB(min)
- Hard Disk - 500 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - LCD,LED

2.2.2 SOFTWARE SPECIFICATION:

- Operating System - Windows/7
- Server - Anaconda, Jupyter
- Front End - tkinter |GUI toolkit
- Server side Scrip - Python

3. SOFTWARE DESCRIPTION

PYTHON:

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence: Machine learning as it has a plethora of libraries implementing machine learning algorithms. Python is a one-stop shop and relatively easy to learn, thus quite popular now.

What other reasons exist for such universal popularity of this programming language and what companies have leveraged its opportunities to the max? Let’s talk about that. Python technology is quite popular among programmers, but the practice shows thatbusiness owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything – from web apps to machine learning. Moreover, it is not just a language but more a technology platformthat has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados. So what is python used for and what are the tangible benefits the language brings to those who decided to use it? Below we’re going to discover that. Productivity and Speed It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++.

The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

PYCHARM:

Pycharm is a dedicated Python Integrated Development Environment (IDE) providinga wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

PyCharm:

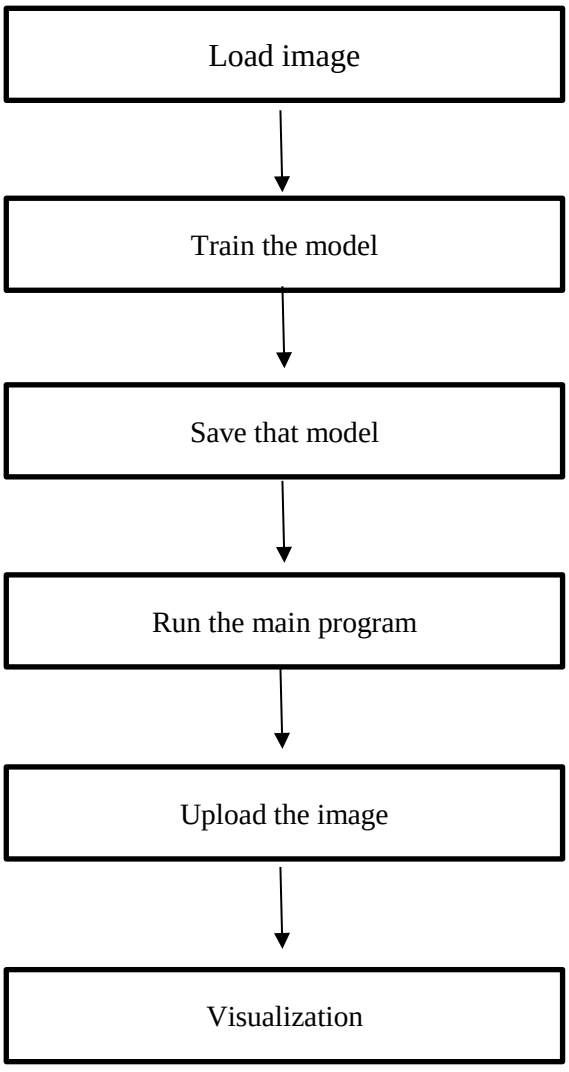
- Professional (paid) : for professional Python, web, and data sciencedevelopment, including code assistance, refactorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.
- Edu (free and open-sourced): for learning programming languages and relatedtechnologies with integrated educational tools.

SUPPORTED PLATFORMS:

Requirement	Minimum	Recommended
RAM	4 GB of free RAM	8 GB of total system RAM
CPU	Any modern CPU	Multi-core CPU. PyCharm supports multithreading for different operations and processes making it faster the more CPU cores it can use.
Disk space	2.5 GB and another 1 GB for caches	SSD drive with at least 5 GBof free space
Monitor resolution	1024x768	1920×1080
Operatingsystem	Officially released 64-bit versions of the following: Microsoft Windows 8 or later macOS 10.13 or later Any Linux distribution that supports Gnome, KDE, or Unity DE. PyCharm is not available for some Linux distributions, suchas RHEL6 or CentOS6, that do not include GLIBC 2.14 or later.	Latest 64-bit version of Windows, macOS, or Linux(for example, Debian, Ubuntu, or RHEL)
	Pre-release versions are not supported.	

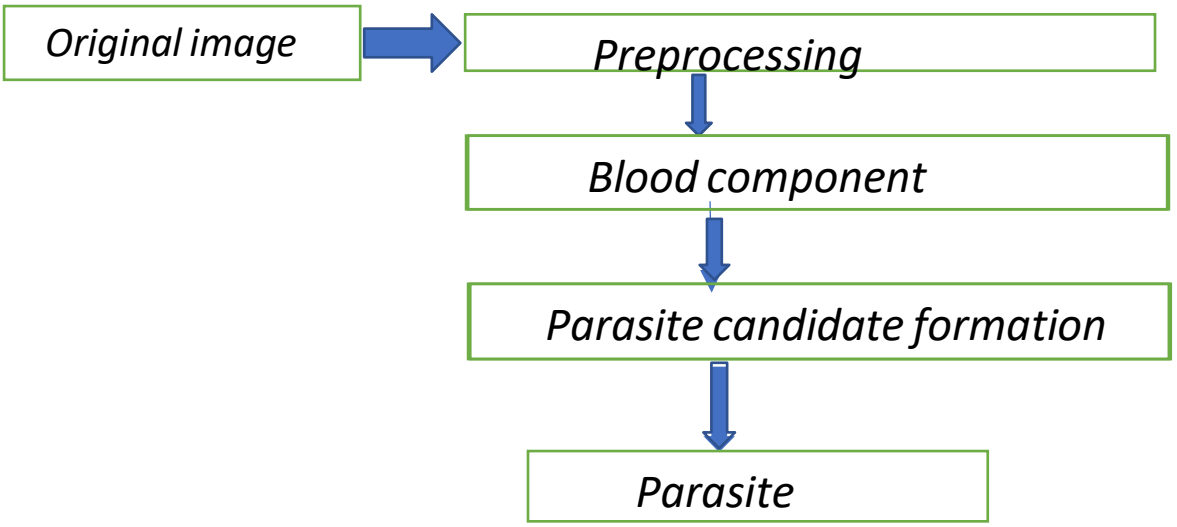
4.SYSTEM DESIGN

DATA FLOW DIAGRAM:



5.SYSTEM IMPLEMENTATION

DETECTION ARCHITECTURE DIAGRAM:



EXPLANATION ABOUT ALGORITHM:

EFFICIENTNET ALGORITHM:

When convolutional neural networks are developed, they are done so at a fixed resource cost. These networks are scaled up later to achieve better accuracies whenmore resources are available. A ResNet 18 model can be scaled up to a ResNet 200 model by adding more layers to the original model. In most situations, this scaling technique has helped provide better accuracies on most benchmarking datasets. Butthe conventional techniques of model scaling are very random. Some models are scaled depth-wise, and some are scaled widthwise. Some models simply take in images of a larger resolution to get better results. This technique of randomly scalingmodels requires manual tuning and many person-hours, often resulting in little or no improvement in performance. The authors of the EfficientNet proposed scaling up CNN models to obtain better accuracy and efficiency in a much more moral way.EfficientNet uses a technique called compound coefficient to scale up models ina simple but effective manner. Instead of randomly scaling up width, depth orresolution, compound scaling uniformly scales each dimension with a certain fixed set of scaling coefficients. Using the scaling method and AutoML, the authors of efficient developed seven models of various dimensions, which surpassed the state-of-the-art accuracy of most convolutional neural networks, and with much better efficiency. EfficientNet is based on the baseline network developed by the neural architecture search using the AutoML MNAS framework. The network is fine-tuned for obtaining maximum accuracy but is also penalized if the network is very computationally heavy. It is also penalized for slow inference time when the networktakes a lot of time to make predictions. The architecture uses a mobile inverted bottleneck convolution similar to MobileNet V2 but is much larger due to the increase in FLOPS. This baseline model is scaled up to obtain the family of EfficientNets.

NEED OF THIS ALGORITHM:

- The Efficient Net models achieve both higher accuracy and better efficiency over existing CNNs, reducing parameter size and FLOPS by an order of magnitude.

- Models such as Efficient Net are particularly useful for using deep learning onthe edge, as it reduces compute cost, battery usage, and also training and inference.

6.SOURCE CODE

Sample code:

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import plotly.express as px

import plotly.graph_objects as go

from plotly.subplots import make_subplots

from sklearn.utils import class_weight

from sklearn.model_selection import train_test_split

import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential

from keras.layers import Conv2D, MaxPool2D,Flatten, Dense, Dropout, BatchNormalization, Input

from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.applications.inception_v3 import InceptionV3

from keras.applications.vgg16 import VGG16

import os

import cv2
```

```
bengin_path = r"Bengin cases"
malignant_path = r"Malignant cases"
normal_path = r"Normal cases"

img_classes = ["Bengin cases", "Malignant cases", "Normal cases"]
path_list = [bengin_path, malignant_path, normal_path]
img_path = []
class_labels = []
for i, dir_list in enumerate(path_list):
    name_img = os.listdir(dir_list)
    for name_file in name_img:
        img = os.path.join(dir_list,name_file)
        img_path.append(img)
        class_labels.append(img_classes[i])

df = pd.DataFrame({"img_path" : img_path,
                   "label" : class_labels})

print(df.head())

for category, group in df.groupby("label"):
    fig, ax = plt.subplots(1,3, figsize = (8,8))
    ax = ax.ravel()
    for i, (_,r) in enumerate(group.sample(3).iterrows()):
        img = cv2.imread(r.img_path)
```

```
        ax[i].imshow(img)
        ax[i].axis("off")
        ax[i].set_title(r.label)
    plt.show()
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
countData = df["label"].value_counts().reset_index()
fig = px.histogram(data_frame = countData, x = "label", y = "count")
fig.show()
sizes = []
resolutions = []
color_distributions = []

train_ratio = 0.70
test_ratio = 0.15
val_ratio = 0.15

df_train, df_test_val = train_test_split(df, train_size = train_ratio, random_state = 42)
df_test, df_val = train_test_split(df_test_val, train_size = test_ratio/(test_ratio+val_ratio), random_state = 42)

print(f"Train shape = {df_train.shape}")
print(f"Test shape = {df_test.shape}")
print(f"Validation shape = {df_val.shape}")
def preprocessing_denoise(img):
```

```
    denoise_img = cv2.medianBlur(img, 1)
    denoise_img = cv2.cvtColor(denoise_img, cv2.COLOR_BGR2RGB)
    return denoise_img
IMG_WIDTH = 224
IMG_HEIGHT = 224

image_size = (IMG_WIDTH, IMG_HEIGHT)
batch_size = 32

TRAIN_DATAGEN = ImageDataGenerator(rescale = 1./255.,
                                   preprocessing_function = preprocessing_denoise,
                                   rotation_range = 30,
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.2,
                                   shear_range = 0.1,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

TEST_DATAGEN = ImageDataGenerator(rescale = 1./255.)

train_generator = TRAIN_DATAGEN.flow_from_dataframe(
    df_train,
    x_col = "img_path",
    y_col = "label",
    target_size = image_size,
```

```
    batch_size = batch_size,
    color_mode = "rgb",
    class_mode = "categorical",
    shuffle = True
)

val_generator = TEST_DATAGEN.flow_from_dataframe(
    df_val,
    x_col = "img_path",
    y_col = "label",
    target_size = image_size,
    batch_size = batch_size,
    color_mode = "rgb",
    class_mode = "categorical",
    shuffle = True
)

test_generator = TEST_DATAGEN.flow_from_dataframe(
    df_test,
    x_col = "img_path",
    y_col = "label",
    target_size = image_size,
    batch_size = batch_size,
    color_mode = "rgb",
    class_mode = "categorical",
```

```
    shuffle = True
)
classes = list(train_generator.class_indices.keys())
classes
#Weight Class
class_weights = class_weight.compute_class_weight(
            class_weight = 'balanced',
             classes = np.unique(train_generator.classes),
             y = train_generator.classes)

train_class_weights = dict(enumerate(class_weights))

for idx, weight, in train_class_weights.items():
    class_name = classes[idx]
    print(f"{class_name} : {weight}")

model1 = Sequential([
    Conv2D(filters = 64, kernel_size = (3,3), strides = (1,1), input_shape = (224, 224, 3),
activation = "relu", padding = "same"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),

    Conv2D(filters = 64, kernel_size = (3,3), strides = (1,1), padding = "valid"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),
```

```
    Conv2D(filters = 128, kernel_size = (3,3), strides = (1,1), padding = "valid"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),

    Conv2D(filters = 128, kernel_size = (3,3), strides = (1,1), padding = "valid"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),

    Conv2D(filters = 256, kernel_size = (3,3), strides = (1,1), padding = "valid"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),

    Conv2D(filters = 256, kernel_size = (3,3), strides = (1,1), padding = "valid"),
    MaxPool2D(pool_size = (2,2)),
    BatchNormalization(),

    Flatten(),
    Dense(4096, activation = "relu"),
    Dropout(0.5),
    Dense(256),
    Dropout(0.25),
    Dense(3, activation = "softmax")
])
```

```
model1.summary()


model1.compile(
    loss = "categorical_crossentropy",
    optimizer = Adam(learning_rate = 0.0005),
    metrics = ["accuracy"],
)


epochs = 50
history = model1.fit(train_generator,
                steps_per_epoch = len(train_generator),
                batch_size = 32,
                validation_data = val_generator,
                validation_steps = len(val_generator),
                class_weight = train_class_weights,
                callbacks=[
                        EarlyStopping(monitor = "val_loss", # watch the val loss metric
                        patience = 5,
                        restore_best_weights = True),
                        ReduceLROnPlateau(monitor='val_loss',    factor=0.2,    patience=4,
mode='min')
                       ],
                epochs = epochs)
def history_plot(epochs, history):
    fig1 = make_subplots()
```

```
    fig1.add_trace(go.Scatter(x = np.arange(1,epochs+1), y = history.history["accuracy"], name = "Training Accuracy"))

    fig1.add_trace(go.Scatter(x = np.arange(1,epochs+1), y = history.history["val_accuracy"], name = "Validation Accuracy"))

    fig1.update_layout(title = "Training and Validation Accuracy", xaxis_title = "Epoch", yaxis_title = "Accuracy")

    fig1.show()


    fig2 = make_subplots()

    fig2.add_trace(go.Scatter(x = np.arange(1,epochs+1), y = history.history["loss"], name = "Training Loss"))

    fig2.add_trace(go.Scatter(x = np.arange(1,epochs+1), y = history.history["val_loss"], name = "Validation Loss"))

    fig2.update_layout(title = "Training and Validation Loss", xaxis_title = "Epoch", yaxis_title = "Loss")

    fig2.show()


history_plot(epochs, history)

def evaluate_model(model, test_generator):

    #Calculate test loss and accuracy

    results = model.evaluate(test_generator, verbose = 0)

    print(f"Test Loss = {results[0]}")

    print(f"Test Accuracy = {results[1]}")

evaluate_model(model1,  test_generator)

model1.save("CNN(Custom).keras")
```

```
base_model = InceptionV3(input_shape = (IMG_WIDTH, IMG_HEIGHT, 3), include_top = False, weights = "imagenet")

for layer in base_model.layers:
    layer.trainable = False

model2 = Sequential()

model2.add(Input(shape = (IMG_WIDTH, IMG_HEIGHT, 3)))

model2.add(base_model)

model2.add(Flatten())

model2.add(Dense(1024, activation = "relu"))

model2.add(Dropout(0.4))

model2.add(Dense(3, activation = "softmax"))


print(model2.summary())


epochs = 50

model2.compile(optimizer = Adam(0.0005),
            loss = "categorical_crossentropy",
            metrics = ["accuracy"])

history = model2.fit(train_generator,
                steps_per_epoch = len(train_generator),
                batch_size = 64,
                validation_data = val_generator,
                validation_steps = len(val_generator),
                class_weight = train_class_weights,
                callbacks=[
```

```
                    EarlyStopping(monitor = "val_loss", # watch the val loss metric
                    patience = 5,
                    restore_best_weights = True),
                    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2,
mode='min')
                   ],
              epochs = epochs)
evaluate_model(model2, test_generator)
base_model.trainable = True

for l in base_model.layers[:10]:
   print(l.name, l.trainable)

model2.compile(optimizer = Adam(0.00001),
            loss = "categorical_crossentropy",
            metrics = ["accuracy"])

print(model2.summary())



epochs = 50
history = model2.fit(train_generator,
              steps_per_epoch = len(train_generator),
              batch_size = 64,
              validation_data = val_generator,
```

```
            validation_steps = len(val_generator),
            class_weight = train_class_weights,
            callbacks=[
                EarlyStopping(monitor = "val_loss", # watch the val loss metric
                patience = 5,
                restore_best_weights = True),
                ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, mode='min')
                ],
            epochs = epochs)
model2.save("InceptionV3.keras")
base_model_vgg16 = VGG16(input_shape = (IMG_WIDTH, IMG_HEIGHT, 3), include_top = False, weights = "imagenet")
for layer in base_model_vgg16.layers:
    layer.trainable = False
model3 = Sequential()
model3.add(Input(shape = (IMG_WIDTH, IMG_HEIGHT, 3)))
model3.add(base_model)
model3.add(Flatten())
model3.add(Dense(1024, activation = "relu"))
model3.add(Dropout(0.4))
model3.add(Dense(3, activation = "softmax"))

print(model3.summary())
```

```
model3.compile(optimizer = Adam(0.0005),
            loss = "categorical_crossentropy",
            metrics = ["accuracy"])


epochs = 50
history = model3.fit(train_generator,
                steps_per_epoch = len(train_generator),
                batch_size = 64,
                validation_data = val_generator,
                validation_steps = len(val_generator),
                class_weight = train_class_weights,
                callbacks=[
                        EarlyStopping(monitor = "val_loss", # watch the val loss metric
                        patience = 5,
                        restore_best_weights = True),
                        ReduceLROnPlateau(monitor='val_loss',  factor=0.2,  patience=4,
mode='min')
                       ],
                epochs = epochs)
base_model_vgg16.trainable = True
set_trainable = False
for layer in base_model_vgg16.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
```

```
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False


model3.compile(optimizer = Adam(0.00001),
               loss = "categorical_crossentropy",
               metrics = ["accuracy"])
epochs = 50
history = model3.fit(train_generator,
                     steps_per_epoch = len(train_generator),
                     batch_size = 64,
                     validation_data = val_generator,
                     validation_steps = len(val_generator),
                     class_weight = train_class_weights,
                     callbacks=[
                         EarlyStopping(monitor = "val_loss", # watch the val loss metric
                         patience = 5,
                         restore_best_weights = True),
                         ReduceLROnPlateau(monitor='val_loss',  factor=0.2,  patience=4,
mode='min')
                        ],
                     epochs = epochs)
model3.save("VGG16.keras")
from IPython.display import FileLink
```

FileLink(r'VGG16.keras')

FILE2.PY:

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define constants
IMAGE_WIDTH = 128
IMAGE_HEIGHT = 128
BATCH_SIZE = 32
NUM_CLASSES = 2
NUM_EPOCHS = 10

# Data preparation
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
import os
# Get the current working directory
current_dir = os.getcwd()
# Construct the path to the images directory
images_dir = os.path.join(current_dir, 'images')
```

```
# Data preparation
train_generator = train_datagen.flow_from_directory(
    directory=images_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    directory=images_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)

# Check if any samples are found
if train_generator.samples == 0:
    raise ValueError("No samples found in the training directory.")

# Define model architecture
model = Sequential([
```

```
    Conv2D(32,(3,3),activation='relu',input_shape=(IMAGE_WIDTH,
IMAGE_HEIGHT,3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(NUM_CLASSES, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=NUM_EPOCHS,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE
)
```

```
# Evaluate the model
test_loss, test_acc = model.evaluate(validation_generator, steps=validation_generator.samples // BATCH_SIZE)
print('Test accuracy:', test_acc)

# Save the model
model.save('lung_cancer_detection_model.h5')
```

7.SYSTEM TESTING

TESTINGSYSTEM TESTING AND MAINTENANCE:

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently.

In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

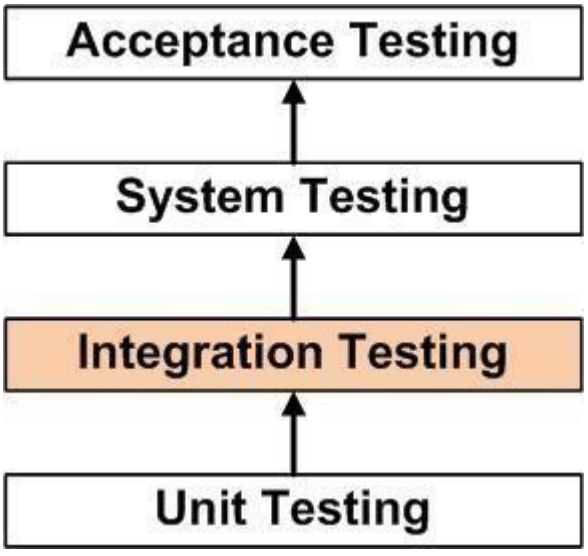
Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non- testing leads to errors that may appear few months later.

This will create two problems, Time delay between the cause and appearanceof the problem. The effect of the system errors on files and records within the system. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring thatall the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the errorsoccurred are noted and eliminated. This is the final step in system life cycle. Here weimplement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors likeruntime errors, long run errors and other maintenances like table verification and reports.

Integration Testing is a level of software testing where individual units are combined and tested as a group.



The purpose of this level is to expose faults in the interaction between integrated units.

Test drivers and test stubs are used to assist in Integration testing.

METHOD:

Any of Black Box Testing, White Box Testing, and Gray Box Testing methods can be used. Normally, the method depends on your definition of 'unit'.

TASKS:

- Integration Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test Cases/Scripts
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test
  - Perform

UNIT TESTING:

Unit testing verification efforts on the smallest unit of software design, module. This is known as "Module Testing". The modules are tested separately. Thistesting is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

BLACK BOX TESTING:

Black box testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usuallyfunctional.

WHITE-BOX TESTING:

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software thattests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

GREY BOX TESTING:

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. To test the Web Servicesapplication usually the Grey box testing is used.

INTEGRATION TESTING:

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole.

In the integration-testing step, all the error uncovered is corrected for the nexttesting steps.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failurescaused by interface defects.

The task of the integration test is to check that components orsoftware applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

ACCEPTANCE TESTING:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

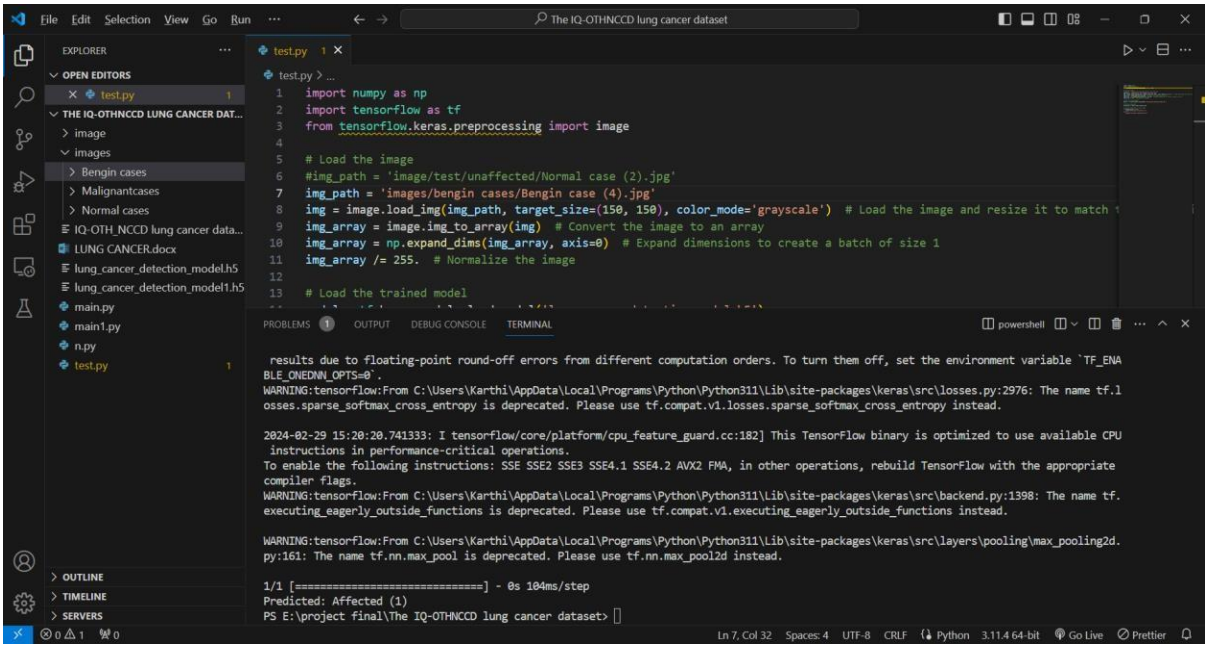
Acceptance testing for Data Synchronization:

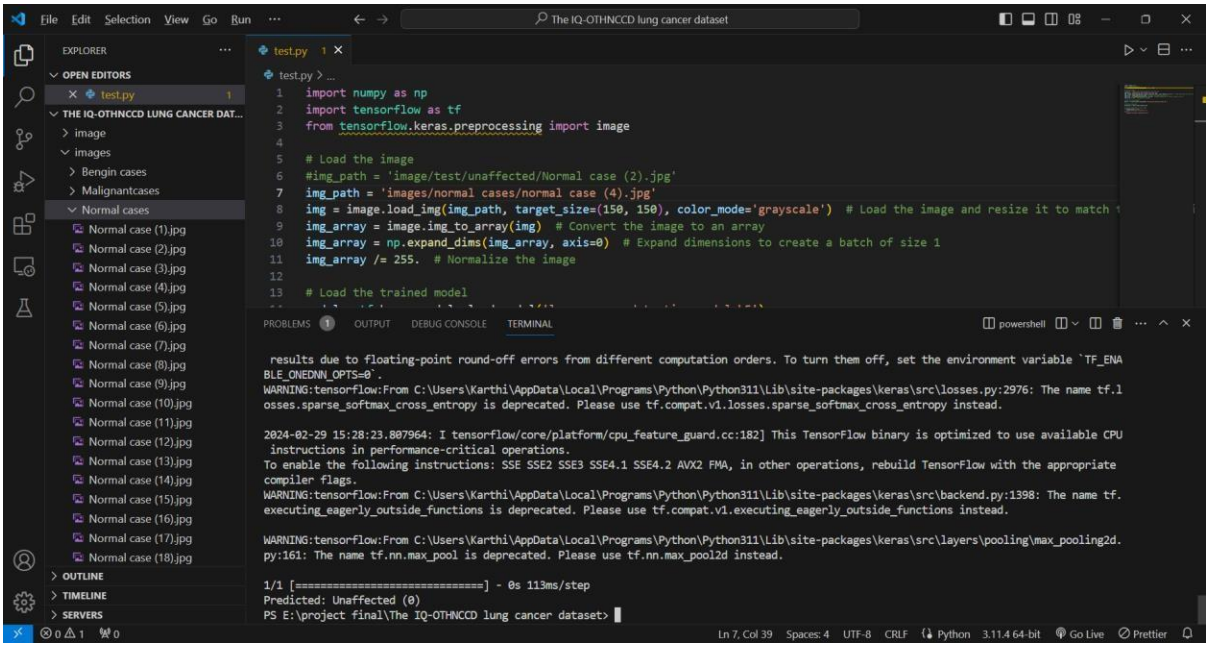
- ❖ The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- ❖ The Route add operation is done only when there is a Route request in need
- ❖ The Status of Nodes information is done automatically in the Cache Updatingprocess

BUILD THE TEST PLAN:

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be uses.

8.SCREEN LAYOUT





9.CONCLUSION

Conclusion:

As mentioned above, this paper has successfully fulfilled its goals with an important contribution over existing machine learning technology where ELM outperforms all existing models on malaria RBC disease prediction with an accuracy of 99.0 and within a short time used of 28 seconds. ELM algorithm is one of the most efficient ML algorithms in the neural networks' world and it works on very large datasets. Because of the non-iterative training all the parameters are tuned once, and this results in a high training speed. Its implementation is easy to understand, and it can be used to solve complex problems in related area. Within the advantages of ELM and its exceedingly performance compared with other ML, its proof that ELM is highly recommend being used in Malaria RBCs classification and prediction. This paper focused on improving existing models and results, which leads to recommend the next research to differentiate plasmodium of malaria RBCs with other Hyperparasitemia located in RBCs in terms of malaria disease level and trace as wellthe malaria growth by considering RBCs image per patient not in general, for the future work.

Future Enhancement:

Early and effective malaria diagnosis is vital to control the disease spread and to prevent the emergence of severe cases and death. Currently, malaria diagnosis relieson optical microscopy and immuno-rapid tests; however, these require a drop of blood, are time-consuming, or are not specific ınd sensitive enough for reliable detection of low-level parasitaemia. Thus, there is an urge forsimpler, prompt, and accurate alternative diagnostic methods. Particularly, hemozoin has beenincreasingly recognized as an attractive biomarker for malaria detection. As the disease prolifera es, parasites digest host hemoglobin, in the process releasing toxic haem that is detoxified into an insoluble crystal, the hemozoin, which accumulates along with infection progression. Given its magnetic, optical, and acoustic unique features, hemozoin has been explored for new label-free diagnostic methods.

Thereby, herein, we review the hemozoin-based malaria detection methods andcritically discuss their challenges and potential for the development of an ideal diagnostic device.

BIBLIOGRAPHY

Reference:

[1] N. Abbas, T. Saba, D. Mohamad, A. Rehman, A. Almazyad, and J. Saleh Al-Ghamdi,"Machine aided malaria parasitemia detection in giemsastained thin blood smears," Neural Computing and Applications, vol. 29, no. 3, p. 803–818, 2016.

[2] A. Capone, I. Ricci, C. Damiani, M. Mosca, P. Rossi, P. Scuppa, E. Crotti, S. Epis, M.Angeletti, M. Valzano, L. Sacchi, C. Bandi, D. Daffonchio, M. Mandrioli, and G. Favia,"Interactions between asaia, plasmodium and anopheles: New insights into mosquito symbiosis and implications in malaria symbiotic control," Parasites & vectors, vol. 6, p. 182, 2013.

[3] I. Bates, V. Bekoe, and A. Asamoa-Adu, "Improving the accuracy of malaria-relatedlaboratory tests in ghana," Malaria journal, 12 2004.

[4] F. T. Boray, G. D. Andrew, and K. Izzet, "Computer vision for microscopy diagnosisof malaria," Malaria Journal, pp. 1–14, 2009.

[5] S. Akanksha, S. Sini, and D. Shatendra, "Recent image enhancement techniques: A review," International Journal of Engineering and Advanced Technology (IJEAT), vol.4, no. 1, p. 40–45, 2014.

[6] S. Jalari, E. B, K. Reddy, and C.-H. Lai, "An image processing approach for accurate determination of parasitemia in peripheral blood smear images," International Journal pf Computer Applications, vol. 1, pp. 23–28, 2011.

[7] O. Nobuyuki, "A threshold selection method from gray-level histograms," IEEETransactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, 1979.

[8] S. N. Mark and S. A. Alberto, "Feature extraction and image processing," The ACMDigital Library, 2008.

[9] L. Zhaohui, P. Andrew, E. IIker, P. Mahdieh, S. Kamolrat, P. Kannappan, G. Peng, H.Md Amir, M. Richard, James, H. Jimmy, Xiangji, J. Stefan, and T. George, "Cnn-based image analysis for malaria diagnosis," in 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2016.

[10] D. Yuhang, J. Zhuocheng, S. Hongda, and P. W. David, "Classification accuraciesof malaria infected cells using deep convolutional neural networks based on decompressed images," in SoutheastCon 2017. IEEE, 2017.