Cancer Image Analysis and Detection Tutorial

Step 1: Importing Required Libraries

The first step is importing the necessary Python libraries. Here's what each library does:

import cv2 # For image processing
import numpy as np # For numerical computations
from tensorflow.keras.models import Sequential # For building the CNN model
from tensorflow.keras.layers import (Conv2D, MaxPooling2D, Flatten, Dense) # CNN layers
import matplotlib.pyplot as plt # For plotting and visualization

Explanation:

- 1. OpenCV (cv2): Used to handle image input, preprocessing, and visualizing results.
- 2. NumPy (np): Essential for handling arrays and numerical computations.
- 3. Keras Sequential Model: Used to define the Convolutional Neural Network (CNN) model architecture.
- 4. Matplotlib: For visualizing the input image, results, and bounding boxes on the detected regions.

Step 2: Downloading Pretrained Model

The pretrained model weights (model.h5) are downloaded using the following code:

!gdown --id 1L5TMRMn78eTn7Z1joS2vfCCtadCKG4Mu

Explanation:

- gdown: A Python package to download files directly from Google Drive using the file ID.
- The file ID is a unique identifier in the URL of the Google Drive file.

```
Downloading pretrained model:
//usr/local/lib/python3-le/dist-packages/gdown/_main_.py:140: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anymor
warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1-yatFvMpokK7fHaaaAM02Tkv]gkUnXhw
To: /content/model.h5
Downloading...
```

Step 3: Creating and Compiling the CNN Model

A simple CNN architecture is created and compiled.

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(500, 500, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid') # Output for binary classification
])
```

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

Explanation:

- 1. Input Layer: Accepts RGB images of size 500x500x3.
- 2. Convolutional Layers: Extract features from the input image.
- Conv2D(32, (3, 3)): 32 filters of size 3x3 are used to extract features.
- 3. Pooling Layers: Reduces dimensionality while retaining important features.
- 4. Flatten Layer: Converts the output into a 1D vector.
- 5. Dense Layers: Fully connected layers for binary classification.
- 6. Output Layer: Uses sigmoid activation for binary output (e.g., tumor or no tumor).
- 7. Compilation:
- Optimizer: rmsprop optimizes the model during training.
- Loss Function: binary_crossentropy is used since this is a binary classification problem.

Step 4: Loading Pretrained Weights

The trained weights are loaded into the model:

model.load_weights('model.h5')

Explanation:

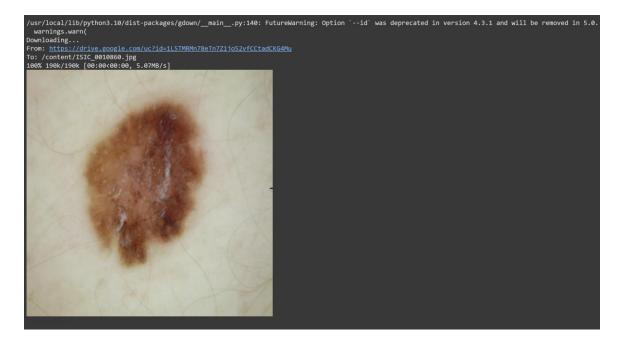
The model.h5 file contains the trained model's weights, enabling inference without retraining the model.

Step 5: Reading and Preprocessing the Image

An input image is read, resized, and preprocessed for both bounding box detection and classification.

Explanation:

- 1. Grayscale Conversion: Simplifies the image for thresholding.
- 2. Gaussian Blur: Reduces noise and smooths the image.
- 3. Otsu's Thresholding: Automatically determines the optimal threshold value for segmentation.



Step 6: Detecting Tumor Regions

Contours are detected from the binary image (thresholded result) to identify tumor regions.

 $contours, _= cv2. find Contours (thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) \\ for contour in contours:$

```
x, y, w, h = cv2.boundingRect(contour) # Get bounding box coordinates cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2) # Draw rectangles
```

Explanation:

- 1. Contours: Extracts boundaries of segmented regions.
- 2. Bounding Rectangles: For each contour, a bounding box is drawn on the original image to

highlight the detected regions.

