Capstone Project By LetsUpgrade

CT Scan Image Classification

Introduction

The aim of this project is to develop an artificial intelligence model capable of identifying SARS-CoV-2 infection through the analysis of CT scans. The dataset consists of 1252 CT scans positive for COVID-19 and 1230 CT scans for non-infected patients, totaling 2482 CT scans collected from hospitals in Sao Paulo, Brazil.

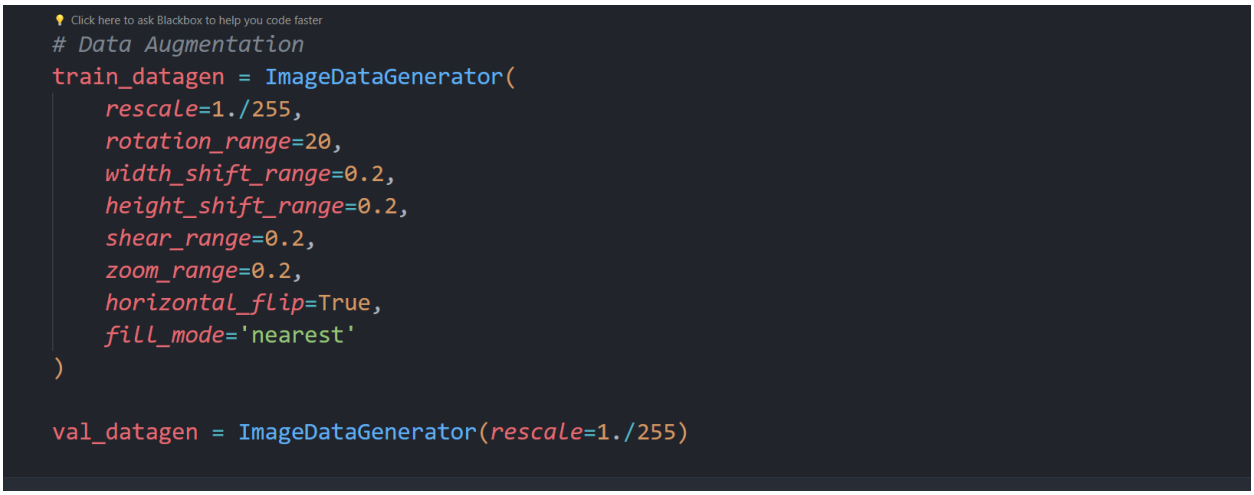
Data Preparation

1. Downloading the Data The dataset was obtained from the provided Google Drive link, containing positive and negative CT scans for SARS-CoV-2 infection.
Data Set : https://drive.google.com/drive/folders/1WOeodRmv1Mw5Cswuip3nUIi6ViQWKpo_

2. Preprocessing CT scan images were loaded and resized to a fixed size of 180x180 pixels..

Data Augmentation

Data augmentation was applied using the Keras ImageDataGenerator to increase the diversity of the training set. The following augmentation techniques were employed: Rescale Shear Zoom Horizontal Flip Width Shift Range Height Shift Range



Model Building

1. Choosing ResNet Model ResNet-50, a pre-trained model on ImageNet, was chosen as the base model for feature extraction.

```
Click here to ask Blackbox to help you code faster
res50 = ResNet50(input_shape = (180, 180, 3), weights = 'imagenet', include_top = False)

Click here to ask Blackbox to help you code faster
for layer in vgg16.layers:
  layer.trainable = False

Click here to ask Blackbox to help you code faster
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization
```

2. Modifying the Model The top layers were removed, and a new classification head suitable for binary classification was added. Pre-trained ResNet layers were frozen to retain learned features.

```
Click here to ask Blackbox to help you code faster
# Fully Connected Layers

flatten = Flatten()(res50.output)

dense = Dense(512, activation = 'relu')(flatten)
dense = Dropout(0.5)(dense)
dense = Dense(128, activation = 'relu')(dense)
dense = Dropout(0.3)(dense)

# Output Layer
prediction = Dense(1, activation = 'sigmoid')(dense)
```

Python

3. Model Compilation The model was compiled using binary crossentropy loss and an optimizer(adam) .

```
Click here to ask Blackbox to help you code faster
model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

Click here to ask Blackbox to help you code faster
from tensorflow.keras.callbacks import EarlyStopping

Click here to ask Blackbox to help you code faster
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,  # Number of epochs with no improvement after which training will be stopped
    restore_best_weights=True
)
```

Model Training

1. Callbacks EarlyStopping were employed during training to prevent overfitting and save the best model.

2. Training Procedure The model was trained on the augmented dataset with a specified number of epochs, utilizing the defined callbacks.

In [57]:
```
history1 = model1.fit(
    train_generator,
    epochs=15,
    validation_data=validation_generator,
    callbacks=[early_stopping],

)
```

```
Epoch 1/15
62/62 [==============================] - 25s 405ms/step - loss: 0.1722 - accuracy: 0.9395 - val_loss: 2.3889 - val_accuracy: 0.
8974
Epoch 2/15
62/62 [==============================] - 25s 395ms/step - loss: 0.1516 - accuracy: 0.9435 - val_loss: 0.3786 - val_accuracy: 0.
8511
Epoch 3/15
62/62 [==============================] - 24s 387ms/step - loss: 0.1548 - accuracy: 0.9430 - val_loss: 0.8187 - val_accuracy: 0.
8692
Epoch 4/15
62/62 [==============================] - 24s 391ms/step - loss: 0.1423 - accuracy: 0.9465 - val_loss: 0.5039 - val_accuracy: 0.
9034
Epoch 5/15
62/62 [==============================] - 25s 396ms/step - loss: 0.1534 - accuracy: 0.9435 - val_loss: 0.3817 - val_accuracy: 0.
9276
```

Model Evaluation

The best model was loaded, and predictions were made on the test set.
1. Performance Metrics Performance metrics such as accuracy, precision, recall, F1 Score, and a Confusion Matrix were calculated to assess the model's effectiveness. Results and Conclusion **The model achieved 85%**. The project demonstrated the potential of AI in identifying SARS-CoV-2 infections from CT scans.

In [59]:
```
test_results = model1.evaluate(validation_generator)
print(f"Test Loss: {test_results[0]}, Test Accuracy: {test_results[1]}")
```

```
16/16 [==============================] - 2s 144ms/step - loss: 0.3786 - accuracy: 0.8511
Test Loss: 0.37858906388282776, Test Accuracy: 0.8511066436767578
```

here train accuray = accuracy: 0.8511

here train accuray = test acc: 0.8511066436767578

after performing many attempts our model works with 85% accuracy it is fine to perform image classification

Model Saving

Saving the model for further use

In [80]:
```
model1.save("Ct_Scan_Covid.h5")
```

In [87]:
```
# library to load the model
from tensorflow.keras.models import load_model
```

Loading the Model to Check the Saved model

In [89]:
```
s_model = load_model('/kaggle/working/Ct_Scan_Covid.h5')
```

Tech Stack :

The technology stack for a project like CT Scan Image Classification involves various tools and libraries for data preprocessing, deep learning, and evaluation.

Programming Language: Python

Libraries and Frameworks: Deep Learning Framework: Keras or TensorFlow

OpenCV: For image processing tasks, such as loading and resizing images.

Data Manipulation and Analysis: NumPy: For numerical operations on arrays and matrices. Pandas: For data manipulation and analysis.

Data Augmentation: Keras ImageDataGenerator: For augmenting images during training. Model Evaluation: scikit-learn: For calculating performance metrics like accuracy, precision, recall, and F1 score.

Matplotlib and Seaborn: For data visualization.

Pre-trained Models: ResNet: A pre-trained ResNet model, such as ResNet-50, which can be loaded using Keras applications module.

Development Environment: IDE: Jupyter Notebooks and colab ,Spyder,Kaggle

Version Control: Git: For version control and collaboration.

GitHub: For code hosting, version control, and collaboration.

Model Deployment : Streamlit