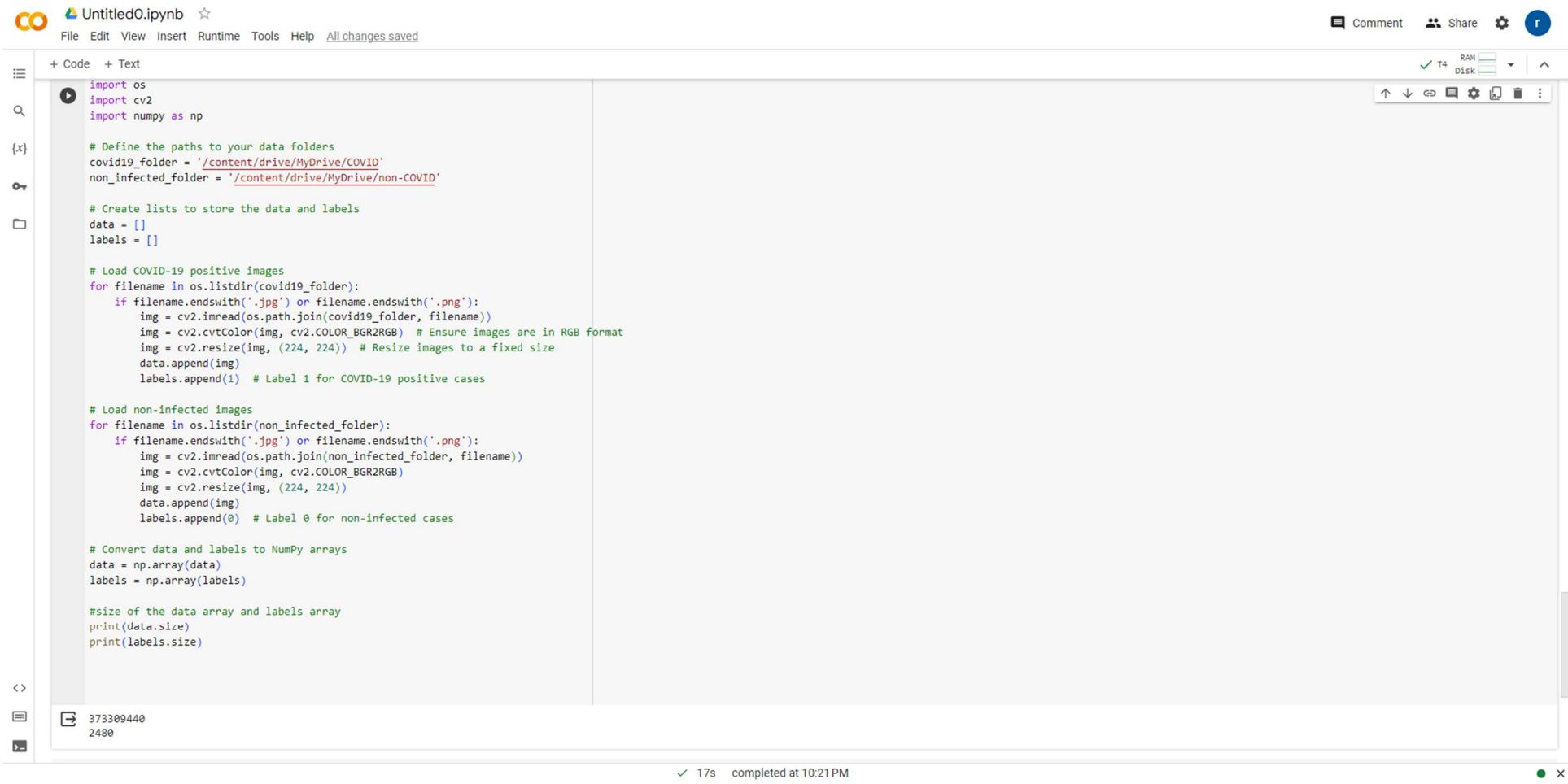
FIRST SOLUTION CODE COMMIT:



EXTENDED DESCRIPTION :

Here I am committing

1. Imports:

- Imported the necessary Python modules

2. Data Preparation:

- Load the data images folder, which contains 1252 CT scans of COVID-19-positive patients and 1230 CT scans of non-infected patients.

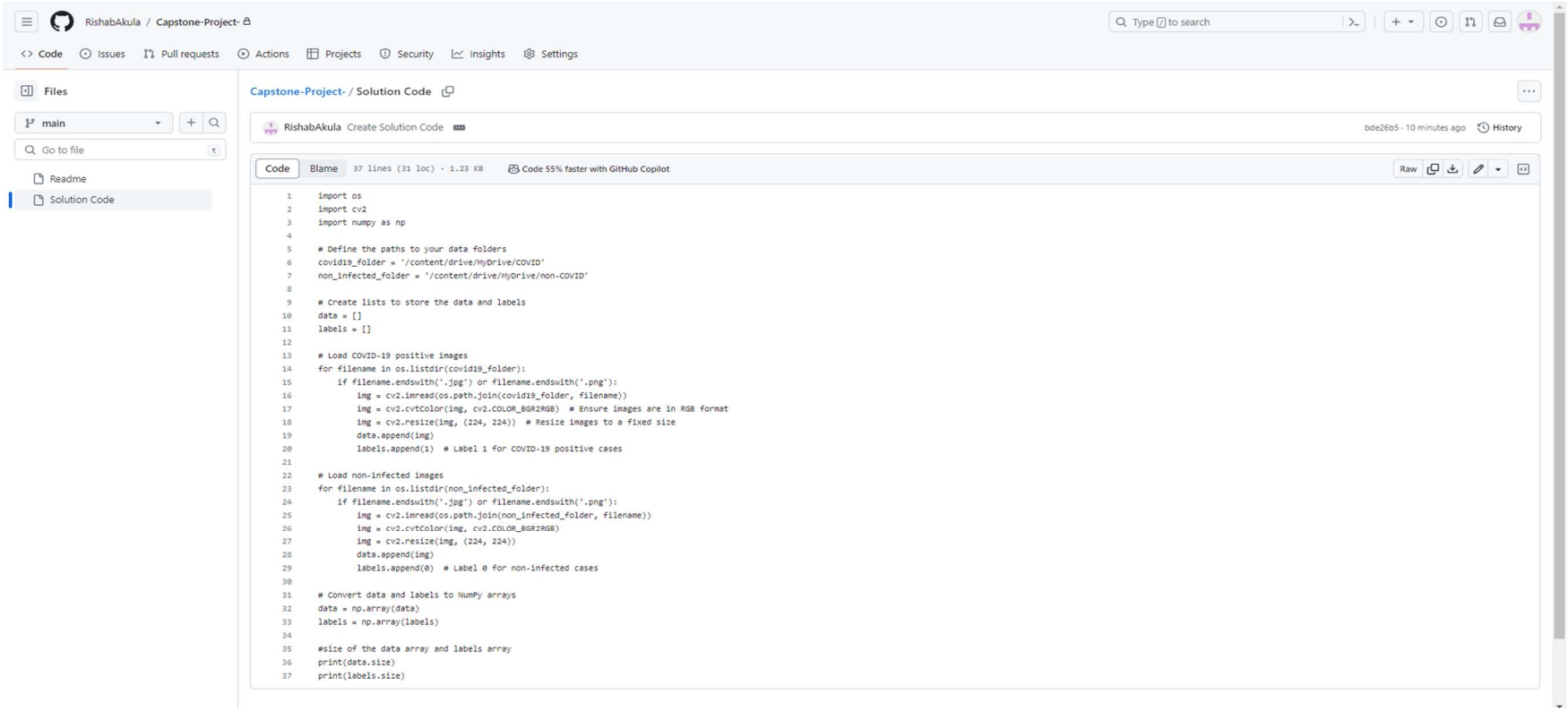
3. Load the data sets:

- Store data of both the data sets in the form of a data list and add label for each Covid-19 Positive image as 1 and non Covid 19 images as 0,and add into label list.

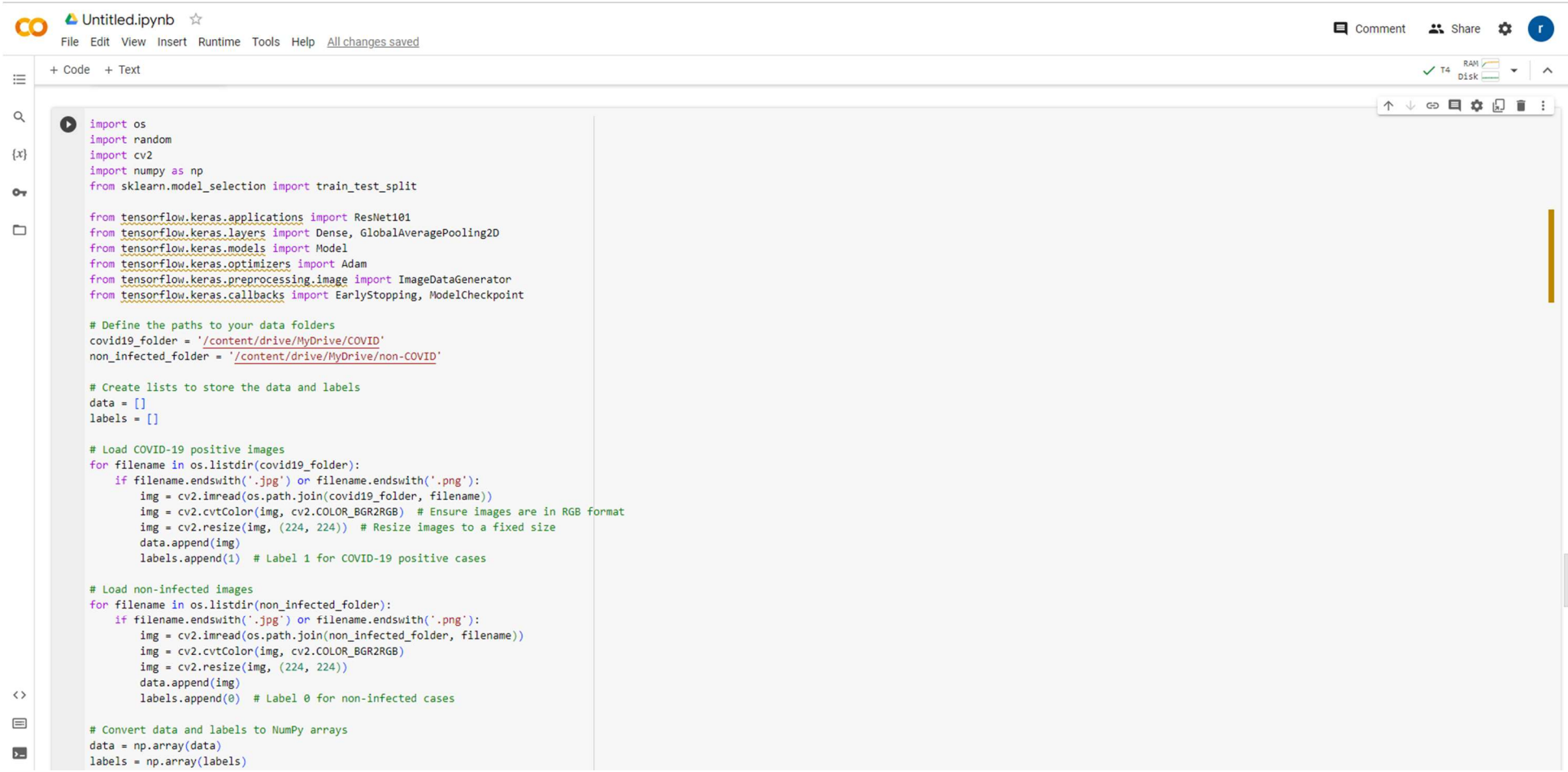
- Convert the data and labels list into NumPy Arrays

- Verified the size of the Data array and Labels array

GITHUB UPLOADS :



SECOND SOLUTION CODE COMMIT:



Untitled.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share r

+ Code + Text

T4 RAM Disk

```
# Convert data and labels to NumPy arrays
data = np.array(data)
labels = np.array(labels)

#size of the data array and labels array
#print(data.size)
#print(labels.size)

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(data, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Define a list of ResNet architectures
#resnet_architectures = [ResNet18, ResNet34, ResNet50, ResNet101, ResNet152]
resnet_architectures = [ResNet101]

for resnet_model in resnet_architectures:
    # Load pre-trained ResNet model
    base_model = resnet_model(weights='imagenet', include_top=False)

    # Add custom top layers for binary classification
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    # Create data generators with data augmentation
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        horizontal_flip=True
    )

    # Set up Early Stopping and Model Checkpoint callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
    model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)
```

Untitled.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

T4 RAM Disk

```
# Set up Early Stopping and Model Checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)

# Train the model
history = model.fit_generator(
    train_datagen.flow(X_train, y_train, batch_size=32),
    validation_data=(X_val, y_val),
    epochs=100,
    callbacks=[early_stopping, model_checkpoint]
)

# Load the best model
model.load_weights(f'best_model_{resnet_model.__name__}.h5')

# Evaluate the model on the test set
test_predictions = model.predict(X_test)
test_predictions = (test_predictions > 0.5).astype(int)

# Selected Model Name
print(f"Model: {resnet_model.__name__}")
```

```
<ipython-input-2-707f49d8fbb8>:86: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
Epoch 1/100
55/55 [==============================] - ETA: 0s - loss: 0.3197 - accuracy: 0.8606
Epoch 1: val_loss improved from inf to 0.50644, saving model to best_model_ResNet101.h5
55/55 [==============================] - 119s 1s/step - loss: 0.3197 - accuracy: 0.8606 - val_loss: 0.5064 - val_accuracy: 0.7473
Epoch 2/100
55/55 [==============================] - ETA: 0s - loss: 0.1971 - accuracy: 0.9217
Epoch 2: val_loss did not improve from 0.50644
55/55 [==============================] - 37s 674ms/step - loss: 0.1971 - accuracy: 0.9217 - val_loss: 1.8342 - val_accuracy: 0.5188
Epoch 3/100
55/55 [==============================] - ETA: 0s - loss: 0.1237 - accuracy: 0.9551
Epoch 3: val_loss did not improve from 0.50644
55/55 [==============================] - 37s 659ms/step - loss: 0.1237 - accuracy: 0.9551 - val_loss: 1.9892 - val_accuracy: 0.4919
Epoch 4/100
55/55 [==============================] - ETA: 0s - loss: 0.1174 - accuracy: 0.9603
Epoch 4: val_loss did not improve from 0.50644
55/55 [==============================] - 38s 686ms/step - loss: 0.1174 - accuracy: 0.9603 - val_loss: 0.8140 - val_accuracy: 0.4409
Epoch 5/100
55/55 [==============================] - ETA: 0s - loss: 0.0872 - accuracy: 0.9712
Epoch 5: val_loss did not improve from 0.50644
55/55 [==============================] - 37s 665ms/step - loss: 0.0872 - accuracy: 0.9712 - val_loss: 6.2644 - val_accuracy: 0.4946
```

EXTENDED DESCRIPTION:

Here I am updating:

1. Imports:

- Import the necessary Python modules and ResNet models for training and prediction.

2. Split the Data:

- Split the dataset into training, validation, and test sets. (i.e. 70-20-10).

3. Load the pre-trained ResNet model :

- Load the pre-trained ResNet model using the ResNet architectures (ResNet101)

. 4. Model Building : -

Modify the ResNet model to suit your binary classification task (i.e. COVID-19 detection). You may want to add additional fully connected layers on top of the ResNet model for fine-tuning.

- Compile the model using any Deep learning model such as Adam Optimization algorithm.

5. Data Augmentation :

- Apply data augmentation to improve the model's generalization.

- You can use the `ImageDataGenerator` from Keras with various augmentation parameters, such as :

- `width_shift_range`

- `height_shift_range`

- `rotation_range`

- `shear_range`

- `zoom_range`.

6. Early Stopping and ModelCheckpoint :

- Implement early stopping and model checkpoint callbacks during training.

- Early stopping helps prevent overfitting, and ModelCheckpoint saves the best model during training.

7. Training the model :

- Train the model using the augmented training dataset.

- Compile the model with appropriate loss function and optimizer for binary classification.

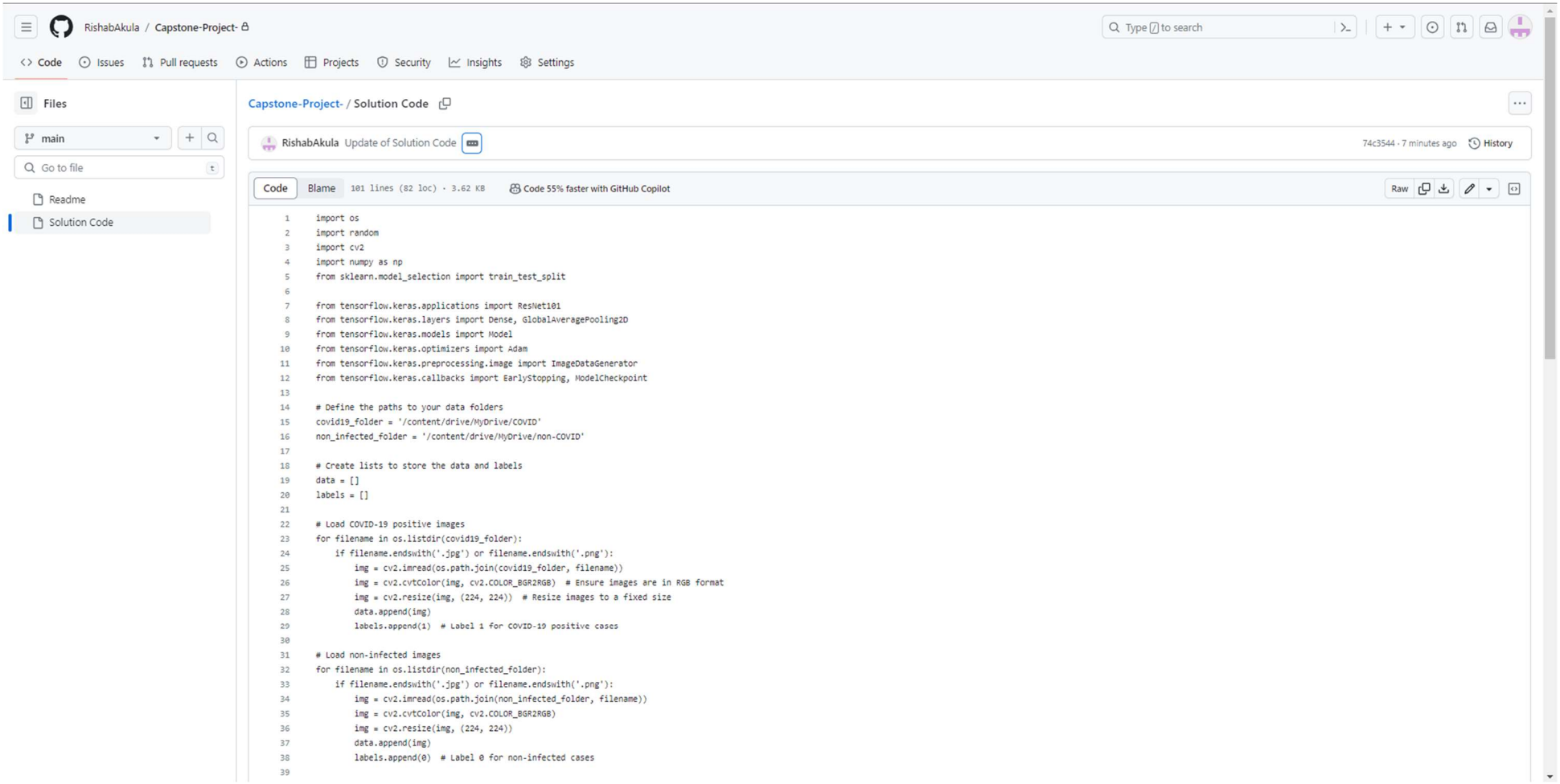
8. Model Evaluation :

- Load the ResNet model. - Using the ResNet model predict weather the test image is Covid-19 positive or non-Covid-19.

9. Selected model name : -

print the selected model name .

GITHUB UPLOADS :



RishabAkula / Capstone-Project-

Type / to search

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

Files

main

Go to file t

Readme

Solution Code

Capstone-Project- / Solution Code

RishabAkula Update of Solution Code

74c3544 · 7 minutes ago History

Code | Blame | 101 lines (82 loc) · 3.62 KB | Code 55% faster with GitHub Copilot | Raw

```
import os
import random
import cv2
import numpy as np
from sklearn.model_selection import train_test_split

from tensorflow.keras.applications import ResNet101
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Define the paths to your data folders
covid19_folder = '/content/drive/MyDrive/COVID'
non_infected_folder = '/content/drive/MyDrive/non-COVID'

# Create lists to store the data and labels
data = []
labels = []

# Load COVID-19 positive images
for filename in os.listdir(covid19_folder):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        img = cv2.imread(os.path.join(covid19_folder, filename))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Ensure images are in RGB format
        img = cv2.resize(img, (224, 224))  # Resize images to a fixed size
        data.append(img)
        labels.append(1)  # Label 1 for COVID-19 positive cases

# Load non-infected images
for filename in os.listdir(non_infected_folder):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        img = cv2.imread(os.path.join(non_infected_folder, filename))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (224, 224))
        data.append(img)
        labels.append(0)  # Label 0 for non-infected cases

```

Capstone-Project- / Solution Code

↑ Top

Code | Blame | 101 lines (82 loc) · 3.62 KB | Code 55% faster with GitHub Copilot | Raw

```
            labels.append(0)  # Label 0 for non-infected cases

# Convert data and labels to NumPy arrays
data = np.array(data)
labels = np.array(labels)

#size of the data array and labels array
#print(data.size)
#print(labels.size)

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(data, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Define a list of ResNet architectures
#resnet_architectures = [ResNet18, ResNet34, ResNet50, ResNet101, ResNet152]
resnet_architectures = [ResNet101]

for resnet_model in resnet_architectures:
    # Load pre-trained ResNet model
    base_model = resnet_model(weights='imagenet', include_top=False)

    # Add custom top layers for binary classification
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    # Create data generators with data augmentation
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        horizontal_flip=True
    )

    # Set up Early Stopping and Model Checkpoint callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
    model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)

    # Train the model
```

Files

main

Go to file t

Readme

Solution Code

Capstone-Project- / Solution Code

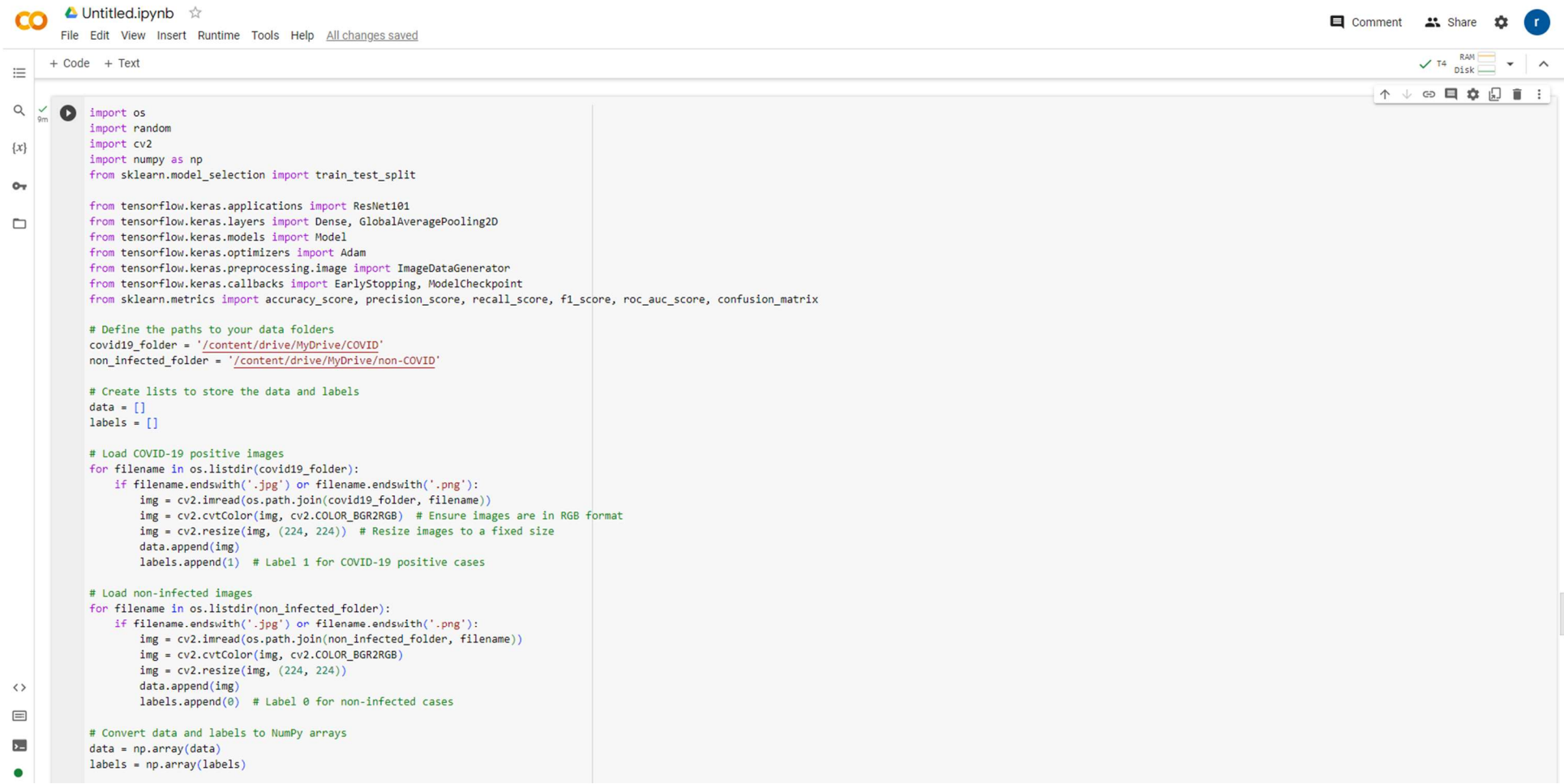
Top

Code Blame 101 lines (82 loc) · 3.62 KB Code 55% faster with GitHub Copilot

Raw

```
for resnet_model in resnet_architectures:
    # Load pre-trained ResNet model
    base_model = resnet_model(weights='imagenet', include_top=False)

    # Add custom top layers for binary classification
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    # Create data generators with data augmentation
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        horizontal_flip=True
    )

    # Set up Early Stopping and Model Checkpoint callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
    model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)

    # Train the model
    history = model.fit_generator(
        train_datagen.flow(X_train, y_train, batch_size=32),
        validation_data=(X_val, y_val),
        epochs=100,
        callbacks=[early_stopping, model_checkpoint]
    )

    # Load the best model
    model.load_weights(f'best_model_{resnet_model.__name__}.h5')

    # Evaluate the model on the test set
    test_predictions = model.predict(X_test)
    test_predictions = (test_predictions > 0.5).astype(int)

    # Selected Model Name
    print(f"Model: {resnet_model.__name__}")
```

FINAL SOLUTION CODE :



Untitled.ipynb

File Edit View Insert Runtime Tools Help All changes saved

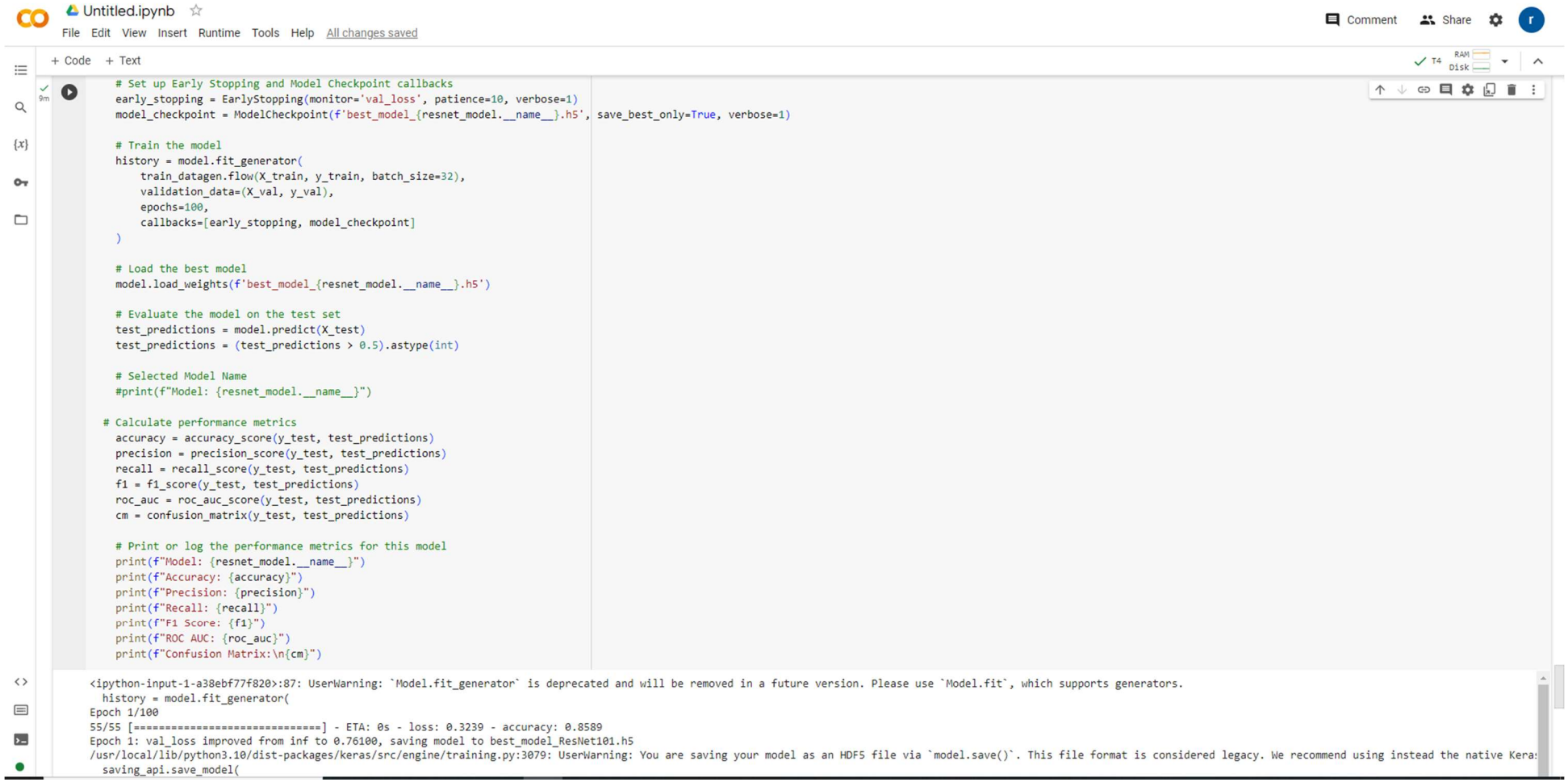
Comment Share

+ Code + Text

T4 RAM Disk

9m

```
# Convert data and labels to NumPy arrays
data = np.array(data)
labels = np.array(labels)

#size of the data array and labels array
#print(data.size)
#print(labels.size)

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(data, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Define a list of ResNet architectures
#resnet_architectures = [ResNet18, ResNet34, ResNet50, ResNet101, ResNet152]
resnet_architectures = [ResNet101]

for resnet_model in resnet_architectures:
    # Load pre-trained ResNet model
    base_model = resnet_model(weights='imagenet', include_top=False)

    # Add custom top layers for binary classification
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    # Create data generators with data augmentation
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        horizontal_flip=True
    )

    # Set up Early Stopping and Model Checkpoint callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
    model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)
```


Untitled.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
T4
RAM
Disk
# Set up Early Stopping and Model Checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)
# Train the model
history = model.fit_generator(
train_datagen.flow(X_train, y_train, batch_size=32),
validation_data=(X_val, y_val),
epochs=100,
callbacks=[early_stopping, model_checkpoint]
)
# Load the best model
model.load_weights(f'best_model_{resnet_model.__name__}.h5')
# Evaluate the model on the test set
test_predictions = model.predict(X_test)
test_predictions = (test_predictions > 0.5).astype(int)
# Selected Model Name
#print(f"Model: {resnet_model.__name__}")
# Calculate performance metrics
accuracy = accuracy_score(y_test, test_predictions)
precision = precision_score(y_test, test_predictions)
recall = recall_score(y_test, test_predictions)
f1 = f1_score(y_test, test_predictions)
roc_auc = roc_auc_score(y_test, test_predictions)
cm = confusion_matrix(y_test, test_predictions)
# Print or log the performance metrics for this model
print(f"Model: {resnet_model.__name__}")
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC AUC: {roc_auc}")
print(f"Confusion Matrix:\n{cm}")
<ipython-input-1-a38ebf77f820>:87: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
history = model.fit_generator(
Epoch 1/100
55/55 [==============================] - ETA: 0s - loss: 0.3239 - accuracy: 0.8589
Epoch 1: val_loss improved from inf to 0.76100, saving model to best_model_ResNet101.h5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Kera
saving_api.save_model(

Untitled.ipynb

File Edit View Insert Runtime Tools Help All changes saved

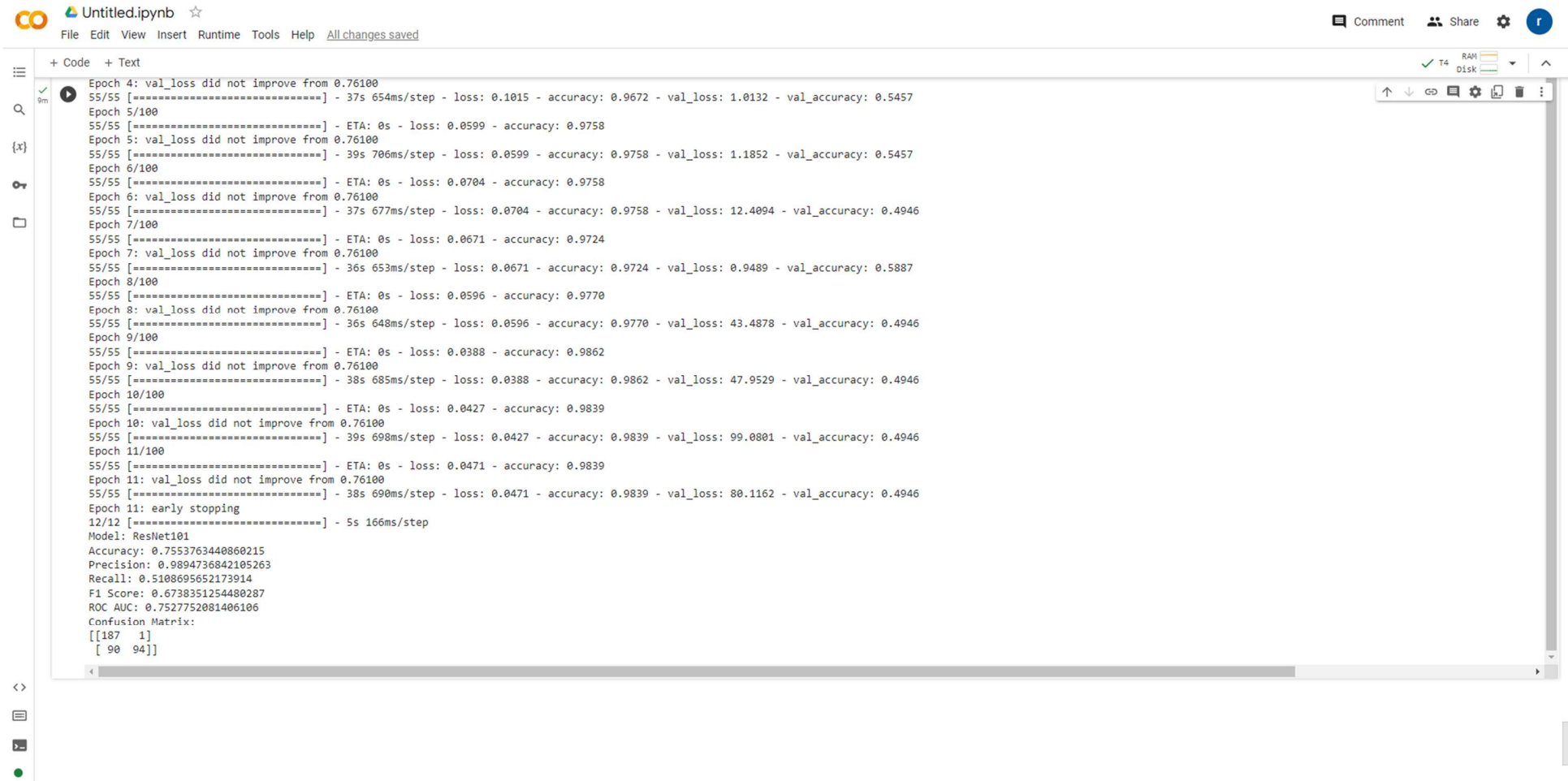
Comment Share

+ Code + Text

T4 RAM Disk

```
    print(f"F1 Score: {f1}")
    print(f"ROC AUC: {roc_auc}")
    print(f"Confusion Matrix:\n{cm}")
```

```
<ipython-input-1-a38ebf77f820>:87: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(
Epoch 1/100
55/55 [==============================] - ETA: 0s - loss: 0.3239 - accuracy: 0.8589
Epoch 1: val_loss improved from inf to 0.76100, saving model to best_model_ResNet101.h5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Kera
  saving_api.save_model(
55/55 [==============================] - 128s 880ms/step - loss: 0.3239 - accuracy: 0.8589 - val_loss: 0.7610 - val_accuracy: 0.7151
Epoch 2/100
55/55 [==============================] - ETA: 0s - loss: 0.1605 - accuracy: 0.9389
Epoch 2: val_loss did not improve from 0.76100
55/55 [==============================] - 37s 667ms/step - loss: 0.1605 - accuracy: 0.9389 - val_loss: 0.7797 - val_accuracy: 0.6828
Epoch 3/100
55/55 [==============================] - ETA: 0s - loss: 0.1202 - accuracy: 0.9505
Epoch 3: val_loss did not improve from 0.76100
55/55 [==============================] - 37s 669ms/step - loss: 0.1202 - accuracy: 0.9505 - val_loss: 4.9981 - val_accuracy: 0.4946
Epoch 4/100
55/55 [==============================] - ETA: 0s - loss: 0.1015 - accuracy: 0.9672
Epoch 4: val_loss did not improve from 0.76100
55/55 [==============================] - 37s 654ms/step - loss: 0.1015 - accuracy: 0.9672 - val_loss: 1.0132 - val_accuracy: 0.5457
Epoch 5/100
55/55 [==============================] - ETA: 0s - loss: 0.0599 - accuracy: 0.9758
Epoch 5: val_loss did not improve from 0.76100
55/55 [==============================] - 39s 706ms/step - loss: 0.0599 - accuracy: 0.9758 - val_loss: 1.1852 - val_accuracy: 0.5457
Epoch 6/100
55/55 [==============================] - ETA: 0s - loss: 0.0704 - accuracy: 0.9758
Epoch 6: val_loss did not improve from 0.76100
55/55 [==============================] - 37s 677ms/step - loss: 0.0704 - accuracy: 0.9758 - val_loss: 12.4094 - val_accuracy: 0.4946
Epoch 7/100
55/55 [==============================] - ETA: 0s - loss: 0.0671 - accuracy: 0.9724
Epoch 7: val_loss did not improve from 0.76100
55/55 [==============================] - 36s 653ms/step - loss: 0.0671 - accuracy: 0.9724 - val_loss: 0.9489 - val_accuracy: 0.5887
Epoch 8/100
55/55 [==============================] - ETA: 0s - loss: 0.0596 - accuracy: 0.9770
Epoch 8: val_loss did not improve from 0.76100
55/55 [==============================] - 36s 648ms/step - loss: 0.0596 - accuracy: 0.9770 - val_loss: 43.4878 - val_accuracy: 0.4946
Epoch 9/100
55/55 [==============================] - ETA: 0s - loss: 0.0388 - accuracy: 0.9862
Epoch 9: val_loss did not improve from 0.76100
55/55 [==============================] - 38s 685ms/step - loss: 0.0388 - accuracy: 0.9862 - val_loss: 47.9529 - val_accuracy: 0.4946
Epoch 10/100
55/55 [==============================] - ETA: 0s - loss: 0.0427 - accuracy: 0.9839
Epoch 10: val_loss did not improve from 0.76100
55/55 [==============================] - 39s 698ms/step - loss: 0.0427 - accuracy: 0.9839 - val_loss: 99.0801 - val_accuracy: 0.4946
Epoch 11/100
```



EXTENDED DESCRIPTION :

1. import :

- imported necessary metrics for Accuracy, Precision, Confusion matrix and so on.

2. Performance Metrics :

- Calculated various performance metrics to assess the model's performance from scikit-learn library , such as:

- Accuracy.

- Precision.

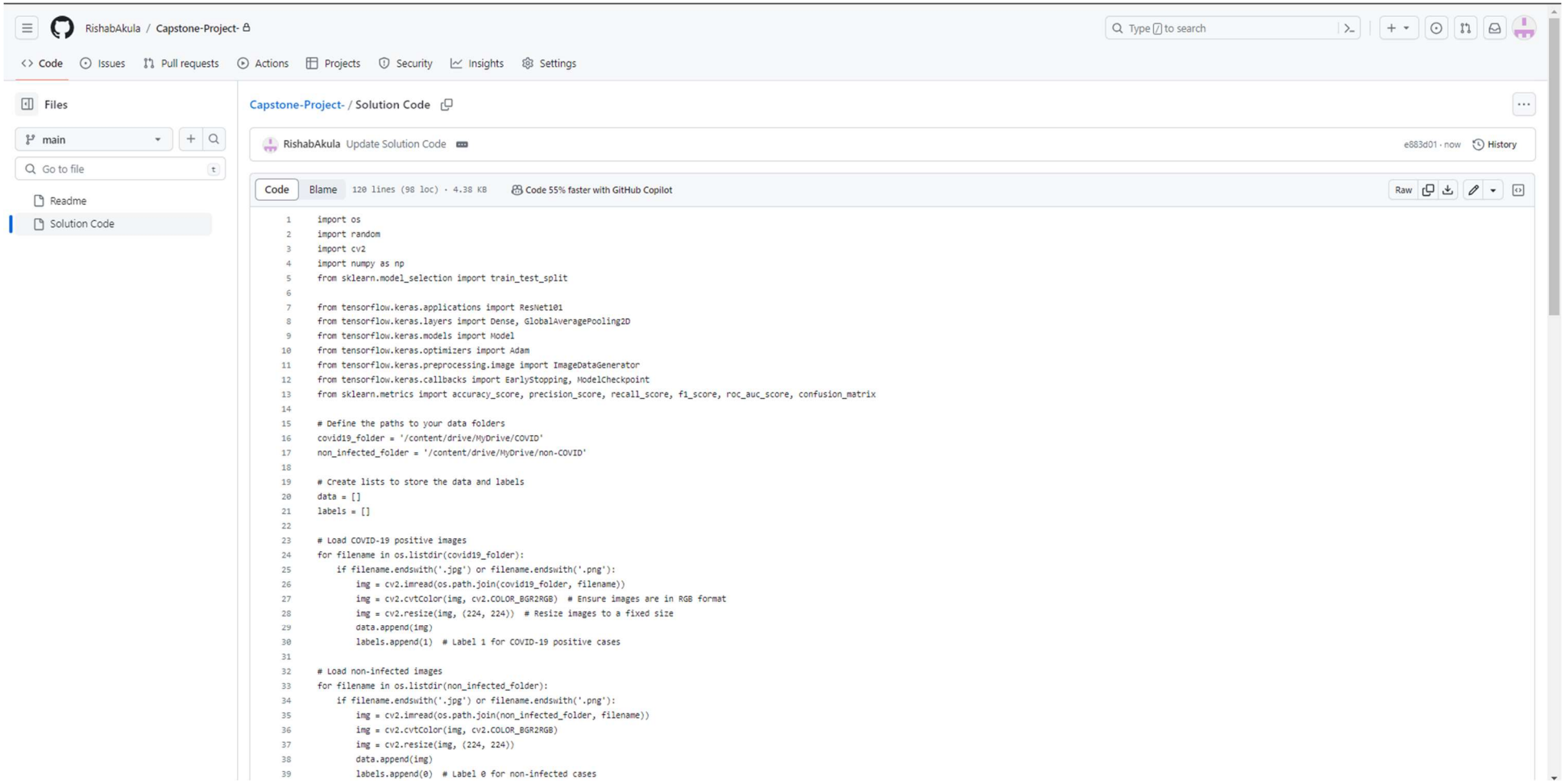
- Recall.

- F1-score.

- ROC curve and AUC (Area Under the Curve).

- Confusion matrix.

GITHUB UPLOADS :



Capstone-Project- / Solution Code

↑ Top

Code | Blame | 120 lines (98 loc) · 4.38 KB | Code 55% faster with GitHub Copilot | Raw

```
            data.append(img)
            labels.append(0)  # Label 0 for non-infected cases

# Convert data and labels to NumPy arrays
data = np.array(data)
labels = np.array(labels)

#size of the data array and labels array
#print(data.size)
#print(labels.size)

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(data, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Define a list of ResNet architectures
#resnet_architectures = [ResNet18, ResNet34, ResNet50, ResNet101, ResNet152]
resnet_architectures = [ResNet101]

for resnet_model in resnet_architectures:
    # Load pre-trained ResNet model
    base_model = resnet_model(weights='imagenet', include_top=False)

    # Add custom top layers for binary classification
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    # Create data generators with data augmentation
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        horizontal_flip=True
    )

    # Set up Early Stopping and Model Checkpoint callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
    model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)

```

Capstone-Project- / Solution Code

↑ Top

Code | Blame | 120 lines (98 loc) · 4.38 KB | Code 55% faster with GitHub Copilot | Raw

```
        train_datagen = ImageDataGenerator(
            rescale=1./255,
            rotation_range=20,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.2,
            horizontal_flip=True
        )

        # Set up Early Stopping and Model Checkpoint callbacks
        early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
        model_checkpoint = ModelCheckpoint(f'best_model_{resnet_model.__name__}.h5', save_best_only=True, verbose=1)

        # Train the model
        history = model.fit_generator(
            train_datagen.flow(X_train, y_train, batch_size=32),
            validation_data=(X_val, y_val),
            epochs=100,
            callbacks=[early_stopping, model_checkpoint]
        )

        # Load the best model
        model.load_weights(f'best_model_{resnet_model.__name__}.h5')

        # Evaluate the model on the test set
        test_predictions = model.predict(X_test)
        test_predictions = (test_predictions > 0.5).astype(int)

        # Selected Model Name
        #print(f"Model: {resnet_model.__name__}")

    # Calculate performance metrics
        accuracy = accuracy_score(y_test, test_predictions)
        precision = precision_score(y_test, test_predictions)
        recall = recall_score(y_test, test_predictions)
        f1 = f1_score(y_test, test_predictions)
        roc_auc = roc_auc_score(y_test, test_predictions)
        cm = confusion_matrix(y_test, test_predictions)

        # Print or log the performance metrics for this model
        print(f"Model: {resnet_model.__name__}")
        print(f"Accuracy: {accuracy}")
        print(f"Precision: {precision}")
        print(f"Recall: {recall}")
        print(f"F1 Score: {f1}")
        print(f"ROC AUC: {roc_auc}")
        print(f"Confusion Matrix:\n{cm}")
```