UTM
UNIVERSITI TEKNOLOGI MALAYSIA

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SEMESTER I, SESSION 2023/2024

SECB3203: PROGRAMMINGS FOR BIOINFORMATICS

GROUP PROJECT

PROGRESS 5

SECTION: 01

LECTURER'S NAME: DR NIES HUI WEN

STUDENTS	MATRIC NUMBER
MUHAMMAD HARITH ZAEEF BIN KHAIRUL NIZAM	B22EC0007
MEGAT ZARUL AKMAL BIN MOHD ZAHIR	B22EC0070

Contents

Introduction:

Colorectal cancer, commonly known as colon cancer, remains a pressing global health concern, casting a significant shadow of morbidity and mortality across the world. This malady transcends borders, impacting diverse populations in distinct ways. The detection and diagnosis of colon cancer pose formidable challenges within the realm of healthcare on a global scale.In this dynamic landscape, the critical importance of detecting colon cancer at an early and treatable stage looms large. The timeliness of diagnosis is a matter of paramount concern, as it bears a direct and profound impact on patient outcomes and the allocation of precious healthcare resources.

As degree students deeply immersed in the field of healthcare, we recognize the pivotal role early detection plays in the prognosis of colon cancer patients globally. It can spell the difference between life and death, significantly influencing the quality of life and long-term survival prospects. Acknowledging the urgency of this issue, we embark on a comprehensive case study that delves into the specific nuances of colon cancer detection within the intricate tapestry of the global healthcare landscape.

Our mission is clear: to identify the unique factors that exert their influence on early diagnosis and the subsequent effectiveness of medical interventions. By doing so, we aim not only to gain a deeper understanding of the challenges and opportunities within our healthcare system but also to contribute to the ongoing efforts to improve the response to this critical health concern in our country. Through rigorous research and analysis, we hope to shed light on the intricacies of colon cancer detection, ultimately forging a path toward better outcomes and enhanced patient care. In an era where healthcare is a cornerstone of societal well-being, we recognize the significance of our work in the broader context of global health and wellness.

Problem Statement:

The crux of the matter underpinning this global case study revolves around the uneven success rate of early colon cancer detection worldwide. Early diagnosis is a linchpin in the quest for improved patient outcomes, yet it's a variable that remains frustratingly inconsistent and suboptimal within the global healthcare framework. This inconsistency in timely detection is a concern of profound significance, with consequences that ripple throughout healthcare systems globally.

When colon cancer is diagnosed at a later stage on a global scale, it often necessitates more aggressive and complex treatments, and unfortunately, the odds of survival decrease significantly. These advanced-stage cases also result in a considerable upswing in healthcare costs, imposing additional burdens on both the global healthcare system and the affected individuals. As degree students in the healthcare field, we're acutely aware of the multifaceted challenges and ramifications associated with these delayed diagnoses on a global scale.

Therefore, uncovering the underlying reasons for this variability in early detection success is not merely an academic pursuit but a mission with tangible and far-reaching consequences globally. Our objective is clear: to gain comprehensive insights into the root causes of this inconsistency, thereby providing a foundation for targeted interventions aimed at enhancing patient outcomes and optimizing the allocation of healthcare resources on a global scale. The implications of our research are profound, and we are committed to making a meaningful contribution towards a more effective and equitable global healthcare system. In a healthcare landscape where time is often the difference between life and death, our work takes on heightened importance, underscoring the urgency of this investigation.

Problem Background:

Colorectal cancer on a global scale is a multifaceted health challenge influenced by various factors specific to each country. Several elements contribute to the difficulties in early detection of colon cancer worldwide:

1. Limited Awareness and Education: There is a lack of public awareness and understanding of colon cancer in many regions globally, leading to lower participation in screening programs and reduced knowledge about the disease.

2. Inadequate Screening Programs:Colon cancer screening, including colonoscopy and fecal occult blood tests, is not universally implemented or accessible to the extent necessary to ensure early detection in many countries.

3. Cultural and Language Diversity: Many countries have diverse populations with various ethnicities and languages, posing communication challenges that can affect healthcare access and awareness of colon cancer risks on a global scale.

4. Healthcare Disparities: Socioeconomic factors and regional disparities impact access to healthcare and resources globally, further complicating early detection efforts in various regions.

5. Diet and Lifestyle Choices:Dietary habits vary globally, and certain culinary traditions may contribute to the risk of colon cancer in different populations. Additionally, lifestyle factors such as physical inactivity can influence disease development on a global scale.

6. Resource Constraints:Globally, many healthcare systems face resource constraints, including shortages of medical personnel and equipment, which can hinder early detection and timely interventions for colon cancer. The allocation of resources for preventive measures, screening programs, and adequate medical infrastructure becomes a critical factor in determining the success of early detection efforts on a global scale.

In summary, the variable success rate in detecting colon cancer at an early stage is a pressing global issue, and understanding the underlying factors specific to each country is essential. This case study endeavours to explore these factors comprehensively, with the ultimate aim of proposing strategies and recommendations to enhance early detection rates and improve the chances of successful treatment for colon cancer worldwide. Addressing challenges such as limited awareness, inadequate screening programs, cultural diversity, healthcare disparities, diet and lifestyle choices, and resource constraints on a global scale is crucial for developing effective, equitable, and sustainable approaches to mitigate the impact of colorectal cancer globally.

Aim & Objectives

The main objective of this study is to design and implement classification models that demonstrate high levels of accuracy in colon cancer diagnosis using advanced machine learning techniques. This goal is driven by the urgent need for more effective and efficient methods to identify colon cancer at an early stage, which is critical for improving patient outcomes and reducing the burden. common to this disease.

Early detection of colon cancer using machine learning:

- The first goal of this study is to focus on early detection. This involves developing machine learning models capable of identifying the presence of colon cancer at an early stage. By analysing diverse clinical and genetic data, these models aim to identify subtle patterns and markers associated with colon cancer before symptoms start to develop
- This aims to contribute to early intervention and rapid treatment of colon cancer, which can significantly increase a patient's chances of successful recovery.

Evaluate the efficiency of colon cancer classification based on machine learning architecture:

- This evaluation includes rigorous testing, cross-validation, and performance metrics to evaluate the effectiveness of the developed models.
- This goal is necessary to understand how the developed model are applied to the real-world and determine their potential to be use as clinical tools.

Research Scope

This study will be focused on clinical and genetic data related to colon cancer colon cancer. The machine learning method is applied for colon cancer classification. Primary data sources will include histopathological images.

The research will use advanced data analysis techniques including deep learning algorithms that are EfficientNet. Method that will be used include data preprocessing, model development, cross-validation, and performance evaluation. It will follow a systematic process to create, test and validate machine learning models for colon cancer classification. The approach of this research is to apply deep learning algorithms, especially EfficientNet, to analyse and classify the collected data. The goal is to create models that can accurately identify colon cancer cases.

Hardware and software requirements

The hardware that will be used in this research is matching the requirements needed and is important in order to ensure the process of designing and developing the EfficientNet for colon cancer classification. Hardware requirements for this research is a laptop with Intel Core i5 9th gen processor. The operating systems are Windows 11 with
8GB RAM memory

The software required is Microsoft Visual Studio as IDE which is easy to use. Python language were implement for the coding with multiple useful packages also installed. Besides that, Microsoft Word 2019 are used for documentation and Microsoft Excel 2019 used for data management

Flowchart


Start
Literature Review From Thesis
Software Installation and Hardware Setup
Collect Patient Data
Process Data using Convolutional Neural Network CNN(EfficientNet)
Analysis Data With Python
Create Model Architecture
Data Splitting and Model Training
Data Testing
Model Training
CNN Learning Method Convolutional Neural Network CNN(EfficientNet)
Prediction Result
Evaluate Prediction Model
Output Result
End

Importing Dataset

With all dataset that have been collected, it is important for us to understand on how the data worked and how to use it. Below are the lines of codes that were applied for importing our dataset. Since images are our data type, OpenCV are used to import the image into Python in Visual Studio Code.

```
import cv2
img = cv2.imread('D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Project/Coding/colon_image_sets')
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
```

After importing the data, we show the images. Below are the codes that read the data and store it inn the data frame

```
data_dir = 'D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Project/Coding/colon_image_sets'
filepaths = []
labels = []

folds = os.listdir(data_dir)
for fold in folds:
    foldpath = os.path.join(data_dir, fold)
    filelist = os.listdir(foldpath)
    for file in filelist:
        fpath = os.path.join(foldpath, file)

        filepaths.append(fpath)
        labels.append(fold)

# Concatenate data paths with labels into one dataframe
Fseries = pd.Series(filepaths, name= 'filepaths')
Lseries = pd.Series(labels, name='labels')
```

```
modules loaded
                                              filepaths     labels
0     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
1     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
2     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
3     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
4     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
...                                                 ...        ...
9994  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9995  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9996  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9997  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9998  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
```

In the terminal, it shows the directory of the file path where the images is located. Labels colon_aca referring to colon adenocarcinoma, while colon_n referring to colon benign tissue.

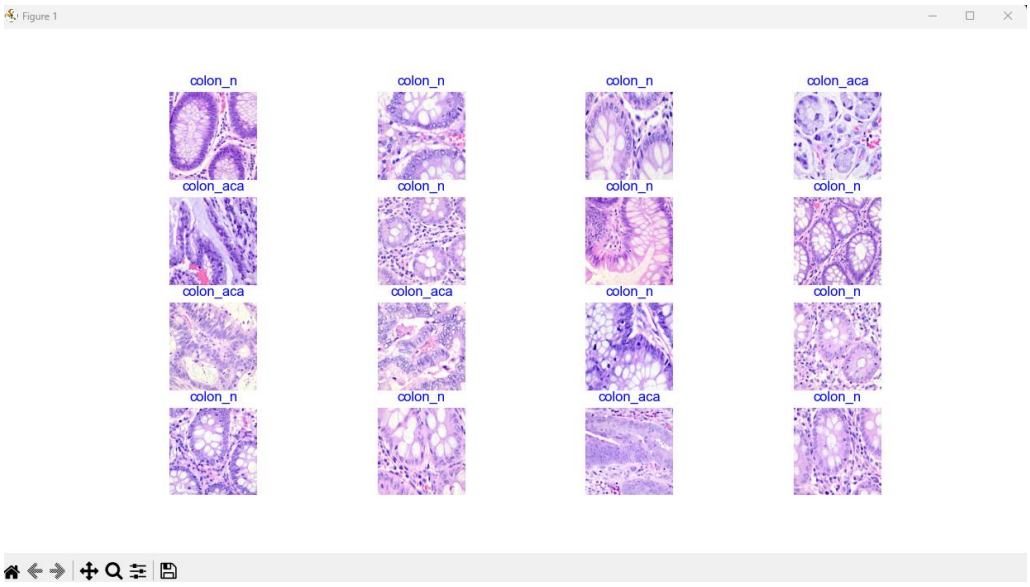
Data Preprocessing

This code uses TensorFlow's ImageDataGenerator to generate image data generators for the training, validation,and test sets. The generator is configured to apply various data augmentation techniques, such as rotation, zoom, and flipping.

```
#crop image
#creat image generator
batch_size = 16
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()

train_gen = tr_gen.flow_from_dataframe( train_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                        color_mode= 'rgb', shuffle= True, batch_size= batch_size)

valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                        color_mode= 'rgb', shuffle= True, batch_size= batch_size)

test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                       color_mode= 'rgb', shuffle= False, batch_size= batch_size)

g_dict = train_gen.class_indices      # defines dictionary {'class': index}
classes = list(g_dict.keys())       # defines list of dictionary's kays (classes), classes names : string
images, labels = next(train_gen)      # get a batch size samples from the generator

plt.figure(figsize= (20, 20))
```

```
Found 8006 validated image filenames belonging to 2 classes.
Found 1201 validated image filenames belonging to 2 classes.
Found 801 validated image filenames belonging to 2 classes.
```

Display Some of the Data



The main packages that were used are Keras and Tensorflow which are essential for image processing.

```
# import Deep learning Libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential    Import "tensorflow.keras.models" could not be resolved
from tensorflow.keras.optimizers import Adam, Adamax    Import "tensorflow.keras.optimizers" could not be resolved
from tensorflow.keras.preprocessing.image import ImageDataGenerator    Import "tensorflow.keras.preprocessing.image"
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras import regularizers    Import "tensorflow.keras" could not be resolved
```

Data Wrangling (Pandas / Numpy)

We identify any missing values or problematic files from the imported image dataset

```
# Create a list to store paths of missing or problematic files
missing_files = []
# Check each image file
for image_file in image_files:
    image_path = os.path.join(data_dir, image_file)
    # Try to read the image using OpenCV
    try:
        img = cv2.imread('D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Project/Coding/colon_image_sets')
        if img is None:
            missing_files.append(image_path)
    except Exception as e:
        print(f"Error reading {image_path}: {str(e)}")
        missing_files.append(image_path)
# Print or handle missing files
if len(missing_files) > 0:
    print("Missing or problematic files:")
    for missing_file in missing_files:
        print(missing_file)
else:
    print("No missing or problematic files found.")
```

```
                                              filepaths     labels
0     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
1     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
2     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
3     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
4     D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...  colon_aca
...                                                 ...        ...
9994  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9995  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9996  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9997  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n
9998  D:/DEGREE/YEAR3SEM1/PROG BIOINFORMATICS/Projec...    colon_n

[9999 rows x 2 columns]
No missing or problematic files found.
```

Data Normalization

Data normalization is important for training deep learning models because it stabilizes the training process and improves convergence. In the code, normalization is applied as follows.

```
#Data Noramalization (Scaling)

for i in range(16):
    plt.subplot(4, 4, i + 1)
    image = images[i] / 255       # scales data to range (0 - 255)
    plt.imshow(image)
    index = np.argmax(labels[i])  # get image index
    class_name = classes[index]   # get class of image
    plt.title(class_name, color= 'blue', fontsize= 12)
    plt.axis('off')
plt.show()
```

Here, each image in the batch (image) is normalized by dividing by 255.This scales the pixel values from the original range of [0, 255] to the normalized range of [0,1]. Pixel value normalization is a common method in image processing and deep learning to ensure the model is insensitive to the absolute scale of pixel values.

Model Development

This code is responsible for creating a convolutional neural network (CNN) model using the EfficientNetB3 architecture as a base model for image classification.

```
# Create Model Structure
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys())) # to define number of classes in dense layer

# we will use efficientnetb3 from EfficientNet family.
base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top= False, weights= "imagenet", input_shape= img_shape, pooling= 'max')
# base_model.trainable = False

model = Sequential([
    base_model,
    BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001),
    Dense(256, kernel_regularizer= regularizers.l2(l= 0.016), activity_regularizer= regularizers.l1(0.006),
    |   |   |   bias_regularizer= regularizers.l1(0.006), activation= 'relu'),
    Dropout(rate= 0.45, seed= 123),
    Dense(class_count, activation= 'softmax')
])

model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy', metrics= ['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb3 (Functional  (None, 1536)              10783535
 )

 batch_normalization (Batch  (None, 1536)              6144
 Normalization)

 dense (Dense)               (None, 256)               393472

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 11183665 (42.66 MB)
Trainable params: 11093290 (42.32 MB)
Non-trainable params: 90375 (353.03 KB)
_________________________________________________________________
```

Model Created

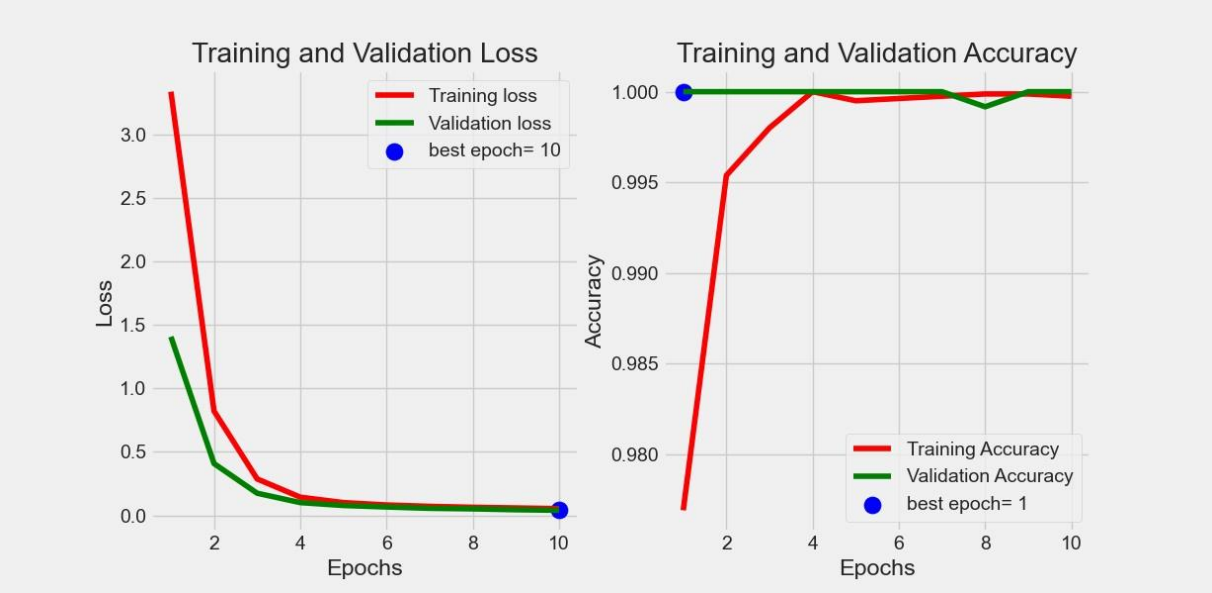
Model Training

The below code is responsible for training the neural network model using the training data generated by the train_gen. This process took a very long time

```
#Train the model

epochs = 10   # number of all epochs in training

history = model.fit(x= train_gen, epochs= epochs, verbose= 1, validation_data= valid_gen,
                    validation_steps= None, shuffle= False)

# Save model weights after training
model.save_weights('model_weights.h5')

# Load model weights before using the model
model.load_weights('model_weights.h5')
```

Display Model Performance

```
# Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

# Plot training history
plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout
plt.show()
```



The provided code is used to analyze and visualize the training history of the neural network model.It extracts relevant information from the history object returned by the model.fit method, then creates a plot to visualize the loss and accuracy during training and validation over epochs .

Model Evaluation

```
#Model Evaluation
ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

train_score = model.evaluate(train_gen, steps= test_steps, verbose= 1)
valid_score = model.evaluate(valid_gen, steps= test_steps, verbose= 1)
test_score = model.evaluate(test_gen, steps= test_steps, verbose= 1)

print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Validation Loss: ", valid_score[0])
print("Validation Accuracy: ", valid_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])

preds = model.predict_generator(test_gen)
y_pred = np.argmax(preds, axis=1)

g_dict = test_gen.class_indices
classes = list(g_dict.keys())
```

Get Predictions

```
Train Loss:  0.04023737460374832
Train Accuracy:  1.0
--------------------
Validation Loss:  0.04016776382923126
Validation Accuracy:  1.0
--------------------
Test Loss:  0.04018598794937134
Test Accuracy:  1.0
```

The evaluate method is used to obtain loss and precision scores for training, validation, and testing datasets. The steps parameter is set to test_steps to determine the number of batches to process. The output provides information about the performance of the trained model on the training, validation, and test datasets. Here's a breakdown of the output:

1. Training Results:
    - Train Loss: The loss (error) on the training set.
    - Train Accuracy: The accuracy achieved on the training set.
2. Validation Results:
    - Validation Loss: The loss on the validation set.
    - Validation Accuracy: The accuracy achieved on the validation set.
3. Test Results:
    - Test Loss: The loss on the test set.
    - Test Accuracy: The accuracy achieved on the test set.

In this case, the accuracy values from training, testing, and validation were all 1.0 (or 100%), meaning the model achieved perfect accuracy on all three datasets.

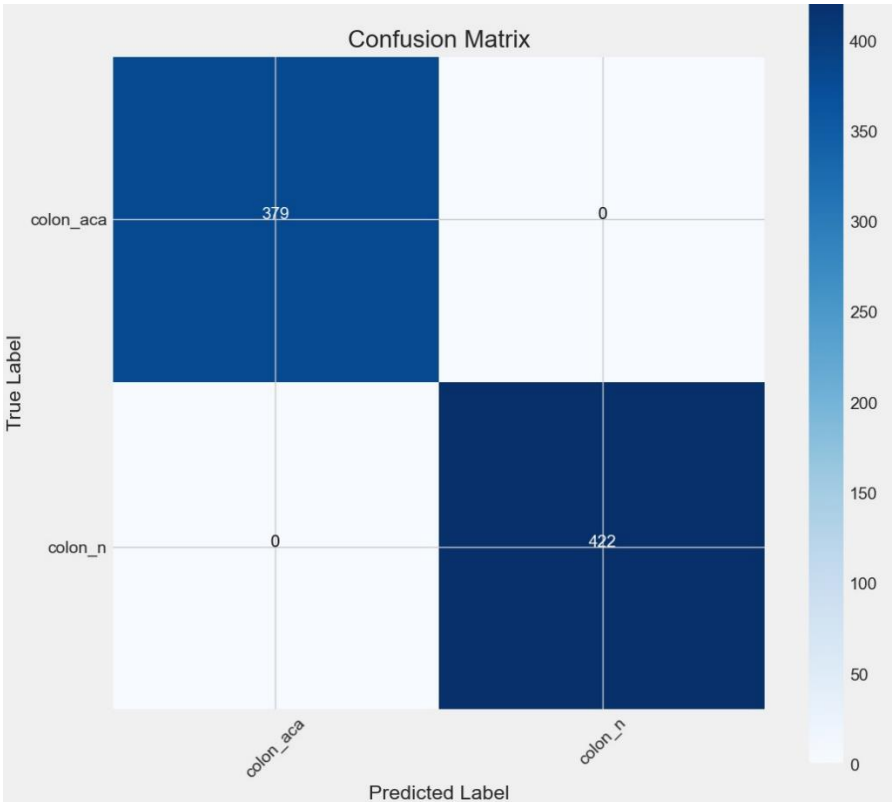
Confusion Matrix

```
# Confusion matrix
cm = confusion_matrix(test_gen.classes, y_pred)

plt.figure(figsize= (10, 10))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation= 45)
plt.yticks(tick_marks, classes)


thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'white' if cm[i, j] > thresh else 'black')

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.show()
```

The confusion_matrix function from the sklearn.metrics module is used to compute the confusion matrix. It takes two arguments:

test_gen.classes: The true labels of the test set obtained from the generator.

y_pred: The predicted labels generated by the model. **plt.ylabel** and **plt.xlabel** set the labels for the y-axis (true labels) and x-axis (predicted labels), respectively.



Confusion Matrix Figure

In this case:

- **True Positives (TP):** 379
- **True Negatives (TN):** 422
- **False Positives (FP):** 0
- **False Negatives (FN):** 0

This shows that your model has completed accurate predictions for all cases in the test set. More precisely: There were 379 cases that were actually positive (colon_aca) and were correctly predicted to be positive. There were 422 cases that were actually negative (colon_n) and were correctly predicted to be negative. No cases were incorrectly predicted as positive (false positive). No cases were incorrectly predicted to be negative (false negative).

Classfication Report

	precision	recall	f1-score	support
colon_aca	1.00	1.00	1.00	379
colon_n	1.00	1.00	1.00	422
accuracy			1.00	801
macro avg	1.00	1.00	1.00	801
weighted avg	1.00	1.00	1.00	801

The **classification_report** function computes and prints several metrics for each class and an overall summary. The metrics include:

Precision: The ratio of true positive predictions to the total number of positive predictions (true positives + false positives). It measures the accuracy of positive predictions.

Recall (Sensitivity or True Positive Rate): The ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). It measures the ability of the model to capture all positive instances.

F1-Score: The harmonic mean of precision and recall. It provides a balance between precision and recall. To calculate, use formula = 2 * (Precision * Recall) / (Precision + Recall)

Support: The number of actual instances of each class in the test set

In this case, we achieved the accuracy with value of 1.0 for all the metrics for each class. This indicates that our model is achieving perfect performance on the metrics for each class

Summary

Colorectal cancer, commonly known as colon cancer, poses a significant global health challenge, marked by uneven success rates in early detection. The primary issue lies in the challenges associated with detecting colon cancer at an early stage, resulting in the need for aggressive treatments, diminished survival rates, and escalated healthcare costs worldwide. Contributing factors encompass limited awareness, inadequate screening programs, cultural diversity, healthcare disparities, dietary choices, and constrained resources on a global scale.

The primary objective of this research is to develop accurate machine learning models dedicated to the early diagnosis of colon cancer, aiming to enhance early intervention and treatment outcomes globally. By concentrating on data from the past five years and utilising advanced data analysis techniques, this study seeks to elevate early detection rates and subsequently improve treatment outcomes for patients worldwide. The integration of machine learning into the global healthcare landscape holds the promise of transforming the diagnostic paradigm, offering more precise and timely assessments that can significantly impact patient well-being and optimize healthcare resources on a global scale.