

```
from IPython.display import Image, display

# Display the image
display(Image(filename='Picture1.png'))
```



How Download data from Kaggle

- Create a Kaggle Account (if you don't have one): Go to <https://www.kaggle.com> and sign up/log in.
- Go this link: <https://www.kaggle.com/datasets/adityamahimkar/iqothnccd-lung-cancer-dataset/versions/1> and download the dataset
- This dataset is about lung cancer and consists of three classes: Benign, Malignant, and Normal.

Splitting dataset

```
import os, shutil
from sklearn.model_selection import train_test_split

original_dataset_dir = 'The IQ-OTHNCCD lung cancer dataset'
base_dir = 'dataset'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

for dir in [train_dir, validation_dir, test_dir]:
    os.makedirs(dir, exist_ok=True)

train_ratio, validation_ratio, test_ratio = 0.7, 0.1, 0.2
```

```
def split_and_copy_files(class_dir, dest_base_dir):
    all_images = [os.path.join(class_dir, f) for f in
os.listdir(class_dir) if os.path.isfile(os.path.join(class_dir, f))]
    train_images, temp_images = train_test_split(all_images,
test_size=(1 - train_ratio), random_state=42)
    validation_images, test_images = train_test_split(temp_images,
test_size=test_ratio/(validation_ratio + test_ratio), random_state=42)

    for images, folder_name in zip([train_images, validation_images,
test_images], ['train', 'validation', 'test']):
        dest_dir = os.path.join(dest_base_dir, folder_name,
os.path.basename(class_dir))
        os.makedirs(dest_dir, exist_ok=True)
        for file in images:
            shutil.copy(file, dest_dir)

# Apply to all class subfolders
for class_dir in os.listdir(original_dataset_dir):
    full_class_dir = os.path.join(original_dataset_dir, class_dir)
    if os.path.isdir(full_class_dir):
        split_and_copy_files(full_class_dir, base_dir)
```

Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing unstructured grid-like data, such as images. CNNs are widely used in computer vision tasks, including image classification, object detection, and segmentation.

How CNN Works

CNNs process images by learning hierarchical patterns using different layers:

- **Input Layer**

The input is an image represented as a matrix of pixel values. Example: A grayscale image is represented as a 2D matrix, while a color image (RGB) is represented as a 3D matrix.

- **Convolutional Layer**

- Applies filters (kernels) to detect features like edges, textures, and shapes.
- Each filter extracts a different feature.
- The result is called a feature map.
- Example: A 3x3 filter moves across the image and performs a dot product between pixel values and filter values.

- Pooling Layer (Downsampling)
 - Reduces the dimensionality while keeping important features.
 - Common pooling methods:
 - Max Pooling: Takes the maximum value in a region.
 - Average Pooling: Takes the average value in a region.

□ Example: A 2x2 max pooling reduces the size of the feature map while keeping key features.

- Fully Connected Layer
 - Flattens the feature maps into a 1D vector.
- Softmax / Sigmoid (Output Layer)
 - Softmax: Used for multi-class classification (assigns probabilities to classes).
 - Sigmoid: Used for binary classification (outputs 0 or 1).

Different types of CNN models

- AlexNet
- VGGNet (VGG-16, VGG-19)
- Inception
- ResNet
 - ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152
- DenseNet (Densely Connected CNNs)
- EfficientNet

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.layers import Dense, Flatten, Dropout,
GlobalAveragePooling2D
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score, classification_report
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

def create_data_generators(train_dir, val_dir, test_dir,
img_size=(240, 240), batch_size=32):
    # Data augmentation for training dataset
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=360,
        horizontal_flip=True,
        vertical_flip=True,
        width_shift_range=0.1,
        height_shift_range=0.1,
```

```

)

# Only rescale for validation and testing
val_test_datagen = ImageDataGenerator(rescale=1./255)

# Generate augmented training images
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

# Generate validation images
val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

# Generate test images
test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

return train_generator, val_generator, test_generator

```

CNN model

```

def build_cnn_model(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape),
        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'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

# Directories for datasets
train_dir = 'dataset/train'
val_dir = 'dataset/validation'
test_dir = 'dataset/test'

# Create data generators
train_gen, val_gen, test_gen = create_data_generators(train_dir,
val_dir, test_dir)

# Get input shape and number of classes
input_shape = (240, 240, 3)
num_classes = len(train_gen.class_indices)

# Build CNN model
model = build_cnn_model(input_shape, num_classes)

# Train the model
history = model.fit(train_gen, epochs=10, validation_data=val_gen)

```

```

Found 766 images belonging to 3 classes.
Found 109 images belonging to 3 classes.
Found 222 images belonging to 3 classes.

```

```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\keras\
src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.

```

```

    super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

```

```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,

```

```
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
```

```
self._warn_if_super_not_called()
```

```
Epoch 1/10
```

```
24/24 _____ 70s 3s/step - accuracy: 0.4312 - loss: 2.1222 - val_accuracy: 0.5138 - val_loss: 0.9573
```

```
Epoch 2/10
```

```
24/24 _____ 59s 2s/step - accuracy: 0.5031 - loss: 0.9496 - val_accuracy: 0.5138 - val_loss: 0.9742
```

```
Epoch 3/10
```

```
24/24 _____ 59s 2s/step - accuracy: 0.5166 - loss: 0.9311 - val_accuracy: 0.6055 - val_loss: 0.9304
```

```
Epoch 4/10
```

```
24/24 _____ 82s 2s/step - accuracy: 0.5449 - loss: 0.9267 - val_accuracy: 0.5872 - val_loss: 0.9534
```

```
Epoch 5/10
```

```
24/24 _____ 58s 2s/step - accuracy: 0.5582 - loss: 0.9305 - val_accuracy: 0.4220 - val_loss: 1.2644
```

```
Epoch 6/10
```

```
24/24 _____ 59s 2s/step - accuracy: 0.4772 - loss: 0.9676 - val_accuracy: 0.5046 - val_loss: 0.9266
```

```
Epoch 7/10
```

```
24/24 _____ 60s 2s/step - accuracy: 0.5310 - loss: 0.9354 - val_accuracy: 0.5321 - val_loss: 0.8965
```

```
Epoch 8/10
```

```
24/24 _____ 58s 2s/step - accuracy: 0.5125 - loss: 0.9284 - val_accuracy: 0.5138 - val_loss: 0.8912
```

```
Epoch 9/10
```

```
24/24 _____ 59s 2s/step - accuracy: 0.5128 - loss: 0.9683 - val_accuracy: 0.5596 - val_loss: 0.9067
```

```
Epoch 10/10
```

```
24/24 _____ 57s 2s/step - accuracy: 0.5258 - loss: 0.9430 - val_accuracy: 0.6972 - val_loss: 0.8837
```

```
def evaluate_model(model_name, y_true, y_pred):
```

```
    labels = list(test_gen.class_indices.keys())
```

```
    accuracy = accuracy_score(y_true, y_pred)
```

```
    precision = precision_score(y_true, y_pred, average='weighted')
```

```
    recall = recall_score(y_true, y_pred, average='weighted')
```

```
    f1 = f1_score(y_true, y_pred, average='weighted')
```

```
    cm = confusion_matrix(y_true, y_pred)
```

```
    print("Accuracy:", accuracy)
```

```
    print("Precision :", precision)
```

```
    print("Recall:", recall)
```

```
    print("F1-score:" , f1)
```

```

print("Classification Report:")
print(classification_report(y_true, y_pred_classes,
target_names=labels))

```

```

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

```

```

y_pred=model.predict(test_gen)
y_true = test_gen.classes
y_pred_classes = np.argmax(y_pred, axis=1)
evaluate_model('CNN', y_true, y_pred_classes)

```

7/7 ————— 6s 836ms/step

Accuracy: 0.6981981981981982

Precision : 0.6169145376566717

Recall: 0.6981981981981982

F1-score: 0.6540387304384471

Classification Report:

	precision	recall	f1-score	support
Bengin cases	0.00	0.00	0.00	25
Malignant cases	0.73	0.88	0.80	113
Normal cases	0.65	0.67	0.66	84
accuracy			0.70	222
macro avg	0.46	0.51	0.48	222
weighted avg	0.62	0.70	0.65	222

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

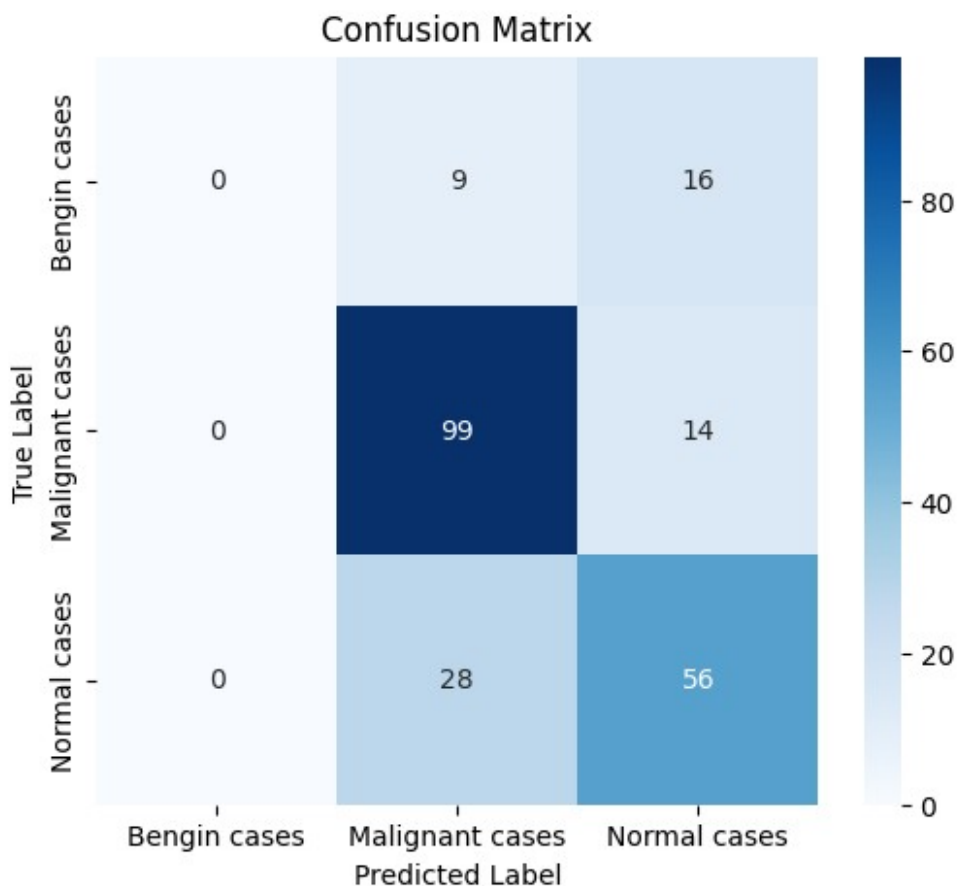
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this

```

behavior.
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\
sklearn\metrics\_classification.py:1531: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this
behavior.
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\
sklearn\metrics\_classification.py:1531: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this
behavior.
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

```



ResNet50

```

def build_resnet8_model(input_shape, num_classes):
    base_model = ResNet50(weights='imagenet', include_top=False,

```

```

input_shape=input_shape)
    base_model.trainable = False # Freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

model = build_resnet8_model(input_shape, num_classes)

history = model.fit(train_gen, epochs=5, validation_data=val_gen)

Epoch 1/5
24/24 ————— 131s 5s/step - accuracy: 0.4710 - loss:
1.2582 - val_accuracy: 0.5321 - val_loss: 0.9655
Epoch 2/5
24/24 ————— 946s 4s/step - accuracy: 0.4549 - loss:
0.9996 - val_accuracy: 0.5138 - val_loss: 0.9520
Epoch 3/5
24/24 ————— 102s 4s/step - accuracy: 0.4515 - loss:
0.9810 - val_accuracy: 0.5138 - val_loss: 0.9463
Epoch 4/5
24/24 ————— 105s 4s/step - accuracy: 0.5025 - loss:
0.9555 - val_accuracy: 0.5138 - val_loss: 0.9474
Epoch 5/5
24/24 ————— 104s 4s/step - accuracy: 0.4970 - loss:
0.9497 - val_accuracy: 0.5138 - val_loss: 0.9400

y_pred=model.predict(test_gen)
y_true = test_gen.classes
y_pred_classes = np.argmax(y_pred, axis=1)
evaluate_model('ResNet50', y_true, y_pred_classes)

7/7 ————— 36s 4s/step
Accuracy: 0.509009009009009
Precision : 0.2590901712523334
Recall: 0.509009009009009
F1-score: 0.3433911523463762
Classification Report:

```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

Bengin cases	0.00	0.00	0.00	25
Malignant cases	0.51	1.00	0.67	113
Normal cases	0.00	0.00	0.00	84
accuracy			0.51	222
macro avg	0.17	0.33	0.22	222
weighted avg	0.26	0.51	0.34	222

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

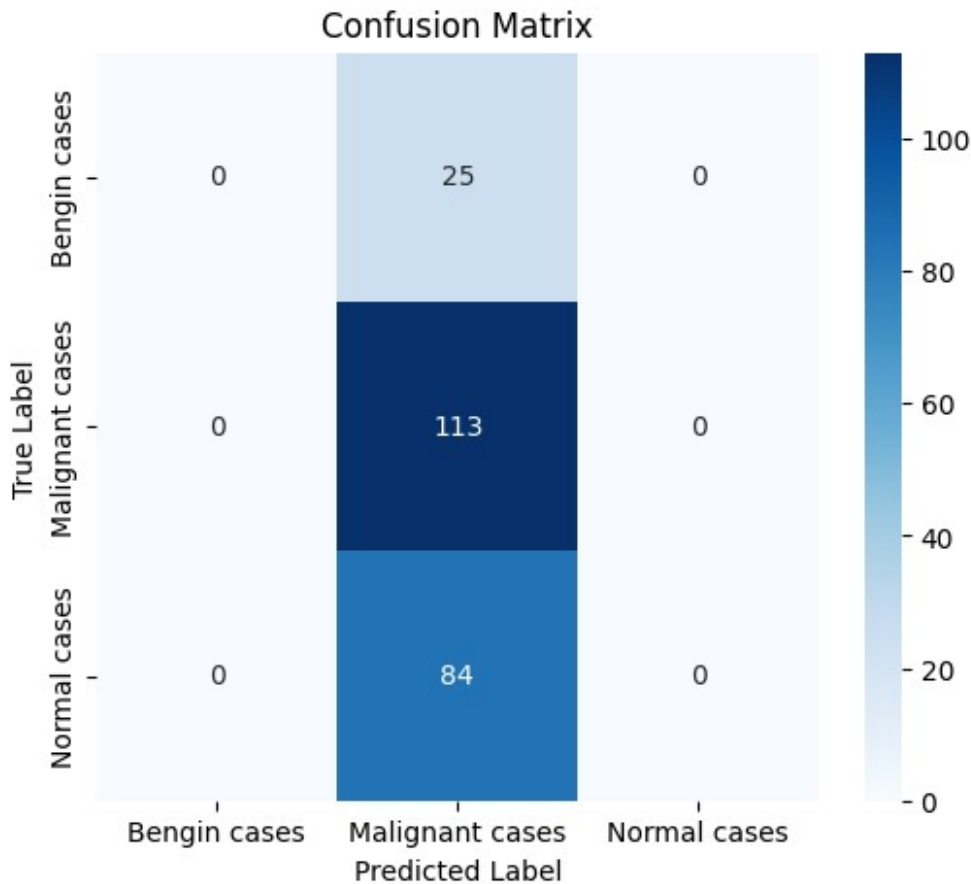
```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```



EfficientNetB0

```
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
Dropout

def build_efficientnetb0_model(input_shape, num_classes):
    base_model = EfficientNetB0(weights='imagenet', include_top=False,
input_shape=input_shape)
    base_model.trainable = False # Freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

```

return model

model = build_efficientnetb0_model(input_shape, num_classes)

history = model.fit(train_gen, epochs=5, validation_data=val_gen)

Epoch 1/5
24/24 _____ 99s 3s/step - accuracy: 0.4299 - loss:
1.0830 - val_accuracy: 0.5138 - val_loss: 0.9563
Epoch 2/5
24/24 _____ 72s 2s/step - accuracy: 0.5051 - loss:
1.0208 - val_accuracy: 0.5138 - val_loss: 0.9702
Epoch 3/5
24/24 _____ 56s 2s/step - accuracy: 0.4599 - loss:
1.0185 - val_accuracy: 0.5138 - val_loss: 0.9556
Epoch 4/5
24/24 _____ 59s 2s/step - accuracy: 0.4997 - loss:
0.9807 - val_accuracy: 0.5138 - val_loss: 0.9531
Epoch 5/5
24/24 _____ 77s 2s/step - accuracy: 0.4936 - loss:
0.9779 - val_accuracy: 0.5138 - val_loss: 0.9546

```

```

y_pred=model.predict(test_gen)
y_true = test_gen.classes
y_pred_classes = np.argmax(y_pred, axis=1)
evaluate_model('EfficientNetB0', y_true, y_pred_classes)

```

WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x00000142830CEDE0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

```
7/7 _____ 21s 2s/step
```

```
Accuracy: 0.509009009009009
```

```
Precision : 0.2590901712523334
```

```
Recall: 0.509009009009009
```

```
F1-score: 0.3433911523463762
```

```
Classification Report:
```

	precision	recall	f1-score	support
Bengin cases	0.00	0.00	0.00	25
Malignant cases	0.51	1.00	0.67	113

Normal cases	0.00	0.00	0.00	84
accuracy			0.51	222
macro avg	0.17	0.33	0.22	222
weighted avg	0.26	0.51	0.34	222

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

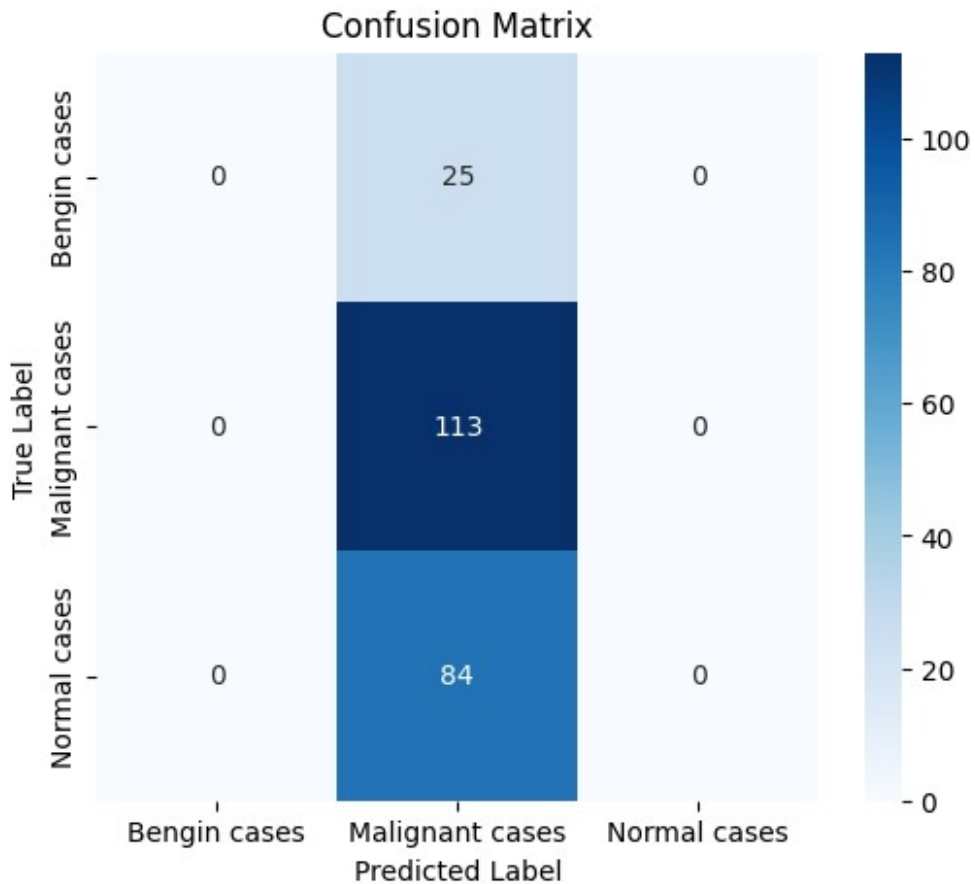
```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```



InceptionV3

```
from tensorflow.keras.applications import InceptionV3

def build_inceptionv3_model(input_shape, num_classes):
    base_model = InceptionV3(weights='imagenet', include_top=False,
input_shape=input_shape)
    base_model.trainable = False # Freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

```

model = build_inceptionv3_model(input_shape, num_classes)

history = model.fit(train_gen, epochs=5, validation_data=val_gen)

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/inception_v3/
inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 _____ 61s 1us/step
Epoch 1/5
24/24 _____ 112s 4s/step - accuracy: 0.5902 - loss:
1.1666 - val_accuracy: 0.8532 - val_loss: 0.4702
Epoch 2/5
24/24 _____ 83s 3s/step - accuracy: 0.8178 - loss:
0.5016 - val_accuracy: 0.8624 - val_loss: 0.3643
Epoch 3/5
24/24 _____ 17s 633ms/step - accuracy: 0.8078 - loss:
0.4777 - val_accuracy: 0.8624 - val_loss: 0.4192
Epoch 4/5
24/24 _____ 14s 576ms/step - accuracy: 0.8070 - loss:
0.5035 - val_accuracy: 0.8624 - val_loss: 0.3581
Epoch 5/5
24/24 _____ 13s 556ms/step - accuracy: 0.8239 - loss:
0.4525 - val_accuracy: 0.8624 - val_loss: 0.3378

y_pred=model.predict(test_gen)
y_true = test_gen.classes
y_pred_classes = np.argmax(y_pred, axis=1)
evaluate_model('InceptionV3(', y_true, y_pred_classes)

WARNING:tensorflow:5 out of the last 15 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000001429366AC00> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors
with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more
details.
7/7 _____ 5s 540ms/step
Accuracy: 0.8513513513513513
Precision : 0.7601815050344461
Recall: 0.8513513513513513
F1-score: 0.8023281484447206
Classification Report:

```

	precision	recall	f1-score	support
Bengin cases	0.00	0.00	0.00	25

Malignant cases	0.93	0.98	0.95	113
Normal cases	0.76	0.93	0.84	84
accuracy			0.85	222
macro avg	0.56	0.64	0.60	222
weighted avg	0.76	0.85	0.80	222

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

C:\Users\compumarts\anaconda3\envs\tensorflow\Lib\site-packages\sklearn\metrics_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

```
_warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

