

```
In [1]: import os, shutil
from sklearn.model_selection import train_test_split

original_dataset_dir = 'The IQ-OTHNCCD lung cancer dataset'
base_dir = 'Split 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)
```

```
In [ ]: # 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)
```

```
In [3]: 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
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
In [4]: 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

```

In [6]:

```

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

```

In [7]:

```

# Directories for datasets
train_dir = 'Split Dataset/Train'
val_dir = 'Split Dataset/Validation'
test_dir = 'Split 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\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: 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)
```

Epoch 1/10

24/24 ————— 61s 2s/step - accuracy: 0.4830 - loss: 1.0645 - val_accuracy: 0.5138 - val_loss: 0.9653

Epoch 2/10

24/24 ————— 70s 2s/step - accuracy: 0.4791 - loss: 0.9595 - val_accuracy: 0.5229 - val_loss: 0.9419

Epoch 3/10

24/24 ————— 46s 2s/step - accuracy: 0.5144 - loss: 0.9558 - val_accuracy: 0.5138 - val_loss: 0.9464

Epoch 4/10

24/24 ————— 46s 2s/step - accuracy: 0.5222 - loss: 0.9391 - val_accuracy: 0.5046 - val_loss: 0.9054

Epoch 5/10

24/24 ————— 45s 2s/step - accuracy: 0.5170 - loss: 0.9235 - val_accuracy: 0.5138 - val_loss: 0.8993

Epoch 6/10

24/24 ————— 44s 2s/step - accuracy: 0.5261 - loss: 0.9126 - val_accuracy: 0.5321 - val_loss: 0.9014

Epoch 7/10

24/24 ————— 44s 2s/step - accuracy: 0.5261 - loss: 0.9159 - val_accuracy: 0.5596 - val_loss: 0.8948

Epoch 8/10

24/24 ————— 44s 2s/step - accuracy: 0.5627 - loss: 0.8974 - val_accuracy: 0.6055 - val_loss: 0.8884

Epoch 9/10

24/24 ————— 46s 2s/step - accuracy: 0.5418 - loss: 0.8761 - val_accuracy: 0.5780 - val_loss: 0.8527

Epoch 10/10

24/24 ————— 44s 2s/step - accuracy: 0.5822 - loss: 0.8873 - val_accuracy: 0.6147 - val_loss: 0.8304

In [12]: `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 887ms/step

Accuracy: 0.6126126126126126

Precision : 0.5885625427918257

Recall: 0.6126126126126126

F1-score: 0.5790360496242849

Classification Report:

	precision	recall	f1-score	support
Bengin cases	0.00	0.00	0.00	25
Malignant cases	0.78	0.58	0.67	113
Normal cases	0.51	0.83	0.63	84
accuracy			0.61	222
macro avg	0.43	0.47	0.43	222
weighted avg	0.59	0.61	0.58	222

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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.

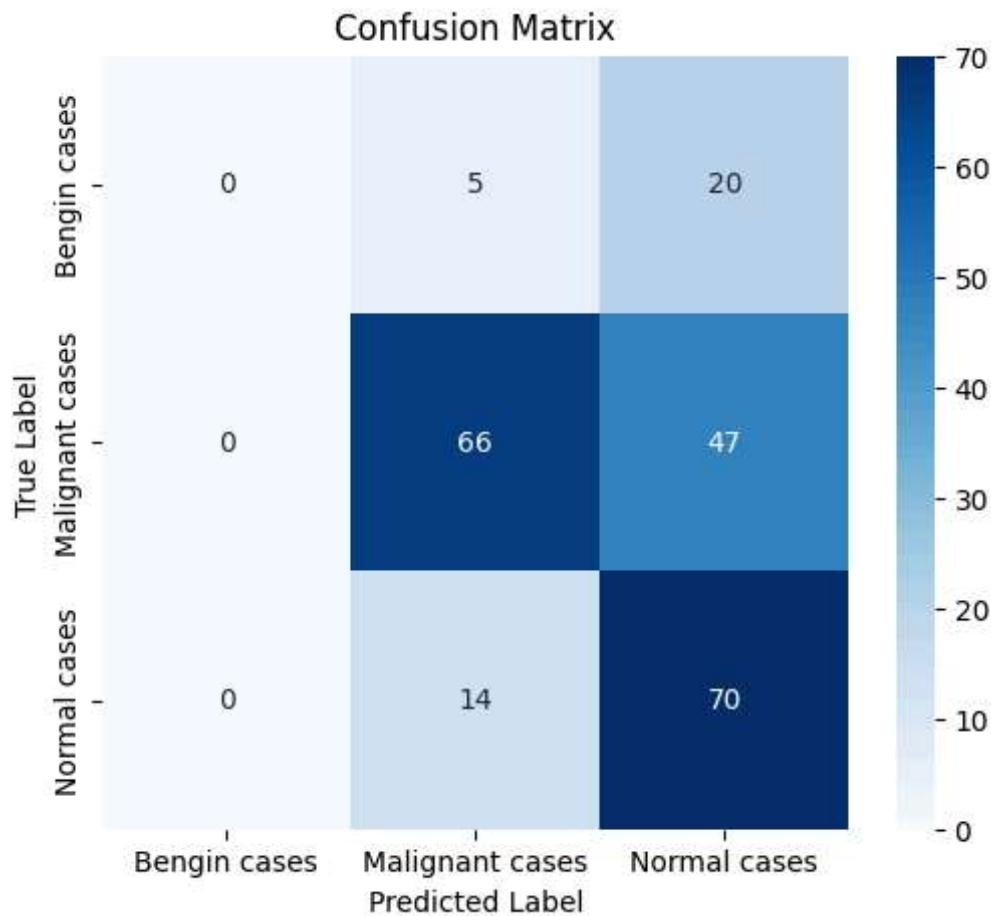
_warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])



```
In [13]: 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)
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5

94765736/94765736  **577s** 6us/step

Epoch 1/5

24/24  **119s** 5s/step - accuracy: 0.4517 - loss: 1.0666 - val_accuracy: 0.5138 - val_loss: 0.9412

Epoch 2/5

24/24  **103s** 4s/step - accuracy: 0.5039 - loss: 0.9535 - val_accuracy: 0.5138 - val_loss: 0.9483

Epoch 3/5

24/24  **104s** 4s/step - accuracy: 0.4935 - loss: 0.9612 - val_accuracy: 0.5138 - val_loss: 0.9481

Epoch 4/5

24/24  **577s** 25s/step - accuracy: 0.5052 - loss: 0.9567 - val_accuracy: 0.5138 - val_loss: 0.9421

Epoch 5/5

24/24  **101s** 4s/step - accuracy: 0.5013 - loss: 0.9528 - val_accuracy: 0.5138 - val_loss: 0.9452

```
In [14]: 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  **30s** 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\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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.

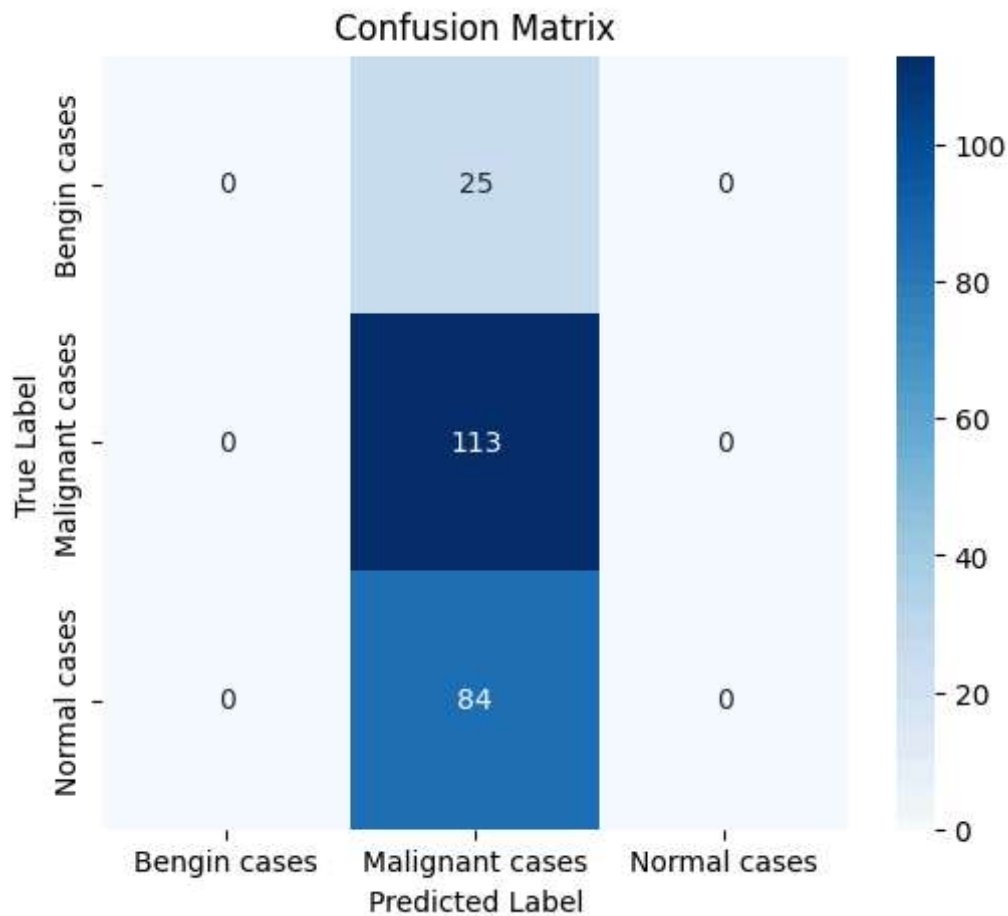
_warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics_classification.py:1731: 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", result.shape[0])



In [17]: `#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
```

In [19]: `model = build_efficientnetb0_model(input_shape, num_classes)`
`history = model.fit(train_gen, epochs=5, validation_data=val_gen)`

Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5

16705208/16705208  **98s** 6us/step

Epoch 1/5

24/24  **70s** 2s/step - accuracy: 0.4582 - loss: 1.0084 - val_accuracy: 0.3761 - val_loss: 0.9673

Epoch 2/5

24/24  **44s** 2s/step - accuracy: 0.4896 - loss: 0.9843 - val_accuracy: 0.5138 - val_loss: 0.9535

Epoch 3/5

24/24  **44s** 2s/step - accuracy: 0.4830 - loss: 0.9822 - val_accuracy: 0.5138 - val_loss: 0.9608

Epoch 4/5

24/24  **82s** 2s/step - accuracy: 0.4843 - loss: 0.9772 - val_accuracy: 0.5138 - val_loss: 0.9647

Epoch 5/5

24/24  **42s** 2s/step - accuracy: 0.4961 - loss: 0.9794 - val_accuracy: 0.5138 - val_loss: 0.9552

```
In [20]: 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 0x0000028AE43507C0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracinings 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  **16s** 2s/step

Accuracy: 0.509009009009009

Precision : 0.2590901712523334

Recall: 0.509009009009009

F1-score: 0.3433911523463762

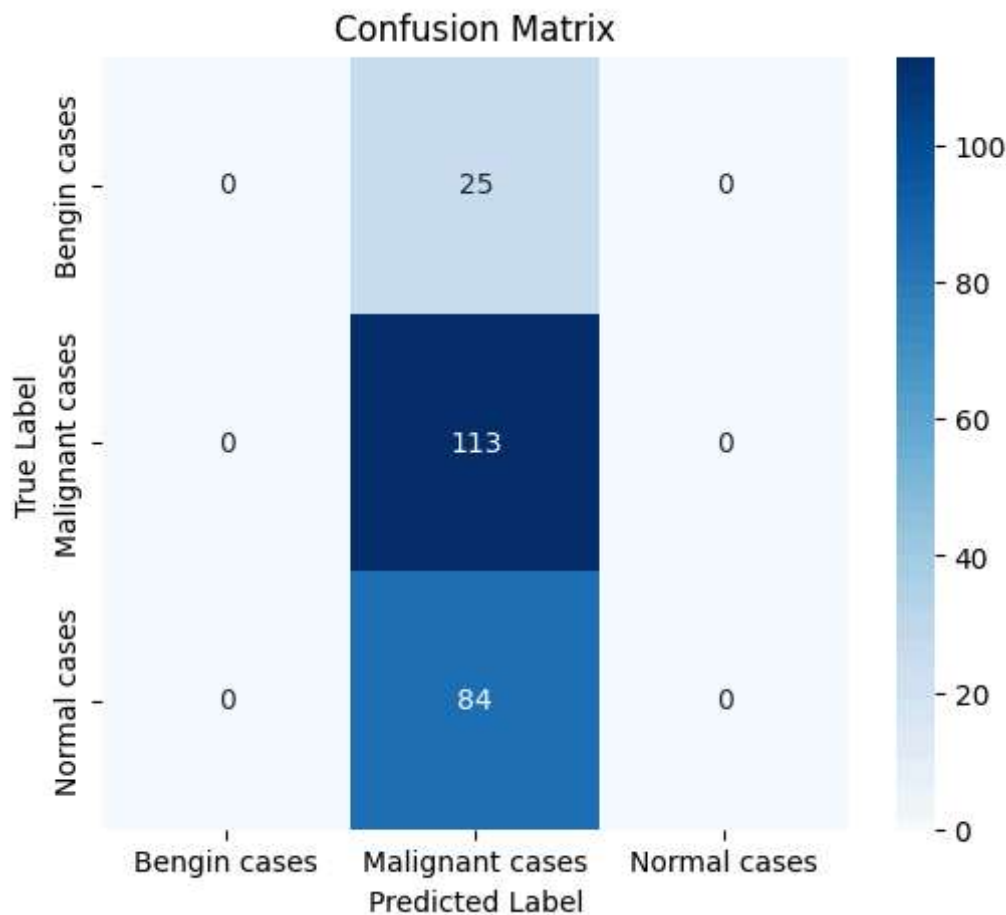
Classification Report:

	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\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])

```



```

In [21]: #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

```

```

In [22]: 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  **536s** 6us/step

Epoch 1/5

24/24  **91s** 3s/step - accuracy: 0.6723 - loss: 0.9054 - val_accuracy: 0.7706 - val_loss: 0.5981

Epoch 2/5

24/24  **74s** 3s/step - accuracy: 0.8081 - loss: 0.5008 - val_accuracy: 0.8716 - val_loss: 0.3774

Epoch 3/5

24/24  **74s** 3s/step - accuracy: 0.8342 - loss: 0.4223 - val_accuracy: 0.8716 - val_loss: 0.3586

Epoch 4/5

24/24  **74s** 3s/step - accuracy: 0.8420 - loss: 0.4330 - val_accuracy: 0.8716 - val_loss: 0.3954

Epoch 5/5

24/24  **74s** 3s/step - accuracy: 0.8486 - loss: 0.4071 - val_accuracy: 0.8716 - val_loss: 0.3526

```

In [23]: 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_distributed at 0x0000028B71B2F920> triggered tf.function retracing. Tracing is expensive and the excessive number of tracinings 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  **24s** 3s/step

Accuracy: 0.8468468468468469

Precision : 0.7695750471588744

Recall: 0.8468468468468469

F1-score: 0.8010758735137937

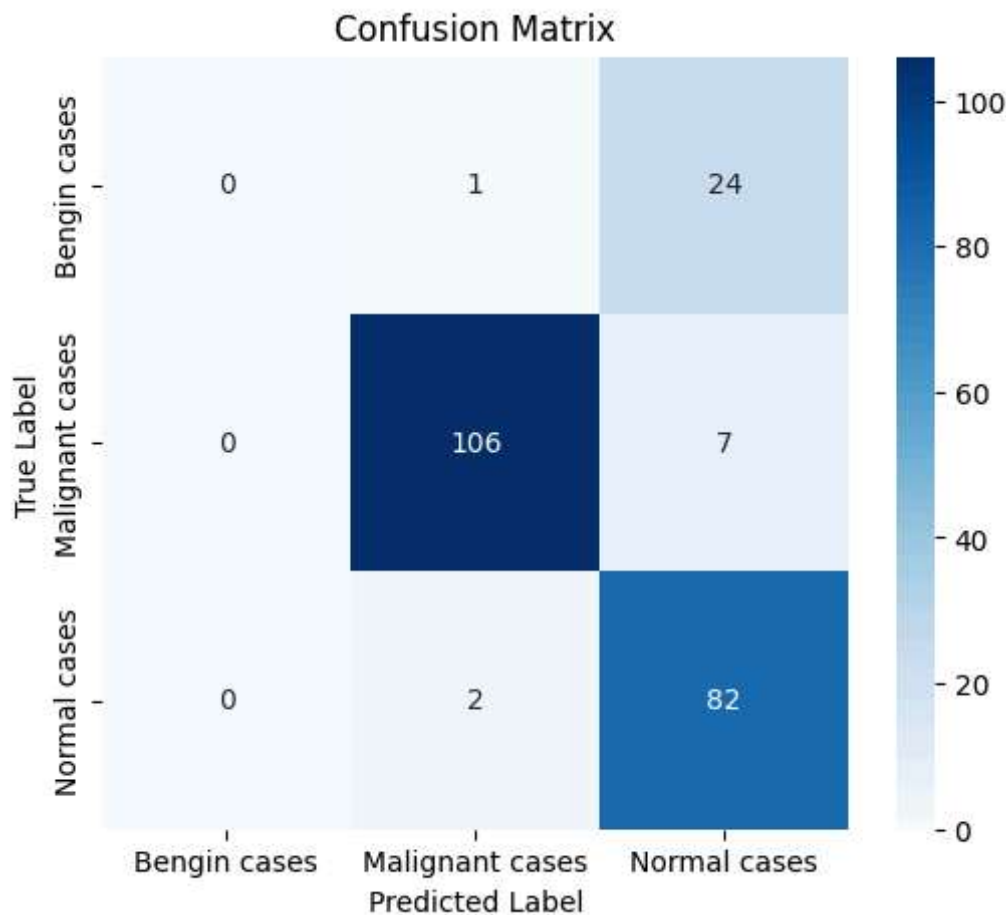
Classification Report:

	precision	recall	f1-score	support
Bengin cases	0.00	0.00	0.00	25
Malignant cases	0.97	0.94	0.95	113
Normal cases	0.73	0.98	0.83	84
accuracy			0.85	222
macro avg	0.57	0.64	0.60	222
weighted avg	0.77	0.85	0.80	222

```

C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])
C:\Users\ale1x\anaconda3\envs\Practical_Image_Processing\Lib\site-packages\sklearn\metrics\_classification.py:1731: 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", result.shape[0])

```



In []:

In []: