ENHANCED CHOLEDOCH CANCER DETECTION USING DEEP LEARNING TECHNIQUES

BRANCH: BSC Software Systems

SEMESTER: VI

BATCH NO: 18

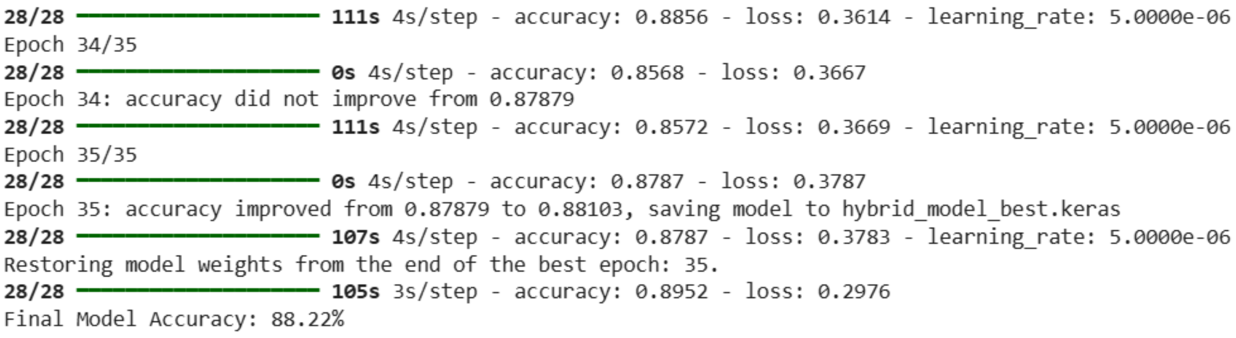
1. HYBRID MODEL (MOBILENET + EFFECIENTNET):

```
# -*- coding: utf-8 -*-
"""Hybrid-mobilenet&Efficient.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1wa93rw-g058-
a7onzNGeaz_n0Kx2XGx3

HYBRID(efficientNetB0,Mobilenetv2)
"""

import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0, MobileNetV2
from tensorflow.keras.applications.efficientnet import preprocess_input
as efficientnet_preprocess
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
as mobilenet_preprocess
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D,
Dropout, Concatenate, Input
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau, EarlyStopping

# Dataset path
BASE_DIR = '/content/drive/MyDrive/Choledoch_RGB/Choledoch_RGB'

# Parameters
BATCH_SIZE = 32
IMG_SIZE = (224, 224)
EPOCHS = 20  # First phase
FINE_TUNE_EPOCHS = 15  # Fine-tuning phase
LEARNING_RATE = 1e-4

# Data Augmentation (Hybrid Preprocessing)
train_datagen = ImageDataGenerator(
    preprocessing_function=efficientnet_preprocess,  # Using EfficientNet
preprocessing
    rotation_range=30,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    BASE_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

# Get number of classes
NUM_CLASSES = len(train_generator.class_indices)

# Input Layer
input_tensor = Input(shape=(IMG_SIZE[0], IMG_SIZE[1], 3))

# Load EfficientNetB0
efficientnet_base = EfficientNetB0(weights='imagenet', include_top=False,
input_tensor=input_tensor)
efficientnet_base.trainable = False  # Freeze base layers
efficientnet_output = GlobalAveragePooling2D()(efficientnet_base.output)

# Load MobileNetV2
mobilenet_base = MobileNetV2(weights='imagenet', include_top=False,
input_tensor=input_tensor)
mobilenet_base.trainable = False  # Freeze base layers
mobilenet_output = GlobalAveragePooling2D()(mobilenet_base.output)

# Merge Features from Both Models
merged_features = Concatenate()([efficientnet_output, mobilenet_output])

# Custom Fully Connected Layers
x = Dense(1024, activation='relu')(merged_features)
x = Dropout(0.4)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(NUM_CLASSES, activation='softmax')(x)

# Define Hybrid Model
model = Model(inputs=input_tensor, outputs=predictions)

# Compile Model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_R
ATE),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Callbacks
checkpoint = ModelCheckpoint('hybrid_model_best.keras',
monitor='accuracy', save_best_only=True, verbose=1)
```

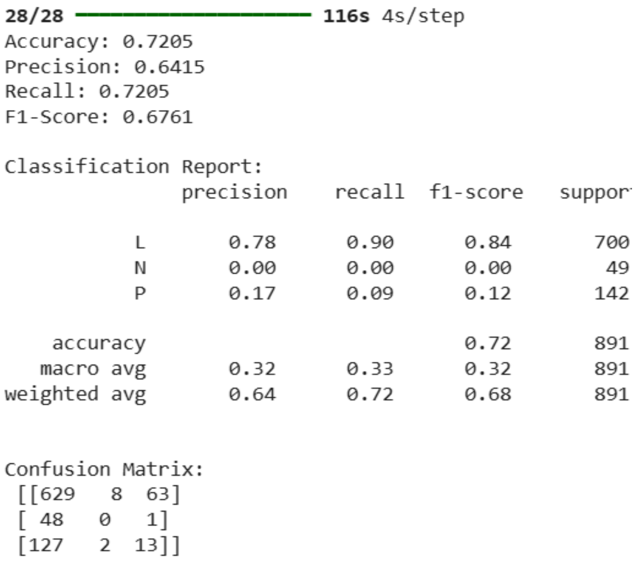
```
reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.5, patience=3,
verbose=1)
early_stopping = EarlyStopping(monitor='loss', patience=5, verbose=1,
restore_best_weights=True)

# Train Hybrid Model (Phase 1 - Frozen Base)
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    callbacks=[checkpoint, reduce_lr, early_stopping]
)

# Unfreeze Some Layers for Fine-Tuning
for layer in efficientnet_base.layers[-20:]:
    layer.trainable = True
for layer in mobilenet_base.layers[-20:]:
    layer.trainable = True

# Recompile Model for Fine-Tuning
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train Hybrid Model (Phase 2 - Fine-Tuning)
history_fine = model.fit(
    train_generator,
    epochs=EPOCHS + FINE_TUNE_EPOCHS,
    callbacks=[checkpoint, reduce_lr, early_stopping],
    initial_epoch=history.epoch[-1]
)

# Save Final Model
model.save('/content/drive/MyDrive/Hybrid-Efficientnet-mobilenet.keras')

# Evaluate Model
loss, accuracy = model.evaluate(train_generator)
print(f'Final Model Accuracy: {accuracy * 100:.2f}%')

from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report, confusion_matrix
import numpy as np

# Get true labels and predictions
y_true = train_generator.classes
y_pred = np.argmax(model.predict(train_generator), axis=1)

# Accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f'Accuracy: {accuracy:.4f}')

# Precision, Recall, F1-Score (Macro and Weighted for multi-class)
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')

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-Score: {f1:.4f}')
```

```
# Detailed Classification Report
print("\nClassification Report:\n", classification_report(y_true, y_pred,
target_names=list(train_generator.class_indices.keys())))

# Confusion Matrix
print("\nConfusion Matrix:\n", confusion_matrix(y_true, y_pred))

import matplotlib.pyplot as plt

# Plot Validation Accuracy and Loss
def plot_validation_metrics(history):
    plt.figure(figsize=(10, 4))

    # Validation Accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Validation Accuracy',
color='blue')
    plt.title('Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    # Validation Loss
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Validation Loss',
color='red')
    plt.title('Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.tight_layout()
    plt.show()

# Call the function for both training phases
plot_validation_metrics(history)       # Phase 1 (Frozen Base)
plot_validation_metrics(history_fine)  # Phase 2 (Fine-Tuning)
```

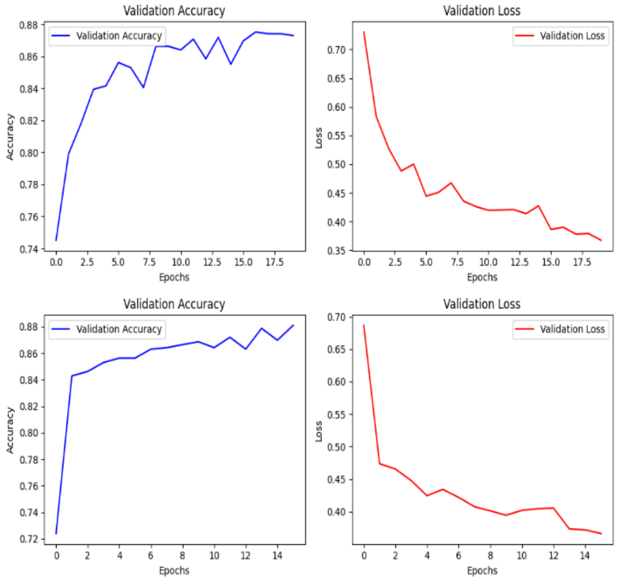
RESULTS :



```
28/28 ━━━━━━━━━━━━━━━━━━━━ 111s 4s/step - accuracy: 0.8856 - loss: 0.3614 - learning_rate: 5.0000e-06
Epoch 34/35
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.8568 - loss: 0.3667
Epoch 34: accuracy did not improve from 0.87879
28/28 ━━━━━━━━━━━━━━━━━━━━ 111s 4s/step - accuracy: 0.8572 - loss: 0.3669 - learning_rate: 5.0000e-06
Epoch 35/35
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.8787 - loss: 0.3787
Epoch 35: accuracy improved from 0.87879 to 0.88103, saving model to hybrid_model_best.keras
28/28 ━━━━━━━━━━━━━━━━━━━━ 107s 4s/step - accuracy: 0.8787 - loss: 0.3783 - learning_rate: 5.0000e-06
Restoring model weights from the end of the best epoch: 35.
28/28 ━━━━━━━━━━━━━━━━━━━━ 105s 3s/step - accuracy: 0.8952 - loss: 0.2976
Final Model Accuracy: 88.22%
```



```
28/28 ━━━━━━━━━━━━━━━━━━━━ 116s 4s/step
Accuracy: 0.7205
Precision: 0.6415
Recall: 0.7205
F1-Score: 0.6761

Classification Report:
              precision    recall  f1-score   suppor

           L       0.78      0.90      0.84       700
           N       0.00      0.00      0.00        49
           P       0.17      0.09      0.12       142

    accuracy                           0.72       891
   macro avg       0.32      0.33      0.32       891
weighted avg       0.64      0.72      0.68       891


Confusion Matrix:
 [[629   8  63]
 [ 48   0   1]
 [127   2  13]]
```


Validation Accuracy
Validation Accuracy
Accuracy
Epochs
Validation Loss
Validation Loss
Loss
Epochs
Validation Accuracy
Validation Accuracy
Accuracy
Epochs
Validation Loss
Validation Loss
Loss
Epochs