```
 import numpy as np
 import cv2
 import os
 import random
 import matplotlib.pyplot as plt
 import pickle
 from sklearn.model_selection import train_test_split
 from tensorflow.keras.models import Sequential
 from tensorflow.keras.models import Model
 import tensorflow as tf
 from sklearn.metrics import classification_report,confusion_matrix

import tensorflow as tf

os.environ["KERAS_BACKEND"] = "tensorflow"

import keras
from keras import layers
from keras import backend

import tensorflow_datasets as tfds

tfds.disable_progress_bar()

 DIRECTORY = r'/kaggle/input/labeled-optical-coherence-tomography-
oct/Dataset - train+val+test/train'
 CATEGORIES = ['CNV', 'DME', 'DRUSEN','NORMAL']
 IMG_SIZE = 224
 patch_size=4
 expansion_factor=2
 train_data = []

for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    #print(folder)
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder, img)
        #print(img_path)
        img_arr = cv2.imread(img_path)
        img_arr = cv2.resize(img_arr, (IMG_SIZE, IMG_SIZE))
        #plt.imshow(img_arr)
        #break
        train_data.append([img_arr, label])

 len(train_data)

76515

random.shuffle(train_data)
```

```python
 X_train = []
 y_train = []

 for features, labels in train_data:
     X_train.append(features)
     y_train.append(labels)

 DIRECTORY = r'/kaggle/input/labeled-optical-coherence-tomography-
oct/Dataset - train+val+test/val'
 CATEGORIES = ['CNV', 'DME', 'DRUSEN','NORMAL']
 IMG_SIZE = 224
 patch_size=4
 expansion_factor=2
 test_data = []

for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    #print(folder)
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder, img)
        #print(img_path)
        img_arr = cv2.imread(img_path)
        img_arr = cv2.resize(img_arr, (IMG_SIZE, IMG_SIZE))
        #plt.imshow(img_arr)
        #break
        test_data.append([img_arr, label])

len(test_data)
```

```
21861
```

```python
X_test = []
y_test = []
for features, labels in test_data:
     X_test.append(features)
     y_test.append(labels)

 X_train = np.array(X_train)
 y_train = np.array(y_train)
 X_test = np.array(X_test)
 y_test = np.array(y_test)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((76515, 224, 224, 3), (21861, 224, 224, 3), (76515,), (21861,))
```

```python
def conv_block(x, filters=16, kernel_size=3, strides=2):
    conv_layer = layers.Conv2D(
        filters,
        kernel_size,
```

```
        strides=strides,
        activation=keras.activations.swish,
        padding="same",
    )
    return conv_layer(x)



def correct_pad(inputs, kernel_size):
    img_dim = 2 if backend.image_data_format() == "channels_first"
else 1
    input_size = inputs.shape[img_dim : (img_dim + 2)]
    if isinstance(kernel_size, int):
        kernel_size = (kernel_size, kernel_size)
    if input_size[0] is None:
        adjust = (1, 1)
    else:
        adjust = (1 - input_size[0] % 2, 1 - input_size[1] % 2)
    correct = (kernel_size[0] // 2, kernel_size[1] // 2)
    return (
        (correct[0] - adjust[0], correct[0]),
        (correct[1] - adjust[1], correct[1]),
    )

def inverted_residual_block(x, expanded_channels, output_channels,
strides=1):
    m = layers.Conv2D(expanded_channels, 1, padding="same",
use_bias=False)(x)
    m = layers.BatchNormalization()(m)
    m = keras.activations.swish(m)

    if strides == 2:
        m = layers.ZeroPadding2D(padding=correct_pad(m, 3))(m)
    m = layers.DepthwiseConv2D(
        3, strides=strides, padding="same" if strides == 1 else
"valid", use_bias=False
    )(m)
    m = layers.BatchNormalization()(m)
    m = keras.activations.swish(m)

    m = layers.Conv2D(output_channels, 1, padding="same",
use_bias=False)(m)
    m = layers.BatchNormalization()(m)

    if keras.ops.equal(x.shape[-1], output_channels) and strides == 1:
        return layers.Add()([m, x])
    return m

def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
```

```
        x = layers.Dense(units, activation=keras.activations.swish)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x


def transformer_block(x, transformer_layers, projection_dim,
num_heads=2):
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(x)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, x])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(
            x3,
            hidden_units=[x.shape[-1] * 2, x.shape[-1]],
            dropout_rate=0.1,
        )
        # Skip connection 2.
        x = layers.Add()([x3, x2])

    return x

def mobilevit_block(x, num_blocks, projection_dim, strides=1):
    # Local projection with convolutions.
    local_features = conv_block(x, filters=projection_dim,
strides=strides)
    local_features = conv_block(
        local_features, filters=projection_dim, kernel_size=1,
strides=strides
    )

    # Unfold into patches and then pass through Transformers.
    num_patches = int((local_features.shape[1] *
local_features.shape[2]) / patch_size)
    non_overlapping_patches = layers.Reshape((patch_size, num_patches,
projection_dim))(
        local_features
    )
    global_features = transformer_block(
        non_overlapping_patches, num_blocks, projection_dim
    )

    # Fold into conv-like feature-maps.
```

```
    folded_feature_map = layers.Reshape((*local_features.shape[1:-1],
projection_dim))(
        global_features
    )

    # Apply point-wise conv -> concatenate with the input features.
    folded_feature_map = conv_block(
        folded_feature_map, filters=x.shape[-1], kernel_size=1,
strides=strides
    )
    local_global_features = layers.Concatenate(axis=-1)([x,
folded_feature_map])

    # Fuse the local and global features using a convoluion layer.
    local_global_features = conv_block(
        local_global_features, filters=projection_dim, strides=strides
    )

    return local_global_features


def create_mobilevit(num_classes=5):
    inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    x = layers.Rescaling(scale=1.0 / 255)(inputs)  # Normalize input

    # Initial conv-stem -> MV2 block
    x = conv_block(x, filters=16)
    x = inverted_residual_block(x, expanded_channels=16)

    # Downsampling with MV2 Block
    x = inverted_residual_block(x, expanded_channels=24, strides=2)
    x = inverted_residual_block(x, expanded_channels=24)
    x = inverted_residual_block(x, expanded_channels=24)

    # First MV2 -> MobileViT Block
    x = inverted_residual_block(x, expanded_channels=48, strides=2)
    x = mobilevit_block(x, num_blocks=2, projection_dim=64)

    # Second MV2 -> MobileViT Block
    x = inverted_residual_block(x, expanded_channels=64, strides=2)

    # Global pooling and classification head
    x = layers.GlobalAveragePooling2D()(x)
    outputs = layers.Dense(num_classes, activation="softmax")(x)

    model = keras.Model(inputs, outputs)
    return model

# Helper functions (assuming you have them defined)
def conv_block(x, filters, kernel_size=3, strides=1, padding="same"):
```

```
    x = layers.Conv2D(filters, kernel_size, strides=strides,
padding=padding, use_bias=False)(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    return x

def inverted_residual_block(x, expanded_channels, strides=1):
    input_channels = x.shape[-1]

    # Expansion
    expanded = layers.Conv2D(expanded_channels, 1, use_bias=False)(x)
    expanded = layers.BatchNormalization()(expanded)
    expanded = layers.ReLU()(expanded)

    # Depthwise Convolution
    expanded = layers.DepthwiseConv2D(3, strides=strides,
padding="same", use_bias=False)(expanded)
    expanded = layers.BatchNormalization()(expanded)
    expanded = layers.ReLU()(expanded)

    # Projection
    projected = layers.Conv2D(input_channels if strides == 1 else
expanded_channels, 1, use_bias=False)(expanded)
    projected = layers.BatchNormalization()(projected)

    # Skip Connection (if no stride)
    if strides == 1 and input_channels == expanded_channels:
        x = layers.Add()([x, projected])

    return projected

def mobilevit_block(x, num_blocks, projection_dim):
    # Placeholder function for MobileViT block implementation
    # Implement MobileViT logic here
    return x

# Create and compile the model
model = create_mobilevit()
model.summary()

Model: "functional"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)              │ (None, 224, 224, 3)          │
0 │
```

```
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ rescaling (Rescaling)                │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d (Conv2D)                      │ (None, 224, 224, 16)        │
432 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ batch_normalization                  │ (None, 224, 224, 16)        │
64 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ re_lu (ReLU)                         │ (None, 224, 224, 16)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 224, 224, 16)        │
256 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ batch_normalization_1                │ (None, 224, 224, 16)        │
64 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ re_lu_1 (ReLU)                       │ (None, 224, 224, 16)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ depthwise_conv2d (DepthwiseConv2D)   │ (None, 224, 224, 16)        │
144 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ batch_normalization_2                │ (None, 224, 224, 16)        │
64 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ re_lu_2 (ReLU)                       │ (None, 224, 224, 16)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 224, 224, 16)        │
```

```
256 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ batch_normalization_3                │ (None, 224, 224, 16)        │
64 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 224, 224, 24)        │
384 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ batch_normalization_4                │ (None, 224, 224, 24)        │
96 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ re_lu_3 (ReLU)                       │ (None, 224, 224, 24)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ depthwise_conv2d_1 (DepthwiseConv2D) │ (None, 112, 112, 24)        │
216 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ batch_normalization_5                │ (None, 112, 112, 24)        │
96 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ re_lu_4 (ReLU)                       │ (None, 112, 112, 24)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ conv2d_4 (Conv2D)                    │ (None, 112, 112, 24)        │
576 │
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ batch_normalization_6                │ (None, 112, 112, 24)        │
96 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
───────────────┤
│ conv2d_5 (Conv2D)                    │ (None, 112, 112, 24)        │
576 │
```

```
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ batch_normalization_7               │ (None, 112, 112, 24)      │
96 │
│ (BatchNormalization)                │                           │
│
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ re_lu_5 (ReLU)                      │ (None, 112, 112, 24)      │
0 │
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ depthwise_conv2d_2 (DepthwiseConv2D) │ (None, 112, 112, 24)      │
216 │
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ batch_normalization_8               │ (None, 112, 112, 24)      │
96 │
│ (BatchNormalization)                │                           │
│
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ re_lu_6 (ReLU)                      │ (None, 112, 112, 24)      │
0 │
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ conv2d_6 (Conv2D)                   │ (None, 112, 112, 24)      │
576 │
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ batch_normalization_9               │ (None, 112, 112, 24)      │
96 │
│ (BatchNormalization)                │                           │
│
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ conv2d_7 (Conv2D)                   │ (None, 112, 112, 24)      │
576 │
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ batch_normalization_10              │ (None, 112, 112, 24)      │
96 │
│ (BatchNormalization)                │                           │
│
├─────────────────────────────────────┼───────────────────────────┤
─────────────────┤
│ re_lu_7 (ReLU)                      │ (None, 112, 112, 24)      │
0 │
├─────────────────────────────────────┼───────────────────────────┤
```

```
────────────────┤
│ depthwise_conv2d_3 (DepthwiseConv2D) │ (None, 112, 112, 24)        │
216 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ batch_normalization_11               │ (None, 112, 112, 24)        │
96 │
│ (BatchNormalization)                 │                              │
│
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ re_lu_8 (ReLU)                       │ (None, 112, 112, 24)        │
0 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ conv2d_8 (Conv2D)                    │ (None, 112, 112, 24)        │
576 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ batch_normalization_12               │ (None, 112, 112, 24)        │
96 │
│ (BatchNormalization)                 │                              │
│
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ conv2d_9 (Conv2D)                    │ (None, 112, 112, 48)        │
1,152 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ batch_normalization_13               │ (None, 112, 112, 48)        │
192 │
│ (BatchNormalization)                 │                              │
│
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ re_lu_9 (ReLU)                       │ (None, 112, 112, 48)        │
0 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ depthwise_conv2d_4 (DepthwiseConv2D) │ (None, 56, 56, 48)          │
432 │
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
│ batch_normalization_14               │ (None, 56, 56, 48)          │
192 │
│ (BatchNormalization)                 │                              │
│
├─────────────────────────────────────┼──────────────────────────────┤
────────────────┤
```

```
│ re_lu_10 (ReLU)                      │ (None, 56, 56, 48)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_10 (Conv2D)                   │ (None, 56, 56, 48)          │
2,304 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_15               │ (None, 56, 56, 48)          │
192 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_11 (Conv2D)                   │ (None, 56, 56, 64)          │
3,072 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_16               │ (None, 56, 56, 64)          │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ re_lu_11 (ReLU)                      │ (None, 56, 56, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ depthwise_conv2d_5 (DepthwiseConv2D) │ (None, 28, 28, 64)          │
576 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_17               │ (None, 28, 28, 64)          │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ re_lu_12 (ReLU)                      │ (None, 28, 28, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_12 (Conv2D)                   │ (None, 28, 28, 64)          │
4,096 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_18               │ (None, 28, 28, 64)          │
256 │
│ (BatchNormalization)                 │                             │
```

```
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ global_average_pooling2d             │ (None, 64)                  │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense (Dense)                        │ (None, 5)                   │
325 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 19,421 (75.86 KB)

 Trainable params: 18,189 (71.05 KB)

 Non-trainable params: 1,232 (4.81 KB)
```

```
class_weights = {0:1.1347,
                 1:0.9095,
                 2:0.9811}

 from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001)
,

loss='sparse_categorical_crossentropy',
                                                  metrics=['accuracy'])

 filepath = "/kaggle/working/My_model.keras"
 checkpoint = ModelCheckpoint(filepath, monitor="val_accuracy",
                              verbose=1, save_best_only=True,
                              mode='max')
 reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',factor=1e-1,
patience=5, verbose=1)
 callbacks = [checkpoint, reduce_lr]

history = model.fit(X_train, y_train,
                    validation_data=[X_test, y_test],
                    class_weight=class_weights,
                    callbacks=callbacks,
                    epochs=20, batch_size=16)
```