In [ ]:

```
import os
from PIL import Image
import numpy as np

# Define the path to the root directory where your data is stored
data_root = '/home/admin1/Downloads/small datast'

# Create empty lists to store images and labels
images = []
labels = []

# Define a dictionary to map folder names to class labels
class_mapping = {
    'Age-related macular degeneration (ARMD )': 0,
    'Branch retinal vein occlusion(BRVO)': 1,
    'Central retinal vein occlusion (CRVO)': 2,
    'Cotton wool spots (CWS)': 3,
    'Central serous retinopathy (CSR)' : 4,
    'Exudative detachment of the retina (EDN)': 5,
    'Microaneurysms (MCA)': 6,
    'Optic disc edema (ODE)' : 7,
    'Posterior retinal hemorrhage (PRH)' : 8,
    'Retinal hemorrhages (HR)' : 9,
    'Tortuous vessels (TV)' : 10,
    'Vitreous hemorrhage ( VH )' : 11


}

# Iterate through each folder in the root directory
for folder_name, class_label in class_mapping.items():
    folder_path = os.path.join(data_root, folder_name)

    # Iterate through each image file in the folder
    for image_file in os.listdir(folder_path):
        if image_file.endswith('.jpg') or image_file.endswith('.jpeg') or image_file.end
swith('.png'):
            image_path = os.path.join(folder_path, image_file)

            # Load and preprocess the image
            img = Image.open(image_path)
            img = img.resize((299, 299))  # Resize to a suitable input size
            img = np.array(img) / 255.0  # Normalize pixel values to [0, 1]

            # Append the preprocessed image and its label to the lists
            images.append(img)
            labels.append(class_label)

# Convert the lists to NumPy arrays
images = np.array(images)
labels = np.array(labels)
```

In [ ]:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint

# Define your data directory
data_dir = '/home/admin1/Downloads/small datast'

# Define image size and batch size
```

```
image_size = (224, 224)
batch_size = 32

# Create data generators with data augmentation
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2  # Split data into training and validation sets
)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'  # Use the training subset
)

val_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'  # Use the validation subset
)

# Load the pre-trained GoogleNet (Inception V1) model without top (fully connected) layer
s
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299, 3
))

# Add custom layers for your classification task
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(12, activation='softmax')(x)  # 12 output classes

model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the layers of the pre-trained model
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accur
acy'])

# Define a callback to save the best model
checkpoint = ModelCheckpoint('best_model.h5', save_best_only=True, monitor='val_loss', m
ode='min', verbose=1)

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=20,
    validation_data=val_generator,
    validation_steps=len(val_generator),
    callbacks=[checkpoint]
)

# Evaluate the model on the test set
test_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
```

```
    shuffle=False
)

test_loss, test_acc = model.evaluate(test_generator, steps=len(test_generator))
print("Testing Accuracy:", test_acc)

# Save the model
model.save('InceptionV3.h5')
```

```
2024-02-27 16:09:58.463264: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-02-27 16:09:58.483598: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2024-02-27 16:09:58.574224: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2024-02-27 16:09:58.574288: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-02-27 16:09:58.587747: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2024-02-27 16:09:58.623530: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2024-02-27 16:09:58.624655: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-02-27 16:09:59.243192: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

```
Found 1186 images belonging to 12 classes.
Found 293 images belonging to 12 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 93s 1us/step
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

```
Epoch 1/20
38/38 [==============================] - ETA: 0s - loss: 1.9727 - accuracy: 0.3870
Epoch 1: val_loss improved from inf to 1.40520, saving model to best_model.h5
```

```
/home/admin1/anaconda3/lib/python3.9/site-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
38/38 [==============================] - 57s 1s/step - loss: 1.9727 - accuracy: 0.3870 - val_loss: 1.4052 - val_accuracy: 0.5495
Epoch 2/20
38/38 [==============================] - ETA: 0s - loss: 1.3928 - accuracy: 0.5422
Epoch 2: val_loss improved from 1.40520 to 1.21896, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 1.3928 - accuracy: 0.5422 - val_loss: 1.2190 - val_accuracy: 0.6143
Epoch 3/20
38/38 [==============================] - ETA: 0s - loss: 1.1515 - accuracy: 0.6206
Epoch 3: val_loss did not improve from 1.21896
38/38 [==============================] - 51s 1s/step - loss: 1.1515 - accuracy: 0.6206 - val_loss: 1.2957 - val_accuracy: 0.5529
Epoch 4/20
38/38 [==============================] - ETA: 0s - loss: 1.0993 - accuracy: 0.6391
Epoch 4: val_loss improved from 1.21896 to 1.09332, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 1.0993 - accuracy: 0.6391 - val_loss: 1.0933 - val_accuracy: 0.6246
Epoch 5/20
38/38 [==============================] - ETA: 0s - loss: 0.9529 - accuracy: 0.6863
Epoch 5: val_loss improved from 1.09332 to 1.04970, saving model to best_model.h5
```

```
38/38 [==============================] - 52s 1s/step - loss: 0.9529 - accuracy: 0.6863 -
val_loss: 1.0497 - val_accuracy: 0.6485
Epoch 6/20
38/38 [==============================] - ETA: 0s - loss: 0.8992 - accuracy: 0.6965
Epoch 6: val_loss improved from 1.04970 to 0.98024, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 0.8992 - accuracy: 0.6965 -
val_loss: 0.9802 - val_accuracy: 0.6655
Epoch 7/20
38/38 [==============================] - ETA: 0s - loss: 0.7563 - accuracy: 0.7445
Epoch 7: val_loss improved from 0.98024 to 0.77553, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 0.7563 - accuracy: 0.7445 -
val_loss: 0.7755 - val_accuracy: 0.7270
Epoch 8/20
38/38 [==============================] - ETA: 0s - loss: 0.7403 - accuracy: 0.7496
Epoch 8: val_loss did not improve from 0.77553
38/38 [==============================] - 51s 1s/step - loss: 0.7403 - accuracy: 0.7496 -
val_loss: 0.8816 - val_accuracy: 0.6689
Epoch 9/20
38/38 [==============================] - ETA: 0s - loss: 0.7177 - accuracy: 0.7513
Epoch 9: val_loss did not improve from 0.77553
38/38 [==============================] - 51s 1s/step - loss: 0.7177 - accuracy: 0.7513 -
val_loss: 0.9618 - val_accuracy: 0.6689
Epoch 10/20
38/38 [==============================] - ETA: 0s - loss: 0.7047 - accuracy: 0.7698
Epoch 10: val_loss did not improve from 0.77553
38/38 [==============================] - 52s 1s/step - loss: 0.7047 - accuracy: 0.7698 -
val_loss: 0.9048 - val_accuracy: 0.7167
Epoch 11/20
38/38 [==============================] - ETA: 0s - loss: 0.6696 - accuracy: 0.7841
Epoch 11: val_loss did not improve from 0.77553
38/38 [==============================] - 52s 1s/step - loss: 0.6696 - accuracy: 0.7841 -
val_loss: 0.7939 - val_accuracy: 0.7304
Epoch 12/20
38/38 [==============================] - ETA: 0s - loss: 0.6445 - accuracy: 0.7656
Epoch 12: val_loss did not improve from 0.77553
38/38 [==============================] - 52s 1s/step - loss: 0.6445 - accuracy: 0.7656 -
val_loss: 0.8618 - val_accuracy: 0.7235
Epoch 13/20
38/38 [==============================] - ETA: 0s - loss: 0.7085 - accuracy: 0.7614
Epoch 13: val_loss did not improve from 0.77553
38/38 [==============================] - 52s 1s/step - loss: 0.7085 - accuracy: 0.7614 -
val_loss: 0.8527 - val_accuracy: 0.7235
Epoch 14/20
38/38 [==============================] - ETA: 0s - loss: 0.6549 - accuracy: 0.7656
Epoch 14: val_loss improved from 0.77553 to 0.76949, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 0.6549 - accuracy: 0.7656 -
val_loss: 0.7695 - val_accuracy: 0.7099
Epoch 15/20
38/38 [==============================] - ETA: 0s - loss: 0.5939 - accuracy: 0.8010
Epoch 15: val_loss did not improve from 0.76949
38/38 [==============================] - 52s 1s/step - loss: 0.5939 - accuracy: 0.8010 -
val_loss: 0.8652 - val_accuracy: 0.7235
Epoch 16/20
38/38 [==============================] - ETA: 0s - loss: 0.5434 - accuracy: 0.8153
Epoch 16: val_loss improved from 0.76949 to 0.60529, saving model to best_model.h5
38/38 [==============================] - 52s 1s/step - loss: 0.5434 - accuracy: 0.8153 -
val_loss: 0.6053 - val_accuracy: 0.8191
Epoch 17/20
38/38 [==============================] - ETA: 0s - loss: 0.5927 - accuracy: 0.7985
Epoch 17: val_loss did not improve from 0.60529
38/38 [==============================] - 51s 1s/step - loss: 0.5927 - accuracy: 0.7985 -
val_loss: 0.8011 - val_accuracy: 0.7338
Epoch 18/20
38/38 [==============================] - ETA: 0s - loss: 0.5248 - accuracy: 0.8331
Epoch 18: val_loss did not improve from 0.60529
38/38 [==============================] - 51s 1s/step - loss: 0.5248 - accuracy: 0.8331 -
val_loss: 0.6884 - val_accuracy: 0.7577
Epoch 19/20
38/38 [==============================] - ETA: 0s - loss: 0.4849 - accuracy: 0.8440
Epoch 19: val_loss did not improve from 0.60529
38/38 [==============================] - 51s 1s/step - loss: 0.4849 - accuracy: 0.8440 -
val_loss: 0.7391 - val_accuracy: 0.7270
```

```
Epoch 20/20
38/38 [==============================] - ETA: 0s - loss: 0.5362 - accuracy: 0.8229
Epoch 20: val_loss did not improve from 0.60529
38/38 [==============================] - 52s 1s/step - loss: 0.5362 - accuracy: 0.8229 -
val_loss: 0.7198 - val_accuracy: 0.7782
Found 1479 images belonging to 12 classes.
47/47 [==============================] - 52s 1s/step - loss: 0.5397 - accuracy: 0.8127
Testing Accuracy: 0.8127112984657288
```

In [ ]: