Eye Disease Classifier

April 3, 2025

[1]:
```
# Import necessary libraries
import os
import numpy as np
import pandas as pd
import tensorflow as tf
import keras as keras
import seaborn as sns
import matplotlib.pyplot as plt

# Import other libraries
from keras import layers, models
from transformers import TFViTModel
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import image_dataset_from_directory
```

[2]:
```
# Physical device configuration
physical_devices = tf.config.list_physical_devices('GPU')
for gpu in physical_devices:
    tf.config.experimental.set_memory_growth(gpu, True)

print("Num GPUs Available: ", len(physical_devices))  # 1 or 0 (1 if GPU is␣
↪available)
```

```
Num GPUs Available:  1
```

[3]:
```
# Constants Variables
EPOCHS = 100

DATA_SET = 'dataset'
IMAGE_SIZE = 224
BATCH_SIZE = 16
CHANNELS = 3
SEED = 123
```

[4]:
```
# Load the dataset
dataset = image_dataset_from_directory(
    DATA_SET,
    image_size=(IMAGE_SIZE, IMAGE_SIZE),
```

```
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=SEED,
)
```

Found 16242 files belonging to 10 classes.

[5]:
```
# Print the class names
class_names = dataset.class_names

# Get the class names
print("Dataset Classes:")
for i, class_name in enumerate(class_names):
    print(f"{i + 1}. {class_name.replace('_', ' ')}")
```

```
Dataset Classes:
1. Central Serous Chorioretinopathy
2. Diabetic Retinopathy
3. Disc Edema
4. Glaucoma
5. Healthy
6. Macular Scar
7. Myopia
8. Pterygium
9. Retinal Detachment
10. Retinitis Pigmentosa
```

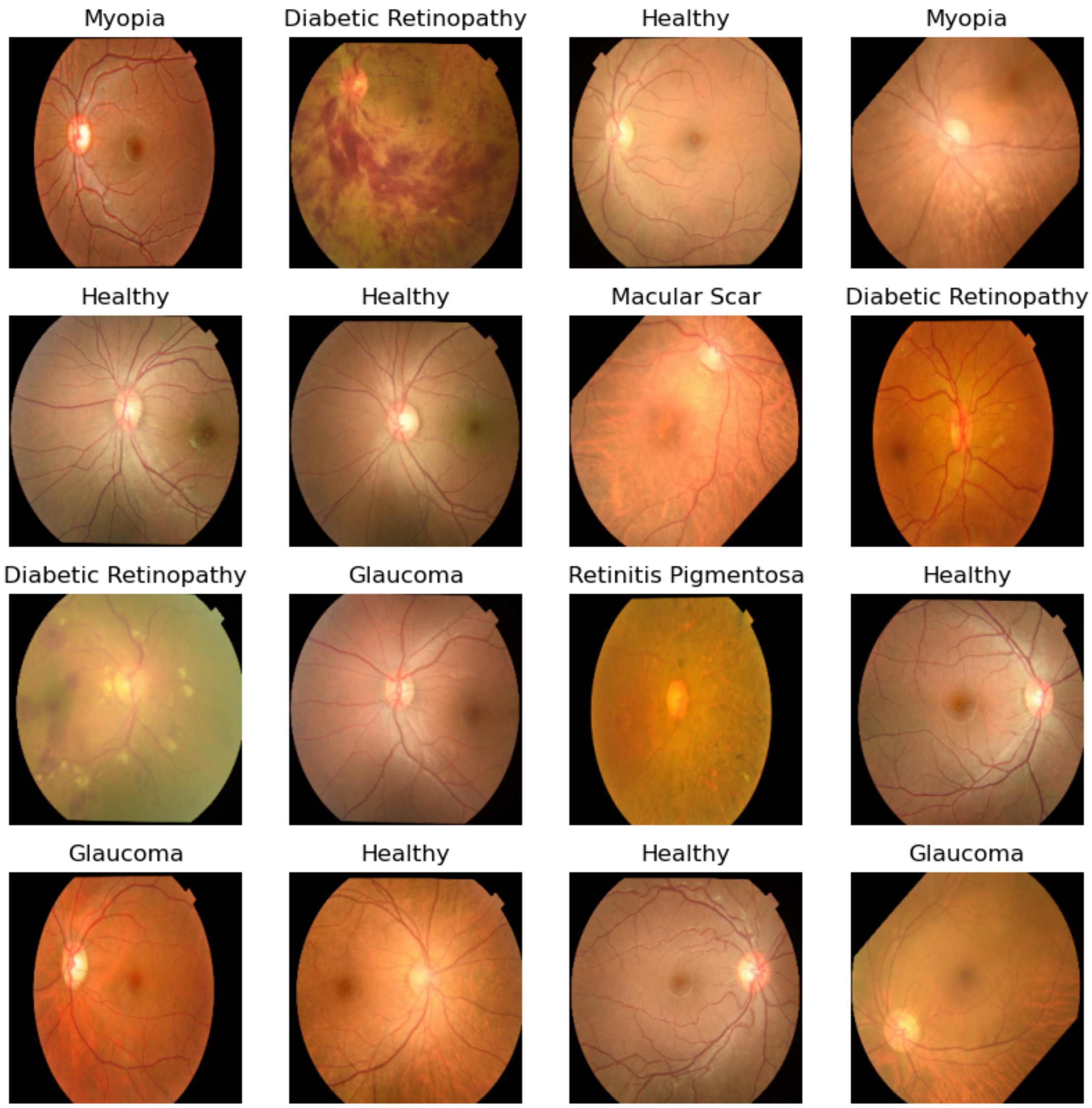
[6]:
```
# Calculate dataset size
dataset_size = tf.data.experimental.cardinality(dataset).numpy()
total_samples = dataset_size * 16  # assuming batch_size = 16

print(f"Total samples: {total_samples}")
```

Total samples: 16256

[7]:
```
# Length of the dataset
dataset_length = len(dataset)
print(f"Dataset Length: {dataset_length}")
```

Dataset Length: 1016

[8]:
```
# Visualize the dataset
plt.figure(figsize=(10, 10))
for iamge_batch, labels_batch in dataset.take(1):
    for i in range(16):
        ax = plt.subplot(4, 4, i + 1)
        plt.imshow(iamge_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



[9]:
```
# Split ratios
train_ratio, val_ratio = 0.7, 0.15

# Calculate sizes
train_size = int(train_ratio * dataset_size)
val_size = int(val_ratio * dataset_size)
test_size = dataset_size - train_size - val_size

# Shuffle with full buffer size
dataset = dataset.shuffle(buffer_size=dataset_size, seed=123)
```

[10]:
```
# Split the dataset
train_dataset = dataset.take(train_size)
val_test_dataset = dataset.skip(train_size)

val_dataset = val_test_dataset.take(val_size)
test_dataset = val_test_dataset.skip(val_size)
```

[11]:
```
# Add performance optimization
AUTOTUNE = tf.data.AUTOTUNE
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
val_dataset = val_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

[12]:
```
# Print dataset sizes
print("Train batches:", train_size)
print("Validation batches:", val_size)
print("Test batches:", test_size)
```

```
Train batches: 711
Validation batches: 152
Test batches: 153
```

[ ]:
```
# Load the pre-trained ViT model
base_model = TFViTModel.from_pretrained("google/vit-large-patch16-224")

# Define the Class for the model
classes = 10


# Define the ViT layer
def vit_layer(x):
    return base_model({"pixel_values": x}).last_hidden_state[:, 0, :]


# Define the model
resize_rescale = keras.Sequential([
    keras.layers.Resizing(224, 224),
    keras.layers.Rescaling(1. / 255),
    keras.layers.Permute((3, 1, 2))
])

inputs = keras.layers.Input(shape=(224, 224, 3))
x = resize_rescale(inputs)
x = keras.layers.Lambda(vit_layer)(x)
outputs = keras.layers.Dense(classes, activation="softmax")(x)

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

[14]:
```
# Summary (optional)
model.summary()
```

```
Model: "model"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0

_________________________________________________________________
sequential (Sequential)      (None, 3, 224, 224)       0

_________________________________________________________________
lambda (Lambda)              (None, 1024)              0

_________________________________________________________________
dense (Dense)                (None, 10)                10250
=================================================================
Total params: 10,250
Trainable params: 10,250
Non-trainable params: 0

_________________________________________________________________
```

[15]:
```
# Early stopping and learning rate reduction callbacks
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

# Learning rate reduction callback
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=3,
    verbose=1,
    min_lr=1e-6
)
```

[16]:
```
base_model.trainable = False  # Freeze ViT base model for 1st phase of training
```

[17]:
```
# compile the model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# then fit
history = model.fit(
    train_dataset,
```

```
    validation_data=val_dataset,
    epochs=EPOCHS,
    callbacks=[early_stop, reduce_lr]
)
```

```
Epoch 1/100
711/711 [==============================] - 215s 251ms/step - loss: 1.0500 -
accuracy: 0.6296 - val_loss: 0.8089 - val_accuracy: 0.7056
Epoch 2/100
711/711 [==============================] - 192s 245ms/step - loss: 0.7449 -
accuracy: 0.7293 - val_loss: 0.6576 - val_accuracy: 0.7558
Epoch 3/100
711/711 [==============================] - 192s 244ms/step - loss: 0.6794 -
accuracy: 0.7493 - val_loss: 0.5913 - val_accuracy: 0.7961
Epoch 4/100
711/711 [==============================] - 191s 243ms/step - loss: 0.6261 -
accuracy: 0.7706 - val_loss: 0.5881 - val_accuracy: 0.7907
Epoch 5/100
711/711 [==============================] - 191s 243ms/step - loss: 0.5914 -
accuracy: 0.7802 - val_loss: 0.5394 - val_accuracy: 0.7915
Epoch 6/100
711/711 [==============================] - 191s 243ms/step - loss: 0.5731 -
accuracy: 0.7833 - val_loss: 0.5405 - val_accuracy: 0.8014
Epoch 7/100
711/711 [==============================] - 191s 243ms/step - loss: 0.5522 -
accuracy: 0.7941 - val_loss: 0.5005 - val_accuracy: 0.8122
Epoch 8/100
711/711 [==============================] - 191s 244ms/step - loss: 0.5365 -
accuracy: 0.7987 - val_loss: 0.5122 - val_accuracy: 0.8121
Epoch 9/100
711/711 [==============================] - 191s 244ms/step - loss: 0.5192 -
accuracy: 0.8034 - val_loss: 0.5010 - val_accuracy: 0.7998
Epoch 10/100
711/711 [==============================] - 191s 243ms/step - loss: 0.5086 -
accuracy: 0.8102 - val_loss: 0.4910 - val_accuracy: 0.8220
Epoch 11/100
711/711 [==============================] - 191s 243ms/step - loss: 0.4905 -
accuracy: 0.8117 - val_loss: 0.4708 - val_accuracy: 0.8220
Epoch 12/100
711/711 [==============================] - 191s 243ms/step - loss: 0.4877 -
accuracy: 0.8143 - val_loss: 0.4312 - val_accuracy: 0.8420
Epoch 13/100
711/711 [==============================] - 191s 244ms/step - loss: 0.4685 -
accuracy: 0.8220 - val_loss: 0.4682 - val_accuracy: 0.8178
Epoch 14/100
711/711 [==============================] - 191s 244ms/step - loss: 0.4648 -
accuracy: 0.8228 - val_loss: 0.4264 - val_accuracy: 0.8376
Epoch 15/100
```

```
711/711 [==============================] - 191s 244ms/step - loss: 0.4611 -
accuracy: 0.8267 - val_loss: 0.4274 - val_accuracy: 0.8450
Epoch 16/100
711/711 [==============================] - 191s 244ms/step - loss: 0.4625 -
accuracy: 0.8226 - val_loss: 0.4333 - val_accuracy: 0.8376
Epoch 17/100
711/711 [==============================] - 191s 244ms/step - loss: 0.4508 -
accuracy: 0.8292 - val_loss: 0.4539 - val_accuracy: 0.8187

Epoch 00017: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 18/100
711/711 [==============================] - 191s 243ms/step - loss: 0.4152 -
accuracy: 0.8380 - val_loss: 0.3815 - val_accuracy: 0.8516
Epoch 19/100
711/711 [==============================] - 191s 243ms/step - loss: 0.4127 -
accuracy: 0.8456 - val_loss: 0.4065 - val_accuracy: 0.8404
Epoch 20/100
711/711 [==============================] - 191s 243ms/step - loss: 0.4074 -
accuracy: 0.8435 - val_loss: 0.3911 - val_accuracy: 0.8540
Epoch 21/100
711/711 [==============================] - 191s 244ms/step - loss: 0.4013 -
accuracy: 0.8485 - val_loss: 0.4163 - val_accuracy: 0.8421

Epoch 00021: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 22/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3927 -
accuracy: 0.8508 - val_loss: 0.3803 - val_accuracy: 0.8540
Epoch 23/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3875 -
accuracy: 0.8534 - val_loss: 0.3627 - val_accuracy: 0.8676
Epoch 24/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3817 -
accuracy: 0.8566 - val_loss: 0.3657 - val_accuracy: 0.8709
Epoch 25/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3855 -
accuracy: 0.8535 - val_loss: 0.3781 - val_accuracy: 0.8479
Epoch 26/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3838 -
accuracy: 0.8519 - val_loss: 0.3971 - val_accuracy: 0.8405

Epoch 00026: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 27/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3730 -
accuracy: 0.8605 - val_loss: 0.3648 - val_accuracy: 0.8660
Epoch 28/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3704 -
accuracy: 0.8624 - val_loss: 0.3753 - val_accuracy: 0.8573
Epoch 29/100
```

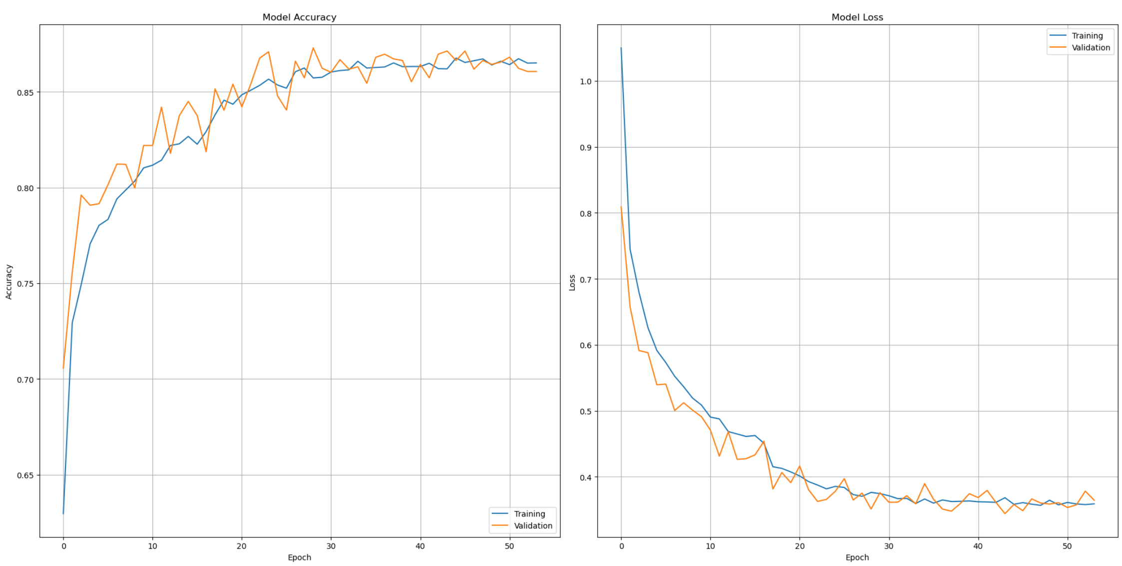
```
711/711 [==============================] - 191s 244ms/step - loss: 0.3763 -
accuracy: 0.8572 - val_loss: 0.3511 - val_accuracy: 0.8729
Epoch 30/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3745 -
accuracy: 0.8576 - val_loss: 0.3759 - val_accuracy: 0.8623
Epoch 31/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3712 -
accuracy: 0.8603 - val_loss: 0.3614 - val_accuracy: 0.8602
Epoch 32/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3668 -
accuracy: 0.8611 - val_loss: 0.3615 - val_accuracy: 0.8668

Epoch 00032: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 33/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3673 -
accuracy: 0.8615 - val_loss: 0.3713 - val_accuracy: 0.8618
Epoch 34/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3595 -
accuracy: 0.8660 - val_loss: 0.3590 - val_accuracy: 0.8631
Epoch 35/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3663 -
accuracy: 0.8624 - val_loss: 0.3896 - val_accuracy: 0.8544

Epoch 00035: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 36/100
711/711 [==============================] - 191s 243ms/step - loss: 0.3600 -
accuracy: 0.8627 - val_loss: 0.3658 - val_accuracy: 0.8680
Epoch 37/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3649 -
accuracy: 0.8630 - val_loss: 0.3511 - val_accuracy: 0.8697
Epoch 38/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3625 -
accuracy: 0.8651 - val_loss: 0.3479 - val_accuracy: 0.8672
Epoch 39/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3628 -
accuracy: 0.8631 - val_loss: 0.3594 - val_accuracy: 0.8664
Epoch 40/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3634 -
accuracy: 0.8632 - val_loss: 0.3743 - val_accuracy: 0.8553
Epoch 41/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3621 -
accuracy: 0.8632 - val_loss: 0.3685 - val_accuracy: 0.8643

Epoch 00041: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 42/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3618 -
accuracy: 0.8649 - val_loss: 0.3794 - val_accuracy: 0.8573
Epoch 43/100
```

```
711/711 [==============================] - 191s 244ms/step - loss: 0.3612 -
accuracy: 0.8621 - val_loss: 0.3609 - val_accuracy: 0.8697
Epoch 44/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3683 -
accuracy: 0.8620 - val_loss: 0.3442 - val_accuracy: 0.8713
Epoch 45/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3583 -
accuracy: 0.8676 - val_loss: 0.3579 - val_accuracy: 0.8664
Epoch 46/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3608 -
accuracy: 0.8653 - val_loss: 0.3486 - val_accuracy: 0.8713
Epoch 47/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3585 -
accuracy: 0.8662 - val_loss: 0.3665 - val_accuracy: 0.8618

Epoch 00047: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
Epoch 48/100
711/711 [==============================] - 192s 244ms/step - loss: 0.3567 -
accuracy: 0.8672 - val_loss: 0.3602 - val_accuracy: 0.8664
Epoch 49/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3645 -
accuracy: 0.8640 - val_loss: 0.3587 - val_accuracy: 0.8643
Epoch 50/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3574 -
accuracy: 0.8660 - val_loss: 0.3608 - val_accuracy: 0.8655

Epoch 00050: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
Epoch 51/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3611 -
accuracy: 0.8642 - val_loss: 0.3534 - val_accuracy: 0.8681
Epoch 52/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3589 -
accuracy: 0.8673 - val_loss: 0.3574 - val_accuracy: 0.8623
Epoch 53/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3578 -
accuracy: 0.8650 - val_loss: 0.3782 - val_accuracy: 0.8606

Epoch 00053: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
Epoch 54/100
711/711 [==============================] - 191s 244ms/step - loss: 0.3590 -
accuracy: 0.8651 - val_loss: 0.3645 - val_accuracy: 0.8606
```

[18]:
```
# Save the model
model.save("model/eyes_diseases.h5")
model.save("model/eyes_diseases.keras")
```

[19]:

```python
# Create figure and axis objects with a single subplot
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

# Plot training & validation accuracy values
ax1.plot(history.history['accuracy'], label='Training')
ax1.plot(history.history['val_accuracy'], label='Validation')
ax1.set_title('Model Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')
ax1.legend(loc='lower right')
ax1.grid(True)

# Plot training & validation loss values
ax2.plot(history.history['loss'], label='Training')
ax2.plot(history.history['val_loss'], label='Validation')
ax2.set_title('Model Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend(loc='upper right')
ax2.grid(True)

# Adjust the layout and display the plot
plt.tight_layout()
plt.show()
```



[24]:

```python
from sklearn.metrics import classification_report

def evaluate_and_report_model(model, test_dataset, class_names):
```

```
    loss, acc = model.evaluate(test_dataset)
    print(f"Test Loss: {loss:.4f}")
    print(f"Test Accuracy: {acc:.4f}")

    y_true = []
    y_pred = []

    for images, labels in test_dataset:
        preds = model.predict(images)
        y_true.extend(labels.numpy())
        y_pred.extend(np.argmax(preds, axis=1))

    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    print("\n=== Classification Report ===")
    print(classification_report(y_true, y_pred, target_names=class_names))

evaluate_and_report_model(model, test_dataset, class_names)
```

```
153/153 [==============================] - 46s 185ms/step - loss: 0.3512 -
accuracy: 0.8668
Test Loss: 0.3512
Test Accuracy: 0.8668

=== Classification Report ===
                                  precision    recall  f1-score   support

Central Serous Chorioretinopathy       0.83      0.81      0.82       111
            Diabetic Retinopathy       0.96      0.96      0.96       536
                      Disc Edema       0.96      0.97      0.96       119
                        Glaucoma       0.75      0.78      0.76       389
                         Healthy       0.79      0.81      0.80       391
                    Macular Scar       0.83      0.77      0.80       294
                          Myopia       0.87      0.86      0.87       359
                       Pterygium       1.00      1.00      1.00        18
              Retinal Detachment       1.00      0.98      0.99        97
            Retinitis Pigmentosa       0.96      0.97      0.97       134

                        accuracy                           0.86      2448
                       macro avg       0.89      0.89      0.89      2448
                    weighted avg       0.87      0.86      0.86      2448
```

[29]:

```
# Predictions using the model
for image_batch, labels_batch in test_dataset.take(1):  # Take one batch from␣
 ↪test dataset
```

```
    ps = model.predict(image_batch)  # Get model predictions for the batch
    images = image_batch.numpy().astype("uint8")  # Convert images to numpy array
    labels = labels_batch.numpy()  # Convert labels to numpy array

    plt.figure(figsize=(15, 15))  # Create a figure with specified size
    for i in range(12):  # Loop through first 12 images
        ax = plt.subplot(4, 4, i + 1)  # Create a 4x4 subplot grid
        plt.imshow(images[i])  # Display the image

        # Get prediction confidence
        confidence = ps[i][ps[i].argmax()] * 100

        # Check if prediction is correct
        is_correct = labels[i] == ps[i].argmax()
        color = "green" if is_correct else "red"

        # Create title with actual label, predicted label, and confidence
        title = f"Actual: {class_names[labels[i]]}\nPredicted: {class_names[ps[i].argmax()]}"
        plt.title(title, color=color)  # Show title with color
        plt.axis("off")  # Hide axes
```