

```
!pip install kaggle tensorflow scikit-learn matplotlib seaborn
```

Show hidden output

```
from google.colab import files
files.upload()
```

Show hidden output

```
import zipfile
with zipfile.ZipFile('eye_diseases_classification.zip', 'r') as zip_ref:
    zip_ref.extractall('dataset_original')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
from PIL import Image
import os

input_root = 'dataset_original/dataset'
output_root = 'output'
resize_size = (256, 256)

os.makedirs(output_root, exist_ok=True)

for class_name in os.listdir(input_root):
    class_path = os.path.join(input_root, class_name)
    if not os.path.isdir(class_path) or class_name == 'masks':
        continue
    output_class_path = os.path.join(output_root, class_name)
    os.makedirs(output_class_path, exist_ok=True)

    for image_name in os.listdir(class_path):
        if image_name.lower().endswith(('png', '.jpg', '.jpeg', '.bmp', '.tiff')):
            input_image_path = os.path.join(class_path, image_name)
            output_image_name = os.path.splitext(image_name)[0] + '.jpg'
            output_image_path = os.path.join(output_class_path, output_image_name)
            try:
                with Image.open(input_image_path) as img:
                    img = img.convert('RGB')
                    img = img.resize(resize_size)
                    img.save(output_image_path, 'JPEG')
            except Exception as e:
                print(f"Error processing {input_image_path}: {e}")
```

```
import os, shutil
from sklearn.model_selection import train_test_split

original_dataset_dir = 'output'
base_dir = 'new_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_path in [train_dir, validation_dir, test_dir]:
    os.makedirs(dir_path, exist_ok=True)

train_ratio, val_ratio, test_ratio = 0.7, 0.1, 0.2
```

```
def split_and_copy(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_imgs, temp_imgs = train_test_split(all_images, test_size=(1-train_ratio), random_state=42)
    val_imgs, test_imgs = train_test_split(temp_imgs,
                                           test_size=test_ratio/(val_ratio+test_ratio),
                                           random_state=42)
    for imgs, folder in zip([train_imgs, val_imgs, test_imgs], ['train', 'validation', 'test']):
        dest_folder = os.path.join(dest_base_dir, folder, os.path.basename(class_dir))
        os.makedirs(dest_folder, exist_ok=True)
        for file in imgs:
            shutil.copy(file, dest_folder)

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(full_class_dir, base_dir)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
batch_size = 32
```

```
img_size = (256, 256)
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    horizontal_flip=True,
    vertical_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1,
    brightness_range=[0.8, 1.2],
    zoom_range=0.2
)
```

```
val_test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(
    'new_dataset/train',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)
```

```
validation_generator = val_test_datagen.flow_from_directory(
    'new_dataset/validation',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
test_generator = val_test_datagen.flow_from_directory(
    'new_dataset/test',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 2949 images belonging to 4 classes.
Found 422 images belonging to 4 classes.
Found 846 images belonging to 4 classes.
```

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

def build_simple_cnn(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(256, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

input_shape = (256,256,3)
num_classes = 4
model = build_simple_cnn(input_shape, num_classes)
model.summary()

```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do
 super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 256)	29,491,456
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

Total params: 29,585,732 (112.86 MB)

Trainable params: 29,585,732 (112.86 MB)

Non-trainable params: 0 (0.00 B)

```

history = model.fit(
    train_generator,
    epochs=2,
    validation_data=validation_generator
)

```

Epoch 1/2

93/93 ————— 547s 6s/step - accuracy: 0.4100 - loss: 1.2804 - val_accuracy: 0.4882 - val_loss: 1.1450

Epoch 2/2

93/93 ————— 546s 6s/step - accuracy: 0.4666 - loss: 1.1450 - val_accuracy: 0.5166 - val_loss: 1.1450

```

from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.models import Model

base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(256,256,3))
base_model.trainable = False # تجميد الطبقات الأولى

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_data_format.h5 87910968/87910968 ————— 0s 0us/step

```

history = model.fit(
    train_generator,
    epochs=2,
    validation_data=validation_generator
)

```

Epoch 1/2
93/93 ————— **661s** 7s/step - accuracy: 0.6080 - loss: 0.9252 - val_accuracy: 0.7062 - val_loss: 0.7062
Epoch 2/2
93/93 ————— **644s** 7s/step - accuracy: 0.6606 - loss: 0.7886 - val_accuracy: 0.7085 - val_loss: 0.7085

```

import numpy as np
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

test_generator.reset()
predictions = model.predict(test_generator, steps=len(test_generator))
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys())

accuracy = accuracy_score(true_classes, predicted_classes)
print(f"Test Accuracy: {accuracy:.4f}")

cm = confusion_matrix(true_classes, predicted_classes)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

print(classification_report(true_classes, predicted_classes, target_names=class_labels))

```

27/27 164s 6s/step
Test Accuracy: 0.6986

