Milestone 3
Prathmesh Kadam

Model Training Process Output: -

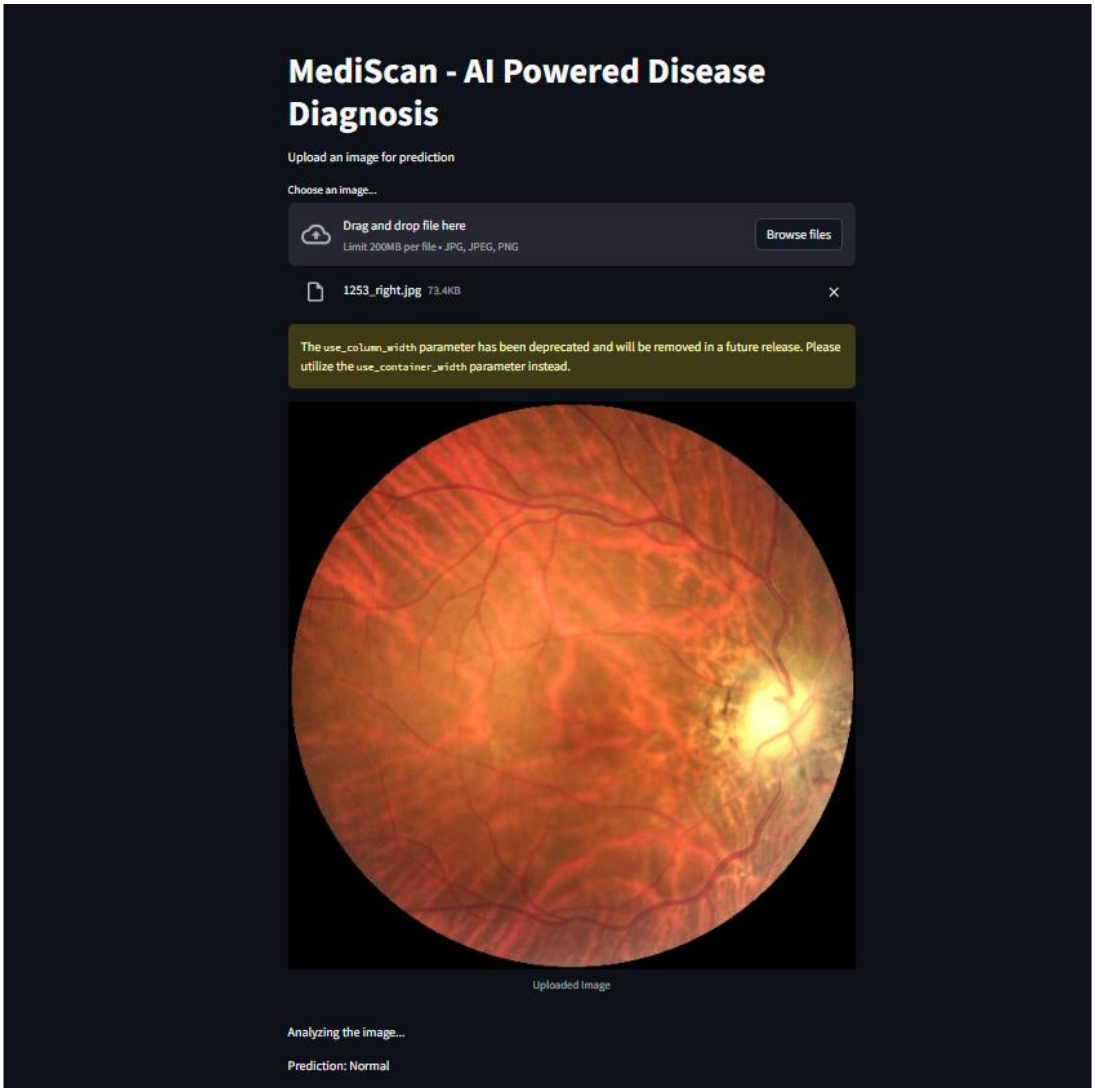
```
E:\Infosys\MediScan-AI-Powered-Medical-Image-Analysis-for-Disease-Diagnosis_0ct_2024\src\main.py

Found 3376 images belonging to 4 classes.

Found 841 images belonging to 4 classes.

Epoch 1/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 0s 6s/step - accuracy: 0.4476 - loss: 1.2019C:\Users\lenovo\PycharmProjects\pythonProject6\.venv\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.

106/106 ━━━━━━━━━━━━━━━━━━━━ 748s 7s/step - accuracy: 0.4484 - loss: 1.2006 - val_accuracy: 0.5755 - val_loss: 1.0069 - learning_rate: 1.0000e-04

Epoch 2/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 783s 7s/step - accuracy: 0.6494 - loss: 0.8103 - val_accuracy: 0.5791 - val_loss: 1.0357 - learning_rate: 1.0000e-04

Epoch 3/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 671s 6s/step - accuracy: 0.6843 - loss: 0.7522 - val_accuracy: 0.6254 - val_loss: 0.9409 - learning_rate: 1.0000e-04

Epoch 4/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 632s 6s/step - accuracy: 0.6897 - loss: 0.7220 - val_accuracy: 0.6492 - val_loss: 0.9009 - learning_rate: 1.0000e-04

Epoch 5/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 602s 6s/step - accuracy: 0.7115 - loss: 0.6912 - val_accuracy: 0.6207 - val_loss: 0.8938 - learning_rate: 1.0000e-04

Epoch 6/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 575s 5s/step - accuracy: 0.7303 - loss: 0.6535 - val_accuracy: 0.6112 - val_loss: 0.9798 - learning_rate: 1.0000e-04

Epoch 7/20

106/106 ━━━━━━━━━━━━━━━━━━━━  593s 6s/step - accuracy: 0.7503 - loss: 0.5987 - val_accuracy: 0.6373 - val_loss: 0.9518 - learning_rate: 1.0000e-04

Epoch 8/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 0s 8s/step - accuracy: 0.7383 - loss: 0.6261

Epoch 8: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.

106/106 ━━━━━━━━━━━━━━━━━━━━  997s 9s/step - accuracy: 0.7384 - loss: 0.6260 - val_accuracy: 0.6171 - val_loss: 1.0028 - learning_rate: 1.0000e-04

Epoch 9/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 598s 6s/step - accuracy: 0.7620 - loss: 0.5824 - val_accuracy: 0.6385 - val_loss: 0.9468 - learning_rate: 5.0000e-05

Epoch 10/20

106/106 ━━━━━━━━━━━━━━━━━━━━ 602s 6s/step - accuracy: 0.7916 - loss: 0.5275 - val_accuracy: 0.6243 - val_loss: 0.9450 - learning_rate: 5.0000e-05

Epoch 10: early stopping

Restoring model weights from the end of the best epoch: 5.

Found 3376 images belonging to 4 classes.

Found 841 images belonging to 4 classes.

27/27 ━━━━━━━━━━━━━━━━━━━━ 106s 4s/step - accuracy: 0.6074 - loss: 0.9268

Validation Loss: 0.9031521081924438, Validation Accuracy: 0.6195005774497986


Process finished with exit code 0
```

Testing the trained model with the use of streamlit: -

Code: -

```
import streamlit as st
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image
import io

# Load the pre-trained model (replace with your model if needed)
model = load_model('trained_model.h5')  # Adjust the model path as needed

# Class labels
class_labels = ['Cataract', ’Diabetic’ , 'Glaucoma', 'Normal']  # Replace with your actual class labels


# Function to preprocess and predict the image
def predict_image(uploaded_file):
    # Open image file
    uploaded_image = Image.open(uploaded_file)

    # Resize image to (299, 299) as required by InceptionV3
    uploaded_image = uploaded_image.resize((299, 299))

    # Convert image to array
    image_array = image.img_to_array(uploaded_image)

    # Add batch dimension
    image_array = np.expand_dims(image_array, axis=0)

    # Preprocess the image for InceptionV3
    image_array = preprocess_input(image_array)

    # Predict using the model
    predictions = model.predict(image_array)

    # Get the predicted class and confidence
    predicted_class = np.argmax(predictions)  # Index of the max probability
    confidence = np.max(predictions)  # Maximum confidence

    return predicted_class, confidence


# Streamlit UI
st.title("MediScan - AI Powered Disease Diagnosis")
st.write("Upload an image for prediction")

# Image uploader
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

# When the user uploads an image
if uploaded_file is not None:
    # Display the uploaded image
    st.image(uploaded_file, caption="Uploaded Image", use_column_width=True)
    st.write("")

    # Show a message while the image is being analyzed
    st.write("Analyzing the image...")

    # Call the prediction function
    predicted_class, confidence = predict_image(uploaded_file)

    # Show prediction result
    st.write(f"Prediction: {class_labels[predicted_class]}")
    st.write(f"Confidence: {confidence:.2f}")
```

Output: -



MediScan - AI Powered Disease Diagnosis

Upload an image for prediction

Choose an image...

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Uploaded Image

Analyzing the image...

Prediction: Normal