SAMPLE CODE

Trained_Model.py

```
import os
import cv2
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# Suppress TensorFlow logs and warnings
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"  # Suppress TensorFlow logs
os.environ["TF_ENABLE_ONEDNN_OPTS"] = "0"  # Disable oneDNN warnings

def load_dataset(folder_path, image_size=(128, 128)):
    """
    Load images from folders where each folder represents a blood group.

    Args:
        folder_path (str): Path to the root dataset folder.
        image_size (tuple): Size to resize the images (width, height).

    Returns:
        images (numpy.ndarray): Array of preprocessed images.
        labels (numpy.ndarray): Corresponding labels for the images.
        label_map (dict): Mapping of blood group names to numeric labels.
    """
    images = []
    labels = []
    label_map = {}
    current_label = 0

    for folder_name in sorted(os.listdir(folder_path)):
        folder_full_path = os.path.join(folder_path, folder_name)
        if not os.path.isdir(folder_full_path):
```

```
            continue
        label_map[folder_name] = current_label
        for file_name in os.listdir(folder_full_path):
            file_path = os.path.join(folder_full_path, file_name)
            img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
            if img is None:
                print(f"Warning: Skipping invalid image {file_path}")
                continue
            img = cv2.resize(img, image_size)
            images.append(img)
            labels.append(current_label)
        current_label += 1

    if not images:
        raise ValueError("No images found in the dataset. Please check the folder structure.")

    images = np.array(images, dtype=np.float32) / 255.0
    labels = np.array(labels, dtype=np.int32)
    return images, labels, label_map

def build_cnn_model(input_shape, num_classes):
    """
    Build and compile a CNN model.

    Args:
        input_shape (tuple): Shape of the input images (height, width, channels).
        num_classes (int): Number of output classes.

    Returns:
        model: Compiled CNN model.
    """
    model = Sequential([
        Input(shape=input_shape),
        Conv2D(32, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
```

```
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

if __name__ == "__main__":
    # Path to your dataset folder
    dataset_path = "dataset"  # Ensure this folder exists in the same directory as this script
    image_size = (128, 128)

    # Load the dataset
    print("Loading dataset...")
    images, labels, label_map = load_dataset(dataset_path, image_size)
    print(f"Loaded {len(images)} images.")
    print(f"Label Map: {label_map}")

    # Preprocess data
    images = images.reshape(-1, image_size[0], image_size[1], 1)  # Reshape for grayscale
    labels = to_categorical(labels, num_classes=len(label_map))

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

    # Build and train the model
    print("Building and training the model...")
    model = build_cnn_model(input_shape=(image_size[0], image_size[1], 1), num_classes=len(label_map))
    history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

    # Save the trained model
    model.save("blood_group_model.h5")
    print("Model saved as blood_group_model.h5")
```

App.py

```
from flask import Flask, request, render_template
import tensorflow as tf
import cv2
import numpy as np
import os

app = Flask(__name__)
model = tf.keras.models.load_model("blood_group_model.h5")  # Ensure this file is in the same folder
label_map = {0: "A+", 1: "A-", 2: "AB+", 3: "AB-", 4: "B+", 5: "B-", 6: "O+", 7: "O-"}  # Adjust based on your dataset



UPLOAD_FOLDER = "static/uploads"
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    if "file" not in request.files:
        return "No file part", 400

    file = request.files["file"]
    if file.filename == "":
        return "No selected file", 400

    # Save the uploaded file
    file_path = os.path.join(app.config["UPLOAD_FOLDER"], file.filename)
    file.save(file_path)

    # Preprocess the image
```

```
    img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (128, 128)) / 255.0  # Normalize pixel values
    img = img.reshape(1, 128, 128, 1)

    # Predict blood group
    predictions = model.predict(img)
    predicted_label = label_map[np.argmax(predictions)]

    return render_template("index.html", prediction=predicted_label, image_path=file_path)

if _name___== "_main_":
    app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blood Group Detection</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css">
    <style>
        body {
            background: linear-gradient(to bottom right, #003366, #ff6600); /* Dark blue to orange gradient */
            color: #fff;
            font-family: 'Arial', sans-serif;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            margin: 0;
        }
        .container {
            width: 90%;
            max-width: 600px;
            padding: 20px;
            background-color: rgba(255, 255, 255, 0.15); /* Transparent white effect */
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
            text-align: center;
        }
        h1 {
            color: #ffcc00; /* Bright yellow-orange for the title */
        }
        p {
            color: #ffffff; /* Keep text readable */
        }
```

```
        .btn-primary {
            background-color: #ff6600; /* Vibrant orange button */
            border-color: #ff6600;
        }
        .btn-primary:hover {
            background-color: #cc5200; /* Darker orange on hover */
            border-color: #cc5200;
        }
        .form-label {
            color: #ffcc00; /* Bright yellow-orange for labels */
        }
        .result-box {
            background-color: rgba(0, 51, 102, 0.8); /* Semi-transparent dark blue */
            padding: 20px;
            border-radius: 10px;
            color: #ffcc00; /* Yellow text in result box */
            margin-top: 20px;
        }
        .result-text {
            font-size: 1.5rem;
            font-weight: bold;
            color: #ffcc00; /* Yellow for the result text */
        }
        .thumbnail {
            margin-top: 20px;
            border: 2px solid #ffcc00; /* Border matches yellow-orange theme */
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Blood Group Detection</h1>
        <p>Upload a fingerprint image to predict the blood group.</p>
        <form action="/predict" method="POST" enctype="multipart/form-data" class="mt-4">
            <div class="mb-3">
                <label for="file" class="form-label">Choose Fingerprint Image:</label>
```

```
            <input type="file" class="form-control" id="file" name="file" required>
        </div>
        <button type="submit" class="btn btn-primary">Predict</button>
    </form>
    {% if prediction %}
    <div class="result-box">
        <h2>Prediction Result</h2>
        <p class="result-text">Blood Group: {{ prediction }}</p>
        <img src="{{ image_path }}" alt="Uploaded Image" class="thumbnail"
        width="300">
    </div>
    {% endif %}
</div>
</body>
</html>
```