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

```
Mounted at /content/drive
```

```
!unzip /content/drive/MyDrive/skinsample_augmenteds.zip
```

```
unzip:  cannot find or open /content/drive/MyDrive, /content/drive/MyDrive.zip or /content/drive/MyDrive.ZIP.
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.applications import MobileNetV2
from keras.models import Sequential
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.optimizers import Adam
#from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.applications import ResNet50



# Mount Google Drive if your dataset is stored there


# Define path to your dataset
data_path = '/content/sample11'

# Define the categories (normal and skin disease)
categories = ['normal', 'skindiseased','NotSkinImages']


# Resize images to match MobileNetV2 input size
img_size = 112

# Load images and labels
data = []

for category in categories:
    path = os.path.join(data_path, category)
    label = categories.index(category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img))
        img_array = cv2.resize(img_array, (img_size, img_size))
        data.append([img_array, label])

# Shuffle the data
np.random.shuffle(data)

# Split the data into features and labels
X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)

# Convert features and labels to numpy arrays
X = np.array(X)
y = np.array(y)

# Normalize the data
X = X / 255.0

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

# Convert labels to one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Load pre-trained MobileNetV2 model without the top classification layer
base_model = MobileNetV2(input_shape=(img_size, img_size, 3), include_top=False, weights='imagenet')

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Initialize Sequential model
model = Sequential()

# Add base model
model.add(base_model)

# Add Flatten layer to convert 2D feature map to 1D feature vector
model.add(Flatten())

# Add Dense layer with 128 neurons and ReLU activation
model.add(Dense(128, activation='relu'))

# Add Dropout layer with dropout rate of 0.5
model.add(Dropout(0.5))

# Additional Convolutional blocks
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))

# Dense block with three Dense layers
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
```

```
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

```
# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# Print model summary
print(model.summary())

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print("Test accuracy:", test_acc)

# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()

# Plot training and validation loss
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_c
9406464/9406464 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Func  (None, 4, 4, 1280)        2257984
 tional)

 flatten (Flatten)           (None, 20480)             0

 dense (Dense)               (None, 128)               2621568

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 256)               16640

 dropout_2 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_3 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 64)                8256

 dropout_4 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 3)                 195

=================================================================
Total params: 4945795 (18.87 MB)
Trainable params: 2687811 (10.25 MB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
None
Epoch 1/20
70/70 [==============================] - 35s 430ms/step - loss: 0.7423 - accuracy: 0.8822 - val_loss: 0.2851 - val_accuracy:
Epoch 2/20
70/70 [==============================] - 30s 437ms/step - loss: 0.2555 - accuracy: 0.9319 - val_loss: 0.1528 - val_accuracy:
Epoch 3/20
70/70 [==============================] - 34s 492ms/step - loss: 0.2103 - accuracy: 0.9418 - val_loss: 0.1480 - val_accuracy:
Epoch 4/20
70/70 [==============================] - 34s 481ms/step - loss: 0.1805 - accuracy: 0.9504 - val_loss: 0.1204 - val_accuracy:
Epoch 5/20
70/70 [==============================] - 30s 431ms/step - loss: 0.1370 - accuracy: 0.9594 - val_loss: 0.1019 - val_accuracy:
Epoch 6/20
70/70 [==============================] - 35s 502ms/step - loss: 0.1566 - accuracy: 0.9553 - val_loss: 0.1281 - val_accuracy:
Epoch 7/20
70/70 [==============================] - 33s 471ms/step - loss: 0.1190 - accuracy: 0.9558 - val_loss: 0.1035 - val_accuracy:
Epoch 8/20
70/70 [==============================] - 29s 414ms/step - loss: 0.1192 - accuracy: 0.9639 - val_loss: 0.0983 - val_accuracy:
```

```
model.save('skinmod1.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is consi
  saving_api.save_model(
```

```
# /content/skindiseasesall/normal/123123.jpg

import numpy as np
import cv2
from keras.models import load_model

# Load the trained model
model = load_model('/content/skinmod1.h5')

# Define categories
categories = ['normal', 'skin_disease','NotSkinImages']

# Function to preprocess and predict an image
def predict_image(image_path):
    img_size = 112
    img_array = cv2.imread(image_path)

    # Check if img_array is empty (None)
    if img_array is None:
        print(f"Error: Unable to read image from {image_path}")
        return None, None

    img_array = cv2.resize(img_array, (img_size, img_size))
    img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    confidence = prediction[0][predicted_class]
    predicted_category = categories[predicted_class]
    return predicted_category, confidence

# Path to the image you want to predict
image_path = '/content/sample11/normal/123123as.jpg'

# Predict the image
predicted_category, confidence = predict_image(image_path)
#if confidence < 0.5:
# print('not skin')
print("Predicted category:", predicted_category)
print("Confidence:", confidence)
```

```
1/1 [==============================] - 1s 1s/step
Predicted category: normal
Confidence: 0.628455
```

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-4.26.0-py3-none-any.whl (17.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 17.1 MB/s eta 0:00:00
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2)
Collecting fastapi (from gradio)
  Downloading fastapi-0.110.1-py3-none-any.whl (91 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 91.9/91.9 kB 5.5 MB/s eta 0:00:00
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.3.2.tar.gz (5.5 kB)
  Preparing metadata (setup.py) ... done
Collecting gradio-client==0.15.1 (from gradio)
  Downloading gradio_client-0.15.1-py3-none-any.whl (313 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 20.8 MB/s eta 0:00:00
Collecting httpx>=0.24.1 (from gradio)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 5.8 MB/s eta 0:00:00
Requirement already satisfied: huggingface-hub>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.20.3)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.4.0)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.3)
Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.1.5)
Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.25.2)
Collecting orjson~=3.0 (from gradio)
  Downloading orjson-3.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (144 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 144.8/144.8 kB 14.7 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.0)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.0.3)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (9.4.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.6.4)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting python-multipart>=0.0.9 (from gradio)
  Downloading python_multipart-0.0.9-py3-none-any.whl (22 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1)
Collecting ruff>=0.2.2 (from gradio)
  Downloading ruff-0.3.5-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 37.0 MB/s eta 0:00:00
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Collecting tomlkit==0.12.0 (from gradio)
  Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB)
Requirement already satisfied: typer[all]<1.0,>=0.9 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.9.4)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.10.0)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.29.0-py3-none-any.whl (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.8/60.8 kB 6.1 MB/s eta 0:00:00
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.15.1->gradio) (2023.6.0)
Collecting websockets<12.0,>=10.0 (from gradio-client==0.15.1->gradio)
  Downloading websockets-11.0.3-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.9/129.9 kB 14.3 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (4.19.2)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.12.1)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (2024.2.2)
```

```
import gradio as gr
from PIL import Image
import numpy as np
import cv2
from keras.models import load_model

# Load the trained model
model = load_model('/content/skinmod1.h5')

# Define categories
categories = ['normal', 'skin_disease','NotSkinImages']

def predict_image(image_array):
    img_size = 112
    # Resize the image to the required size
    img_array = cv2.resize(image_array, (img_size, img_size))
    img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    confidence = prediction[0][predicted_class]
    predicted_category = categories[predicted_class]
    return predicted_category, confidence

def process_image(image_array):
    # Convert the PIL Image to NumPy array if necessary
    if isinstance(image_array, Image.Image):
        image_array = np.array(image_array)

    # Ensure the image has three color channels (RGB)
    if len(image_array.shape) == 2:
        image_array = cv2.cvtColor(image_array, cv2.COLOR_GRAY2RGB)

    predicted_category, confidence = predict_image(image_array)
    result_text = f"Predicted category: {predicted_category}\nConfidence: {confidence:.2f}"
    #if confidence < 0.5:
    #   result_text += "\nNot confident enough to be considered as skin disease."

    return result_text

# Create the Gradio interface
iface = gr.Interface(fn=process_image,
                     inputs=gr.Image(label="Upload Image"),
                     outputs="text",
                     title="Skin Disease Classifier",
                     description="Upload an image of skin, and get a disease classification.")

# Launch the application
iface.launch(debug=True)
```

```
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=F

Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in la
Running on public URL: https://3b58b6fa2148ed9ea3.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to
```

No interface is running right now

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 40ms/step
```