```
# =================================================
# Skin Cancer Detection: Hybrid CNN (EfficientNetB4) + XGBoost
# Dataset: Skin Cancer MNIST: HAM10000
# =================================================

# Install only the necessary packages
!pip install kagglehub -q
!pip install efficientnet xgboost scikit-learn -q

import kagglehub
import os
import pandas as pd
import numpy as np
import tensorflow as tf
import efficientnet.tfkeras as efn
# The problematic 'tensorflow_addons' import is now removed
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, f1_score
import xgboost as xgb
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img


# ==============================
# 1. DOWNLOAD DATASET
# ==============================
path = kagglehub.dataset_download("kmader/skin-cancer-mnist-ham10000")
print("Path to dataset files:", path)

# Dataset CSV and image paths
csv_path = os.path.join(path, "HAM10000_metadata.csv")
img_dir_1 = os.path.join(path, "ham10000_images_part_1")
img_dir_2 = os.path.join(path, "ham10000_images_part_2")

df = pd.read_csv(csv_path)
print(df.head())

# ==============================
# 2. PREPROCESS
# ==============================
# Combine both image folders into one mapping
img_paths = {}
for folder in [img_dir_1, img_dir_2]:
    for fname in os.listdir(folder):
        img_paths[fname.split(".")[0]] = os.path.join(folder, fname)

# Map image_id to file path
df["path"] = df["image_id"].map(img_paths)

# Map disease type to label index
label_map = {label: idx for idx, label in enumerate(df["dx"].unique())}
df["label"] = df["dx"].map(label_map)

# Train-test split
train_df, test_df = train_test_split(df, test_size=0.2, stratify=df["label"], random_state=42)

print(f"Train size: {len(train_df)}, Test size: {len(test_df)}")

# ==============================
# 3. IMAGE GENERATOR
# ==============================
IMG_SIZE = 380
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_dataframe(
    train_df,
    x_col="path",
    y_col="label",
    target_size=(IMG_SIZE, IMG_SIZE),
    class_mode="raw",
    batch_size=BATCH_SIZE,
    shuffle=False
)

test_gen = test_datagen.flow_from_dataframe(
    test_df,
    x_col="path",
    y_col="label",
    target_size=(IMG_SIZE, IMG_SIZE),
    class_mode="raw",
    batch_size=BATCH_SIZE,
    shuffle=False
)

# ==============================
# 4. FEATURE EXTRACTION (EfficientNetB4)
# ==============================
base_model = efn.EfficientNetB4(weights="imagenet", include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
base_model.trainable = False

feature_extractor = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D()
])

# Extract features
def extract_features(generator):
    features = []
    labels = []
    for imgs, lbls in tqdm(generator, total=len(generator)):
        feat = feature_extractor.predict(imgs, verbose=0)
        features.append(feat)
        labels.append(lbls)
        if len(features) * generator.batch_size >= generator.n:
            break
    return np.vstack(features), np.hstack(labels)

X_train, y_train = extract_features(train_gen)
X_test, y_test = extract_features(test_gen)

print("Feature shapes:", X_train.shape, X_test.shape)

# ==============================
# 5. TRAIN XGBOOST CLASSIFIER
# ==============================
clf = xgb.XGBClassifier(
    objective="multi:softmax",
    num_class=len(label_map),
    eval_metric="mlogloss",
    use_label_encoder=False,
    n_estimators=300,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)

clf.fit(X_train, y_train)

# ==============================
# 6. EVALUATION
# ==============================
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average="macro")
print(f"Test Accuracy: {acc*100:.2f}%")
print(f"Macro F1 Score: {f1:.4f}")

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_map.keys()))

# ==============================
# 7. PREDICT ON SINGLE IMAGE
# ==============================
def predict_single(img_path):
    img = load_img(img_path, target_size=(IMG_SIZE, IMG_SIZE))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    feat = feature_extractor.predict(img_array)
    pred = clf.predict(feat)[0]
    label_name = [k for k,v in label_map.items() if v == pred][0]
    return label_name

sample_img = test_df.iloc[0]["path"]
print("Sample Prediction:", predict_single(sample_img))
```

```
Path to dataset files: /kaggle/input/skin-cancer-mnist-ham10000
     lesion_id      image_id   dx dx_type   age   sex localization
0  HAM_0000118  ISIC_0027419  bkl   histo  80.0  male        scalp
1  HAM_0000118  ISIC_0025030  bkl   histo  80.0  male        scalp
2  HAM_0002730  ISIC_0026769  bkl   histo  80.0  male        scalp
3  HAM_0002730  ISIC_0025661  bkl   histo  80.0  male        scalp
4  HAM_0001466  ISIC_0031633  bkl   histo  75.0  male          ear
Train size: 8012, Test size: 2003
Found 8012 validated image filenames.
Found 2003 validated image filenames.
Downloading data from https://github.com/Callidior/keras-applications/releases/download/efficientnet/efficientnet-b4_weights_tf
71892840/71892840 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
100%|██████████| 250/251 [1:28:39<00:21, 21.28s/it]
 98%|█████████▊| 62/63 [21:01<00:20, 20.34s/it]
Feature shapes: (8012, 1792) (2003, 1792)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [07:25:36] WARNING: /workspace/src/learner.cc:738
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
Test Accuracy: 75.84%
Macro F1 Score: 0.4240

Classification Report:
              precision    recall  f1-score   support

         bkl       0.54      0.51      0.52       220
          nv       0.83      0.95      0.88      1341
          df       0.00      0.00      0.00        23
         mel       0.46      0.39      0.42       223
        vasc       1.00      0.39      0.56        28
         bcc       0.64      0.31      0.42       103
       akiec       0.50      0.09      0.16        65

    accuracy                           0.76      2003
   macro avg       0.57      0.38      0.42      2003
weighted avg       0.73      0.76      0.73      2003

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 510ms/step
Sample Prediction: nv
```