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

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

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.applications import EfficientNetB0
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

```
import os
import cv2
from collections import defaultdict

# Specify the main folder path
main_folder_path = '/content/drive/MyDrive/split_minip/train'  # Change this to your main folder path

# Dictionary to store image sizes and their respective counts
size_counts = defaultdict(int)

# Process each subfolder inside the main folder
for subfolder in os.listdir(main_folder_path):
    subfolder_path = os.path.join(main_folder_path, subfolder)

    if os.path.isdir(subfolder_path):  # Check if it's a folder
        print(f"Processing subfolder: {subfolder}")

        # Process each image in the subfolder
        for image_file in os.listdir(subfolder_path):
            image_path = os.path.join(subfolder_path, image_file)

            if image_file.lower().endswith(('.jpg', '.jpeg', '.png', '.bmp')):  # Add more extensions if needed
                # Read the image
                image = cv2.imread(image_path)

                if image is not None:
                    # Get the shape of the image (height, width, channels)
                    image_size = image.shape[:2]  # Only get height and width (ignore channels)
                    size_counts[image_size] += 1
                else:
                    print(f"Failed to read image: {image_file}")

# Print the total number of images for each size
print("\nSummary of image sizes and counts:")
for size, count in size_counts.items():
    print(f"Size {size[0]}x{size[1]}: {count} images")
```

```
Processing subfolder: A-
Processing subfolder: A+
Processing subfolder: B-
Processing subfolder: B+
Processing subfolder: O-
Processing subfolder: AB-
Processing subfolder: AB+
Processing subfolder: O+

Summary of image sizes and counts:
Size 103x96: 5556 images
Size 298x241: 42 images
Size 96x103: 2 images
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report


IMAGE_SIZE = (103, 96)
BATCH_SIZE = 32
NUM_CLASSES = 8

train_dir = '/content/drive/MyDrive/split_minip/train'
val_dir = '/content/drive/MyDrive/split_minip/val'
test_dir = '/content/drive/MyDrive/split_minip/test'

def create_data_generator(directory):
    data_gen = ImageDataGenerator(rescale=1.0/255)
    return data_gen.flow_from_directory(
        directory,
        target_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        shuffle=False
    )

train_data = create_data_generator(train_dir)
val_data = create_data_generator(val_dir)
test_data = create_data_generator(test_dir)
```

```
Found 5600 images belonging to 8 classes.
Found 1201 images belonging to 8 classes.
Found 1162 images belonging to 8 classes.
```

```
efficientnet_base = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(*IMAGE_SIZE, 3))
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
efficientnet_base.trainable = False


# Add custom layers for feature extraction
x = GlobalAveragePooling2D()(efficientnet_base.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
feature_extractor = Model(inputs=efficientnet_base.input, outputs=x)


# Extract features
def extract_features(data_generator, model):
    features = []
    labels = []
    for images, lbls in data_generator:
        feats = model.predict(images)
        features.append(feats)
        labels.append(lbls)
        if len(features) * BATCH_SIZE >= data_generator.samples:
            break
    return np.vstack(features), np.vstack(labels)

train_features, train_labels = extract_features(train_data, feature_extractor)
val_features, val_labels = extract_features(val_data, feature_extractor)
```

Show hidden output

```
print(f"Train features shape: {train_features.shape}, Train labels shape: {train_labels.shape}")
print(f"Validation features shape: {val_features.shape}, Validation labels shape: {val_labels.shape}")
```

```
Train features shape: (5600, 512), Train labels shape: (5600, 8)
Validation features shape: (1201, 512), Validation labels shape: (1201, 8)
```

```
print(f"Train samples: {train_data.samples}, Train batches: {len(train_data)}")
print(f"Validation samples: {val_data.samples}, Validation batches: {len(val_data)}")
print(f"Test samples: {test_data.samples}, Test batches: {len(test_data)}")
```

```
Train samples: 5600, Train batches: 175
Validation samples: 1201, Validation batches: 38
Test samples: 1162, Test batches: 37
```

```
# Flatten labels for SVM training
train_labels = np.argmax(train_labels, axis=1)
val_labels = np.argmax(val_labels, axis=1)


# Train SVM Classifier
svm = SVC(kernel='linear', probability=True)
svm.fit(train_features, train_labels)
```

```
▾ SVC
SVC(kernel='linear', probability=True)
```

```
# Validate the model
val_preds = svm.predict(val_features)
val_accuracy = accuracy_score(val_labels, val_preds)
val_report = classification_report(val_labels, val_preds, target_names=train_data.class_indices.keys())

print("Validation Accuracy:", val_accuracy)
print("Validation Classification Report:\n", val_report)
```

```
Validation Accuracy: 0.22814321398834306
Validation Classification Report:
               precision    recall  f1-score   support

          A+       0.17      0.03      0.05       150
          A-       0.19      0.47      0.27       150
         AB+       0.38      0.11      0.17       150
         AB-       0.29      0.10      0.15       150
          B+       0.29      0.11      0.16       150
          B-       0.34      0.43      0.38       151
          O+       0.11      0.03      0.05       150
          O-       0.20      0.54      0.29       150

    accuracy                           0.23      1201
   macro avg       0.25      0.23      0.19      1201
weighted avg       0.25      0.23      0.19      1201
```

```
# Extract test features
test_features, test_labels = extract_features(test_data, feature_extractor)
test_labels = np.argmax(test_labels, axis=1)

# Test the model
test_preds = svm.predict(test_features)
test_accuracy = accuracy_score(test_labels, test_preds)
test_report = classification_report(test_labels, test_preds, target_names=test_data.class_indices.keys())

print("Test Accuracy:", test_accuracy)
print("Test Classification Report:\n", test_report)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Test Accuracy: 0.22289156626506024
Test Classification Report:
              precision    recall  f1-score   support

          A+       0.31      0.06      0.10       145
          A-       0.19      0.44      0.26       145
         AB+       0.42      0.10      0.17       145
         AB-       0.27      0.08      0.12       145
          B+       0.29      0.14      0.19       145
          B-       0.29      0.41      0.34       147
          O+       0.12      0.04      0.06       145
          O-       0.19      0.51      0.28       145

    accuracy                           0.22      1162
   macro avg       0.26      0.22      0.19      1162
weighted avg       0.26      0.22      0.19      1162
```