Hvt: lò Văn Tâm msv:202111641

import os  
import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import svm  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
from sklearn.preprocessing import LabelEncoder  
from skimage.feature import hog  
  
  
def load\_images\_from\_folder(folder, max\_images\_per\_class=100):  
 images = []  
 labels = []  
 for label\_name in os.listdir(folder):  
 label\_folder = os.path.join(folder, label\_name)  
 if os.path.isdir(label\_folder):  
 image\_count = 0  
 for filename in os.listdir(label\_folder):  
 img\_path = os.path.join(label\_folder, filename)  
 img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)  
 if img is not None:  
 img = cv2.resize(img, (64, 64)) # Resize ảnh về kích thước cố định  
 images.append(img)  
 labels.append(label\_name)  
 image\_count += 1  
 if image\_count >= max\_images\_per\_class:  
 break  
 return np.array(images), np.array(labels)  
  
  
def extract\_features(images):  
 features = []  
 for img in images:  
 hog\_features = hog(img, pixels\_per\_cell=(8, 8), cells\_per\_block=(2, 2), orientations=9, block\_norm='L2-Hys')  
 features.append(hog\_features)  
 return np.array(features)  
  
  
def evaluate\_models(X, y, train\_test\_splits):  
 results = []  
 for train\_size, test\_size in train\_test\_splits:  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=train\_size, test\_size=test\_size,  
 random\_state=42)  
  
 # SVM Model  
 svm\_model = svm.SVC(kernel='linear')  
 svm\_model.fit(X\_train, y\_train)  
 y\_pred\_svm = svm\_model.predict(X\_test)  
 svm\_report = classification\_report(y\_test, y\_pred\_svm, output\_dict=True)  
  
 # KNN Model  
 knn\_model = KNeighborsClassifier(n\_neighbors=3)  
 knn\_model.fit(X\_train, y\_train)  
 y\_pred\_knn = knn\_model.predict(X\_test)  
 knn\_report = classification\_report(y\_test, y\_pred\_knn, output\_dict=True)  
  
  
 results.append({  
 "train\_test\_split": f"{int(train\_size \* 100)}-{int(test\_size \* 100)}",  
 "svm\_accuracy": svm\_report['accuracy'],  
 "knn\_accuracy": knn\_report['accuracy'],  
 "svm\_report": svm\_report,  
 "knn\_report": knn\_report  
 })  
  
 return results  
  
  
input\_folder = r'C:\Users\84846\PycharmProjects\Project1\Yte\_dataset'  
  
images, labels = load\_images\_from\_folder(input\_folder, max\_images\_per\_class=100)  
  
  
if len(images) == 0:  
 raise ValueError("No images were loaded. Please check the input folder path and ensure it contains images.")  
  
  
features = extract\_features(images)  
  
  
label\_encoder = LabelEncoder()  
encoded\_labels = label\_encoder.fit\_transform(labels)  
  
  
train\_test\_splits = [(0.8, 0.2), (0.7, 0.3), (0.6, 0.4), (0.4, 0.6)]  
  
  
results = evaluate\_models(features, encoded\_labels, train\_test\_splits)  
  
  
  
def plot\_results(results):  
 splits = [result['train\_test\_split'] for result in results]  
 svm\_accuracies = [result['svm\_accuracy'] for result in results]  
 knn\_accuracies = [result['knn\_accuracy'] for result in results]  
  
 x = np.arange(len(splits))  
 width = 0.3  
  
 fig, ax = plt.subplots()  
 bars1 = ax.bar(x - width / 2, svm\_accuracies, width, label='SVM Accuracy')  
 bars2 = ax.bar(x + width / 2, knn\_accuracies, width, label='KNN Accuracy')  
  
  
 ax.set\_ylabel('Accuracy')  
 ax.set\_title('Model Accuracy Comparison')  
 ax.set\_xticks(x)  
 ax.set\_xticklabels(splits)  
 ax.legend()  
  
  
 def autolabel(bars):  
  
 for bar in bars:  
 height = bar.get\_height()  
 ax.annotate(f'{height:.2f}',  
 xy=(bar.get\_x() + bar.get\_width() / 2, height),  
 xytext=(0, 3), # 3 điểm trên thanh  
 textcoords="offset points",  
 ha='center', va='bottom')  
  
 autolabel(bars1)  
 autolabel(bars2)  
  
 plt.show()  
  
  
  
plot\_results(results)

kết quả trương trình:

