Universidad Nacional de San Antonio Abad del Cusco
Departamento Académico de Ing. Informática
VISION COMPUTACIONAL
Práctica Nº 12

RECONOCIMIENTO DE IMÁGENES MEDIANTE DESCRIPTOR HOG Y SVM LINEAL

Iván C. Medrano Valencia

1. OBJETIVO.

- Implementar el reconocimiento de imágenes mediante el descriptor HOG y Support Vector Machine

2. PROYECTO DE RECONOCIMIENTO DE IMÁGENES.

2.1. LA ESTRUCTURA DEL PROYECTO Y EL CONJUNTO DE DATOS

Para el reconocimiento de imágenes, usaremos dos conjuntos de datos. Primero, usaremos un pequeño conjunto de datos de flores para entrenar y predecir las características de HOG usando Linear SVM. Luego usaremos otro conjunto de datos que consiste en humanos, autos y tazas.
La estructura del proyecto es:

```
    hog_image_recognition.py
├──input
│   ├──flowers
│   │   ├──daffodil
│   │   ├──lily
│   │   ├──rose
│   │   └──sunflower
│   └──person_car_cup
│       ├──car
│       ├──cup
│       └──person
├──outputs
└──test_images
    ├──flowers
    └──person_car_cup
```

Tenemos **hog_image_recognition.py** que contendrá nuestro código Python.
Luego, tenemos la carpeta **input** que contendrá nuestros dos conjuntos de datos, **flowers** y **person_car_cup**.

- **flowers** tiene cuatro subcarpetas y contienen las imágenes de flores según sus nombres. Esas son **daffodil flowers**, **lily flowers**, **rose flowers**, and **sunflowers**. Cada uno de los tipos de flores contiene 6 imágenes.
- **person_car_cup** tiene tres subcarpetas, **car**, **cup** y **person**. Cada carpeta tiene 5 imágenes del tipo respectivo.

La carpeta **output** guardará las salidas al ejecutar el script de Python.

Finalmente, la carpeta **test_images** nuevamente tiene dos subcarpetas con los nombres correspondientes a los conjuntos de datos.

- La subcarpeta **flowers** contiene una flor de cada tipo. Y **person_car_cup** tiene una imagen de cada tipo.

Algunas de las imágenes del conjunto de datos son:
Primero, las siguientes son algunas de las imágenes de flores.




La siguiente figura muestra una imagen de cada categoría de persona, automóvil y taza de la carpeta **input**


2.2. SCRIPT DE PYTHON PARA EL RECONOCIMIENTO DE IMÁGENES USANDO EL DESCRIPTOR HOG

Escribiremos un solo script de Python para entrenar y predecir usando un modelo SVM lineal en los dos conjuntos de datos. Eso significa que tenemos que emplear

algunos métodos con los que podemos ingresar el nombre del conjunto de datos y nuestro script se entrenará y predecirá automáticamente sobre eso.

2.2.1. IMPORTAR LOS MÓDULOS

Usaremos la implementación de Scikit-Image del descriptor de características del HOG. También hay una implementación de OpenCV. Pero se basa más en el papel original y facilita el reconocimiento y la detección de humanos. Entonces, usaremos la implementación de Scikit-Image.

```
'''
USAGE:
python hog_image_recognition.py --path person_car_cup
python hog_image_recognition.py --path flowers
'''

import os
import cv2
import argparse

from sklearn.svm import LinearSVC
from skimage import feature
```

Estamos importando un módulo de características de skimage que tiene una implementación para calcular las características de HOG a partir de imágenes. También estamos importando LinearSVC del módulo SVM de Scikit-Learn. Estamos utilizando LinearSVC ya que los autores también hicieron lo mismo en su artículo original.

2.2.2. CONSTRUYENDO EL ANALIZADOR DE ARGUMENTOS

Ejecutaremos el script de Python desde la línea de comandos. Durante la ejecución, solo daremos el nombre del conjunto de datos como uno de los argumentos. Luego, en el script de Python, el entrenamiento y la predicción ocurrirán automáticamente en ese conjunto de datos.

```
# construct the argument parser and parser the arguments
parser = argparse.ArgumentParser()
parser.add_argument('-p', '--path', help='what folder to use for HOG
description',
                    choices=['flowers', 'person_car_cup'])
args = vars(parser.parse_args())
```

Entonces, el argumento --path tomará **flowers** o **person_car_cup** como opción. Solo de acuerdo con esto, se llevará a cabo el resto del entrenamiento y la predicción.

2.2.3. OBTENER LAS FUNCIONES DEL HOG A PARTIR DE LAS IMÁGENES DE ENTRENAMIENTO

Para entrenar un modelo SVM lineal, necesitamos las características HOG. Estas características actuarán como datos. Y las etiquetas (nombres de las carpetas) actuarán como etiquetas.

```
1. images = []
2. labels = []
3. # get all the image folder paths
4. image_paths = os.listdir(f"input/{args['path']}")
5. for path in image_paths:
6. # get all the image names
7. all_images = os.listdir(f"input/{args['path']}/{path}")
8.
9. # iterate over the image names, get the label
10.for image in all_images:
      11.image_path = f"input/{args['path']}/{path}/{image}"
      12.image = cv2.imread(image_path)
      13.image = cv2.resize(image, (128, 256))
      14.
      15.# get the HOG descriptor for the image
      16.hog_desc = feature.hog(image, orientations=9,
                  pixels_per_cell=(8, 8),
      17.cells_per_block=(2, 2), transform_sqrt=True,
            block_norm='L2-Hys')

      18.

      19.# update the data and labels
      20.images.append(hog_desc)
      21.labels.append(path)
```

- Creamos dos listas vacías, imágenes y etiquetas que almacenarán nuestras características y etiquetas de hog, respectivamente.
- En la línea 4, obtenemos todas las rutas de la carpeta de imágenes de acuerdo con el argumento de la línea de comando.
- Comenzamos a iterar sobre todas las carpetas de imágenes en la línea 5. En la línea 7, obtenemos todas las imágenes de la carpeta de imágenes respectiva.
- Luego, desde la línea 10, comenzamos a iterar sobre todas las imágenes de la carpeta.
    - Leemos la imagen usando OpenCV y la redimensionamos a dimensiones de 128 × 256 (ancho x alto). Recuerde que la proporción debe ser 1: 2 en formato ancho x alto.
    - En la línea 16, obtenemos las características HOG de la imagen. No obtenemos las visualizaciones, ya que no las necesitamos en este caso.
    - Tenemos 9 contenedores (bins) de orientación, 8 × 8 celdas, 2 × 2 bloques y el esquema de normalización es L2-Hys.
- Finalmente, agregamos las imágenes y etiquetas a la lista.

2.2.4. ENTRENAMIENTO DE SVM LINEAL EN LAS CARACTERÍSTICAS DEL HOG

Una vez que organizamos nuestros datos y etiquetas correctamente, el entrenamiento es solo dos líneas de código. Necesitamos inicializar un objeto SVM lineal y llamar al método **fit ()** mientras pasamos la característica y las etiquetas como argumentos.

El siguiente bloque de código entrena una SVM lineal en las características del HOG que obtuvimos anteriormente.

```
# train Linear SVC
print('Training on train images...')svm_model =
LinearSVC(random_state=42, tol=1e-5)
svm_model.fit(images, labels)
```

para el modelo SVM lineal, tenemos un estado aleatorio de 42 y una tolerancia de 1e-5.

2.2.5. PREDICCIÓN EN IMÁGENES DE PRUEBA

```
1.  # predict on the test images
2.  print('Evaluating on test images...')
3.  # loop over the test dataset folders
4.  for (i, imagePath) in enumerate(os.listdir(f"test_images/{args['path']}/")):
5.
6.  image = cv2.imread(f"test_images/{args['path']}/{imagePath}")
7.  resized_image = cv2.resize(image, (128, 256))
8.
9.  # get the HOG descriptor for the test image
10. (hog_desc, hog_image) = feature.hog(resized_image, orientations=9, pixels_per_cell=(8, 8),
11. cells_per_block=(2, 2), transform_sqrt=True, block_norm='L2-Hys', visualize=True)
12. # prediction
13. pred = svm_model.predict(hog_desc.reshape(1, -1))[0]
14.
15. # convert the HOG image to appropriate data type. We do...
16. # ... this instead of rescaling the pixels from 0. to 255.
17. hog_image = hog_image.astype('float64')
18. # show thw HOG image
19. cv2.imshow('HOG Image', hog_image)
20.
21. # put the predicted text on the test image
22. cv2.putText(image, pred.title(), (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1.0,
23. (0, 255, 0), 2)
24. cv2.imshow('Test Image', image)
25. cv2.imwrite(f"outputs/{args['path']}_hog_{i}.jpg", hog_image*255.) # multiply by 255. to
    bring to OpenCV pixel range
26. cv2.imwrite(f"outputs/{args['path']}_pred_{i}.jpg", image)
27. cv2.waitKey(0)
```

- En las líneas 6 y 7, leemos y cambiamos el tamaño de la imagen.
- En la línea 10 obtiene las características HOG. El argumento de visualización es true para que podamos visualizar las características del HOG.
- En la línea 13, predecimos la salida de las características del HOG, es decir, hog_desc.
- La línea 17 convierte hog_image en tipo float64 antes de la visualización en la línea 19.

- En la línea 22, colocamos la etiqueta pre indicada en la imagen original. Luego visualizamos la imagen con la etiqueta.
- Finalmente, guardamos la imagen de las características del HOG y la imagen y etiqueta pre indicadas para su posterior análisis.

2.2.6. EJECUTANDO EL SCRIPT DE PYTHON

Ahora, ejecutaremos el script de Python para entrenar y probar en los dos conjuntos de datos.
Comenzaremos con el conjunto de datos flowers.

```
python hog_image_recognition.py --path flowers
```

En la terminal, verá la siguiente salida.

```
Training on train images...
Evaluating on test images...
```

Y para el conjunto de datos `person_car_cup`, hacemos:

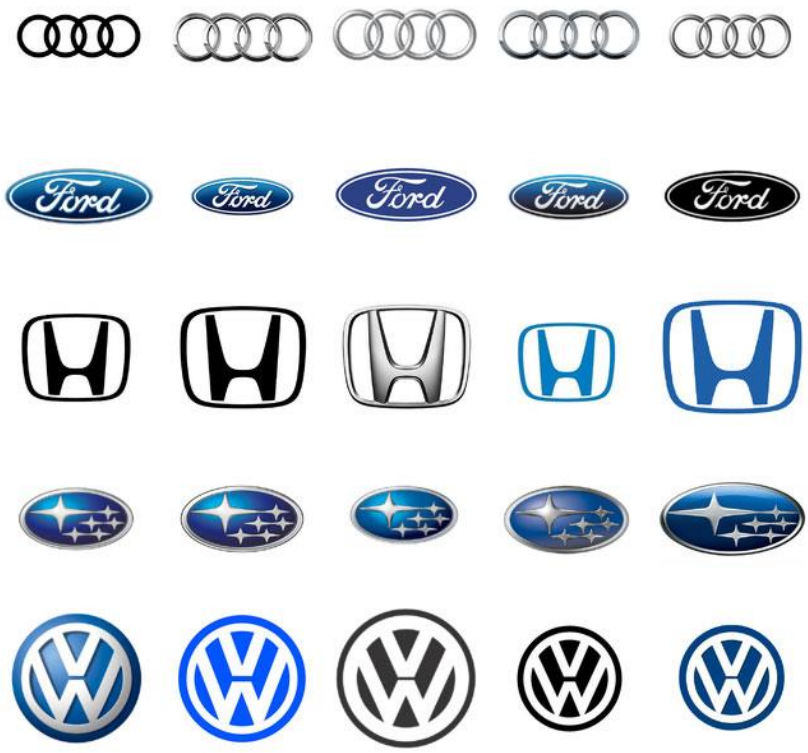
```
python hog_image_recognition.py --path person_car_cup
```

3. CUESTIONARIO

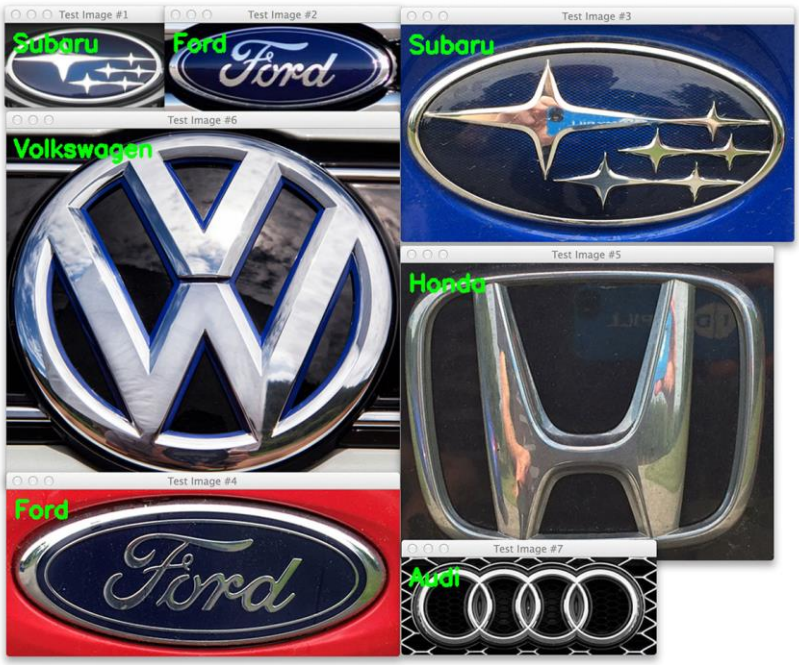
- Analice los resultados de los conjuntos de datos de flowers y person_car_up. Comente si están bien o no las predicciones.

4. TAREA

Escribir un programa para reconocer el logo de diferentes marcas de automóviles. Para ello se dispone de imágenes de entrenamiento como se ven a continuación:



El resultado debe ser el siguiente:



5. REFERENCIAS BIBLIOGRÁFICAS

- Dawson-Howe, K. (2014). A Practical Introduction to Computer Vision With OpenCV, Wiley & Sons Ltd.
- Hafsa Asad, W. R., Nikhil Singh (2020). The Computer Vision Workshop. Birmingham U.K., Pack Publishing.
- Szeliski, R. (2011). Computer Vision Algorithms and Applications. London, Springer.