UNIVERSIDAD REY JUAN CARLOS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

PRÁCTICA Nº 2 de Visión Artificial
Reconocimiento de objetos

Pedro David Parla García, Raquel Rodríguez Recio y Francisco Javier García Arias Tajuelo

ÍNDICE

1. EXPLICACIÓN DEL ALGORITMO UTILIZADO

Para el reconocimiento de objetos de esta práctica, el algoritmo desarrollado se explica a continuación.

Nuestro código se estructura en 3 clases:

1. Una clase llamada ***clasificador***, la cual tiene los siguientes métodos:

- Un constructor.
- Un método ***fit*** que, con las características extraídas de las imágenes y las etiquetas, entrena el clasificador.
- Un método *predict que se encarga de echarte las cartas y predecir tu vida de mierda.*
- *Un método llamado comprobar_resultados* que se encarga de calcular la tasa de acierto obtenida.

2. Una clase ***extCaracteristicas***, que se encarga de la extracción de características. Se compone de:

- Un constructor.
- Una función **ecualizar**, que se encarga de ecualizar el histograma de cada imagen para mejorar su contraste. Si la imagen es muy oscura, la aclarará. De lo contrario, la oscurecerá.
- Un método llamado ***tratar***, que ecualiza cada imagen para mejorar su contraste con el método ***ecualizar***, la transforma a escala de grises, alinea los bloques con HOG con la característica *interpolation* y, finalmente, devuelve la imagen.
- Un método llamado ***extraerCaracteristicas*** que trata cada imagen (método **tratar**), aplica el descriptor de HOG para calcular el histograma de gradientes orientados y obtiene los descriptores de las imágenes escaladas tras ejecutar HOG con una dimensión menos. Finalmente, los descriptores se almacenan en un array.

3. Una clase ***main***, que se explica a continuación:

- En primer lugar, se recogen todos los 3 argumentos indicados en el enunciado (***método recoger_args***).

- Después se comprueba qué clasificador se desea ejecutar, y según el que se indique, se ejecutará uno u otro (método ***clasificar***).
- Finalmente, se obtienen los resultados tras ejecutar un clasificador u otro.

Funcionamiento del método clasificar

Primero se recorre el directorio *train* recibido en el parámetro *train_path* (función ***recorrerDirectorios***) para coger las imágenes junto con las etiquetas.

Según el clasificador que indiquemos en los parámetros, se instanciará un tipo u otro (según el cual se indique en el parámetro clasificador) y se entrenará con las imágenes y las etiquetas obtenidas tras recorrer los directorios mediante el método *fit*.

A continuación, se obtienen los resultados tras ejecutar el *predict* con las imágenes que se encuentran dentro de la ruta recibida en el parámetro *test_path*.

Después, en función del nombre de cada imagen, es decir, si empieza por 0 cogemos el dígito que le continúa, y si no, cogemos los dos dígitos completos.

Tras esto, se ejecuta el método ***comprobar_resultados***, que obtiene la tasa de acierto en forma de porcentaje y se imprime por pantalla.

2. *MÉTODOS DE OPENCV UTILIZADOS*

Los métodos de OpenCV que hemos utilizado en esta práctica son los siguientes:

- **cv.imread:** para leer imágenes de un directorio.
- **Cv.HOGDescriptor**: para obtener el histograma de gradientes orientados.
- **cv.split:** para separar una imagen en los 3 canales de color que tiene.
- **cv.CvtColor**: para las transformaciones de color de las imágenes.
- **cv.resize**: para redimensionar el tamaño de una imagen.
- **cv.CLAHE**: para ecualizar el histograma y así mejorar el contraste de la imagen.
- **cv.merge**: para combinar varios arrays en un único array multicanal.

2.1. Procedencia de trozos de código extraídos de Internet

En nuestro programa hay pequeños trozos de código procedentes de Internet. A continuación, indicamos de dónde hemos sacado cada uno de ellos.

- El código utilizado para ecualizar el histograma de una imagen y así mejorar el contraste y visibilidad de la imagen lo hemos obtenido del siguiente enlace.

  https://stackoverflow.com/questions/39308030/how-do-i-increase-the-contrast-of-an-image-in-python-opencv

3. *EJECUCIÓN DEL PROGRAMA*

En este apartado se muestra un pantallazo de la ejecución del programa con cada uno de los 4 clasificadores que hemos utilizado.

4. *ESTADÍSTICAS DE LAS EJECUCIONES DEL PROGRAMA*

Respecto a las estadísticas, con los clasificadores que se indican a continuación, nuestro código obtiene las siguientes tasas de acierto:

- **LDA**: 97.52%.
- **KNN**: 93.39%, con n_neighbours = 3 (es el máximo porcentaje que hemos obtenido).
- **Random Forest**: 96.69%, con 500 árboles (es el porcentaje con menos varianza).
- **LDA**-KNN: 98.35%, con n_components = por defecto y neighbours = 6. Hemos probado con más n_components pero nos da el mismo resultado.