Ingeniería Informática (Universidad Rey Juan Carlos)

Asignatura: “Visión Artificial”
Curso 2023/24

Práctica obligatoria 2: “”Aprendizaje automático para la lectura de paneles informativos en autopistas

Normas generales:

- La práctica deberá realizarse en Python usando el entorno Jupyter Notebook, Visual Studio Code o similar (en combinación con las librerías correspondientes).

- Habrá que entregar: 1) los códigos desarrollados/usados para la realización de la práctica y 2) una breve memoria explicativa, que incluirá: la descripción de la solución, los resultados conseguidos junto con su análisis y las conclusiones del trabajo. La entrega se realizará a través de una ´”carpeta de entregas” que se habilitará para tal fin en el Aula Virtual de la asignatura.

- La práctica se realizará **exclusivamente en grupos de dos o tres alumnos** (como máximo). Para ello, se mantendrán los grupos de prácticas creados para la Práctica 1.

- El plazo límite de entrega será de la práctica será el día 23 de mayo de 2024 (incluido).

1 Copias de código o de la memoria

El código desarrollado en las prácticas debe de ser original. La copia (total o parcial) de prácticas será sancionada, al menos, con el SUSPENSO global de la asignatura en la convocatoria correspondiente. En estos casos, en la siguiente convocatoria supondrá el tener que **resolver nuevas pruebas (distintas de las de mayo) y la defensa de las mismas de forma oral**. **Las sanciones derivadas de la copia, afectarán tanto al alumno que copia como al alumno copiado.**

Para evitar que **cuando se usa código de terceros** sea considerado una copia, se debe **citar siempre la procedencia** de cada parte de código no desarrollada por el propio alumno (con comentarios en el propio código y con mención expresa en la memoria). El plagio o copia de terceros (p.ej. una página web) ya sea en el código a desarrollar en las prácticas y/o de parte de la memoria de las prácticas, sin la cita correspondiente, acarrearán las mismas sanciones que en la copia prácticas de otros alumnos.

2 Reconocimiento Óptico de Caracteres (OCR)

En esta práctica se desea construir un sistema que permita: (a) la localización de los caracteres que aparecen en los paneles informativos de autopista y (b) su lectura automática (ver Figura 1). Para ello partiremos de las detecciones de paneles ya realizada en la Práctica 1. En esta práctica implementaremos los pasos la lectura automática de los paneles (reconocimiento de los caracteres).

Figura 1: Ejemplo de imagen de un panel detectado (es importante darse cuenta de que las líneas de texto podrían aparecer algo inclinadas).

Para resolver el problema tendremos que:

- **Entrenar y validar un clasificador de caracteres**. Se utilizarán las imágenes provistas en el directorio *train_ocr* (ver Figura 2) para entrenar un clasificador multiclase, con una clase por cada tipo de carácter (letra o número). En las imágenes tenemos cada uno de los caracteres que podemos ver en los paneles de carretera con algunas transformaciones geométricas y con algo de ruido añadido, lo que proporcionará mayor variabilidad a las muestras que usará el clasificador. Sobre estas imágenes habrá que localizar el carácter con el mismo procedimiento que se utilice luego en el panel de carretera.

Figura 2: Ejemplos de imágenes de entrenamiento para el clasificador de texto.

- **Detectar los caracteres alineados.** En este caso se tratará de detectar los caracteres y las líneas de texto que forman (p.ej. con RANSAC de líneas). En la Figura 1 se muestra una posible detección de caracteres y líneas de texto.

- **Lectura del texto del panel (OCR).** Para la lectura automática del panel de carretera procederemos a ejecutar el clasificador entrenado a las regiones de imagen recortadas de izquierda a derecha y de arriba a abajo del panel, siguiendo las líneas de texto.

2.1 Ejercicio1: Entrenamiento y validación del clasificador de caracteres

Para construir el sistema de clasificación se proporciona un conjunto de entrenamiento que contiene 625 variaciones de cada uno de los caracteres que puede aparecer en un panel azul de carretera (los caracteres: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ).

En el Aula Virtual se proporciona un esquema de código para facilitar la implementación de las pruebas a realizar. En particular se propone (aunque no es obligatorio) utilizar una jerarquía con una clase base *OCRClassifier* y el clasificador pedido como clase derivada, *LdaNormalBayesClassifier*, e implementando los métodos *train* y *predict*.

Para entrenar el clasificador básico se pide seguir los siguientes pasos:

- Cargar y umbralizar las imágenes de los caracteres de entrenamiento (por ejemplo, cv2.adaptiveThreshold + cv2.findContours + cv2.boundingRect). Este proceso (umbralización + findContours) habrá que hacerlo también en las imágenes de los paneles reales. De esta manera el clasificador verá el mismo tipo de imágenes en entrenamiento y pruebas reales.

- Construir el vector de características a partir de las imágenes de entrenamiento del carácter ya recortadas usando el rectángulo detectado. En la versión básica a implementar se sugiere utilizar directamente los niveles de gris. Todos los vectores de características deben de ser del mismo tamaño con lo que habrá que redimensionar la imagen del carácter a ese tamaño fijo (por ejemplo a 25x25 píxeles). Convertir la matriz con los niveles de gris del carácter ya redimensionada, con un tamaño de 25x25 y valores entre 0 y 255, en una matriz con una sola fila y 625 columnas. Cada una de estas matrices filas se usará como vector de características asociado al carácter de entrenamiento.

- Reducir la dimensión de los vectores de características usando LDA (*Linear Discriminant Analysis*). Para ello se debe:

    - Agrupar en una matriz *C* (características) todas las filas generadas en el paso anterior. En este caso, será una matriz con tantas filas como dígitos usemos para aprender y 625 columnas.

    - Crear un vector de enteros *E* (etiquetas) con las clases en la que se debe clasificar cada uno de los dígitos de aprendizaje. Obsérvese que la fila *f* de *C* contiene la imagen y la posición *f* de *E* contendrá la clase a la que pertenece (Por ejemplo, 1 para el 0, 2 para el 1, … , 11 para la a, etc.).

    - Utilizar la implementación de LDA disponible en el paquete de *scikit-learn* en la clase *LinearDiscriminantAnalysis*. El método *LinearDiscriminantAnalysis.fit()* se utilizará para encontrar la matriz de proyección LDA.

    - Utilizar el método *transform* del objeto de la clase *LinearDiscriminantAnalysis* utilizado para proyectar *C* en un espacio de dimensión menor, creando una nueva matriz *CR* (características reducidas) .

- Entrenar un clasificador con la matriz *CR* y las etiquetas *E,* y usarlo para reconocer los caracteres de cualquier panel. En la biblioteca *scikit-learn* y también en la biblioteca OpenCV (en el módulo *ml* de "machine learning") tenemos implementados los clasificadores vistos en la asignatura. En la práctica se trata de probar los diferentes clasificadores y escoger el que de el mejor resultado. Hay que tener en cuenta que los métodos de OpenCV en el módulo *ml* únicamente aceptan matrices de tipo `np.float32`, y no `np.float64`, como utiliza por defecto *scikit-learn.* Habrá que hacer una conversión de tipos antes de pasar matrices de una a otra bibliotecas. Además, el vector de etiquetas *E* en OpenCV tiene que ser de tipo `np.int32` para que no se produzca un error de ejecución.

- En este primer ejercicio se pide evaluar el clasificador sobre el conjunto de imágenes en el directorio de *validation_ocr*. Se deberán utilizar cuantas medidas de rendimiento del clasificador se crean oportunas (lo que redundará en obtener la máxima nota) aunque el mínimo será la tasa de acierto (*accuracy*).

La implementación de este ejercicio se podrá hacer (aunque no es obligatorio) completando el *script* proporcionado *evaluar_clasificadores_OCR.py*. A este se le pasará el *path* al directorio *train_ocr*, al directorio *validation_ocr* y el nombre del clasificador que se quiera probar (internamente en el *script* el alumno permitirá elegir el clasificador a probar con este parámetro).

2.2 Ejercicio 2. Probar otras alternativas para el reconocimiento.

Este ejercicio consiste en comparar los resultados que ofrece el sistema de reconocimiento desarrollado con otras alternativas. Entre las posibles alternativas se proponen:

- Utilizar otros vectores de características (aparte de los niveles de gris) en el clasificador básico:
    - Otras medidas disponibles en OpenCV o en *skimage.feature* (http://scikit-image.org/docs/dev/api/skimage.feature.html), descriptores de características, etc .
- Otros algoritmos de reducción de dimensionalidad:
    - Por ejemplo, PCA (Principal Component Analysis) disponible en sklearn (http://scikit-learn.org/stable/modules/decomposition.html) o también la ausencia de reducción de dimensionalidad.

      Nótese que al usar LDA o PCA como reducción de dimensionalidad con otros clasificadores (distinto del que proporciona la clase LDA de *sklearn*), implicará que hay que usar el método *transform* para bajar la dimensionalidad antes de entrenar y también al ejecutar los clasificadores.
- Utilizar Otros clasificadores:
    - Euclídeo, KNN o cualquier otro clasificador (supervisado) disponible en OpenCV y/o *sklearn*. Es importante hacer notar aquí que el clasificador tiene que ser multi-clase.

Si se utiliza el código *evaluar_clasificadores_OCR.py,* el código desarrollado para esta parte de la práctica deberá permitir elegir cada uno de los diferentes clasificadores mediante el parámetro correspondiente. Además, la comparativa entre el sistema básico y las alternativas implementadas en esta sección quedará reflejada en la memoria de la práctica. La claridad y adecuación en la exposición de las medidas de rendimiento (tasas de acierto, matrices de confusión, gráficos, curvas ROC o de Precisión/Recall, etc.) será valorada en la nota.

Para obtener la **máxima puntuación en este ejercicio** habría que probar con relativo éxito sobre el conjunto de datos del directorio *validation_ocr*, **2 alternativas al sistema básico del Ejercicio 1** (entendiendo como alternativa el cambio de cualquiera de los componentes del sistema básico – extracción de características, algoritmo de reducción de dimensionalidad o clasificador) con una discusión adecuada de los resultados.

2.3 Ejercicio 3: Lectura automática de paneles recortados

En este caso se trata de poder localizar los caracteres y las líneas de texto en los paneles ya recortados y que se encuentran en el directorio *test_ocr_panels* (ver Figura 3, centro). Adicionalmente, en este ejercicio se pretende que el alumno aplique el mejor clasificador del apartado anterior a los caracteres

localizados en la imagen del panel (ver Figura 3, derecha).

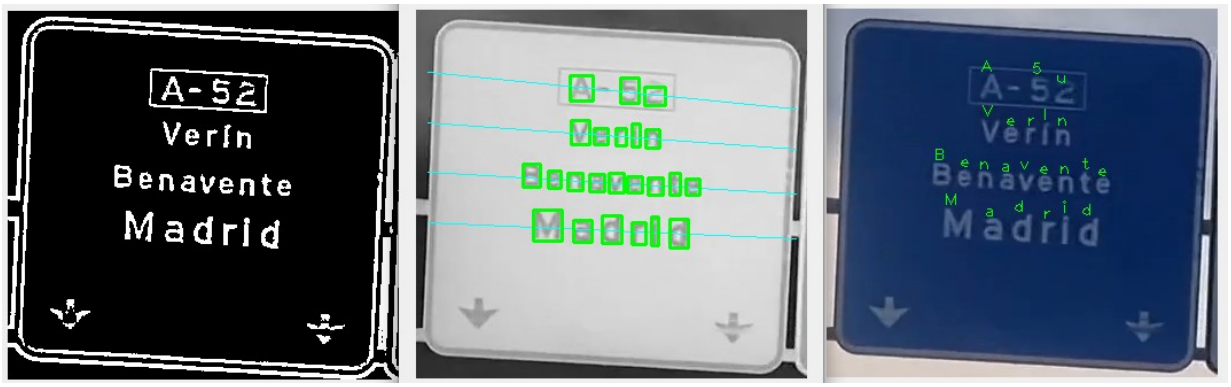

Figura 3: Ejemplo de imagen de un panel umbralizado (izquierda), detecciones de caracteres con MSER (centro), detección de líneas de texto con RANSAC (centro) y estimación de la clase de cada carácter (derecha).

Un posible algoritmo para este ejercicio sería:

1. Detectar los caracteres mediante umbralización y búsqueda de componentes conexas (cv2.adaptiveThreshold + cv2.findContours). Se deberán eliminar aquellos rectángulos con relaciones de aspecto no comparables a las de un carácter.

2. Encontrar los caracteres alineados mediante, por ejemplo, RANSAC sobre los centros de cada carácter. Si se usa RANSAC, será un proceso iterativo ya que se procederá a encontrar la primera línea, eliminando después los *inliers* a la misma y se procederá a encontrar la siguientes línea de texto.

3. Se ejecutará el clasificador sobre cada uno de los caracteres en el orden de izquierda a derecha y de arriba a abajo (orden de lectura). Se generará un único *string* de salida que representará todas las líneas de texto sin espacios y con el salto de línea sustituido por un '+'. Por ejemplo, el texto "Hola+por la+mañana" representa tres líneas de texto "Hola", "por la" y "mañana".

Todas las prácticas en este apartado se podrán implementar (aunque no es obligatorio) en un *script main_panels_ocr.py* del que se proporciona un esquema en el Aula Virtual. Además, la salida de este script será un fichero *resultado.txt*. El fichero de texto con las anotaciones, una por línea, sigue el formato indicado:

```
<nombre_fichero>;<x1>;<y1>;<x2>;<y2>;<tipo>;<score>;<texto_ocr>
```

En los parámetros *x1*, *y1*, *x2*, *y2* se colocarán los de una ventana que ocupa toda la imagen (al fin y al cabo la imagen completa es un panel recortado). En el parámetro *texto_ocr* deberemos escribir el *string* de salida del panel con el formato explicado anteriormente (sin espacios y con el '+' como separador de líneas).

Se proporciona un script, *evaluar_resultados_test_ocr_panels.py.* que permite comparar el resultado de nuestro fichero de texto con el resultado de la práctica (ver Figura 4) con el fichero con las anotaciones hechas a mano (*gt.txt* en *test_ocr_panels*). Este *script* nos dará una gráfica con el histograma que compara el texto de los paneles reconocidos por nuestra práctica con las correctas utilizando la *distancia de Levenshtein* (número de cambios en una cadena de texto para convertirla en otra).

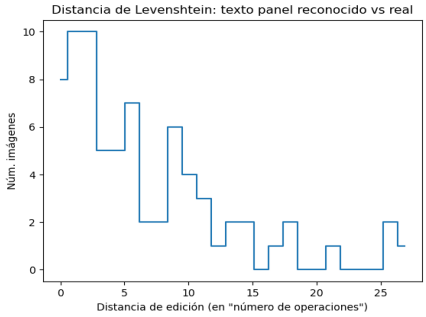

Figura 4: Resultados de ejemplo para el directorio test_ocr_panels. El histograma nos dice que hay 38 paneles con distancia de Levenshtein por debajo de 5 y unas pocas con errores en más caracteres.

2.4 Ejercicio 4: Lectura automática de paneles

El ejercicio 4 consiste en unir el sistema de reconocimiento diseñado en el apartado anterior con el detector MSER desarrollado en la Práctica 1. De esta manera tendremos el sistema de lectura de paneles de carretera completo modificando el *script* de la Práctica 1, *main.py*, añadiendo la nueva funcionalidad.

Durante la ejecución del *script se mostrará en pantalla, al menos* (se puede decidir enseñar más información): la imagen que se está procesando, los paneles detectados marcados con rectángulos y su *score*, la detección de los caracteres (un rectángulo sobre cada uno de los detectados), rectas que unan los caracteres de la misma línea de texto y el carácter reconocido sobre cada rectángulo de texto del panel (ver Figura 3). La visualización será de tal manera que el usuario tenga que dar a una tecla para pasar a la siguiente imagen. (Se deberá dar la opción de desactivar la visualización o no de resultados con un parámetro "--visualize_ocr" que se le pasa al script *main.py* y que por defecto será False si no se le pasa ese parámetro).

3 Datos de entrenamiento, test y formato de salida

Los **datos de entrenamiento del OCR** proporcionados son imágenes .png con caracteres (directorio *train_ocr*).

Los **datos de validación del OCR** proporcionados son imágenes .png con caracteres (directorio validation_*ocr*).

Las **imágenes de test de la lectura automática de paneles** se encuentra en el directorio *test_ocr_panels* contiene imágenes de paneles recortados (resultado de una detección). Para **estas imágenes** se proporciona también un fichero de anotaciones *gt.txt* para poder establecer la distribución de errores en el texto reconocido.

Las prácticas entregadas podrán utilizar los *scripts python evaluar_clasificadores_OCR*.py **(Ejercicios 1 y 2)**, *evaluar_resultados_test_ocr_panels*.py **(Ejercicio 3)** que se proporcionan junto con este enunciado.

Para el **Ejercicio 4** se modificará el *script* de la Práctica 1 (incluyendo la salida nueva con el texto reconocido en el fichero *resultado.txt*)

4 Normas de presentación

La presentación seguirá las siguientes normas:

- Su presentación se realizará a través del Aula Virtual**.**
- Para presentarla se deberá entregar un único fichero ZIP que contendrá el código fuente en python y un fichero PDF con la descripción del sistema desarrollado.
- Dicho fichero PDF incluirá una explicación con el algoritmo desarrollado, los métodos de OpenCV y/o *sklearn* utilizados, copias de las pantallas correspondientes a la ejecución del programa y unas estadísticas correspondientes al resultado de la ejecución del programa sobre las muestras de test. Si se han probado diferentes sistemas de clasificación deberán reflejarse los diferentes resultados que se hayan obtenido.

La puntuación de esta práctica corresponde al 30 % de la asignatura. La práctica se valorará sobre 10 y cada parte tendrá la siguiente puntuación:

- Ejercicio 1: **4 puntos.**
- Ejercicio 2: **2.5 puntos.**
- Ejercicio 3: **2.5 puntos.**
- Ejercicio 4: **1 puntos.**

Para obtener la máxima nota en cada apartado, se valorará:

- Implementar los *scripts* como se pide en los diferentes ejercicios.
- La limpieza y organización del código en clases con un *Diseño Orientado a Objetos* razonable.
- El funcionamiento del código en las imágenes de test.
- Las ideas propias o técnicas pensadas por los alumnos para mejorar lo que se propone en el enunciado.

5 Referencias

1. Ejemplo de uso de LDA:

   http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html