UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

SISTEMAS INTELIGENTES

SECCIÓN 1369

CATEDRÁTICO:

PHD. KENNY DÁVILA

PRESENTADO POR:

Diego Ramos Ucles 21951033

Mini Proyecto 2: Clasificador de imágenes de refrescos portátiles

SAN PEDRO SULA, CORTÉS, HONDURAS

19 de diciembre del 2021

Tabla de Contenidos

Introduccion

El objetivo de este proyecto es desarrollar un clasificador capaz de reconocer la marca de una botella de refresco portátil. Este trabajo es de carácter investigativo y es parte del objetivo averiguar qué técnicas (ya sea de la vistas en clase o alguna que de cual investigamos) nos producen el mejor resultado. Es necesario aplicar técnicas de procesamiento de imágenes para extraer features que se utilizarán para entrenar un clasificador. Aunque el proyecto permite utilizar otros métodos para generar un clasificador funcional (con redes neuronales o redes convolucionales), en el presente trabajo se utilizan métodos menos complejos de ingeniería de features. Extraemos las features que creamos sean las más útiles y haremos pruebas de escritorio utilizando validación cruzada. De esta manera, podremos saber si nuestra selección de features es relevante y evitaremos el overfitting de nuestro calificador.

Implementación y Arquitectura

Se desarrollaron 4 scripts de python 3.10. Se utilizaron las siguientes librerías y módulos:

- Scikit-Learn
- OpenCV
- Pandas
- Numpy
- json
- pickle

OpenCV fue utilizado para el procesamiento de imágenes y la extracción de fuentes. **Scikit-Learn** fue la base del sistema clasificador tanto para entrenarlo y para sacar métricas de rendimiento. **Pandas** fue utilizado para la facilitación de lectura de archivos csv. **Numpy** se utilizó para la facilitación de escritura al archivo de características y también fue de mucha utilidad en el procesamiento de imágenes. El módulo **json** se utilizó para la lectura y escritura de archivos json y el módulo **pickle** se utilizó para la serialización del objeto clasificador de Scikit-learn

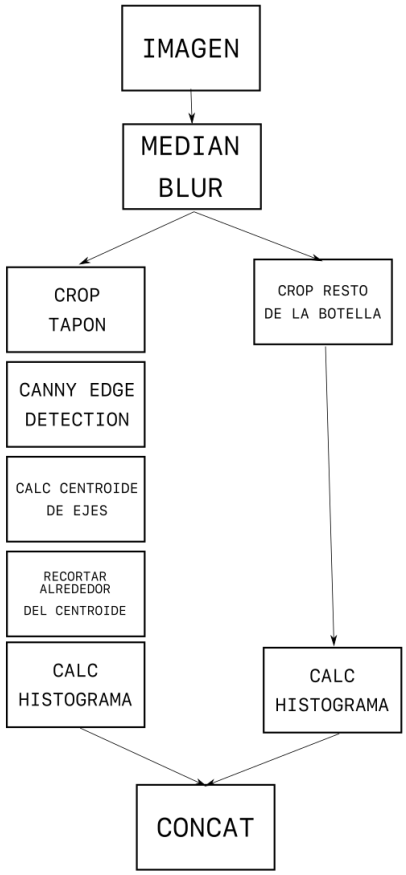
De los cuatro scripts, el primero (mp2_01_extraer_caracteristicas.py) hace el trabajo más pesado e incluso utiliza 5 hilos para distribuir los varios trabajos independientes que debe completar. Es encargado de extraer los features usando técnicas de procesamiento de imágenes que se utilizaran en los demás scripts. Las features extraídas son escritas en formato csv a un archivo especificado como el último argumento al correr el script.

El segundo script (mp2_02_cross_validation.py) es más que todo útil para hacer pruebas de escritorio. Estas pruebas tienen la intención de ayudarnos a saber si tanto las features que seleccionamos en el primer script como los algoritmos de clasificación (e incluso sus configuraciones) son las más óptimas para desarrollar el programa y darnos los resultados deseados.

El tercer script (mp2_03_entrenar_clasificador.py), como lo dice su nombre, entrena nuestro clasificador. Tiene como entradas el archivo de features que nos generó nuestro primer script, y un archivo de etiquetas que hace un mapeo de los nombre de las imágenes con la marca que estamos intentando clasificar. También tiene como última entrada, el nombre del archivo al cual se guardará una serialización del objeto clasificador en formato npy.

El último script (mp2_04_evaluar_clasificador.py), tiene como entradas el archivo de características y el archivo con el clasificador serializado. Su último argumento es el archivo al cual guardará las predicciones en forma de etiquetas en formato json.

Motivación del proceso



Se le aplica un filtro borroso a la imagen para simplificar los colores y eliminar las texturas. De esta manera, diferentes tonos de un color pueden ser simplificados a un solo tono. Luego se le aplican dos diferentes recortes a la imagen. Asumiendo que conocemos la posición aproximada de la botella dentro de la imagen, podemos recortar la imagen para enfocar las partes de interés (en este caso, el tapón y el cuerpo de la botella). Luego, al recorte del tapón se le aplica un filtro canny de edge detection para después calcular el centroide donde se acumulan la mayoría de ejes. Una vez calculado el centroide, recortamos una vez más la imagen del tapón para solo tener un recuadro de tapón. Esto se hace ya que es difícil saber donde esta el tapon en la imagen y no se puede aislar solo un recorte fijo. Hay imágenes que tienen el tapón muy arriba mientras otras están a casi ⅓ de la altura de la imagen. El recorte del resto de la botella se supone que no incluye toda la botella. Se supone que el recorte toma en cuenta el pequeño espacio sin líquido de la mayoría de las botellas. Así podemos enfocarnos en solo los colores de las etiquetas de los líquidos contenidos en el recipiente. Por último, se calculan los histogramas de colores de cada imagen recortada. Estos histogramas son tridimensionales, siendo formado por 5 x 5 x 5 “bins” de valores de colores de cada canal. Estos se terminan convirtiendo en arreglos de una dimensión y se concatenan para formar las características. Cada valor numérico de los resultados de los histogramas se vuelve un “feature” para el clasificador. No se aplica la normalización ya que se asume que todos los valores son de la misma unidad (valor de la intensidad de un canal color por área).

Ejemplos del proceso

imagen 01285 original

imagen 01285 con filtro median blur

imagen 01285 recortada: tapon

imagen 01285 recortada: botella

imagen 01285 Primer recorte

imagen 01285 Canny Edge Detection

imagen 00428 original

imagen 00428 con filtro median blur

imagen 00428 recortada: tapon

imagen 00428 recortada: botella

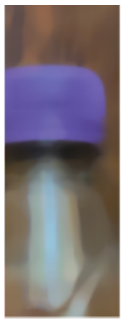


imagen 00428 Primer recorte

imagen 00428 Canny Edge Detection

Resultados: Validación Cruzada

Matriz de confusion

12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	23	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	50	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	6	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0
0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15

Efectividad:	0.9733333333
F-1 promedio por clase	0.974488

Métricas por cada iteración del K-Fold:

	P1	P2	P3	P4	P5	Promedio
F-1	0.9835	0.9693	0.9708	0.9653	0.9814	0.9741
Accuracy	0.98	0.9733	0.97	0.9667	0.9767	0.9733

El KNN se probó con los siguiente valores para K: 3, 5, 7, 9

El random forest se probó con cada combinación de los siguientes parámetros:

Criterion: [“entropy”, “gini”]
N_estimators: [25, 50, 75, 100]
Max_depth: [10, 15, 20]
Max_features: [20, 25, 30]

Los resultados mostrados fueron conseguidos con el clasificador Random Forest con la siguiente configuración:

Criterion = “gini”
N_estimators = 100
Max_depth = 20
Max_features = 25

Análisis: Validación Cruzada

La validación cruzada fue un guía necesario para poner a prueba nuestra selección de features tanto como el clasificador de elección. Desde un comienzo fue mi plan extraer histogramas de colores a porciones de la imagen como features. Mi dilema principal fue decidir entre recortar alrededor del centroide de la etiqueta o si utilizar todo el cuerpo de la botella para calcular el histograma. La primera opción resultó ser más prometedora. Otro dilema que logró resolver la validación cruzada es el de la selección del algoritmo clasificador. Se probaron el KNN y el Random Forest de los cuales, al comienzo, pareció que el KNN era más prometedor. Fue hasta que se comenzó a hacer pruebas de escritorio con diferentes configuraciones del Random Forest que se pudo observar una mejora en rendimiento. La matriz de confusión fue generada al hacer una sumatoria de cada matriz generada en cada iteración del K-Fold. La diagonal de la matriz muestra que tantos aciertos tuvo el clasificador al predecir datos del grupo de entrenamiento con los datos del grupo de prueba.

Sorpresas

Fue algo sorprendente ver como al comienzo (cuando estaba recién comenzando a probar diferentes algoritmos clasificadores en la validación cruzada), casi ninguno logró tener mejores resultados que el algoritmo de K-Nearest Neighbors. Incluso intentando diferentes combinaciones de configuración de parámetros, ninguno logró superarlo. Fue hasta que me di cuenta que las configuraciones que había hecho en otro dataset en una tarea de la clase, no necesariamente tienen que ser las más óptimas para este dataset. Fue entonces cuando probé una configuración recomendada por el Dr. Kenny Davila que me di cuenta de la verdadera capacidad del Random Forest. Superó por mucho al K-Nearest Neighbors y terminó siendo mi algoritmo clasificador seleccionado para este trabajo. Otra sorpresa fue al momento de procesar las imágenes. Al principio, había hecho que las imágenes del tapón y la etiqueta se recortan alrededor del centroide de ejes calculado con el filtro de Canny. De esta manera (pensé yo), que daría mejores resultados ya que podría enfocarse solo en las etiquetas y no en cualquier fondo distractor o alguna botella vacía. Resulta que la validación cruzada mostró mejor rendimiento cuando no se aplica esta técnica. No estoy seguro del porque aunque tengo algunas ideas. Puede ser que no hayan suficientes datos de botellas vacías para que estas distraigan al clasificador. Podría ser también que la diferencia de los histogramas no es tanta entre botellas llenas y vacías ya que ambos resultados son influenciados mayormente por la etiqueta.

Dificultades Encontradas

No hubieron muchas dificultades al desarrollar este proyecto. La más notoria fue al momento de escoger features para entrenar el clasificador. No tengo mucha experiencia con la ciencia de los datos y mucho menos determinando que cuenta como un buen feature más que lo básico: que hace una botella distinta a las demás. Intente distintas técnicas de procesamiento de imágenes para intentar encontrar cuales features serian mas utiles. La forma de la botella quizás hubiera sido útil aunque la única idea que tuve para poder medirla era con un filtro de edge detection. Esto fracasó ya que las diferentes iluminaciones dentro de las imágenes hacen que partes de la botella estuviesen sobreexpuestas y creaba edges innecesarios. El fondo también creaba edges no deseados que causaron un dolor de cabeza al intentar lidiar con ellos. Entonces esta idea se botó.

Planteamiento

No es difícil pensar en las múltiples aplicaciones que tiene el procesamiento de imágenes. Basta con ver lo que la industria de la tecnología ha logrado desarrollar para saber el impacto que esto puede tener hacia nuestro futuro como especie. Aunque lo que se desarrolló en este proyecto no fue de alta complejidad, sus fundamentos son aplicados comúnmente en áreas comerciales e incluso en como base en otros sistemas inteligentes como aquellos basados en el computer vision. Una posible aplicación: reconocimiento facial. La implementación del reconocimiento facial involucra técnicas de procesamiento de imágenes mas complejos de los que fueron aplicados en este trabajo. Sin embargo es posible y ha sido posible desde ya casi décadas. Una posible manera de extraer features faciales podría ser con detección de key points, donde encuentre las esquinas de las cejas, boca, ojos, y cualquier otra parte facial que se distingue entre personas. Otra técnica que podría ser útil es la de la extracción de histograma de colores ya que el tono y el pelo de las personas forman una combinación más a la cual sería útil agregar dentro de la lista de features. El reconocimiento facial se puede ver en casi todos lados y se puede ver como se ha implementado el para varios distintos propósitos. En áreas tanto como de vigilancia, medicina, e incluso en áreas recreativas. Apple tiene en la aplicación de fotos en sus iPhones, una feature que le permite recopilar automáticamente fotografías que contengan a una persona. En otras palabras, organiza las fotos de manera automática dependiendo de quien esté en ellas. La vigilancia es otra área donde el reconocimiento facial está muy bien pronunciado. Podría identificar criminales que hayan sido grabados en una cámara de seguridad. Esto le quitaría el trabajo a varias personas encargadas de vigilar pero es un trabajo tedioso y agotador si se toma en cuenta cuanto tiempo tiene que estar uno vigilando. Incluso el mismísimo gobierno de Honduras comenzó a utilizar reconocimiento facial para la entrega de cédulas este año 2021. Las aplicaciones son muchas sin duda y es un área que vale mucho la pena estudiar.

Conclusiones

Este proyecto fue sin duda uno de los más divertidos que he hecho en mi carrera. La idea de reconocimiento de imágenes siempre ha sido algo increíble y casi de ciencia ficción. Pero ahora que entiendo sus fundamentos, me siento capaz y motivado de indagar más acerca del tema. El procesamiento de imágenes fue la parte más dura sin duda. Fue mucha prueba y error hasta encontrar algo que parecería que iba a funcionar. Pero una vez ya hecha, el resto fue como una caminata cuesta abajo. Aprendí a manejar el API de varias librerías/módulos populares de python y de la ciencia de datos como: numpy, pandas, pickle, json, scikit-learn y opencv. El manejo de recolección y lectura de features se facilitó muchisimos una vez que aprendí las bases sobre cómo utilizar estas APIs. Sin duda explorar mucho más el tema de clasificación tanto como el procesamiento de imágenes y en un futuro no muy lejano, incluso el computer vision.