Procesamiento de imágenes (pre TP1)

Víctor A. Bettachini
Datamining en ciencia y tecnología 2023
Especialización en Explotación de Datos y Descubrimiento del Conocimiento
bettachini@gmail.com

Resumen

Cuca.

1. Introducción

Cuca

2. Materiales y métodos

Datos 210 imágenes de flores acompañados de un listado de las correspondientes especies dentro de una variedad de 10. Las imágenes en formato png tienen una dimensión de 128 x 128 píxeles con tres canales de color. El conjunto se descargó de una fuente pública [**belitskaya_flower_2020**].

Recurso informático Un cuaderno (notebook) Jupyter provisto por los docentes en el sitio web denominado "Campus" [**kamienkowski_curso_2023**] es la plantilla donde se escribió código en lenguaje Python. Este explotó funciones de las bibliotecas OpenCV (`cv2`) y Clustimage para el trabajo con imágenes.

3. Resultados

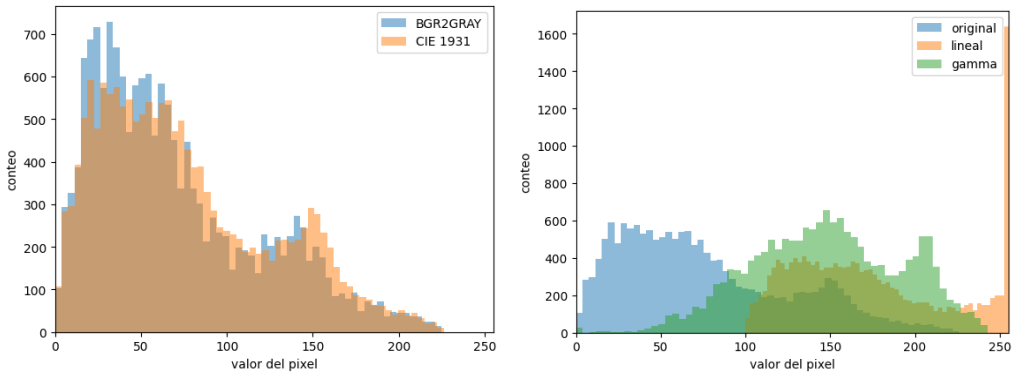
3.1. Preprocesamiento de los datos

Carga del conjunto de datos La función `cv2.imread` carga los canales es azúl, verde, rojo (BGR: blue, green, red). Invertiendo la carga en un array (arreglo de la biblioteca numpy) se los ordena como RGB con la sentencia `cv2.imread(path[0])[...,::-1]`, en este caso para la primer imagen en el directorio al que apunta `path`. La figura 3a muestra en colores naturales la imagen generada a partir del array por la función `imshow` de la biblioteca matplotlib.

3.2. Manipulación de datos

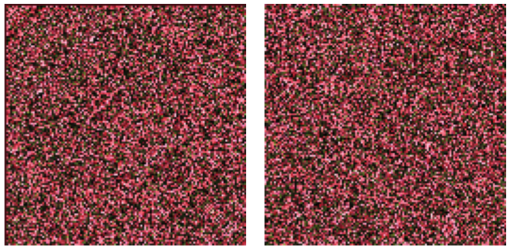
Escala de grises Se utilizó la combinación lineal que preserva la luminancia perceptual de la codificación de color sRGB de la Commission Internationale de l'éclairage en 1931 según el consorcio W3 [**noauthor_standard_1996**] $Y_{\mathrm{lineal}} = 0{,}2126 R_{\mathrm{lineal}} + 0{,}7152 G_{\mathrm{lineal}} + 0{,}0722 B_{\mathrm{lineal}}$. La figura 3b muestra la luminancia obtenida en una escala lineal de grises.

Una alternativa a realizar esto manualmente es recurrir a la función `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`. El producto es similar, como lo muestra el comparar histogamas de ambas alternativas (ver figura figura 1a).



(a) Escala de grises. (b) Incremento brillo.

Figura 1: Comparación de histogramas.



(a) Reordenar índices (b) Mezclar píxeles

Figura 2: Aleatorización de la primer imagen en el conjunto de datos a nivel de píxeles individuales.

Brillo Aplicar una función lineal $Y_{\text{salida}} = \alpha Y_{\text{entrada}} + \beta$, donde α la ganancia controla el contraste y β el sesgo controla el brillo modifica el histograma de luminancia [**noauthor_opencv_nodate**]. Si $beta \gg 0$ hace que $Y_{\text{linealfinal}} > 255$ se les asigna este último valor. Esto produce saturación (ver figura figura 3c). Para evitar esto se utiliza una ley de potencias $Y_{\text{salida}} = 255 \left(\frac{Y_{\text{entrada}}}{255}\right)^{\gamma}$ conocida como corrección gamma por el exponente utilizado. Un $\gamma = ,4$ redundó en la figura 3d, que muestra mayor brillo sin saturación (ver figura figura 1b). Se reduce la luminancia con un $\gamma > 1$, como ejemplifica la figura 3e para $\gamma = 3{,}5$.

Imágen binarizada Se creó una matríz de ceros de la misma dimensión que la de luminancia. A los píxeles con valores por sobre la mediana (`statistics.median`) se les asigno el valor unidad en la nueva matriz que se muestra en la figura 3f

Recorte circular Se generó un arreglo con tres matrices de ceros sobre las que se copiarian píxeles de la imagen original de cumplire una condición en función de un radio r del cuatro del número de columnas. La condición define un círculo centrado con $(i - i_0)^2 + (j - j_0)^2 < r^2$ siendo (i, j) e (i_0, j_0) filas y columnas del arreglo y sus valores centrales respectivamente. Aplicando la condición en los tres canales de color se obtuvo el viñetado con un círculo de diámetro mitad del ancho de la imágen que muestra la figura 3g.

Aleatorizar imagen Aleatorizar índices de píxeles a Copiar píxeles de la imagen original con `imgAzar[i,j,c] = img[ indices[i*j][0], indices[i*j]` dentro de dos ciclos for anidados siendo (i, j) los de una lista aleatorizada produjo una imagen que visualmente muestra tener estructura (ver figura 2a.

La figura figura 2b muestra en este aspecto un mejor resultado. Para lograrle se pasaron las valores para cada color en tres vectores y cuyos ordenemientos fueron simultáneamente cambiados al azar. Luego se reconstruyeron las tres matrices en el orden original.

(a) Original (b) Solo luminancia (c) Saturada (d) Abrillantada

(e) Oscurecida (f) Binarizada (g) Viñetado

Figura 3: Procesamientos de la primer imagen en el conjunto de datos

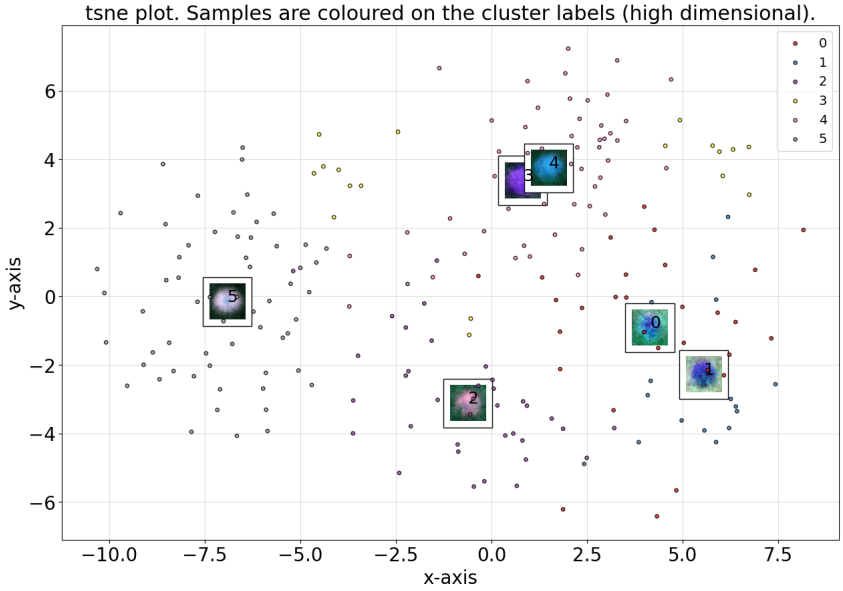


Figura 4: Ubicación de cada imágen en componentes principales(tsne plot)

3.3. Búsqueda de *features*

Análisis de componentes principales Una centena de componentes principales por imagen se obtuvieron con el método `exctract_feat` [**taskesen_pca_2020**].

4. Discusión