Compresion de imagenes

Este ejercicio trata sobre compresión de imágenes a color. Considera una imagen en color como la que se muestra en la Figura 0.14 (derecha). El objetivo es reducir el tamaño (en Kb) de la imagen, tratando de mantener un balance entre tamaño y calidad de la misma, y para esto, utilizarás dos métodos.

a) **k-means**: En este caso, se simplifica la imagen identificando k grupos de colores en los píxeles de la imagen, y posteriormente, se asigna cada píxel a su grupo de color correspondiente. Para esto, considera cada píxel $\mathbf{x}_i \in \mathbb{N}^3$, es decir, como un punto en el espacio RGB. Tu matriz de datos será entonces $\mathbf{X} \in \mathbb{N}^{(h \times w) \times 3}$, donde w y h son el ancho y la altura de la imagen, respectivamente. Los pasos se describen en el Algoritmo 1.

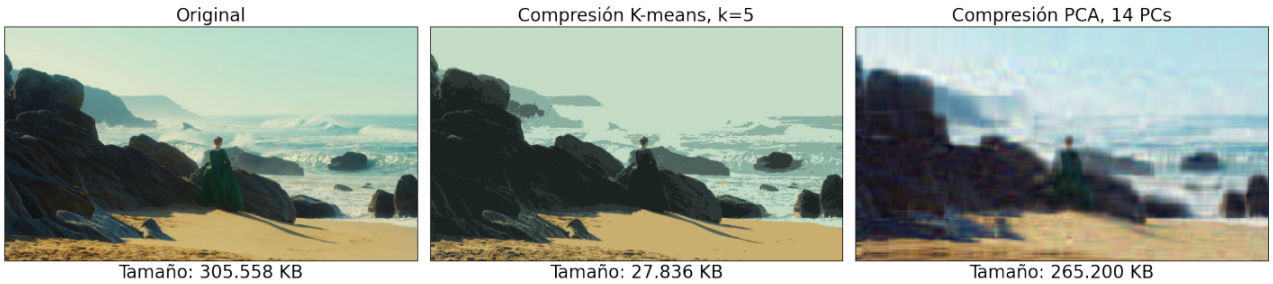

Figura 0.1: Compresión de imágenes. Izquierda: imagen original. Centro: compresión obtenida con k-means usando $k = 5$ colores. Derecha: compresión obtenida con PCA usando 14 componentes principales.

Algorithm 1 Compresión de imágenes 1

1: **Input:** imagen RGB $\mathbf{I}_{h \times w}$, número de clústers k
2: Obtén la matriz de datos $\mathbf{X} \in \mathbb{N}^{(h \times w) \times 3}$
3: Realizar k-means y obtener la asignación del clúster $C(i)$ para cada píxel $\mathbf{x}_i$, así como los centroides $\boldsymbol{\mu}_k$ de cada clúster.
4: Asignar los valores RGB de cada píxel, de acuerdo a su clúster correspondiente, es decir: $x_i^{(k)}$
5: Reconstruir la imagen comprimida $\tilde{\mathbf{I}}$ con su tamaño original

b) **Componentes principales**: En este método, realizarás la compresión simplificando la imagen mediante algunos componentes principales obtenidos en cada canal de color de la imagen. En este caso, considera representaciones de la forma $X_{\text{channel}} \in \mathbb{N}^{h \times w}$, y el procedimiento se describe en el Algoritmo 2.

Algorithm 2 Compresión de imágenes 2

1: **Input:** imagen RGB $\mathbf{I}_{h\times w}$, número de componentes p o varianza acumulada v.
2: Obtener las matrices de datos por cada canal X_R, X_G y X_B cada una del tamaño de la imagen original.
3: Realizar PCA en cada canal, para reducir la dimensión de cada canal de la imagen según los p primeros componentes principales, o equivalentemente, según los p componentes principales que acumulan una varianza v.
4: Sea $\tilde{X}h \times p$ los scores obtenidos con PCA, $Vw \times p$ los componentes principales, y $\boldsymbol{\mu}_w$ el vector de promedios de la imagen original. Reconstruir cada canal de la imagen con su descomposición PCA correspondiente mediante

$$\tilde{\mathbf{I}}_{\text{channel}} = \tilde{X}V' + \mathbf{M},$$

donde $\mathbf{M}$ contiene el vector de medias $\boldsymbol{\mu}$ por renglones.
5: Obtener la imagen final comprimida mediante $\tilde{\mathbf{I}} = [\tilde{\mathbf{I}}_R\, \tilde{\mathbf{I}}_G\, \tilde{\mathbf{I}}_B]$.

Para cada inciso, verifica el resultado con distintos valores de k y p, reportando también el tamaño en Kb de la imagen. Escribe tus conclusiones de este ejercicio donde consideres el balance entre compresión y calidad de la imagen. ¿Qué método prefieres y por qué? ¿Qué nivel de compresión te parece adecuado? ¿Qué criterio (cuantitativo) se te ocurre para evaluar la compresión de la imagen original?Detalles de implementación:
En el Ejercicio 4, puedes calcular el tamaño de la imagen con la siguiente función:

```
from io import BytesIO
def imageSize(img):
    img_file = BytesIO()
    image = Image.fromarray(np.uint8(img))
    image.save(img_file, 'png')
    return img_file.tell()/1024
```

donde `img` es una imagen RGB en forma de tensor, cuyas entradas son enteros en un rango de 0 a 255.Todos los elementos para la reconstrucción de la imagen del paso 4 del Algoritmo 2, puedes obtenerlos de la clase PCA del módulo `sklearn.decomposition`. Alternativamente, puedes usar el método `inverse_transform()` del mismo módulo.

S O L U C I O N E S

Figura 0.2: Imagen original para comprimir

Utilizaremos esta imagen para poder llevar acabo nuestro experimento siguiente, la imagen corresponde a una imagen de una guacamaya, el tamaño original de la imagen es de 210.22 KB

INCISO A.

Primero vamos a abordar el problema de compresión de imagenes con con k- means. La compresión de imágenes utilizando el algoritmo k-means implica la reducción del número de colores en la imagen original. La imagen se representa como una matriz de píxeles, donde cada píxel está compuesto por valores de intensidad de color. En una imagen RGB, cada píxel tiene tres valores que representan los componentes Rojo, Verde y Azul (Red, Green, Blue).
Antes de aplicar k-means, la imagen se convierte en un formato que se puede manipular fácilmente. Por lo general, se convierte en una matriz NumPy para operaciones eficientes. El algoritmo k-means se utiliza para agrupar los píxeles de la imagen en 'k' clusters. El objetivo es encontrar 'k' centroides que minimicen la distancia entre los puntos de datos (píxeles) y los centroides de los clusters a los que pertenecen. En este contexto, los píxeles se agrupan según su similitud en términos de color. Después de ejecutar k-means, cada píxel de la imagen se asigna al centroide más cercano (cluster) en términos de distancia euclidiana en el espacio de color. Esto significa que en la imagen comprimida, los píxeles comparten el mismo color que el centroide al que están asignados.
Se realizan pruebas con diferentes k, que son el número de cluster a utilizar, y para ello podemos observar los resultados siguientes:

Figura 0.3: K-means con con k = 2

Figura 0.4: K-means con con k = 5

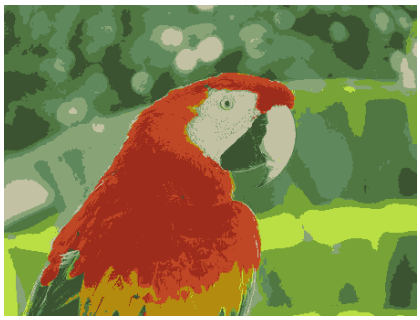

Figura 0.5: K-means con con k = 10

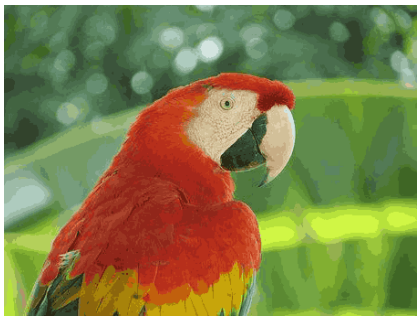

Figura 0.6: K-means con con k = 50.

1. Figura 4.3, para esta parte podemos notar que con solo dos cluster no rescatamos mucha información de la imagen original ya que solo nos representa dos colores significativos.

2. Figura 4.4, notamos una mejora significativa en que podemos apreciar más cosas dentro de la imagen, pero la calidad sigue siendo mala

3. Figura 4.5, notamos aquí que los pesos con respecto a las anteriores imágenes no cambia mucho y aun podemos seguir agregando mas clusters sin afectar nuestro peso

4. En la figura 4.6 vemos que ya la imagen tiene una buena calidad pero el peso es muy alto con respecto a las anteriores.

Podemos definir como una imagen ideal comprimida según su valor de tamaño y la calidad que demuestra es aquella con k=20 clusters Para justificar lo anterior podemos verlo en una

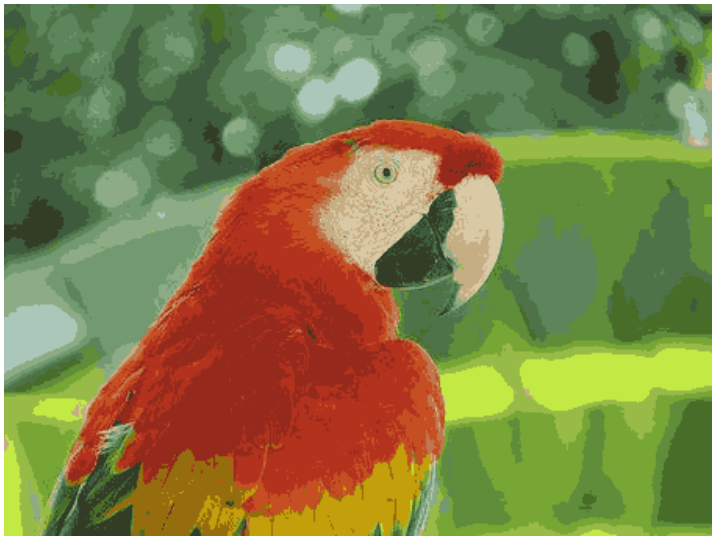

Figura 0.7: Mejor balance entre tamaño calidad de la imagen, con k = 20 y tamaño de imagen del 36.52 KB

grafíca creada co610630n diferentes clusters, en donde mostramos el tamaño de la imagen obtenida según cuantos cluster se usaban. Si bien en la figura 4.8 podemos observar que el

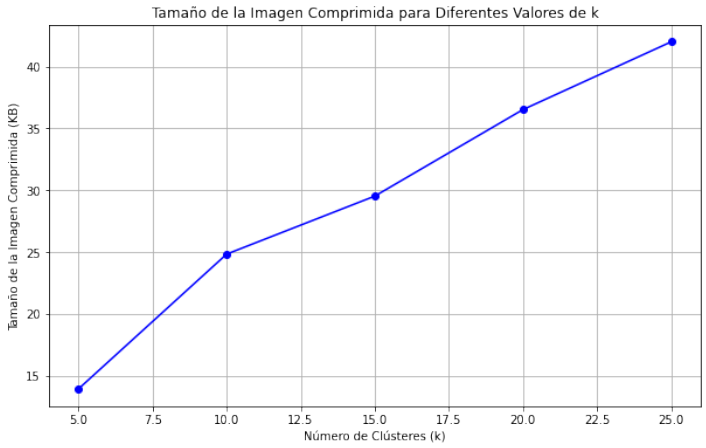

Figura 0.8: Comportamiento entre tamaño de la imagen con diferentes valores de k

tamaño de la imagen crece rapidamente dependiendo el número de k que se elija notamos que estamos en una zona en la cual no afectamos tanto nuestro tamaño y además de que la figura nos muestra una buena calidad.

INCISO B

La función implementa la compresión de imágenes utilizando el análisis de componentes principales (PCA) en cada canal de color (Rojo, Verde, Azul). La imagen original se divide en tres canales de color: Rojo (R), Verde (G) y Azul (B).

Se aplica PCA por separado a cada uno de los canales de color. El parámetro p controla el número de componentes principales retenidas en la descomposición. Estas componentes representan las direcciones principales de variación en cada canal. Los datos en cada canal se transforman utilizando las componentes principales obtenidas mediante PCA. Luego, los datos transformados se utilizan para reconstruir la información original, incorporando únicamente las componentes principales seleccionadas durante la transformación.

Los canales reconstruidos se combinan para formar la imagen comprimida final. La compresión se logra al reducir el número de componentes principales y, por lo tanto, la información almacenada en cada canal. En resumen, el método utiliza PCA para reducir la dimensionalidad de cada canal de color por separado, preservando las principales características de la imagen. Cuando se ajusta el parametro p, puedes controlar el grado de compresión y, en consecuencia, la calidad de la imagen resultante.

Al igual que en el inciso anterior se realizaron pruebas con diferentes valores de p, resultando lo siguiente

Figura 0.9: PCA con con p = 10

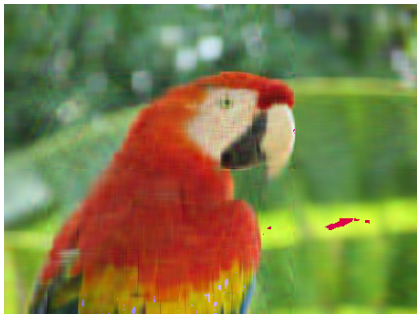

Figura 0.10: PCA con con p = 20

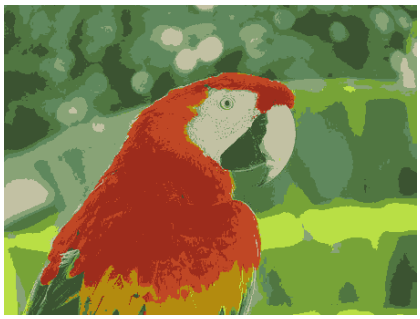

Figura 0.11: PCA con con p = 40

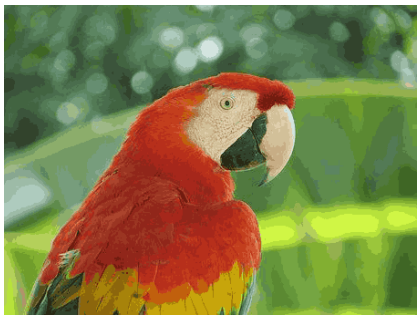

Figura 0.12: PCA con con p = 70.

De estás figuras presentadas anteriormente podemos analizar la siguiente situación:

1. La imágenes comprimidas con p menos que 20 como lo son las figuras 4.9 y 4.10 no logran tener una buena calidad de la iamgen independientemente del tamaño de la misma
2. Podemos notar que la estrategia primero es poder ver las calidades de la imagen, como la calida va mejorando y definir el punto que queremos poner como referencia de calidad.
3. Con p=40 como se muestra en la figura 4.11 podemos notar que ya mejoramos la calidad

de la imagen bastante bien por lo que podemos pensar que nos acercamos a un buen candidato

Con el analisis anterior podemos definir una buena calidad de nuestra imagen y un buen nivel de compresión de tamaño a la siguiente

De nuestra figura 4.13 podemos concluir que la calidad es baja pero no queremos afectar

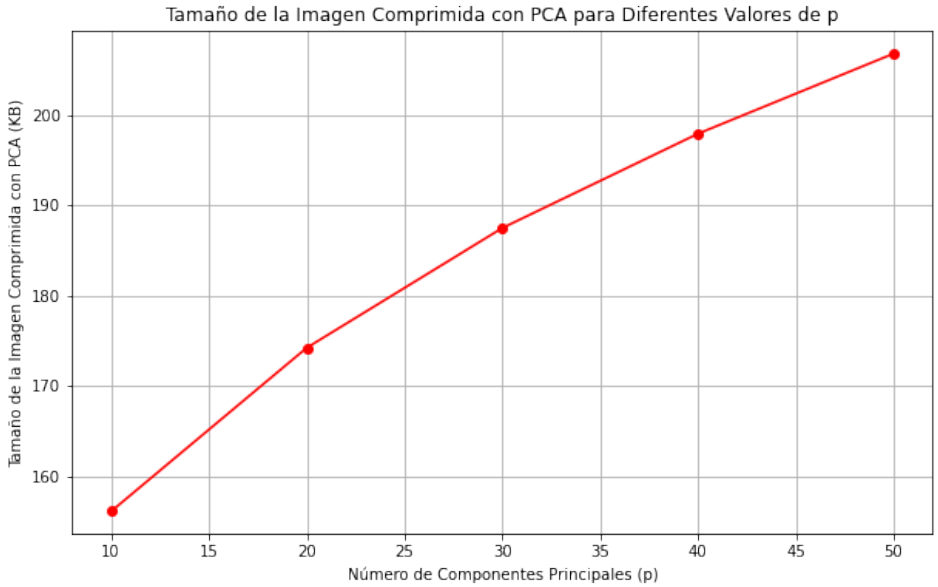

Figura 0.13: Grafico para ver el comportamiento del tamaño de la imagen a lo largo de diferentes p

tanto el tamaño de la imagen por lo que elegimos la siguiente opción.

Constestando a las preguntas

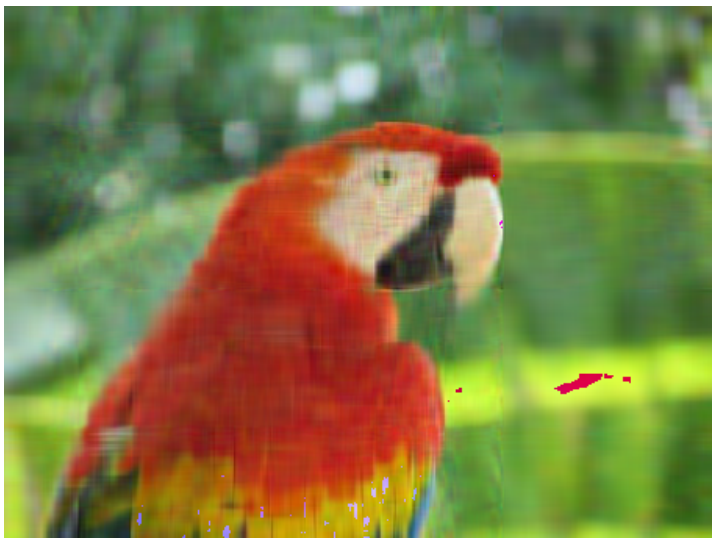

Figura 0.14: Eleccion de imagen manteniendo la buena calidad y un tamaño de compresión bueno, con p = 20

La elección entre el método de compresión utilizando k-means y el método que utiliza PCA depende de tus necesidades específicas y las características de las imágenes que estás compri-

miendo.

Método de K-means:

Ventajas:

Simple y fácil de implementar.

No asume una estructura lineal en los datos.

Puede proporcionar una compresión visualmente atractiva al agrupar píxeles de colores similares.

Desventajas:

Puede no funcionar tan bien en imágenes con variaciones más sutiles en el color.

No tiene en cuenta la correlación entre píxeles vecinos.

Método de PCA:

Ventajas:

Captura la variación más importante en los datos, lo que puede ser eficiente para la compresión.

Puede funcionar bien en imágenes con patrones lineales y variaciones sutiles.

Desventajas:

Puede perder información importante si se selecciona un número muy bajo de componentes principales (p bajo).

Asume una estructura lineal en los datos.

Nivel de Compresión Adecuado:

El nivel de compresión adecuado depende de tus requisitos específicos. Un nivel más alto de compresión (menos colores o componentes principales) resultará en una pérdida de detalles, pero también en archivos de imagen más pequeños. Un nivel bajo de compresión conservará más detalles, pero generará archivos más grandes.

Criterio Cuantitativo para Evaluar la Compresión:

Para evaluar cuantitativamente la compresión, puedes utilizar métricas como el error cuadrático medio (MSE) o el índice de similitud estructural (SSI). Estas métricas comparan la imagen comprimida con la imagen original y proporcionan una medida objetiva de la calidad de la compresión. Un valor bajo de MSE o un alto índice de similitud estructural indicará una mejor calidad de compresión.

En última instancia, la elección entre los métodos y el nivel de compresión adecuado dependerá de tus prioridades específicas, como el equilibrio entre el tamaño del archivo y la calidad visual de la imagen comprimida. También puedes experimentar con diferentes configuraciones y métricas para encontrar la combinación que mejor se adapte a tus necesidades.