Trabajo práctico 1
Compresión de Imágenes

1. Introducción

La compresión de imágenes constituye una herramienta esencial dentro del procesamiento digital, ya que permite reducir la cantidad de datos necesarios para su almacenamiento o transmisión sin comprometer de manera significativa la calidad. Esta característica resulta fundamental para facilitar la gestión de grandes volúmenes de datos en aplicaciones como la fotografía digital, streaming, etc. En este trabajo práctico exploramos la técnica del Análisis de Componentes Principales (PCA) como estrategia de compresión de imágenes, destacando su capacidad para concentrar y preservar la información más relevante de la imagen reduciendo el volumen de datos necesario para representarla.

1.1. Imagen digital

Una imagen digital en escala de grises[1] es esencialmente una matriz de números en 2D, como se ve en las Figuras 1 y 2, donde cada elemento representa el brillo de un píxel, cuantificado con diferentes niveles de intensidad (256 niveles si se codifica cada píxel con 8 bits).

Figura 1: Imagen en escala de grises.

$$\begin{bmatrix}
128 & 128 & 160 & 128 & 128 & 96 & 128 & 128 & 128 & 128 \\
128 & 160 & 128 & 64 & 32 & 0 & 0 & 192 & 160 & 128 \\
160 & 160 & 32 & 32 & 0 & 0 & 32 & 64 & 128 & 160 \\
192 & 64 & 128 & 255 & 255 & 255 & 192 & 96 & 64 & 192 \\
160 & 96 & 192 & 255 & 192 & 192 & 255 & 96 & 64 & 192 \\
160 & 64 & 160 & 160 & 192 & 192 & 128 & 64 & 0 & 160 \\
160 & 64 & 160 & 0 & 160 & 128 & 0 & 32 & 0 & 128 \\
192 & 160 & 128 & 255 & 192 & 128 & 160 & 64 & 64 & 255 \\
160 & 192 & 96 & 255 & 160 & 64 & 128 & 32 & 192 & 255 \\
160 & 192 & 128 & 192 & 192 & 96 & 96 & 64 & 255 & 192 \\
160 & 192 & 160 & 64 & 128 & 64 & 32 & 96 & 255 & 192 \\
160 & 192 & 255 & 0 & 0 & 0 & 96 & 128 & 255 & 25
\end{bmatrix}$$

Figura 2: Matriz asociada a la Figura 1 .

1.2. Compresión de imágenes

La mayoría de las imágenes no son simples patrones de intensidad arbitrarios, sino que cada imagen que vemos contiene algún tipo de estructura, lo que implica que existe cierta correlación entre los píxeles vecinos. El objetivo de comprimir una imagen es almacenarla de una forma más compacta, es decir, que se requiera una menor cantidad de datos en memoria para su codificación, y que a la vez no se afecte apreciablemente su calidad. Esto es posible si se tiene en cuenta que una imagen en su forma "sin procesar" contiene un alto grado de información

[1] En una imagen a color se utilizan tres matrices, una por cada color primario RGB (Red, Green, Blue).

redundante. De este modo, las técnicas de compresión buscan eliminar esas redundancias de modo tal que se codifique la mayor parte de la información con una menor cantidad de bits, para lo cual existen diferentes métodos. En el presente trabajo nos enfocaremos en la técnica de PCA, con la que se podrá reducir la dimensión de los datos. Cabe aclarar que en la práctica se utilizan normalmente otros métodos, como los basados en transformada coseno (DCT) para el formato JPEG [3] que resulta computacionalmente más eficiente, aunque con una calidad de compresión menos óptima que con PCA.

1.3. Covarianza de una imagen

Para entender como aplicar la compresión con este método, es necesario primero definir una matriz de covarianza que represente la variabilidad de la imagen. En este caso, el vector aleatorio asociado debe representar una vecindad de cierta cantidad de píxeles. Suponiendo que una imagen se segmenta en bloques de 8×8 píxeles (que es lo habitual para compresión de imágenes), como se muestra en la Figura 3 (a), se define el vector $\mathbf{X} \in \mathbb{R}^{64\times 1}$ para representar el conjunto de píxeles vecinos luego de estirar el bloque, como se muestra en la Figura 3 (b). Con esta definición, tomaremos todos los bloques extraídos de una imagen como realizaciones particulares del vector aleatorio $\mathbf{X}$. De esta forma, la matriz de covarianza $C_{\mathbf{X}} = E\left[(\mathbf{X}-\boldsymbol{\mu}_{\mathbf{X}})(\mathbf{X}-\boldsymbol{\mu}_{\mathbf{X}})^T\right]$ dará una idea de cuánta variabilidad tiene la imagen en relación a píxeles vecinos para un área del tamaño de un bloque. Asumiendo que la imagen queda segmentada en L bloques totales[2], se podrán utilizar esas mismas realizaciones para estimar tanto el vector de medias $\widehat{\boldsymbol{\mu}}_{\mathbf{x}}$ como la matriz $\widehat{C}_{\mathbf{x}}$, que serán posteriormente utilizadas para la compresión con PCA.

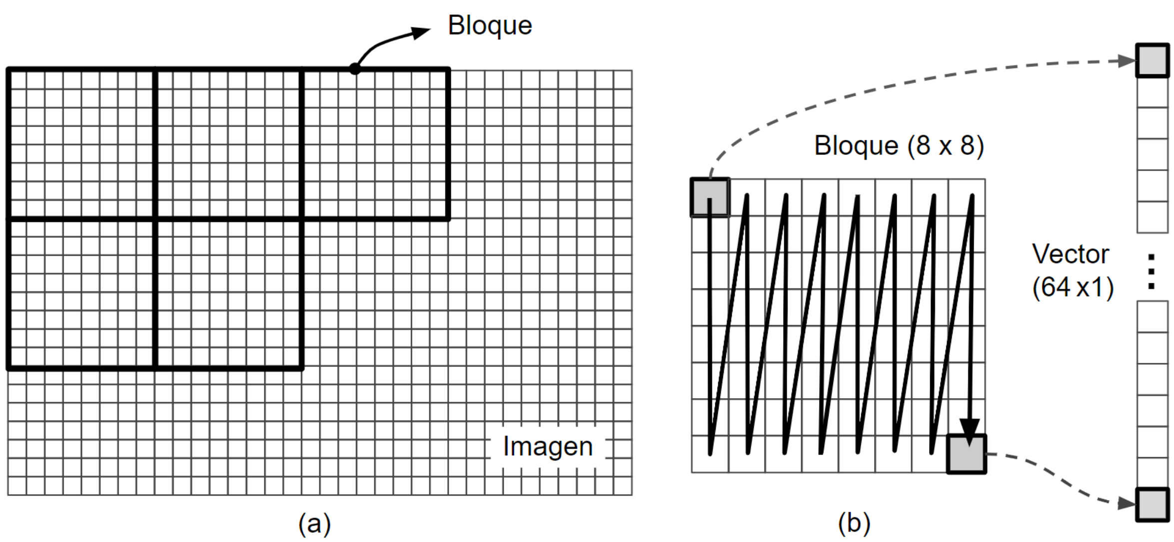

Figura 3: (a) Segmentación de la matriz de una imagen en bloques (submatrices) de 8×8. (b) Armado de una realización $\mathbf{x}_i$, con $i = \{1, ..., L\}$, del vector aleatorio $\mathbf{X}$.

[2]Es claro que no siempre habrá una cantidad L entera de bloques que particionen exactamente una imagen completa. Por lo tanto, la imagen deberá ser previamente recortada para cumplir con una segmentación adecuada.

1.4. PCA aplicado a la compresión de imágenes

El método de compresión de imágenes con PCA consiste en hallar una transformación lineal reversible para eliminar la redundancia presente en la imagen, almacenando solo los datos que conservan la mayor parte de la información. Lo que se busca es eliminar los elementos de menor varianza, asociados a los píxeles vecinos que presenten cambios menos perceptibles. Como se definió en la sección anterior, de la misma imagen se puede estimar una matriz de covarianza C_X, cuyos autovectores definen la matriz de proyección del método PCA [1]. De esta forma, se pueden transformar las realizaciones $\mathbf{x}_i \in \mathbb{R}^{64\times 1}$, con $i = 1, ..., L$, en vectores de menor dimensión $\mathbf{y}_i \in \mathbb{R}^{k\times 1}$, con $k < 64$. Cabe destacar que esta transformación resulta óptima en el sentido de menor error cuadrático, según se define en [2][3]. En resumen, el procedimiento para la compresión y descompresión de imágenes con esta técnica, sigue los siguientes pasos:

Compresión

1. Segmentar la imagen en bloques de 8×8 para definir las realizaciones $\mathbf{x}_i \in \mathbb{R}^{64\times 1}$.
2. Aplicar PCA para obtener las proyecciónes $\mathbf{y}_i \in \mathbb{R}^{k\times 1}$ de menor dimensión.
3. Guardar la colección de vectores $\mathbf{y}_i$, la matriz de autovectores y la media $\boldsymbol{\mu}_X$.

Descompresión

1. Abrir la colección de vectores $\mathbf{y}_i$, la matriz de autovectores y la media $\boldsymbol{\mu}_X$.
2. Aplicar la proyección inversa de PCA para la reconstrucción aproximada de los vectores aleatorios originales.
3. Convertir los vectores reconstruidos en bloques de 8×8 y recomponer con ellos nuevamente la imagen,.

1.5. Métricas

Algunas medidas para evaluar el desempeño de la compresión son el error cuadrático medio (MSE), Ec. (1), donde p_{ij} y $\hat{p}_{ij}$ representan el valor del píxel original y reconstruido, respectivamente, en la posición ij, siendo N_w y N_h la cantidad de píxeles de ancho y de alto de la imagen, respectivamente. Por otro lado, se puede cuantificar el porcentaje de *espacio ahorrado* o *Space Saving* (S [%]) definido como el porcentaje de memoria ahorrada[4] luego de la compresión, Ec. (2), siendo $m = 64$ para el presente caso.

$$MSE = \frac{1}{N_w\,N_h}\sum_{i=1}^{N_w}\sum_{j=1}^{N_h}(p_{ij} - \hat{p}_{ij})^2 \tag{1}$$

$$S = \left(1 - \frac{\text{cantidad de componentes principales (k)}}{\text{cantidad de componentes totales } (m)}\right) \times 100\,\% \tag{2}$$

[3]En el contexto de imágenes se suele denominar KLT (Karhunen–Loève Transform) a la transformación para compresión de imágenes que coincide con PCA, pero justificado desde la optimización que minimiza el MSE.

[4]Si bien debería tenerse en cuenta también la cantidad de memoria que requiere guardar el vector de medias y la matriz de proyección, lo omitimos ya que nos interesa una comparación relativa y no absoluta del ahorro.

2. Ejercicios

Ejercicio 1: Correlación

El objetivo de este ejercicio es descomponer la matriz de imagen en bloques de solo dos píxeles con el propósito de poder ver gráficamente en $\mathbb{R}^2$ la correlación entre píxeles vecinos.

(a) Cargar las imágenes suministradas `img_01.jpg` y `img_02.jpg` (convertirlas a escala de grises y a tipo `float`). Formar bloques de 2×1 para definir el vector $\mathbf{X} = [X_1 \ X_2]^T$ (con X_1 y X_2 dos píxeles contiguos verticalmente). Hacer un gráfico de dispersión, junto a la imagen asociada, para ver gráficamente cuánta correlación existe entre los dos píxeles vecinos para cada imagen.

(b) Estimar el coeficiente de correlación de cada vector. ¿Qué diferencias observa para ambas imágenes? ¿Cuál de ellas podría comprimir la información con una mayor calidad?

(c) Aplicar una transformación que permita desacoplar las variables de cada vector, haga un gráfico de dispersión de ese nuevo vector y saque conclusiones.

Ejercicio 2: Compresión

Para realizar el proceso de compresión será necesario segmentar la imagen en bloques para luego aplicar la reducción de dimensionalidad mediante PCA.

(a) Implementar la función `pca_transform()` que el algoritmo de PCA, ya sea basado en la descomposición de la covarianza o mediante SVD. Nota: no utilice paquetes con funciones que ya resuelvan el algoritmo de PCA.

(b) Desarmar la imagen `img_03.jpg` en bloques de 8×8 y aplicar PCA para obtener los vectores de menor dimensión, tal que el ahorro sea $S = 80\,\%$. Haga un gráfico de los autovalores de C_X, diferenciando los que se conservan de los que se descartan.

Ejercicio 3: Descompresión

(a) En base a los datos del Ejercicio anterior, implementar el proceso inverso, teniendo en cuenta la transformación que revierte la compresión y el vector de medias. Regenerar la imagen a partir de los vectores reconstruidos.

(b) Graficar con la imagen luego de la reconstrucción y compararla con la original.

Ejercicio 4: Medidas de desempeño

(a) Abrir la imagen `img_04.jpg`, calcular y graficar MSE en función de diferentes porcentajes de espacio ahorrado $S = \{5n : n = 1, 2, ..., 19\}$.

(b) Graficar la imagen original y todas las versiones comprimidas y reconstruidas para los casos $S = \{75,\ 80,\ 85,\ 90,\ 95\}$.

3. Conclusiones

Como conclusiones, elabore un resumen breve y conciso comentando características que considere relevantes del método propuesto en este trabajo y los resultados obtenidos, así como dificultades encontradas y cómo fueron abordadas.

4. Condiciones de entrega

- **Informe**: El informe debe entregarse en formato PDF (**no se aceptarán otros formatos**) y con nombre: `TP1_GXX.PDF` (donde `XX` es el número de grupo). No debe incluirse código fuente en el PDF. El código de Python debe presentarse en un archivo aparte.
- **Código**: El código debe incluirse junto al informe en un archivo ZIP (con mismo nombre que el informe) que deberá subirse al campus.
- Se recuerda a los estudiantes que las entregas deben ser un producto original de cada estudiante, por lo que se les pide revisar la sección 6 del programa de la materia y el Código de Honor y Ética de la Universidad.

Referencias

[1] Steven M. KAy. Intuitive probability random processes using MATLAB. Springer 1951.

[2] Ed. K. R. Rao and P.C. Yip. Boca Raton. The Transform and Data Compression Handbook, CRC Press LLC, 2001.

[3] CCITT. Information technology - Digital compression and coding of continuous-tone still images - requirements and guidelines. The international telegraph and telephone consultative committee. T.81, 1992.