DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Procesamiendo de Imágenes, análisis de componentes principales y reducción de la dimensionalidad

Métodos Numéricos - Grupo 5

Integrante	LU	Correo electrónico
Leandro Higa	1444/21	leanhiga12@gmail.com
Kenet Chapetón	682/19	kenetchape@gmail.com
Macarena Lopez Bianco	268/18	maca_lopezbianco@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Resumen

En este trabajo práctico utilizaremos un conjunto de imagenes de rostros y realizaremos reducción de dimensionalidad de los datos mediante análisis de componentes principales. Esta representación sirve para hacer compresión de una imagen al descartar información que tiene poco impacto en la estructura de la imagen. Este permite comparar imagenes y realizar análisis de similaridad buscando poder agrupar las imagenes por individuo. Estudiaremos dos método de reducción de dimensiones: Análisis de Componentes Principales (PCA) y una variante especial para tratar imagenes 2DPCA.

Palabras clave

PCA, 2DPCA, Eigenfaces, Matriz de similaridad

1. Introducción

El procesamiento de imagenes es un campo relacionado al procesamiento de señales y el reconocimiento de patrones donde se trabaja con conjuntos de datos de imagenes, es decir, matrices cuyos elementos codifican los colores de los píxeles.
En este trabajo práctico utilizaremos un conjunto de imagenes de rostros y realizaremos reducción de dimensionalidad de los datos mediante análisis de componentes principales. Se cuenta con un conjunto de imagenes de rostros de N personas en escala de grises, todas con el mismo tamaño y resolución. A su vez, para cada una de las personas se cuentan con M imagenes distintas. Las imagenes a considerar se encuentran en un espacio de dimensión alta, buscamos reducir la cantidad de dimensiones de las muestras para trabajar con una cantidad de variables más acotada y, simultáneamente, buscando que las nuevas variables tengan información representativa para diferenciar los objetos de la base de entrada. Con este objetivo, estudiaremos dos métodos de reducción de dimansiones: PCA y 2DPCA.

2. Método de la potencia con deflación

Dada una matriz $A \in \mathbb{R}^{mxn}$, un autovector de A es un vector x no nulo de $\mathbb{R}^n$ tal que $Ax = \lambda x$, para algun escalar λ. Llamamos a λ autovalor de A.

Si nuestra matriz A tiene un autovalor dominante, es decir, existe λ_1 autovalor de A tal que $|\lambda_1| > |\lambda_i|$, $\forall \lambda_i$ autovalor de A, entonces podemos usar el método de la potencia para aproximar el valor de λ_1 y de v_1, su autovector asociado.

Implementamos entonces el método de la potencia en C++ de la siguiente manera:

Algorithm 1 $power\_iteration(A, niter, eps)$

```
v = creamos un vector aleatorio del tamaño R^n;
for j = 0; j < niter; j++ do
    v_0 = v;
    v = A * v/norm(A * v);
    a = v^t * A * v;
    if criterio(v, v_0, eps) then
        break;
    end if
end for
res = make_pair(a, v);
return res;
```

Con los siguiente criterios de parada:

Algorithm 2 $criterio1(v, v_0, eps)$

```
cos_ang = v.dot(v_0);
res = (1 - eps) < cos_ang <= 1;
return res;
```

Por la identidad del coseno. sabemos que dados v_1, $v_2 \in \mathbf{R}^n$
se cumple que:

$$< v_1, v_2 >= cos(\alpha) * ||v_2|| * ||v_1||$$

donde α es el angulo entre ambos vectores, por lo que si cos_ang esta cerca de 1 podemos decir ambos vectores son parecidos, ya que α (el angulo entre ellos) estara cerca de 0.

Entonces este criterio usa el angula entre el vector previo, y luego de aplicar Av, para ver que si se cumple una tolerancia pedida, pare el algoritmo.

Algorithm 3 $criterio2(v, v_0, eps)$

```
return fabs(v - v_0)/fabs(v) < eps
```

Usamos el error relativo ya que sin saber nada sobre el autovector dominante, es el mejor criterio simple que podemos optar, por ejemplo si usaramos el error $||v_0 - v||$, puede ocurrir que nos topemos con una sucesion que no sea convergente pero las diferencias si lo hagan. Por lo que tambien concideramos este criterio.

Si nuestra matriz tiene n autovalores distintos $|\lambda_n| < |\lambda_{n-1}| < ... < |\lambda_1|$ y una base ortonormal de autovectores $v_1, ..., v_n$, entonces podemos calcular todos sus autovalores y autovectores con el método de deflación.

Algorithm 4 $calcular\_autovalores(A, num, niter, eps)$

```
autovalores = creamos un vector de tamaño num;
autovectores = creamos una matriz de tamaño num x num;
for i = 0; i < num; i++ do
    calculo = power_iteration(A, niter, eps);
    λ = calculo.first;
    vec = calculo.second;
    autovalores(i) = λ;
    autovectores.col(i) = vec;
    A = A − λ * vec * vec^t;
end for
res = make_pair(eigenvalues, eigenvectors);
return res;
```

Este algorítmo calcula los primeros num autovalores de A y sus autovectores asociados.

Cuando aplicamos PCA o 2DPCA a conjuntos de datos de imágenes, como la matriz de covariancia es simétrica semidefinida positiva, podemos aplicar el método de la potencia con deflación para calcular sus autovalores y autovectores, ya que podemos suponer que los autovalores difieren de forma que numericamente el algoritmo no de errores, pues es improbable que se parezcan mucho.

Primero veamos que A_N es simetrica para todo N, ya A lo era desde un comienzo

$$A_{N+1}^t = (A_N - \lambda_N v_N * v_N^t)^t$$
$$A_N^t - \lambda_N (v_N * v_N^t)^t = A_N^t - \lambda_N v_N * v_N^t$$

Como para N = 1, A es por hipotesis simetrica, con un argumento inductivo tenemos que A_N $(\forall N)$

Como A es simetrica existe una base de autovectores ortonormales B donde $|A|_B$ es diagonal. Supongamos que B esta ordenada de mayor a menor en funcion de los autovalores asociados a los autovectores.

Veamos que estos vectores en las iteraciones que vamos haciendo no van cambiando, pues estamos cambiando la matriz en cada iteracion, y esto puede ocurrir

Sea v_{N+1}, el siguiente autovector de la base B de A_N :

$$A_{N+1} * v_{N+1} = (A_N - \lambda_N v_N * v_N^t) v_{N+1}$$
$$A_N * v_{N+1} - \lambda_N v_N * v_N^t * v_{N+1} = \lambda_{N+1} * v_{N+1} - \lambda_N v_N * v_N^t * v_{N+1}$$

como es una base ortonomal tenemos que $v_N^t * v_{N+1} = 0$ luego

$$\lambda_{N+1} * v_{N+1} = A_N * vN + 1$$

Por lo tanto podemos asegurar que el metodo de deflacion va a ir calculando el autovector y autovalor correspondiende.

3. Procesamiento de imágenes

El procesamiento de imágenes es un campo relacionado al procesamiento de señales y el reconocimiento de patrones. Se trabaja a la imágen como una matriz cuyos elementos codifican los colores de los píxeles.

En este trabajo práctico utilizaremos un conjunto de imágenes de rostros y realizaremos reducción de dimensionalidad de los datos mediante análisis de componentes principales.

3.1. PCA

El método PCA consiste en cambiar la base del conjunto de datos de entrada a una base ortogonal donde las dimensiones individuales de los datos se ordenen en componentes que expliquen la varianza en los datos de mayor a menor. Al hacerlo se pierde información del mismo, quedándose con las componentes más importantes. El método consiste en diagonalizar la matriz de covarianza de los datos.

Sea $x_i \in \mathbb{R}^n$ la i-ésima imagen de nuestra base de datos. Podemos apilar todas las imágenes como filas de una matrix $X \in \mathbb{R}^{mtimesn}$. Sea $\mu \in \mathbb{R}^m$ un vector con el promedio de cada columna de X entonces definimos una matriz centrada $X_c \in \mathbb{R}^{m \times n}$ con filas $(x_i - \mu)^t$. Entonces, la matriz de covarianza $C \in \mathbb{R}^{n \times n}$ se define como $C = X_c^t X_c/(n-1)$.

Vamos a buscar una base que diagonalice la covarianza. Para eso vamos a buscar los autovalores y autovectores de la matriz C. Como C es simétrica, sabemos que es diagonalizable $C = VDV^t$, donde V es una matriz con los autovectores en las columnas y D es una matriz diagonal con los autovalores, ordenados de mayor a menor en módulo.

Entonces, es posible reducir la dimensionalidad de la imagen x_i, tomando el vector $z_i = (v_1^t x_i, v_2^t x_i, ..., v_k^t x_i) \in \mathbb{R}^k$. Este proceso corresponde a proyectar la imagen sobre las k primeras componentes principales. Este factor k es el que determina la calidad de este análisis, así que va a ser relevante para el estudio determinar un valor adecuado.

En la figura 1 podemos ver el valor de los autovalores de la matriz de covarianza. El primer autovalor está cerca del 6 y rápidamente decrecen a 0.

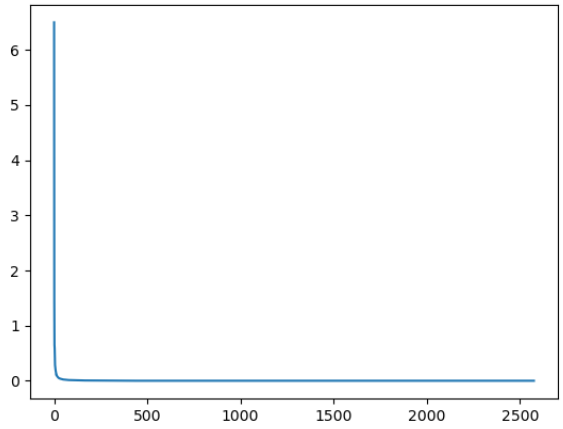

Figura 1: Autovalores de la matriz C

Cada uno de los autovectores se puede redimensionar a la resolución de las imágenes, y todos ellos representan a una 'eigenface'. A partir de la combinación lineal de los autovectores se puede reconstruir cualquiera de las imágenes. En la figura 2 se pueden ver las primeras 10 'eigenfaces'.

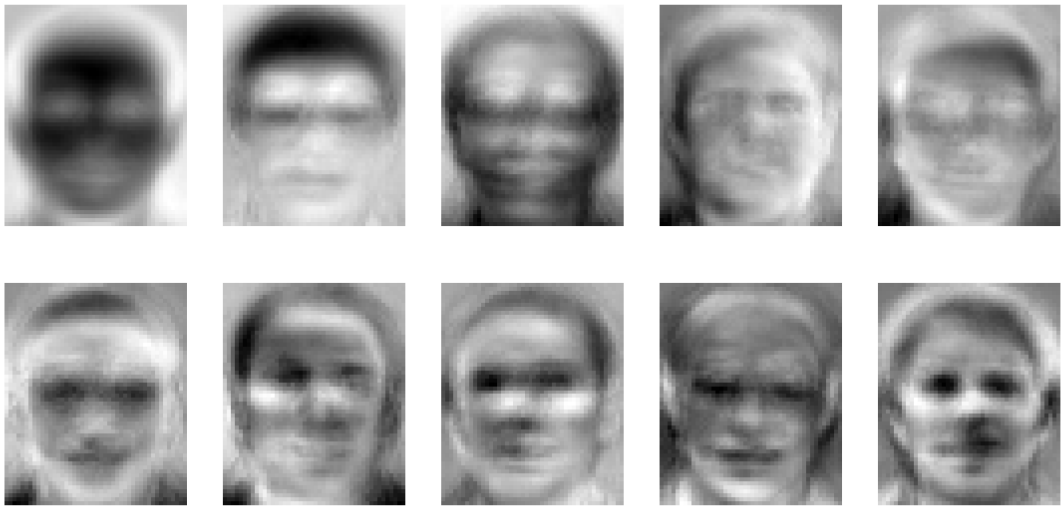

Figura 2: Eigenfaces en PCA

Veamos qué ocurre si tomamos una imagen y regeneramos el rostro tomando las k primeras componentes principales en un rango entre 20 y 50. Cuando aplicamos PCA a una imagen y seleccionamos las primeras k componentes principales, estamos manteniendo solo la información más significativa de la imágen.

En la figura 3 se pueden ver las imágenes regeneradas tomando k para algunos valores entre 20 y 500.

Podemos observar que, si tomamos un número bajo de componentes principales, la imagen regenerada será una aproximación muy simplificada de la imagen original. A medida que aumentamos el número de componentes principales, la imagen regenerada se acercará cada vez más a la original. Pero, una vez que tomamos más de 400 componentes, podemos ver que la imágen comienza a tener ruidoz pierde fidelidad respecto de la imagen original. En la figura 3 se pueden ver las primeras 10 'eigenfaces'.

Figura 3: Reconstrucción para distintos valores de k

3.2. 2DPCA

En PCA cada imagen se transformaba en vector para construir la matriz de datos X y a esa matriz se le computa la covarianza y autovectores. El cálculo de autovalores y autovectores resulta muy costoso para esa matriz.

La variante 2DPCA conserva a las imágen en su dimensión original como una matriz $A \in \mathbb{R}^{a \times b}$ y calculamos la denominada image covariance matrix

$$G = \frac{1}{n} \sum_{j=1}^{n} (A_j - \bar{A})^T (A_j - \bar{A})$$

donde A_j es la j-ésima imagen de nuestro conjunto y $\bar{A}$ siendo la imagen promedio. El tamaño de esta matriz resulta ser menor que la matriz de covarianza de PCA ($G \in \mathbb{R}^{b \times b}$).

A G le calculamos los autovectores y obtenemos la matriz $U = [X_1, \ldots, X_b]$ la base de autovectores ortonormales de G entonces $V = AU$, donde $V = [Y_1, \ldots, Y_b]$. A Y se lo denomina feature vector, asociado a la imagen A mediante la transformación lineal $Y = AX$ siendo $X \in \mathbb{R}^b$ el vector que maximiza la dispersión de los feature vectors.

En la figura 4 podemos ver el valor de los autovalores de la matriz G. A diferencia de PCA, los primeros autovalores son mayores y se aproximan con menor rapidez a 0.

Anteriormente vimos que $V = AU$ y por lo tanto $A = VU^T$ por ser U ortogonal. A su vez, como el producto de matrices puede descomponerse como suma de productos externos:

$$A = VU^T = \sum_{j=1}^{b} Y_j X_j^T$$

donde cada $Y_j X_j^T$ es una subimagen de A. De manera similar a PCA, se puede recrear cualquier imagen como combinación de estas.

En la figura 5 podemos ver las primeras 10 subimágenes de A.

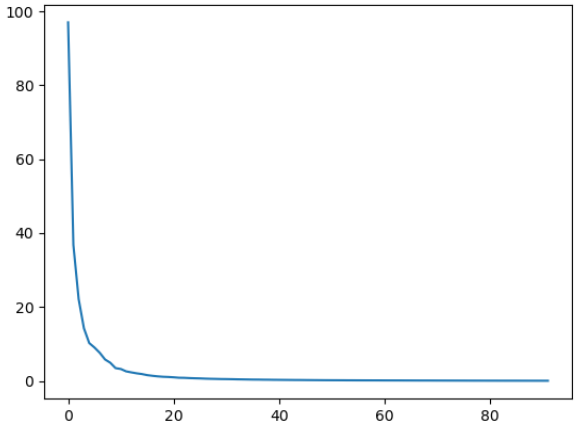

Figura 4: Autovalores de G

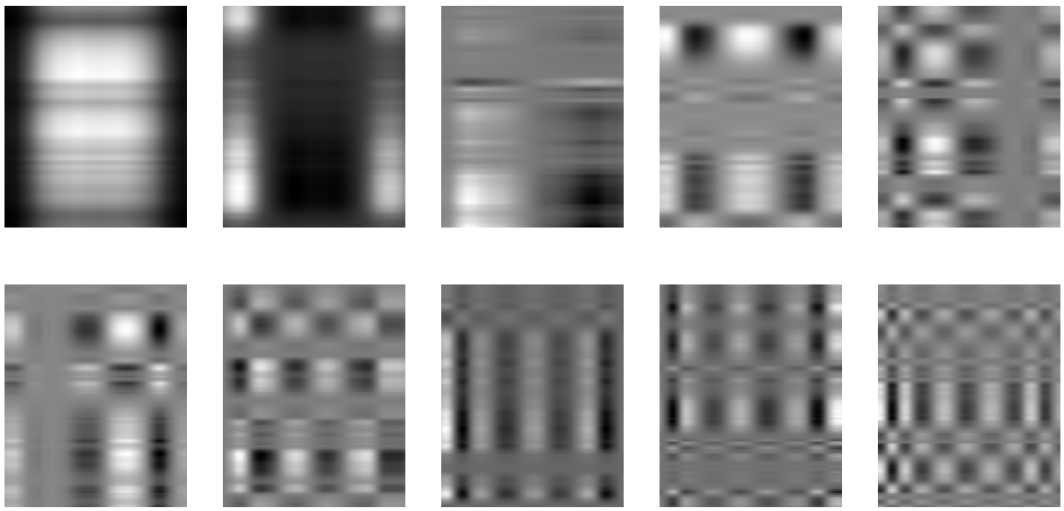

Figura 5: Subimágenes de A

Al igual que en el método de PCA, en general se suelen considerar k componentes principales de manera de poder reconstruir aproximadamente la imagen original A a partir de k subimágenes.

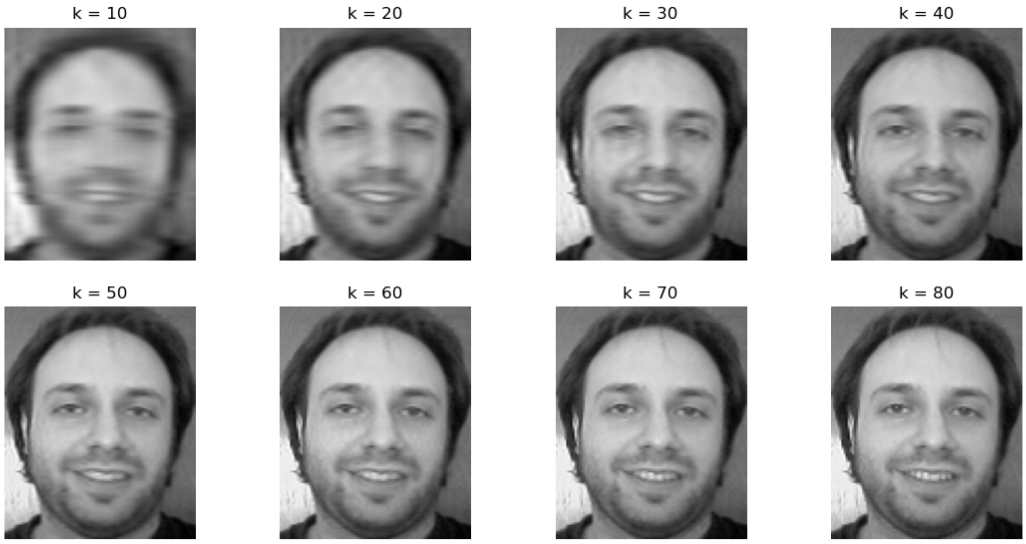

Figura 6: Reconstrucción para distintos valores de k

4. Análisis de similaridad

Dado un conjunto de datos $X \in \mathbb{R}^{m \times n}$ con m datos y n atributos, definimos la matriz de similaridad como la matriz cuadrada D de mxm, donde cada D_{ij} calcula la similaridad entre X_i y X_j.

La similaridad que vamos a utilizar viene dada por la matriz de correlación R, calculada a partir de la matriz de covarianza C. Obtenemos C mediante la fórmula:

$C = (X_c X_c^t)/(n-1)$, donde X_c es la matriz centrada de X.

Luego, calculamos la matriz R utilizando la siguiente fórmula para cada elemento R_{ij}:

$R_{ij} = C_{ij}/\sqrt{(C_{ii} * C_{jj})}$

En la figura 7, calculamos la matriz de similaridad para el conjunto de imagenes original centrado. Como podemos observar, cuando comparamos imágenes de una misma persona, tiene una similitud alta, por lo que se forman en la antidiagonal de la matriz cuadrados de $10x10$ con un color más oscuro. Esto significa que las imágenes de una misma persona son muy similares entre sí. De todos modos, en algunos casos esto no ocurre, como por ejemplo con el sujeto 29.

Entre diferentes personas, la mayoría de las imágenes tienen una similitud más baja. Esto implica que las imágenes de diferentes personas son menos similares entre sí.

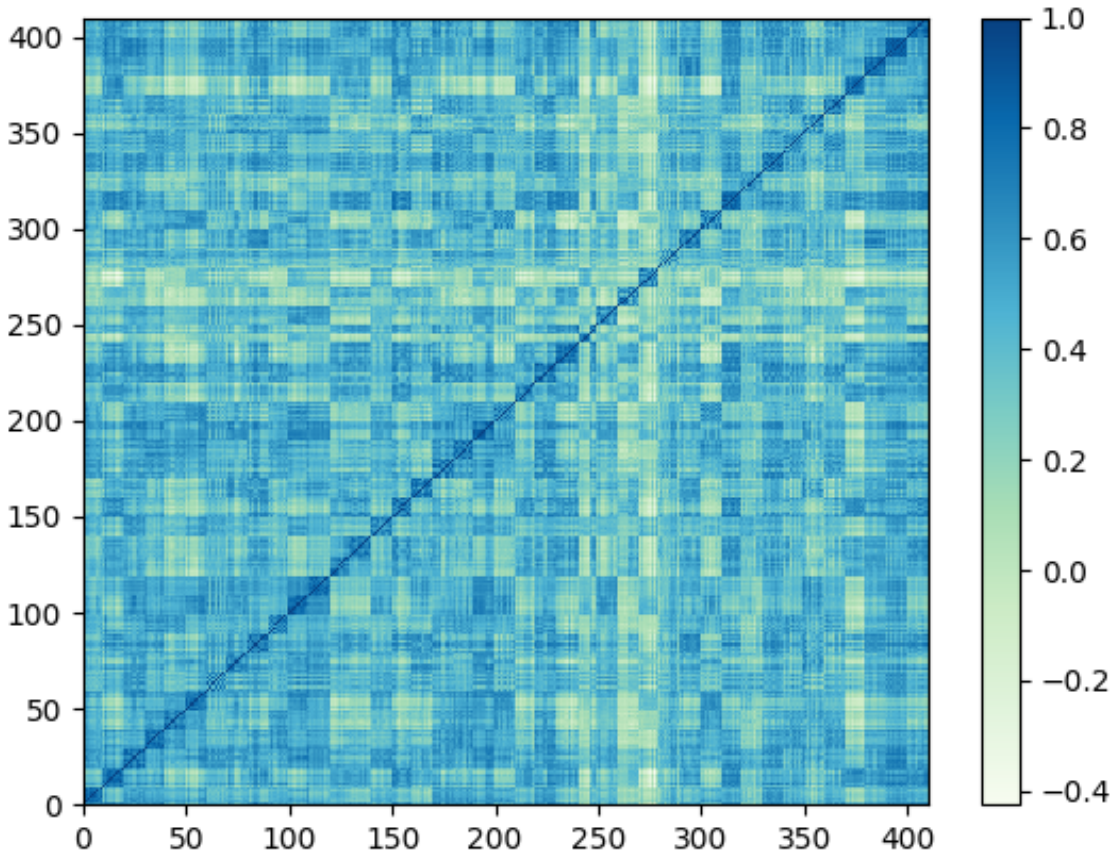

Figura 7: Similaridad para el conjunto de imagenes original centrado

Calculamos ahora la matriz de similaridad para los datos en el espacio z luego de aplicar PCA O 2DPCA a las imágenes. En la figura 8, calculamos la matriz luego de aplicar PCA con $k = 10$ y en la figura 9, calculamos la matriz luego de aplicar 2DPCA con $k = 3$. Como nos estamos quedando con pocas componentes principales en ambos casos, las imágenes se diferencian menos unas de otras, sobre todo cuando comparamos imágenes de diferentes sujetos.

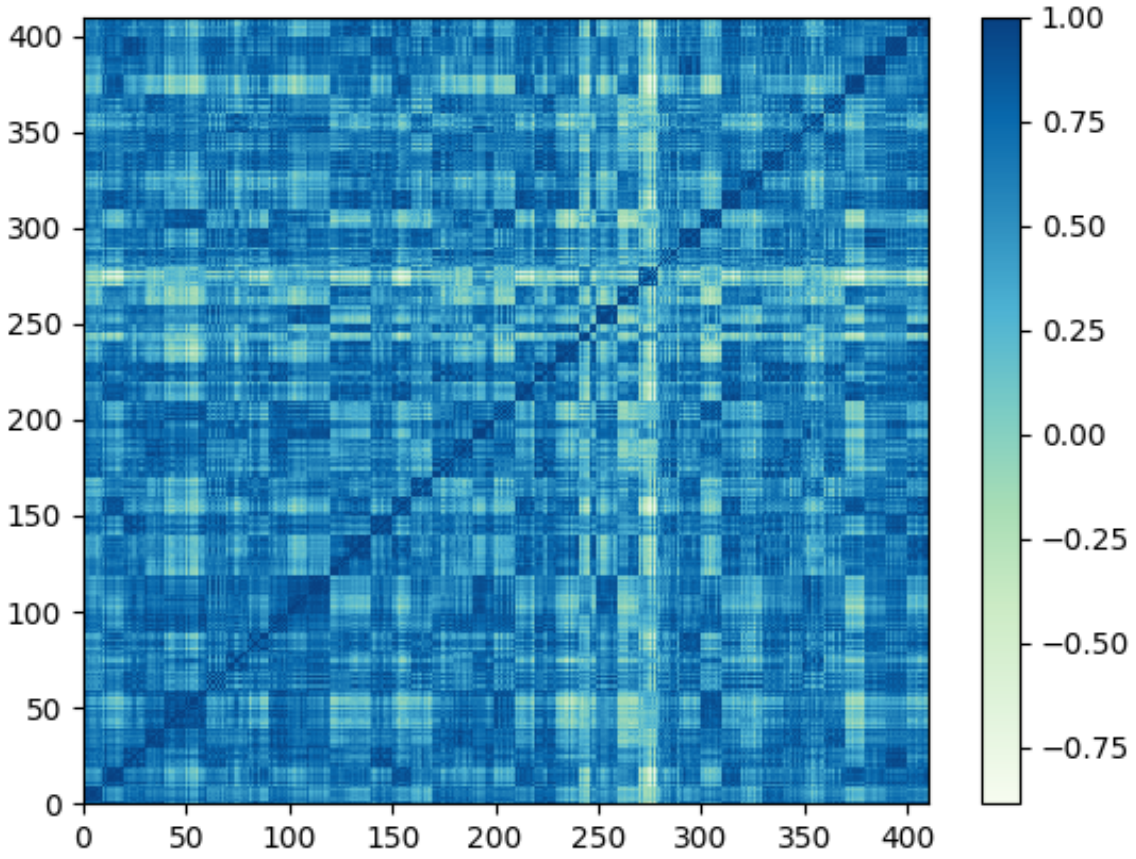

Figura 8: Similaridad para PCA, con k chico

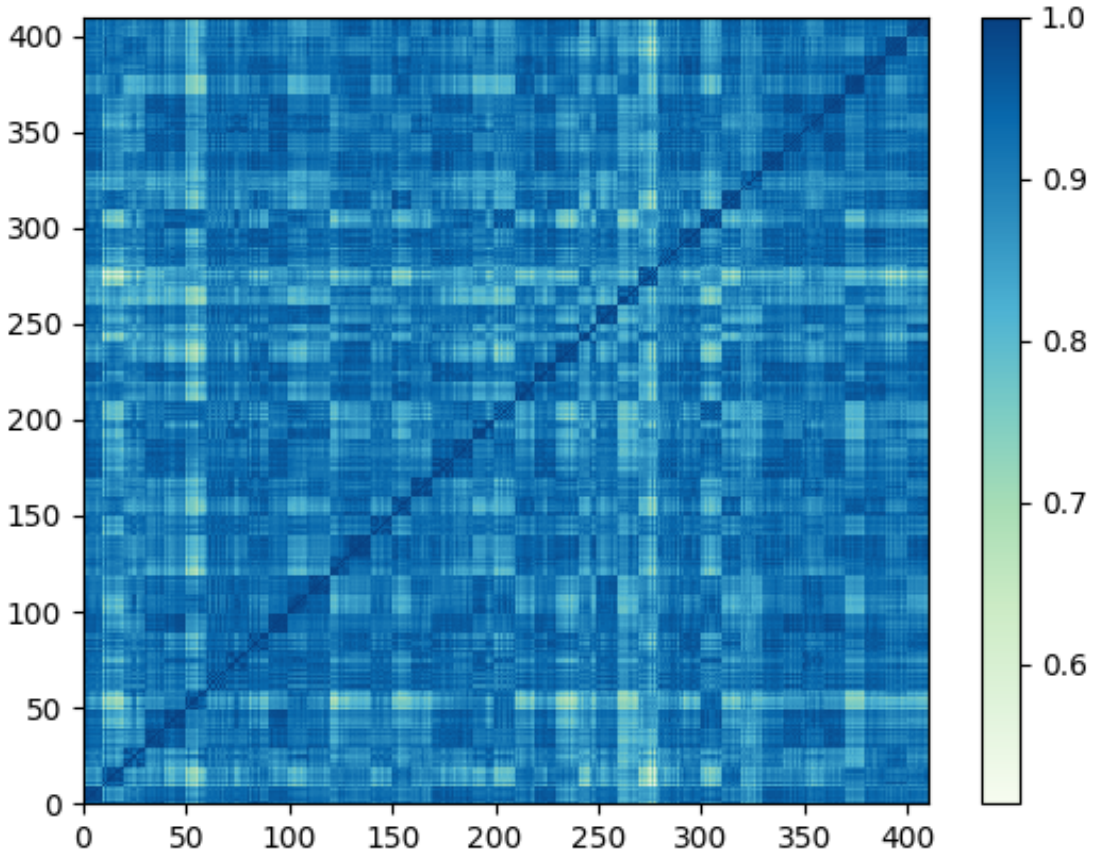

Figura 9: Similaridad para 2DPCA, con k chico

En la figura 10, calculamos la matriz luego de aplicar PCA con $k = 400$ y en la figura 11, calculamos la matriz luego de aplicar 2DPCA con $k = 90$. Como nos estamos quedando más componentes principales, tenemos imágenes con más detalles. Podemos notar entonces más diferencias entre las imágenes de diferentes sujetos.

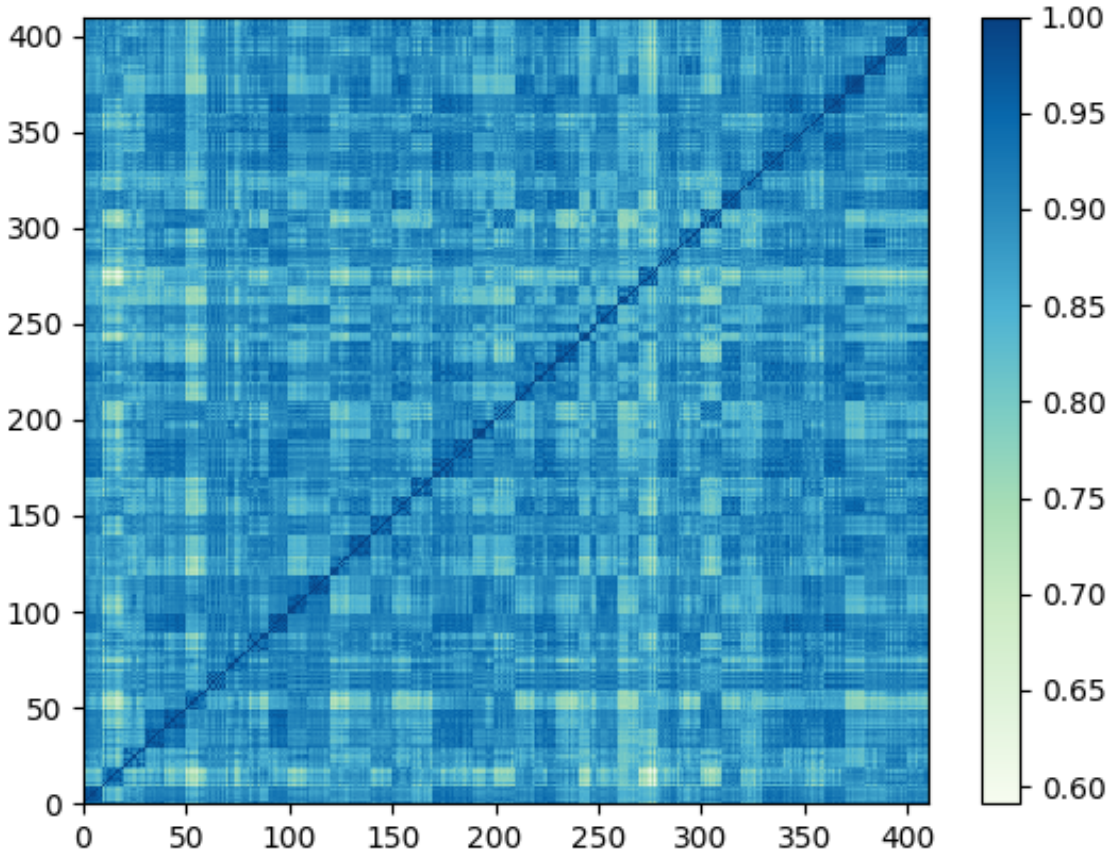

Figura 10: Similaridad para PCA, con k grande

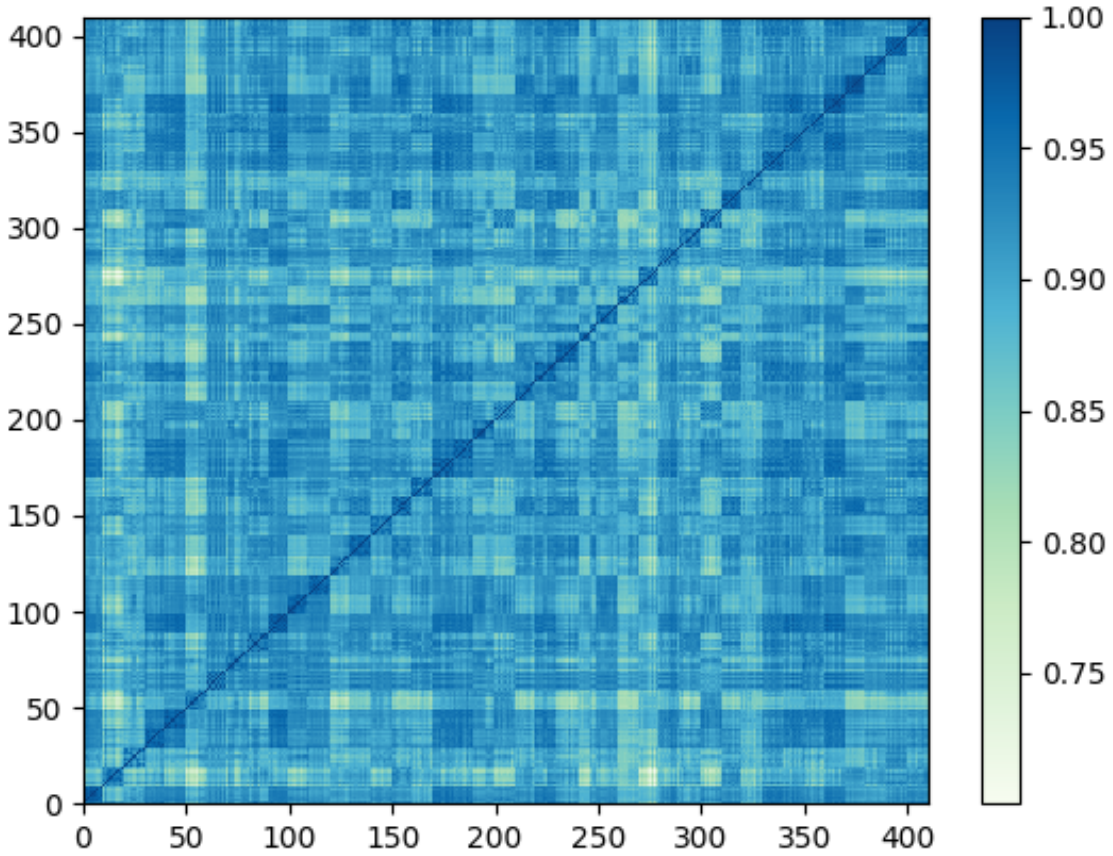

Figura 11: Similaridad para 2DPCA, con k grande

Luego de calcular las matrices de similitud, comparamos la calidad de la similaridad entre imágenes de una misma persona y entre imágenes de distintas personas. Definimos entonces la "Métrica mismo", que toma el promedio de los valores de similaridad para las comparaciones entre imágenes de una misma persona, y la "Métrica distintos", que hace lo mismo pero para todas las comparaciones de imágenes de diferentes personas.

Como podemos observar en la figura 12, el comportamiento de la calidad de similaridad luego de aplicar PCA en función de k es inesperado. Esperábamos que el valor de similaridad a medida que aumentamos el valor de k decrezca por lo menos hasta llegar a $k = 400$, porque como ya vimos en ese valor se alcanza una buena reconstrucción de las imágenes. En cambio, lo que ocurre es que a partir del valor $k = 20$ las métricas de similaridad crecen ambas mucho. Esto se puede deber a un error en el código.

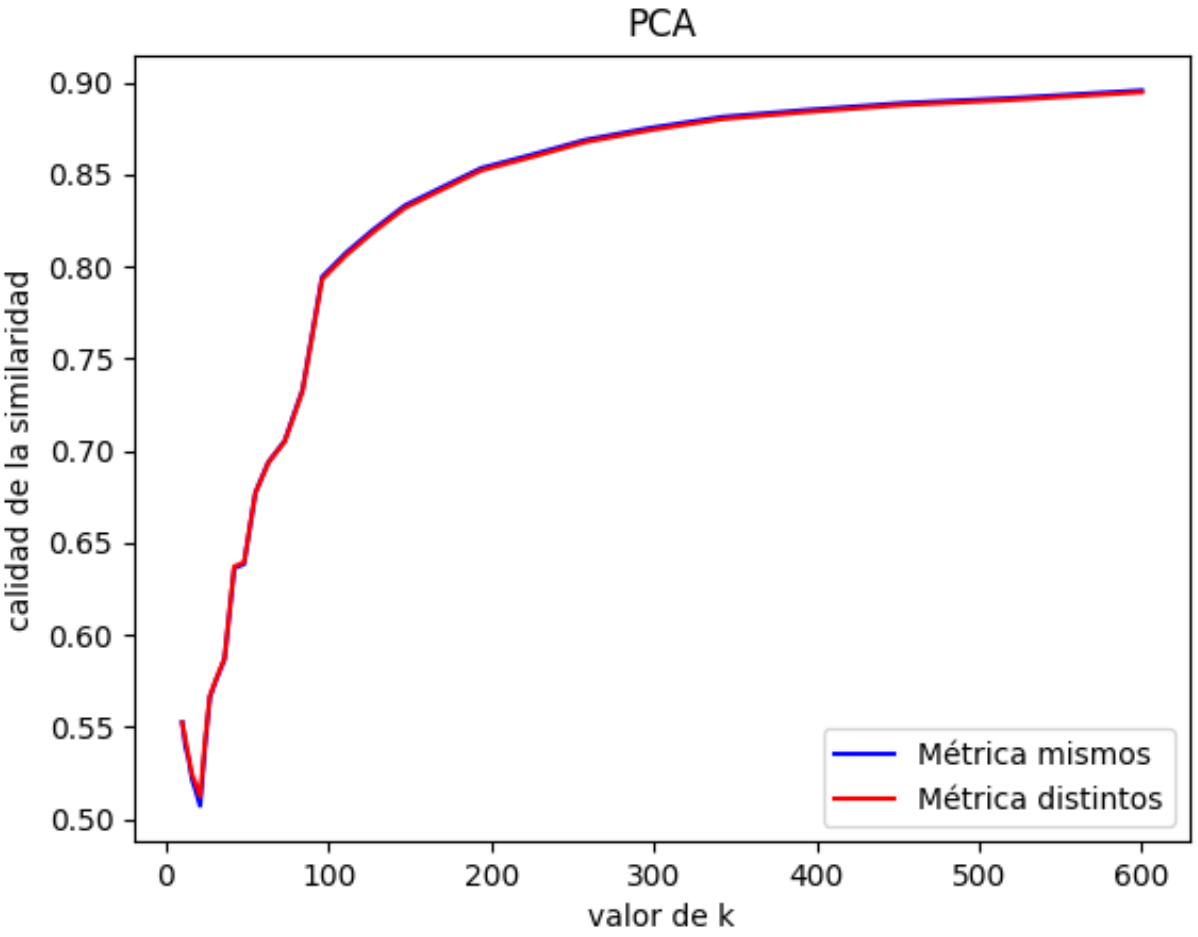

Figura 12: Métricas para PCA en función de k

En cambio, en la figura 13, el comportamiento de las métricas de similaridad luego de aplicar 2DPCA es el esperado. A medida que aumentamos el valor de k, ambas métricas decrece, lo que se debe a que estamos tomando más "detalles"de las imágenes.

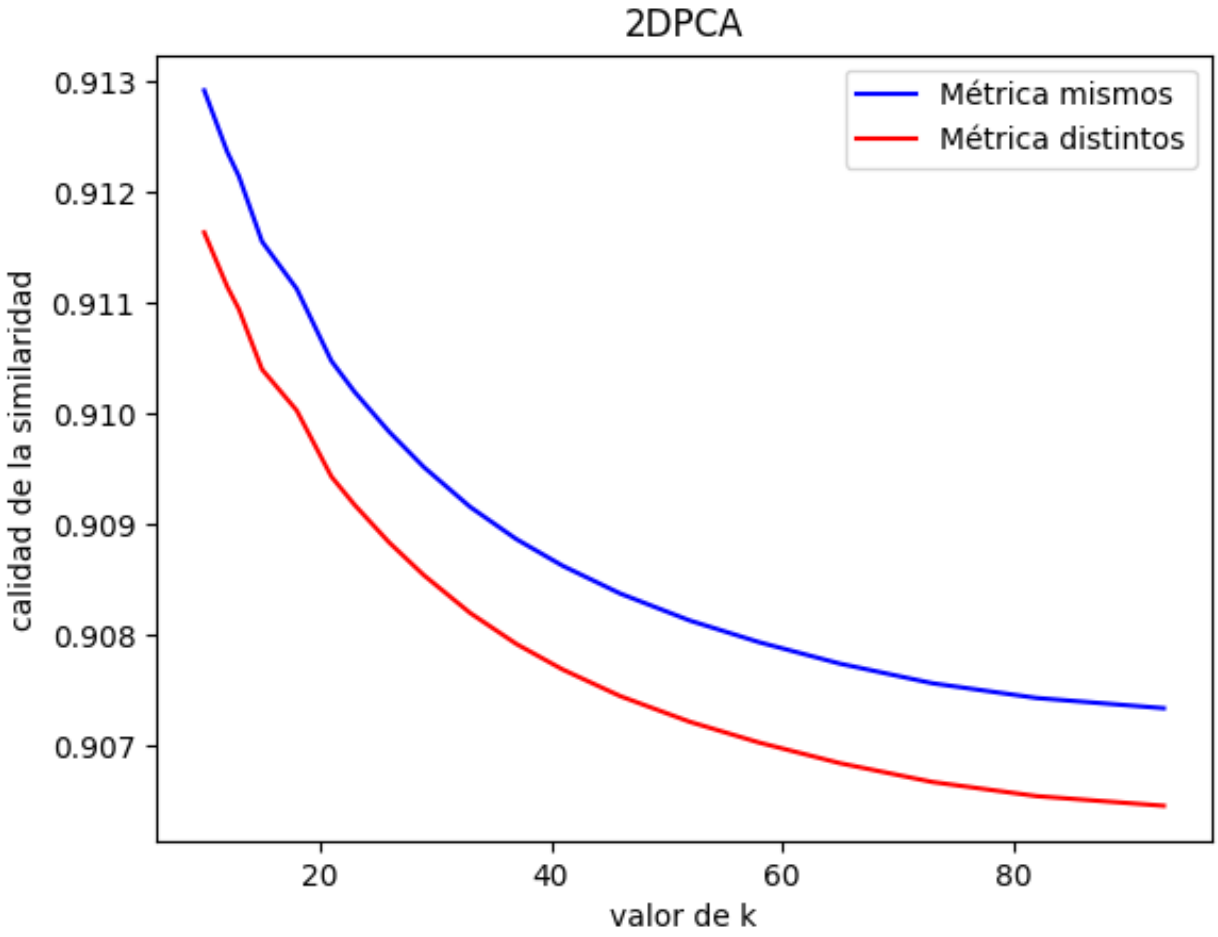

Figura 13: Métricas para 2DPCA en función de k

Para analizar la calidad de las imágenes comprimidas, calculamos el error absoluto medio entre la imágen original y la imágen comprimida luego de aplicarle PCA o 2DPCA. Esta métrica se calcula tomando la diferencia absoluta entre los valores de píxeles de las dos imágenes y luego tomando el promedio.

Tomando $k = 400$ para PCA, obtenemos el gráfico de la figura 14.

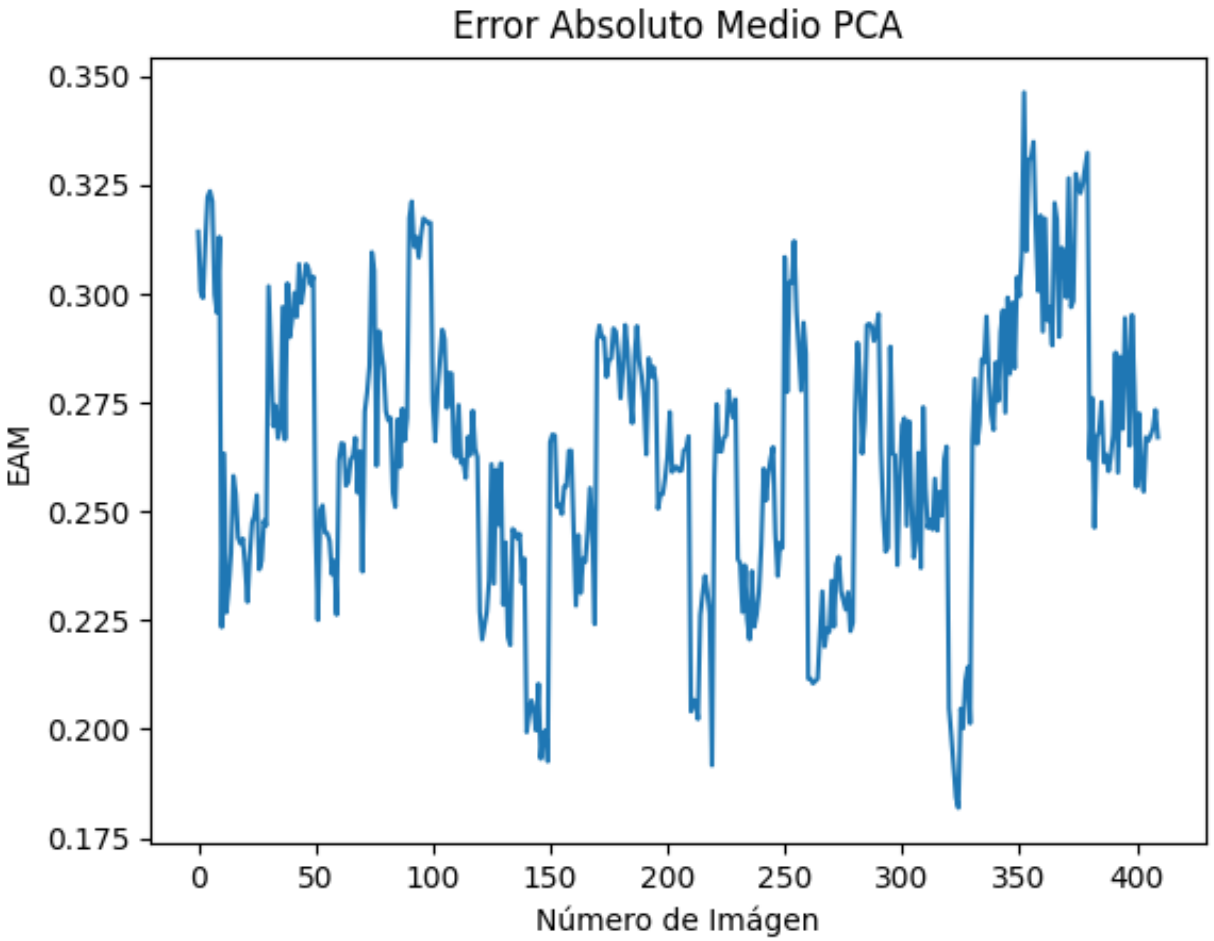

Figura 14: EAM para PCA

Tomando $k = 92$ para 2DPCA, obtenemos el gráfico de la figura 15.

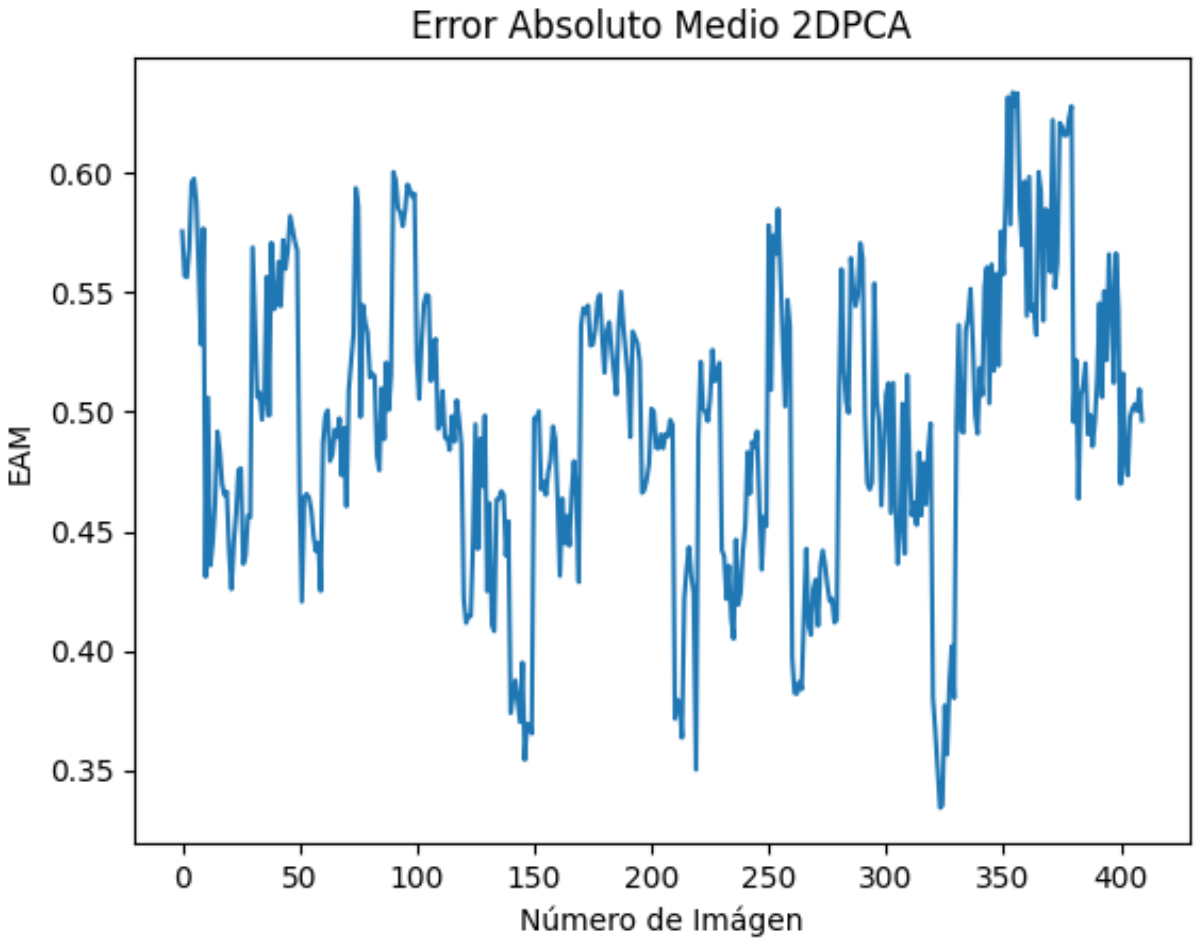

Figura 15: EAM para 2DPCA

En ambos casos tomamos valores de k para los cuales la reconstrucción de las imgágenes era parecida a la original simple vista. Como podemos observar en la figura 16, para PCA la mediana del error es 0.263, mientras que en 2DPCA es de 0.5. La caja de rango intercuartil va para PCA es más aplanada que para 2DPCA, lo que significa que hay más cantidad de datos cerca de la mediana. En ambos casos, los bigotes son relativamente cortos y, en el caso de 2DPCA, encontramos un valor atípico menor que el resto. Tomando estos valores para k, el EAM para PCA es más bajo que para 2DPCA.

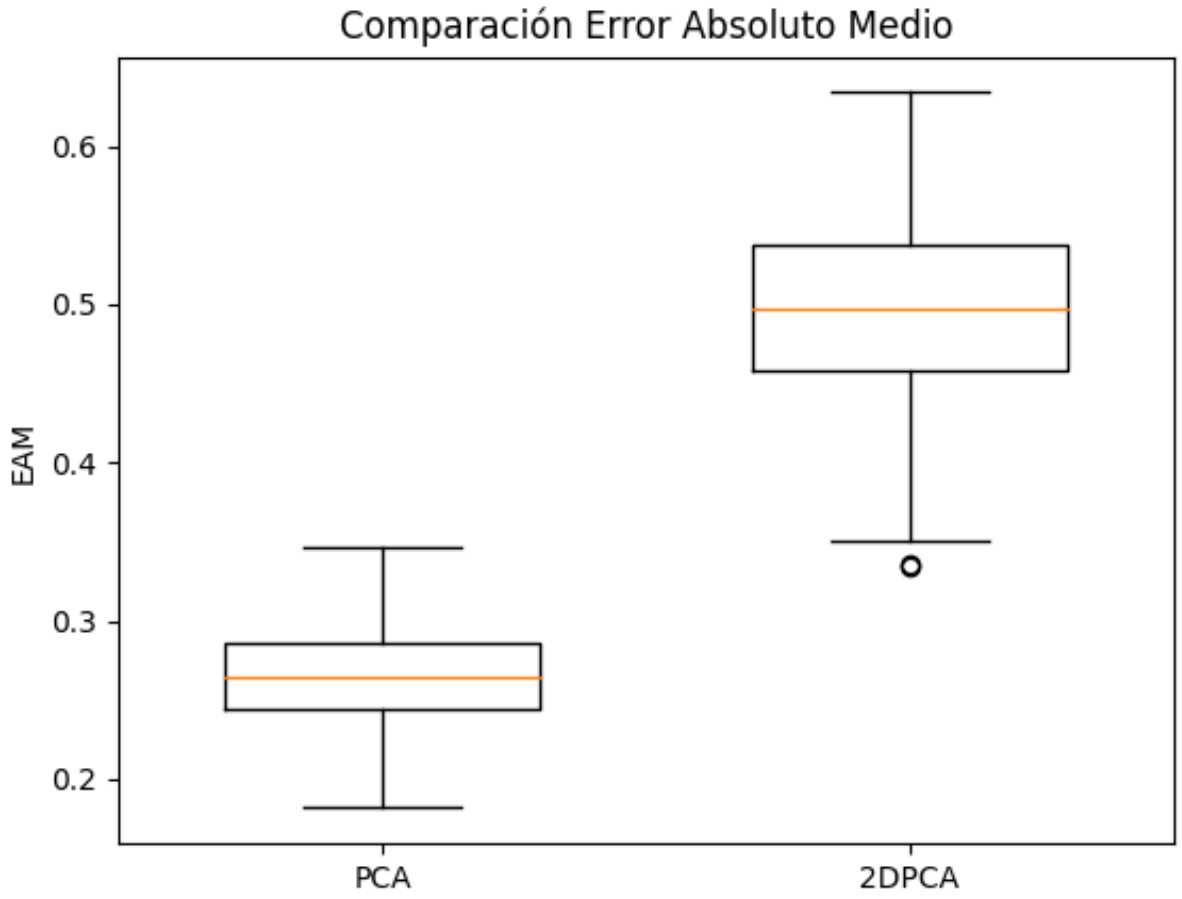

Figura 16: Comparación EAM PCA y 2DPCA

¿Qué sucedería si quitamos a una persona del conjunto de datos antes de hacer PCA/2DPCA y luego calculamos el EAM?

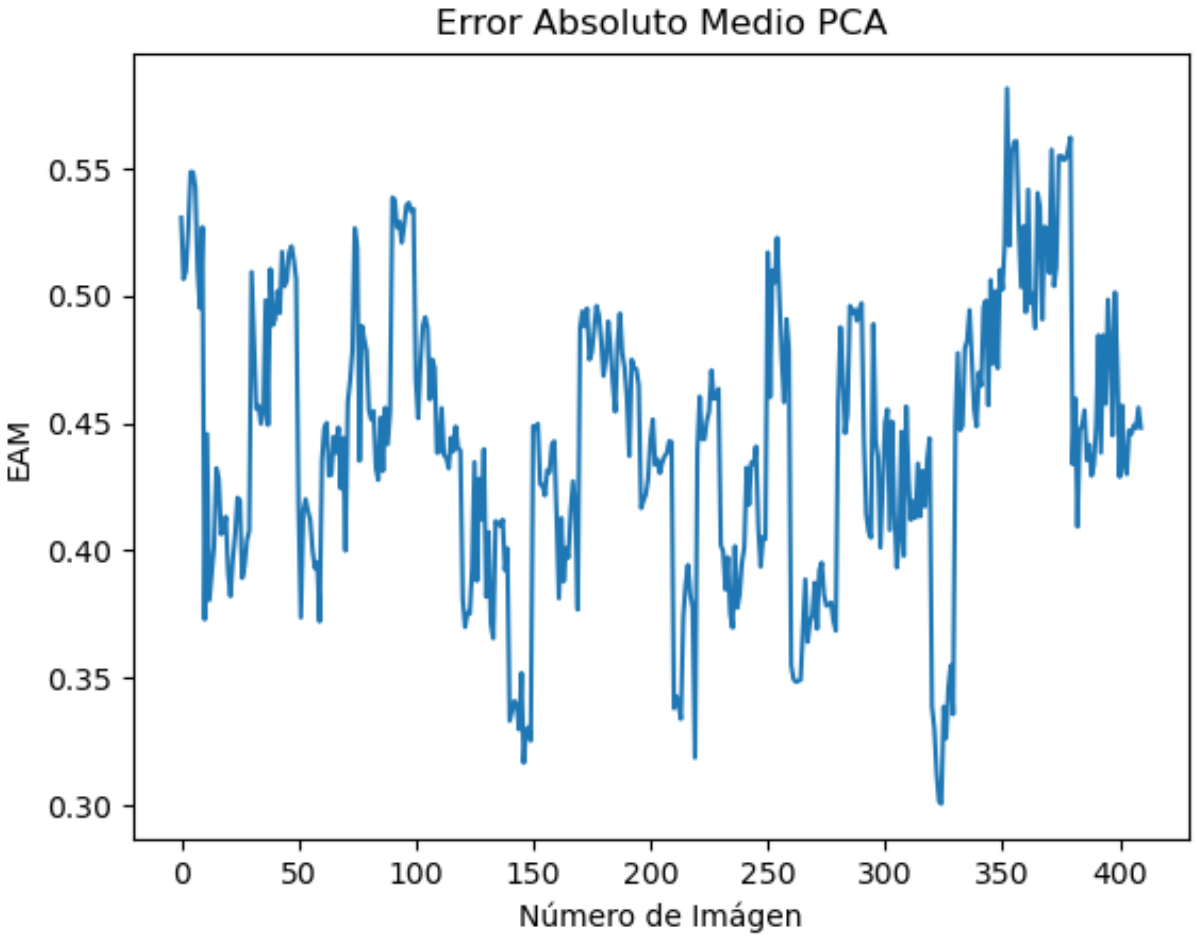

Figura 17: EAM en PCA quitando una persona

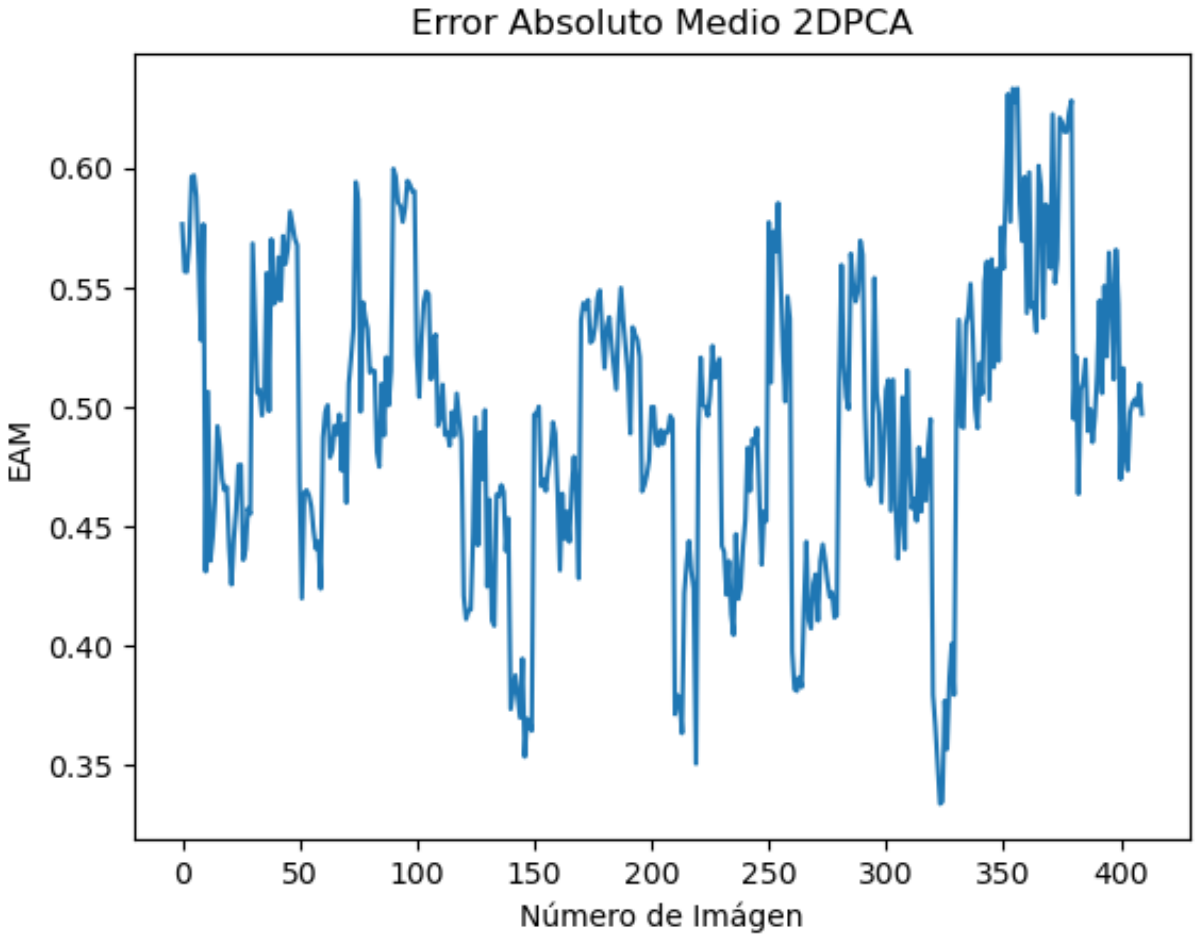

Figura 18: EAM en 2DPCA quitando una persona

Observando la figura 19, el EAM de 2DPCA no sufre cambios drásticos mientras que PCA se ve afectado por la modificación. La mediana de PCA pasa de 0.263 a 0.44 y la de 2DPCA permanece en 0.5.

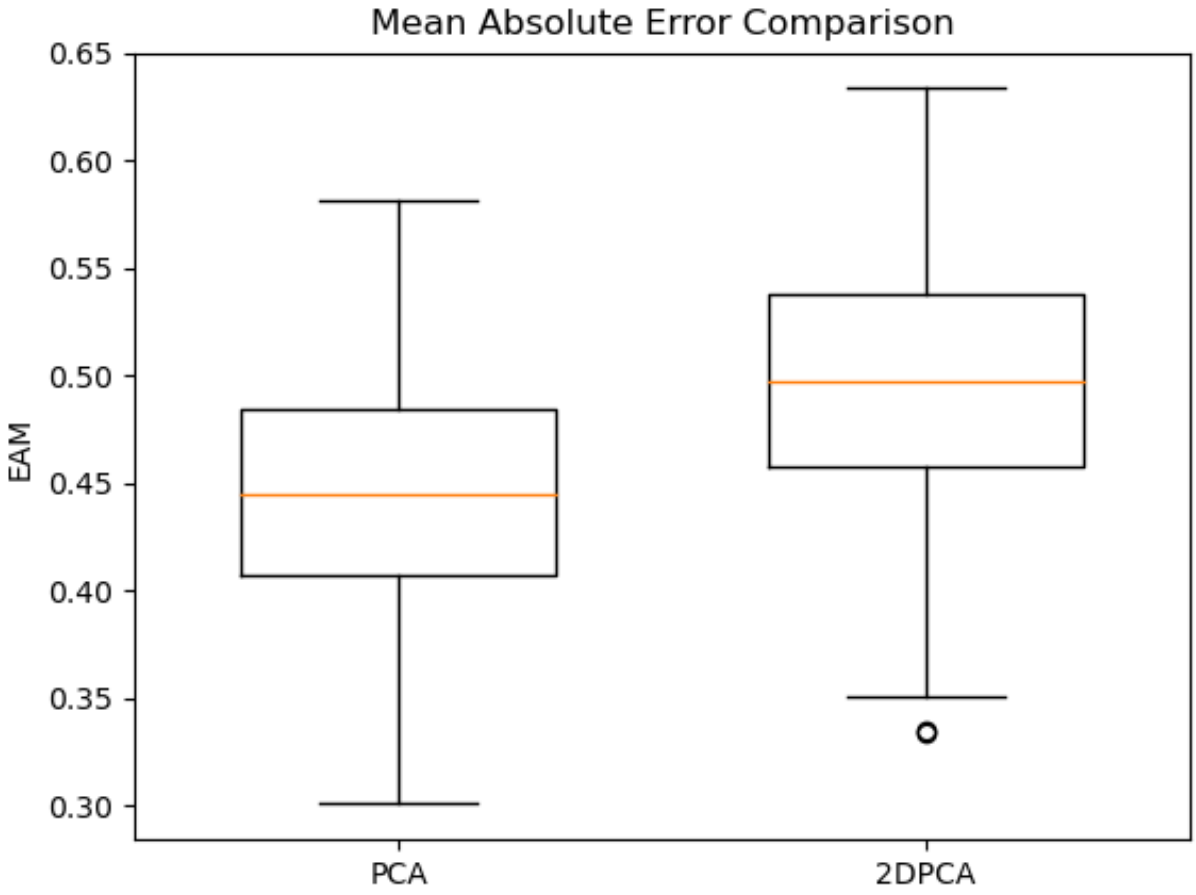

Figura 19: Comparación EAM PCA Y 2DPCA quitando una persona

Como podemos ver en la figura 20, el tiempo de cómputo al aplicar PCA aumenta considerablemente al aumentar la cantidad de componentes principales que tomamos. Para un k grande, puede llegar a tardar al rededor de 0.25 segundos.

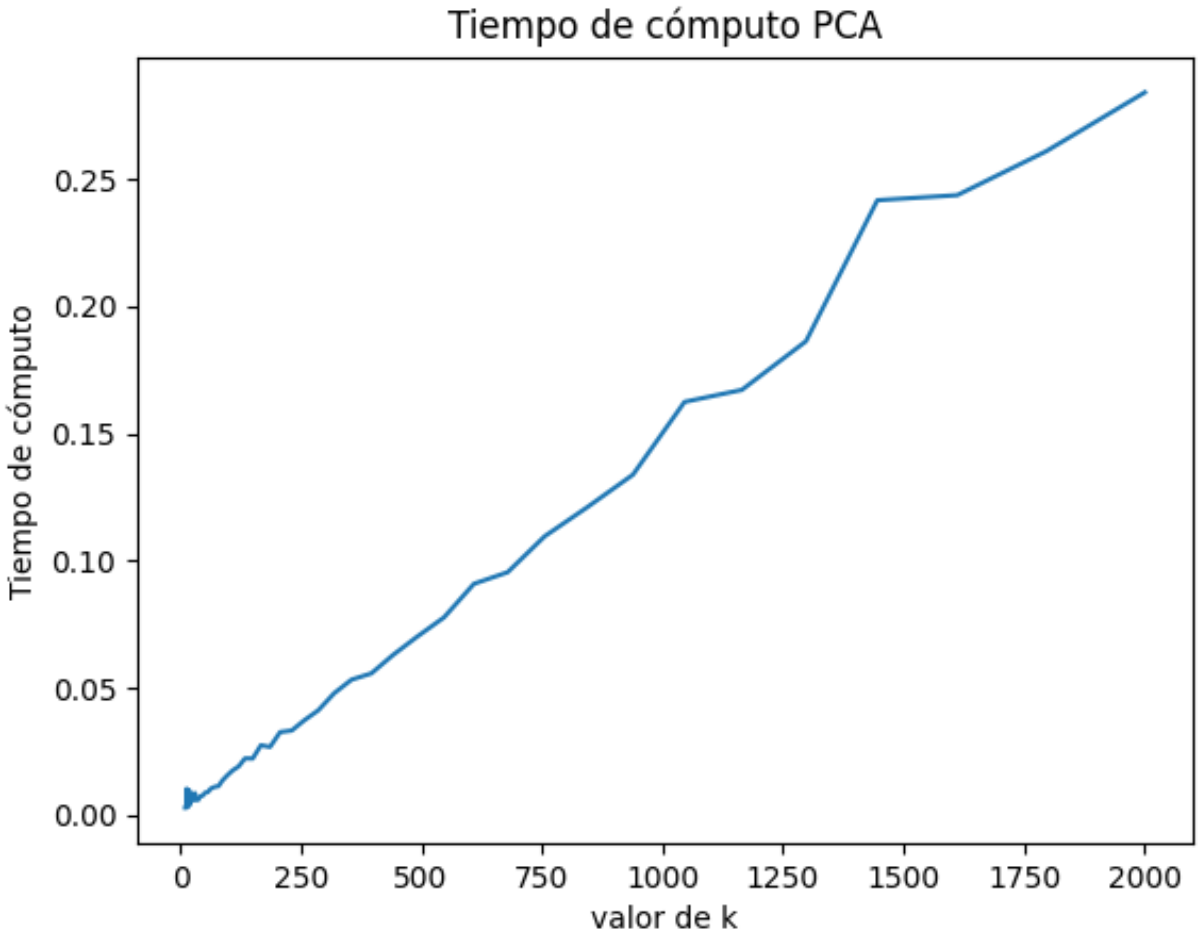

Figura 20: Tiempo de cómputo PCA en función de k

En cambio, en la figura 21, observamos que el tiempo de cómputo de aplicar 2DPCA es mucho más bajo. Incluso tomando $k = 92$ no supera los 0.1 segundos.

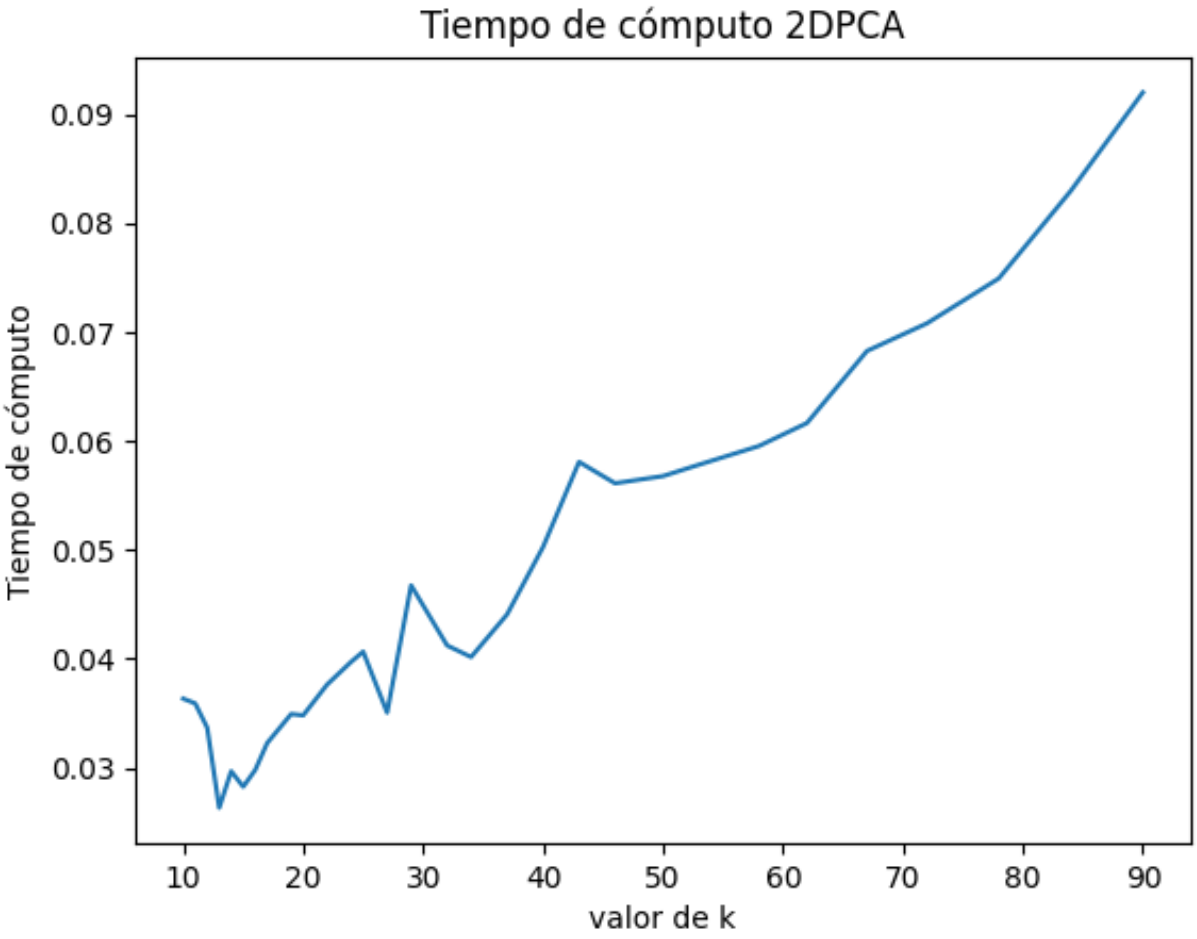

Figura 21: Tiempo de cómputo 2DPCA en función de k

5. Conclusión

En ambos métodos tenemos que calcular los autovectores de una matriz, en PCA se los calculamos a la matriz de covarianza C y en 2DPCA a la matriz G. La dimensión de C (10304×10304) es mucho mayor a la de G (92×92) y esto se ve reflejado en el cálculo de autovectores.

Calcular un autovector con nuestro método de la potencia en el peor caso resulta $\mathcal{O}(10000 * n^2)$ ya que realizamos 10000 iteraciones y dentro del cilco hay una multiplicación vector traspuesto por matriz por vector.

Pero como tenemos que calcular todos los autovectores, hay que repetir este proceso 10304 veces para PCA y 92 veces para 2DPCA. En nuestro experimento, el tiempo de ejecución de PCA puede tardar horas mientras que en 2DPCA tarda segundos.

Como decidimos almacenar las matrices C y G en archivos txt, PCA requiere mayor espacio de almacenamiento que 2DPCA. La matriz C llega a ocupar 2.7 GB en memoria mientras que la matriz G ocupa sólo 211.8 kB.

En conclusión, en el procesamiento de imágenes 2DPCA resulta más eficiente que PCA en términos de tiempo y almacenamiento. Sin embargo, PCA resulta más preciso midiendo el EAM. En nuestros experimentos, usamos el método de la potencia para calcular autovalores y autovectores y la diferencia en el tiempo de cálculo es amplia. Utilizando otro método, quizás la diferencia varíe, pero si se usa el método de la potencia, 2DPCA resulta más efectivo para procesar imágenes.