Implementación de sistema de reconocimento facial usando Flask

Horacio Lamas Arellano A01367213
Víctor Hugo Franco Juárez A01366475
Juan Pablo Ortiz Ortega A01366969
Adrián Torres González A01369810
Jorge Alberto Flores Ponce A01769059

Docentes:
Juan Manuel Alvarado López
Marcial Roberto Leyva Fernández
Víctor Manuel de la Cueva Hernández

04 de mayo de 2023

Abstract

Se cree que para la creación de un sistema de reconocimiento facial lo que definirá por completo la calidad de los resultados son los algoritmos encargados de detectar y reconocer el rostro de una persona en una imagen, y aunque son piezas esenciales en un sistema de este tipo, no son todo lo que se debe de tomar en cuenta; subprocesos como lo son las funciones de vectorización de una imagen o de obtención de la similaridad entre vectores también desempeñan un papel importante; el elegir la implementación incorrecta puede conllevar a obtener resultados menos precisos. Este trabajo se enfoca en la implementación de tres modelos de vectorización y tres métodos de detección de similaridad en un sistema de reconocimiento facial implementado en el lenguaje de programación *Python* y con el framework de aplicaciones web *Flask* para poder explicar y comprobar cómo es que la disminución desmedida de dimensiones de una imagen puede ocasionar reconocimientos erróneos.

Introducción

El reconocimiento facial es un campo que ha ganado gran importancia en los últimos años debido al gran avance que ha tenido en cuanto a precisión y velocidad, teniendo aplicaciones en campos como lo son el de la seguridad y el de almacenamiento de datos biométricos.Para poder realizar un reconocimiento certero sobre una persona, se requiere tener un modelo entrenado con fotografías de las personas que se quieren identificar y, a su vez, una forma de comparar una fotografía recientemente ingresada con el modelo entrenado. Sin embargo, eso no es todo lo que se tiene que tomar en cuenta al querer obtener resultados precisos de un sistema de reconocimiento facial. Los procesos de preprocesamiento de imágenes, de vectorización, de obtención de similaridad, entre otros, también pueden influir a la hora de tratar de optimizar el uso de memoria, la velocidad del sistema, y la calidad de los resultados. Es por ello que la siguiente investigación se basa en el análisis de tres modelos de vectorización de matrices obtenidas a partir de imágenes, y tres métodos de detección de similaridad entre un par de vectores que contienen las características faciales más significativas de un rostro detectado en una imagen, para comprobar su eficacia al comparar los beneficios que contrae cada una, y analizar si los métodos son efectivos para un software con las características del proyecto.

Trabajos Previos

La implementación de métodos alternativos a los implementados en este proyecto de reducción de dimensiones de la matriz de una imagen, así como del cálculo de la similaridad entre un par de vectores, puede ser encontrada en varias librerías de Python, entre ellas *Scikit Learn* (sklearn) , la cual es una librería de aprendizaje automático que proporciona algoritmos y herramientas para, entre otras cosas, la clasificación y regresión de datos. El módulo *decomposition* de sklearn incluye algoritmos de descomposición de matrices como PCA y SVD, las cuales son utilizados como técnicas de reducción de dimensiones de una matriz. Existen otras técnicas las cuáles no fueron implementadas ni estudiadas durante la realización del proyecto pero que, dependiendo de las características específicas de cada implementación, pueden generar resultados similares o incluso mejores, como por ejemplo:

- PCA Incremental (*sklearn.decomposition.IncrementalPCA*): Realiza una reducción de dimensiones lineal utilizando el SVD (Descomposición de Valores Singulares) de los datos, manteniendo sólo los vectores singulares más significantes para proyectar los datos en un espacio dimensional más bajo. Dependiendo del tamaño de los datos de entrada, este algoritmo puede ser más eficiente en cuanto al uso de memoria que el PCA.
- NMF (*sklearn.decomposition.NMF*): El método de NMF (Factorización de Matrices No Negativas) encuentra dos matrices no negativas cuyo producto se aproxima a la matriz de entrada no negativa.

El submódulo *pairwise* del módulo *metrics* de sklearn incluye métricas de distancia y kernels como la similaridad coseno y la distancia euclidiana para evaluar la paridad entre distancias o afinidad de conjuntos de datos. Otras técnicas de obtención de similaridad que pudieron haber sido implementadas y analizadas en el proyecto son:

- Distancia Haversine (*sklearn.metrics.pairwise.haversine_distances*): Calcula la distancia angular entre dos puntos en la superficie de una esfera, donde uno de los datos de entrada es la longitud y la otra la latitud; ambas dadas en radianes.
- Kernel de chi cuadrada (*sklearn.metrics.pairwise.chi2_kernel*): Calcula el kernel de chi cuadrada exponencial de dos conjuntos para determinar si existe una relación significativa entre ellos.

Métodos y herramientas

Para la realización de este proyecto, se hizo uso de la versión 3.10 del lenguaje de programación *Python*. Las librerías utilizadas durante todo el desarrollo del proyecto son las siguientes:

- *Flask*: Micro framework para el desarrollo de aplicaciones web.
- *face_recognition*: Librería que permite la detección y reconocimiento facial en imágenes y videos.
- *scikit-image*: Librería para el preprocesamiento, segmentación y manipulación de imágenes.
- *matplotlib*: Librería para la creación de gráficos y visualizaciones.
- *cmake*: Herramienta de construcción de software utilizada para la generación de archivos de configuración y la construcción de proyectos de software. Esta es necesaria para poder utilizar la librería face_recognition.
- *numpy*: Librería para la realización de cálculos matemáticos y científicos.
- *opencv-python*: Librería de visión por computadora que brinda herramientas para el procesamiento de imágenes y videos

Para la detección y localización de rostros en una imagen se utiliza la función *face_locations()* de la librería *face_recognition* de *Python*. Esta función toma una imagen como entrada y devuelve una lista de coordenadas con las ubicaciones de los rostros detectados. Asimismo utiliza un algoritmo basado en redes neuronales convolucionales (CNN); las cuales son un tipo de red diseñado para procesar datos de

alta dimensionalidad como lo son las imágenes, y posee una precisión del 99.38 % en la base de datos pública de imágenes faciales "Labeled Faces in the Wild". (University of Massachussetts, 2018)

En la implementación del proyecto solo se toma en cuenta la primera cara detectada de una imagen por lo que es posible que no se reconozca correctamente a una persona si se detectan varios rostros en una imagen.

La vectorización de las imágenes ha sido generada con tres diferentes métodos, dos de ellos implementados haciendo uso del módulo *decomposition* de la librería sklearn de *Python*. Los métodos son los siguientes:

- Análisis de componentes principales o PCA (*sklearn.decomposition.PCA*): Realiza una reducción de dimensionalidad linear usando el SVD de los datos para encontrar una representación más simple y mantener la mayor cantidad de información relevante posible.
- Descomposición de Valores Singulares o SVD (sklearn.decomposition.TruncatedSVD): Realiza la descomposición de una matriz en tres matrices más simples; una matriz ortogonal de vectores izquierdos, una matriz diagonal de vectores singulares y una matriz ortogonal de vectores derechos, para posteriormente combinarlas.
- La función *face_encodings* de la librería *face_recognition*, la cual calcula un vector numérico de 128 dimensiones que representa las características faciales únicas de un rostro, es aplicada a una imagen para posteriormente reducir su vector resultante con PCA o SVD. Sin embargo la función *face_encodings* puede ser considerada también como un método de vectorización o de reducción de dimensiones. El tercer método de vectorización se basa tan sólo en utilizar el resultado obtenido de esta función, sin aplicar ningún modelo de reducción. Este será implementado para poder analizar las ventajas y desventajas que conlleva el utilizar modelos de reducción.

La similaridad de las imágenes fue calculada con tres diferentes métodos. Cada uno consiste en lo siguiente:

- Distancia Euclidiana normalizada L2 (*numpy.linalg.norm)*: Se calcula la norma del resultado de la resta de dos vectores. La norma es una medida de longitud o magnitud de una matriz o vector en un espacio vectorial. Esta función fue implementada con la ayuda de la librería *numpy* de *Python*.
- Similitud coseno (*sklearn.metrics.pairwise.cosine_similarity*): Se calcula el coseno del ángulo entre dos vectores, si los vectores de entrada apuntan a la misma dirección, el resultado será 1, si apuntan en direcciones opuestas se obtendrá -1, y si son perpendiculares el resultado será 0. Esta función fue implementada haciendo uso de la librería sklearn de *Python*.
- Distancia Manhattan: Se calcula la distancia entre dos puntos en un espacio de varias dimensiones al calcular la suma de las diferencias absolutas de las coordenadas de los puntos. Esta función fue implementada manualmente.

Implementación

La implementación consta del *front end* realizado con *HTML*, *CSS*, y *JS*, que se integra al *back end* realizado en *Python* por medio del framework *Flask*.

El *back end* consta de cinco archivos *.py* que permiten el funcionamiento correcto del sistema:

camera_func:

Accede a la cámara del equipo donde se ejecuta el sistema y permite la captura de la fotografía que va a ser almacenada.

compareFace:

Realiza la comparación e identificación facial de acuerdo al modelo establecido (sin reducción, PCA, SVD), y al método de similitud elegido (L2, Cosenos, Manhattan).

matrixRecognition:

Este script permite vectorizar las imágenes de entrenamiento y las guarda en un archivo *csv* para uso posterior. A este archivo se le conocerá como la matriz original. También vectoriza la imagen capturada.

matrixReduction:

Reduce la matriz original a una que cuente con dos dimensiones mediante los métodos de reducción PCA y SVD. También reduce la fotografía a comparar.

App: Es el archivo principal que usa *Flask* para conectar entre el *front end* y *back end*. Hace uso de los archivos *.py* definidos anteriormente, para generar y compartir los resultados de la comparación e identificación al *front end* en forma de un arreglo multidimensional que se despliega como trío de imagen, nombre y valor (en porcentaje).

Resultados y análisis

El sistema funciona y es capaz de identificar rostros de personas que pertenecen al dataset de entrenamiento, que contiene fotos de alumnos y profesores de la materia TC3002B.

Hay una variación entre resultados dependiendo de la combinación del modelo de reducción y método de comparación elegido, así como de la calidad y de las condiciones de iluminación y posición de la imagen capturada.

1. Se necesita de una imagen con condiciones similares a las del set de prueba para garantizar resultados más certeros, ya que cambios en el entorno y contexto de la imagen pueden generar inconsistencias en los resultados.

2. Usando la matriz original de 128 dimensiones se obtienen resultados de similitud más precisos que cuando se hace uso de los métodos SVD y PCA. En los casos de prueba realizados utilizando la matriz original, el sistema siempre regresó cuatro caras correctas sin importar el método de comparación. Sin embargo, en datasets más grandes, esta matriz tiene un mayor costo de cómputo que un método con reducción debido a la alta dimensionalidad del vector resultante (128 dimensiones contra 2 dimensiones de un modelo reducido).

3. En cuanto al uso de los métodos de reducción como SVD y PCA, si bien se ahorra espacio en memoria y CPU, los resultados no son tan certeros como con la matriz no reducida, ya que empíricamente fallaron más de una vez en detectar a la persona correcta.

4. Se podría mejorar el reconocimiento con los modelos reducidos si se alimentara el set de entrenamiento con un mayor número de fotografías por sujeto y con distintas condiciones de luz, expresiones faciales, ángulos y entorno.

5. Los métodos de reducción son particularmente útiles para la visualización de imágenes ya que generan una escala bidimensional. Al usar el modelo PCA se obtuvo una representación más precisa de cómo los rostros de una misma persona se agrupan según sus características que el modelo SVD. Esto se puede observar analíticamente en la nube de puntos generada, donde se aprecia más cohesión en el modelo PCA y mayor dispersión en el modelo SVD.

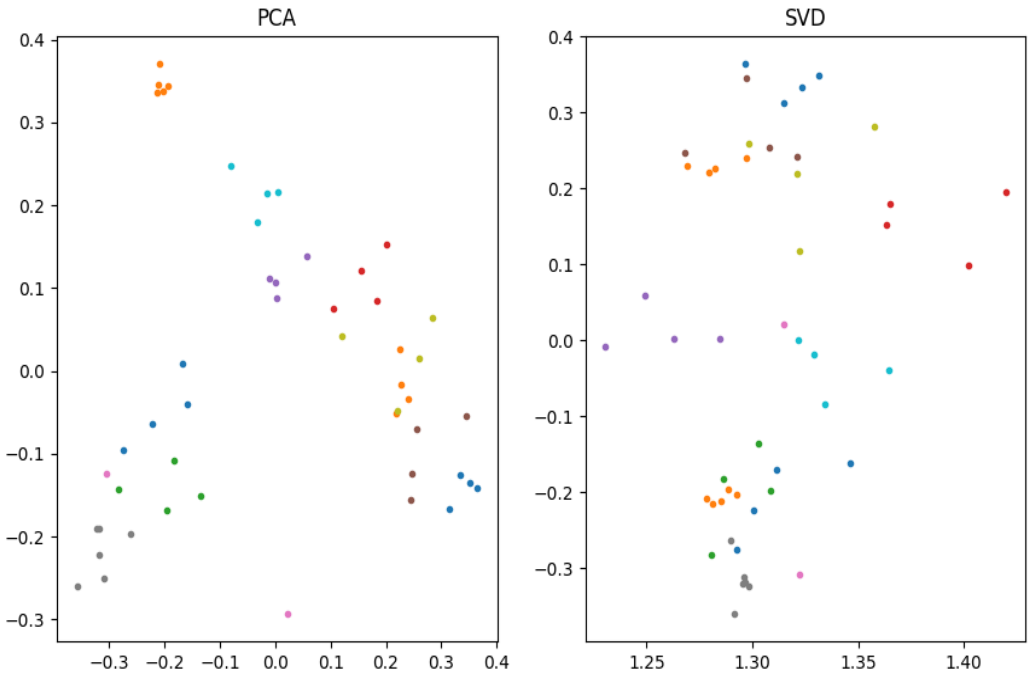

Figura 1. Gráfica de dispersion de las imágenes del dataset reducidas usando PCA y SVD

Conclusiones

La reducción de dimensiones de una imagen para un sistema de reconocimiento facial puede servir para disminuir la cantidad de información de la misma, conservando unicamente la información más relevante, logrando así que se utilice menos almacenamiento para la base de datos que contiene los rostros que el sistema es capaz de reconocer, reduciendo también el tiempo de procesamiento.

El que se haya experimentado con disminuir la cantidad de dimensiones de n * m píxeles (la cantidad total de píxeles de la imagen de entrada) a 128 dimensiones (los descriptores faciales obtenidos con *face_encoding)* a dos dimensiones, se hizo con el objetivo de comprobar si esta reducción puede llegar a obtener resultados tan certeros como los sistemas que utilizan más dimensiones (como *face_encoding*) y ofrecer ventajas diferentes además de tener la posibilidad de visualizar los datos en gráficos para poder analizarlos. Se puede concluir que el utilizar tan solo dos atributos (correspondientes al par de dimensiones antes mencionado) para el reconocimiento facial ofrece resultados irregulares; cualquier cambio mínimo en la iluminación o el añadido de algún accesorio como lentes a la imagen de entrada puede ocasionar que no se reconozca correctamente a la persona, esto independientemente del método de obtención de similaridad utilizado. Haciendo uso del vector con 128 dimensiones se pudieron obtener resultados más consistentes y certeros, por lo que este método de vectorización se mantiene como la mejor alternativa para un sistema de este tipo.

De los métodos de obtención de similaridad entre dos vectores no hubo uno que destacara más que los demás, ninguno obtuvo resultados lo suficientemente consistentes como para asegurar que alguno fuera el indicado para ser usado en un sistema de reconocimiento facial. Además, cada método requirió de una normalización especial para poder presentar los valores en escala de 0 a 100 %. Se necesitaría una base de datos más grande (con más fotografías por persona) para poder realizar experimentos que permitan verificar cuál método de obtención de similaridad es el mejor en un proyecto con características similares al desarrollado y analizado en este trabajo.

Bibliografía

Anggo, M., & Arapu, L. (2018). Face Recognition Using Fisherface Method. Journal of Physics: Conference Series, 1028(1), 012119. doi:10.1088/1742-6596/1028/1/012119

Anónimo. 2022. "Calculate Manhattan Distance in Python (City Block Distance)". Datagy. Recuperado de : Calculate Manhattan Distance in Python (City Block Distance) • datagy

Anónimo. 2023. "API Reference". Scikit Learn. Recuperado de: API Reference — scikit-learn 1.2.2 documentation

Anónimo. 2023. "NumPy Documentation". NumPy. Recuperado de: NumPy Documentation

Avinash Kumar Singh; G. C. Nandi; "Face recognition using facial symmetry". CCSEIT '12: Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology. October 2012. Pages 550–554. Recuperado de: https://0-doi-org.biblioteca-ils.tec.mx/10.1145/2393216.2393308

Hanumanthappa, D., & Ashoka, D. (2018, June). Face Detection and Recognition Using K-Means and Enhanced K-Means Algorithm with ANN. International Journal of Recent Research Aspects, pp. 23-27.

Pengcheng, W., Zhen, Z., Li, L., & Jiao, J. (2018). Research on face feature extraction based on K-mean algorithm. EURASIP Journal on Image and Video Processing.

Sageitgey. 2022. "face_recognition". GitHub. Recuperado de: ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line (github.com)

University of Massachussetts,. 2018. "Labeled Faces in the Wild Home". University of Massachusetts. Recuperado de: LFW Face Database : Main (umass.edu)

Yifan Wei; "Review of Face Recognition Algorithms". ICBSP'20: 2020 5th International Conference on Biomedical Imaging, Signal Processing. September 2020. Pages 28–31. Recuperado de: https://0-doi-org.biblioteca-ils.tec.mx/10.1145/3436349.3436367