Universidad Central de Venezuela
Facultad de Ciencias
Postgrado en Ciencias de la Computación

Reconocimiento facial combinando técnicas de extracción de características con redes neuronales convolucionales

Trabajo de Grado de Maestría

presentado ante la Ilustre

Universidad Central de Venezuela

Por el Licenciado

Yilber Sisco

Tutor: Prof. Rhadamés Carmona

7 de agosto de 2018

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
COMISIÓN DE ESTUDIOS DE POSTGRADO

Comisión de Estudios de Postgrado

VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el **Trabajo de Grado** presentado por: Yilber Jesus Sisco Estrada, cédula de identidad 11.990.980, bajo el titulo "Reconocimiento facial combinando técnicas de extracción de características con redes neuronales convolucionales", a fin de cumplir con el requisito legal para optar al grado académico de **MAGÍSTER SCIENTIARUM, MENCIÓN CIENCIAS DE LA COMPUTACIÓN**, dejan constancia de lo siguiente:

1.- Leído como fue dicho trabajo por cada uno de los miembros del jurado, se fijó el día 30 de Julio de 2018 a las 12:30pm, para que el autor lo defendiera en forma pública, lo que éste hizo en el Cetro de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, mediante un resumen oral de su contenido, luego de lo cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el jurado, todo ello conforme con lo dispuesto en el Reglamento de Estudios de Postgrado.

2.- Finalizada la defensa del trabajo, el jurado decidió **aprobarlo**, por considerar, sin hacerse solidario con la ideas expuestas por el autor, que se ajusta a lo dispuesto y exigido en el Reglamento de Estudios de Postgrado

Para dar este veredicto, el jurado estimó que el trabajo examinado propone un esquema hibrido basado en aprendizaje profundo y técnicas de extracción de características para el reconocimiento de rostros humanos en escenas no controladas.

En fe de lo cual se levanta la presente ACTA, a los 30 días del mes de Julio del año 2018, conforme a lo dispuesto en el Reglamento de Estudios de Postgrado, actuó como Coordinador del jurado el Prof. Rhadamés Carmona.

Haydemar Núñez / C.I. 5.538.772
Universidad Central de Venezuela

Eugenio Scalise / C.I. 10.184.983
Universidad Central de Venezuela

Rhadamés Carmona / C.I. 10.804.242
Universidad Central de Venezuela
Tutor(a)

POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
Universidad Central de Venezuela

RC/30-07-2018

Resumen

Este documento describe un método para el reconocimiento de rostros a partir de la combinación de distintos modelos utilizando técnicas de transferencia del aprendizaje y redes neuronales. El trabajo tiene como objetivo evaluar y comparar el rendimiento de distintos modelos de reconocimiento facial, así como también, el resultado de la combinación de ellos. Los modelos estudiados están basados en Histogramas de la Orientación de los Gradientes (HOG), Transformaciones de Características Invariantes a la Escala (SIFT), Histogramas de Patrones Locales Binarios (HLBP) y Redes Neuronales Convolucionales (CNN). El entrenamiento de los modelos de reconocimiento facial se efectuá a partir de un conjunto de imágenes adquiridas bajo distintas condiciones de iluminación, pose y expresión.

Palabras Claves: visión por computador, data mining, HOG, SIFT, LBP, redes neuronales, redes neuronales convolucionales, OpenCV, machine learning , inteligencia artificial

Tabla de Contenidos

Índice de figuras

Índice de cuadros

Listados de Código

Introducción

En la actualidad se ha incrementado el uso de algoritmos de inteligencia artificial para hacer que las máquinas "aprendan" a reconocer patrones o distinguir la presencia de ciertos objetos de interés. En condiciones controladas los sistemas de reconocimiento de rostros han alcanzado un nivel de precisión cercano al de los seres humanos [22]. Un ambiente controlado, es aquel en donde las condiciones de iluminación, posición, escala y rotación están predefinidas; las fotografías utilizadas en documentos de identidad como pasaportes, son un ejemplo de imágenes adquiridas en ambientes controlados. Si bien, el reconocimiento de rostros en ambientes controlados resulta útil cuando se pueden establecer condiciones similares para la captura de las imágenes, en algunas situaciones esto no es posible, como por ejemplo, cuando las imágenes son obtenidas a partir de videos.

Figura 1: Rostros en ambientes no controlados.

El rostro humano es un objeto dinámico y posee un alto grado de variabilidad en su apariencia. Factores como la posición, la iluminación y la expresión son determinantes en el reconocimiento de rostros, convirtiéndose, en un problema no trivial. La

figura 1 muestra algunas de las imágenes típicas utilizadas por los sistemas automáticos de reconocimiento de rostros. En un escenario no controlado, el rostro podría encontrarse en un fondo complejo y/o en posiciones diferentes [12].

El éxito de los algoritmos de aprendizaje automático depende de la representación de datos, y esto se debe a que diferentes representaciones pueden contener u ocultar en menor o mayor grado los diferentes factores explicativos de la variación de los datos. Se han desarrollado una gran variedad de técnicas para el reconocimiento facial que se basan en distintos enfoques para la determinación de la identidad del rostro contenido en una imagen. Entre las que han logrado mayores rendimientos se encuentran: las basadas en la reducción de la dimensionalidad como Eigenfaces [11], que se apoya en el análisis de componentes principales (PCA) [11] y Fisherfaces [10], que se fundamenta en el Discriminante Lineal de Fisher (LDA) [10]. Otro grupo de métodos se basan en el conocimiento del dominio para extraer información de los objetos presentes en una imagen mediante un descriptor (Histogramas de Gradientes Orientados (HOG) [3], Patrones Locales Binarios (LBP) [17], Histogramas de Patrones Locales Binarios (HLBP) [17], Transformada de Característica Invariantes a la Escala SIFT [7]). El descriptor no es más que una nueva representación que se obtiene a partir del procesamiento de la imagen mediante un algoritmo; estos métodos luego utilizan un algoritmo de aprendizaje automático (Máquinas de Soporte Vectorial (SVM) [31], Perceptrón Multicapas (MLP) [32], etcétera) para encontrar patrones en los datos proporcionados por los descriptores. Los descriptores pueden ser invariantes a la posición, perspectiva e iluminación entre otras, lo que significa que descriptores de la misma imagen con variaciones en estas condiciones deberían ser similares si el descriptor es suficientemente robusto.

Los métodos más recientes utilizan el trabajo realizado por Lecum et al. [2], en el cual se propuso la arquitectura de una Red Neuronal Convolucional (CNN), con el objetivo de reconocer dígitos escritos a mano. Estas redes han resultado exitosas en numerosas aplicaciones prácticas, su nombre proviene del hecho de que emplean una operación matemática llamada convolución para generar descriptores de las imágenes en estudio. Las CNN están biológicamente inspiradas, pueden aprender

características invariantes y forman parte de los principios relacionados con lo que en la actualidad se conoce como aprendizaje profundo o "Deep Learning". Estas redes aprenden qué características son importantes para describir un objeto de interés.

Hasta la fecha se han empleado distintos enfoques para el reconocimiento de rostros, algunos utilizan descriptores basados en la extracción de características, mientras que los más recientes se basan en el uso de redes neuronales convolucionales. Ambos enfoques, salvo algunas excepciones, han sido estudiados de manera individual. Podemos encontrar trabajos en los que se entrenan algoritmos de reconocimiento facial con descriptores HOG [4], HLBP [13] , SIFT [6] o redes neuronales convolucionales con imágenes (RAW) [22], pero muy pocos estudios han enfrentado el problema de reconocimiento facial mediante la combinación de la información suministrada por distintos descriptores [42]. Un descriptor proporciona un conjunto de características de una imagen mediante el procesamiento de la misma a través de un algoritmo. Este procesamiento permite extraer datos que el investigador considera claves para la solución del problema. Cada algoritmo proporciona una *"vision"* alternativa de la imagen en estudio. Algunos consideran la textura como un elemento de gran importancia, para otros los bordes tienen mayor relevancia, mientras que en varios enfoques suele seleccionarse un conjunto de puntos a partir de un criterio determinado.

Este trabajo plantea un estudio de distintos métodos de extracción de características basados en el conocimiento del dominio y su combinación con técnicas de "Deep Learning", para el reconocimiento de imágenes de rostros con distintas condiciones de iluminación, pose y expresión . El estudio incluirá métricas de clasificadores entrenados con imágenes (RAW) así como también las métricas de clasificadores entrenados mediante representaciones basadas en descriptores de imágenes (LBP, HLBP, HOG, SIFT).

Objetivo General

Este Trabajo de Grado tiene como objetivo construir un modelo de reconocimiento facial mediante la combinación de clasificadores entrenados a partir de descriptores basados en HOG, SIFT, LBP, HLBP y redes neuronales convolucionales (CNN), y evaluar su rendimiento sobre un conjunto de rostros con distintas condiciones de iluminación, pose y expresión.

Objetivos Específicos

- Aplicar una metodología de minería de datos para la construcción de un clasificador de rostros.
- Construir y aplicar algoritmos de procesamiento digital de imágenes en la preparación y extracción de información.
- Diseñar y construir representaciones de imágenes a partir de descriptores HOG, SIFT, LBP y HLBP para el entrenamiento de algoritmos de aprendizaje automático.
- Construir modelos de clasificación de rostros, basados en distintas representaciones de las imágenes.
- Evaluar, mediante un estudio comparativo, el rendimiento distintos modelos de aprendizaje automático entrenados a partir de distintas representaciones de imágenes.
- Utilizar técnicas de validación como la validación cruzada (Cross-Validation), para garantizar la calidad de los modelos obtenidos.
- Diseñar y construir un prototipo para una aplicación de reconocimiento facial a partir de un modelo combinado de clasificadores basados en Redes Neuronales Convolucionales (CNN).

Alcance

En este Trabajo de Grado se utiliza la base de datos de imágenes de rostros FEI, para entrenar una arquitectura de Red Neuronal Convolucional (CNN), y compara su rendimiento con el obtenido al entrenar Redes Neuronales (RNA) con descriptores SIFT, HOG, LBP y HLBP generados a partir de las mismas imágenes. Describe además un modelo combinado de aprendizaje automático utilizando técnicas de *"Transferencia de Aprendizaje"*. El análisis de los resultados se basa en comparaciones de las métricas obtenidas y pretende determinar el efecto de la representación utilizada en el rendimiento alcanzado por las redes de forma individual así como también de forma combinada.

Plataforma de Desarrollo

En la implementación de los algoritmos descritos en este trabajo se utiliza el lenguaje de programación Python [44], el cual es ampliamente utilizado en inteligencia artificial porque permite construir fácilmente modelos de aprendizaje automático y posee una gran cantidad de librerías para evaluar y procesar estadísticamente los datos. Para el procesamiento de las imágenes se utiliza una librería de código abierto conocida como OpenCV [43], la cual implementa gran cantidad de algoritmos de visión por computador, así como también contiene implementaciones de los algoritmos de extracción de características (SIFT, HOG, LBP). En la implementación de los algoritmos de aprendizaje automático se usan las librerías Keras [39] y Scikit-Learn [41]. Keras es una librería de Python que permite de manera limpia y sencilla la creación de una gama de modelos de Deep Learning por encima de otras librerías como TensorFlow [40]. Keras fue desarrollado y es mantenido por François Chollet, un ingeniero de Google, y su código ha sido liberado bajo la licencia permisiva del MIT. Keras requiere un backend para su funcionamiento, en este trabajo de grado se ha utilizado TensorFlow como backend para la librería Keras. Durante el desarrollo de la propuesta se utiliza un procesador Intel Core i3, 2.5 Ghz, con memoria RAM de 16 Gb. y un sistema operativo Windows 10 de 64 Bits. Se utiliza un dispositivo de almacenamiento SSD de 525GB donde se almacenan las imágenes con la intención de acelerar la velocidad del procesamiento.

Metodología

La metodología aplicada está basada en el Proceso Estándar de Minería de Datos para la Industria *(CRISP-MD: Cross Industry Standard Process for Data Mining)* [28], que generaliza los enfoques comunes que utilizan los expertos en minería de datos. El ciclo de vida de un proyecto de minería de datos consta de varias fases. La secuencia de las fases no es estricta por lo que se puede ir hacia atrás o hacia adelante entre las diferentes fases siempre que sea necesario. Las lecciones aprendidas durante el proceso pueden provocar nuevas preguntas, a menudo más centradas y los procesos posteriores se beneficiarán de la experiencia obtenida en los anteriores. Esta metodología divide el proceso de minería de datos en cinco fases principales.

- Recopilación e Integración: Se entiende como la fase en donde se adquieren los datos de distintos repositorios y se integran en un conjunto de datos para el proceso de entrenamiento. Se requiere la comprensión del dominio para determinar cuáles son los posibles datos que favorecen el entrenamiento según el objetivo de negocios que se persigue.

- Preparación de los Datos: Consiste en aplicar un preprocesamiento a los datos para eliminar posibles valores anómalos o inconsistentes en el conjunto obtenido en la primera fase. Se pueden aplicar filtros, transformaciones, selección de características, así como construir nuevos atributos mediante el procesamiento de los datos recopilados.

- Minería de datos: En esta fase se seleccionan y aplican varias técnicas de modelado y se calibran los parámetros de entrenamiento para obtener un modelo. Hay varias técnicas que tienen requerimientos específicos para la forma de los datos, por lo que frecuentemente es necesario volver a la fase de preparación de datos.

- Evaluación e Interpretación: En esta etapa se pretende evaluar el rendimiento del modelo generado a partir del algoritmo de aprendizaje automático aplicado a los datos. La medida de evaluación depende del tipo de tarea que se ha seleccionado. Los criterios de calidad de basan en determinar la precisión, compresión e intereses en los datos para verificar si son útiles o novedosos.

Se recomienda el uso de la técnica de validación cruzada para evaluar los resultados y garantizar que sean independientes de la partición entre datos de entrenamiento y prueba.

- Difusión y Uso: Una vez validado el modelo puede utilizarse para el desarrollo de software de apoyo a la toma de decisiones, aplicarse a otro conjunto de datos o ser utilizado en otras aplicaciones.

Organización del Documento

Este documento es presentado en 2 capítulos. El Capítulo 1 introduce los conceptos básicos sobre el tema a tratar y los estudios previos que sustentan el problema a resolver. El Capítulo 2 describe la solución propuesta y sus detalles de implementación. Para finalizar se presentan las conclusiones alcanzadas con este trabajo de grado y algunas propuestas para nuevos proyectos.

Capítulo 1

Marco Teórico

En el reconocimiento de objetos y en el caso particular del reconocimiento de rostros, han sido diversos los enfoques que se han presentado. A continuación se resumen algunas de las técnicas más comúnmente utilizadas, se describen en líneas generales los esquemas basados en la reducción de la dimensionalidad, en descriptores y los más recientes que utilizan redes neuronales convolucionales.

1.1. Reducción de la dimensionalidad

Algunos de los métodos más populares y eficientes para el reconocimiento de rostros se basan en técnicas de reducción de la dimensionalidad. Turk et al. [9] se apoya en el enfoque de reducción de la dimensionalidad, conocido como el Análisis de Componentes Principales (PCA). La idea básica es tratar cada imagen como un vector en un espacio de alta dimensión. A continuación, a cada imagen se le aplica PCA para producir un nuevo subespacio reducido que captura la mayor parte de la variabilidad entre las imágenes de entrada. Los vectores de componentes principales (vectores propios de la matriz de covarianza de la muestra), se denominan *Eigenfaces* o *Autocaras*.

Cada imagen de entrada se puede representar como una combinación lineal de estos Eigenfaces mediante la proyección de la imagen en el nuevo espacio reducido (Ver figura 1.1). Entonces, podemos realizar el proceso de identificación encontrando coincidencias en este espacio reducido. Una imagen de entrada se transforma al

espacio reducido, y la cara más cercana se identifica usando el enfoque del vecino más cercano. Se pueden usar dos versiones de los clasificadores de vecinos más cercanos. El primero compara la imagen de entrada con todas las imágenes de la base de datos. El segundo calcula los centros de cada clúster y utiliza esos centros de agrupamiento para la comparación.

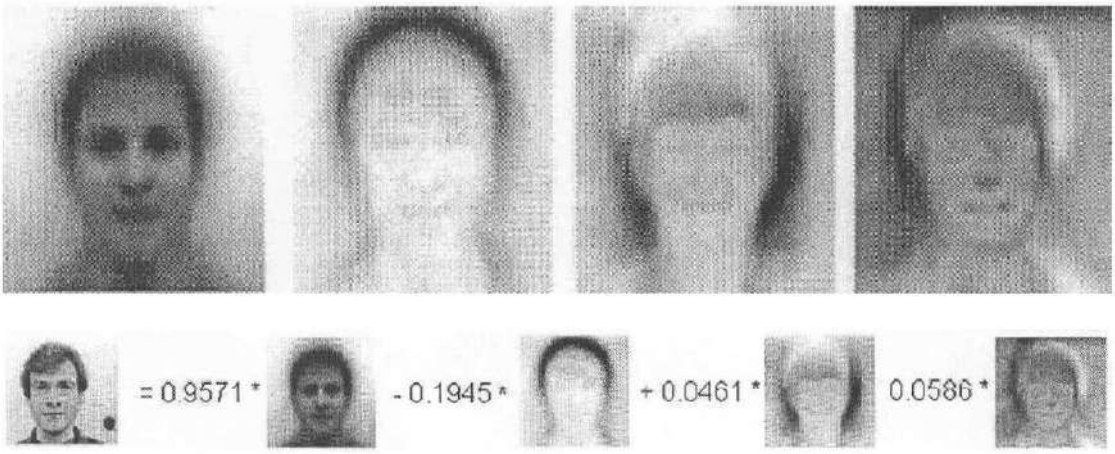


Figura 1.1: Representación de un rostro a partir de PCA [9]

El enfoque de Fisherfaces [10] [11] se basa en el Análisis Discriminante Lineal de Fisher (LDA) [20]. LDA es un método de reducción de dimensionalidad supervisado que tiene como objetivo, encontrar combinaciones lineales de los datos que maximizan la variabilidad entre clases mientras que minimizan la variabilidad dentro de la clase, es decir, trata de encontrar un nuevo subespacio reducido que proporciona la mejor separación entre las diferentes clases en los datos de entrada. Esta idea base se aplica al reconocimiento facial de una manera similar a la aplicación de la PCA. Cada imagen de un rostro se considera un punto en un espacio de nxm dimensiones, siendo n el ancho y m el alto de la imagen. Entonces, LDA se aplica a los datos para obtener los nuevos vectores de base, llamados los Fisherfaces. Las imágenes de los rostros se proyectan sobre esta base, donde se realiza la comparación. Podemos usar de nuevo los dos métodos de vecinos más cercanos descritos anteriormente para el proceso de comparación.

1.2. Ingeniería de Características

En su forma más básica, una imagen se puede representar mediante un arreglo bidimensional que contiene una cantidad determinada de píxeles de ancho y otros de alto. Cada uno de esos píxeles corresponde a la intensidad del nivel de gris o de las componentes del color en ese punto de la imagen. Si bien, se pueden utilizar los píxeles como descriptores de imágenes, la información de la intensidad de nivel de gris o del color no es suficiente para reconocer un rostro en una imagen. Se requiere además información del entorno que sirva de alguna manera para representar las formas contenidas en una región de la imagen. En esta sección se estudian distintos procedimientos mediante los cuales se pueden generar descriptores de imágenes a partir de la información de los píxeles que han alcanzado buenos resultados en el reconocimiento de imágenes, como lo son: los descriptores de Histogramas de la Orientación de los Gradientes (HOG), los Histogramas de Patrones Locales Binarios (HLBP) y los Descriptores Invariantes a la Escala (SIFT).

1.2.1. Histogramas de la Orientación de los Gradientes (HOG)

El gradiente calculado en un píxel de una imagen, proporciona la dirección del mayor cambio de intensidad alrededor del píxel. La información proporcionada por los gradientes suele ser utilizada como la base para la construcción de descriptores más complejos. Los Histogramas de la Orientación de los Gradientes *(Histogram of Oriented Gradients)* [7], son uno de los métodos que utilizan la información obtenida a partir del cálculo de gradientes para la representación de una imagen (Ver figura 1.2). Los descriptores HOG, se basan en la orientación del gradiente en áreas locales de una imagen. Se obtienen dividiendo la imagen en celdas, calculando el histograma de la dirección del gradiente en cada una de las celdas. Para mejorar la invarianza frente a cambios de iluminación o de sombras, se realiza una normalización del contraste de cada uno de los histogramas. La combinación de estos histogramas representa el descriptor, el cual es utilizado para entrenar el clasificador.

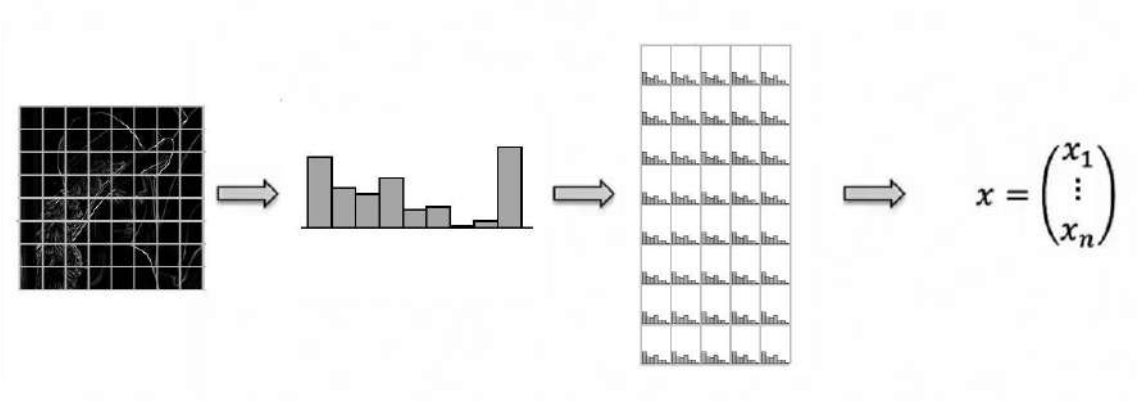


Figura 1.2: Descriptor Básico de HOG de una imagen [17].

Déniz et al. [4], proponen un enfoque basado en descriptores de HOG para el reconocimiento de rostros. Los autores proponen extraer descriptores HOG de una cuadrícula regular para compensar los errores en la detección de rasgos faciales debido a oclusiones, cambios de pose y de iluminación. La fusión de descriptores HOG en diferentes escalas, les permite capturar una estructura importante para el reconocimiento facial. Los autores utilizan algoritmos de reducción de la dimensionalidad como PCA para eliminar el ruido y hacer el proceso de clasificación menos propenso al sobreajuste, un efecto que hace que el éxito del modelo se incremente al responder a las muestras de entrenamiento mientras que empeore al utilizarse nuevas muestras.

1.2.2. Patrones Locales Binarios (LBP)

Uno de los descriptores más utilizado en el reconocimiento de objetos es el denominado, Patrones Locales Binarios *(Local Binary Patterns)* [23]. Para el cálculo del descriptor se requiere un código LBP para cada píxel de la imagen, que se basa en definir una vecindad del píxel y luego ir comparando los niveles de gris de este píxel central con el de sus vecinos. En este caso definiremos la vecindad como todos aquellos píxeles que tocan el píxel central. Los números, dos, siete, siete, uno, ocho, siete, dos, ocho (Ver figura 1.3), representan los niveles de gris en cada uno de estos píxeles vecinos. Para cada píxel vecino debemos asociar un valor de uno o cero: si el valor del nivel de gris del vecino es mayor o igual del nivel de gris del píxel central entonces a ese vecino se le asocia un uno y en otro caso un cero. Como hay ocho vecinos, podemos construir un byte donde cada bit tendrá un valor de cero o uno según el resultado de aplicar esta regla anterior a cada uno de los vecinos. Como

esto es un código binario, lo podemos traducir a un número decimal. Para el ejemplo de la figura 1.3, el código decimal que se obtiene para el píxel central es de 182. Por tanto, el código LBP de ese píxel central es 182.

Figura 1.3: Código LBP de un píxel [17].

Utilizando LBP se ha pasado de un valor en la imagen original de gris que va de cero a 255 a otro valor equivalente en la imagen de los LBP en el mismo rango. La diferencia entre ambos valores es que el valor original es el que captura la cámara que se ha utilizado, mientras que en la imagen LBP se codifica la relación del nivel de gris que había en el píxel central en la ventana superpuesta a la imagen original con el nivel de gris de los píxeles vecinos. Por tanto, la información es de más alto nivel, no solo es información del propio píxel, sino de cómo se relaciona con los píxeles vecinos. Dada una imagen de entrada, podemos producir una imagen de salida donde en cada píxel tengamos un código LBP, mediante un barrido de la imagen original calculando los valores LBP correspondientes y asignándolos al píxel central. Los LBP contemplan distintas invarianzas, una de ellas es la invarianza es a los cambios mono tónicos del nivel de gris y otra es a la invarianza a la traslación [17].

Existen algunas variantes de los códigos LBP, entre ellas una llamada código ULBP *(Uniform Local Binary Patterns)*. El valor de los códigos LBP va de cero a 255, es decir que hay 256 posibles valores. Si definimos U como el número de transiciones de cero a uno o de uno a cero (Ver figura 1.4), en la representación binaria del código LBP podemos encontrar valores de U=0 cuando no hay transición, U=2 si hay dos cambios y así sucesivamente. Sin embargo, se ha observado que cuando calculan códigos LBP en imágenes reales, hay muchos patrones que tienen

muy poca probabilidad ser encontrados, y por eso se decide reducir el número de patrones a solo aquellos que sí aparecen con regularidad.

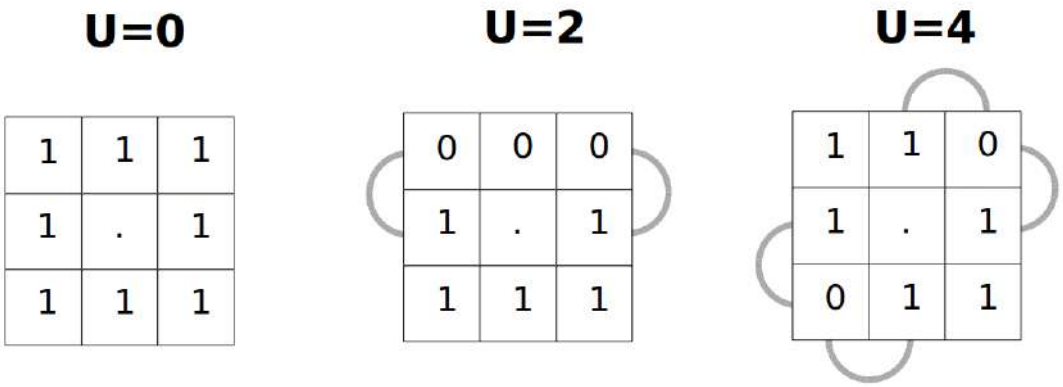


Figura 1.4: Cálculo de distintos valores de U [17].

Posteriormente a los patrones cuyo valor de U es cero o cuyo valor de U es dos, simplemente se les reasigna un código. Todos los patrones cuyo U no es ni cero ni dos, a todos ellos se les asigna también un código arbitrario, pero es el mismo para todos. Con este procedimiento se pasa de 256 patrones en el LBP normal a 59 patrones del LBP uniforme. Tenemos que 58 de estos códigos, son aquellos que tienen valor U igual a cero o U igual a dos y el código adicional es el que se ha reservado para el resto de patrones, es decir aquellos cuyo valor de U no era ni cero ni dos.

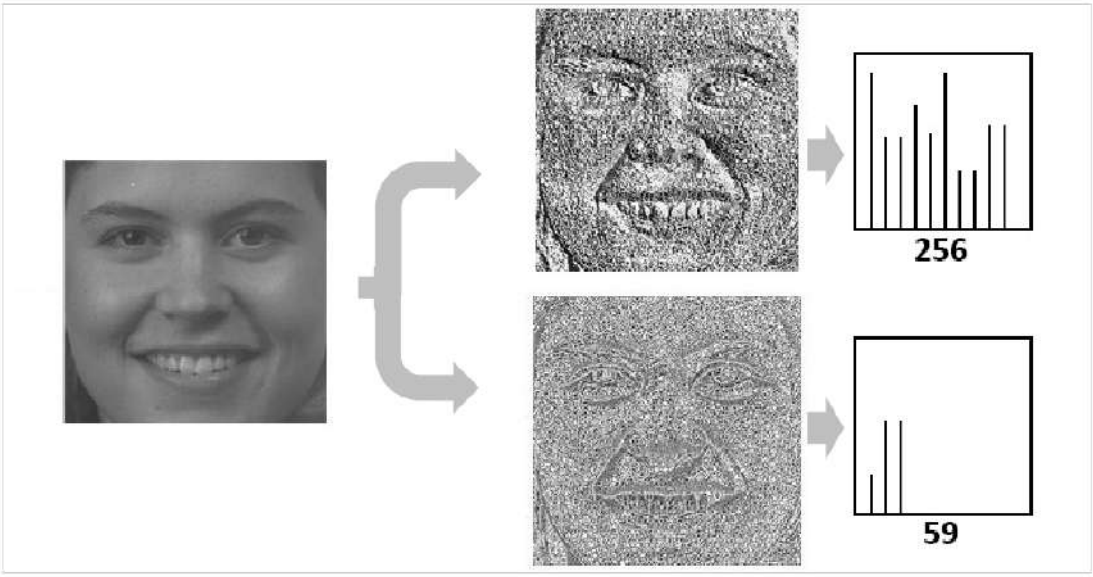


Figura 1.5: Histogramas LBP- Uniformes.

La figura 1.5 muestra un ejemplo de la diferencia entre el código LBP que habíamos visto hasta ahora y el ULBP a partir de la misma imagen y sus correspondientes histogramas. La cantidad de entradas en el histograma se reduce significativamente en los ULBP. Aunque los códigos LBP y los ULBP podrían ser utilizados para definir descriptores de imágenes que posteriormente se utilizarán para clasificar las mismas, no suelen utilizarse directamente en el reconocimiento de objetos pero sí sus histogramas. La figura 1.6 muestra una imagen de 300 columnas y 300 filas y su correspondiente con los códigos LBP.

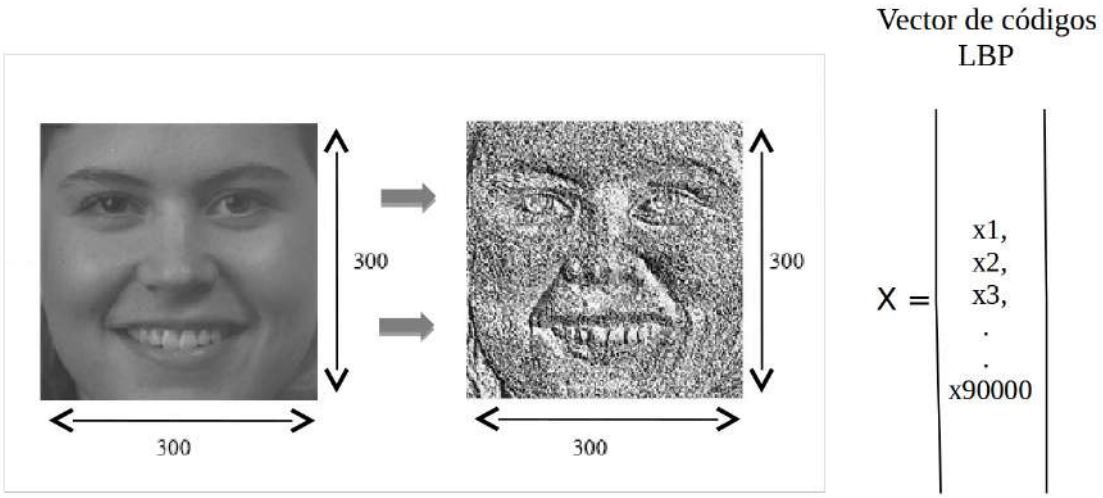


Figura 1.6: Vector de códigos LBP.

Una vez calculados los códigos LBP o ULBP, se divide la imagen en bloques y se calcula un histograma para cada uno de los bloques por separado. El descriptor de toda la ventana se obtiene al concatenar todos estos histogramas. Estos histogramas son normalizados en un rango entre cero y uno; esta normalización se hace individualmente para cada bloque, aunque los bloques podrían ser disjuntos, para tener descriptores más robustos, se añade cierta redundancia mediante bloques solapados. Los descriptores LBP son tolerantes a los cambios mono tónicos de iluminación, son computacionalmente eficientes, altamente discriminantes y no requieren una normalización de la escala de grises.

T. Ahonen et al. [22], presentan un enfoque para el reconocimiento de rostros en el cual consideran la información de la forma y la textura. El área del rostro es dividida en pequeñas regiones a las que se les extraen descriptores LBP, sus correspondientes histogramas que son concatenados según el procedimiento explicado anteriormente. Sus experimentos demostraron la superioridad de esta técnica sobre métodos como Eingenfaces. Las pruebas muestran su robustez utilizando diferentes expresiones, condiciones de iluminación y envejecimiento de los sujetos. Además de su eficiencia, su sencillez permite la rápida extracción de características.

1.2.3. Transformación de Característica Invariable a la Escala (SIFT)

Uno de los algoritmos de extracción de características más conocido fue propuesto por David Lowe en 2004, recibe el nombre de Transformación de Característica Invariable a la Escala *(Scale Invariant Feature Transform)*. Este Algoritmo ofrece una serie de características importantes, como lo son la invarianza a los cambios de escala, rotación, punto de vista tridimensional e iluminación de los objetos. Además permite que a partir de una sola imagen se puedan extraer un gran número de características.

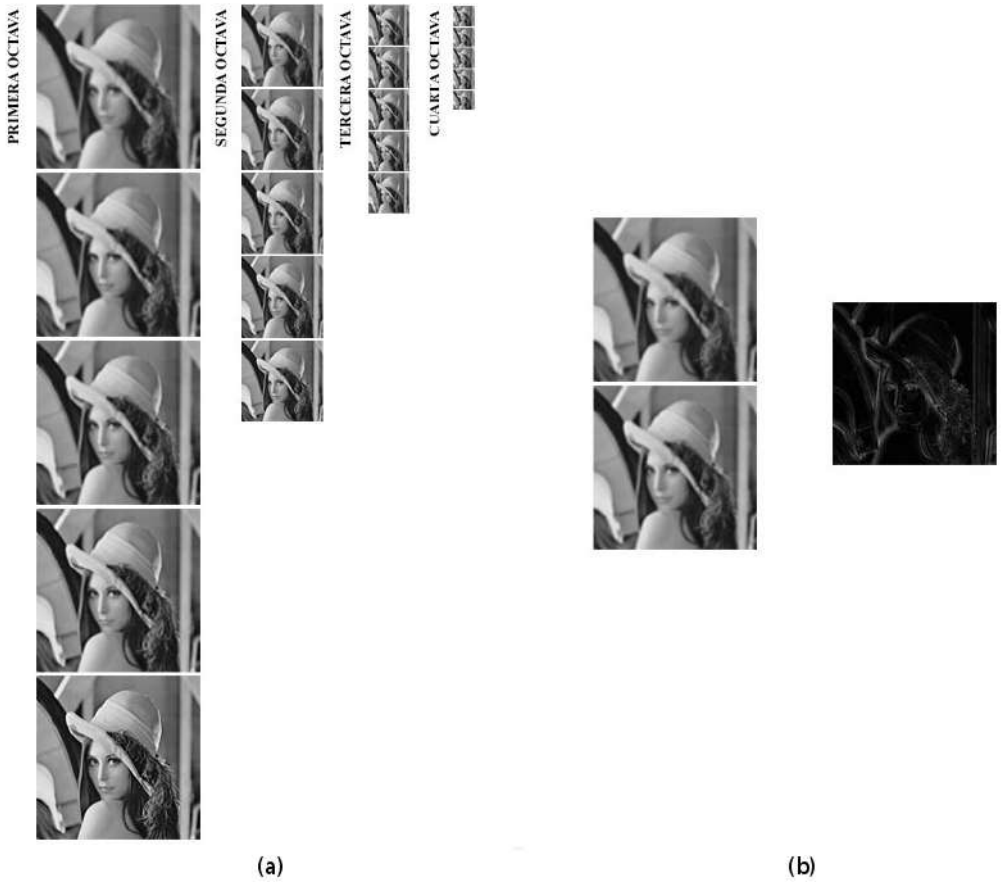


Figura 1.7: (a) Desenfoque Gaussinano, (b) Diferecia Gaussiana (DoG)

El descriptor SIFT sigue una estrategia de búsqueda de características locales robusta y se divide en cuatro partes: Detección Espacio-Escala, Localización de Puntos clave, Asignación de Orientaciones y la generación de descriptores de Puntos claves. En la detección Espacio-Escala, se aplica un desenfoque gaussiano a la imagen

progresivamente. Una secuencia progresiva de estas imágenes borrosas constituye lo que se conoce como una octava. Una nueva octava se forma cambiando el tamaño de la imagen original de la octava anterior a la mitad y desenfocando progresivamente (Ver figura 1.7). Lowe recomienda usar cuatro octavas de cinco imágenes para mejores resultados. Posteriormente se calcula la diferencia entre dos imágenes consecutivas en una octava, lo que se conoce como la diferencia de Gauss (DoG), (Ver figura 1.7). Utilizar DoG, añade el beneficio de que la imagen obtenida es también invariante a la escala. El segundo paso de este algoritmo consiste en la localización de puntos clave; para esto tenemos que iterar sobre cada píxel y comparar su valor no solo con todos sus ocho vecinos en esa imagen, sino también con las imágenes por arriba y por debajo en esa octava, que tienen nueve píxeles cada una. Se realizan un total de 26 comparaciones (Ver figura 1.8). Un punto es considerado un "punto clave"si se trata del más grande o el más pequeño de los 26 vecinos.

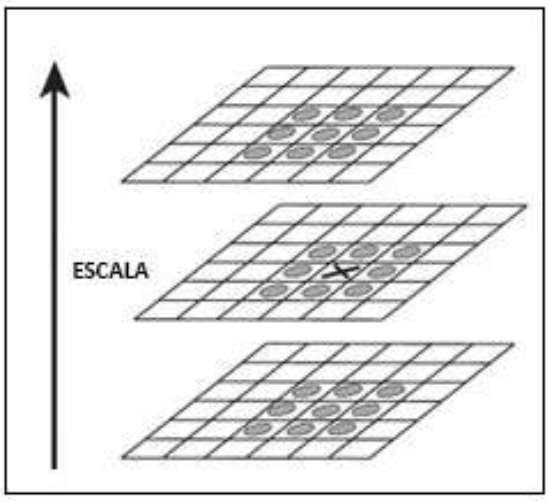


Figura 1.8: Localización de Puntos Claves.

Mediante el uso de las octavas con distintas escalas se introduce invarianza a la escala en el descriptor SIFT. Una vez que tenemos un conjunto de puntos clave el siguiente paso es asignarles una orientación a cada punto. La idea calcular la dirección de los gradientes y sus magnitudes para todos los píxeles alrededor de cada punto clave. Entonces podemos averiguar la orientación más prominente en esa región y asignar esta orientación para el punto clave. Todos los cálculos posteriores se realizan en relación con esta orientación lo que garantiza invarianza a la rotación.

Una vez obtenido el punto clave y su orientación debemos asignarle un descriptor. Este descriptor se construye tomando una ventana de 16x16 píxeles alrededor del punto clave y dividiendo está en celdas de 4x4 píxeles (Ver figura 1.9). Para cada celda de 4x4 se calcula un histograma de orientaciones de 8 columnas, discretizando los gradientes a intervalos de 45 grados. El descriptor consiste en el resultado de la concatenación de estos histogramas, dado que cada histograma tiene 8 valores el descriptor tendrá 128 valores, los cuales son normalizados posteriormente. Los descriptores SIFT son robustos a los cambios de orientación, escala, iluminación y perspectiva tridimensional.

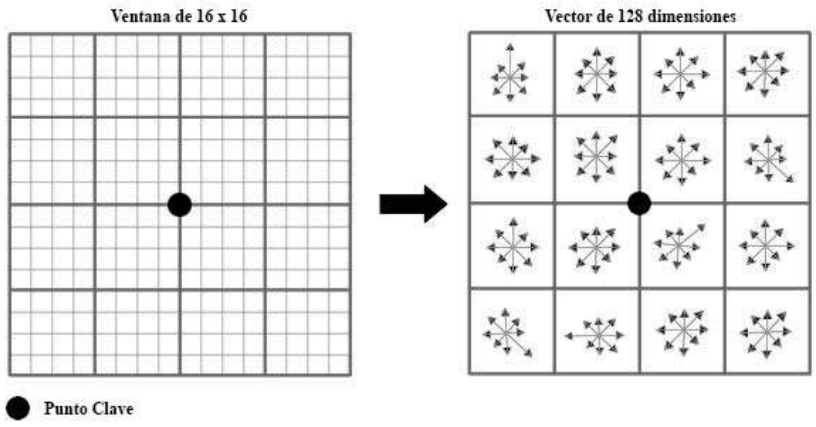


Figura 1.9: Construcción del descriptor SIFT.

Aly M. [6], propone un enfoque basado en SIFT que en comparación con los algoritmos de Eigenfaces [9] y Fisherfaces [10] [11], muestra una mayor precisión en el reconocimiento de rostros. En este enfoque se extraen las características SIFT de todos los rostros de la base de datos. Entonces, dada una nueva imagen de un rostro, las características extraídas de ese rostro se comparan con las características de cada rostro en la base de datos. El rostro en la base de datos con el mayor número de puntos coincidentes, es considerado el rostro más cercano, y se utiliza para la clasificación del nuevo rostro. Gautam et al. [8], propone un método donde se generan descriptores SIFT para cada imagen de entrenamiento y se calculan las características llamadas puntos clave, luego se entrena un clasificador de vecinos más cercanos y se realiza el reconocimiento de las imágenes del conjunto de pruebas.

Los resultados demuestran un desempeño robusto bajo diferentes condiciones de expresión, variaciones de pose, cambios de iluminación y oclusión parcial.

1.3. Aprendizaje de Características

Uno de los enfoques más utilizados en la actualidad para el reconocimiento de objetos, se basa en las Redes Neuronales Convolucionales (CNN). Estas redes utilizan un conjunto de capas de neuronas, localmente conectadas para extraer características de las imágenes que les son suministradas para su aprendizaje. Las características aprendidas por estas capas son suministradas a una red completamente conectada que se encarga de clasificar las imágenes a partir de los descriptores generados. Esta sección describe el funcionamiento de las CNN así como algunos trabajos relacionados.

1.3.1. Redes Neuronales Convolucionales

En el aprendizaje a partir de imágenes mediante redes neuronales artificiales (RNA), se presentan algunos inconvenientes específicamente cuando se trata de imágenes de gran tamaño. Con una imagen RGB de tamaño 32x32 tendremos que una sola neurona completamente conectada ubicada en la primera capa oculta tendría que calcular 3.072 pesos. Aunque esta cantidad parece manejable, es evidente que esta estructura resultaría costosa con imágenes mucho más grandes. Por ejemplo, una imagen RGB de tamaño 200x200 daría lugar a 120.000 pesos. Esta cantidad aumenta proporcionalmente al número de neuronas que tenga la capa.

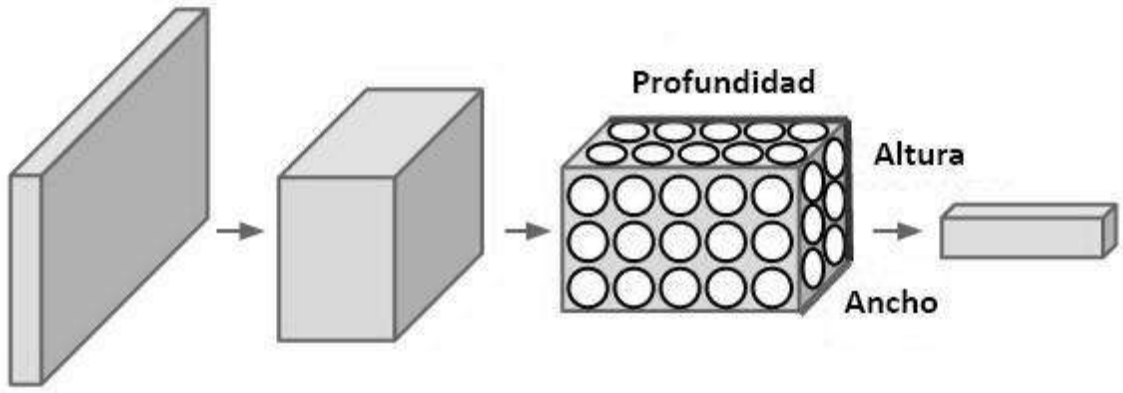


Figura 1.10: Volúmenes manejados por una CNN.

Las Redes Neuronales Convolucionales (CNN) [4], asumen que la entrada de la red está conformada por imágenes lo que condiciona la arquitectura de la red. Las capas de una CNN están conformadas por neuronas dispuestas en volúmenes de tres dimensiones: anchura, altura, profundidad (ver figura 1.10). Para imágenes RGB de 32 x 32 el volumen de entrada estará constituido por un volumen de 32x32x3. Además a diferencia de las RNA las neuronas de una capa solo se conectan a una pequeña región de la capa anterior. La salida está conformada por un único vector, donde cada componente representa la puntuación de cada clase.

Una CNN está formada por una secuencia de capas, donde cada una transforma un volumen de entrada en un volumen de salida. Existen tres tipos de capas principalmente: Convolucionales, Pooling y las Completamente Conectadas que son similares a las de una RNA. La figura 1.11, muestra un esquema de la estructura básica de una CNN; cada capa de la red provee una funcionalidad distinta. La capa de entrada almacena los píxeles de las imágenes y sus correspondientes canales RGB. Las capas de convolución están conectadas a regiones locales de la capa anterior y calculan para cada una el producto entre sus pesos y la pequeña región correspondiente. Las capas ReLU (Rectified Linear Unit), aplican una función de activación. las capas de pooling efectúan un submuestreo sobre las dimensiones espaciales y para finalizar una capa completamente conectada calcula la puntuación por clase, permitiendo de esta forma clasificar las imágenes de entrada.

A comienzos del año 2014, Facebook desarrolla un algoritmo al que denominó "DeepFace" [22] para el reconocimiento facial en ambientes no controlados a partir del trabajo realizado por Yaniv et al. [22]. El equipo fue liderado por Yan Lecum [2] y el algoritmo fue presentado en la conferencia de la IEEE (CVPR), sobre visión por computador y reconocimiento de patrones en Junio de 2014. El objetivo del proyecto es la verificación de rostros, que consiste en probar si dos imágenes pertenecen al mismo rostro. Esas imágenes fueron tomadas en entornos no controlados, como se podría esperar de una base de datos de fotos de la red social Facebook. De manera general este algoritmo consta de cuatro etapas: Detección, Alineación,

Representación y Clasificación (Ver figura 1.12).

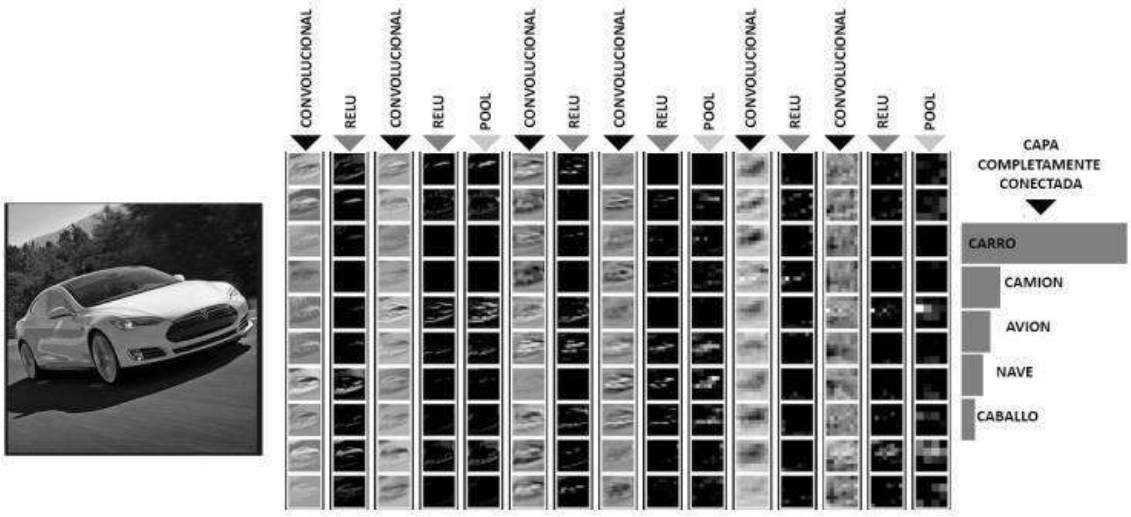


Figura 1.11: Arquitectura de una CNN. El volumen inicial almacena los píxeles de la imagen en bruto (izquierda), y el último volumen las puntuaciones de las clases (derecha) [4].

El problema de alineación se ha abordado mediante el empleo de un modelo tridimensional del rostro humano y una transformación afín. La representación del rostro se obtiene utilizando nueve capas de una red neuronal profunda. En el entrenamiento se utilizaron cuatro millones de fotos de rostros que pertenecen a casi 4.000 personas todos usuarios de Facebook. Se obtuvo una tasa de éxito del 97,23 %, que está muy cerca de rendimiento estimado de un 97,53 % para los humanos.

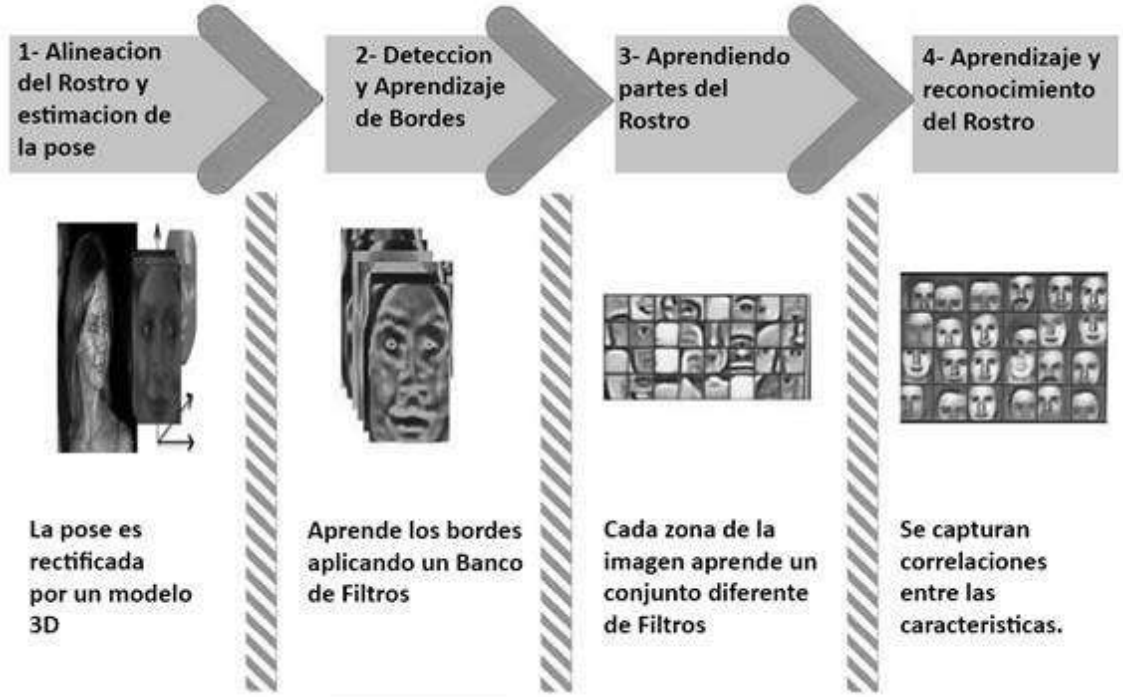


Figura 1.12: Etapas del algoritmo DeepFace: Detección, Alineación, Representación y Clasificación

1.4. Enfoques Híbridos.

La mayoría de los algoritmos de reconocimiento facial utilizan una sola representación del rostro en la imagen. Los descriptores son restrictivos al captar la información contenida en una imagen. Algunos trabajos combinan varios descriptores con el fin de aumentar la capacidad de encontrar patrones en las imágenes. Los descriptores son utilizados para entrenar algoritmos de aprendizaje como SVM o MLP, que al ser combinados logran un mejor desempeño. Es frecuente la reducción de la dimensionalidad de las características mediante el Análisis de Componentes Principales (PCA). Los sistemas multimodales que combinan múltiples descriptores también se han empleado en otras disciplinas como la detección de emociones, sentimientos y expresiones faciales [24].

Zhang et al. [25], desarrollan un trabajo para la detección de expresiones faciales

en donde extraen un número fijo de características SIFT de algunos hitos faciales (boca, ojos, nariz), y construyen una matriz de características a partir de los vectores de características SIFT extraídos que se utilizan como entrada para una CNN. El tamaño de matriz es de "m x n" donde "m" es el número de características SIFT, y "n" es el tamaño de cada característica. Otro trabajo basado en la combinación de descriptores LBP, SIFT y en redes neuronales convolucionales (CNN), es propuesto por Sun et al. [26] para el reconocimiento de expresiones faciales. Ellos utilizan los descriptores de características SIFT, LBP y el descriptor aprendido por una CNN para entrenar máquinas de soporte vectorial SVM. Finalmente se combinan las salidas de todos los clasificadores para obtener un clasificador más robusto.

Los métodos de ensamblaje de clasificadores como el Bagging y el Boosting, se han utilizado en el reconocimiento de expresiones faciales. Varios enfoques populares, como el que utiliza Kahuo et al. [27], emplean CNN para el análisis de vídeo combinado con la información de audio y han obtenido mejores rendimientos que el alcanzado por modelos no combinados.

Capítulo 2

Desarrollo de la Solución

En este capítulo describimos el desarrollo de la solución planteada. Utilizamos la metodología CRISP-DM por lo que se divide en 5 secciones. La sección 2.1 describe el conjunto de datos utilizados y sus características generales. En la sección 2.2 el proceso de extracción de características mediante los descriptores HLBP, HOG y SIFT. Luego en la sección 2.3 se muestran el entrenamiento de los distintos algoritmos de aprendizaje automático. La sección 2.4 muestra la interpretación de los resultados de la evaluación. Por último en la sección 2.5 describimos el funcionamiento de prototipo de aplicación desarrollado a partir del modelo obtenido mediante el procedimiento propuesto. La base de datos utilizada en el entrenamiento de los modelos es la base de datos FEI, la cual describimos a continuación.

2.1. Recopilación e Integración de los datos

En este trabajo utilizaremos la base de datos de imágenes FEI (Fundação Educacional Inaciana). Una base de datos brasileña que contiene un conjunto de imágenes faciales tomadas entre junio de 2005 y marzo de 2006 en el Laboratorio de Inteligencia Artificial de la Universidad de FEI de Sao Bernardo do Campo, Sao Paulo, Brasil. Se trata de un conjunto de datos balanceado que contiene imágenes de 200 personas, cada una de ellas con 14 imágenes, para un total de 2800 imágenes. El repositorio esta formado por imágenes en colores RGB adquiridas sobre un fondo blanco homogéneo en posición frontal con rotaciones del perfil en un rango de [-90,90].

Clases	Imágenes	Ancho	Alto
200	2.800	640	480

Cuadro 2.1: Datos estadísticos del conjunto de imágenes FEI.

La escala puede variar alrededor del 10 % y el tamaño original de cada imagen es de 640x480 píxeles. Todos los rostros están representados principalmente por estudiantes y personal de la FEI, entre 19 y 40 años de edad, con apariencia, peinado y adornos distintos. El número de sujetos masculinos y femeninos es exactamente el mismo e igual a 100. La figura 2.1 muestra algunos ejemplos de las imágenes de la base de datos de rostros FEI. Es importante señalar que aunque las imágenes fueron adquiridas en un fondo uniforme, no son frontales y presentan variaciones en la pose, expresión y escala que se corresponden con ambientes no controlados.

Figura 2.1: Ejemplo de las imágenes de la base de datos FEI.

Para cada una de las clases tenemos un total de 14 imágenes por lo que el conjunto se encuentra balanceado. A continuación describiremos el tratamiento que han recibido estas imágenes para obtener los descriptores de cada uno de los modelos que presentamos en este trabajo de grado.

2.2. Preparación de los datos

La solución planteada en este documento se basa en la combinación de cuatro clasificadores, cada clasificador es entrenado con una representación distinta de los datos. La figura 2.2 muestra un diagrama esquemático de la solución, en ella vemos que hay una etapa de extracción de características donde se extraen de la imagen de entrada los descriptores HOG, HLBP y SIFT. La imagen cruda (RAW), es también

preprocesada antes de ser utilizada para entrenar un clasificador basado en una Red Neuronal Convolucional. En este capítulo explicaremos el procedimiento que se utilizó para la extracción de los descriptores de cada uno de las imágenes, así como las transformaciones que se aplicaron a la imagen de entrada.

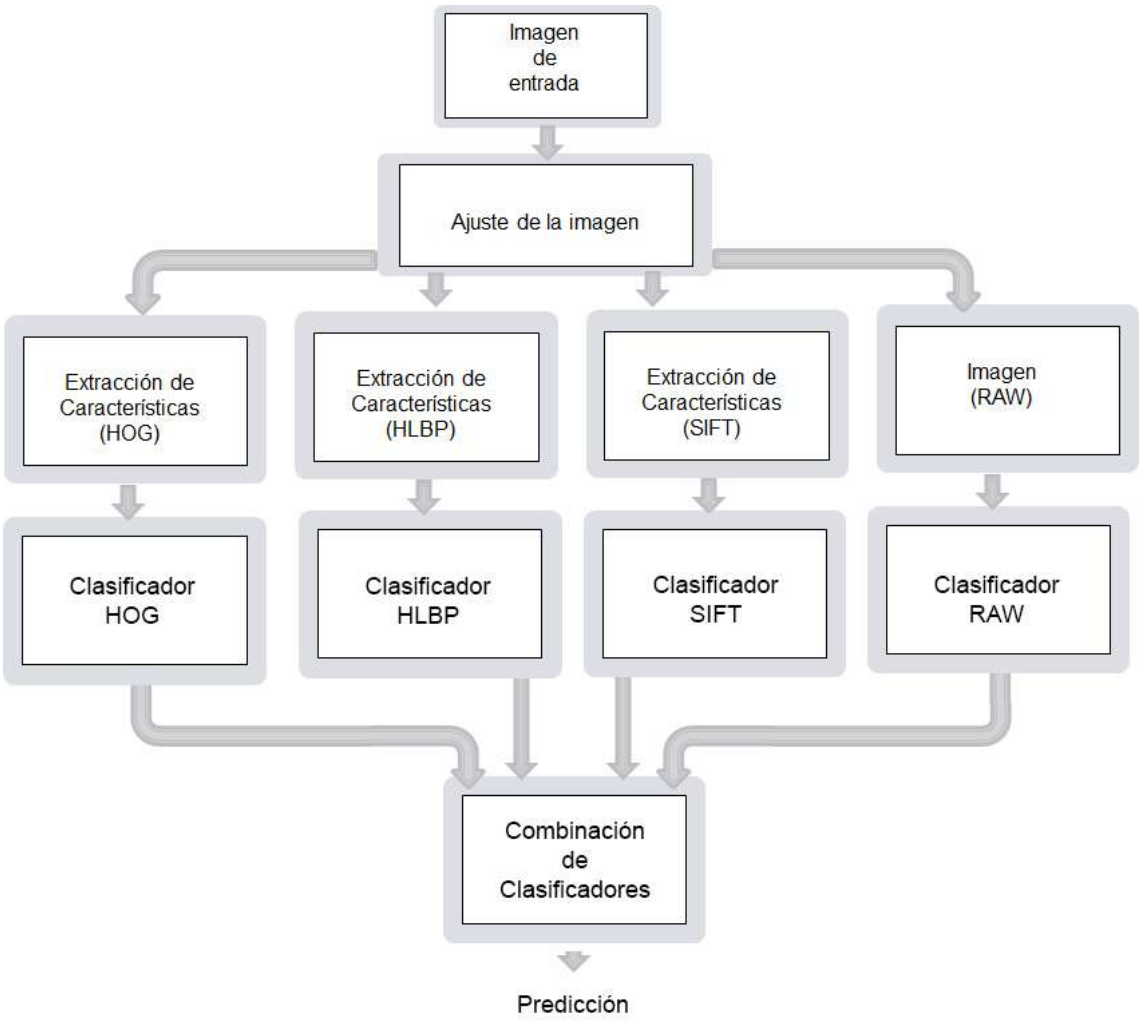


Figura 2.2: Diagrama esquemático de la solución propuesta.

Además veremos cómo, a partir de la información extraída de las imágenes, se crean las vistas minables necesaria para el entrenamiento de los algoritmos de aprendizaje automático de la etapa siguiente. La base de datos FEI es dividida inicialmente en dos conjuntos, cada uno con 80 % y 20 % de los datos, respectivamente. Para ambos conjuntos se extraen los descriptores HOG, HLBP, un descriptor basado

en SIFT y una imagen RAW ajustada según explicaremos en el apartado siguiente. El primer conjunto de datos lo llamaremos de *Entrenamiento* , estos datos serán utilizados para entrenar todos los modelos, así como también para la validación de los mismos. Los datos del segundo conjuntos no serán utilizados para entrenar los modelos, ni tampoco durante el proceso de validación cruzada, estos datos serán utilizados para ajustar los parámetros de la red durante el entrenamiento midiendo con ellos la capacidad de generalización del algoritmo. La capacidad de generalización del algoritmo nos permite determinar como se comporta el modelo con respecto a datos no conocidos, es decir que no han sido vistos durante el entrenamiento. El conjunto de datos conformado por el 20 % del las imágenes de la base de datos FEI lo llamaremos datos de Validación , pero unicamente seran utilizados para determinar la capacidad de generalización de nuestro proceso de entrenamiento.

2.2.1. Preprocesamiento de imagenes, RAW

Las imágenes utilizadas en este trabajo, ver figura 2.1, forman parte del conjunto de datos público que los investigadores de diversas universidades han desarrollado con el fin de promover la investigación en el área del reconocimiento de rostros. Muchas de estas imágenes requieren un preprocesamiento previo para su uso. Entre los factores que deben ser tomados en cuenta se encuentran la determinación de la región de interés y el ajuste de la calidad. Como podemos observar las imágenes de la base de datos FEI a pesar de que fueron adquiridas en un ambiente semi-controlado, contienen variaciones en la posición, escala, iluminación y apariencia; el primer paso en el pre-procesamiento de estas imágenes consistirá en el recorte del área útil tal y como se muestra en la figura 2.3.

Figura 2.3: Imágenes recortadas del conjunto de datos FEI.

Adicionalmente aplicaremos a estas imágenes un ajuste del histograma que hará

nuestros algoritmos más robustos a los cambios de luminosidad como se muestra en la figura 2.4. El ajuste consistió en en escalar los valores de intensidad de los píxeles en el rango de 0 a 255.

Figura 2.4: Imágenes con el histograma ajustado.

Además aplicamos a estas imágenes una normalización en la escala de 0 - 1 para mejorar el aprendizaje de los algoritmos utilizados. La vista minable está conformada por las imágenes, recortadas y escaladas, a 50 píxeles de ancho por 50 píxeles de alto. El cuadro 2.2 muestra un resumen de las características de la vista minable para las imágenes RAW.

Base de Datos	Bloque	Ancho	Alto	Canales	Bins	Rango	Tam. Desc.
FEI	1	50	50	3	256	0 - 1	2.500

Cuadro 2.2: Resumen estadístico de la vista minable.

2.2.2. Histogramas Locales de Patrones Binarios, HLBP

La extracción de histogramas locales de patrones binarios tiene sus raíces en el análisis de texturas. La idea básica de los patrones binarios locales es resumir la estructura local en una imagen, comparando cada píxel con su vecindad. Para cada una de las imágenes de los conjuntos seleccionados en la etapa previa se procede a extraer sus códigos LBP, tal como se muestra en la figura 2.5.

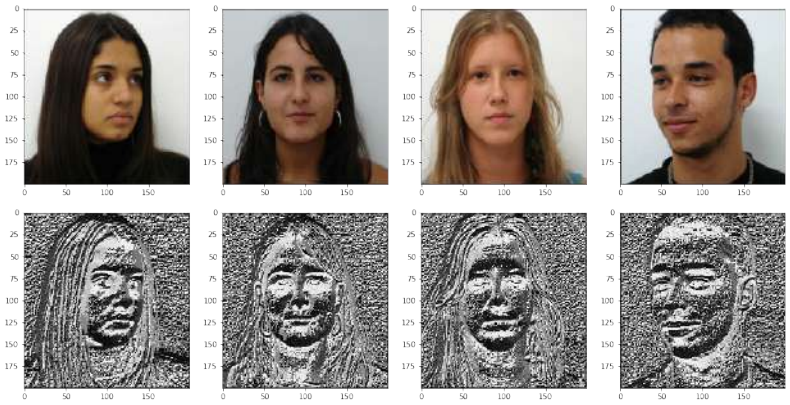
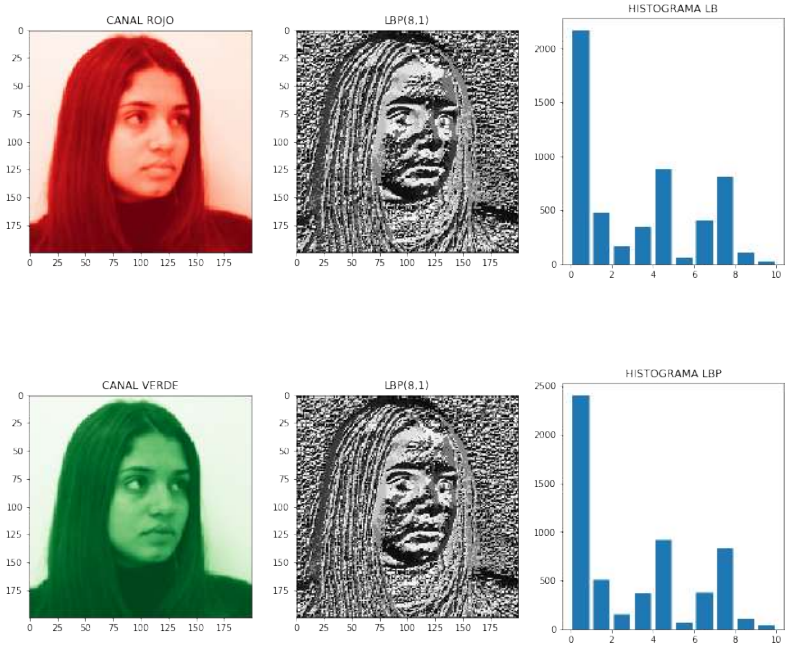


Figura 2.5: Códigos LBP para imágenes de la base de datos FEI.

Se han separado las imágenes en sus canales RGB, debido a que calcularemos estos descriptores para cada uno de los canales para aprender los patrones presentes en cada uno de los ellos. La información de color en el dominio del reconocimiento de rostros es relevante, porque el color de la piel se encuentra en un rango bastante definido y podría proporcionar información relevante para el reconocimiento de rostros. La figura 2.6, muestra los histogramas HLBP para una imagen de la base de datos FEI en todos sus canales.



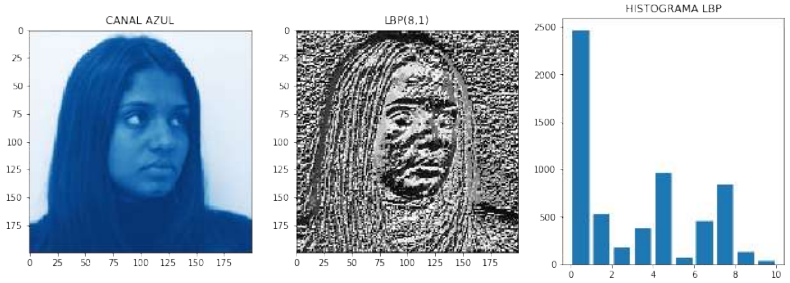


Figura 2.6: Imágenes de códigos LBP por canal.

Una vez calculados los códigos LBP para cada uno de los píxeles y sus correspondientes canales debemos obtener una representación de la imagen global utilizando estos códigos LBP. Para dar localidad al descriptor se divide la imagen en bloques de NxM píxeles y se calcula el histograma para cada uno de ellos, este nuevo descriptor tendrá información local de los distintos parches.

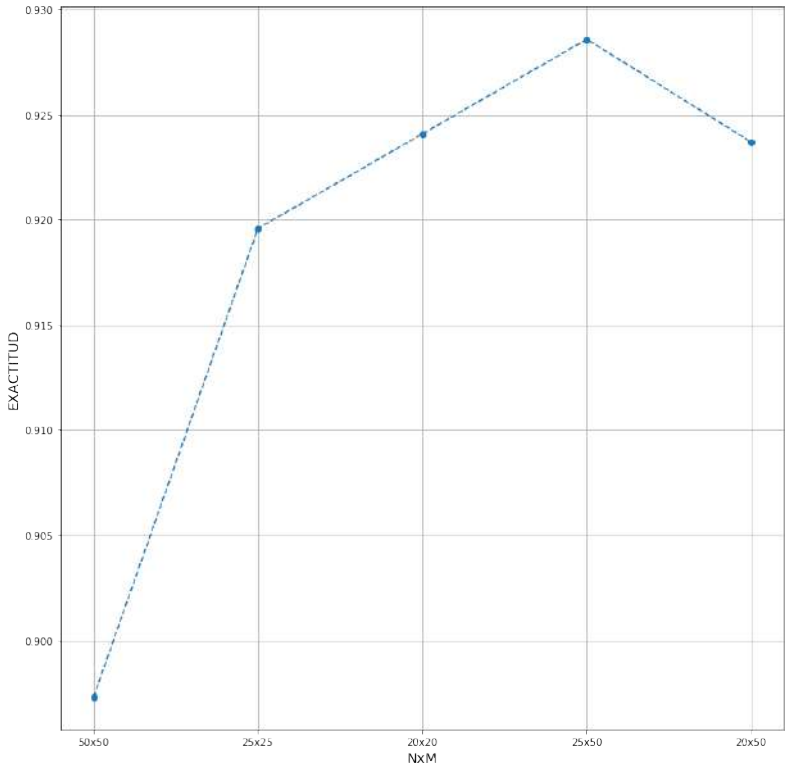


Figura 2.7: Exactitud del modelo con distintos valores de N y M

Los mejores valores de N y M se obtienen entrenando un algoritmo de aprendizaje automático con descriptores de distintos tamaños. La figura 2.7 muestra la exactitud del modelo para distintos tamaños de los parches. En el caso particular de las imágenes de la base de datos FEI, se obtiene una mayor exactitud utilizando

bloques de 25x50, por lo que se utilizó este valor para el cálculo de los descriptores HLBP. Es importante señalar que los descriptores son extraídos a partir de imágenes de 200x200 píxeles.

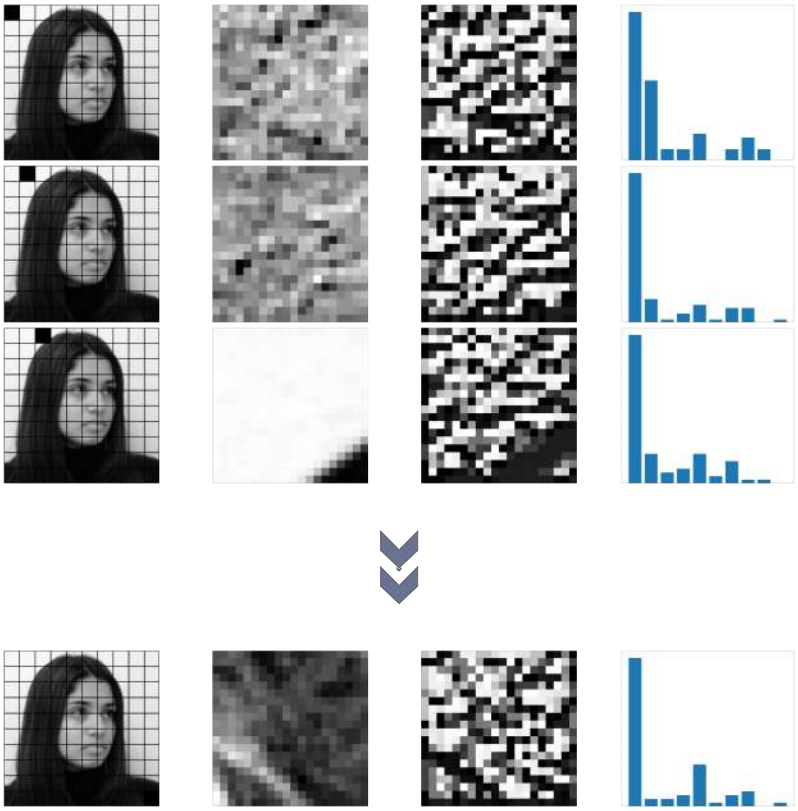


Figura 2.8: Generación de Histogramas Locales

La figura 2.8, ilustra como se obtiene el descriptor HLBP para cara uno de los canales RGB. Una vez obtenidos los vectores correspondientes a cada canal se concatenan en un solo descriptor.La figura 2.9 muestra el resultado de calcular los histogramas LBP sobre bloques de tamaño fijo. Para cada canal de la imagen se calcula la representación LBP, y para cada bloque el correspondiente histograma. Los histogramas son concatenados de izquierda a derecha y de arriba hacia abajo. El vector de características resultante contiene información espacial, de textura y color de toda la imagen.

Figura 2.9: Combinación de los Histogramas LBP.

Los histogramas de patrones locales binarios miden la frecuencia en la que se encuentran los patrones en la imagen, por lo tanto se trata de números enteros. Las redes neuronales obtienen mejores resultados cuando los datos están normalizados entre cero y uno. Los valores obtenidos son normalizados en este rango; así los números que conforman el descriptor estarán en este rango. La vista minable se obtiene del cálculo de los descriptores para cada una de las imágenes de la base de datos utilizada.

Bloque	Ancho	Alto	Canales	Bins	Rango	Tam. Desc.	PCA Desc.
8x4	200	200	3	128	0 - 1	12.288	479

Cuadro 2.3: Resumen estadístico de la vista minable LBP.

Para cada canal se calcularon los códigos LBP y sus histogramas, pero se observó que entre las imágenes LBP obtenidas existe gran similitud, por lo que se infirió que hay gran cantidad de información correlacionada. Además el descriptor tiene un gran tamaño (4096x3). Se utilizó la técnica de reducción de la dimensionalidad conocida como PCA para obtener un menor número de dimensiones, para lo cual indicamos al algoritmo que retorne un número de características tales que la variabilidad de la data sea igual al 95 %. El cuadro 2.3 muestra la cantidad de componentes que arroja PCA, así como un resumen estadístico de nuestra vista minable. Para el cálculo de los descriptores HLBP se utilizan las imágenes con un tamaño de 200x200, se seleccionó esta escala debido a que experimentalmente se obtuvieron mejores resultados que a

escalas más pequeñas, y métricas equivalentes para valores mayores.

2.2.3. Histogramas de Orientaciones de los Gradientes, HOG

En el descriptor de características de HOG, la distribución de las direcciones de los gradientes se utilizan como características. Los gradientes de una imagen son útiles porque la magnitud de los gradientes es grande alrededor de los bordes y esquinas debido a los cambios abruptos de intensidad. Los bordes y esquinas contienen mucha más información sobre la forma del objeto que las regiones planas.

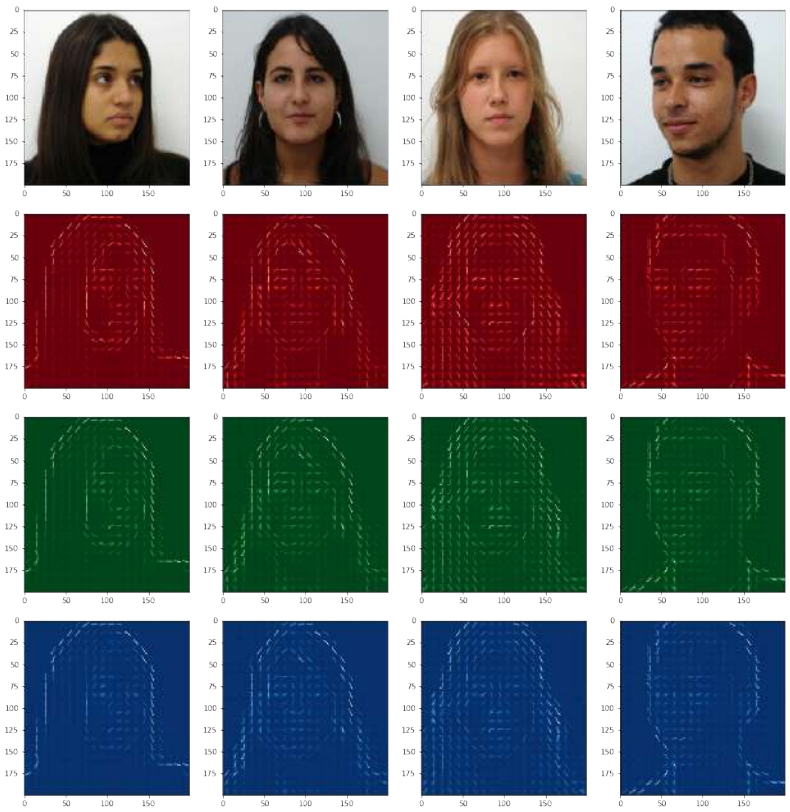


Figura 2.10: Códigos LBP para imágenes de la base de datos FEI.

La figura 2.10 muestra imágenes ilustrativas de los descriptores HOG generadas a partir del conjunto de datos FEI, separadas por canal. Si bien el resultado por canal parece ser similar, existen detalles del comportamiento de los gradientes que es detectado por el descriptor con menor o mayor intensidad según el canal.

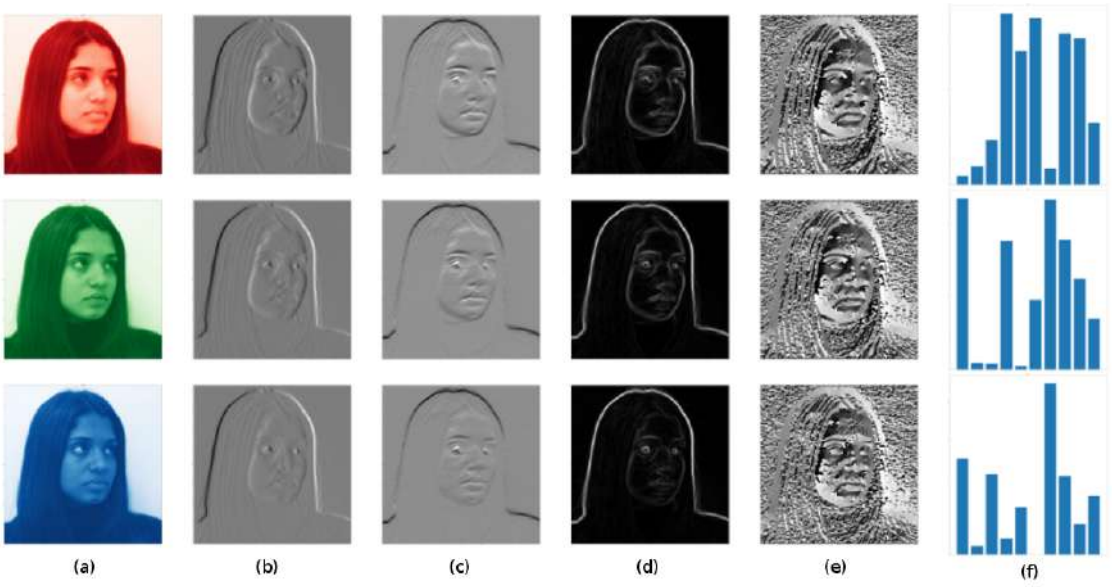


Figura 2.11: Componentes del gradiente por canal: (a) Imagen del canal,(b) Gradiente en X, (c) Gradiente en Y,(d) Magnitud del gradiente, (e) Angulo del Gradiente, (f) Histograma global LBP del canal.

En la figura 2.11 vemos que el gradiente horizontal alcanza una mayor intensidad (color más claro) en líneas verticales y el gradiente vertical en las horizontales. La magnitud del gradiente se incrementa cuando hay un cambio brusco en la intensidad. Ninguno de los gradientes crece en las regiones donde no hay cambio de intensidad en la imagen. La imagen resultante del cálculo de los gradientes eliminó una gran cantidad de información que no es esencial como, por ejemplo, el fondo de color constante, pero resaltó los contornos. Como se observa en la imagen de la magnitud del gradiente aún se distinguen rasgos importantes de la persona.

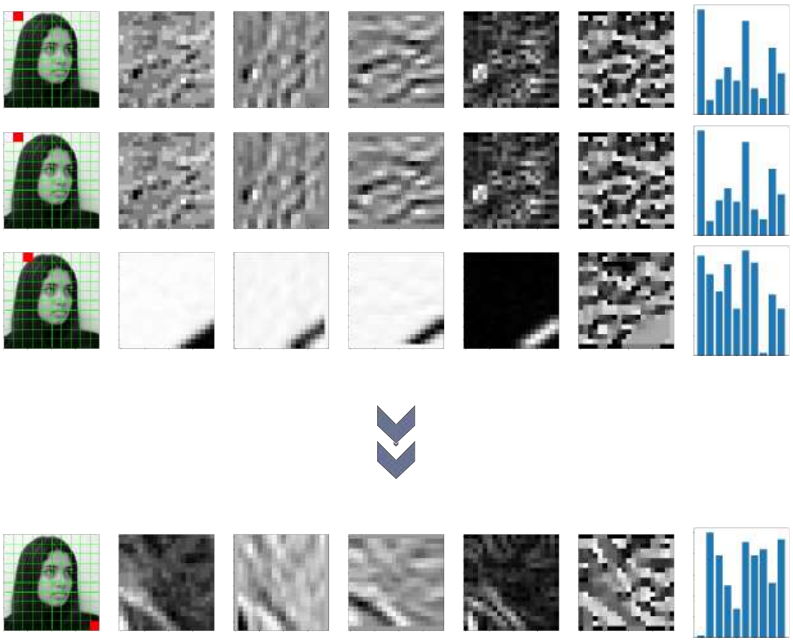


Figura 2.12: Cálculo de descriptores locales HOG.

En el caso de los Histogramas de Gradientes (HOG) el descriptor también debe incorporar información local. Para calcular el descriptor aplicamos sobre la imagen un filtro sobel en dirección horizontal y vertical, posteriormente calculamos la magnitud y la dirección del gradiente para cada píxel. En la figura 2.12 se muestran los resultados relacionados al cálculo del gradiente localizado; el histograma se obtiene contabilizando la frecuencia de los ángulos en cada celda y ponderando por su magnitud. Este histograma contiene información local debido a que se construye en base a una rejilla localizada sobre la imagen. El descriptor se obtiene al concatenar los descriptores de 9 valores de todos los bloques. El tamaño del descriptor será entonces, proporcional a las dimensiones de la imagen original.

La imagen se divide en 20 bloques horizontales y 20 verticales, la dimensión del bloque es un parámetro de diseño que se obtiene a partir de la escala de las características que estamos buscando. La figura 2.13 muestra una gráfica de la exactitud alcanzada por un clasificador para distintos tamaños del bloque, el valor de 10 obtiene una mayor exactitud que otros valores utilizados experimentalmente. El histograma se calcula para cada uno de estos bloques y por cada canal como muestra la figura 2.14.

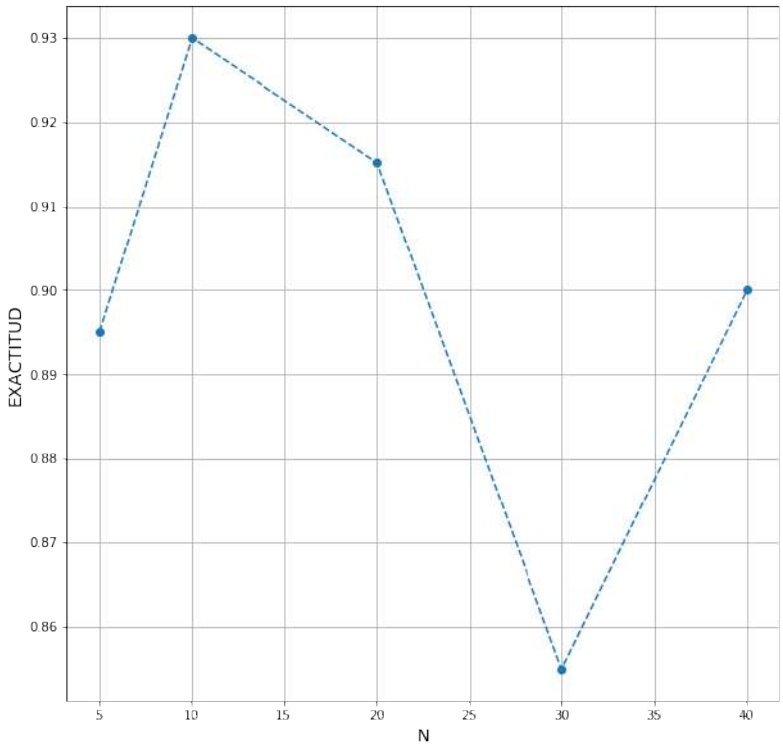


Figura 2.13: Exactitud para distintos tamaños del bloque para el descriptor HOG.

Una de las razones importantes para usar un descriptor de características es que proporciona una representación compacta, es decir, la dimensión del descriptor suele ser más pequeña que la dimensión de la imagen completa. Además el cálculo de un histograma sobre un parche hace que esta representación sea más robusta al ruido, los gradientes individuales pueden tener ruido, pero un histograma de un bloque hace que la representación sea mucho menos sensible al ruido.

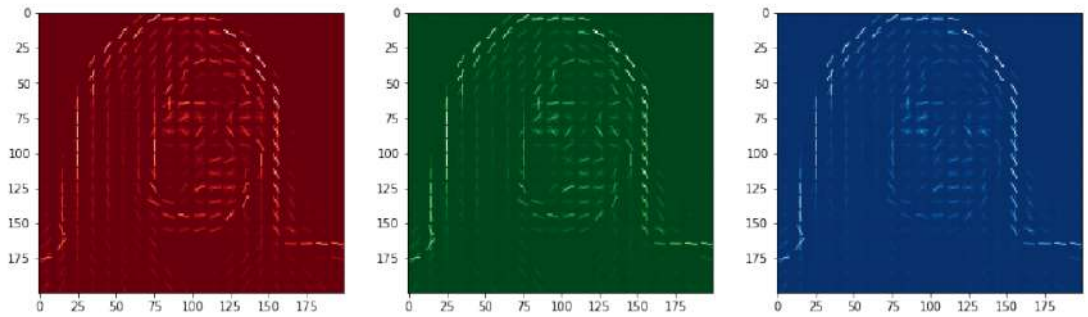


Figura 2.14: Descriptores HOG por canal RGB.

Los descriptores HOG fueron utilizados inicialmente para la detección de peatones. Los bloques permiten capturar suficientemente características interesantes para

el reconocimiento facial, como por ejemplo, la forma de la parte inferior de la cabeza o de la parte superior. El histograma es esencialmente un vector de 9 valores correspondiente a ángulos entre 0 y 180 en intervalos 20 grados. El cuadro 2.4 muestra un resumen de las características de la vista minable obtenida. Se ha generado (a partir de una imagen de tres canales RGB y 160.000 píxeles) un descriptor HOG de apenas 10.800 valores, lo que representa una disminución considerable de la dimensionalidad de los datos de entrenamiento en comparación con la imagen RAW.

Bloque	Ancho	Alto	Canales	Bins	Rango	Tam. Desc.	PCA Desc.
20 x 20	200	200	3	9	0 - 1	10.800	1.243

Cuadro 2.4: Resumen estadístico de la vista minable HOG.

Si observamos las imágenes HOG obtenidas en cada canal podemos apreciar su similitud, por lo que inferimos que hay gran cantidad de información correlacionada, además el descriptor tiene un gran tamaño 10.800. Utilizaremos PCA para obtener un menor número de dimensiones. Para ellos indicaremos al algoritmo que retorne un número de características tales que la variabilidad de la data sea igual al 95 %. El cuadro 2.4, muestra la cantidad de componentes que arroja PCA, así como un resumen estadístico de nuestra vista minable. Para el cálculo de los descriptores HOG se utilizan las imágenes con un tamaño de 200x200. Se seleccionó esta escala debido a que experimentalmente se obtuvieron mejores resultados que a escalas más pequeñas, y métricas equivalentes para valores mayores.

2.2.4. Descriptores basados en SIFT

Las descriptores SIFT son descriptores basados en puntos claves y son utilizados en distintos campos, incluyendo la reconstrucción de volúmenes tridimensionales y el reconocimiento de objetos. Tomando como referencia el marco teórico inicial se observa la conveniencia de que los descriptores de característica basadas en puntos claves sean invariantes a las transformaciones geométricas y a los cambios de iluminación. El descriptor SIFT es uno de los algoritmos de extracción y descripción de características más populares, permite extraer puntos característicos de una imagen y sus descriptores son invariantes a la escala, la iluminación y la rotación.

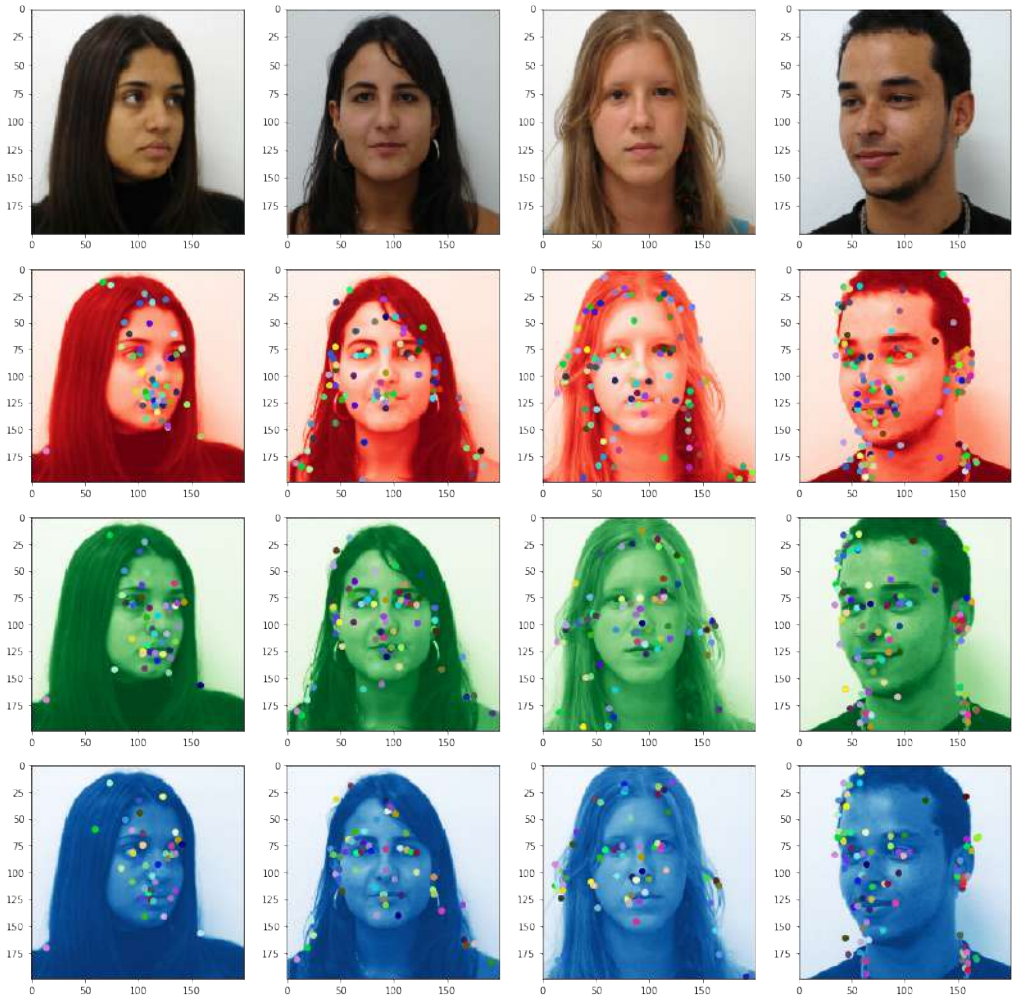


Figura 2.15: Puntos claves SIFT de distintas imágenes por canal.

A diferencia de los descriptores HOG y HLBP, SIFT no proporciona una impresión global de la imagen, en cambio, detecta puntos claves y describe cada punto con un descriptor que contiene 128 números, por lo que para cada imagen se puede obtener una cantidad distinta de puntos claves como se muestra en la figura 2.15. Se requiere construir un descriptor que tenga número finito de características para poder entrenar nuestro algoritmo de aprendizaje supervisado. Se deben seleccionar algunos de estos puntos según un determinado criterio.

El criterio utilizado para obtener nuestro descriptor está basado en un muestreo de los puntos claves por su cercanía a una malla de irregular de puntos de referencia. Los nodos de esta malla se obtienen luego de aplicar el algoritmo de aprendizaje

no supervisado: k-means, sobre una muestra de las coordenadas de puntos claves obtenidas del conjunto de datos. Los nodos de la malla irregular están formados por los centroides de los grupos obtenidos. Una vez que contamos con una muestra de puntos claves provenientes del conjunto de datos, aplicamos en algoritmo de agrupamiento. Nuestra primera decisión es elegir en cuántos grupos debemos separar los datos, para obtener este valor seleccionamos el mejor K para el algoritmo K-means. La figura 2.16 muestra el procedimiento de selección del número de nodos de nuestra malla irregular, en lugar de tomar una decisión arbitraria, podemos usar una *Curva Elbow* para resaltar la relación entre la cantidad de clústeres que elegimos y la suma de errores cuadráticos (SSE) que resulta del uso de esa cantidad de grupos. La gráfica muestra la relación entre estas variables y en ella podemos identificar la cantidad óptima de grupo.

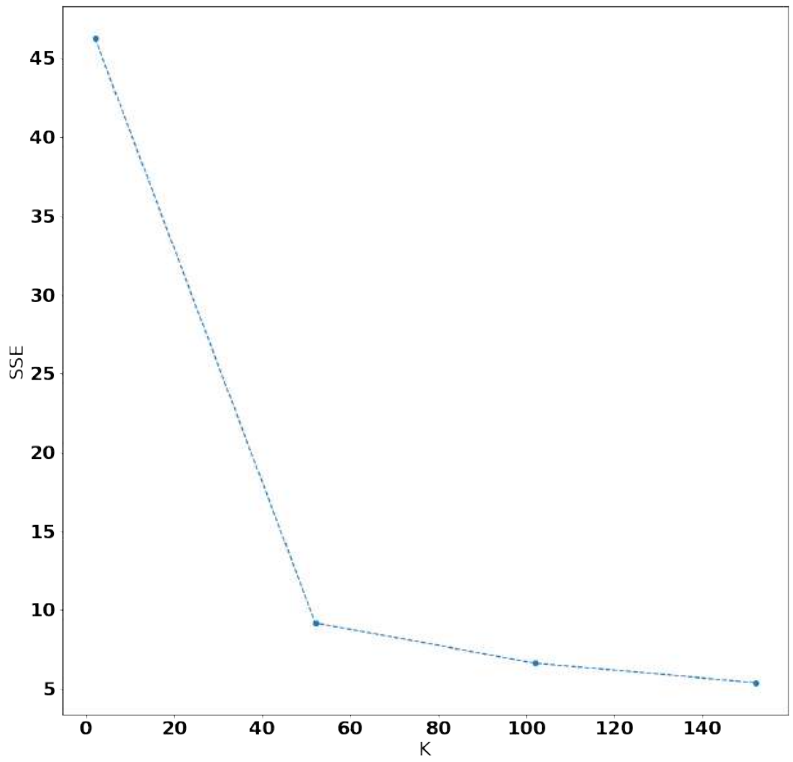


Figura 2.16: Curva K Vs. SSE.

Una vez que el número de grupo se hace igual a k=50 (en el eje X), la diferencia en el SSE se hace más pequeña para cada incremento en el número de grupos. Se puede observar que la cantidad óptima de grupos donde comienza a reducirse el SSE está alrededor k=50, así que elegimos este valor para k. Los centros de estos

grupos serán utilizadas como los nodos de una malla irregular sobre las imágenes para seleccionar los descriptores, la malla se muestra en la figura 2.17. En esta malla irregular los espacios entre los nodos no tienen la misma distancia, y su ubicación está muy cerca de las coordenadas de los puntos claves SIFT del conjunto de imágenes de entrenamiento seleccionadas.

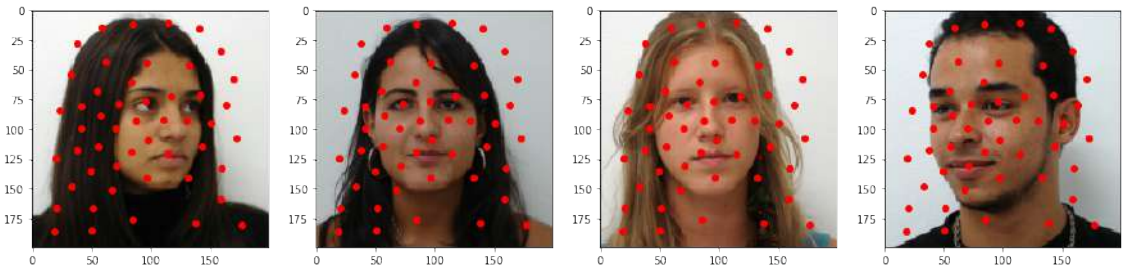


Figura 2.17: Malla irregular SIFT con k=50.

Una vez construida la malla irregular recorremos cada nodo y seleccionamos el punto clave más cercano utilizando el algoritmo del vecino más próximo. El algoritmo es entrenado con las coordenadas de los puntos claves de cada imagen y luego determinamos cuál de estos puntos es el más cercano al nodo correspondiente de la malla irregular. El descriptor asignado a este nodo se corresponderá con el del punto clave más cercano en la imagen.

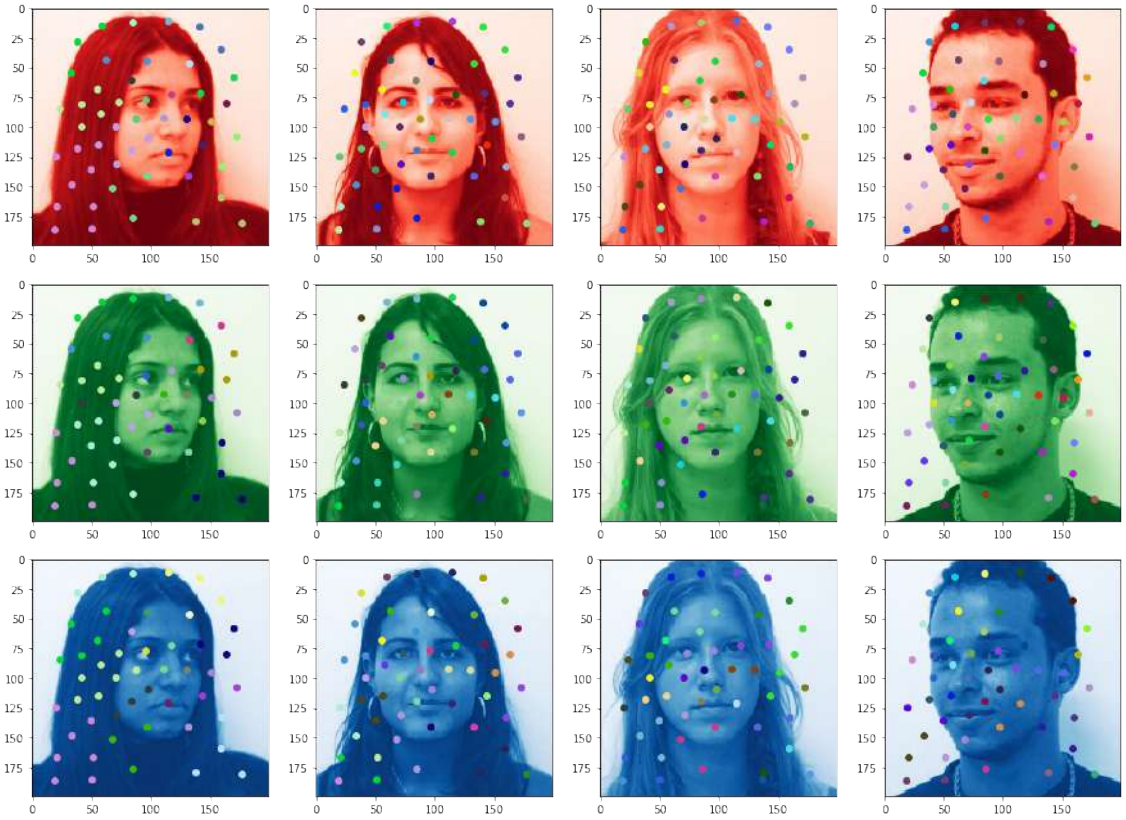


Figura 2.18: Malla irregular con el descriptor SIFT más cercano.

La figura 2.18 muestra los puntos claves asignados a cada imagen según su cercanía a la malla irregular. El color representa el punto clave más cercano en la imagen SIFT original. Para cada una de las imágenes se obtienen 50 descriptores SIFT de 128 valores. La figura 2.19, muestra una imagen de los descriptores SIFT encontrados para cada canal.

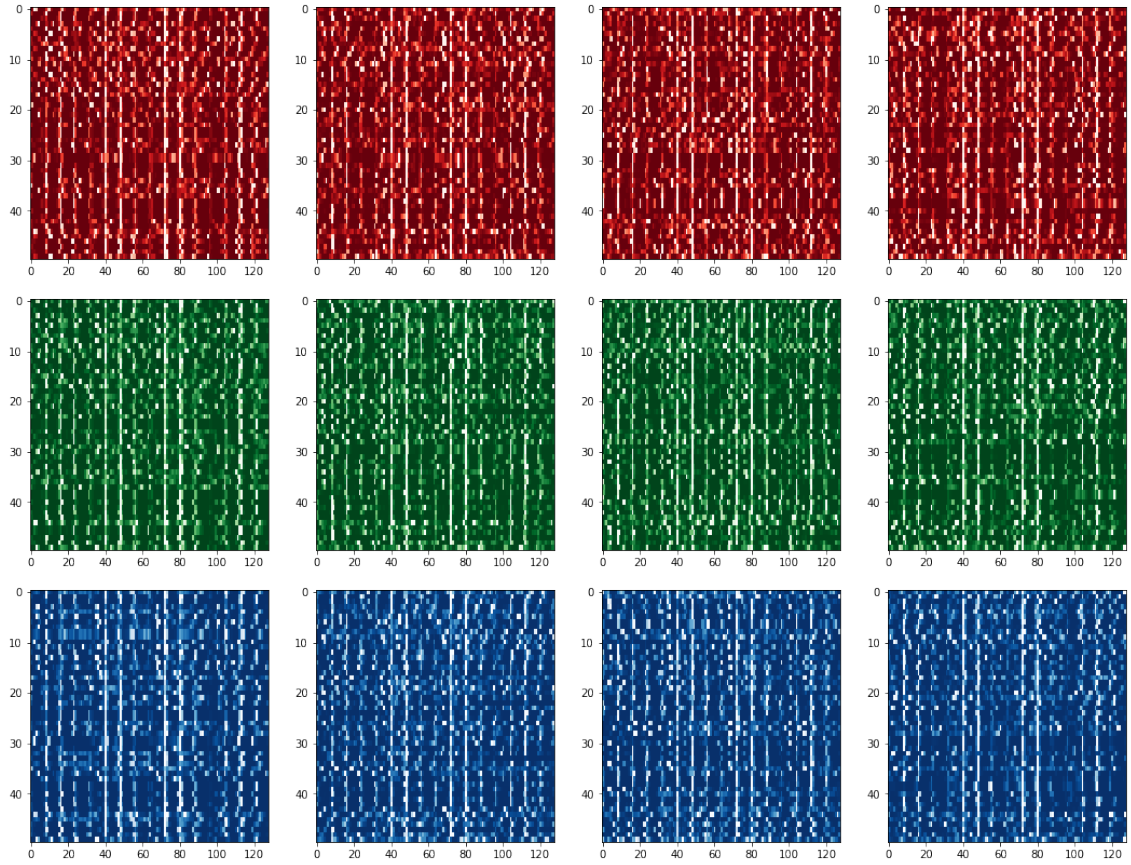


Figura 2.19: Imagen representativa de los descriptores SIFT por canal.

Para cada una de las imágenes del conjunto de datos seleccionados de la base de datos FEI, obtenemos mediante esta técnica un descriptor de 50x128=6.400 valores, que además contiene información del comportamiento del color en la imagen ya que fue generado en todos sus canales. La tabla muestra el detalle de la vista minable obtenida para el entrenamiento de este modelo. El cuadro 2.5 muestra que para este descriptor el número de características se ha reducido en un 88 % por lo que es significativamente menor.

Bloque	Ancho	Alto	Canales	Bins	Rango	Tam. Desc.	PCA Desc.
NA	200	200	3	NA	0 - 1	19.200	1.685

Cuadro 2.5: Resumen estadístico de la vista minable SIFT.

Si observamos puntos claves obtenidos por cada canal podemos apreciar que muchos de ellos están ubicados en posiciones similares, por lo que inferimos que hay gran cantidad de información correlacionada, utilizaremos PCA para obtener

características menos relacionadas. Para ello indicaremos al algoritmo que retorne un número de características tales que la variabilidad de la data sea igual al 95 %. El cuadro 2.5 muestra la cantidad de componentes que arroja PCA, así como un resumen estadístico de nuestra vista minable.

2.3. Minería de datos

La tarea de minería de datos requerida en este caso es una tarea de clasificación para lo cual elegimos una representación basada en redes neuronales. En la construcción de los modelos utilizaremos la librería Keras de Python. La estructura de datos principal en Keras es un modelo (model), que está pensada para facilitar la organización de capas, pues en realidad los modelos en Keras son básicamente una secuencia de capas. La clase *"keras.models.Sequential"* es una envoltura para el modelo de red neuronal. Los modelos en Keras se definen como una secuencia de capas. Hay capas *"Fully Connected"* (Completamente Conectadas), capas de *"MaxPool"*, capas de activación, etc. Se puede agregar una capa a un modelo Keras mediante la función *add()*. La librería deducirá automáticamente la forma de todas las capas después de la primera capa, esto significa que solo se tienen que establecer las dimensiones de entrada para la primera capa.

2.3.1. Clasificador HLBP

El clasificador LBP_FACE_NET utiliza una red neuronal como lenguaje de representación y será entrenado con los descriptores HLBP que obtuvimos en la sección anterior. El código que utilizamos para construir el modelo se muestra en el listado 2.1. La red neuronal recibe como entrada un vector de 479 valores punto flotante conformado por las componentes principales de los descriptores HLBP. La red está conformada de forma general por una capa de entrada, una oculta y una de salida. La capa de entrada recibe los valores reducidos luego de aplicar PCA a los histogramas LBP. La capa oculta está conformada por 2.048 neuronas y una capa de salida de 200 neuronas que se corresponden al número de clases. El proceso de aprendizaje se configura con la función *compile()* y utiliza las librerías numéricas de que dispone el backend de TensorFlow.

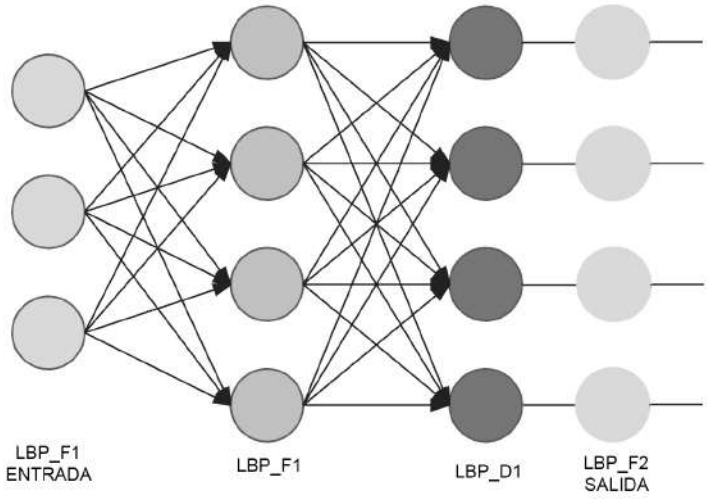


Figura 2.20: Arquitectura del clasificador HLBP.

El backend selecciona automáticamente la mejor forma de representar la red para el entrenamiento, y hace que los cálculos se ejecuten de forma eficiente en el hardware. Al compilar, debemos especificar algunas propiedades adicionales requeridas, como la función de costo (loss), el optimizador que utilizaremos para actualizar los pesos de la red y las métricas que deseamos registrar durante el proceso. En nuestro caso utilizaremos inicialmente la exactitud predictiva (Accuracy). La función de pérdida que utilizaremos es sparse_categorical_crossentropy y nuestro optimizador será el optimizador Adam.

```
1
2  def create_lbp_face_net():
3
4    model= Sequential()
5    model.add(Dense(2048, name='LBP_F1',
6           activation='relu',
7           input_shape=(479,),
8           kernel_initializer='normal')
9           )
10
11   model.add(Dropout(0.2, name='LBP_D1'))
12
13   model.add(Dense(200, name='LBP_F3',
14          activation='softmax')
15          )
16
17   model.compile(loss='sparse_categorical_crossentropy',
18          optimizer='Adam',
19          metrics=['accuracy'])
20
21   return model
22 .
```

Listado de Código 2.1: Función python modelo HLBP-FACE-NET

Se ha utilizado una capa *Dropout* en base a los resultados experimentales obtenidos. Las capas Dropout omiten aleatoriamente la salida de una neurona en base a un valor de probabilidad, lo que permite aumentar la capacidad de generalización de la red neuronal. Mediante un procedimiento de búsqueda se han ajustado los valores para obtener la mejor exactitud. El cuadro 2.6 muestra el resumen de las salidas de cada capa y los parámetros [1] que serán calculados en el proceso de entrenamiento de nuestra red. El total de parámetros a calcular durante el entrenamiento es de 1,392,840. Los parámetros a calcular no son más que los pesos sinápticos que serán actualizados por el algoritmo para la resolución del problema.

Nombre	Tipo	Forma de salida	Parámetros
LBP_F1_input	INPUT	(, 479)	0
LBP_F1	DENSE	(, 2.048)	983.040
LBP_D1	DROPOUT	(, 2.048)	0
LBP_F3	DENSE	(, 200)	409.800

Cuadro 2.6: Resumen de la arquitectura del modelo LBP_FACE_NET.

Para el entrenamiento de esta red neuronal utilizamos el conjunto de descriptores HLBP, que contiene una cantidad equivalente al 80 % de los descriptores obtenidos para las imágenes que conforman el conjunto de datos FEI. Utilizaremos además el 20 % restante como conjunto de validación durante el entrenamiento, lo que permitirá verificar en cada época [2] el comportamiento del modelo cuando es expuesto a datos que no están en el conjunto de entrenamiento. Esto permite controlar el *Overfitting* y el aprendizaje mediante el ajuste de los parámetros de configuración la red. El listado de código 2.2 muestra la sintaxis utilizada para el entrenamiento del modelo.

```
1
2 LBP_FACE_NET=create_lbp_face_net()
3
4 history =  LBP_FACE_NET.fit(X_train_lbp,
5                  y_train,
6                  validation_data=(X_test_lbp, y_test),
7                  epochs=55,
8                  batch_size=64)
9 .
```

Listado de Código 2.2: Entrenamiento del modelo LBP_FACE_NET

[1] Pesos sinápticos

[2] Periodo en que se procesa el conjunto de entrenamiento

El entrenamiento se inicia invocando la función *fit()* de nuestro modelo una vez construido y el proceso se ejecutará durante un número fijo de iteraciones establecido mediante el parámetro *epochs*. Podemos establecer el número de instancias de nuestros conjunto de datos que serán cargadas simultáneamente en memoria para ser procesadas por nuestro algoritmo de aprendizaje mediante el parámetro *batch_size*. *Verbose* controla la cantidad de información que se despliega en la salida estándar durante la ejecución. El número de épocas se ha ajustado a 55 mientras que el tamaño del lote de entrenamiento se fija en 64.

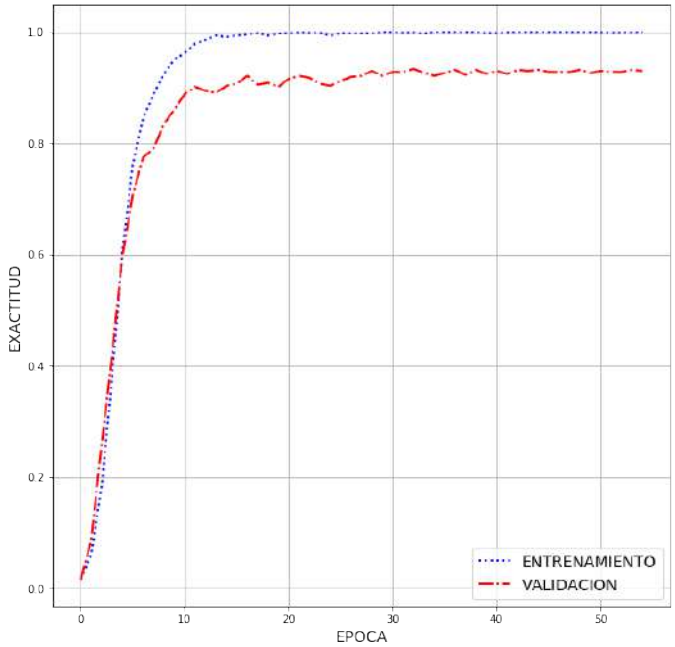


Figura 2.21: Exactitud del modelo LBP_FACE_NET.

El procedimiento de ajuste de los parámetros de entrenamiento de la red se lleva a cabo mediante un procedimiento de búsqueda exhaustiva, las librerías de Keras ofrecen varias alternativas para la búsqueda. El procedimiento de ajuste permite, a partir de una serie de valores tabulados, obtener la mejor combinación para alcanzar la máxima exactitud. La exactitud es una métrica para evaluar modelos de clasificación que expresa numéricamente la relación del número de predicciones correctas entre el número total de predicciones. El proceso de entrenamiento retorna un histórico de donde se obtiene la gráfica de la figura 2.21. El modelo LBP_FACE_NET luego de unas 10 épocas, comienza a tener una exactitud cercana al 0.90 para el conjunto de validación, mientras que la pérdida se ubica alrededor de 0.50. La exac-

titud luego de 55 épocas para el modelo está cerca del 0.93 mientras que la pérdida es cercana a 0.30. La pérdida es una penalidad por una predicción incorrecta. Esto quiere decir que la pérdida es un número que indica qué tan incorrecta fue la predicción del modelo en un solo ejemplo. Si la predicción del modelo es perfecta, la pérdida es cero; de lo contrario, la pérdida es mayor.

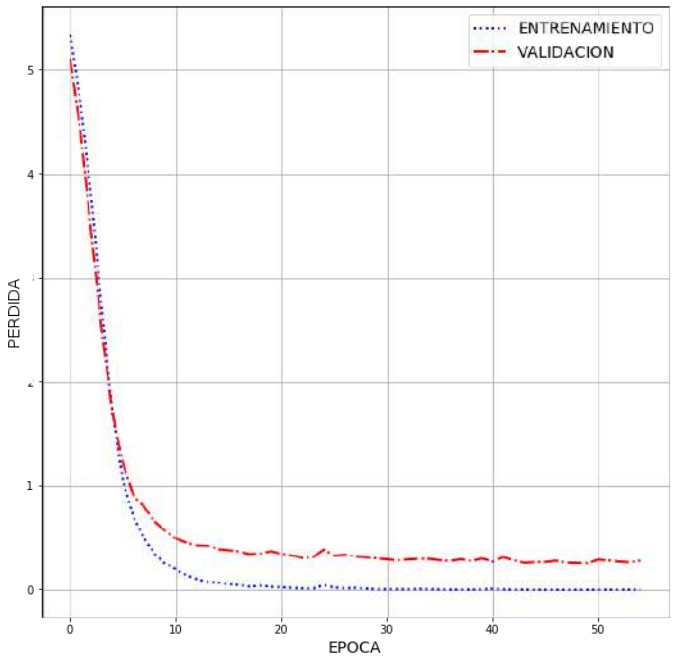


Figura 2.22: Pérdida en el modelo LBP_FACE_NET.

Finalizado el proceso de entrenamiento se realiza el cálculo de algunas de las métricas estándar : Exactitud, Precisión, Recall y F1, el cuadro 2.7 muestra los resultados obtenidos sobre los datos de validación. Es importante señalar que al tratarse de un problema de clasificación multiclases los cálculos que aquí se muestran son el promedio del cálculo de las métricas para cada clase de forma individual, esto es conocido como el *macropromedio*. La *Precisión* es una medida que indica, de los ejemplos clasificados como positivos, cuántos son clasificados correctamente. El *Recall* es una medida de la completitud, que indica cuántos ejemplos de la clase positiva fueron clasificados correctamente. El *Score F1* es el promedio armónico entre la Precisión y el Recall.

Exactitud	Precisión	Recall	f1
0.93	0.93	0.94	0.93

Cuadro 2.7: Métricas obtenidas para el modelo LBP_FACE_NET.

Una vez entrenado el modelo almacenamos su estructura y los pesos generamos en el entrenamiento para su posterior uso como se muestra en el listado de código 2.3.

```
1
2  LBP_FACE_NET_JSON= LBP_FACE_NET.to_json()
3  open('LBP_FACE_NET_JSON_ARCHITECTURE.json','w').write(LBP_FACE_NET_JSON)
4  LBP_FACE_NET.save_weights('LBP_FACE_NET_WEIGHTS.h5',overwrite=True)
5 .
```

Listado de Código 2.3: Almacenando el modelo y los pesos de LBP-FACE-NET

2.3.2. Clasificador HOG

El clasificador HOG_FACE_NET utiliza al igual que el anterior una red neuronal como lenguaje de representación y es entrenado con los descriptores HOG obtenidos en la etapa de extracción de características. La figura 2.23 muestra la arquitectura del clasificador.

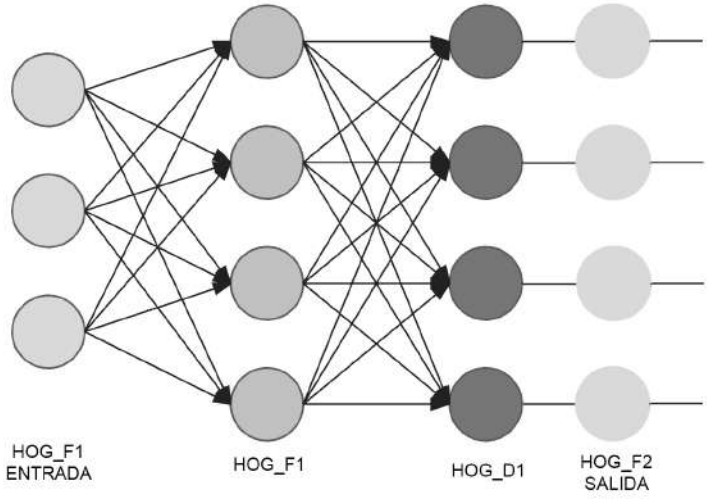


Figura 2.23: Arquitectura del clasificador HOG.

La red neuronal recibe como entrada un vector de 1243 números punto flotante, obtenidos luego de aplicar PCA sobre el descriptor HOG obtenido en la etapa de extracción de características, y cuenta con tres capas: una capa completamente conectada de entrada con 1243 neuronas, una capa oculta de 512 neuronas y una capa de salida de 200 valores. Además se ha utilizado una capa *Dropout* para regularización. Los parámetros de entrenamiento de la red, como la cantidad de neuronas en cada capa, son ajustados mediante un procedimiento de búsqueda exhaustiva para obtener la máxima exactitud predictiva. El listado 2.4 muestra el código Python requerido para la construcción de esta red neuronal.

```
def create_hog_face_net():

    model= Sequential()
    model.add(Dense(512,
    name='HOG_F1',
    activation='relu',
    input_shape=(1243,),
    kernel_initializer='normal'))
    model.add(Dropout(0.4,name='HOG_D1'))
    model.add(Dense(200, name='HOG_F2', activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy',
    optimizer='Adam',
    metrics=['accuracy'])
```

```
14
15      return model
```

Listado de Código 2.4: Función Python modelo HOG-FACE-NET

El entrenamiento de esta red utilizamos los descriptores HOG reducidos mediante PCA obtenidos en el proceso de extracción de datos, el conjunto de entrenamiento tiene 2240 muestras. Además se utiliza un conjunto de validación de 560 para validar la respuesta del modelo sobre datos distintos al conjunto de entrenamiento en cada época.

Nombre	Tipo	Forma de salida	Parámetros
HOG_F1_input	INPUT	(, 1.243)	0
HOG_F1	DENSE	(, 512)	636.928
HOG_D1	DROPOUT	(, 512)	0
HOG_F2	DENSE	(, 200)	102.600

Cuadro 2.8: Resumen de la arquitectura del modelo HOG_FACE_NET.

El número de épocas es ajustado a 55 mientras que el tamaño del lote de entrenamiento se fija en 64. El procedimiento de ajuste de los parámetros de entrenamiento se lleva a cabo mediante una búsqueda exhaustiva utilizando la funcionalidad de la librería Keras: GridSearchCV, esta evalúa las posibles combinaciones a partir de una matriz suministrada por el usuario y despliega la opción que alcanza un mejor rendimiento. Este procedimiento permite buscar en una retícula de opciones las posibles combinaciones que den el mayor rendimiento.

```
1
2 HOG_FACE_NET=create_hog_face_net()
3
4 history = HOG_FACE_NET.fit(X_train_hog,
5                 y_train,
6                 batch_size=64,
7                 epochs=55,
8                 verbose=1,
9                 validation_data=(X_test_hog, y_test))
```

Listado de Código 2.5: Entrenamiento del modelo HOG-FACE-NET

Las gráficas de las figuras 2.24 y 2.25 muestran el desempeño de nuestro modelo durante el entrenamiento. El modelo HOG_FACE_NET luego de unas 10 épocas, comienza a tener una exactitud cercana al 0.85 para el conjunto de validación,

mientras que la pérdida se ubica alrededor de 0.7. La exactitud luego de 55 épocas para el modelo está cerca del 0.90 mientras que la pérdida es cercana a 0.5.

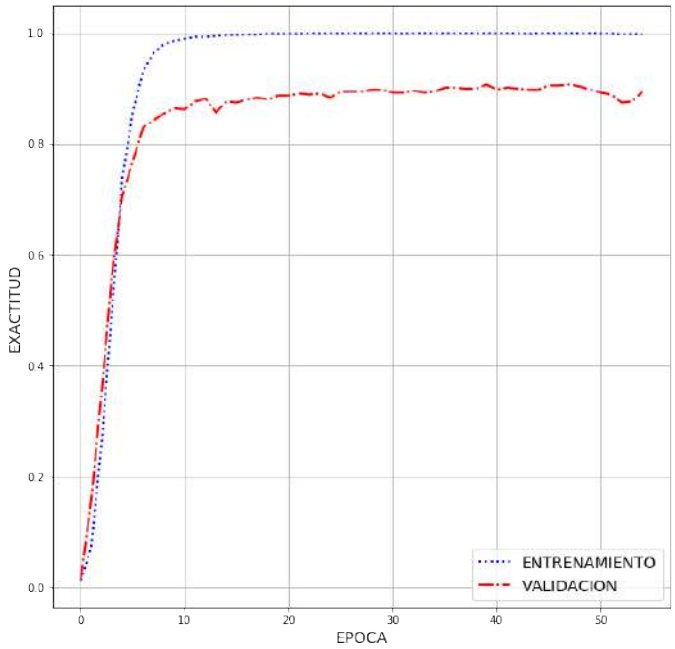


Figura 2.24: Exactitud del modelo HOG_FACE_NET.

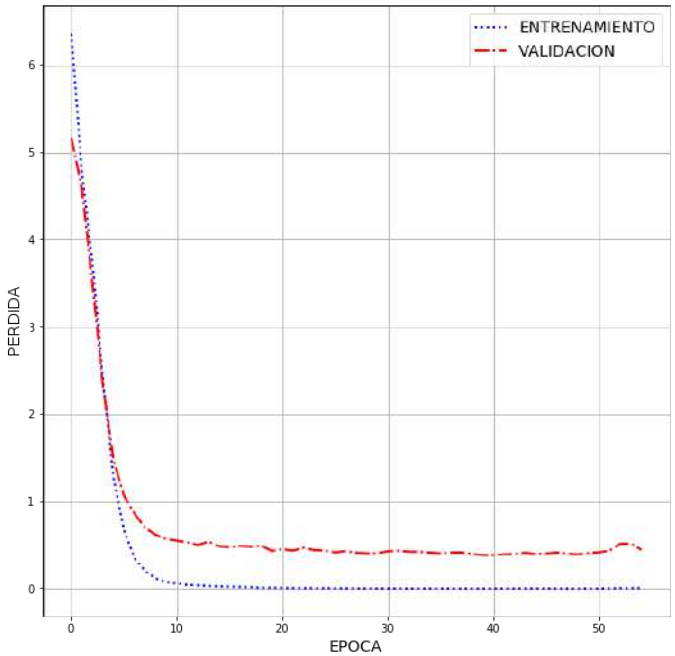


Figura 2.25: Perdida en el modelo HOG_FACE_NET.

El cuadro 2.9 muestra los resultados obtenidos sobre los datos de validación, los valores mostrados corresponden al macropromedio calculado para las 200 clases que

conforman el conjunto de datos.

Exactitud	Precisión	Recall	f1
0.89	0.89	0.90	0.88

Cuadro 2.9: Métricas obtenidas para el modelo HOG_FACE_NET.

El listado de código 2.6 muestra como se almacena el modelo almacenamos su estructura y los pesos para su posterior uso.

```
1
2 HOG_FACE_NET_JSON= HOG_FACE_NET.to_json()
3 open('HOG_FACE_NET_JSON_ARCHITECTURE.json','w').write(HOG_FACE_NET_JSON)
4 HOG_FACE_NET.save_weights('HOG_FACE_NET_WEIGHTS.h5',overwrite=True)
5 .
```

Listado de Código 2.6: Almacenando el modelo y los pesos de HOG-FACE-NET

2.3.3. Clasificador SIFT

La red neuronal utilizada para la clasificación a partir de descriptores basados en SIFT es similar a las anteriores. El modelo recibe como entrada un vector de 1685 valores punto flotante, que se corresponden con el número de componentes principales obtenido mediante PCA sobre el descriptor basado en SIFT propuesto en la etapa de extracción de características, y cuenta con una capa oculta de 512 neuronas, una capa Dropout, y una capa de salida de 200 neuronas. El número de neuronas se obtiene a partir de un procedimiento de búsqueda exhaustiva.

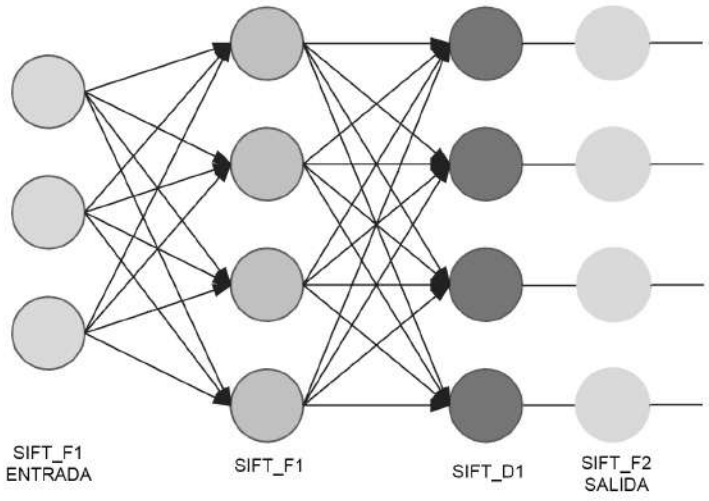


Figura 2.26: Arquitectura del clasificador SIFT_FACE_NET.

La figura 2.26 muestra la arquitectura del clasificador. Los parámetros de este modelo se ajustan siguiendo los mismos procedimientos anteriores. El listado de código 2.7 muestra el programa en Python requerido para la construcción de esta red neuronal.

```
1
2  def create_sift_face_net():
3
4      model= Sequential()
5      model.add(Dense(neurons1, name='SIFT_F1', activation='relu', input_shape=(1685,),
       kernel_initializer=init))
6      model.add(Dropout(0.5, name='SIFT_D1'))
7      model.add(Dense(200, name='SIFT_F2', activation='softmax'))
8      model.compile(loss='sparse_categorical_crossentropy',
9      optimizer='Adam',
10     metrics=['accuracy'])
11
12     return model
```

Listado de Código 2.7: Función python modelo SIFT-FACE-NET

Nombre	Tipo	Forma de salida	Parámetros
SIFT_F1_input	INPUT	(, 1.685)	0
SIFT_F1	DENSE	(, 512)	863.232
SIFT_D1	DROPOUT	(, 512)	0
SIFT_F2	DENSE	(, 200)	102.600

Cuadro 2.10: Resumen de la arquitectura del modelo SIFT_FACE_NET.

En el entrenamiento de este modelo utilizamos los descriptores basados en SIFT obtenidos en la etapa de extracción sobre los que se ha aplicado la técnica de reducción de dimensionalidad PCA.

```
1 sift_face_net=create_sift_face_net()
2 history = sift_face_net.fit(X_train, y_train, batch_size=256, epochs=64,verbose=1,
3 validation_data=(X_test, y_test))
```

Listado de Código 2.8: Entrenamiento del modelo SIFT-FACE-NET

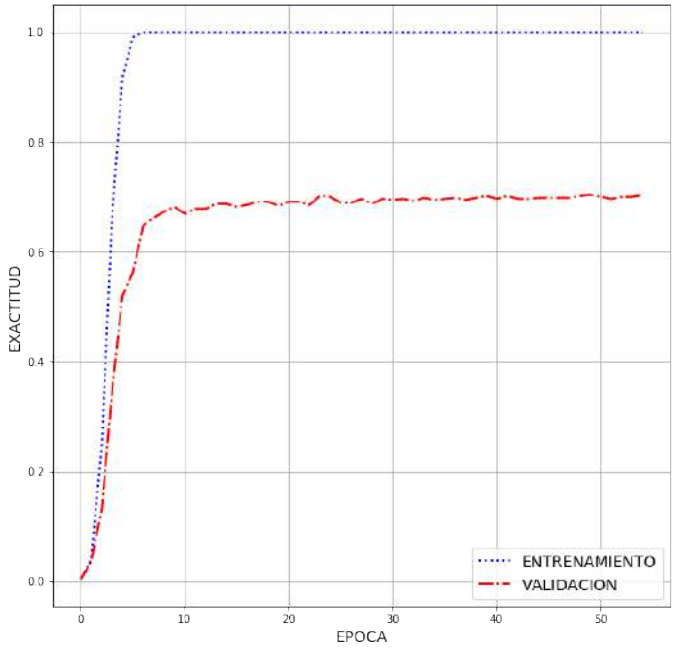


Figura 2.27: Exactitud del modelo SIFT_FACE_NET.

El número de épocas es ajustado a 55 mientras que el tamaño del lote de entrenamiento se fija en 64 de forma similar a los modelos anteriores. Las gráficas con los históricos obtenidos se muestran en las figuras 2.27 y 2.28.

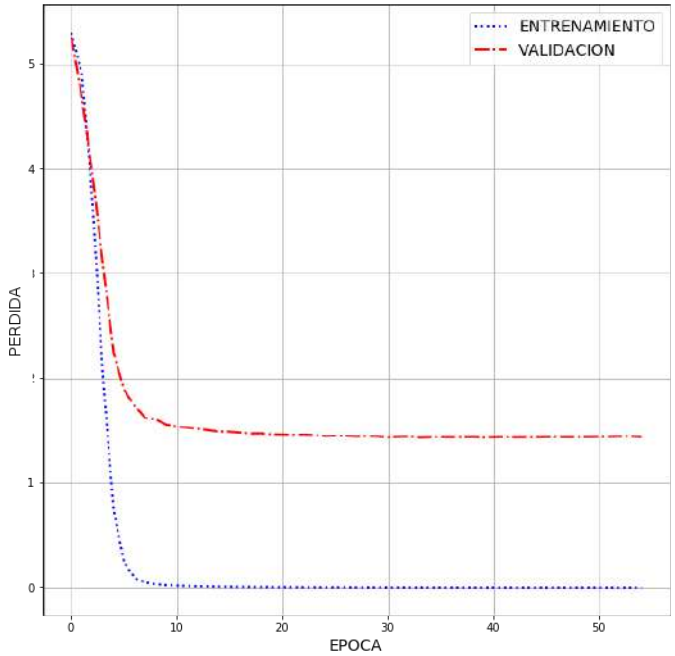


Figura 2.28: Pérdida del modelo SIFT_FACE_NET.

Las métricas de Exactitud, Precisión, Recall y f1 correspondientes al modelo en-

trenado se muestran en el cuadro 2.11. Como vemos este modelo alcanza una menor precisión que los anteriores, recordemos que a diferencia de los otros descriptores SIFT no suministra información global de la imagen, y el descriptor utilizado se crea a partir de un muestreo de puntos claves sobre la imagen de entrada.

Exactitud	Precisión	Recall	f1
0.75	0.77	0.77	0.73

Cuadro 2.11: Métricas obtenidas para el modelo RAW_FACE_NET.

El listado de código 2.9 muestra como almacenamos el modelo RAW_FACE_NET para un uso posterior.

```
1
2 RAW_FACE_NET_JSON= RAW_FACE_NET.to_json()
3 open('RAW_FACE_NET_JSON_ARCHITECTURE.json','w').write(RAW_FACE_NET_JSON)
4 RAW_FACE_NET.save_weights('RAW_FACE_NET_WEIGHTS.h5',overwrite=True)
5 .
```

Listado de Código 2.9: Almacenando el modelo y los pesos de RAW_FACE_NET

2.3.4. Clasificador RAW

Una red neuronal convolucional está formada por colecciones de neuronas organizadas de forma estructurada mediante capas. La entrada es una imagen y su salida la puntuación para cada una de las clases. A diferencia de los modelos anteriores la extracción de características en una CNN se efectúa en las primeras capas de nuestro modelo. La figura 2.29 describe el modelo utilizado para el reconocimiento de rostros. El cantidad de clases que se utilizó en el modelo correspondiente al número de personas distintas en la base de datos FEI. La red se entrenó con las imágenes de los del conjunto de datos y a cada imagen se le etiquetó con el valor numérico correspondiente a la etiqueta de la clase.

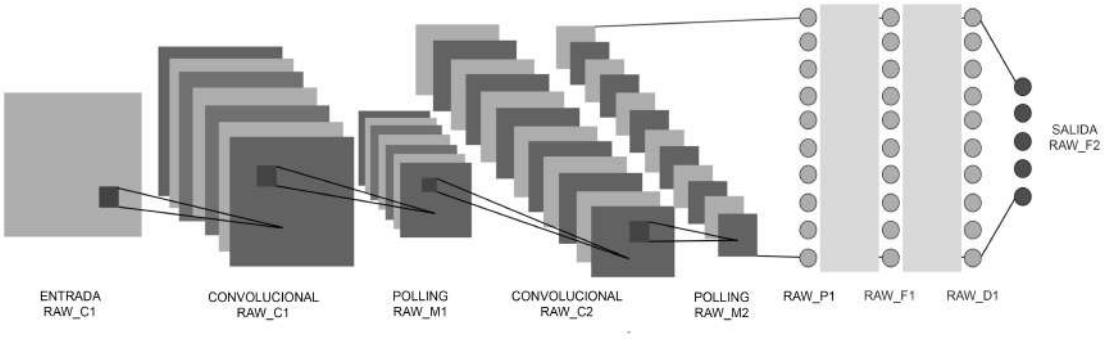


Figura 2.29: Arquitectura del modelo RAW_FACE_NET.

En la construcción de la red utilizaremos el modelo Secuencial de la API Keras. Un modelo secuencial permite apilar distintos tipos de capas, las capas superiores [3] realizan la tarea de extracción de características aplicando núcleos de convolución de distinto tamaño sobre las imágenes de entrada mientras que la capa inferior [4], completamente conectada, es la encargada de clasificar las características extraídas. El listado de Código 2.10 muestra los detalles de construcción del modelo con Keras.

```
def create_cnn_face_net():

    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', name='RAW_C1',activation='relu', input_shape
    =(50, 50,3)))
    model.add(MaxPooling2D(pool_size=(2, 2), name='RAW_M1'))
    model.add(Conv2D(64, (3, 3), padding='same', name='RAW_C2', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), name='RAW_M2'))
    model.add(Flatten(name='RAW_P1'))
    model.add(Dense(512, activation='relu', name='RAW_F1'))
    model.add(Dropout(0.4, name='RAW_D1'))
    model.add(Dense(200, activation='softmax', name='RAW_F2'))

    model.compile(loss='sparse_categorical_crossentropy',
    optimizer='Adam',
    metrics=['accuracy'])

    return model
```

Listado de Código 2.10: Función que crea el modelo CNN-FACE-NET

[3] Capas cercanas a la de entrada

[4] Capa cercana a la salida

La red está formada por dos capas convolucionales (*Conv2D()*) cada una seguida de una capa de submuestreo (*MaxPooling2D()*). Estas reciben como entrada el mapa de características generado por las capas convolucionales, determinan el máximo valor en un vecindario de 2x2 píxeles y generan nuevos mapas de características que sirven de entrada al siguiente bloque. Los pesos de las sinapsis son inicializados siguiendo una distribución uniforme. Las capas aplican una función de activación *Relu()* a su salida lo que les permite detectar límites de decisión no lineales. Las imágenes de entrada son preprocesadas ajustando sus niveles de contraste y su tamaño a 50x50. Luego de pruebas con dimensiones mayores se determinó que con este tamaño se alcanza una exactitud similar con un menor consumo de recursos.

Nombre	Tipo	Forma de salida	Parámetros
RAW_C1_input	INPUT	(None, 50, 50, 3)	0
RAW_C1	CONVOLUTION	(None, 50, 50, 32)	896
RAW_M1	POOLING	(None, 25, 25, 32)	0
RAW_C2	CONVOLUTION	(None, 25, 25, 64)	18496
RAW_M2	POOLING	(None, 12, 12, 64)	0
RAW_P1	FLATTEN	(None, 9216)	0
RAW_F1	DENSE	(None, 512)	4.719.104
RAW_D1	DROPOUT	(None, 512)	0
RAW_F2	DENSE	(None, 200)	102.600

Cuadro 2.12: Resumen del modelo RAW_FACE_NET.

En este modelo al igual que los anteriores incluimos capas de *Dropout()* para evitar el sobreajuste y las capas completamente conectadas con una función de activación *Softmax()*. Para entrenar el modelo raw_face_net, es compilada estableciendo una función de costo *"sparse_ categorical_ crossentropy"* para la clasificación multiclase, así como también especificando las métricas que queremos calcular en cada época del proceso de capacitación, en este caso la exactitud. El cuadro 2.12 muestra un resumen de la arquitectura, las forma de salida y los parámetros [5] a calcular por cada capa. El total de parámetros a calcular para este modelo es de 4.841.096.

[5]Pesos Sinápticos

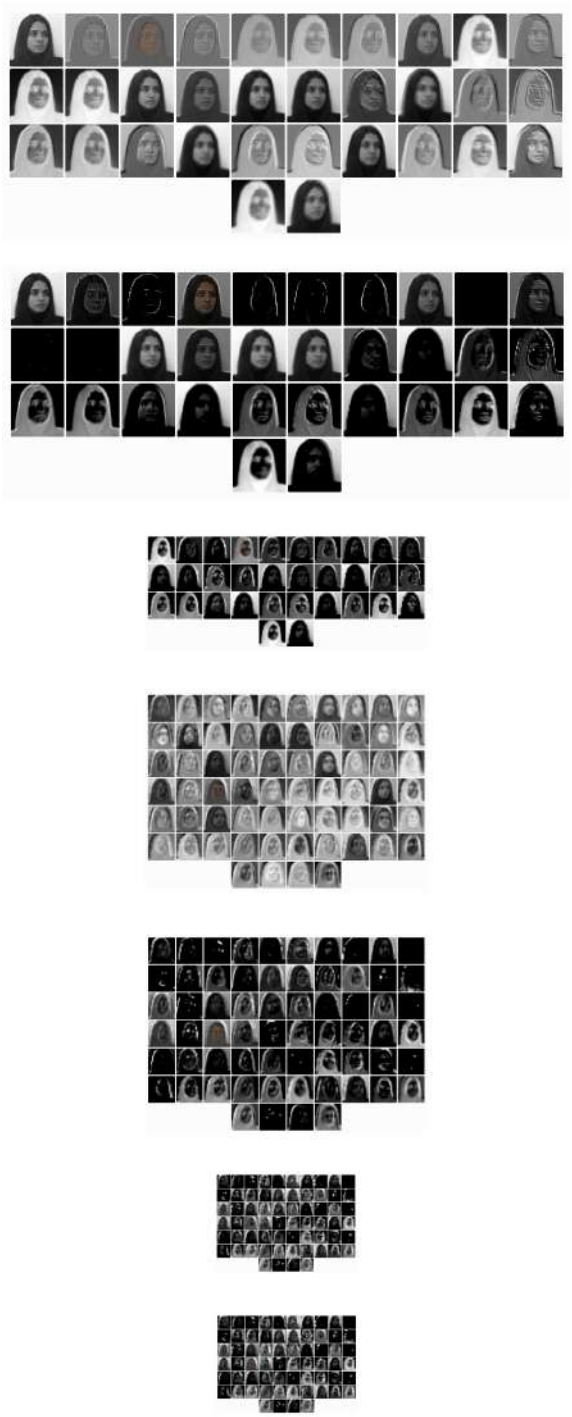


Figura 2.30: Mapas de características modelo RAW_FACE_NET.

En el proceso de entrenamiento se producen mapas de características luego de cada capa. La figura 2.30 muestra un ejemplo de los mapas obtenidos durante el

entrenamiento para una imagen de entrada del conjunto de datos FEI. La primera de las capas convolucionales genera 32 mapas de características aplicando un kernel de 3x3 sobre la imagen de entrada. La segunda capa genera 64 mapas de características a partir de la entrada con kernels del mismo tamaño que la anterior. Las capas de Pooling submuestrean estos mapas calculando el máximo valor en vecindarios de 2x2 por lo que la salida de cada capa de Pooling reduce a la mitad las dimensiones del mapa de entrada.

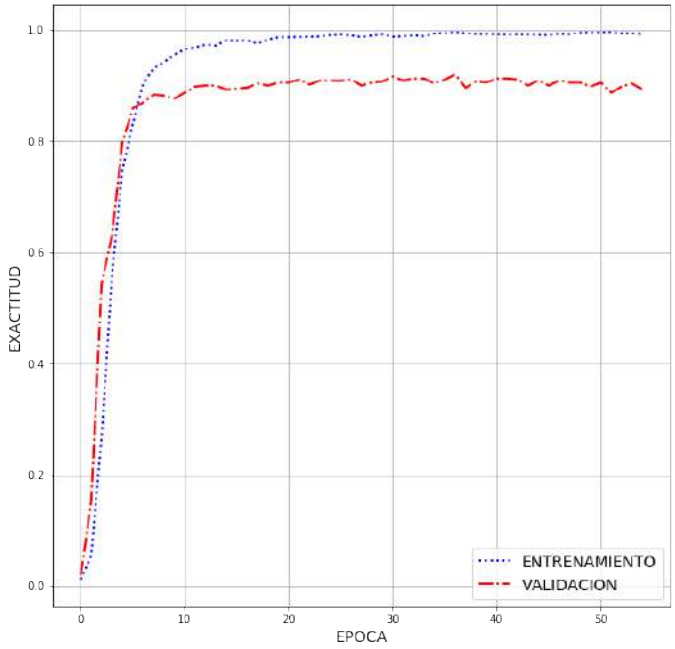


Figura 2.31: Exactitud del modelo RAW_FACE_NET.

Luego de entrenar la red neuronal convolucional con las imágenes del conjunto de datos FEI preprocesadas obtenemos las gráficas de exactitud y pérdida que se muestran en las figuras 2.31 y 2.32. La exactitud alcanza un valor cercano a 0.90 mientras que la pérdida es próxima a 0.60.

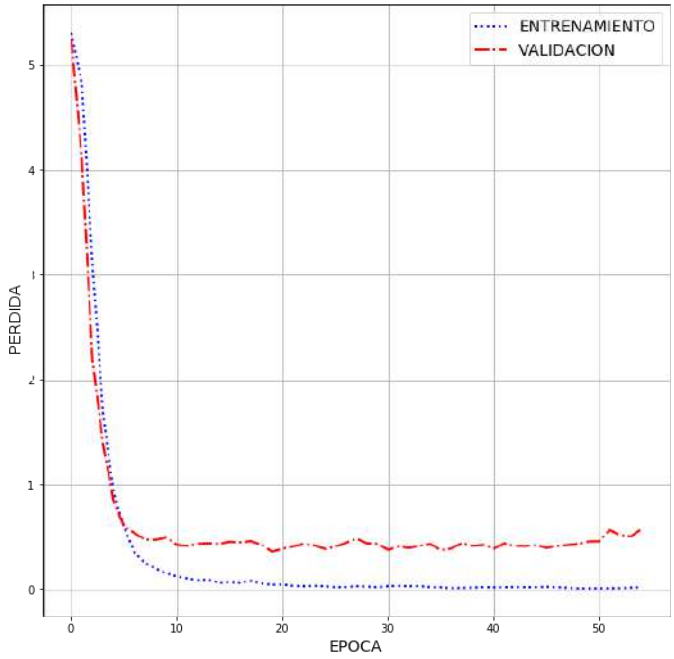


Figura 2.32: Pérdida en el modelo RAW_FACE_NET.

El modelo se entrena utilizando la función *fit()* especificando el número de épocas (*epochs*) en 55 y el tamaño del lote (*batch_size*) en 64. Esto significa que todo el conjunto de entrenamiento pasará por la red 55 veces en lotes de 64 imágenes, luego de procesar cada lote se actualizan los pesos sinápticos. EL cuadro 2.13 muestra las métricas resultantes del entrenamiento del modelo.

Exactitud	Precisión	Recall	f1
0.90	0.90	0.91	0.88

Cuadro 2.13: Métricas obtenidas para el modelo RAW_FACE_NET.

2.3.5. Clasificador Híbrido

El poder las redes convolucionales no solo proviene de su profundidad sino también de su flexibilidad para construir estructuras de red más complejas. La figura 2.33 muestra un modelo híbrido construido a partir de los los clasificadores individuales. Este modelo posee varias ramas, una por cada uno de clasificadores descritos anteriormente. Los clasificadores individuales conjuntamente con los pesos son almacenados y cargados para la construcción del nuevo modelo. Una vez cargado se utiliza otro concepto importante dentro del universo de las redes neuronales, el concepto de *"Transfer Learning"* [19] o transferencia del aprendizaje, una técnica que consiste en reutilizar modelos ya entrenados en una tarea posterior. La reutilización de la información aprendida por los modelos anteriores se logra una vez cargados. Como sabemos durante el proceso de entrenamiento se ajustan los pesos sinápticos mediante un proceso conocido como *"Backpropagation"*, estos pesos son almacenados conjuntamente con el modelo. Una vez cargados los modelos individuales y sus pesos se construye un nuevo modelo a partir de estos, eliminando la última capa densa de cada uno. Una vez eliminada debemos indicarle a la red que los pesos de las capas restantes no deben ser calculados en un nuevo proceso de entrenamiento, es decir permanecerán congelados, de esta forma construimos ramas que ya tienen asociados sus pesos. Estas ramas se combinan a través de una capa "Merge()"que concatena sus salidas. En el aprendizaje de transferencia, primero entrenamos modelo base, y luego reutilizamos las características aprendidas, o las transferimos, a un nuevo modelo.

Nuestro modelo consta así de 4 ramas : lbpBranch, hogBranch, siftBranch y rawBranch, obtenidas de los modelos LBP, HOG, SIFT y RAW correspondientes. Cada descriptor proporciona una representación distinta de las imágenes. Estos patrones son aprendidos por cada modelo, puesto que cada descriptor es generado mediante algoritmos diferentes las redes neuronales en su penúltima capa permiten resumir los rasgos importantes detectados mediante el uso del descriptor. La listado de código 2.11 muestra como obtenemos la *"lbpBranch"* de nuestro modelo, de forma similar se obtiene el resto de las ramas.

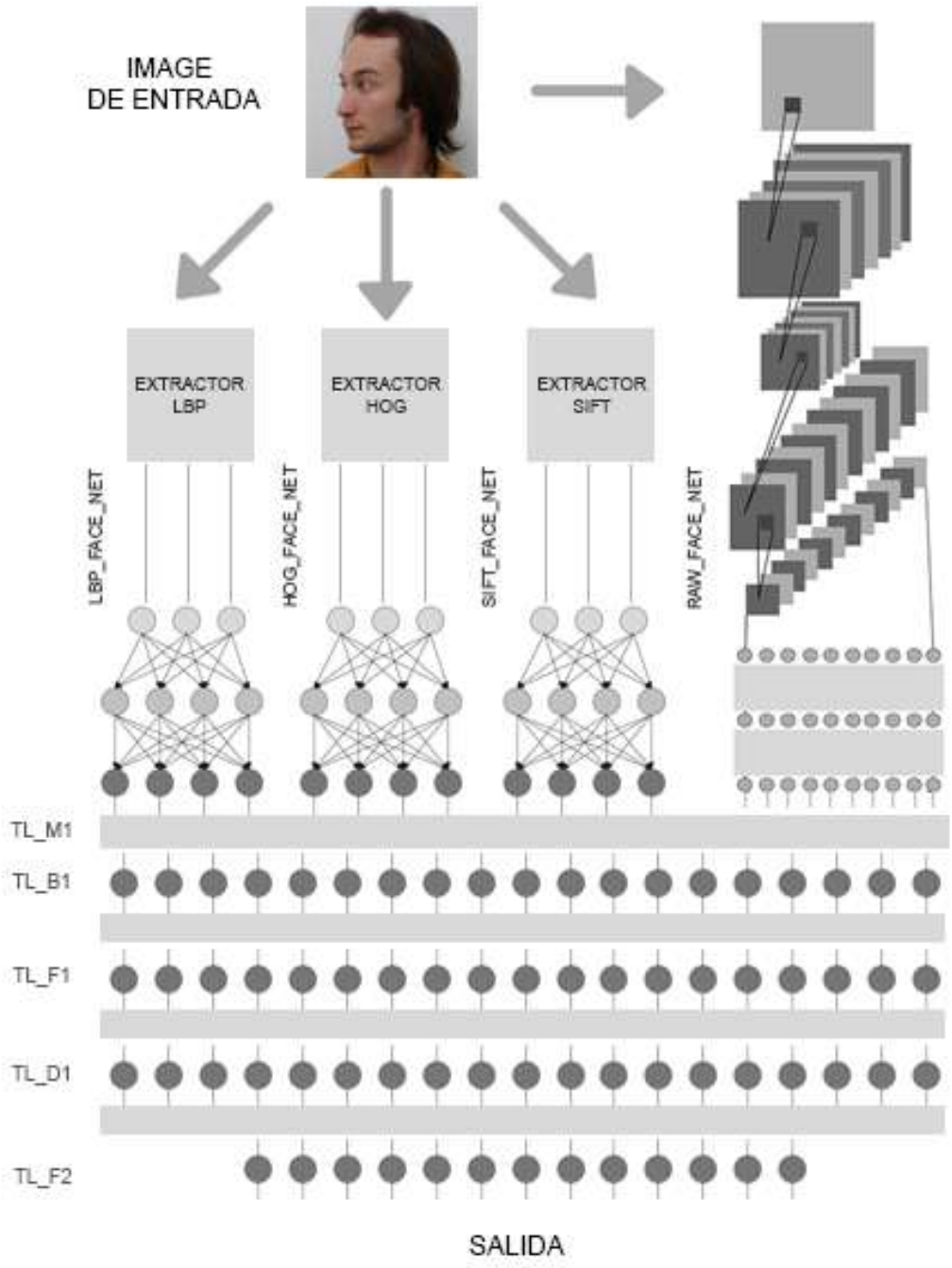


Figura 2.33: Arquitectura del modelo Híbrido TL_FACE_NET.

```
1
2  # importar el modelo
3  json_file = open('LBP_FACE_NET_JSON_ARCHITECTURE.json', 'r')
4  LBP_FACE_NET_JSON = json_file.read()
5  json_file.close()
6  LBP_FACE_NET = model_from_json(LBP_FACE_NET_JSON)
7
8  # Carga de los pesos
9  LBP_FACE_NET.load_weights("LBP_FACE_NET_WEIGHTS.h5")
10
11 # Contruccion de la rama
12 lbpBranch = Model(inputs=LBP_FACE_NET.input,
13            outputs=LBP_FACE_NET.get_layer('LBP_D1').output)
14
15 # Congelamiento de los pesos
16 for layer in lbpBranch.layers:
17   layer.trainable = False
18
19 .
```

Listado de Código 2.11: Creación de una rama del modelo TL-FACE-NET

El procedimiento consiste en cargar el modelo ya entrenado y sus pesos, construir un nuevo modelo cuyas entradas son las mismas del modelo original pero que su salida se tomará de la penúltima capa, en este caso LBP_D1. Este procedimiento se repite para cada uno de los modelos y se utiliza una capa de fusión (Merge()) la cual concatena las salidas de cada rama (Ver listado de código 2.12).

```
1
2 # Fusion de las ramas
3 merged = Merge([lbpBranch, hogBranch,siftBranch,rawBranch], mode='concat',name='TL_M1')
4
5 .
```

Listado de Código 2.12: Fusión de ramas

Hay múltiples beneficios de las redes fusionadas como lo son la flexibilidad de manejar diferentes entradas y la de agregar nuevas características sin tener que lidiar con la estructura de red existente. La longitud del vector de salida de la capa de fusión tendrá una longitud igual a la suma de las longitudes de las salidas de la penúltima capa de los modelos individuales, es decir *2048+512+512+512 = 3584*. Las nuevas capas del modelo TL_FACE_NET (Ver cuadro) posteriores a la capa de fusión recibirán un vector de esta longitud como entrada. El número de pesos a calcular durante el entrenamiento es de 764.368. Recordemos que los pesos de las ramas han sido congelados y solo se utilizan para extraer características antes de la fusión.

Nombre	Tipo	Forma de salida	Parámetros
TL_M1	MERGE	(None, 3584)	0
TL_B1	BATCH_NORMALIZATION	(None, 3584)	14.336
TL_F1	DENSE	(None, 200)	717.000
TL_D1	DROPOUT	(None, 200)	0
TL_F2	DENSE	(None, 200)	40.200
TL_A2	ACTIVATION	(None, 200)	0

Cuadro 2.14: Resumen del modelo RAW_FACE_NET.

Por último de agregan a nuestro modelo capas completamente conectadas que se encargaran de clasificar los rostros tomando como entrada la fusión de las rama anterior. El listado de código 2.13 muestra las distintas capas añadidas al modelo luego de la fusión.

```
1
2  # Capas completamente conectadas del modelo hibrido
3  TL_FACE_NET = Sequential()
4  TL_FACE_NET.add(merged)
5  TL_FACE_NET.add(BatchNormalization(name='TL_B1'))
6  TL_FACE_NET.add(Dense(128, name='TL_F1', activation='relu'))
7  TL_FACE_NET.add(Dropout(0.5, name='TL_D1'))
8  TL_FACE_NET.add(Dense(200, name='TL_F2'))
9  TL_FACE_NET.add(Activation("softmax"))
10
11 TL_FACE_NET.compile(loss='sparse_categorical_crossentropy',
12 optimizer='Adam',
13 metrics=['accuracy'])
14
15 .
```

Listado de Código 2.13: Fusión de ramas

El listado de código 2.14 muestra el uso de la función *"fit()"* para entrenar el modelo. Vemos como nuestros datos de entrenamiento están formados por la tupla [lbpDescriptors, hogDescriptors, siftDescriptors, rawDescriptors], los conjunto de entrenamiento de cada modelo. Es importante señalar que los únicos pesos actualizados durante el entrenamiento serán los pesos de las capas posteriores a la capa de fusión puesto que los pesos de las ramas fueron congelados.

```
1
2  # Entrenamiento del modelo
3  history = model.fit([lbpDescriptors,hogDescriptors,siftDescriptors,rawDescriptors],
4            nb_epoch=55,
5            batch_size = 100,
```

```
6             verbose=1)
7 .
```

Listado de Código 2.14: Fusión de ramas

La figuras 2.34 y 2.35 muestra las gráficas de exactitud y la pérdida para los datos de entrenamiento y validación generadas a partir del historial.

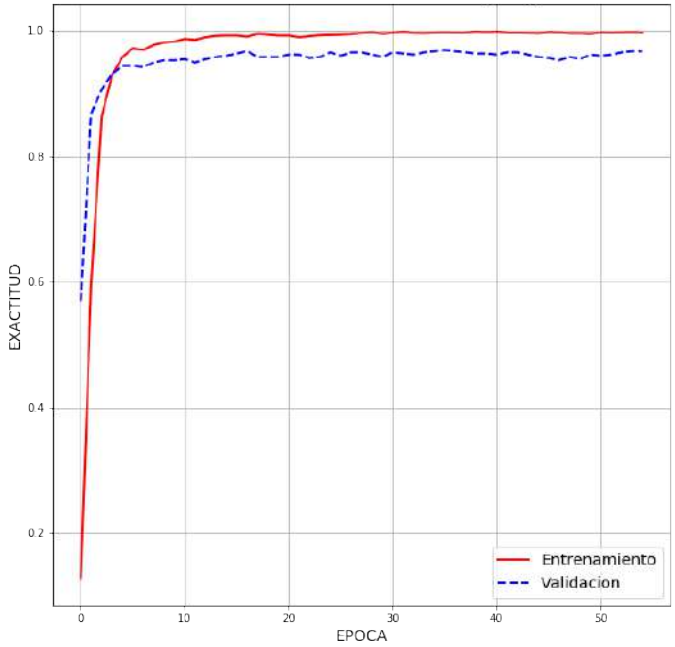


Figura 2.34: Exactitud del modelo TL_FACE_NET.

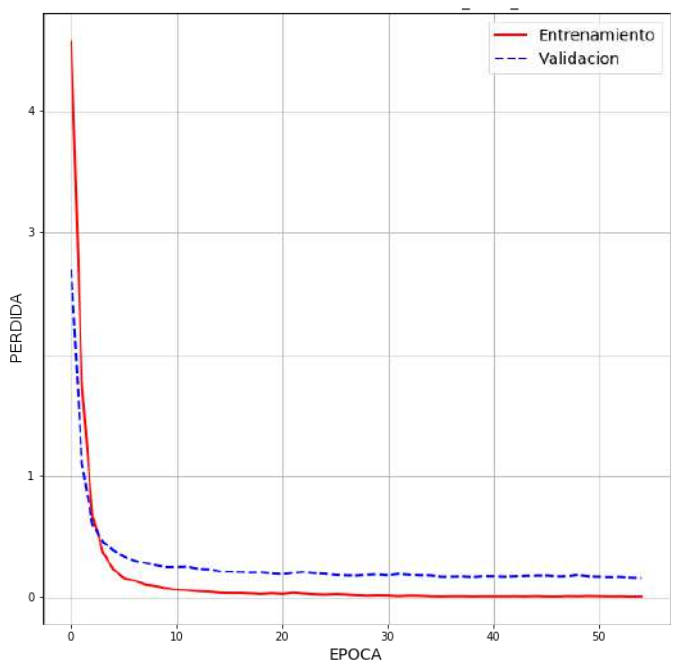


Figura 2.35: Pérdida en el modelo TL_FACE_NET.

El cuadro 2.15 resume los resultados obtenidos para las métricas del clasificador TL_FACE_NET.

Exactitud	Precisión	Recall	f1
0.97	0.97	0.97	0.97

Cuadro 2.15: Métricas obtenidas para el modelo TL_FACE_NET.

El listado de código 2.15 muestra como almacenamos el modelo TL_FACE_NET y los pesos generados durante su entrenamiento.

```
1
2 TL_FACE_NET_JSON= TL_FACE_NET.to_json()
3 open('TL_FACE_NET_JSON_ARCHITECTURE.json','w').write(TL_FACE_NET_JSON)
4 TL_FACE_NET.save_weights('TL_FACE_NET_WEIGHTS.h5',overwrite=True)
5 .
```

Listado de Código 2.15: Almacenando el modelo y los pesos de TL-FACE-NET

2.4. Evaluación e Interpretación

A partir del proceso de entrenamiento se ha obtenido un historial que contiene información del proceso de aprendizaje para cada modelo. En cada uno de ellos estableció el número de épocas en 55, el promedio resultante de las búsquedas exhaustivas ejecutadas sobre los parámetros de entrenamiento. Por otra parte se ha incorporado información de validación durante el entrenamiento con la finalidad de verificar su evolución y su capacidad de generalización. La capacidad de generalización de un modelo se mide a partir de la verificación del comportamiento del mismo con datos distintos a los datos de entrenamiento. Los datos suministrados para la validación consisten en una muestra tomada de la base de datos FEI del 20 % de las imágenes, el 80 % restante forma parte del conjunto de entrenamiento. La figura 2.36 muestra una gráfica de los resultados históricos de exactitud. El modelo SIFT_FACE_NET es el que alcanza la menor exactitud seguido por el HOG_FACE_NET, mientras que los modelos RAW_FACE_NET y LBP_FACE_NET logran mejores niveles con respecto a los anteriores. El clasificador, TL_FACE_NET producto de la fusión de los clasificadores anteriores tiende a generalizar mejor que los individuales alcanzando un nivel de exactitud superior.

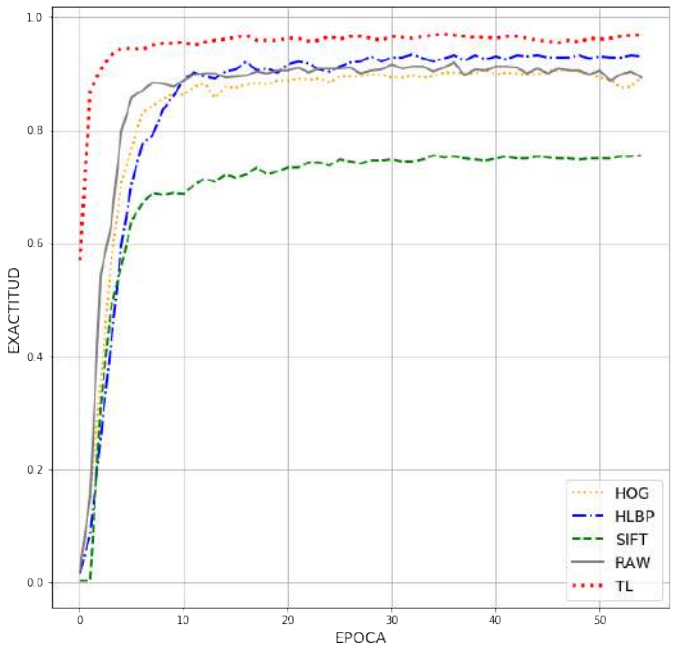


Figura 2.36: Exactitud de los modelos base y el modelo híbrido.

La pérdida disminuye en cada época ya que el modelo aprende más características

de los datos, mientras menor exactitud se obtiene una mayor pérdida, siendo el clasificador de fusión, nuevamente, TL_FACE_NET el que logra un mejor desempeño con una menor pérdida. (Ver figura 2.37)

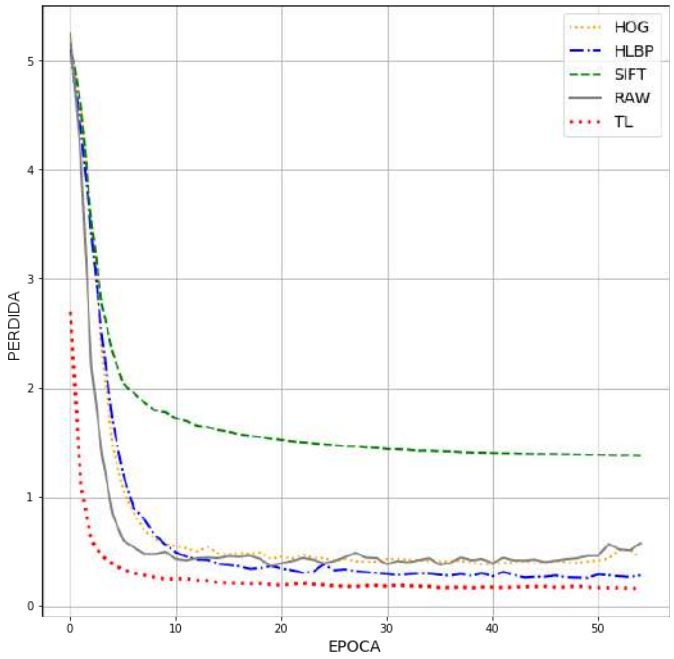


Figura 2.37: Pérdida de los modelos base y el modelo híbrido.

El cuadro 2.16 muestra el resultado de la exactitud para cada unos de los modelos anteriores sobre un conjunto de datos validación. Este cuadro indica que el proceso de fusión de los clasificadores ha sido exitoso, sin embargo la validación a partir de un único conjunto de datos resulta poco efectiva por lo que es necesarios aplicar técnicas más avanzadas. La validación cruzada es una técnica estándar de validación que permite validar el rendimiento de los modelos con mayor efectividad.

Modelo	Exactitud	Pérdida
LBP_FACE_NET	0.93	0.28
HOG_FACE_NET	0.89	0.60
SIFT_FACE_NET	0.75	1.57
RAW_FACE_NET	0.90	0.48
TL_FACE_NET	0.97	0.16

Cuadro 2.16: Exactitud y Pérdida sobre los datos de validación de los modelos.

La validación cruzada *(cross-validation)* sirve para determinar la validez de un modelo, mientras más independiente de los datos con los cuales es entrenado mejor el modelo. Para aplicar esta técnica los datos se dividen en K subconjuntos *(folds)*. Uno de los subconjuntos se utiliza como datos de validación y el resto (K-1) como datos de entrenamiento. El proceso es repetido durante K iteraciones con cada uno de los posibles subconjuntos de datos. La métricas se obtienen de la media aritmética de los de los resultados obtenidos en cada iteración.

```
1
2  # Capas completamente conectadas del modelo hibrido
3  from sklearn.metrics import accuracy_score
4  from sklearn.metrics import f1_score
5  from sklearn.metrics import recall_score
6  from sklearn.metrics import precision_score
7
8  kf = KFold(n_splits=10)
9
10 f1s=[]
11 accs=[]
12 recalls=[]
13 presicions=[]
14
15 fold=1
16
17 for train_index, test_index in kf.split(X_train_lbp_pca):
18
19   json_file_lbp=open('lbp_face_net_pca_json_ARCHITECTURE.json', 'r')
20   loaded_model_json_lbp = json_file_lbp.read()
21   json_file_lbp.close()
22   loaded_model_lbp = model_from_json(loaded_model_json_lbp)
23
24   loaded_model_lbp.compile(loss='sparse_categorical_crossentropy',
25                optimizer='Adam',
26                metrics=['accuracy'])
27
28   loaded_model_lbp.fit(X_train_lbp_pca[train_index],
29               y_train[train_index],
30               verbose=1,
31               validation_data=(X_train_lbp_pca[test_index],y_train[test_index]),
32               epochs=55,
33               batch_size=64)
34
35   y_pred=loaded_model_lbp.predict(X_train_lbp_pca[test_index])
36
37   y_classes = y_pred.argmax(axis=-1)
38
39   f1=f1_score(y_train[test_index],y_classes,average='macro')
40   recall=recall_score(y_train[test_index],y_classes,average='macro')
41   presicion=precision_scoX_train_lbp_pca[test_index]re(y_train[test_index],y_classes,average='
     macro')
42   acc=accuracy_score(y_train[test_index],y_classes)
43
44   f1s=np.asarray(f1s)
45   accs=np.asarray(accs)
46   recalls=np.asarray(recalls)
47   presicions=np.asarray(presicions)
48
49 print("Accuracy Test: %0.2f (+/- %0.2f)" %(accs.mean(), accs.std() * 2))
50 print("\n")
51 print("Precision Test: %0.2f (+/- %0.2f)" %(presicions.mean(), presicions.std() * 2))
```

```
52  print("\n")
53  print("Recall Test: %0.2f (+/- %0.2f)" %(recalls.mean(), recalls.std() * 2))
54  print("\n")
55  print("F1-Score Test: %0.2f (+/- %0.2f)" %(f1s.mean(), f1s.std() * 2))
56
57 .
```

Listado de Código 2.16: Validacion cruzada en Phyton

El listado de código 2.16 muestra la implementación de proceso de validación cruzada en Python. Este procedimiento se repite para todos los modelo simples y para el clasificador híbrido. Las métricas calculadas son la exactitud, precisión, recall y f1. El cuadro 2.17 muestra los resultados de aplicar el procedimiento al los modelos propuestos.

Modelo	Exactitud	Precisión	Recall	f1
LBP	0.95 (+/- 0.03)	0.93 (+/- 0.05)	0.93 (+/- 0.05)	0.93 (+/- 0.05)
HOG	0.86 (+/- 0.04)	0.82 (+/- 0.04)	0.81 (+/- 0.04)	0.80 (+/- 0.05)
SIFT	0.71 (+/- 0.05)	0.64 (+/- 0.07)	0.63 (+/- 0.06)	0.61 (+/- 0.07)
RAW	0.90 (+/- 0.04)	0.87 (+/- 0.06)	0.87 (+/- 0.05)	0.86 (+/- 0.06)
TL	0.99 (+/- 0.01)	0.99 (+/- 0.01)	0.99 (+/- 0.01)	0.99 (+/- 0.01)

Cuadro 2.17: Métricas obtenidas mediante validación cruzada.

La figura 2.38 muestra un *"Heatmap"* o *"Mapa de Calor"* donde los valores menos oscuros se corresponden a los más cercanos al uno y caso contrario los más cercanos al cero. Como podemos observar los valores del promedio de las métricas reportados por la técnica de validación cruzada indican que el modelo con métricas más bajas corresponde al que ha sido entrenado con los descriptores basados en SIFT, mientras que el basado en la técnica de *"Transferencia de aprendizaje"* y la *"Fusión"* de los clasificadores alcanza valores más altos de exactitud, precisión, recall y f1. Uno de los aspectos interesantes de esta gráfica es la intensidad del color similar para los modelos HOG_FACE_NET y RAW_FACE_NET, puesto que de todos los descriptores ambos están basados en la operaciones de convolución sobre las imágenes sobre las que se aplica alguna técnica de reducción de la dimensionalidad como PCA o el submuestreo.

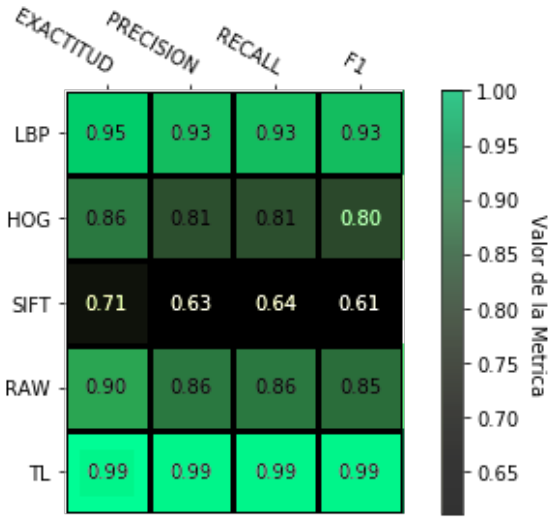


Figura 2.38: Heatmap de las métricas obtenidas en la evaluación de los modelos.

El rendimiento obtenido por el modelo propuesto supera al de los estudios desarrollados por otros investigadores. El modelo TL_FACE_NET ha alcanzado una exactitud cercana al 99 %, lo que constituye un nuevo estado del arte en el reconocimiento de rostros sobre la base de datos FEI. El cuadro 2.18 muestra el valor de la exactitud alcanzado por publicaciones previas donde se utiliza el conjunto de datos FEI.

Nombre de la Publicación	**Algoritmo**	**Exactitud**
Improved Face Recognition Rate Using HOG Features and SVM Classifier [45]	HOG + SVM	0.8013
Person Authentication Using Color Face Recognition [46]	PCA + LBP + YCbCr	0.9057
Person Authentication Using Color Face Recognition [46]	PCA + Gabor + YCbCr	0.9393
Human Recognition from Multi Angular Images [47]	Distancia Euclidiana + Puntos Claves (Frontales)	0.9450

Cuadro 2.18: Métricas de trabajos similares sobre la base de datos FEI.

Naik et al. [47] muestra que el valor máximo de exactitud alcanzado por estudios

similares sobre el conjunto de datos FEI es de 0.9450, considerando unicamente las imágenes frontales. El cuadro 2.18 revela que el modelo propuesto en este Trabajo de Grado es significativamente mas exacto que otros que aplican distintas técnicas de reconocimiento facial publicados hasta el año 2017.

2.5. Difusión y uso

Una vez construido el modelo y evaluadas las métricas correspondientes para asegurar la calidad del mismo, podemos hacer uso del mismo para construir aplicaciones que a partir de estos modelos generen alguna utilidad al usuarios final. Como se ha observado el modelo alcanza excelentes niveles de exactitud para el conjunto de datos FEI. Para demostrar las capacidades del modelo se ha construido un prototipo de aplicación de reconocimiento facial. Esta aplicación recibe como entrada una imagen de la base de datos FEI y es capaz de identificar a al sujeto en la imagen. La figura 2.39 muestra la pantalla inicial.

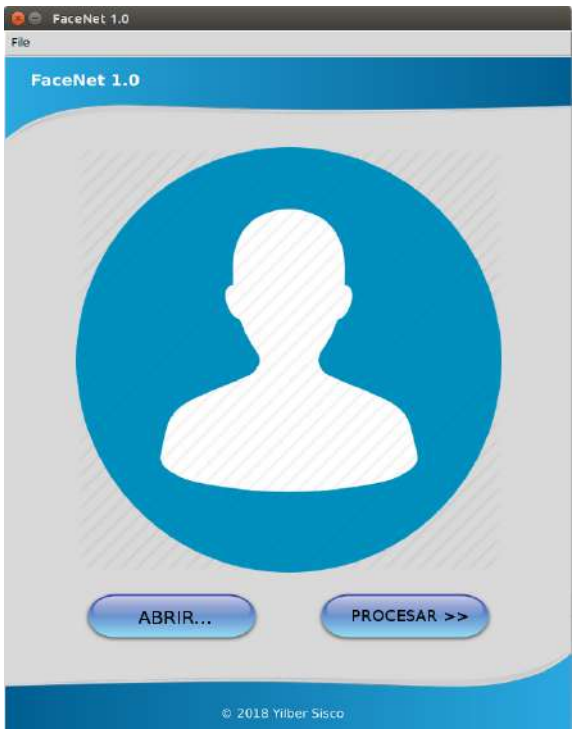


Figura 2.39: Pantalla inicial de la Aplicación.

La pantalla principal cuenta con dos opciones , la opción *Abrir* permite cargar una imagen almacenada en el disco rígido a través de una caja de diálogo. Puesto que el modelo fue entrenado con imágenes de la base de datos FEI, se deben cargar imágenes pertenecientes a la base de datos o de personas registradas en esta base de datos para obtener predicciones adecuadas. La figura 2.40 muestra la pantalla que se despliega al activar la opción *Abrir*.

Figura 2.40: Selección y carga de imágenes.

Una vez seleccionada la imagen que queremos clasificar, es desplegada por el sistema, como muestra la figura 2.41. La imagen del conjunto de datos tiene una dimensión de 640x480 y está en formato RGB.

Figura 2.41: Imagen cargada para su clasificación.

La opción *Procesar* se encarga de recortar la región de interés en la imagen , una región de 400x400 en el centro de la imagen, donde se encuentran centrados los rostros de la base de datos FEI. Además esta opción ajusta los valores RGB de la imagen en un rango de 0 a 255.La figura 2.42 muestra el resultado de este procedimiento.

Figura 2.42: Imagen recortada con el histograma ajustado en el rango 0 a 255.

El proceso de extracción de características para esta imagen se inicia al seleccionar la opción *Extraer*, que ocasiona que se despliegue una pantalla que muestra las imágenes correspondientes a los descriptores: LBP, HOG, SIFT y la imagen RAW correspondientes que son la base del proceso de reconocimiento. Las imágenes se muestran en la figura 2.43

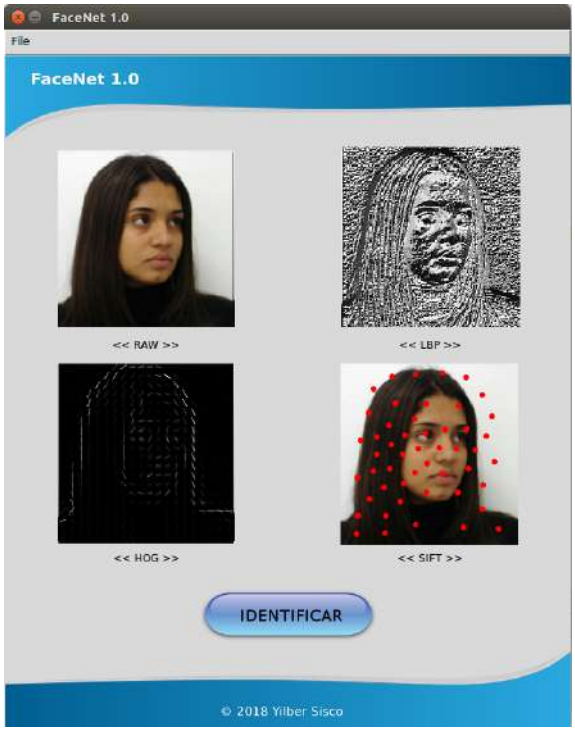


Figura 2.43: Imagen de los descriptores LBP, HOG, SIFT y RAW para la imagen de entrada.

Por último la opción *Identificar* despliega la imagen frontal correspondiente a la persona que aparece en la fotografiá cargada, tal y como se muestra en la figura 2.44

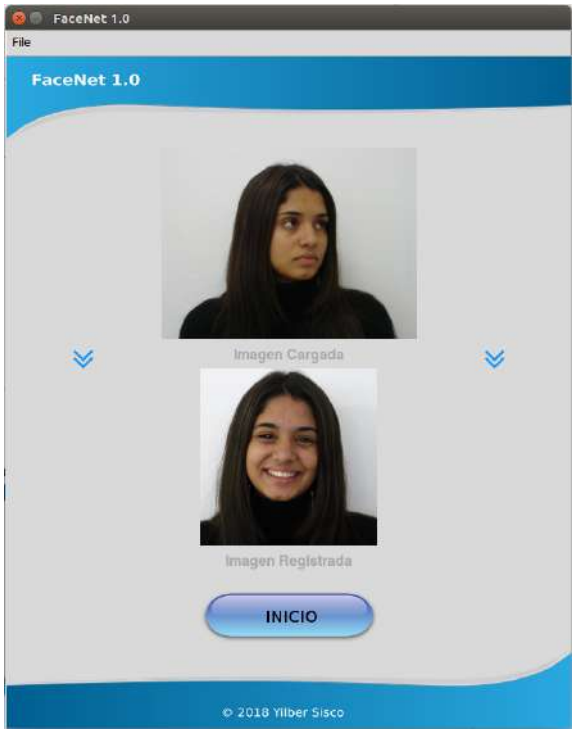


Figura 2.44: Imagen frontal de la persona identificada.

Conclusiones y Trabajos Futuros

En el presente trabajo de grado se han utilizado distintos descriptores de imágenes con el objetivo de identificar rostros humanos. Se observó que cada uno de los algoritmos de procesamiento digital de imágenes, utilizados para extraer características, proporciona información distinta sobre la imagen.

Algunos descriptores como los *"Histogramas de Patrones Binarios"* proporcionan información relacionada con la textura y una descripción global de la imagen. De igual manera los descriptores de *"Histogramas De la Orientación de Gradientes"* ofrecen una representación global de la imagen pero el contenido de esta información se relaciona con los bordes de la imagen ya que se genera a partir de la información de los gradientes de los píxeles. Por el contrario, otros descriptores como los SIFT no ofrecen una representación global de la imágenes pues se generan a partir de puntos claves dentro de la imagen y sus descriptores. Las Redes Neuronales Convolucionales (CNN) permiten obtener descriptores a partir de mapas de características generados mediante operaciones de convolución y submuestreados progresivamente por la estructura de la red.

En las pruebas realizadas encontramos que las redes neuronales entrenadas con descriptores basados en SIFT alcanzaron el menor rendimiento; esto se debe a que este descriptor solo aporta información de los gradientes alrededor de los puntos claves. Por otro lado, las RNA entrenadas con descriptores HLBP alcanzaron una mayor exactitud con respecto al resto de los clasificadores individuales debido a que estos descriptores utilizan el vecindario del píxel y describen tanto los bordes como las texturas lo que se traduce en que las RNA entrenadas con esta representación reportan mejores métricas. Las CNN por otra parte logran una buena exactitud

sobre este conjunto de datos pero sus métricas son inferiores a las que alcanza una RNA entrenada con descriptores HLBP.

Las Redes Neuronales Convencionales (RNA) pueden aprender patrones a partir de los descriptores generados mediante algoritmos de procesamiento digital de imágenes o bien de los que se obtienen mediante el uso de CNN. Si bien las CNN son el paradigma más investigado en la actualidad para problemas de reconocimiento en general, se observó que otros métodos, como por ejemplo, el entrenamiento de RNA con descriptores HLBP alcanza una exactitud superior sobre el conjunto de datos FEI mediante un descriptor de longitud significativamente menor al tamaño de las imágenes utilizadas por las CNN. En consecuencia, debemos tomar en cuenta en la resolución de problemas de visión artificial, y de aprendizaje automático en general, la importancia de los datos en la solución de un problema determinado pues de estos depende en gran medida el rendimiento de los modelos.

La técnica de *"Transferencia del Aprendizaje"* o *"Transfer Learning"* en combinación con la de *"Fusión"* o *"Merge"* de clasificadores ha permitido construir un modelo de clasificación basado en el aprendizaje obtenido por los clasificadores individuales. Este modelo de clasificador Híbrido ha reportado un rendimiento considerablemente mayor al de los clasificadores individuales basados en descriptores e inclusive sobre el modelo construido mediante imágenes crudas (RAW) sobre una CNN.

La técnica de *Transfer Learning* ha permitido reutilizar el conocimiento adquirido por modelos individuales. El modelo híbrido alcanza una exactitud superior a los individuales, mediante el cálculo de un reducido número de parámetros. Las capas de fusión ayudan a combinar los datos proveniente de distintos modelos para construir clasificadores más exactos. El clasificador híbrido generaliza en un menor número de épocas, lo cual se traduce en la reducción del tiempo de entrenamiento y del costo de procesamiento. Nuestro modelo alcanza una exactitud superior al 99 % mediante la combinación del aprendizaje adquirido por los modelos basados en descriptores SIFT, HOG, RAW y HLBP.

El modelo desarrollado en este Trabajo de Grado ha alcanzado una exactitud considerablemente superior sobre el Estado del Arte actual. Un estudio comparativo efectuado sobre publicaciones anteriores que utilizan la base de datos FEI revela que la exactitud de 99 % del modelo basado en *"Transferencia de Aprendizaje"* planteado es significativamente mayor a los resultados alcanzados por otros investigadores [47]

En definitiva, el principal aporte de este trabajo está en el hecho de que se probó experimentalmente que es posible construir un modelo combinado de clasificadores utilizando las técnicas de *"Transferencia del Aprendizaje"* y *"Fusión"* de clasificadores, alcanzando una exactitud significativamente mayor en relación a los modelos bases entrenados con el conjunto de datos FEI. La combinación de los descriptores HOG, HLBP, SIFT y el aprendido por CNN permite obtener información que ayuda al algoritmo de aprendizaje a discriminar mejor entre las distintas clases.

Para futuros trabajos, se propone comparar los resultados de aplicar la técnica propuesta sobre distintos conjuntos de datos de imágenes de rostros. También se plantea la posibilidad de evaluar su rendimiento e intentar generalizar nuestro método para conjuntos de datos mas complejos.

Bibliografía

[1] Y. Bengio, A. Courville y P. Vincent, *Representation Learning: A Review and New Perspectives* , Department of computer science and operations research, U. Montreal and Canadian Institute for Advanced Research (CIFAR). 2014

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE, Vol. 86, No. 11, p.p. 2278-2324, 1998.

[3] Dalal, N. y Triggs, B., *Histograms of Oriented Gradients for Human Detection*, In: Proc. CVPR 2005, vol. 1, pp. 886–893, 2005

[4] O. Déniz, G. Bueno, J. Salido y F. De la Torre, *Face Recognition Using Histograms of Oriented Gradients*, Pattern Recognition Letters , Vol. 32, No.12 , pp. 1598–1603, 2011

[5] I. Jolliffe, *Principal Component Analysis*, Springer-Verlag, New York, Secaucus, NJ, 1986.

[6] A. Mohamed, *Face Recognition Using SIFT Features* CNS/Bi/EE report 186, 2006.

[7] D. Lowe, *Distinctive image Features from Scale-invariant Keypoints*, International Journal of Computer Vision, Vol. 60, no. 2, 91–110, 2004.

[8] Mr. Amit Kr. Gautam, Ms. Twisha, *Improved Face Recognition Technique using SIFT*, IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), 2014.

[9] M. Turk and A. Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, Massachusetts Institute of Technology, vol.3, no. 1, pp. 71-86, 1991.

[10] P. Belhumeur, J. Hespanha, y D. Kriegman, *Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19 , No. 7, pp. 711–720, 1996.

[11] D. Swets, J. Weng, *Using discriminant eigenfeatures for image retrieval*, IEEE Transactions On Pattern Analysis And Machine Intelligence,18(8):831–836, 1996

[12] A. Kumar, M. Datta y P. Kumar, *Face Detection and Recognition", Theory and Practice*, CRC Press, Taylor and Francis Group, pp. 19-39, 2016.

[13] S. Li y A. Jain, *Handbook of Face Recognition*, Second Edition, Springer, pp.13-37, Oxford, 2011.

[14] J. Palma y R. Marín, *Inteligencia Artificial: Técnicas, Métodos y Aplicaciones*, Mac Graw Hill, pp. 674-682, 2008.

[15] A. Karpathy, F. Li y J. Johnson, *CS231n: Convolutional Neural Networks for Visual Recognition*, Disponible: http://cs231n.stanford.edu/, Enero, 2017.

[16] Deep Learning for Java, *Disponible: https://deeplearning4j.org*,Enero 2017.

[17] A. Peña, E. Valveny y M. Vanrell, *Detección de Objetos*, Disponible: https://es.coursera.org/learn/deteccion-objetos, Enero, 2017.

[18] J. Heaton, *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*, Heaton Research, Vol. 3, 2015.

[19] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning", MIT Press, Massachusetts Institute of Technology*,2016

[20] A. Fisher, *The use of multiple measures in taxonomic problems*,Ann. Eugenics, No. 7, pp. 179–188, 1936.

[21] A. Davison, *Vision-based User Interface Programming in Java*, Kindle Edition, 2013.

[22] Y. Taigman, M. Yang, M. Ranzato y Lior Wolf, *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1701-1708, 2014.

[23] T. Ahonen, A. Hadid y M. Pietikainen, *Face description with Local Binary Patterns*,, IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 2037-2041, 2006.

[24] D. Changxing y T. Dacheng, *Robust Face Recognition via Multimodal Deep Face Representation*,IEEE Trans. Multimedia Vol. 17, No. 11, pp. 2049–2058 , 2015.

[25] T. Zhang, W. Zheng, Z. Cui, Y. Zong, J. Yan, y K. Yan, *A Deep Neural Network Driven Feature Learning Method for Multi-view Facial Expression Recognition*, IEEE Trans. on Multimedia, Vol. 18 ,No.12 , pp. 2528 – 2536, 2016.

[26] B. Sun, L. Li, G. Zhou, y J. He, *Facial expression recognition in the wild based on multimodal texture features*, J. Electron. Imaging, vol. 25, no. 6, pp. 061407–061407, 2016.

[27] S. Kahou , *Combining Modality Specific Deep Neural Networks for Emotion Recognition in Video,*, 15th ACM on International Conference on Multimodal Interaction, New York, NY, USA, pp. 543–550, 2013

[28] R. Wirth y J. Hipp, *CRISP-DM: Towards a standard process model for data mining*,4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, pages 29–39, Manchester, UK, 2000

[29] Viola, P. and Jones, *Robust Real-Time Face Detection*, International Journal of Computer Vision, Vol. 57, No. 2, pp. 137-154, 2004.

[30] Labeled Faces in the Wild, *Disponible: http://vis-www.cs.umass.edu/lfw*,Febrero, 2017

[31] S. Busuttil, *Support Vector Machines*, Department of Computer Science and AI, University of Malta, 2009.

[32] N. Mastorakis, M. Popescu, V. Balas y L.Perescu, *Multilayer Perceptron and Neural Networks*, Faculty of Electromechanical and Environmental Engineering, University of Craiova, Bulgaria, 2010.

[33] Ch. Tsai, *Bag-of-Words Representation in Image Annotation: A Review*, ISRN Artificial Intelligence, 2012.

[34] L. Breiman, *Bagging Predictors*, Department of Statistics, University of California, 1994.

[35] T. Dietterich, *Ensemble Methods in Machine Learning*, Volume 1857 of the series Lecture Notes in Computer Science pp 1-15, 2000.

[36] R.Senthilkumar1, R.K. Gnanamurthy, *Face Databases for 2D and 3D Facial Recognition: A Survey*, Institute of Road and Transport Technology, 2014.

[37] E. Keogh, A. Mueen, *Curse of Dimensionality*, Encyclopedia of Machine Learning, pp. 257-258, 2010.

[38] V. Nair y G. Hinton, *Rectified linear units improve restricted Boltzmann machines*, ICML,2010

[39] A. Gulli y S. Pal , *Deep Learning with Keras*, Packt ,2017

[40] TensorFlow, *Disponible: https://www.tensorflow.org/*, Junio 2018.

[41] scikit-learn, *Disponible: http://scikit-learn.org*, Junio 2018.

[42] Y. Tayade1, S.M. Bansode, *An Efficient Face Recognition and Retrieval Using LBP and SIFT*, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, No. 4, 2013.

[43] OpenCV, *Disponible: https://www.opencv.org/*, Junio 2018.

[44] Python, *Disponible: https://www.python.org/*, Junio 2018.

[45] H. Santosh y G.Mohan, *Improved Face Recognition Rate Using HOG Features and SVM Classifier*, IOSR Journal of Electronics and Communication Engineering, Vol. 11, No. 4, pp. 33-34, 2016.

[46] K.Davakhar, S. B. Mule y A. Deshmukh, *Person Authentication Using Color Face Recognition*, Int. Journal of Engineering Research and Applications, Vol. 3, No. 5, pp. 178-182, 2013.

[47] R. Naik y K. Lad, *Human Recognition from Multi Angular Images*, International Conference on Research and Innovations in Science, Vol. 2, pp. 1-12, 2017.