UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Inteligencia Artificial

TRABAJO DE DIPLOMA

Desarrollo de una biblioteca para el reconocimiento de emociones faciales.

Guillermo Soto Gómez

Gerardo Martínez Rodríguez

Tutor: Ing. Roberto Vicente Rodríguez

Santa Clara, junio, 2018

“Talk is cheap, show me the code.”

Linus Torvalds

RESUMEN

Recientes estudios neurológicos indican que el rol de la emoción en la cognición humana es esencial y que las emociones no son un lujo, en vez de eso, estas juegan un papel importante en la toma de decisiones, en la percepción y en la interacción humana.

Las personas que trabajan con computadoras, generalmente pasan más tiempo con estas del que pasan con el resto de las personas; por tanto, las computadoras están en una única posición para sentir nuestro estado afectivo. En el presente trabajo se desarrolla una biblioteca que permite la detección automática de emociones faciales en tiempo real en aplicaciones de Computación Afectiva.

El lenguaje de programación utilizado es Python y se hacen uso de los módulos OpenCV, Dlib, y Sklearn para la detección del rostro, extracción de los rasgos correspondientes a los ojos, nariz y boca y para la clasificación utilizando Máquinas de Vectores Soporte y Perceptrón Multicapa.

La biblioteca fue probada en los conjuntos de datos Cohn-Kanade y FER2013 obteniendo resultados similares a los que se muestran en la literatura consultada.

ABSTRACT

Recent neurological studies indicate that the role of emotion in human cognition is essential and that emotions are not a luxury, instead, they play an important role in decision making, perception and human interaction.

In addition, people who work with computers generally spend more time with these than they do with other people; therefore, computers are in a unique position to feel our affective state. In the present work a library is developed that allows the automatic detection of facial emotions in real time in Affective Computing applications.

The programming language used is Python and the OpenCV, Dlib, and Sklearn modules are used for face detection, extraction of the features corresponding to the eyes, nose and mouth and for classification using Multilayer Perceptron and Support Vector Machines.

The library was tested in the Cohn-Kanade and FER2013 datasets, obtaining results similar to those shown in the literature consulted.

Tabla de contenido

Lista de Figuras

Lista de Tablas

INTRODUCCIÓN

Los seres humanos poseen la capacidad de comunicarse a través de emociones faciales en las interacciones con otros. Algunos estudios para percibir emociones faciales han propiciado la creación de entornos de interacción del humano con la computadora.

En los últimos años, se ha desarrollado un creciente interés en mejorar todos los aspectos de la interacción entre humanos y computadoras especialmente en el área del reconocimiento de emociones humanas al observar expresiones faciales.

La Computación Afectiva consiste en el desarrollo de sistemas inteligentes capaces de proporcionar a las computadoras la habilidad de reconocer, interpretar y procesar las emociones humanas (Picard, 1997). Desde que se acuñó el término Computación Afectiva a finales de los años 90, han aparecido en la literatura numerosos trabajos enfocados a la extracción automática de afecto.

Se han desarrollado hasta la fecha sistemas que reconocen las emociones a través de diferentes canales humanos, como en expresiones faciales (Hammal *et al.*, 2007), señales fisiológicas (Chanel, Ansari-Asl and Pun, 2007), voz (Graciarena *et al.*, 2006b), texto (Graciarena *et al.*, 2006a), etc. con altas tasas de acierto.

El rol del reconocimiento automático de emociones está creciendo de forma continua actualmente. Esto se debe a que se ha aceptado la importancia que tiene la reacción de la computadora a los estados afectivos del usuario en la interacción persona-computadora.

A medida que las computadoras se vuelven más y más sofisticadas, ya sea a nivel profesional o social, se torna más importante que estas sean capaces de interactuar de forma natural, similar a como se interactúa con otros agentes humanos. La característica más importante de la interacción humana que garantiza la realización del proceso de forma natural, es el medio por el cual podemos inferir el estado emocional de otros. Esto permite ajustar los patrones de comportamiento y respuestas, optimizando el proceso interactivo tal y como se señala en Fragopanagos and Taylor (2005).

La interacción de la persona con la computadora puede ser mejorada en gran medida si se toma en cuenta el estado emocional del ser humano, haciéndola más cercana y natural; elementos ausentes

en la gran mayoría de los sistemas actuales. Diversos autores tratan este tema, y la investigación en este campo ha sido numerosa en la última década. En el trabajo de Picard, Vyzas and Healey (2001) se dice que el reconocimiento de los estados afectivos del usuario es un problema muy importante en el ámbito de la interacción humano-computadora. En este mismo trabajo, se dice que "se ha argumentado que la inteligencia emocional humana es incluso más importante que la inteligencia matemática o verbal". Es por este motivo, que se considera que las máquinas deben incluir este tipo de inteligencia de reconocer el estado afectivo dado ciertas señales psicológicas.

Respecto a este tema, ha habido diversos esfuerzos desde distintos puntos de vista. Se han descrito técnicas para el análisis de la voz humana o de las expresiones faciales, sin embargo, la aplicación de estas tecnologías en sistemas reales sigue siendo escasa.

Uno de los trabajos pioneros en este campo (Ekman and Friesen, 1975) sugiere que existen seis expresiones faciales prototípicas básicas reconocidas universalmente. Estas son: ira, asco, miedo, alegría, tristeza y sorpresa. El reconocimiento automatizado de estos seis estados básicos es el primer paso para implementar una solución, que, a diferencia de otras, es universal, ya que es común a todos los seres humanos.

Existen varias bibliotecas (informáticas) que implementan métodos y funciones que desarrollan las diferentes etapas que intervienen en un proceso de detección de emociones, a partir del rostro humano; ejemplo de las mismas son OpenCV, DLib, Scikit-Learn, etc. Sin embargo, el empleo de sus recursos requiere de un conocimiento profundo de las mismas y de los lenguajes en que están implementadas (C++, Python). Es muy conveniente poder contar con una biblioteca que implemente métodos que integren y simplifiquen la forma de reconocer las emociones expresadas en el rostro humano, de manera que puedan ser empleados en diferentes aplicaciones.

En la revisión bibliográfica realizada no se han evidenciado soluciones definitivas a esta situación problémica, por lo que se justifica el planteamiento del siguiente problema científico:

La falta de una biblioteca que integre los recursos informáticos existentes para la detección de emociones reflejadas en el rostro humano.

Del planteamiento anterior se derivan las siguientes **preguntas de investigación**:

- ¿Cuál es el estado del arte en el estudio del reflejo facial de las emociones y su reconocimiento mediante el empleo de las Tecnologías de la Información y la Comunicación (TIC)?
- ¿Qué facilidades debe brindar una biblioteca informática para la detección de emociones?
- ¿Qué aspectos deben considerarse en la implementación de una biblioteca para la detección de emociones que se reflejan en el rostro humano?
- ¿Cómo validar la efectividad de la biblioteca propuesta?

A partir del problema planteado se proponen el objetivo general y objetivos específicos siguientes:

Objetivo general:

Desarrollar una biblioteca que implemente métodos y funciones para la detección de emociones humanas analizando las expresiones faciales en tiempo real.

Objetivos específicos:

- Estudiar las tendencias actuales en la detección y clasificación de las emociones del rostro humano, así como las tecnologías que permiten su reconocimiento.
- Fundamentar los recursos informáticos (lenguajes y bibliotecas) que deben ser empleados en la implementación de una biblioteca que permita el reconocimiento de emociones.
- Implementar una biblioteca que permita el reconocimiento de emociones faciales.
- Validar los resultados de la biblioteca implementada.

CAPÍTULO 1. Emociones humanas, su clasificación y detección automática

En este capítulo se presenta una revisión bibliográfica para introducir y facilitar la comprensión de los tópicos: emociones, expresiones faciales y computación afectiva. En el caso de la emoción se profundiza en sus funciones y en los tipos que existen. De las expresiones faciales se analizan varios aspectos como la detección y representación del rostro y el reconocimiento de las emociones en los mismos.

Igualmente se describen el modelo que detalla el proceso de reconocimiento de emociones y los algoritmos utilizados para cada etapa del mismo.

1.1 Emociones Humanas

El ser humano ha observado cómo detectar las emociones desde la antigüedad. Para ello se han utilizado diversos métodos, siendo uno de los más habituales la observación de expresiones faciales y corporales.

En el siglo XIX, un obrero de los ferrocarriles llamado Phineas Gage, tuvo un accidente (el 13 de septiembre de 1848) que le produjo daños severos en el cerebro al ser atravesado su cráneo por una barra de hierro. A pesar de la gravedad sobrevivió, aunque con secuelas que le produjeron cambios en su comportamiento y personalidad. El estudio de estos cambios indicó que lesiones en el lóbulo frontal podían producir cambios en las emociones, la personalidad y la interacción social de los individuos. Este caso es considerado el inicio de los estudios de la base biológica del comportamiento.

Un poco más tarde, en 1872, Charles Darwin escribió: “La expresión de las emociones en hombres y animales” (Darwin and Fernaández Rodríguez, 1984), donde relacionaba las respuestas faciales con estados emocionales idénticos en todos los seres humanos. Esto le permitió intuir un origen evolutivo de ciertas emociones.

En 1884 James y Lange (Sutil, 1998) propusieron una teoría según la cual la corteza cerebral recibe estímulos que provocan respuestas fisiológicas que a su vez provocan las emociones. Así, la acción de llorar provoca tristeza, y no al contrario.

Pero en el siglo XX, Cannon y Bard (Palmero Cantero, 1996) presentaron una nueva teoría que contradecía la de James-Lange. Según ella, las emociones no eran provocadas por estímulos fisiológicos, sino que ambos se producían simultáneamente. Por tanto, las respuestas emocionales y los sentimientos ocurrirían al mismo tiempo.

En 1937, James Papez postuló un circuito neuronal para las emociones (Dalgleish, 2004). Este circuito está formado por diferentes estructuras nerviosas involucradas en el control de las emociones.

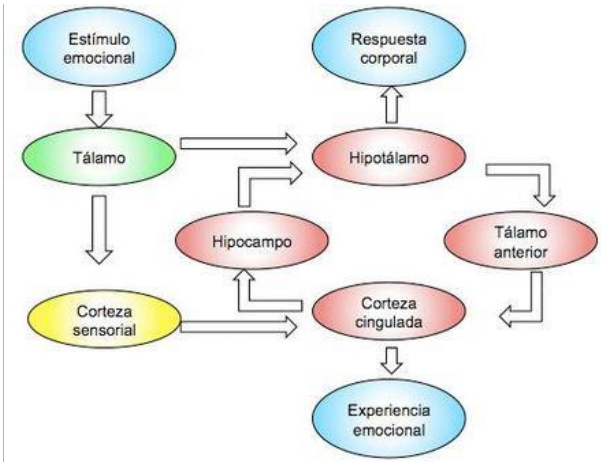

Figura 1.1: Circuito de Papez.

Ampliando los estudios anteriores, MacLean propuso un sistema límbico. Según el mismo el cerebro se divide en tres sistemas (López Mejía *et al.*, 2009):

- El cerebro reptiliano, en el que se observan emociones primitivas (agresión, miedo).
- El cerebro de mamífero, que aumenta las respuestas emocionales del cerebro reptiliano y elabora las emociones sociales.
- El nuevo sistema de mamífero, representa la interfaz de la emoción con la cognición.

El siguiente gran paso lo dieron Schachter y Singer (Pérez Nieto and Delgado, 2018) con su teoría de la activación cognitiva. Para ello se basaron en estudios de Gregorio Marañón en los cuales inyectaba epinefrina en pacientes y a continuación les preguntaba por sus sensaciones. Los pacientes indicaron notar ciertas sensaciones, pero ante la falta de estímulos, no fueron capaces de relacionarlos con una emoción concreta.

1.2 Ekman y las expresiones faciales

Siguiendo la línea de Darwin y James, Paul Ekman pensaba que las expresiones faciales de las emociones tenían un origen biológico y no cultural. Así, personas de diferentes culturas deberían mostrar con las mismas expresiones las mismas emociones, siendo estas expresiones universales (Ekman and Rosenberg, 2005).

Para demostrarlo realizó un estudio (Ekman, Sorenson and Friesen, 1969) en el que mostraba diferentes fotografías a integrantes de una tribu aislada de Papua Nueva Guinea. Dichas fotografías pertenecían a sujetos de otras culturas totalmente diferentes y observó que eran capaces de reconocer las emociones que expresaban con un alto grado de acierto. De este modo concluyó que existe un conjunto de emociones biológicamente universales en la especie humana. En 1972 elaboró una lista de las seis emociones que pensó formaban este conjunto:

- alegría
- ira
- miedo
- asco
- sorpresa
- tristeza

Por tanto, era posible mediante el estudio de las expresiones faciales descubrir cuál de estas emociones estaba sintiendo un sujeto.

Existe una séptima emoción, desprecio, que no fue incluida pues, aunque se encontraron indicios de ser universal, no eran tan claros. Se puede llegar a considerar una mezcla entre ira y asco.

En los años 90, Ekman creó una lista adicional en la que no todas las emociones pueden ser codificadas por expresiones faciales, y en la que sí incluía la emoción desprecio (Ekman, 1993):

- alivio
- bochorno
- complacencia o contento
- culpa

- diversión
- desprecio o desdén
- entusiasmo o excitación
- felicidad
- ira o rabia
- miedo o temor
- orgullo o soberbia
- placer sensorial
- repugnancia, repulsa, asco o repulsión
- satisfacción
- sorpresa
- tristeza
- vergüenza

1.3 Computación afectiva

La computación afectiva es una rama de la inteligencia artificial que intenta diseñar y desarrollar sistemas y dispositivos capaces de reconocer, interpretar y procesar emociones humanas. Para ello utiliza tanto la detección del lenguaje corporal como el habla. Se considera la precursora de la computación afectiva a Rosalind Picard y su artículo "Affective Computing" (Picard, 1995).

La Computación Afectiva es de gran importancia ya que se ocupa de:

- El reconocimiento de emociones (y de expresiones emotivas) humanas por parte de una computadora.
- La simulación (o generación) de estados y expresiones emocionales con computadoras.

En la primera, el objetivo es captar aquellos signos relacionados con la expresión de emociones y lograr interpretar estados emocionales en función de dichos signos.

En la segunda, se intenta que las computadoras puedan simular procesos emocionales en base a ciertos modelos. Aquí se puede reflexionar respecto a si una computadora puede realmente tener emociones, pero, esta disciplina sólo intenta simular dichos procesos de forma tal que resulten verosímiles, dejando de lado estas controversias. Si bien el fin último es desarrollar ambas líneas

para lograr la mejor interacción humano-computadora posible, estas problemáticas pueden ser abordadas de forma aislada.

El rápido crecimiento de la Computación Afectiva se ve reflejado en la cantidad de aplicaciones y trabajos realizados hasta el momento. Estos trabajos abarcan áreas muy diferentes entre sí, desde seguridad, salud, educación, entretenimiento, robótica o marketing. La evaluación en tiempo real de emociones como el estrés, el aburrimiento o la distracción puede ser de gran valor en trabajos en los que se realizan tareas repetitivas, pero en los cuales la atención es crucial, como por ejemplo en control de tráfico aéreo o la supervisión de una planta nuclear.

Existen numerosas aplicaciones terapéuticas para ayudar a personas con problemas emocionales. Por ejemplo, se utilizan sensores fisiológicos para monitorizar el cuerpo y proporcionar información visual a personas con autismo para que sean capaces de reconocer su propio estado emocional y el de otras personas.

A pesar del gran potencial que ofrece la computación afectiva y la interacción emocional, especialmente en campos en los que es complicado detectar cómo se siente el usuario (por ejemplo, en el caso de una persona autista) hay que tener en cuenta que actualmente el mayor esfuerzo en desarrollar esta tecnología viene dado por grandes compañías, para aplicarlo a sus campañas de marketing. Por lo tanto, hay que reflexionar y considerar las implicaciones éticas, ¿no estamos manipulando sus emociones para provocar ciertas reacciones? ¿No estamos invadiendo su privacidad? Aunque todo el software que recoge datos de emociones requiere autorización explícita del usuario, en muchos casos los consumidores no son conscientes de haber dado su consentimiento. Por lo tanto, al diseñar sistemas interactivos afectivos hay que informar de forma clara si se monitoriza o registra información emocional o privada.

1.4 Retos en la detección de expresiones faciales

El principal problema encontrado en la detección de expresiones faciales ha sido la exactitud de las técnicas utilizadas. Con el tiempo estas tecnologías han ido avanzando y cada vez se consigue mayor exactitud.

Otro problema se encuentra en la falta de conjunto de datos grandes donde entrenar y evaluar los procedimientos implementados. Además, los conjuntos de datos disponibles suelen ser de actores posando, de manera que la expresión de la emoción es muy exagerada. Esto puede facilitar el

correcto funcionamiento de las diversas técnicas con el conjunto de datos, pero una dificultad mayor al intentar extrapolar a fotos reales en poses más naturales.

1.5 Sistemas automáticos de reconocimiento de emociones faciales

Los sistemas automáticos de reconocimiento de emociones faciales se basan todos en un modelo que se dividen en etapas como las que aparecen representadas en el siguiente esquema (Zapatero Olmedillo, 2016):

Figura 1.2: -Etapas de un sistema automático de reconocimiento de emociones faciales.

Primero la imagen es realzada para aumentar la efectividad en la detección del rostro, luego se detecta el rostro, seguidamente se extraen de ella las características más significativas, y, por último, el clasificador previamente entrenado, detecta la emoción en función de las particularidades de cada una.

El clasificador se entrena a partir de un conjunto amplio de imágenes de las que se conoce qué emoción representa. Para ello se toma cada imagen y se realizan unos pasos similares a los descritos anteriormente: realce de la imagen, detección del rostro, extracción de rasgos y asociación de esta a la emoción representada.

1.6 Realce y normalización de la imagen

Las variaciones en la dirección y la intensidad de la iluminación son factores que modifican significativamente la apariencia de los objetos en una imagen digital (Negi and Bhandari, 2014). La iluminación ambiental puede variar a lo largo del día, en ambientes interiores y exteriores. Debido a la forma tridimensional de los objetos, una fuente de iluminación puede generar sombras que acentúan o disminuyen ciertos rasgos de la imagen; más aún, distintas condiciones de iluminación pueden producir representaciones desiguales de un mismo objeto, dichas variaciones son indeseables pues dificultan el proceso de reconocimiento de patrones (Hussain Shah *et al.*, 2015).

La ecualización del histograma es un método para ajustar el contraste de una imagen modificando su histograma de manera que este siga una distribución uniforme. Una característica importante es que puede ser usada como una herramienta de normalización. Para un conjunto de imágenes, se puede ecualizar cada histograma de manera tal que todas las imágenes resulten condicionadas a las mismas variaciones de iluminación.

Esta transformación retorna buenos resultados cuando los valores de distribución de los píxeles son similares por toda la imagen. Cuando una imagen contiene regiones que son significativamente más oscuras o brillantes que el resto de la imagen, el contraste de estas regiones no será bien procesado.

La ecualización adaptativa del histograma es un método que mejora la ecualización ordinaria al transformar cada píxel por el resultado de una función de transformación derivada de su región vecina (Magudeeswaran and Singh, 2017). En este procedimiento, la imagen es dividida en pequeñas regiones, luego éstas son ecualizadas siguiendo el método de ecualización ordinario. Como resultado, cada histograma ecualizado está vinculado a una pequeña región de la imagen. Al ecualizar por regiones, el ruido presente en éstas será de más importancia, por tanto, será amplificado como resultado. Para evitar amplificar el ruido, se aplica el método de ecualización adaptativa del histograma con contraste limitado. Este método básicamente consiste en distribuir los píxeles de un nivel de intensidad que sobrepase un límite dado, en otros niveles de intensidad de manera uniforme antes de aplicar la ecualización del histograma. En la Figura 1.3 (OpenCV: Histograms - 2: Histogram Equalization, 2018) se muestra un ejemplo de ecualización del histograma en una imagen.

a) Imagen Original

b) Ecualización global c) Ecualización Adaptativa

Figura 1.3: Ejemplo de ecualización del histograma en una imagen.

1.7 Detección de la cara o reconocimiento facial

El reconocimiento facial es una tarea sencilla para los humanos. Los experimentos (Turati *et al.*, 2006) han demostrado que incluso los bebés de uno a tres días son capaces de distinguir entre caras conocidas. Entonces, ¿qué tan difícil podría ser para una computadora? Resulta que, hasta la fecha, se conoce poco sobre cómo el cerebro analiza y codifica una imagen. David Hubel y Torsten Wiesel (Wurtz, 2009) muestran que nuestro cerebro tiene células nerviosas especializadas que responden a las características locales específicas de una escena, como líneas, bordes, ángulos o movimiento.

Los primeros métodos de detección y seguimiento de la cara se empezaron a desarrollar a partir de principios de la década de 1990. (Hjelmås and Low, 2001) reunieron los métodos de detección facial que existían hasta el 2001 dividiéndolos en dos grandes grupos: métodos basados en rasgos y métodos basados en imágenes. (Hatem, Beiji and Majeed, 2015) la detalla en mayor profundidad dividiendo además estos grupos en subgrupos.

1.7.1 Algoritmo Viola-Jones

Un método de detección de rostros de gran éxito es el desarrollado por (Viola and Jones, 2001). Es un método de detección de objetos que se usa ampliamente en la detección de caras en imágenes y video. El algoritmo se basa en la comparación entre las intensidades luminosas de regiones rectangulares de las imágenes denominadas *Haar-like features* las que calcula empleando una imagen integral.

Este método, englobado dentro de los basados en características y en subgrupo de análisis de características, ha sido frecuentemente utilizado gracias a su rapidez y a los grandes resultados que proporciona en caras de frente. Debido al bajo coste computacional, este procedimiento, supuso una revolución en el campo de detección de rostros, permitiendo por primera vez la detección de caras en tiempo real en dispositivos de modestas prestaciones.

El método de Viola-Jones y variaciones de este siguen siendo ampliamente utilizadas en la actualidad.

Haar-like features

Las *Haar-like features* son los elementos básicos con los que se realiza la detección. Son rasgos o características muy simples que se buscan en las imágenes y consisten en la diferencia de intensidades luminosas entre regiones rectangulares adyacentes, por tanto, estos rasgos son definidos por rectángulos cuya posición es relativa a la ventana de búsqueda y adquieren un valor numérico resultado de la comparación que evalúan. En el trabajo presentado por Viola-Jones se usan específicamente tres tipos de rasgos Haar, en la Figura 1.4 (Viola and Jones, 2001) pueden ser observados estos tres tipos de rasgos. El valor de un rasgo de dos rectángulos es la diferencia entre las sumas de los píxeles contenidos en ambos rectángulos. Las regiones tienen la misma área y forma y son horizontal o verticalmente adyacentes. Un rasgo de tres rectángulos calcula la diferencia entre la suma de los píxeles de las regiones exteriores y la suma de la región del centro multiplicada por un peso para compensar la diferencia de áreas. Los rasgos de cuatro rectángulos calculan la diferencia entre las áreas de cada par de rectángulos en diagonal.

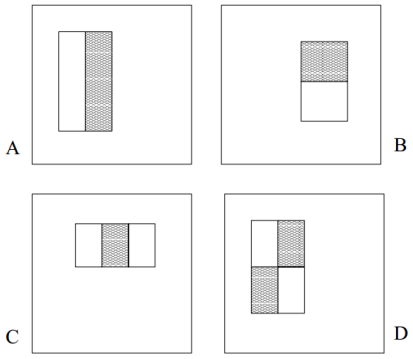

Figura 1.4: Ejemplo de rasgos de dos, tres y cuatro rectángulos en posición relativa a la ventana de búsqueda. La suma de los píxeles de las áreas blancas se resta de la suma de los píxeles de las áreas sombreadas.

Imagen integral

La suma de los píxeles de un rectángulo puede ser calculada de manera muy eficiente empleando una representación intermedia denominada imagen integral. La imagen integral en el punto (x, y) contiene la suma de todos los píxeles que están arriba y hacia la izquierda de ese punto en la imagen original:

$$ii(x,y) = \sum_{x' \le x,\ y' \le y} i(x',y') \tag{1.1}$$

donde $ii(x,y)$ es la imagen integral y $i(x,y)$ es la imagen original (ver Figura 1.5 (Viola and Jones, 2001)).

La imagen integral puede calcularse en un solo barrido de la imagen original empleando las siguientes recurrencias:

$$s(x,y) = s(x,y-1) + i(x,y), \tag{1.2}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{1.3}$$

donde $s(x,y)$ es la suma acumulada de la fila x, con $s(x,-1) = 0$ y $ii(-1,y) = 0$.

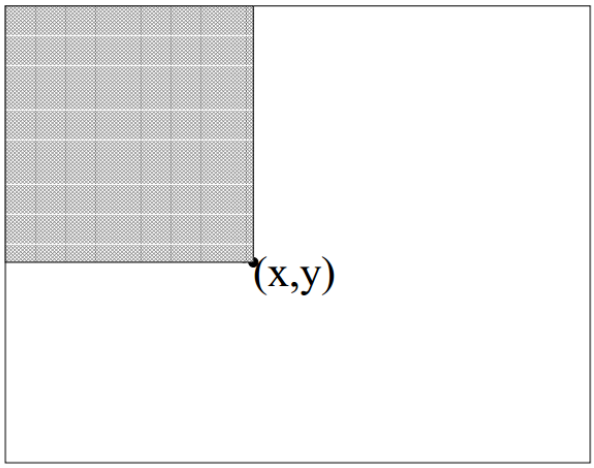

Figura 1.5: La imagen integral en el punto (x, y) es la suma de todos los píxeles arriba y a la izquierda.

Con la imagen integral, cualquier suma rectangular puede calcularse con cuatro referencias a memoria como se muestra en la Figura 1.6 (Viola and Jones, 2001). Los rasgos de dos rectángulos se pueden computar con 6 referencias a memoria puesto que comparten vértices. En el caso de rasgos de tres rectángulos con 8, y con 9 para características de cuatro rectángulos.

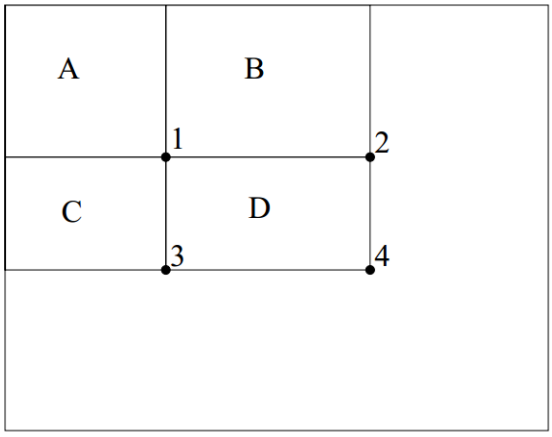

Figura 1.6: Ejemplo del cálculo de la suma de píxeles en un rectángulo. El valor de la suma de los píxeles en el rectángulo D es igual al valor de la imagen integral en 4 (A+B+C+D) menos el valor de la imagen integral en 2 y en 3 (A+B, A+C) y más el valor de la imagen integral en 1 (A).

Proceso de aprendizaje

El proceso de entrenamiento de los clasificadores se realiza mediante un algoritmo basado en AdaBoost, un meta-algoritmo adaptativo de aprendizaje de máquinas cuyo nombre es una abreviatura para *adaptative boosting*.

Este meta-algoritmo consiste en tomar una serie de clasificadores débiles y combinarlos para construir un clasificador fuerte con la precisión deseada. AdaBoost fue introducido por (Freund and Schapire, 1997) y resuelve muchas de las dificultades asociadas al proceso de aprendizaje.

En el procedimiento de Viola-Jones, AdaBoost se utiliza tanto para seleccionar un pequeño grupo de rasgos de los posibles como para entrenar el clasificador.

Para seleccionar rasgos, se entrenan clasificadores débiles limitados a usar un único rasgo. Para cada rasgo, el clasificador débil determina el valor umbral que minimiza los ejemplos mal clasificados. Un clasificador débil $h_j(x)$ por tanto consiste en un rasgo f_j, un valor umbral θ_j y un coeficiente p_j indicando la dirección del signo de desigualdad.

$$h_j(x) = \begin{cases} 1, & si\ p_j f_j(x) < p_j \theta_j \\ 0, & e.o.c \end{cases}$$

A continuación, se describe el algoritmo AdaBoost empleado en Viola-Jones. En cada ronda se selecciona un clasificador débil, como cada clasificador está limitado a usar un único rasgo, el procedimiento selecciona también un rasgo.

- Se parte de un conjunto de imágenes $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_i = 0, 1$ para ejemplos negativos y positivos respectivamente.
- Se inicializan los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente donde m es el número de ejemplos negativos y l el número de positivos.
- Para cada ronda, $t = 1, \dots, T$:
    1. Normalizar los pesos:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,i}}$$

    2. Para cada rasgo, j, entrenar un clasificador h_j que solo use un rasgo. El error se evalúa teniendo en cuenta los pesos w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

3. Se escoge el clasificador, h_t, con menor error ϵ_j.
4. Se actualizan los pesos:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

Donde $e_i = 0$ si el ejemplo x_i se clasifica correctamente y 1 en caso contrario y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

- El clasificador fuerte final es:

$$h(x) = \begin{cases} 1, & si \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0, & e.o.c \end{cases}$$

Donde $\alpha_t = log\frac{1}{\beta_t}$

Cascada atencional

En vez de construir un único clasificador mediante el proceso descrito en el apartado anterior, se pueden construir clasificadores más pequeños y eficientes que rechacen muchas ventanas negativas (es decir, aquellas que no incluyan ninguna instancia del objeto buscado) manteniendo casi todas las positivas (es decir, las que contienen una instancia del objeto buscado). Estos clasificadores más simples se utilizan para rechazar la mayoría de las ventanas de búsqueda y solo en aquellas en las que hay mayores probabilidades de encontrar caras se llama a clasificadores más complejos que disminuyan el número de falsos positivos. Este proceso se representa en la Figura 1.7 (Klette, 2014).

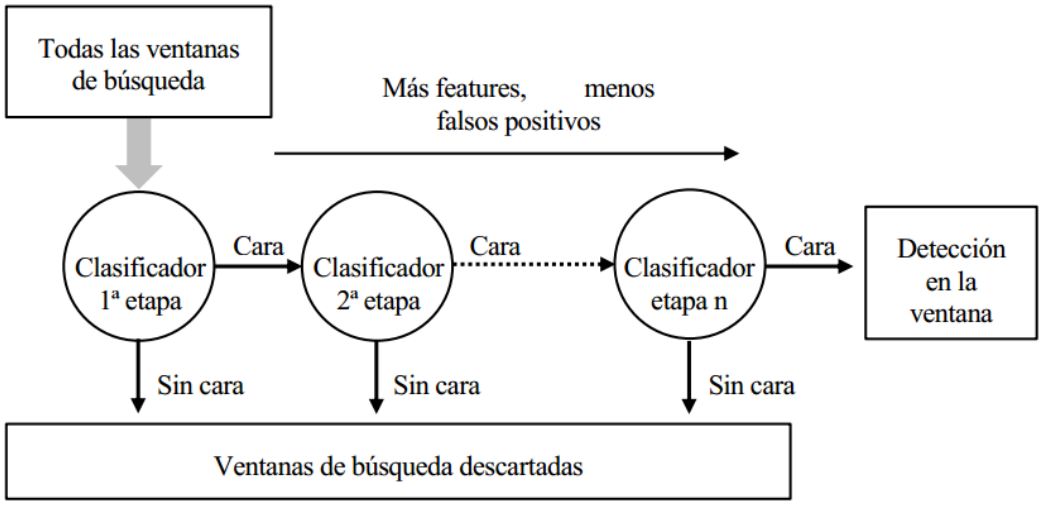

Figura 1.7: Descripción esquemática de una cascada atencional. Una serie de clasificadores se aplican a cada ventana de búsqueda. Los clasificadores iniciales rechazan un gran número de ventanas rápidamente.

Proceso de detección

Las imágenes usadas para entrenar al algoritmo fueron normalizadas para minimizar los efectos de diferentes condiciones de iluminación, por tanto, también resulta necesario realizar la normalización en el proceso de detección. Para ello, en vez de normalizar la imagen antes de comenzar el análisis, lo cual implicaría cambiar el valor de todos los píxeles, resulta más sencillo corregir los valores de los rasgos conforme se van calculando.

Para normalizar se emplea la varianza:

$$\sigma^2 = m^2 - \frac{1}{N}\sum x^2 \tag{1.4}$$

Donde m es la media del valor de los píxeles, que puede calcularse a partir de la imagen integral. La suma de los píxeles al cuadrado se puede obtener a partir de una imagen integral de la imagen al cuadrado.

La cascada de clasificadores se evalúa sobre una ventana de búsqueda cuadrada que barre la imagen con incrementos de unos pocos píxeles. La búsqueda se realiza a distintas escalas obtenidas al multiplicar la escala anterior por un factor de escala, normalmente entre 1.1 y 1.3.

Puesto que el detector es poco sensible a pequeñas translaciones y diferencias de escala, pueden producirse múltiples detecciones alrededor de cada cara. Es necesario exigir que las detecciones

tengan un determinado número mínimo de detecciones vecinas para disminuir la cantidad de falsos positivos.

Para combinar las detecciones que se refieren al mismo objeto, se fusionan aquellas cuyas áreas se solapen en más de un determinado valor umbral y el recuadro con la detección final se calcula como la media de todos los recuadros que se han fusionado.

El algoritmo Viola-Jones presenta una selección eficiente de rasgos, es invariante a la escala y a la localización, utiliza filtros Haar en lugar de puntos de concordancia. Por otro lado, es sensible a los cambios de iluminación y la cascada deslizante puede obtener varias detecciones del mismo rostro.

1.7.2 Patrones Binarios Locales

El método Patrones Binarios Locales (LBP del inglés Local Binary Patterns) fue introducido en 2002 por (Ojala *et al.*, 1996) y utilizado en general para detectar todo tipo de objetos basado en texturas. Se trata de un sistema capaz de extraer la estructura local en una imagen a partir del entorno de cada uno de los píxeles que la componen.

El proceso consiste en ir comparando el valor de cada píxel (considerado como píxel central) con el de sus vecinos distribuidos en una circunferencia a su alrededor. La comparación que se hace responde a la ecuación:

$$s(x) = \begin{cases} 1 \; if (x \geq x_c) \\ 0 \; else \end{cases} \tag{1.5}$$

Siendo x_c el valor central, y x el valor a evaluar.

De tal manera que, si el valor del píxel vecino es mayor que el central entonces el valor será 1, mientras que si es menor será 0, como puede observarse en la Figura 1.8 (Local Binary Patterns, 2018).

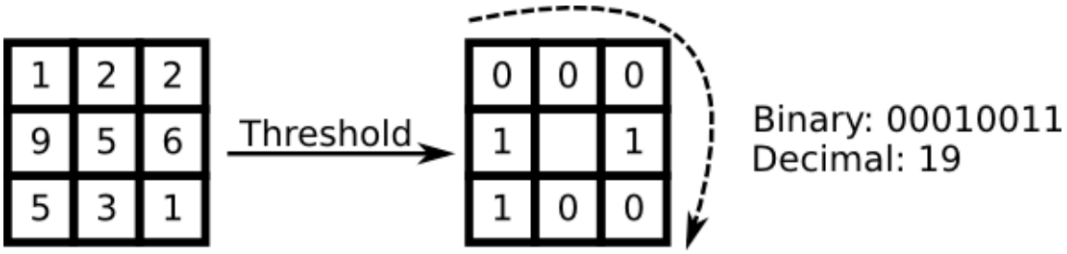

Figura 1.8: Patrón Binario para un píxel.

De este modo, se asocia un valor binario a cada píxel formando un número binario. Si hay 8 píxeles vecinos se obtendrá un número de 8 dígitos, obteniendo así combinaciones, llamadas LBP, y obtenidas matemáticamente a partir de la expresión de la ecuación:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \tag{1.6}$$

Donde i_c hace referencia a la intensidad del punto central y i_p a la intensidad de cada vecino.

Luego de sustituir cada píxel por su correspondiente LBP obtenemos una imagen bien detallada como se muestra en la Figura 1.9 (Local Binary Patterns with Python & OpenCV - PyImageSearch, 2018).

Figura 1.9: Aplicación del LBP en una imagen.

Finalmente se calcula el histograma de la imagen resultante, siendo este un excelente vector de características para utilizar en un clasificador, por ejemplo, una máquina de soporte vectorial.

(Varsha Gupta and Dipesh, 2014) comparan estos dos métodos comentados hasta ahora para detectar caras explicando las ventajas e inconvenientes de cada uno. El algoritmo de Viola-Jones es vanguardia en cuanto a precisión y sobre todo a la razón de precisión/velocidad. En cambio, aunque el algoritmo de LBP es menos preciso, es más rápido e invariante frente a cambios de iluminación.

Este algoritmo posee alto poder discriminativo, simplicidad en el código, invariancia ante los cambios de iluminación y buen desempeño computacional. Por otro lado, se ve afectado por rotaciones, al utilizar un mayor número de píxeles vecinos aumenta su costo computacional, la información estructural que captura es limitada y la información de magnitud es ignorada.

1.7.3 Pirámide de histogramas de gradientes orientados (PHOG).

PHOG fue propuesto por (Bosch, Zisserman and Munoz, 2007) y se ha utilizado de manera eficiente en la clasificación de objetos. En PHOG, una imagen es descompuesta en una secuencia de subregiones cada vez más pequeñas denominadas células en varios niveles de la pirámide. En cada nivel las células tienen diferentes resoluciones de cuadrícula. Entonces se combinan los histogramas de gradientes orientados (HOG) de cada célula para lograr una representación de objetos con mayor discriminación local que de forma global. Este resultado es utilizado como vector de características para los clasificadores y se denomina descriptor.

La técnica para PHOG está principalmente basada en los histogramas de gradientes orientados (HOG) de (Dalal and Triggs, 2005). En (Grauman and Darrell, 2005) se introdujo la pirámide espacial al representar cada imagen de manera eficiente subdividiéndolas en una secuencia de cuadrículas cada vez más pequeñas duplicando repetidamente el número de divisiones en cada subregión en diferentes niveles.

(Dalal and Triggs, 2005) utilizaron HOG dividiendo la imagen en bloques, entonces cada bloque está organizado en celdas al combinar los píxeles vecinos. En consecuencia, para cada bloque se calcula un histograma, el cual se normaliza para compensar variaciones en la iluminación.

(Bosch, Zisserman and Munoz, 2007) utilizaron PHOG para representar una imagen por su forma local y la información espacial de la forma. En este caso, la imagen se divide en una secuencia de subregiones de diferentes resoluciones duplicando repetidamente la división del área de interés en cada nivel de la pirámide. Para cada subregión se calcula un vector HOG. La combinación de estos vectores de características HOG representa el descriptor final de PHOG. Aquí la forma local se describe por las ocurrencias de orientaciones de borde en las celdas en cada nivel. Por lo tanto, PHOG representa tanto la dirección del borde como la ubicación.

Imagen piramidal

Generalmente las imágenes procesadas tienen tamaño constante. Pero en ciertas ocasiones es necesario procesar la misma imagen en diferentes resoluciones. Por ejemplo, en el caso particular de este trabajo, al buscar un rostro, no se conoce de antemano el tamaño que tendrá este.

Para ello un grupo de imágenes con diferentes resoluciones es creado y el objeto se busca en ellas. Este grupo de imágenes de diferentes resoluciones se conoce como imagen piramidal, debido a que, si son puestas en una pila, con la imagen de mayor resolución en el fondo de la pila y la imagen con menor resolución en el tope, se asemeja a una pirámide (Figura 1.10).

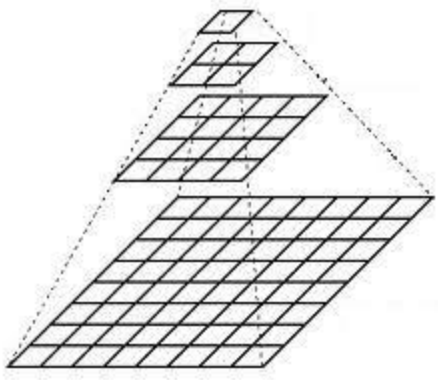

Figura 1.10: Diferentes niveles en una imagen piramidal.

Existen dos tipos de imágenes piramidales: la gaussiana y la laplaciana.

En las imágenes piramidales gaussianas el nivel más alto está formado por la eliminación consecutiva de filas y columnas de la imagen original. Luego cada píxel en un nuevo nivel es formado por la combinación de 5 píxeles del nivel anterior con pesos gaussianos.

Las imágenes piramidales laplacianas son formadas a partir de las diferencias entre los niveles de las gaussianas.

Gradientes orientados

Los gradientes orientados de una imagen se obtienen a partir de la convolución de esta con un kernel de Sobel.

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Figura 1.11: Kernels de Sobel para detección de bordes verticales y horizontales.

El tamaño del kernel y la distribución de los valores dependen del grosor de los bordes y la rotación del objeto respectivamente.

La dirección de los gradientes indica hacia dónde el cambio de intensidad es mayor y la magnitud, cuán abrupto es.

En una imagen RGB se calcula el gradiente en cada canal y se toma el de mayor magnitud para cada píxel.

La Figura 1.12 (*Gil's CV blog*, 2013) muestra una subdivisión de 16x16 de una imagen dividida en 16 subregiones a las cuales se les calculan los gradientes orientados.

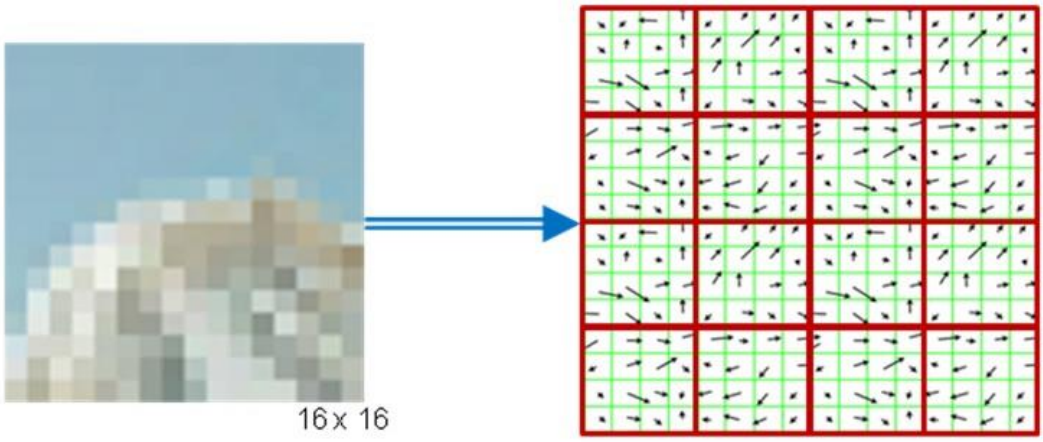

Figura 1.12: Cálculo de los gradientes en una imagen 16x16.

Luego el histograma de cada subdivisión asocia cada gradiente por su orientación Figura 1.12.

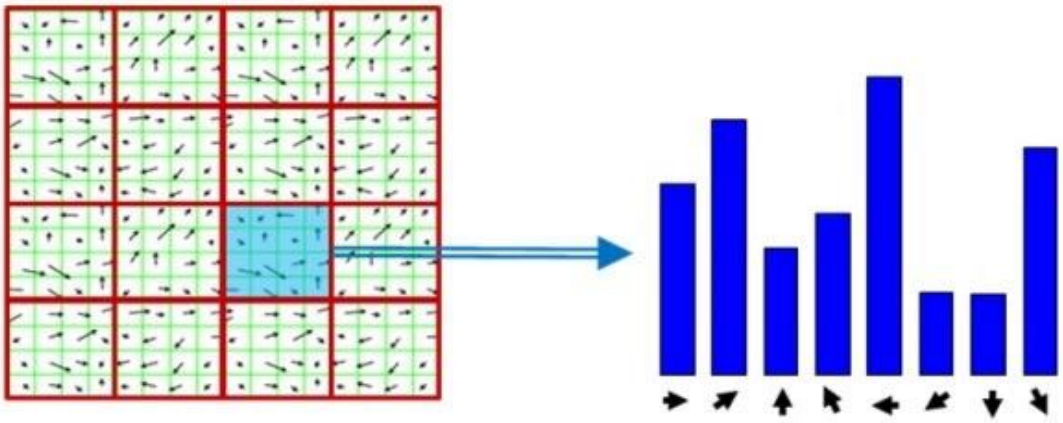

Figura 1.13: Histograma de la orientación de los gradientes para le imagen de la Figura 1.12.

El desempeño computacional y la efectividad de la detección de este algoritmo es similar a Viola-Jones, pero un incremento de las características extraídas de la imagen no reduce el número de falsos positivos.

1.7.4 Comparación

Según la bibliografía consultada el método LBP debido a sus características es utilizado mayormente en sistemas embebidos, siendo PHOG y Viola-Jones los dos algoritmos con mejor razón de detección de rostros manteniendo un costo computacional bajo. Tras comparar ambos algoritmos con corridas sobre el conjunto de datos *Faces in the Wild* la cual cuenta con 5171 rostros se obtuvieron los resultados mostrados en la Tabla 1.1.

Algoritmo	Parámetros	Correctos	Falsos positivos	Falsos Negativos	Tiempo(Min)
Viola-Jones	Cuatro conjuntos de características	3564(68.92%)	128(2.48%)	1607(31.08)	14.6
	Un conjunto de características	3497(67.63%)	132(2.55%)	1672(32.37%)	3.1
PHOG	Imagen remuestreada 4 veces	4059(78.5%)	155(3.0%)	1112(21.5%)	303.1
	Imagen remuestreada 1 vez	3974(76.85%)	19(0.37%)	1197(23.15%)	5.1
	Imagen no remuestreada	3698(71.51%)	4(0.08%)	1473(28.49%)	1.4

Tabla 1.1: Desempeños de los algoritmos Viola-Jones y PHOG para el conjunto de datos *Faces in the Wild*.

El número entre paréntesis, a la derecha de los valores, representa el por ciento con respecto a la cantidad de rostros reales.

Al utilizar un solo conjunto de características en el Viola-Jones, aunque se redujo un poco la efectividad, el tiempo de ejecución disminuyó significativamente respecto a la prueba con cuatro conjuntos de características.

En el PHOG, al remuestrear las imágenes, el tiempo de ejecución aumentó sin un significativo incremento en la cantidad de detecciones correctas y aumentando la cantidad de falsos positivos.

Teniendo en cuenta la cantidad de aciertos, falsos positivos y falsos negativos, así como el tiempo de ejecución se determinó utilizar como algoritmo para la detección de rostros el PHOG remuestreando la imagen una vez.

1.8 Extracción de rasgos

En esta etapa, la imagen facial es procesada por un algoritmo que genera a partir de esta un vector de características o rasgos el cual describe, precisamente, las características de la imagen. Usar una representación como esta provee un modo flexible de agrupar las imágenes en clases. Las clases deben tener una distancia máxima entre ellas y las imágenes dentro de las clases una distancia mínima, por tanto, el método que genere el vector de características que más discrimine entre las imágenes será el de mejores resultados en el proceso de clasificación. Desafortunadamente resulta imposible determinar cuáles son estas características, por tanto, los métodos de extracción de rasgos utilizan diferentes enfoques para construir el vector. También las técnicas de clasificación son sensibles a los tipos de rasgos extraídos por lo que constituyen otro factor importante a tener en cuenta en esta etapa.

Los métodos de extracción de rasgos en imágenes de rostros para la detección automática de emociones, varían tanto en las características que extraen como en el procedimiento utilizado para extraerlos.

1.8.1 Auto Correlación Local de Mayor Orden

En (Lajevardi and Hussain, 2009) uno de los métodos descritos extrae el conjunto de características mediante la Auto Correlación Local de Mayor Orden (HLAC por sus siglas en inglés). Las funciones de autocorrelación de n-ésimo orden son definidas como:

$$x(a_1, a_2, \dots, a_N) = \int f(r) f(r + a_1) \dots f(r + a_N) dr \tag{1.7}$$

donde $f(r)$ denota la intensidad en el píxel de observación r, y $a_1, a_2, \dots, a_N$ representan N desplazamientos. El orden y el desplazamiento son arbitrarios. La imagen se barre con un conjunto de 25 máscaras, como se muestran en la Figura 1.14 (Lajevardi and Hussain, 2009).

Es calculado el producto de los píxeles correspondientes a los espacios en blanco de la máscara. Luego, todos los productos resultantes de una máscara son sumados para obtener un elemento del vector de características.

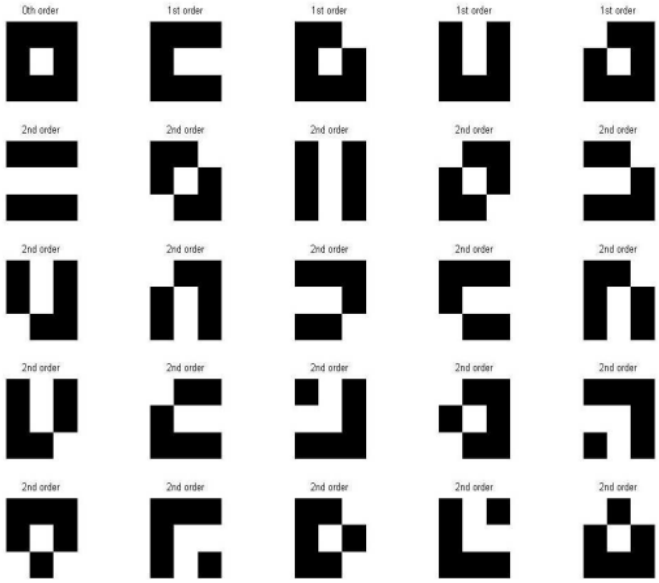

Figura 1.14: Máscaras de tamaño 3x3 para formar los rasgos HLAC.

1.8.2 Local Minima

En (Mohammad and Surapong, 2013), el método propuesto, basado en LBP, calcula la característica local de un píxel a partir de la intensidad de sus píxeles vecinos. Como se muestra en la Figura 1.15 (Mohammad and Surapong, 2013), en un patrón de tamaño 3x3, el píxel central está rodeado de 8 píxeles vecinos en 8 direcciones posibles. Las direcciones se denotan como: 0º, 45º, 90º, 135º, 180º, 225º, 270º y 315º.

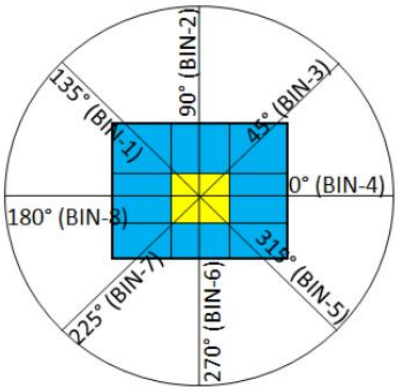

Figura 1.15: Las 8 direcciones posibles denotadas como: 0º, 45º, 90º, 135º, 180º, 225º, 270º y 315º.

La característica local de cada píxel es la dirección donde se encuentre el píxel de menor intensidad. Como variantes del método pueden ser escogidas las direcciones donde se encuentre el píxel de segundo, tercer y así sucesivamente, valor de intensidad. La imagen es dividida en bloques en los cuales se computa un histograma de acuerdo a las características locales de cada píxel perteneciente a la región. Cada histograma tendrá solo 8 barras correspondientes a las 8

direcciones respectivamente. Los histogramas calculados son concatenados para formar el vector de características de la imagen analizada.

1.8.3 Marcas Faciales o Puntos de Referencia Facial

El algoritmo expuesto en (Kazemi and Sullivan, 2014) encuentra la posición de los puntos de referencia faciales de forma precisa y computacionalmente eficiente.

A continuación, se exponen los principales elementos que conforman este algoritmo.

Cascada de regresores

Sea $x_i \in \mathbb{R}^2$ las coordenadas del i-ésimo punto de referencia facial en la imagen I. Entonces el vector $S = (x_1^T, x_2^T, \dots, x_3^T)^T \in \mathbb{R}^{2p}$, denota las coordenadas de los puntos de referencia en I. $\hat{S}^{(t)}$ representa la estimación actual de S. Cada regresor r_t en la cascada predice una actualización a partir de la imagen y $\hat{S}^{(t)}$ que es añadida a $\hat{S}$(t) para mejorar esta estimación.

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t(I, \hat{S}^{(t)}) \tag{1.8}$$

El punto crítico de la cascada es que el regresor r_t hace su predicción basado en características, tales como los valores de intensidad de los píxeles, calculados a partir de I, e indexados relativos a la estimación actual de S. Esto agrega invariancia geométrica al proceso.

Aprendizaje de cada regresor en la cascada.

Sean $(I_1, S_1), \dots, (I_n, S_n)$ el conjunto de entrenamiento donde cada I_i representa la imagen de un rostro con sus correspondientes puntos de referencia S_i.

Para el aprendizaje de la primera función de regresión r0 en la cascada se crea de los datos de entrenamiento la tripleta de una imagen de un rostro (I), la estimación inicial del vector S ($\hat{S}_i^{(0)}$) y el paso de actualización objetivo ($\Delta\hat{S}_i^{(0)} = S_i - \hat{S}_i^{(0)}$).

El número de tripletas depende de la cantidad de aproximaciones iniciales del vector S para imagen, las cuales son seleccionadas uniformemente del conjunto de vectores S de las demás imágenes.

El conjunto de tripletas entonces es actualizado y dado como datos de entrenamiento al siguiente regresor, proceso que continúa hasta obtener el nivel de efectividad deseado.

El árbol de regresión en cada nodo se decide basado en el umbralado de la diferencia de intensidad entre los píxeles.

Los puntos de referencia faciales extraídos por este algoritmo se muestran en la Figura 1.16 (Facial landmarks with dlib, OpenCV, and Python - PyImageSearch, 2017).

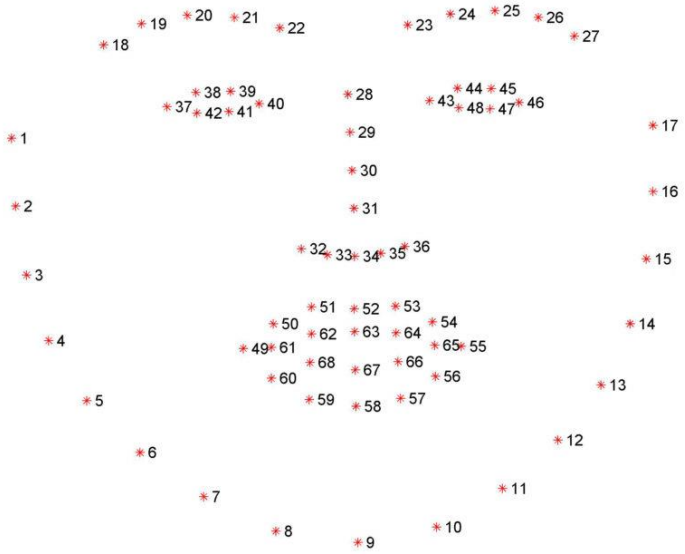

Figura 1.16: Puntos de referencia facial.

Estos 68 puntos de referencia no solo describen la posición de los ojos, nariz y boca sino además la forma de estos.

La extracción de características es extremadamente importante en el proceso de clasificación. Si no son extraídas las características adecuadas, incluso el mejor clasificador no podría lograr un reconocimiento con precisión.

1.9 Técnicas de clasificación utilizadas en el proceso de detección de emociones

En esta etapa el vector de características generado por un algoritmo de extracción de rasgos, es utilizado para entrenar un clasificador.

A continuación, se exponen varias técnicas de clasificación utilizadas en el proceso de detección de emociones. Máquinas de Vectores Soporte

Las *máquinas de vectores soporte* o *máquinas de soporte vectorial* (SVM, del inglés Support Vector Machine) son un conjunto de algoritmos de aprendizaje automático desarrollados por Vladimir Vapnik (Vapnik, 1979). Estos algoritmos forman parte de la familia de clasificadores lineales. Aunque originariamente las SVMs fueron pensadas para resolver problemas de

clasificación binaria, actualmente se utilizan para resolver otros tipos de problemas (regresión, agrupamiento, multiclasificación).

La idea es seleccionar un hiperplano de separación que equidiste de los ejemplos más cercanos de cada clase para, de esta forma, calcular lo que se denomina un margen máximo a cada lado del hiperplano. En el momento de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de *vectores soporte*.

La SVM es una generalización de un clasificador muy simple e intuitivo llamado *clasificador de margen máximo*. Este sencillo clasificador no puede aplicarse en muchos conjuntos de datos pues requiere que las clases permitan ser separadas por un límite lineal. Una extensión de este que puede aplicarse a un conjunto más amplio de casos es el *clasificador de vectores soporte*. La *máquina de vectores soporte* es una extensión del clasificador de vectores soporte con el objetivo de establecer entre las clases un límite no lineal. Estos tres clasificadores en conjunto se conocen popularmente como las máquinas de vectores soporte.

Clasificador de Margen Máximo

En un espacio p-dimensional, un *hiperplano* es un subespacio plano de p de dimensión $p-1$. Por ejemplo, en un espacio de dos dimensiones, un hiperplano es un subespacio plano unidimensional, en otras palabras, una línea. En tres dimensiones, un hiperplano es un subespacio plano bidimensional lo que se traduce a un plano. En $p > 3$ dimensiones un hiperplano es difícil de visualizar, pero la noción de un subespacio plano de $p-1$ dimensiones se sigue aplicando.

Un hiperplano de dos dimensiones es definido por la siguiente ecuación:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \tag{1.9}$$

lo cual indica que cualquier $X = (X_1, X_2)^T$ que satisface (1.9), representa un punto en el hiperplano. Esta ecuación representa una línea por lo que puede afirmarse que en dos dimensiones un hiperplano es una línea.

La ecuación (1.9) puede ser extendida para el caso p-dimensional:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{1.10}$$

nuevamente, si un punto $X = (X_1, X_2, \dots, X_p)^T$ en el espacio p-dimensional satisface la ecuación entonces X pertenece al hiperplano.

Si X satisface en lugar de (1.10), la siguiente:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0, \tag{1.11}$$

entonces X se encuentra a un lado del hiperplano, si por el contrario

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0, \tag{1.12}$$

entonces X se encuentra en el otro lado del hiperplano. Por lo tanto un hiperplano no es más que una división de un espacio p-dimensional a la mitad. Calculando el signo del miembro derecho de (1.10) se puede determinar el lado del hiperplano en el cual se encuentra un punto dado.

Suponiendo una matriz X de dimensiones $n \times p$ que consiste en n observaciones de entrenamiento en un espacio p-dimensional,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}, \tag{1.13}$$

y cada observación puede clasificarse en una de dos clases: $y_1, \dots, y_n \in \{-1, 1\}$ donde -1 representa una clase y 1 la otra clase. También se tiene un p-vector de prueba con rasgos observados $x^* = (x_1^* \ \dots \ x_p^*)^T$. El objetivo es crear un clasificador basado en los datos del conjunto de entrenamiento que clasifique correctamente el vector de prueba.

Suponiendo que es posible encontrar un hiperplano que separe perfectamente las observaciones de prueba de acuerdo a sus clases entonces un *hiperplano separador* (ver Figura 1.17) cumple con la siguiente propiedad:

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1, \tag{1.14}$$

y

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1. \tag{1.15}$$

Equivalente a la propiedad anterior, un hiperplano separador cumple que:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0 \tag{1.16}$$

para toda $i = 1, \ldots, n$.

Si existe un hiperplano separador, puede construirse con este un clasificador bastante simple: a cada observación se le asigna una clase en dependencia del lado del hiperplano en que se encuentre. En el panel derecho de la Figura 1.17 se muestra un clasificador como este. Entonces la clasificación de un caso de prueba x^* se obtiene a partir del signo de $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$. Si $f(x^*)$ es positiva el caso de prueba pertenece a la clase 1 y si $f(x^*)$ es negativa el caso de prueba pertenece a la clase -1. La magnitud de $f(x^*)$ indica la certeza de la clasificación de un caso de prueba. Si el valor de $f(x^*)$ se encuentra lejano a cero, significa que x^* yace lejos del hiperplano y por tanto se puede tener certeza de la clasificación otorgada a x^*. Por otro lado, si el valor de $f(x^*)$ está cercano a cero, entonces x^* se localiza cerca del hiperplano y por tanto no es muy alta la certeza de la clasificación otorgada a x^*.

En términos generales, si los datos pueden ser perfectamente separados utilizando un hiperplano, en efecto, existirán infinitos hiperplanos que separen dicho conjunto de datos. Esto se debe a que un hiperplano separador dado puede ser trasladado o rotado tanto como se quiera en pequeñas proporciones, sin entrar en contacto con ninguna de las observaciones. En el panel izquierdo de la Figura 1.17 (James et al., 2000) se muestran tres posibles hiperplanos separadores de un conjunto de datos. En orden de construir un clasificador basado en un hiperplano separador, se requiere de un método capaz de decidir por un solo hiperplano de los infinitos posibles.

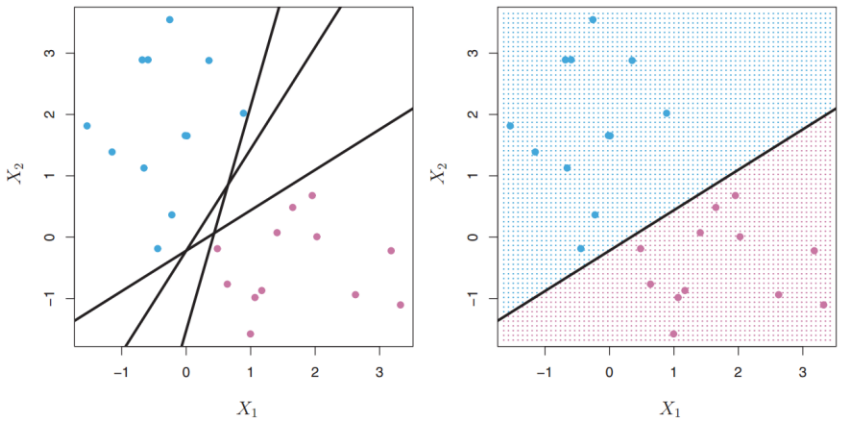

Figura 1.17: En el panel izquierdo, se muestran observaciones de dos clases, en azul y rojo respectivamente. Tres hiperplanos separadores de los infinitos posibles se muestran en negro. En el panel derecho, se muestra un hiperplano separador en negro. Las regiones en azul y en rojo representan la regla de decisión del clasificador basado en este hiperplano.

Una elección natural es el *hiperplano de margen máximo*, también conocido como *hiperplano separador óptimo*. Este es el hiperplano separador que más lejos se encuentra de las observaciones de entrenamiento. La menor distancia entre los puntos de observación y el hiperplano separador se conoce como *margen*. El hiperplano de margen máximo es el hiperplano separador para el cual su margen es el mayor. El clasificador basado en el hiperplano de margen máximo se denomina como *clasificador de margen máximo*. Teniendo en cuenta lo descrito anteriormente, se espera, que un clasificador de margen máximo para el conjunto de entrenamiento, también tenga margen máximo para el conjunto de prueba, de esta forma todas las observaciones de prueba serán clasificadas correctamente.

Si $\beta_0, \beta_1, \dots, \beta_p$ son los coeficientes del hiperplano de margen máximo, entonces el clasificador de margen máximo, clasifica la observación de prueba x^* basado en el signo de $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$.

La Figura 1.18 (James et al., 2000) muestra el hiperplano de margen máximo para el conjunto de datos mostrados en la Figura 1.17. En esta, pueden observarse tres puntos del conjunto equidistantes al hiperplano de margen máximo y yacen en las líneas discontinuas que indican el ancho del margen. Estas observaciones se conocen como *vectores soporte*. Si estos vectores son trasladados en el espacio, el hiperplano de margen máximo también lo será, por tanto, este depende directamente de los

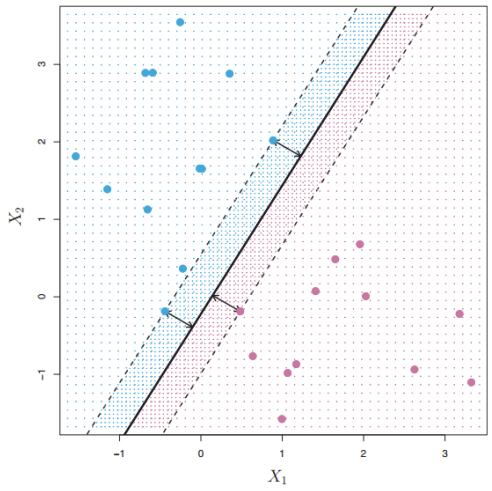

Figura 1.18: Hiperplano de margen máximo para los datos de la Figura 1.17.

vectores soporte. Sin embargo, un cambio en las restantes observaciones, siempre que estas no crucen el límite impuesto por el margen, no influencia cambios en el hiperplano de margen máximo.

Construcción del clasificador de margen máximo

La tarea de construir el hiperplano de margen máximo para un conjunto de n observaciones de entrenamiento $x_1, \dots, x_n \in \mathbb{R}^p$ y sus correspondientes etiquetas de clase $y_1, \dots, y_n \in \{-1, 1\}$ se resume en la resolución del siguiente problema de optimización:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximizar}} \; M \tag{1.17}$$

$$\text{sujeto a} \sum_{j=1}^{p} \beta_j^2 = 1, \tag{1.18}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M \;\; \forall\, i = 1, \dots, n. \tag{1.19}$$

La restricción (1.19) garantiza la ubicación de cada observación en el lado correcto del hiperplano con cierto margen M positivo. Nótese que la restricción (1.18) no es realmente una restricción al hiperplano; si $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} = 0$ define un hiperplano de margen máximo, entonces $k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) = 0$ también lo define para cualquier $k \neq 0$. Con la restricción (1.18), la distancia perpendicular de la i-ésima observación al hiperplano se obtiene de $y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right)$. Entonces las restricciones (1.18) y (1.19) aseguran que cada observación se encuentre en el lado correcto del hiperplano y a una distancia de al menos M unidades del mismo.

En la mayoría de los casos no existe hiperplano separador y a consecuencia no puede construirse el clasificador de margen máximo. En estos casos, el problema de optimización (1.17)–(1.19) no tiene solución para $M > 0$. En la Figura 1.19 (James et al., 2000) se muestra un ejemplo para el cual las dos clases no pueden ser exactamente separadas. Se puede extender el concepto de hiperplano separador en orden de hallar un hiperplano que *casi* separe las clases a través del *margen suave*.

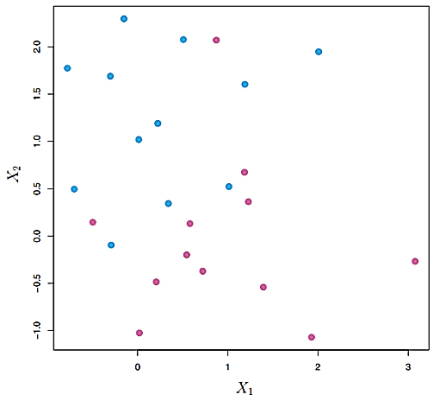

Figura 1.19: En este caso las clases no pueden ser separadas por un hiperplano y por tanto el clasificador de margen máximo no puede ser utilizado.

La generalización del clasificador de margen máximo para el caso en que las clases no son linealmente separables se conoce como *clasificador de vectores soporte*.

Clasificador de vectores soporte

Como se muestra en la Figura 1.19, las observaciones pertenecientes a dos clases no son necesariamente separables por un hiperplano. También, puede ocurrir el caso en el que exista un hiperplano separador y no sea esta la mejor solución. Un clasificador basado en un hiperplano separador clasificará perfectamente todos los casos de entrenamiento, lo que hace al clasificador sensible a casos extremos, un ejemplo se muestra en la Figura 1.20 (James et al., 2000). En el panel derecho de la Figura 1.20 se puede observar como al añadir una observación al entrenamiento puede ocasionar un cambio drástico en el hiperplano de margen máximo. El hecho de que el hiperplano de margen máximo sea extremadamente sensible a un único cambio sugiere que puede sobreajustarse a los datos de entrenamiento.

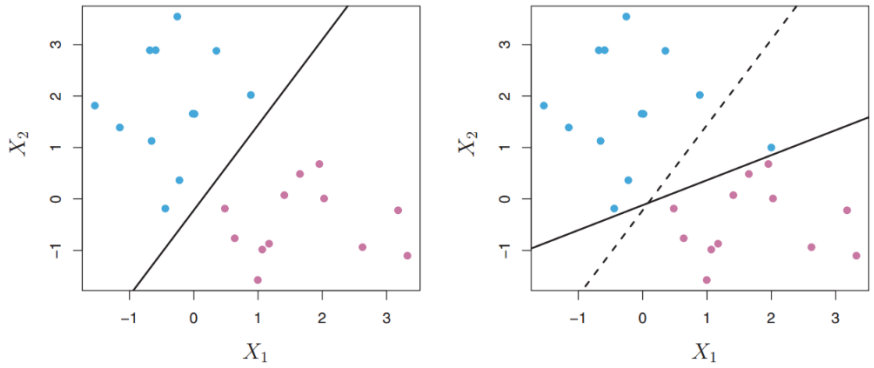

Figura 1.20: En el panel izquierdo se representan dos clases separadas por el hiperplano de margen máximo. En el panel derecho puede observarse un gran cambio del hiperplano al añadir una nueva observación.

El *clasificador de vectores soporte* o *clasificador de margen suave*, clasifica incorrectamente un número limitado de las observaciones de entrenamiento con el objetivo de realizar una mejor clasificación del resto de las observaciones. En lugar de buscar el mayor margen posible para el cual cada observación se encuentre en lado correcto del margen y del hiperplano, este clasificador, permite algunas observaciones en el lado incorrecto del margen e incluso del hiperplano. En la Figura 1.21 (James et al., 2000) se muestra un ejemplo; en el panel izquierdo puede observarse un subconjunto de observaciones que se encuentran en el lado incorrecto del margen, en el panel derecho se muestra un subconjunto de observaciones en el lado incorrecto del hiperplano.

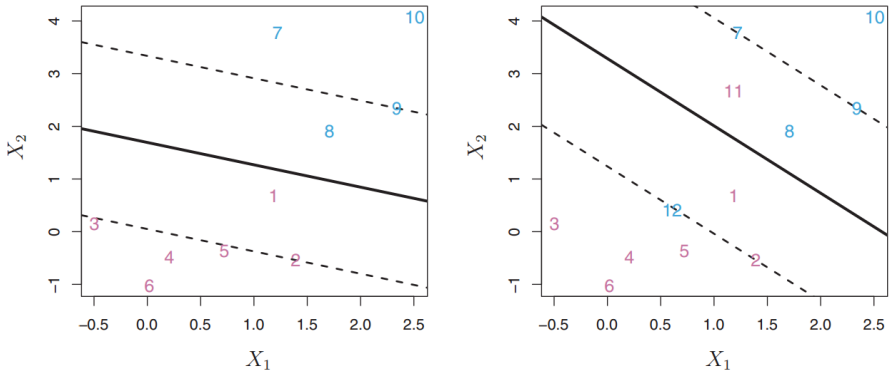

Figura 1.21: Resultado de aplicar un clasificador de vectores soporte en un conjunto de observaciones. La línea sólida muestra el hiperplano y las líneas discontinuas el margen. En el panel izquierdo se observan los casos 9 y 2 que se encuentran sobre el margen y los casos 1 y 8 en el lado incorrecto del margen. En el panel derecho se observan los casos 11 y 12 en el lado incorrecto del hiperplano.

Construcción del clasificador de vectores soporte

La tarea es encontrar el hiperplano que separe correctamente la mayoría de las observaciones de entrenamiento permitiendo dejar algunas en el lado incorrecto, esta es, la solución del siguiente problema de optimización:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximizar } M} \tag{1.20}$$

$$\text{sujeto a } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{1.21}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1-\epsilon_i), \tag{1.22}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C. \tag{1.23}$$

El anterior problema de optimización cumple con la siguiente propiedad: solo las observaciones que yacen sobre el margen o del lado incorrecto del margen afectan el hiperplano. Estas observaciones son los vectores soportes.

Cuando en el problema a enfrentar, las clases no pueden separarse por un límite lineal, el clasificador de vectores soporte arroja resultados muy pobres. Un ejemplo se muestra en la Figura 1.22 (James et al., 2000). Una posible solución para estos problemas es agrandar el espacio de características de manera tal que en el espacio transformado pueda hallarse una frontera lineal para separar las clases.

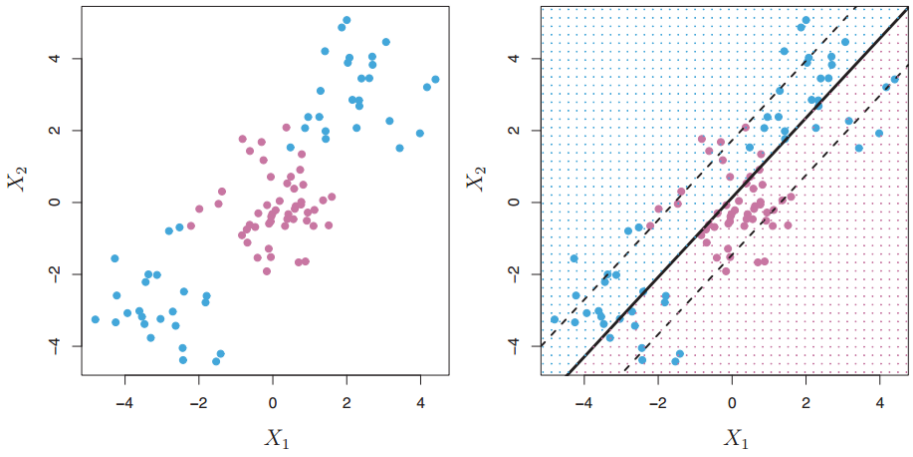

Figura 1.22: En el panel izquierdo se observan dos clases entre una frontera no lineal entre ellas. En el panel derecho el clasificador de vectores soporte busca una frontera lineal entre las clases y por tanto su desempeño es muy pobre.

Máquina de vectores soporte

La *máquina de vectores soporte* (SVM) es una extensión del clasificador de vectores soporte que resulta de agrandar el espacio de características de una manera específica usando los llamados *kernels*.

A continuación, se describe brevemente la SVM:

El algoritmo para la solución del problema de optimización descrito en (1.20)–(1.23) solo necesita los productos escalares de las observaciones de entrenamiento en lugar de las observaciones mismas. El producto escalar entre dos observaciones $x_i,\ x_{i'}$ se define como:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{1.24}$$

Está demostrado que el clasificador de vectores soporte lineal se puede representar como:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x_i, x_{i'} \rangle, \tag{1.25}$$

con n parámetros $\alpha_i,\ i = 1, \dots, n$, uno por cada observación de entrenamiento. Para estimar los coeficientes β_0 y α_i solo se necesita calcular los $\binom{n}{2} = n(n-1)/2$ productos escalares entre todos los pares de observaciones de entrenamiento.

En orden de evaluar la función descrita en (1.25), se necesita calcular el producto escalar entre el nuevo punto x y cada uno de los puntos de entrenamiento x_i. Resulta que si una observación de entrenamiento no es un vector soporte de la solución entonces $\alpha_i = 0$. Por tanto si S es la colección índices de los vectores soporte entonces (1.25) puede reescribirse como:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x_i, x_{i'} \rangle. \tag{1.26}$$

En resumen, para representar el clasificador lineal y para calcular sus coeficientes, solo se necesitan productos escalares.

Al sustituir cada producto escalar, tanto en la representación (1.26) como en el cálculo de la solución para el clasificador de vectores soporte, por una generalización de la forma:

$$K(x_i, x_{i'}), \tag{1.27}$$

donde K es una función la cual se conoce como *kernel*; la función (1.26) toma la forma:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i). \tag{1.28}$$

La siguiente ecuación representa el kernel lineal:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{1.29}$$

Obsérvese que esta es la definición del producto escalar definido en (1.24) y utilizar este kernel equivale al clasificador de vectores soporte.

Un kernel muy conocido es el *kernel polinomial*:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \tag{1.30}$$

donde d es un entero positivo que representa el grado del polinomio. Cuando $d = 1$ la SVM se reduce en el clasificador de vectores soporte.

Otro kernel muy utilizado es el llamado *kernel radial*:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right). \tag{1.31}$$

En la Figura 1.23 (James et al., 2000) se muestran dos SVMs, en el panel izquierdo puede observarse una SVM que utiliza un kernel polinomial de grado 3 y en el panel derecho una SVM que utiliza un kernel radial.

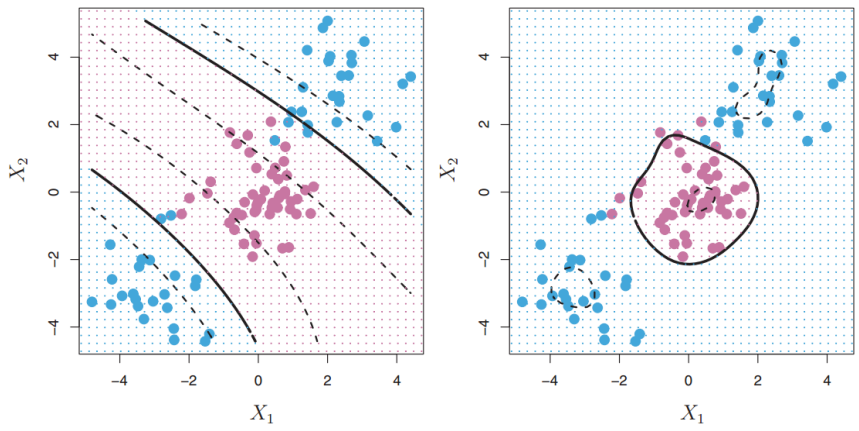

Figura 1.23: En el panel izquierdo una SVM polinomial de grado 3 es aplicada a los datos de la Figura 1.22. En el panel derecho una SVM con kernel radial es aplicada a los mismos datos. Para este ejemplo ambos kernels son capaces de separar las clases satisfactoriamente.

1.9.1 Perceptrón Multicapa

Perceptrón simple

El perceptrón simple (Figura 1.24 (Perceptrón - Wikipedia, la enciclopedia libre, 2015)), es un tipo de neurona artificial que utiliza como función de activación la función de paso Heaviside (Duff, 1966). El término perceptrón fue utilizado por primera vez por Frank Rosemblatt (1958) cuya idea central es la incorporación del aprendizaje al modelo de la neurona de McCulloch y Pitts (McCulloch and Pitts, 1943). El propio Frank Rosemblatt introdujo al perceptrón simple como un tipo particular de red neuronal.

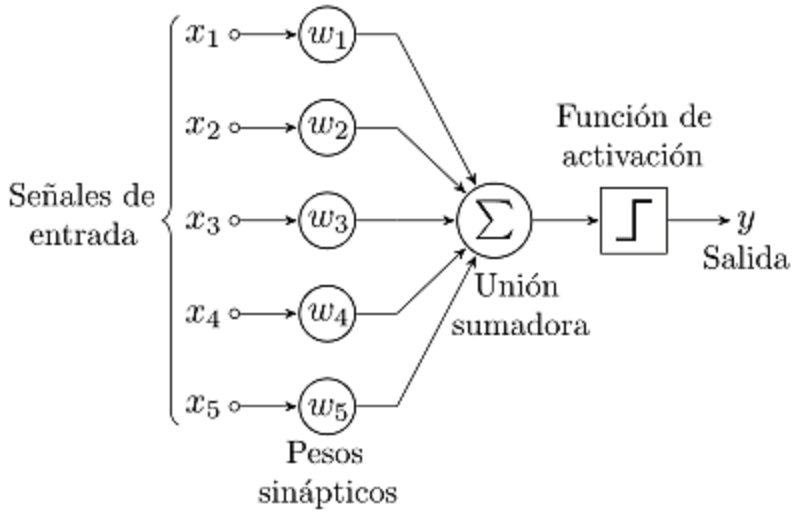

Figura 1.24: Diagrama de un perceptrón con cinco señales de entrada.

La siguiente ecuación representa la función de activación del perceptrón simple:

$$f(x) = \begin{cases} 1, & \langle x, w \rangle - u > 0 \\ 0, & e.o.c. \end{cases} \tag{1.32}$$

donde x es un vector con los valores de entrada, w es un vector de pesos reales y u es un umbral el cual representa el nivel de inhibición de la neurona.

El perceptrón es un clasificador lineal, por tanto, en el proceso de aprendizaje nunca llegará al estado en el que todos los vectores estén correctamente clasificados si el conjunto de entrenamiento no es linealmente separable.

Perceptrón multicapa

El perceptrón multicapa (MLP del inglés Multilayer Perceptron), es una generalización del perceptrón simple, con alimentación hacia delante y surge como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. El MLP como su nombre indica, está conformado por varias capas de neuronas, estas se clasifican en:

- Capa de entrada: Constituida por aquellas neuronas que introducen los valores de entrada en la red.
- Capas ocultas: Estas capas están formadas por neuronas cuyas entradas provienen de capas anteriores y sus salidas pasan a neuronas de capas posteriores.
- Capa de salida: La integran neuronas cuya salida corresponde con la salida de toda la red.

La arquitectura multicapa del MLP le permite establecer límites no lineales entre clases.

La propagación hacia atrás o retro-propagación es el algoritmo utilizado en el entrenamiento de este tipo de redes neuronales. Este algoritmo es también conocido como regla delta generalizada.

1.10 Conclusiones parciales

La computación afectiva es una rama de la inteligencia artificial que intenta diseñar y desarrollar sistemas y dispositivos capaces de reconocer, interpretar y procesar emociones humanas. Para ello utiliza tanto la detección del lenguaje corporal como el habla.

Las emociones expresan un estado de ánimo, una reacción del ser humano ante un desencadenante (que puede ser una acción externa). Su estudio no tiene solamente un valor en el campo de la medicina y la psicología, sino que sirve para la toma de decisiones en sistemas que interactúen con el ser humano.

La clasificación y detección de emociones del rostro humano, así como las tecnologías que permiten su reconocimiento son un problema abierto en el campo de la Computación Afectiva.

Los sistemas automáticos de reconocimiento de emociones faciales se basan todos en un modelo que se dividen en cuatro etapas: realce, detección del rostro, extracción de rasgos, y clasificación.

Los algoritmos más recomendados para el reconocimiento facial son: Viola-Jones y el algoritmo de Pirámide de Histogramas de Gradientes (PHOG). A partir de las pruebas realizadas se determina implementar en la biblioteca el algoritmo PHOG con un solo remuestreo de la imagen.

CAPÍTULO 2. Diseño de una biblioteca para la detección emociones faciales

En el presente capítulo se presenta el diseño de la biblioteca creada para el reconocimiento de emociones reflejadas en el rostro de los seres humanos.

Inicialmente se describen los recursos y lenguajes utilizados. Para la implementación de la biblioteca se utiliza el lenguaje Python en la versión 3. Algunos de los métodos utilizados se encuentran en las bibliotecas OpenCV, Dlib y Sklearn.

Finalmente se describe el proceso de instalación de la biblioteca para su uso.

2.1 Python

Python (*Welcome to Python.org*, 2018) es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Usa tipado dinámico, conteo de referencias para la administración de memoria y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse, en algunas situaciones, hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

Los usuarios de Python se refieren a menudo a la Filosofía Python que es bastante análoga a la filosofía de Unix. El código que sigue los principios de Python de legibilidad y transparencia se dice que es "pythonico". Contrariamente, el código opaco u ofuscado es bautizado como "no pythonico" ("unpythonic" en inglés). Estos principios fueron descritos por el desarrollador de Python Tim Peters en El Zen de Python.

2.2 OpenCV

OpenCV (Open Source Computer Vision Library) (*OpenCV library*, 2018) es una biblioteca de software abierto de visión por computadora y aprendizaje automático. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión por computadora. Al ser un producto con licencia de BSD, OpenCV facilita utilizar y modificar el código.

La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos de visión artificial y de aprendizaje automático tanto clásico como avanzado. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de la cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D desde cámaras estéreo, unir imágenes para producir una imagen de alta resolución de una escena completa, encontrar imágenes similares en una base de datos, eliminar ojos rojos de imágenes tomadas con flash, seguir movimientos oculares, reconocer el escenario y establecer marcadores para superponerlo con realidad aumentada, etc. OpenCV tiene más de 47 mil usuarios y un número estimado de descargas que excede los 14 millones. La biblioteca se usa ampliamente en compañías, grupos de investigación y por organismos gubernamentales.

Junto con compañías bien establecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota que emplean la biblioteca, hay muchas nuevas empresas como Applied Minds, VideoSurf y Zeitera, que hacen un uso extensivo de OpenCV. Algunos de los usos implementados de OpenCV son: detectar intrusiones en video de vigilancia en Israel, monitorear equipos mineros en China, ayudar a los robots a navegar y recoger objetos en Willow Garage, detectar accidentes de ahogamiento en piscinas en Europa, ejecutar arte interactivo en España y Nueva York, revisar las pistas de aterrizaje en busca de escombros en Turquía, inspeccionar etiquetas de productos en fábricas de todo el mundo y detección rápida de rostros en Japón.

Tiene interfaces para C ++, Python, Java y MATLAB. Es compatible con Windows, Linux, Android y Mac OS. OpenCV se inclina principalmente hacia las aplicaciones de visión en tiempo real y aprovecha las instrucciones de MMX y SSE cuando están disponibles. Está escrito nativamente en C ++.

2.3 Dlib

Dlib (*dlib C++ Library*, 2018) es una biblioteca multiplataforma de uso general escrita en el lenguaje de programación C ++. Contiene algoritmos de aprendizaje automático y herramientas para crear software para resolver problemas del mundo real. Su diseño está fuertemente influenciado por las ideas del diseño por contrato y la ingeniería de software basada en componentes.

Desde que comenzó su desarrollo en 2002, Dlib ha crecido para incluir una amplia variedad de herramientas. A partir de 2016, incluye componentes de software para trabajar con redes, hilos, interfaces gráficas de usuario, estructuras de datos, álgebra lineal, aprendizaje automático, procesamiento de imágenes, minería de datos, análisis XML y de texto, optimización numérica, redes bayesianas y muchas otras tareas.

Se usa tanto en la industria como en el campo académico en una amplia gama de dominios, incluidos la robótica, los dispositivos integrados, los teléfonos móviles y los grandes entornos informáticos de alto rendimiento. La licencia de código abierto de Dlib permite usarla en cualquier aplicación, sin costo alguno.

En el desarrollo de la biblioteca se utilizó la versión 3.4.0 de OpenCV. El paquete a importar para su uso es cv2 (por cuestiones de compatibilidad se sigue usando en lugar de cv3).

2.4 NumPy

NumPy (*NumPy — NumPy*, 2018) es el paquete fundamental para la computación científica en Python. Es una biblioteca que permite el trabajo con arreglos. Dentro de sus facilidades se encuentran:

- Operaciones matemáticas y lógicas entre arreglos.
- Manipulación de la forma de los arreglos.
- Transformadas discretas de Fourier.
- Álgebra lineal básica.

- Operaciones estadísticas básicas.
- Simulación aleatoria.
- Facilidades para la entrada y salida de los datos.

El objeto ndarray, núcleo del paquete, encapsula matrices n- dimensionales de tipos de datos homogéneos, con muchas operaciones que se realizan en código compilado para mayor rendimiento. Existen varias diferencias importantes entre las matrices NumPy y las secuencias estándar de Python:

- Las matrices NumPy tienen un tamaño fijo en la creación, a diferencia de las listas de Python (que pueden crecer dinámicamente). Cambiar el tamaño de un ndarray creará un nuevo conjunto y eliminará el original.
- Se requiere que los elementos en una matriz NumPy sean del mismo tipo de datos y, por lo tanto, tendrán el mismo tamaño en memoria. Aunque usando objetos de Python como tipo de datos se puede tener un arreglo con elementos de varios tamaños.
- Las matrices NumPy facilitan operaciones matemáticas avanzadas y de otro tipo en grandes cantidades de datos. Típicamente, tales operaciones se ejecutan de manera más eficiente y con menos código de lo que es posible usando las secuencias incorporadas de Python.

Una gran cantidad creciente de paquetes científicos y matemáticos basados en Python están utilizando matrices NumPy; aunque estos suelen ser compatibles con la entrada de secuencia de Python, convierten dicha entrada en matrices NumPy antes del procesamiento, y a menudo dan como resultados matrices NumPy. En otras palabras, para utilizar de manera eficiente gran parte (quizás incluso la mayoría) del software actual basado en Python científico / matemático, simplemente saber cómo usar los tipos de secuencia incorporados de Python es insuficiente; también se necesita saber cómo usar las matrices NumPy.

2.5 Sklearn

Sklearn (*scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation*, 2018) es una biblioteca de software libre para el campo del aprendizaje automatizado en el lenguaje de programación Python. Presenta varios algoritmos de clasificación, regresión y agrupación. Incluye máquinas de soporte vectorial y bosques aleatorios. Está diseñado para interoperar con las bibliotecas numéricas y científicas de Python, NumPy y SciPy .

Está escrito principalmente en Python, con algunos algoritmos centrales escritos en Cython para lograr mayor rendimiento. Las máquinas de soporte vectorial se implementan mediante un contenedor Cython alrededor de LIBSVM; máquinas de vector de regresión logística y soporte lineal por un envoltorio similar alrededor de LIBLINEAR.

El proyecto Sklearn comenzó como scikits.learn, un proyecto Google Summer of Code de David Cournapeau. Su nombre proviene de la noción de que es un "SciKit" (SciPy Toolkit), una extensión de terceros desarrollada por separado y distribuida para SciPy. El código base original fue posteriormente reescrita por otros desarrolladores. En 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort y Vincent Michel, todos del Instituto Francés de Investigación en Ciencias de la Computación y Automatización, tomaron el liderazgo del proyecto e hicieron su primer lanzamiento público el 1 de febrero de 2010. Los usuarios de esta biblioteca la calificaron como "bien mantenida y popular" en noviembre de 2012.

A partir de 2018, Sklearn está en desarrollo activo.

2.6 Implementación de la biblioteca

Las emociones a detectar, así como la forma de expresar estas, pueden variar de una aplicación a otra. Por ello, la biblioteca permite crear un clasificador personalizado mediante el método `getEmotionClassifier` que recibe como parámetro la dirección del conjunto de datos de entrenamiento. Cada imagen debe estar contenida en una carpeta con el nombre de la emoción que representa, es decir, la estructura del conjunto de datos quedaría como se ilustra en la Figura 2.1.

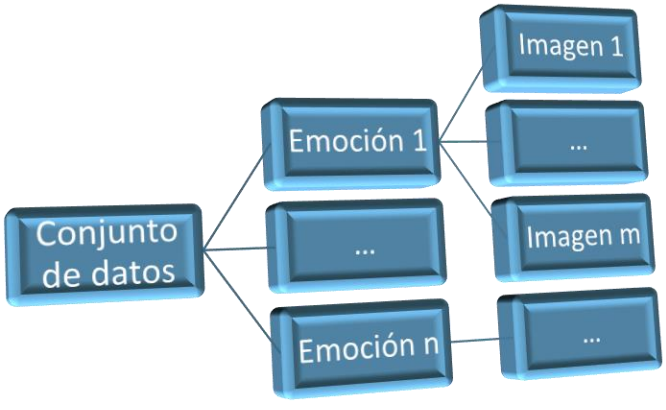

Figura 2.1: Estructura del conjunto de datos de entrenamiento.

La biblioteca está pensada para imágenes frontales de una sola persona.

De no suministrar ningún parámetro, un clasificador por defecto será retornado. Este es una SVM con kernel polinomial entrenado con 18 imágenes para cada una de las clases: miedo, tristeza, sorpresa, alegría, asco, desprecio, neutral e ira, del conjunto de datos Cohn-Kanade (Lucey *et al.*, 2010).

A cada imagen dentro del conjunto de entrenamiento se le aplica el método `describe`, el cual recibe la dirección de una imagen y devuelve el vector de características que la representa.

El parámetro `verbose` recibe un valor booleano. En caso de ser verdadero se mostrará qué imagen está siendo analizada actualmente, datos sobre su procesamiento y si en alguna imagen se han detectado más de un rostro o no se ha detectado ninguno.

El método `describe` carga la imagen haciendo uso del método `imread` de la biblioteca OpenCV. Este método recibe la dirección de la imagen y devuelve, en formato BGR, un arreglo de NumPy con la información del nivel de intensidad de los píxeles en cada canal.

La imagen es convertida a escala de grises usando el método de la biblioteca OpenCV `cvtColor`, el cual recibe un arreglo que represente a una imagen en un formato y devuelve otro que representa la misma imagen en otro formato. El formato de entrada y el de salida son determinados por el segundo parámetro, que es una constante de la propia biblioteca. En el caso actual, la constante utilizada es `COLOR_BGR2GRAY`.

Para normalizar la iluminación en la imagen un ecualizador adaptativo de histograma es creado con la función `createCLAHE` de OpenCV. El primer parámetro de este método (`clipLimit`) especifica el mayor número de píxeles que puede tener una barra del histograma, y el segundo (`tileGridSize`) especifica el tamaño de la ventana de la cual se calculará cada histograma. En el caso actual se utiliza como límite para las barras 2 y como tamaño de ventana (8,8). Si alguna barra del histograma está por encima del límite, esta es truncada y los píxeles sobrantes son redistribuidos uniformemente en las demás barras. Luego se aplica la ecualización del histograma, seguido de una interpolación bilineal.

Para evitar crear un nuevo ecualizador por cada imagen, este es creado una sola vez y luego con la función `apply` del propio objeto se le aplica a cada imagen del conjunto de datos.

Se pasa a la fase de detección del rostro haciendo uso del método `get_frontal_face_detector` de la biblioteca Dlib. Esta función es usada una sola vez para crear un reconocedor de rostros. Seguidamente, al reconocedor se le pasan cada una de las imágenes de la base de datos, el segundo parámetro indica cuántas veces se debe remuestrear la imagen. En este caso, la imagen es remuestreada una vez. Este método utiliza el algoritmo de histograma de orientación de gradientes, combinado con un clasificador lineal, y el método de detección de imagen piramidal y ventana deslizante. Este devuelve el rectángulo en el que está contenido el rostro.

Si ningún rostro es detectado, la imagen es ignorada. Si es detectado más de un rostro, es procesada solo la primera detección.

La imagen es recortada a la zona del rostro solamente, usando el operador `slice` de NumPy y posteriormente, con el método `resize` de la biblioteca OpenCV, es llevada a un tamaño de 350X350 para agregar invariancia ante la proximidad del rostro a la cámara.

Para detectar las marcas faciales se hace uso de la biblioteca Dlib, con el método `shape_predictor` se carga el archivo shape_predictor_68_face_landmarks.dat, que permite detectar las 68 marcas faciales reflejadas en la Figura 1.16. Este método es llamado una sola vez para crear el detector de marcas faciales, después, en cada imagen se especifica el rectángulo donde debe buscar la cara y devuelve una lista con las coordenadas de los 68 puntos.

El detector de marcas faciales está basado en el artículo (Kazemi and Sullivan, 2014) y fue entrenado con la base de datos iBUG 300-W.

Para agregar invariancia ante la rotación de la cabeza, una rotación es hecha a cada marca, recorriendo todas las marcas y separándolas en dos listas, una para cada componente, y restándole el valor correspondiente de las coordenadas de la marca facial 34, esto centra el eje de coordenadas en el centro del rostro. Seguidamente se calcula en ángulo de rotación de la cara usando la pendiente de la recta formada por la marca facial 28 y el origen de coordenadas, marca facial 34. Esta pendiente representa la tangente del ángulo que forma la línea calculada con respecto al eje de las x. Para calcular el ángulo de la rotación se le resta a $-\pi/2$ la arcotangente de la pendiente de la recta. Utilizando las transformaciones expresadas en la ecuación (2.1) se realiza la rotación:

$$x' = x * cos(\theta) - y * sin(\theta)$$
$$y' = y * cos(\theta) + x * sin(\theta) \quad (2.1)$$

donde x y y son las coordenadas originales del punto, θ es el ángulo de rotación y x' y y' son las coordenadas del punto luego de la transformación.

La Figura 2.2 muestra las marcas faciales antes y después de la rotación.

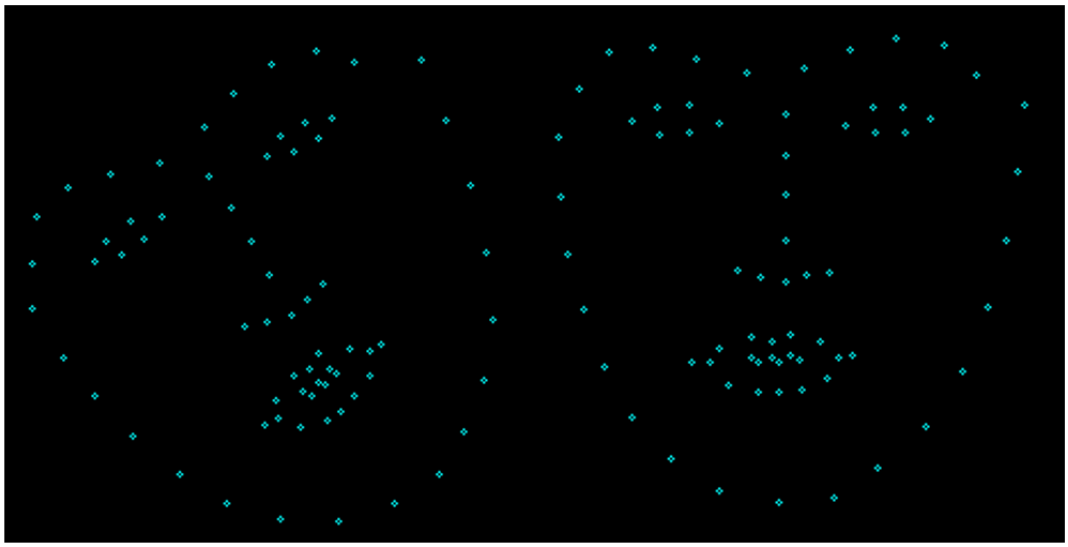

Figura 2.2: Rotación de un conjunto de marcas faciales.

Los valores rotados, almacenados en un solo arreglo de NumPy, son ordenados de acuerdo a la marca facial que representan. Es decir, el vector de rasgos obtenido tendrá 136 rasgos ordenados de la siguiente forma:

X de la marca 1, Y de la marca 1, ..., X de la marca 68, Y de la marca 68.

Luego de tener los vectores de rasgos de todas las imágenes en la base de datos se utiliza la biblioteca Sklearn para entrenar un clasificador, el cual puede ser de uno de los tres tipos siguientes:

- SVM con kernel lineal, donde la clasificación es probabilística, el valor gamma es 1/68, la penalización 1 y la tolerancia 0,0001.
- SVM con kernel polinomial, donde la clasificación es probabilística, el valor gamma es 1/68, la penalización 1 y la tolerancia 0,0001.

- MLP, con una capa oculta, donde la cantidad de nodos es igual a la semisuma de la cantidad de atributos y la cantidad de clases.

Este clasificador entrenado es retornado por el método `getEmotionDetector`.

Una vez que se tenga preparado el clasificador se utiliza el método `predictEmotion` que recibe como parámetros la dirección de la imagen a clasificar y un reconocedor entrenado. Este utiliza el método `describe` para obtener el vector de características de la imagen y luego pasa este resultado al clasificador. El nombre de la clase predicha es retornado como una cadena.

La Figura 2.3 muestra el diagrama de actividad del método `describe`.

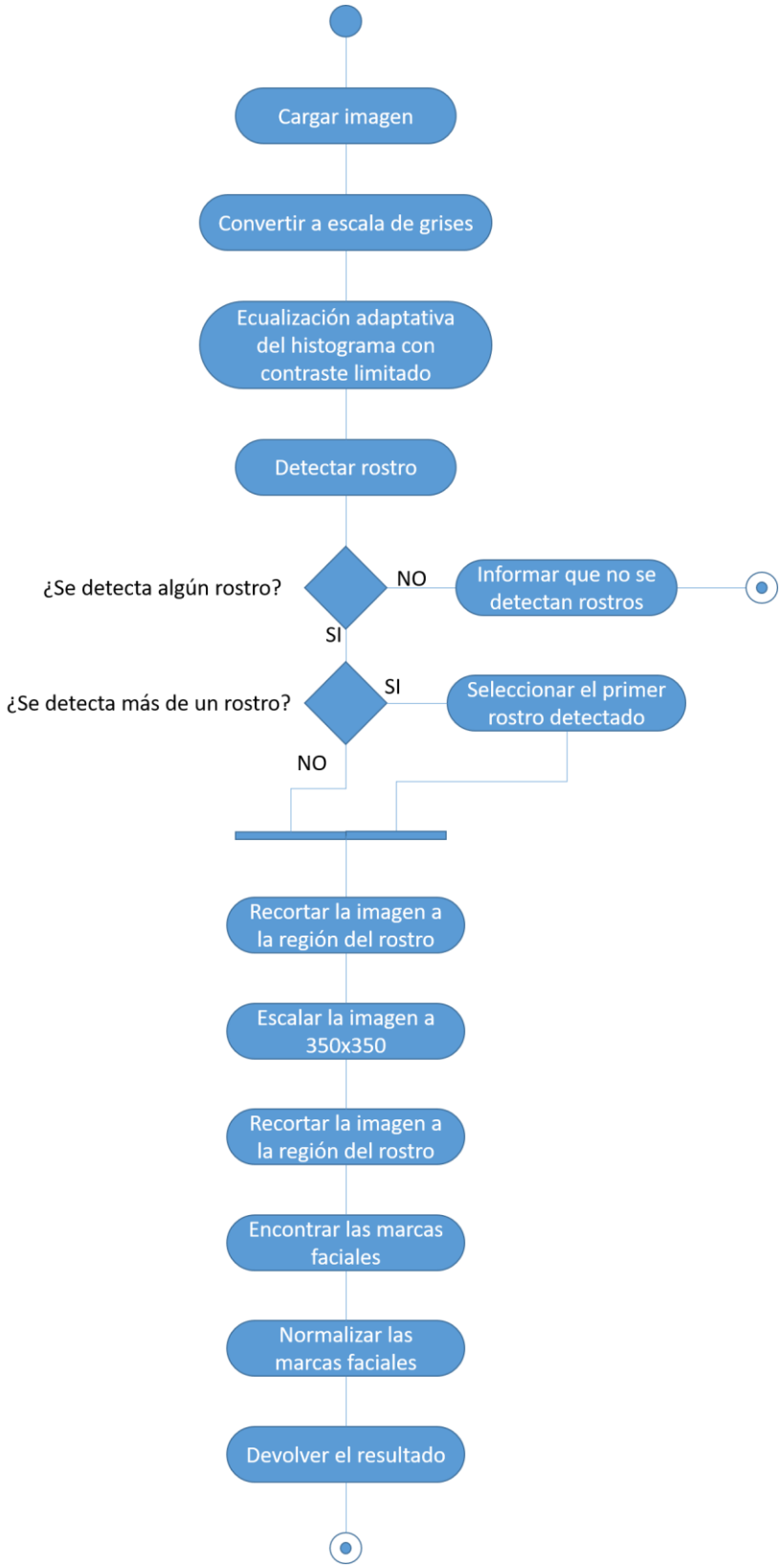

Figura 2.3: Diagrama de actividad del método *describe*.

En la Figura 2.4 se presenta el diagrama de actividad del método `getEmotionDetector`.

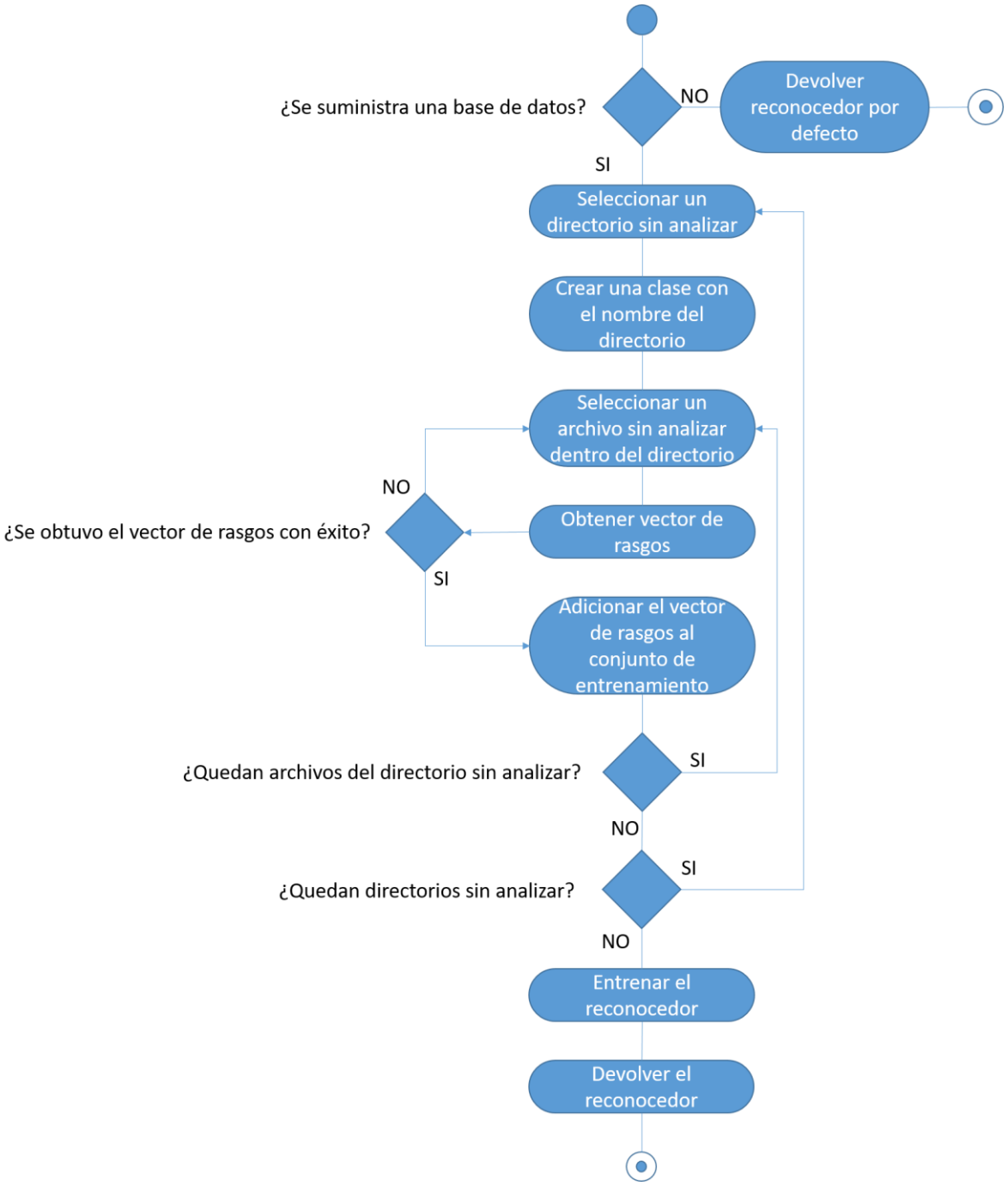

Figura 2.4: Diagrama de actividad del método *getEmotionDetector*.

En la Figura 2.5 se muestra la interacción con las bibliotecas utilizadas, así como el flujo de datos durante el procesamiento de un conjunto de datos y la clasificación de una imagen.

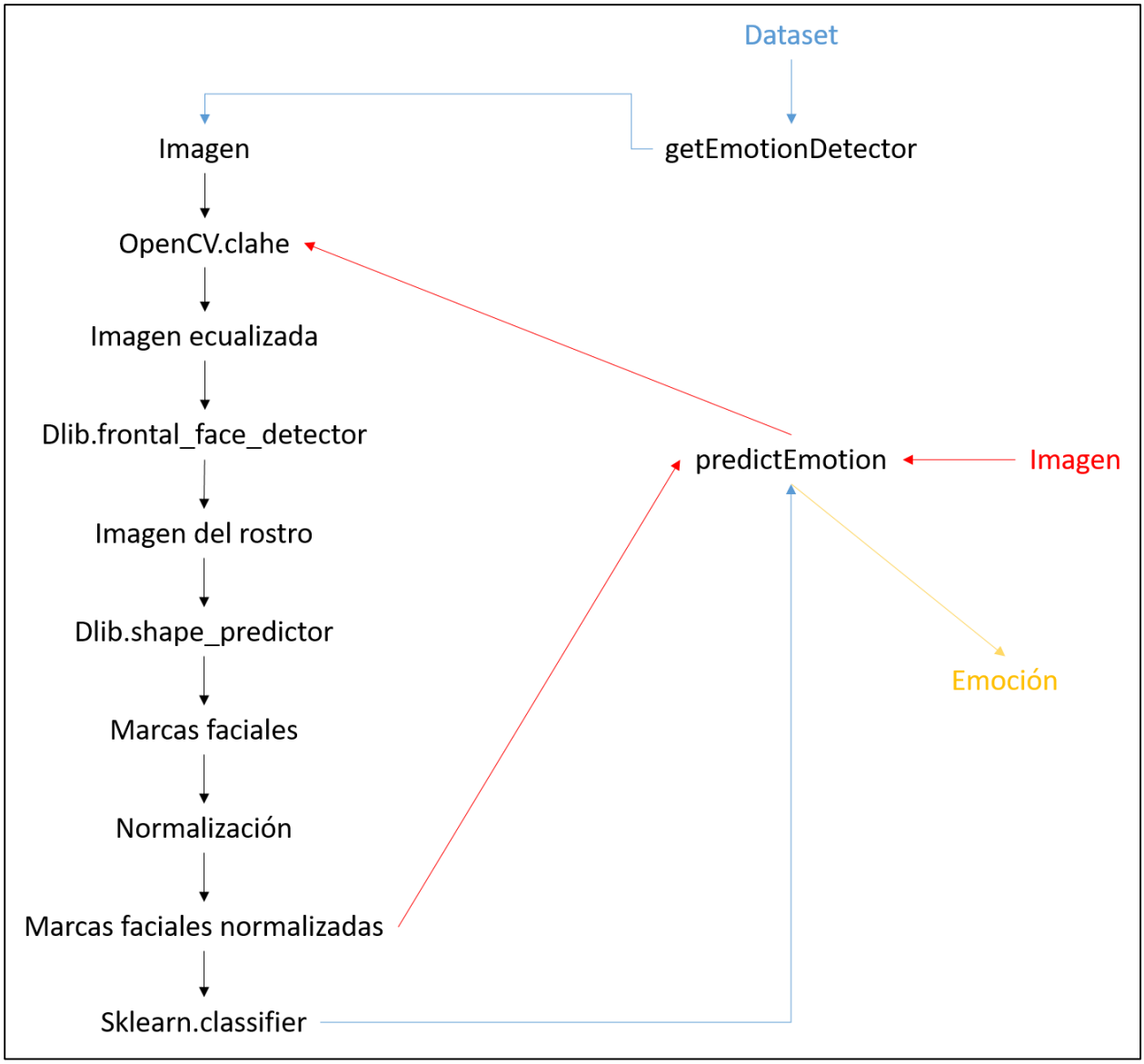

Figura 2.5: Flujo de datos durante el procesamiento de un conjunto de datos y la clasificación de una imagen.

Las flechas azules indican el procesamiento que se le realiza a cada una de las imágenes del conjunto de datos para luego entrenar el clasificador, el cual es suministrado como parámetro al método `predictEmotion`. Este método recibe además una imagen, cuyo procesamiento es representado por las flechas rojas. Haciendo uso del clasificador y del vector de rasgos correspondiente a esta imagen la emoción predicha es devuelta como se indica con la flecha amarilla.

2.7 Instalación

Para instalar la biblioteca y poder utilizarla como un módulo de Python3 se tienen dos vías:

- Descargar el código fuente de GitHub.
- Instalar el módulo desde Pypi usando el comando pip3.

Descargar el código fuente de GitHub

GitHub (*GitHub*, 2018) es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git que se usa principalmente para la creación de código fuente de programas de computadora. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

La descarga del proyecto puede hacerse mediante el link https://github.com/g-soto/euclv.git . Para su instalación, es necesario ubicarse en el directorio del proyecto y ejecutar el comando python3 setup.py install.

Desde el código en python solo se debe importar el paquete euclv.

La instalación de las dependencias debe hacerse de forma manual por el usuario.

Instalar utilizando el comando pip3

El Python Package Index o PyPI (*PyPI*, 2018) es el repositorio de software oficial para aplicaciones de terceros en el lenguaje de programación Python. PyPI es análogo a CPAN para Perl, o PEAR para PHP. Existen diversas herramientas para administrar los módulos instalados en Python que utilizan PyPI como su repositorio principal, entre ellas EasyInstall, pip, y PyPM.

Solo ejecutando el comando pip3 install euclv se tiene instalado el módulo, así como las dependencias de este.

Una vez hecho esto, se puede proceder a descargar el archivo shape_predictor_68_face_landmarks.dat, no incluido en el módulo por cuestiones de tamaño, desde el link http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2 y se copia para la

carpeta de instalación del módulo, la que varía en dependencia del sistema operativo y de si se utilizan entornos virtuales. Luego se puede importar el módulo euclv en el proyecto.

2.8 Conclusiones parciales

Python resulta ser uno de los mejore lenguajes, en la actualidad, para el procesamiento de imágenes y el trabajo con algoritmos de aprendizaje automático.

Entre las diversas bibliotecas, OpenCV, Dlib y Scikit-learn, resultan las más recomendadas para el procesamiento de imágenes, la extracción de rasgos y la clasificación de emociones.

Los métodos y funciones desarrollados en la implementación de la biblioteca permiten la detección en tiempo real de emociones humanas analizando las expresiones faciales.

Las emociones a detectar, así como la forma de expresar estas, pueden variar de una aplicación a otra. Por ello, la biblioteca permite crear un clasificador personalizado que recibe como parámetro la dirección del conjunto de datos de entrenamiento.

El diseño de la biblioteca fue concebido para que su uso posterior no exija de habilidades o conocimientos específicos por parte de los programadores.

CAPÍTULO 3. Análisis de resultados

En el presente capítulo se realizan pruebas de los algoritmos utilizados en la biblioteca con el empleo de dos conjuntos de imágenes organizados de acuerdo a los tipos de emociones identificadas por Paul Eckman y que son universalmente reconocidas.

3.1 Conjunto de datos Cohn-Kanade

En el año 2000 el conjunto de datos Cohn-Kanade (CK) (Lucey *et al.*, 2010) fue lanzado con el propósito de promover la investigación en el campo de la detección automática de expresiones faciales. Desde entonces CK se ha convertido en uno de los más usados para el desarrollo y evaluación de algoritmos en este campo.

Para la construcción de este conjunto de datos el comportamiento facial de 210 adultos fue grabado usando dos cámaras Panasonic AG-7500 sincronizadas por hardware. Los participantes se encuentran entre los 18 y 50 años de edad. De ellos 69% es del sexo femenino, 81% euro-americanos, 13% afro-americanos y 6% de otros grupos.

El conjunto de datos está dividido en sesiones de imágenes que muestran la transformación de la expresión facial de un individuo a partir de la neutral hasta llegar a una de las siguientes siete emociones:

- desprecio
- ira
- alegría
- tristeza
- sorpresa
- asco
- miedo

En la experimentación del presente trabajo se tomaron como muestras en cada sesión la primera imagen para la expresión neutral y la última imagen para su expresión correspondiente. La Figura 3.1 muestra la distribución por emociones de las imágenes seleccionadas.

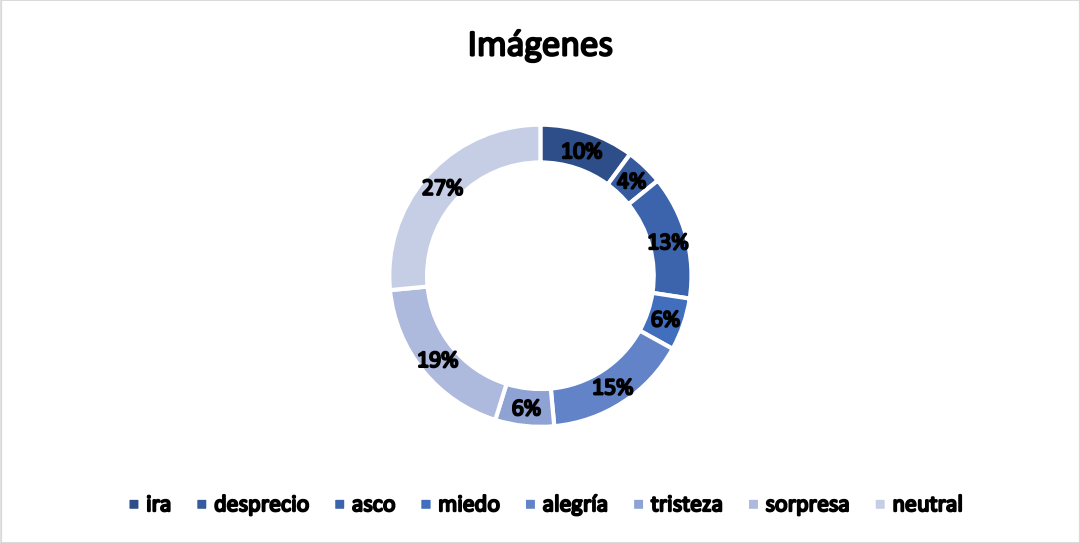

Figura 3.1: Distribución por emociones de las imágenes seleccionadas.

Se realizó una implementación del algoritmo de extracción de rasgos Local Minima, el cual reporta muy buenos resultados en la literatura, para comparar el desempeño del algoritmo propuesto en el presente trabajo.

3.2 Conjunto de datos FER

Este conjunto de datos fue preparado por Pierre-Luc Carrier y Aaron Courville (Goodfellow *et al.*, 2013), como parte de un proyecto de investigación. Los rostros tienen baja resolución y en ocasiones se encuentran girados, por lo que la exigencia de este conjunto de datos es mayor al del anteriormente expuesto y los resultados que se obtienen no son tan buenos.

Los datos consisten en imágenes de caras de 48x48 píxeles en escala de grises. Las caras se han registrado automáticamente para que estén más o menos centradas y ocupen aproximadamente la misma cantidad de espacio en cada imagen.

Los rostros están clasificados con un número correspondiente a una de siete categorías:

0 = ira, 1 = asco, 2 = miedo, 3 = alegría, 4 = tristeza, 5 = sorpresa, 6 = neutral.

El conjunto de datos está en formato CSV, donde la primera columna indica la emoción, la segunda columna contiene los niveles de intensidad de los píxeles o sea la imagen como tal, y la tercera columna indica si esa muestra debe ser usada para clasificación o entrenamiento.

El conjunto de entrenamiento (Figura 3.2, en azul) cuenta con 28709 casos. De ellos 7215 son de alegría, 4965 de neutral, 3171 de sorpresa, 3995 de ira, 4830 de tristeza, 4097 de miedo y 436 de asco.

El conjunto de prueba (Figura 3.2, en anaranjado) cuenta con 7178 casos. De ellos 1774 son de alegría, 1233 de neutral, 1247 de tristeza, 831 de sorpresa, 958 de ira, 1024 de miedo y 111 de asco.

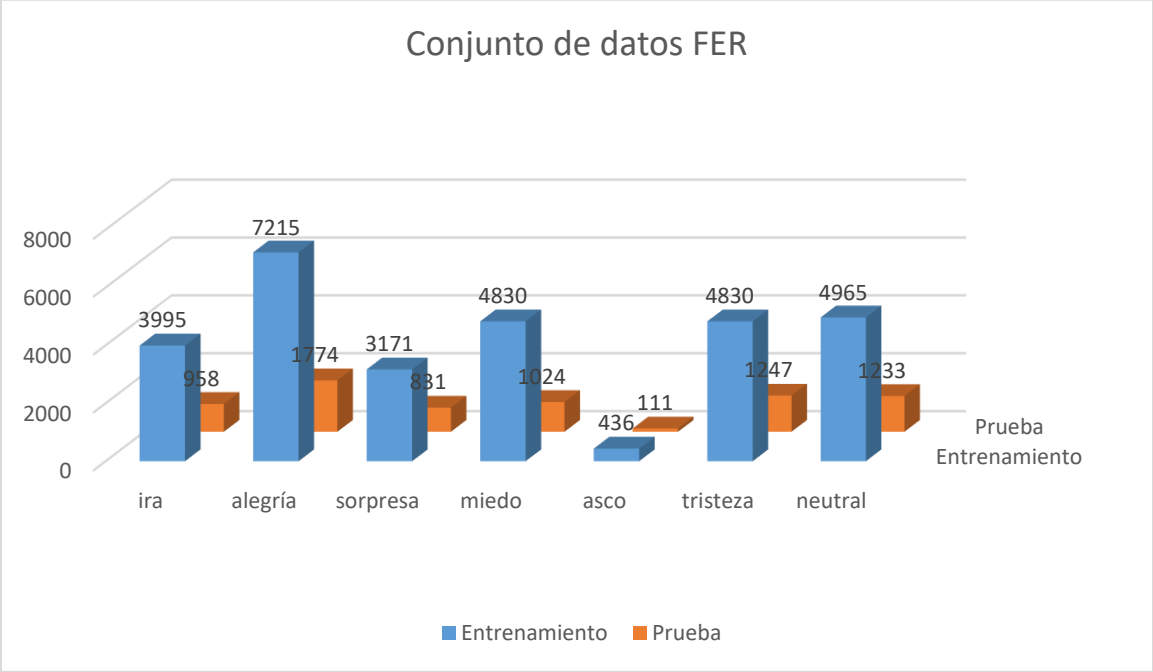

Figura 3.2: Distribución de imágenes del conjunto de datos FER2013.

En las pruebas con el método de marcas faciales se omitieron algunos casos donde falló el algoritmo de detección de rostros, quedando el conjunto de entrenamiento (Figura 3.3, en azul) con 21125 imágenes, de ellas 2837 de ira, 350 de asco, 2627 de miedo, 6013 de alegría, 2886 de tristeza, 2429 de sorpresa y 3983 de neutral. Mientras que el de prueba (Figura 3.3, en anaranjado) contiene 5299, de ellas 683 de ira, 87 de asco, 693 de miedo, 1474 de alegría, 738 de tristeza, 633 de sorpresa y 991 de neutral.

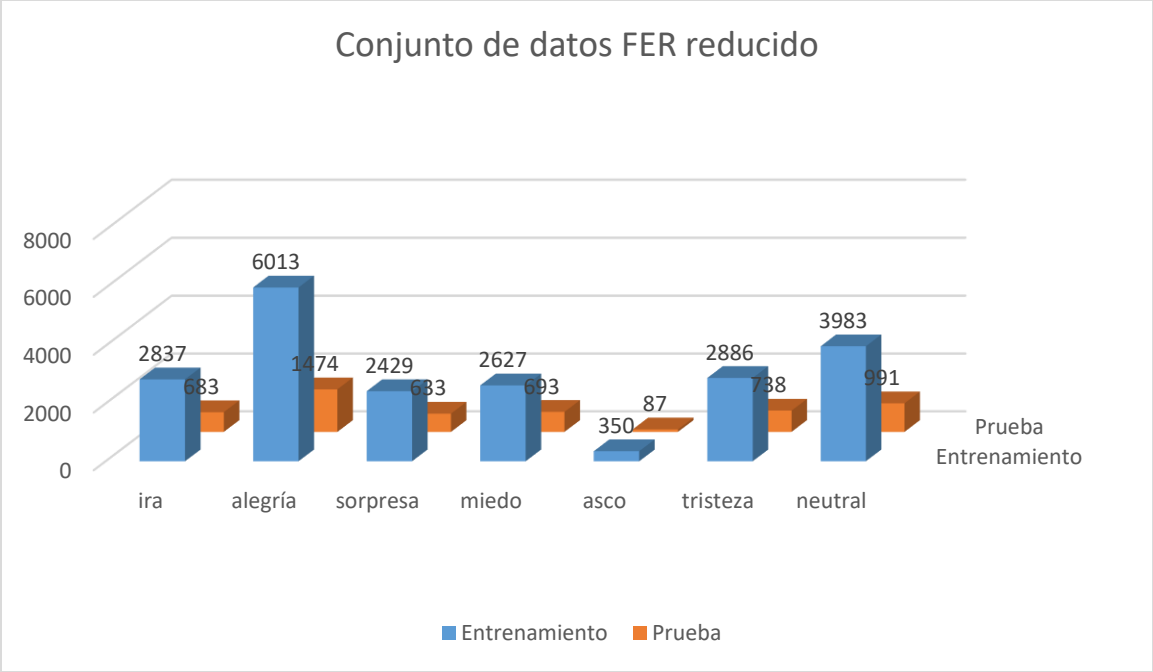

Figura 3.3: Conjunto de datos FER2013 reducido luego de aplicar detección de rostros en las imágenes.

3.3 Resultados obtenidos

Para realizar las pruebas se extrajeron los rasgos de las imágenes utilizando los algoritmos Marcas Faciales y Local Minima y estos fueron guardados en archivos arff. Los clasificadores utilizados son los implementados en la herramienta Weka. Para el conjunto de datos Cohn-Kanade se utilizaron cinco vueltas de validación cruzada. Mientras que para la FER2013 se utilizaron los conjuntos de entrenamiento y prueba que en esta se indican.

Debido al tamaño del conjunto de datos FER2013 las pruebas realizadas en este fueron corridas en el HPC de la Universidad.

3.3.1 Validación utilizando SVM con kernel lineal

El primer clasificador utilizado fue una SVM con kernel lineal, γ = 1/cantidad de atributos, ε = 0.001 y C = 1.

Utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade se clasificaron correctamente 358 (80.45%) imágenes y fueron mal clasificadas 87 (19.55%).

En la Tabla 3.1 se muestran los resultados por clase para este ejemplo, la clase sorpresa fue la de mejores resultados y las de peores resultados fueron ira, tristeza y desprecio.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.53	0.04	0.60	ira
0.44	0.02	0.53	desprecio
0.83	0.02	0.86	asco
0.84	0.01	0.88	miedo
0.97	0.01	0.97	alegría
0.83	0.10	0.75	neutral
0.94	0.01	0.94	sorpresa
0.46	0.03	0.52	tristeza
0.80	**0.04**	**0.08**	**Promedio**

Tabla 3.1: Estadísticas por clase del clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

La Tabla 3.2 muestra la matriz de confusión para este caso donde se muestra que los malos resultados obtenidos en ira, desprecio y tristeza fueron debido a imágenes clasificadas en su lugar como neutral.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
24	0	5	0	1	10	0	5	ira
1	**8**	0	0	0	7	0	2	desprecio
4	0	**49**	0	1	5	0	0	asco
0	0	0	**21**	0	0	2	2	miedo
1	0	0	1	**67**	0	0	0	alegría
8	3	3	0	0	**98**	3	3	neutral
0	1	0	2	0	2	**78**	0	sorpresa
2	3	1	0	0	9	0	**13**	tristeza

Tabla 3.2: Matriz de confusión para el clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

Aplicando la actual combinación de algoritmo de extracción de rasgos y clasificador SVM con kernel lineal para clasificar las emociones de las imágenes que integran el conjunto de datos FER2013, se obtuvieron los siguientes resultados: 2836 imágenes correctamente clasificadas para un 53.52% y 2836 imágenes mal clasificadas para un 46.48%.

La Tabla 3.3 muestra las estadísticas por clase de la clasificación en este conjunto de datos, para este caso todas las imágenes de la emoción asco fueron mal clasificadas, las emociones miedo, tristeza e ira obtuvieron porcientos de precisión por debajo del 40%. La emoción alegría obtuvo mejores resultados que el resto de las clases, pero el nivel de precisión no es tan bueno siendo este de 75%.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.51	0.13	0.37	ira
0	0	0	asco
0.15	0.05	0.3	miedo
0.83	0.11	0.75	alegría
0.25	0.08	0.35	tristeza
0.61	0.04	0.65	sorpresa
0.60	0.16	0.47	neutral
0.54	**0.01**	**0.51**	**Promedio**

Tabla 3.3: Estadísticas por clase del clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

La Tabla 3.4 muestra la matriz de confusión resultado de aplicar esta combinación en el conjunto de datos FER2013. Se puede observar que la emoción asco fue mal clasificada en su mayoría como ira. De hecho, las clases ira y neutral fueron las que más errores introdujeron en la clasificación de las imágenes en el resto de las emociones.

ira	asco	miedo	alegría	tristeza	sorpresa	neutral	←Clasificado como
349	0	40	90	68	30	106	ira
39	**0**	8	13	14	1	12	asco
138	0	**104**	82	103	98	168	miedo
69	0	34	**1218**	23	30	100	alegría
160	0	64	79	**183**	25	227	tristeza
45	0	50	74	20	**385**	59	sorpresa
138	0	47	71	117	21	**597**	neutral

Tabla 3.4: Matriz de confusión para el clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

Utilizando el método de extracción de rasgos Local Minima, en el conjunto de datos Cohn-Kanade se clasificaron correctamente 364 (81.8%) imágenes y fueron mal clasificadas 81 (18.2%).

La Tabla 3.5 muestra los resultados por clase obtenidos para este ejemplo, las clases de peores resultados en precisión fueron tristeza, ira y neutral, mientras que la precisión en el resto de las clases fue muy favorable.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.60	0.04	0.66	ira
0.50	0.01	0.90	desprecio
0.86	0.01	0.91	asco
0.60	0.01	0.94	miedo
0.94	0.02	0.92	alegría
0.87	0.12	0.72	neutral
0.86	0.01	0.95	sorpresa
0.50	0.02	0.58	tristeza

0.82	**0.04**	**0.82**	**Promedio**

Tabla 3.5: Estadísticas por clase del clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

La matriz de confusión mostrada en la Tabla 3.6, muestra nuevamente a las clases de peor precisión siendo mal clasificadas en su mayoría como neutral.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
27	0	1	0	0	15	0	2	ira
1	**9**	0	0	0	7	0	1	desprecio
2	0	**51**	0	2	3	0	1	asco
0	0	1	**15**	3	2	2	2	miedo
1	0	1	0	**65**	2	0	0	alegría
7	1	2	0	1	**103**	1	3	neutral
0	0	0	1	0	1	**80**	1	sorpresa
3	0	0	0	0	10	1	**14**	tristeza

Tabla 3.6: Matriz de confusión para el clasificador SVM con kernel lineal utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

3.3.2 Validación utilizando SVM con kernel polinomial

El siguiente clasificador utilizado fue una SVM con kernel polinomial de grado 3, γ = 1/cantidad de atributos, ε = 0.001 y C = 1.

Este clasificador con el método de extracción de rasgos Marcas Faciales obtuvo 355 (79.78%) imágenes bien clasificadas y 90 (20.22%) mal clasificadas.

En la Tabla 3.7 puede observarse la emoción tristeza como la clase de menor precisión obtenida en la clasificación, seguida de ira y desprecio. En este caso la clase de mayor precisión es alegría y a continuación le sigue sorpresa.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.53	0.05	0.54	ira
0.50	0.02	0.56	desprecio
0.81	0.02	0.89	asco
0.84	0.01	0.84	miedo
0.97	0.01	0.96	alegría
0.78	0.09	0.76	neutral
0.94	0.01	0.94	sorpresa
0.50	0.04	0.47	tristeza
0.80	**0.04**	**0.80**	**Promedio**

Tabla 3.7: Estadísticas por clase del clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

La Tabla 3.8 muestra que la principal causa de la baja precisión en la clasificación de las emociones ira, desprecio y tristeza es la clasificación de estas como neutral.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
24	1	4	0	1	10	0	5	ira
1	**9**	0	0	0	6	0	2	desprecio
4	0	**48**	0	2	5	0	0	asco
0	0	0	**21**	0	0	2	2	miedo
1	0	0	1	**67**	0	0	0	alegría
10	2	2	0	0	**94**	3	7	neutral
0	1	0	2	0	2	**78**	0	sorpresa
4	3	0	1	0	6	0	**14**	tristeza

Tabla 3.8: Matriz de confusión para el clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

Aplicando la actual combinación de algoritmo de extracción de rasgos Marcas Faciales y clasificador SVM con kernel polinomial de tercer grado para clasificar las emociones de las imágenes que integran el conjunto de datos FER2013, se obtuvieron los siguientes resultados: 2619 imágenes correctamente clasificadas para un 49.40% y 2681 imágenes mal clasificadas para un 50.60%.

La Tabla 3.9 muestra las estadísticas por clases obtenidas para esta configuración. La emoción asco con solo 17% de precisión es la de peores resultados, les sigue miedo y tristeza. La clase de más alta precisión fue sorpresa con solo 65%.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.31	0.09	0.35	ira
0.01	0.01	0.17	asco
0.12	0.05	0.25	miedo
0.85	0.19	0.63	alegría
0.26	0.09	0.32	tristeza
0.55	0.04	0.65	sorpresa
0.53	0.16	0.43	neutral
0.49	**0.12**	**0.46**	**Promedio**

Tabla 3.9: Estadísticas por clase del clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

La matriz de confusión obtenida se muestra en la Tabla 3.10, puede interpretarse a partir de esta que la principal fuente de errores en la clasificación de las imágenes fueron las clases neutral y alegría.

ira	asco	miedo	alegría	tristeza	sorpresa	neutral	←Clasificado como
214	3	49	143	94	35	145	ira
42	**1**	3	18	4	4	15	asco

112	1	**82**	139	105	84	170	miedo
36	1	36	**1255**	42	21	83	alegría
89	0	77	138	**192**	28	214	tristeza
27	0	37	116	26	**350**	77	sorpresa
90	0	40	188	129	20	**524**	neutral

Tabla 3.10: Matriz de confusión para el clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

Utilizando el método de extracción de rasgos Local Minima aplicado en el conjunto de datos Cohn-Kanade, se clasificaron correctamente 358 (80.45%) imágenes y fueron mal clasificadas 87 (19.55%).

En la Tabla 3.11 puede observarse la emoción tristeza como la clase de más baja precisión seguida de ira y neutral. La clase sorpresa es la de precisión más alta. Las emociones asco y miedo también muestran altos valores de precisión.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.60	0.04	0.64	ira
0.50	0.01	0.75	desprecio
0.85	0.01	0.94	asco
0.56	0.01	0.93	miedo
0.96	0.02	0.90	alegría
0.84	0.13	0.70	neutral
0.96	0.01	0.95	sorpresa
0.46	0.03	0.54	tristeza
0.80	**0.05**	**0.81**	**Promedio**

Tabla 3.11: Estadísticas por clase del clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

La matriz de confusión para este ejemplo (Tabla 3.12) muestra la emoción ira clasificada en 16 ejemplos como neutral y la emoción tristeza 11 veces clasificada también como neutral. Esto justifica la baja precisión de la clasificación alcanzada en estas clases.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
27	0	1	0	0	16	0	1	ira
1	**9**	0	0	0	7	0	1	desprecio
3	0	**50**	0	2	3	0	1	asco
0	0	1	**14**	3	3	2	2	miedo
1	0	0	0	**66**	2	0	0	alegría
7	3	1	0	2	**99**	1	5	neutral
0	0	0	1	0	1	**80**	1	sorpresa
3	0	0	0	0	11	1	**13**	tristeza

Tabla 3.12: Matriz de confusión para el clasificador SVM con kernel polinomial utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

3.3.3 Validación utilizando Perceptrón multicapa

En esta sección se utiliza un clasificador tipo perceptrón multicapa (MLP), con solo una capa oculta y 72 neuronas en esta.

Para el método de extracción de rasgos Marcas Faciales, se clasificaron correctamente 364 (81.8%) imágenes e incorrectamente 81 (18.2%) imágenes.

La Tabla 3.13 muestra las clases desprecio, tristeza e ira como las de menor precisión y las emociones alegría y sorpresa alcanzan valores muy altos de precisión.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.64	0.04	0.67	ira
0.33	0.02	0.46	desprecio
0.92	0.02	0.9	asco
0.8	0.01	0.83	miedo

0.99	0.01	0.97	alegría
0.81	0.01	0.75	neutral
0.95	0.01	0.95	sorpresa
0.43	0.03	0.5	tristeza
0.81	**0.04**	**0.81**	**Promedio**

Tabla 3.13: Estadísticas por clase del clasificador MLP utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

La Tabla 3.14 muestra la matriz de confusión obtenida para este ejemplo. La expresión neutral es la clase que más genera errores en la clasificación de las clases ira, desprecio y tristeza, justamente las clases de menor precisión en esta configuración.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
29	1	3	0	1	8	0	3	ira
1	**6**	0	0	0	9	0	2	desprecio
2	0	**54**	1	0	2	0	0	asco
0	0	1	**20**	1	0	2	1	miedo
0	1	0	0	**68**	0	0	0	alegría
10	2	2	1	0	**96**	1	6	neutral
0	1	0	1	0	2	**79**	0	sorpresa
1	2	0	1	0	11	1	**12**	tristeza

Tabla 3.14: Matriz de confusión para el clasificador MLP utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos Cohn-Kanade.

Aplicando en el conjunto de datos FER2013 la actual combinación de algoritmo de extracción de rasgos Marcas Faciales y clasificador MLP, se obtuvieron los siguientes resultados: 3009 imágenes correctamente clasificadas para un 56.78% y 2290 imágenes mal clasificadas para un 43.21%.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.46	0.09	0.44	ira
0.3	0.01	0.413	asco
0.31	0.01	0.33	miedo
0.83	0.09	0.79	alegría
0.3	0.08	0.4	tristeza
0.66	0.05	0.66	sorpresa
0.6	0.12	0.52	neutral
0.57	**0.09**	**0.56**	**Promedio**

Tabla 3.15: Estadísticas por clase del clasificador MLP utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

La Tabla 3.15 muestra las estadísticas por clases obtenidas, las emociones de menor precisión fueron miedo y tristeza seguidas de asco e ira. La emoción alegría es la de mejor precisión con resultados aceptablemente buenos (79%).

En la Tabla 3.16 se presentan los datos de la matriz de confusión.

ira	asco	miedo	alegría	tristeza	sorpresa	neutral	←Clasificado como
314	13	82	66	68	41	99	ira
25	**26**	9	7	11	3	6	asco
109	8	**215**	63	103	77	118	miedo
47	0	59	**1220**	31	33	84	alegría
96	6	127	68	**224**	25	192	tristeza
31	3	86	47	16	**419**	31	sorpresa
93	7	68	79	114	39	**591**	neutral

Tabla 3.16: Matriz de confusión para el clasificador MLP utilizando el método de extracción de rasgos Marcas Faciales en el conjunto de datos FER2013.

Para el método de extracción de rasgos Local Minima, se clasificaron correctamente 348 (78.2%) imágenes e incorrectamente 97 (21.8%) imágenes.

En la Tabla 3.17 se observan las emociones tristeza y desprecio como las clases de menor precisión mientras que las emociones sorpresa y asco son las clases de mejor precisión obtenida para este caso.

Razón de verdaderos positivos	Razón de falsos positivos	Precisión	Clase
0.62	0.03	0.7	ira
0.5	0.02	0.53	desprecio
0.86	0.01	0.91	asco
0.52	0.01	0.76	miedo
0.94	0.03	0.87	alegría
0.82	0.13	0.7	neutral
0.94	0.02	0.92	sorpresa
0.25	0.02	0.44	tristeza
0.78	**0.05**	**0.77**	**Promedio**

Tabla 3.17: Estadísticas por clase del clasificador MLP utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

La Tabla 3.18 muestra la matriz de confusión obtenida. Puede apreciarse que las imágenes de las emociones ira, desprecio y tristeza fueron mal clasificadas en su mayoría como neutral.

ira	desprecio	asco	miedo	alegría	neutral	sorpresa	tristeza	←Clasificado como
28	1	0	0	1	12	0	3	ira
1	**9**	0	0	0	7	0	1	desprecio
1	0	**51**	0	0	5	1	1	asco

0	1	0	**13**	6	2	3	0	miedo
0	1	1	1	**65**	1	0	0	alegría
8	4	2	0	2	**97**	2	3	neutral
0	1	0	2	1	0	**78**	1	sorpresa
2	0	2	1	0	15	1	**7**	tristeza

Tabla 3.18: Matriz de confusión para el clasificador MLP utilizando el método de extracción de rasgos Local Minima en el conjunto de datos Cohn-Kanade.

Para el conjunto de datos Cohn-Kanade la configuración que mejor resultado obtuvo en precisión fue Marcas Faciales como método de extracción rasgos y MLP como clasificador. Las clases de mayor precisión obtenida fueron sorpresa y alegría. La precisión obtenida en el resto de las clases fue afectada por malas clasificaciones de estas en neutral. La principal causa de esta deficiencia es la falta de ejemplos por clases y equilibrio entre estas para este conjunto de datos.

3.4 Conclusiones parciales

El conjunto de datos Cohn-Kanade (CK) fue lanzado con el propósito de promover la investigación en el campo de la detección automática de expresiones faciales, siendo uno de los más usados para el desarrollo y evaluación de algoritmos en este campo. Para la construcción de este conjunto de datos fue grabado el comportamiento facial de 210 adultos entre los 18 y 50 años de edad. De ellos 69% es del sexo femenino, 81% euro-americanos, 13% afro-americanos y 6% de otros grupos.

El conjunto de datos FER fue preparado por Pierre-Luc Carrier y Aaron Courville, como parte de un proyecto de investigación. Los rostros tienen baja resolución y en ocasiones se encuentran girados, por lo que la exigencia de este conjunto de datos es mayor y los resultados que se obtienen no son tan buenos. El conjunto de entrenamiento cuenta con 28709 casos. El conjunto de prueba cuenta con 7178 casos.

Los resultados obtenidos en las validaciones son similares a los reportados en la literatura.

CONCLUSIONES

- La computación afectiva es una rama de la inteligencia artificial que intenta diseñar y desarrollar sistemas y dispositivos capaces de reconocer, interpretar y procesar emociones humanas. Para ello utiliza tanto la detección del lenguaje corporal como el habla.
- La detección y clasificación de las emociones reflejadas en el rostro humano está cobrando mucha importancia en la actualidad, en especial para mejorar la interacción con sistemas computacionales.
- La clasificación y detección de emociones del rostro humano, así como las tecnologías que permiten su reconocimiento son un problema abierto en el campo de la Computación Afectiva.
- Los algoritmos más recomendados para el reconocimiento facial son: Viola-Jones y el algoritmo de Pirámide de Histogramas de Gradientes (PHOG). A partir de las pruebas realizadas se determina implementar en la biblioteca el algoritmo PHOG con un solo muestreo de la imagen.
- Python resulta ser uno de los mejore lenguajes, en la actualidad, para el procesamiento de imágenes y el trabajo con algoritmos de aprendizaje automático.
- Entre las diversas bibliotecas, OpenCV, Dlib y Scikit-learn, resultan las más recomendadas para el procesamiento de imágenes, la extracción de rasgos y la clasificación de emociones.
- Los métodos y funciones desarrollados en la implementación de la biblioteca permiten la detección en tiempo real de emociones humanas analizando las expresiones faciales.
- Las emociones a detectar, así como la forma de expresar estas, pueden variar de una aplicación a otra. Por ello, la biblioteca permite crear un clasificador personalizado que recibe como parámetro la dirección del conjunto de datos de entrenamiento.
- El diseño de la biblioteca fue concebido para que su uso posterior no exija de habilidades o conocimientos específicos por parte de los programadores.
- Los resultados obtenidos son similares a los reportados en la literatura.
- La biblioteca propuesta es un primer acercamiento al empleo de la computación afectiva. Los resultados aquí alcanzados y su posterior implementación abren las posibilidades al desarrollo de investigaciones futuras que exploten el empleo de esta disciplina científica.

RECOMENDACIONES

- Debido a que las pruebas se realizaron en conjuntos de datos desequilibrados, el promedio de precisión obtenido pudiera mejorarse al medir el desempeño de la biblioteca en un conjunto de datos consistente y con las clases bien equilibradas.
- Adicionar nuevos clasificadores a la biblioteca y añadir parámetros que permitan ajustar el clasificador para flexibilizar las pruebas y la propia clasificación.
- Trabajar en implementaciones de programas que utilicen la biblioteca aquí desarrollada.

BIBLIOGRAFÍA

A Short introduction to descriptors | Gil's CV blog (2013). Available at: https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/ (Accessed: 13 June 2018).

Bosch, A., Zisserman, A. and Munoz, X. (2007) 'Representing shape with a spatial pyramid kernel', in *Proceedings of the 6th ACM international conference on Image and video retrieval - CIVR '07*. New York, New York, USA: ACM Press, pp. 401–408. doi: 10.1145/1282280.1282340.

Chanel, G., Ansari-Asl, K. and Pun, T. (2007) 'Valence-arousal evaluation using physiological signals in an emotion recall paradigm', in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 2662–2667.

Dalal, N. and Triggs, B. (2005) 'Histograms of Oriented Gradients for Human Detection'. Available at: https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf (Accessed: 13 June 2018).

Dalgleish, T. (2004) 'The emotional brain', *Nature Reviews Neuroscience*, 5(7), pp. 583–589. doi: 10.1038/nrn1432.

Darwin, C. and Fernaández Rodríguez, T. R. (1984) *La expresión de las emociones en los animales y en el hombre*. Alianza. Available at: https://www.casadellibro.com/libro-la-expresion-de-las-emociones-en-los-animales-y-en-el-hombre/9788420600116/431053 (Accessed: 12 June 2018).

dlib C++ Library (2018). Available at: http://dlib.net/ (Accessed: 13 June 2018).

Duff, G. (1966) *Differential equations of applied mathematics*. New York: Wiley. Available at: http://www.worldcat.org/title/differential-equations-of-applied-mathematics/oclc/527930 (Accessed: 10 June 2018).

Ekman, P. (1993) 'Facial expression and emotion.', *American Psychologist*, 48(4), pp. 384–392. doi: 10.1037/0003-066X.48.4.384.

Ekman, P. and Friesen, W. V. (1975) *Unmasking the face; a guide to recognizing emotions from facial clues*. Prentice-Hall.

Ekman, P. and Rosenberg, E. L. (2005) 'What the Face Reveals: Basic and Applied Studies of

Spontaneous Expression Using the Facial Action Coding System (FACS), Second Edition'. Available at: https://www.aqualide.com/upload/texte/text98.pdf (Accessed: 13 June 2018).

Ekman, P., Sorenson, E. R. and Friesen, W. V (1969) 'Pan-cultural elements in facial displays of emotion.', *Science (New York, N.Y.)*, 164(3875), pp. 86–8. Available at: http://www.ncbi.nlm.nih.gov/pubmed/5773719 (Accessed: 13 June 2018).

Facial landmarks with dlib, OpenCV, and Python - PyImageSearch (2017). Available at: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/ (Accessed: 13 June 2018).

Fragopanagos, N. and Taylor, J. G. (2005) 'Emotion recognition in human–computer interaction', *Neural Networks*, 18(4), pp. 389–405. doi: 10.1016/j.neunet.2005.03.006.

Freund, Y. and Schapire, R. E. (1997) 'A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting', *Journal of Computer and System Sciences*. Academic Press, 55(1), pp. 119–139. doi: 10.1006/JCSS.1997.1504.

Goodfellow, I. J. *et al.* (2013) 'Challenges in Representation Learning: A report on three machine learning contests'. Available at: https://arxiv.org/pdf/1307.0414.pdf (Accessed: 13 June 2018).

Graciarena, M. *et al.* (2006a) 'Combining prosodic lexical and cepstral systems for deceptive speech detection', in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 1033–1036.

Graciarena, M. *et al.* (2006b) 'Combining Prosodic Lexical and Cepstral Systems for Deceptive Speech Detection', in *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*. IEEE, p. I-1033-I-1036. doi: 10.1109/ICASSP.2006.1660200.

Grauman, K. and Darrell, T. (2005) 'Pyramid Match Kernels: Discriminative Classification with Sets of Image Features'. Available at: https://pdfs.semanticscholar.org/fc30/98cff5469c55c3e81dc127563afe6dbadf22.pdf (Accessed: 13 June 2018).

Hammal, Z. *et al.* (2007) 'Facial expression classification: An approach based on the fusion of facial deformations using the transferable belief model', *International Journal of Approximate Reasoning*. Elsevier, 46(3), pp. 542–567. doi: 10.1016/J.IJAR.2007.02.003.

Hatem, H., Beiji, Z. and Majeed, R. (2015) 'A Survey of Feature Base Methods for Human Face Detection', *International Journal of Control and Automation*, 8(5), pp. 61–78. doi: 10.14257/ijca.2015.8.5.07.

Hjelmås, E. and Low, B. K. (2001) 'Face Detection: A Survey', *Computer Vision and Image Understanding*. Academic Press, 83(3), pp. 236–274. doi: 10.1006/CVIU.2001.0921.

Hussain Shah, J. *et al.* (2015) 'Robust Face Recognition Technique under Varying Illumination', *Journal of Applied Research and Technology*. Elsevier, 13(1), pp. 97–105. doi: 10.1016/S1665-6423(15)30008-0.

James, G. *et al.* (2000) *An introduction to Statistical Learning*, *Current medicinal chemistry*. doi: 10.1007/978-1-4614-7138-7.

Kazemi, V. and Sullivan, J. (2014) 'One millisecond face alignment with an ensemble of regression trees', in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1867–1874. doi: 10.1109/CVPR.2014.241.

Klette, R. (2014) 'Concise Computer Vision-An Introduction into Theory and Algorithms. Undergraduate Topics in Computer Science'. Available at: https://epz661djg03.storage.googleapis.com/MTQ0NzE2MzE5Mg==01.pdf (Accessed: 13 June 2018).

Lajevardi, S. M. and Hussain, Z. M. (2009) 'Local Feature Extraction Methods for Facial Expression Recognition', *Signal Processing*, 3(Eusipco), pp. 60–64. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5335546.

Local Binary Patterns (2018). Available at: https://bytefish.de/blog/local_binary_patterns/ (Accessed: 13 June 2018).

Local Binary Patterns with Python & OpenCV - PyImageSearch (2018). Available at: https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/ (Accessed: 13 June 2018).

López Mejía, D. I. *et al.* (2009) 'El Sistema Límbico y las Emociones: Empatía en Humanos y Primates', *Psicología Iberoamericana*, 17, pp. 60–69.

Lucey, P. *et al.* (2010) 'The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression', in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. IEEE, pp. 94–101. doi: 10.1109/CVPRW.2010.5543262.

Magudeeswaran, V. and Singh, J. F. (2017) 'Contrast limited fuzzy adaptive histogram equalization for enhancement of brain images', *International Journal of Imaging Systems and Technology*. Wiley-Blackwell, 27(1), pp. 98–103. doi: 10.1002/ima.22214.

McCulloch, W. S. and Pitts, W. (1943) 'A logical calculus of the ideas immanent in nervous activity', *The Bulletin of Mathematical Biophysics*. Kluwer Academic Publishers, 5(4), pp. 115–133. doi: 10.1007/BF02478259.

Mohammad, S. I. and Surapong, A. (2013) 'A Novel Feature Extraction Technique for Facial Expression Recognition', *International Journal of Computer Science*, 10(1), pp. 9–14. doi: 10.2991/jcis.2006.259.

Negi, S. S. and Bhandari, Y. S. (2014) 'A hybrid approach to Image Enhancement using Contrast Stretching on Image Sharpening and the analysis of various cases arising using histogram', in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*. IEEE, pp. 1–6. doi: 10.1109/ICRAIE.2014.6909232.

NumPy — NumPy (2018). Available at: http://www.numpy.org/ (Accessed: 13 June 2018).

Ojala, T. *et al.* (1996) 'A comparative study of texture measures with classification based on featured distributions', *Elsevier*. Available at: https://www.sciencedirect.com/science/article/pii/0031320395000674 (Accessed: 13 June 2018).

OpenCV: Histograms - 2: Histogram Equalization (2018). Available at: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html (Accessed: 13 June 2018).

OpenCV library (2018). Available at: https://opencv.org/ (Accessed: 13 June 2018).

Palmero Cantero, F. (1996) 'Aproximación biológica al estudio de la emoción', *Anales de psicología*, 12(1), pp. 61–86. Available at: http://dialnet.unirioja.es/servlet/articulo?codigo=815050&info=resumen&idioma=SPA.

Perceptrón - Wikipedia, la enciclopedia libre (2015). Available at: https://es.wikipedia.org/wiki/Perceptrón (Accessed: 13 June 2018).

Pérez Nieto, M. and Delgado, M. (2018) *Procesos de valoración y emoción: características, desarrollo, clasificación y estado actual., REME, ISSN 1138-493X, Vol. 9, Nº. 22, 2006*.

Picard (1995) 'Affective computing', (321).

Picard, R. W. (1997) 'Affective Computing', *The MIT Press*.

Picard, R. W., Vyzas, E. and Healey, J. (2001) 'Toward machine emotional intelligence: analysis of affective physiological state', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), pp. 1175–1191. doi: 10.1109/34.954607.

PyPI – the Python Package Index · PyPI (2018). Available at: https://pypi.org/ (Accessed: 13 June 2018).

Rosenblatt, F. (1958) 'The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain', *PSYCHOLOGICAL REVIEW*, pp. 65--386. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.588.3775 (Accessed: 10 June 2018).

scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation (2018). Available at: http://scikit-learn.org/stable/index.html (Accessed: 13 June 2018).

Sutil, C. R. (1998) 'Emoción y cognición. James, más de cien años después', 29(3), pp. 3–23. Available at: https://www.raco.cat/index.php/AnuarioPsicologia/article/viewFile/61489/88336 (Accessed: 13 June 2018).

The world's leading software development platform · GitHub (2018). Available at: https://github.com/ (Accessed: 13 June 2018).

Turati, C. *et al.* (2006) 'Newborns' Face Recognition: Role of Inner and Outer Facial Features', *Child Development*, 77(2), pp. 297–311. doi: 10.1111/j.1467-8624.2006.00871.x.

Vapnik (1979) 'Support Vector Machines'.

Varsha Gupta, M. and Dipesh, S. (2014) 'A Study of Various Face Detection Methods', *International Journal of Advanced Research in Computer and Communication Engineering*, 3(5), pp. 2278–1021. Available at: www.ijarcce.com (Accessed: 13 June 2018).

Viola, P. and Jones, M. (2001) ‘Robust Real-time Object Detection’, *INTERNATIONAL JOURNAL OF COMPUTER VISION*. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.4868 (Accessed: 26 April 2018).

Welcome to Python.org (2018). Available at: https://www.python.org/ (Accessed: 13 June 2018).

Wurtz, R. H. (2009) ‘Recounting the impact of Hubel and Wiesel.’, *The Journal of physiology*. Wiley-Blackwell, 587(Pt 12), pp. 2817–23. doi: 10.1113/jphysiol.2009.170209.

Zapatero Olmedillo, D. (2016) ‘Herramienta de reconocimiento facial de emociones en Android’, pp. 2–5. Available at: http://oa.upm.es/44722/.