VOLUMEN 21
XXVIII Verano De la Ciencia
ISSN 2395-9797
www.jóvenesenlaciencia.ugto.mx

Reconocimiento de Expresiones Faciales en Imágenes Usando Bosquejos Simples Conectando Puntos de Referencia

Fernando Vargas Rodríguez[1], Eduardo D. Rodríguez Anda[2], Carlos H. García-Capulín[3], Raúl E. Sánchez-Yáñez[4]

[1,2,3,4] Universidad de Guanajuato, División de ingenierías del Campus Irapuato-Salamanca; Salamanca, Gto. 36885, México

f.vargasrodriguez@ugto.mx[1], ed.rodriguezanda@ugto.mx[2], carlosg@ugto.mx[3], sanchezy@ugto.mx[4]

Resumen

En este trabajo se presenta un nuevo enfoque para el reconocimiento de expresiones faciales en imágenes mediante el uso de bosquejos (o bocetos, aunque no son exactamente sinónimos, los términos se usarán indistintamente en esta trabajo) y el descriptor denominado Histograma de Gradientes Orientados, combinado con una Red Neuronal Artificial usada como clasificador. El objetivo principal de esta investigación es desarrollar un sistema preciso y eficiente, capaz de reconocer expresiones en rostros humanos, posibilitando aplicaciones significativas en campos como el análisis del comportamiento humano, las interfaces hombre-máquina y los sistemas de vigilancia emocional.

En el desarrollo de esta propuesta, se utilizó la base de datos conocida como “FACES” en su versión de libre uso, la cual consta de un total de 72 imágenes faciales divididas en seis expresiones distintas: Enojo, Disgusto, Miedo, Felicidad, Neutralidad y Tristeza. Las imágenes fueron preprocesadas realizando un cambio de escala y convirtiéndolas a escala de grises para posteriormente efectuar la detección facial y la estimación de los puntos de referencia faciales (landmarks) y con éstos dibujar el bosquejo de las cejas, ojos y boca. Finalmente se aplicó una corrección de ángulo y una normalización para mejorar la consistencia del conjunto de datos.

El Histograma de Gradientes Orientados fue aplicado para extraer un vector de características de 81 datos, mediante el cálculo de los gradientes, la división de celdas y la construcción de histogramas de orientación de dichos gradientes. El modelo de reconocimiento utilizado fue una Red Neuronal Artificial tipo Perceptrón Multicapa, la cual se construyó con 3 capas constituidas de la siguiente manera: con un total de 81 neuronas en su capa de entrada, 163 en su capa intermedia y 6 en la capa de salida, realizando el entrenamiento de la Red Neuronal mediante el algoritmo de Retro-propagación.

Para evaluar el desempeño del modelo se utilizaron métricas de evaluación comúnmente empleadas en problemas de clasificación, tales como la exactitud, precision y la F1-score. Los resultados obtenidos fueron altamente satisfactorios, demostrando la efectividad del enfoque propuesto, donde el modelo alcanzó una exactitud del 97.22% en el reconocimiento de expresiones, además de mostrar valores elevados en las métricas calculadas, lo que respalda la precisión y la robustez del modelo.

En conclusión, el enfoque basado en los bocetos, el Histograma de Gradientes y la Red Neuronal Artificial ha demostrado ser una estrategia efectiva para la detección de emociones en imágenes faciales. Los resultados obtenidos sugieren que este sistema podría ser aplicado en diversas áreas, como interfaces interactivas más intuitivas, detección de estados emocionales en tiempo real y análisis de comportamiento humano. No obstante, es importante señalar que aún existen desafíos y oportunidades para mejorar el modelo, incluyendo la expansión del conjunto de datos y la exploración de arquitecturas de redes neuronales más complejas.

Palabras clave: Reconocimiento de expresiones; Expresiones faciales; Bosquejos; Puntos de referencia faciales.

Introducción

El reconocimiento de las expresiones faciales es un campo de investigación en constante crecimiento dentro de la visión por computadora y la inteligencia artificial. La capacidad de detectar y comprender las emociones expresadas por las personas a través de sus rostros es de gran importancia debido a su amplio rango de aplicaciones en diversas áreas, incluyendo la interacción humano-máquina, el análisis del comportamiento humano, y la detección de emociones en sistemas de vigilancia, entre otros (Shah, 2015).

Las expresiones faciales son una parte fundamental de la comunicación no verbal humana y desempeñan un papel crucial en la transmisión de emociones, estados de ánimo y actitudes (Frank, 2001). La capacidad de un sistema automatizado para reconocer y entender estas expresiones podría mejorar significativamente la interacción con dispositivos tecnológicos, permitiendo interfaces más naturales y adaptativas.

En el presente trabajo, nos enfocamos en el problema específico de la detección de emociones en imágenes faciales utilizando el descriptor denominado Histograma de Gradientes Orientados, en lo sucesivo referido como HOG, por sus siglas en inglés (Dalal, 2005), y una Red Neuronal Artificial (ANN, por sus siglas en inglés) del tipo Perceptrón Multicapa (Lek, 2008). Nuestro objetivo es desarrollar un método de solución eficiente y preciso que permita identificar emociones relevantes en imágenes de personas, lo que podría ser de gran utilidad en aplicaciones como sistemas de asistencia emocional, de análisis del comportamiento humano y la detección temprana de signos de angustia o malestar.

En investigaciones recientes, se han propuesto diversas técnicas para abordar la detección de emociones en imágenes faciales. Entre estas técnicas, destacan el uso de Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) (Caroppo, 2020), características de textura (Wang, 2019) y patrones geométricos (Suhaila, 2015). Si bien estas metodologías han mostrado resultados prometedores, presentan ciertas limitaciones en términos de complejidad computacional, requerimientos de datos de entrenamiento y generalización a diferentes condiciones de iluminación y expresiones faciales.

La razón principal que nos motiva a proponer un nuevo método basado en el descriptor HOG radica en su capacidad para capturar características locales relevantes y su relativa simplicidad computacional en comparación con otras técnicas más complejas. Además, el descriptor HOG ha demostrado ser efectivo en aplicaciones de detección de objetos y personas, lo que sugiere su potencial para la detección de patrones emocionales en rostros.

El método que proponemos en este trabajo se basa en hacer uso del preprocesamiento de imágenes para así eliminar datos irrelevantes para la detección, permitiendo dibujar un bosquejo o boceto únicamente con las zonas de interés para la detección de emociones, en este caso, las cejas, los ojos y la boca (Lekdioui, 2017). Posteriormente se realiza el cálculo del descriptor HOG para extraer un vector de características relevantes de las imágenes faciales. Por último, utilizamos una ANN tipo Perceptrón Multicapa para entrenar el modelo de detección de emociones mediante al algoritmo de retro-propagación. El conjunto de datos utilizado proviene de la base de datos FACES (Ebner, 2010), que contiene imágenes faciales etiquetadas con emociones específicas. Nuestro enfoque combina la simplicidad del dibujo de un bosquejo para después utilizar el descriptor HOG con la efectividad de la ANN tipo Perceptrón Multicapa para lograr una detección precisa y eficiente de emociones en imágenes faciales. En la Figura 1, se muestra un diagrama general del proceso.

Para llevar a cabo el proyecto se utilizó la biblioteca “OpenCV” (Bradski, 2000), la cual es una biblioteca de código abierto ampliamente utilizada en el campo de la visión por computadora y el procesamiento de imágenes. Fue desarrollada originalmente por Intel en 1999 y posteriormente mantenida por Willow Garage y después por Itseez (que fue adquirida por Intel en 2016). OpenCV proporciona una amplia gama de funciones y algoritmos que permiten a los desarrolladores realizar tareas relacionadas con la visión por computadora de manera eficiente, utilizando el lenguaje de programación C++.

A continuación, describiremos en detalle las etapas de preprocesamiento de la imagen, así como el cálculo del descriptor HOG, la metodología de entrenamiento y evaluación del clasificador, además de los resultados obtenidos y su discusión. Esperamos que este estudio contribuya a una mejor comprensión del campo del reconocimiento de expresiones faciales y su posible aplicación en diversas áreas de la tecnología y el análisis humano.

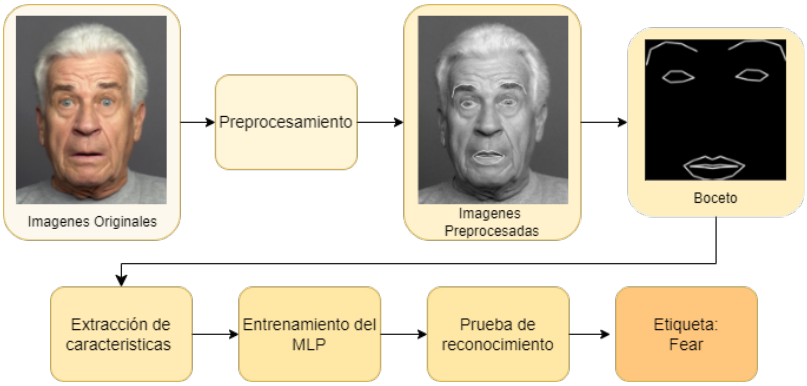

Figura 1. Diagrama general del proceso.

Metodología

El primer paso del proyecto consistió en realizar una búsqueda de una base de imágenes de referencia que sirviera como estándar para nuestro trabajo. Dentro de las diferentes opciones encontradas se optó por utilizar la base de datos FACES, la cual ofrece un total de 72 imágenes faciales de libre acceso. Estas imágenes están etiquetadas con seis categorías emocionales distintas: Enojo, Disgusto, Miedo, Felicidad, Neutralidad y Tristeza (Anger, Disgust, Fear, Happiness, Neutrality y Sadness). Cada una de estas categorías cuenta con 12 imágenes, lo que resulta en un conjunto de datos equilibrado para el análisis y entrenamiento de nuestro modelo.

La base de datos FACES es una valiosa fuente de imágenes que representa una variedad de expresiones emocionales humanas, lo que la convierte en una elección adecuada para desarrollar y evaluar nuestro sistema de detección de emociones en imágenes faciales. Una vez obtenido el conjunto de imágenes FACES de uso libre, se llevó a cabo el reacomodo de las imágenes, creando una carpeta conteniendo en su interior subcarpetas con el nombre de cada una de las emociones, y a su vez, dichas carpetas conteniendo 12 imágenes de su respectiva emoción. Con el conjunto de datos FACES adquirido y organizado, podemos proceder con nuestra metodología.

A continuación, se describen detalladamente cada una de las etapas del proyecto, el cual, después de un pre-procesamiento de cada imagen, está dividido en 3 etapas principales: la preparación de muestras, el entrenamiento del perceptrón multicapa y la prueba de reconocimiento, como se muestra en la Figura 1.

Etapa 1. Preparación de muestras

De manera general, en la Figura 2 podemos observar el diagrama a bloques que muestra los pasos a seguir para obtener el conjunto de características que servirán como entrada al clasificador. Esto con el fin de poder entrenar al clasificador para reconocer la expresión del rostro exhibido en una imagen.

Lectura de las imágenes

Para poder manipular las imágenes, es necesario leerlas de tal manera que el programa las reconozca como matrices de datos. Esto se logra mediante el uso de la función "`imread()`" que forma parte de la biblioteca de OpenCV. La función `imread()` se utiliza para leer imágenes desde un archivo en el disco y cargarlas en la memoria de un programa. Esta función trabaja de la siguiente manera:

- Abre el archivo de imagen: La función `imread()` toma la ruta del archivo de imagen como entrada y abre el archivo correspondiente para su lectura.
- Verificación del archivo: Una vez que se abre el archivo, la función `imread()` verifica si el archivo existe y si se puede acceder a él. Si el archivo no se encuentra o no se puede abrir, la función retorna un error.

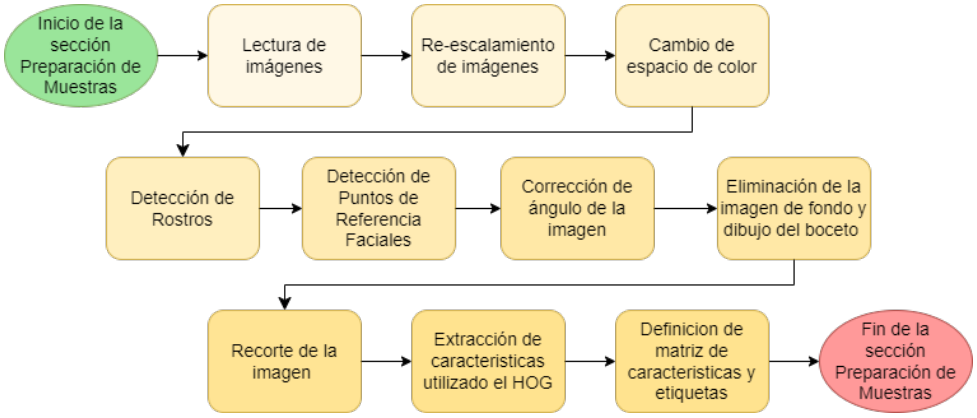

Figura 2. Diagrama de bloques que ilustra el procedimiento hecho dentro de la etapa de preparación de las muestras.

Este proceso se llevará a cabo para cada una de las 72 imágenes de muestras mediante el uso de una estructura de control iterativa, la cual nos permite ejecutar de manera repetitiva un bloque de código.

Escalamiento de imágenes

El escalado se hace mediante el uso de la función "`resize()`" de OpenCV. Reescalar imágenes tiene diversas utilidades. Se utiliza para normalizar las imágenes, es decir, para establecer un tamaño estándar en todas las imágenes con las que se trabajará. En el caso específico de la biblioteca FACES, las imágenes proporcionadas ya tienen un tamaño uniforme de 2835 x 3543 píxeles, sin embargo, el tamaño de las imágenes proporcionadas es demasiado grande para manipularlas eficientemente. El objetivo es reducir el tamaño de las imágenes, pero mantenerlas lo suficientemente grandes como para no perder la cara y sus referentes faciales. Encontramos útil el dividir el tamaño original de la imagen en 10, ya que este permite que el programa trabaje de manera más rápida en comparación que el tamaño original y además conserva las características de la imagen que nos permiten identificar correctamente los 68 referentes faciales.

Cambio de espacio de color

Las imágenes de la base de datos están en el espacio de color RGB, pero el color es despreciable para la detección de caras y, por ende, de las emociones; por lo tanto, se realiza la conversión a escala de grises. La escala de grises proporciona beneficios tales como:

- Simplificación de los datos: reduce la complejidad de los datos y simplifica el procesamiento de la imagen.
- Eficiencia computacional: Las imágenes en escala de grises requieren menos recursos computacionales para su manipulación en comparación con las imágenes en color.
- Características visuales: En este caso, el análisis de la intensidad de brillo es suficiente para extraer características o realizar operaciones específicas en la imagen.

Este cambio de espacio de color se realiza por medio de la función "`cvtColor()`" de OpenCV. Esta función permite cambiar entre distintos espacios de color, por lo cual es necesario especificar la imagen que queremos pasar del espacio RGB a escalas de grises por medio del parámetro "`COLOR_BGR2GRAY`".

Detección de Rostros

El siguiente paso fue hacer la detección de rostros, esto se logró mediante el uso una técnica de detección y localización de objetos conocida como clasificador en cascada, más específicamente, se usó un clasificador en cascada pre-entrenado, el cual es un archivo que lleva el nombre de "`haarcascade_frontalface_alt2.xml`"

El clasificador en cascada está compuesto por múltiples etapas, y cada etapa está compuesta por clasificadores débiles basados en características visuales. El proceso implica desplazar una ventana rectangular sobre la imagen en diferentes escalas y ubicaciones, y en cada posición, se calcula un conjunto de características. Estas características se comparan con los clasificadores débiles en cascada definidos en un archivo XML para determinar si la región de la ventana contiene un rostro o no. Ya con los rostros detectados, se identifican los referentes faciales.

Detección de Puntos de Referencia Faciales

Los referentes faciales (comúnmente referidos, por su término en inglés, como *landmarks*) son puntos específicos y distintivos ubicados en la cara de una persona que sirven para identificar y describir características faciales particulares. Estos puntos faciales pueden incluir características como los ojos, las cejas, la nariz, los labios, la barbilla, entre otros. El número y la ubicación exacta de estos puntos pueden variar dependiendo de la aplicación o el modelo específico utilizado, en nuestro caso utilizamos el modelo LBF (Local Binary Features) por medio del archivo "`lbfmodel.yaml`" que contiene los parámetros y datos necesarios para cargar y utilizar el modelo LBF en una aplicación o código.

El modelo LBF se basa en el enfoque de características binarias locales para la detección de puntos faciales. En lugar de utilizar características basadas en intensidad de píxeles, utiliza características binarias para describir las relaciones espaciales locales entre los píxeles. Esto permite una detección rápida y precisa de los referentes faciales. Como se puede apreciar en la Figura 3, este modelo cuenta con un total de 68 referentes faciales que abarcan las zonas de:

- 0 – 16 contorno facial
- 17 – 21 ceja derecha
- 22 – 26 ceja izquierda
- 27 – 35 nariz
- 36 – 41 ojo derecho
- 42 – 47 ojo izquierdo
- 48 – 67 Labios

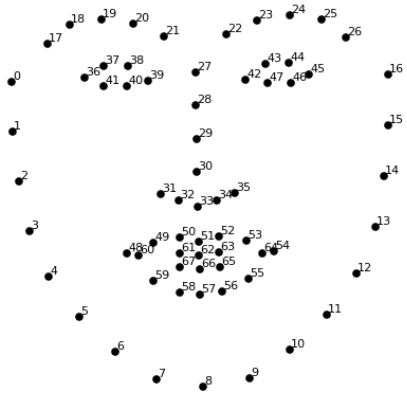

Figura 3. Diagrama que muestra la ubicación de los 68 referentes faciales.

Corrección de la inclinación de la cara

Se realizó una corrección de ángulo, esto con el propósito de que los referentes faciales de los puntos 0 y 16, en los que se identifican los puntos superiores del contorno facial, queden alineados horizontalmente. Ello permite que las muestras estén normalizadas, en caso de que se quiera utilizar el modelo con imágenes externas a la base de datos FACES, además de mejorar la exactitud de la ANN.

Eliminación de la imagen de fondo y dibujo del bosquejo

Una vez identificados los puntos de referencia faciales, el resto de la imagen se considera irrelevante y, con el objetivo de optimizar el proceso, se elimina dejando una imagen completamente negra. Sobre esta base, utilizando los referentes faciales identificados, se procederá a dibujar un bosquejo que ilustra el rostro del

sujeto y, por ende, su expresión. El bosquejo se obtiene dibujando líneas que unen los puntos de los referentes faciales, por medio de un función que dibuja las líneas de cada elemento facial, por lo cual es necesario utilizar la función tantas veces como elementos haya por dibujar. En este caso se buscó dibujar ambas cejas y ojos, así como la parte interior y exterior de los labios, dejando sin dibujar las zonas correspondientes a la nariz y al contorno facial, los cuales no aportan información relevante al reconocimiento de emociones.

Recorte de la imagen y normalización en base a los puntos de referencia faciales con corrección de ángulo

Con los referentes faciales corregidos, se encuentran los valores máximos y mínimos en los ejes X y Y, lo que permitirá realizar un recorte de la imagen específicamente a donde se encuentra el boceto, eliminando zonas completamente negras que no contienen datos y que pueden generar ruido y confusión a la hora de extraer las características, generando así un rendimiento más bajo en las etapas de reconocimiento. Por último, se realiza nuevamente un escalado de la imagen para así mantener la normalización de los datos, a un tamaño de 128x128 píxeles. Finalmente, en la Figura 4, se muestra el resultado de cada uno de los pasos descritos anteriormente.

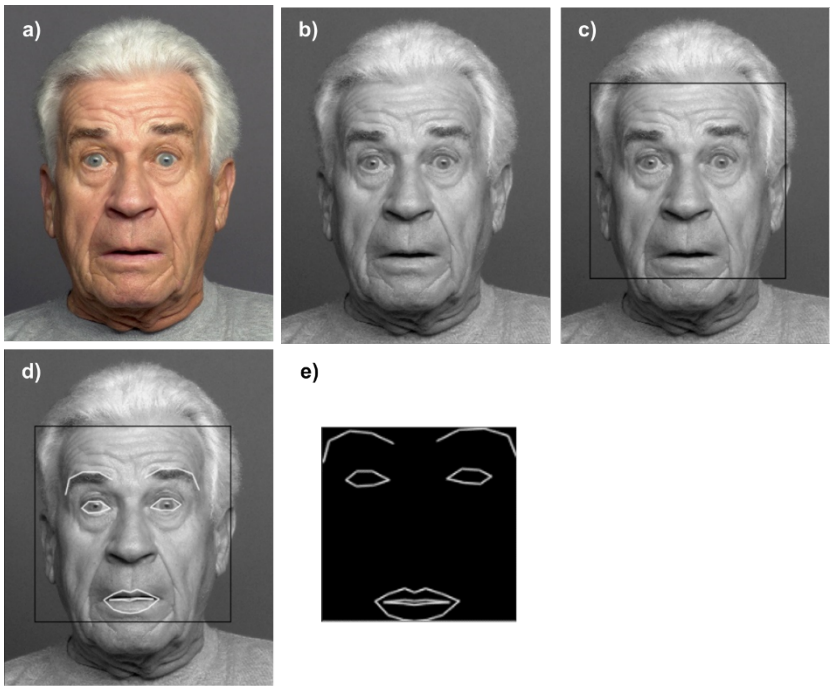

Figura 4. Procesamiento de la imagen a) Imagen original, b) Cambio de espacio de color, c) Detección de rostros d) Detección de Puntos de Referencia Faciales e) Bosquejo recortado y normalizado.

Descripción de los parámetros del descriptor HOG

El descriptor de HOG, es una técnica utilizada para la detección de objetos en imágenes muy popular en aplicaciones de visión por computadora. El funcionamiento del descriptor de HOG se puede resumir en los siguientes pasos:

- Preprocesamiento: Se convierte la imagen de entrada en escala de grises y se realiza cualquier preprocesamiento adicional necesario, como normalización de contraste o ajuste de tamaño.

- Cálculo de gradientes: Se calcula el gradiente de la imagen para obtener información sobre los cambios de intensidad en diferentes direcciones. Esto se puede hacer aplicando filtros de derivadas, como los filtros de Sobel.
- División en celdas: La imagen se divide en regiones más pequeñas llamadas celdas. Para cada celda, se calculan los histogramas de orientaciones de gradientes locales.
- Construcción del histograma: Se calcula un histograma de orientaciones de gradientes para cada celda. Esto implica agrupar las direcciones del gradiente en intervalos (por ejemplo, 9 intervalos de 20 grados) y contar la frecuencia de cada orientación en la celda.
- Normalización: Los histogramas de las celdas se normalizan para tener en cuenta las variaciones locales de iluminación. Esto se puede hacer mediante técnicas como la normalización de contraste o la normalización L2.
- Construcción del descriptor: El descriptor de HOG se construye concatenando los histogramas de todas las celdas de la imagen. Esto proporciona una representación compacta de la imagen que captura la información de la distribución espacial de los gradientes.

Para poder utilizar el HOG, es necesario definir una estructura en la que se definan los parámetros con los cuales se aplicará este descriptor, para este proyecto utilizamos los siguientes parámetros:

- El tamaño de la ventana deslizante (winSize) que se utilizará para extraer las características HOG de la imagen. Esta ventana se desliza por toda la imagen para calcular los histogramas de gradientes en cada posición: [`Size(128, 128)`].
- El tamaño del bloque (blockSize) utilizado en la normalización de los histogramas de gradientes. Los bloques son regiones rectangulares dentro de cada celda y se utilizan para mejorar la invarianza a la iluminación: [`Size(64, 64)`].
- El desplazamiento del bloque (blockStride) entre celdas adyacentes. Indica cuántos píxeles se desplaza el bloque para calcular los siguientes histogramas de gradientes: [`Size(32, 32)`].
- El tamaño de la celda (cellSize) utilizado para calcular los histogramas de orientaciones de gradientes locales. Cada celda es una región rectangular que contiene varios píxeles: [`Size(64, 64)`].
- El número de compartimentos (*bins*) que se utilizará para dividir el rango de orientaciones de gradientes. Cada *bin* representa una dirección específica de gradiente: nbins [9].
- El parámetro derivAper indica el factor de amplificación para calcular los derivados de la imagen: [`1`].
- El valor de winSigma, que controla el nivel de suavizado de la imagen antes de calcular los gradientes. Un valor negativo indica que se utilizará el valor predeterminado: [`-1`].
- El tipo de normalización del histograma (histogramNormType) utilizado en el descriptor de HOG. En este caso, se utiliza L2-Hys, que es una normalización combinada L2 y truncada: [`HOGDescriptor::HistogramNormType::L2Hys`].
- El umbral utilizado en la normalización L2-Hys controla la normalización truncada de los valores del histograma para evitar que los gradientes dominantes dominen los demás: [`0.2`].
- El valor de corrección gamma aplicado a la imagen antes de calcular los gradientes. Un valor de 0 indica que no se realizará ninguna corrección gamma: [`0`].
- El número de niveles (nlevels) utilizados para la representación piramidal de la imagen. La pirámide se utiliza para escalar la imagen y calcular los histogramas de gradientes en diferentes resoluciones: [`64`].
- El valor de signedGradient que indica si se utilizan gradientes con signo (dirección) o solo magnitud. Un valor de 1 indica que no se utilizarán gradientes con signo, lo que resulta en orientaciones no direcciones: [`1`].

Una vez que se han definido los parámetros del descriptor es posible calcular el vector de características que posterioemente será utilizao en lugar de la imagen y en lugar del bosquejo de la cara. Para realizar este cálculo es necesario haber definido un vector que almacene dichas características. Posteriormente se hace uso del método "`compute`" perteneciente a la clase "`HOGDescriptor`" de la biblioteca OpenCV para obtener las características de la imagen que contiene el boceto. Estas características son luego escritas en el vector previamente definido. El vector que se obtuvo por medio de estos parámetros resultó en un total de 81 características.

Definición de la matriz de características y etiquetas

En el primer paso se describió el proceso de organización de muestras en distintas carpetas dependiendo de la expresión que posee el sujeto, esto se hizo para facilitar el proceso de etiquetado que se realiza durante este paso. Así que dependiendo de a qué carpeta pertenezca la imagen, se le asignará la etiqueta correspondiente.

Al momento de terminar de leer el total de muestras se tendrán dos matrices, una matriz de características de 72 renglones y 81 columnas y una matriz de etiquetas con 72 renglones y una columna. Es necesario combinar estas dos matrices de forma que tengamos una matriz de 82 columnas y 72 renglones en las que se tenga tanto la información de las características como la etiqueta a las que corresponden dichas características. Esto se logra haciendo uso de la función "`hconcat()`" de OpenCV, la cual recibe las matrices que se desean concatenar, así como la matriz en la que se guardará dicha concatenación.

Para que el entrenamiento se realice correctamente es común realizar un reacomodo aleatorio de las muestras, utilizando la función "`shuffleRows()`" generando una nueva matriz de características y etiquetas ordenadas de forma aleatoria.

En pro de evaluar el desempeño de nuestro programa decidimos hacer uso del método "N-Fold Cross-Validation (validación cruzada con N particiones)" con un total de 6 folds o particiones, por lo cual debemos dividir las muestras de forma que un sexto de ellas sean utilizadas para las pruebas y las muestras restantes sean usadas para el entrenamiento. Se optó usar un 6-Fold principalmente debido a que 6 particiones es un número considerable para poder evaluar el desempeño, además de que la cantidad de muestras de la base de datos FACES es múltiplo de 6, así como 6 es la cantidad de expresiones a identificar.

Etapa 2. Entrenamiento del Perceptrón Multicapa

La siguiente etapa, es el entrenamiento de la ANN tipo Perceptrón Multicapa, la cual será la responsable de clasificar las imágenes en las distintas clases, pero primero se debe llevar a cabo un entrenamiento supervisado, alimentando con muestras y etiquetas, para configurar los pesos en cada neurona de la red. En la Figura 5 podemos ver un diagrama a bloques general de esta etapa.

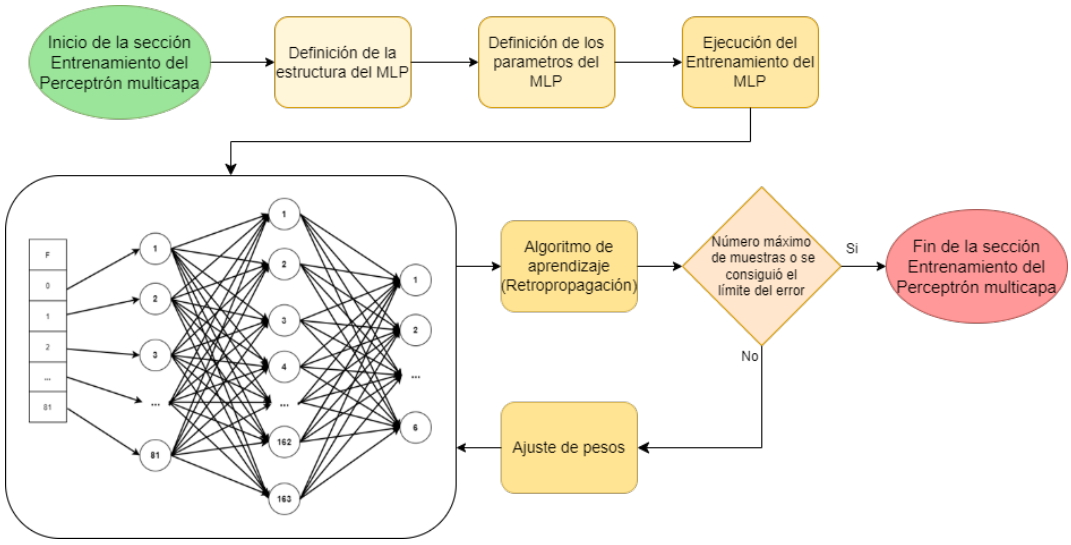

Figura 5. Diagrama de bloques que ilustra el procedimiento hecho dentro de la etapa de Entrenamiento del Perceptrón Multicapa.

Definición de los parámetros del Perceptrón Multicapa

Un Perceptrón Multicapa (*multilayer Perceptron*, MLP), es un tipo de arquitectura de una ANN, que consiste en múltiples capas de neuronas unidades e interconectadas. Es una extensión del perceptrón básico, que es una ANN de una sola capa.

El MLP está compuesto por tres tipos de capas:

- Capa de entrada: Esta es la primera capa de la red y corresponde a la entrada de datos al modelo. Cada neurona en esta capa representa una característica o atributo de los datos de entrada.
- Capas ocultas: Entre la capa de entrada y la capa de salida, puede haber una o más capas ocultas. Cada capa oculta está compuesta por un número de neuronas (nodos) que se conectan con todas las neuronas de la capa anterior y la siguiente. La cantidad de capas ocultas y el número de neuronas en cada capa son parámetros ajustables del modelo y son determinados durante el proceso de diseño y entrenamiento.
- Capa de salida: Esta es la última capa de la red y produce los resultados o predicciones del modelo. El número de neuronas en esta capa depende del tipo de problema que se esté resolviendo. Por ejemplo, en un problema de clasificación, el número de neuronas en la capa de salida corresponderá al número de clases o categorías a predecir.

Para esta aplicación decidimos utilizar un total de neuronas en la capa de entrada igual al número de características dentro del vector (81); una única capa oculta, con el doble más uno de este mismo número (163) y, finalmente, la capa de salida contiene una neurona por cada etiqueta (6)

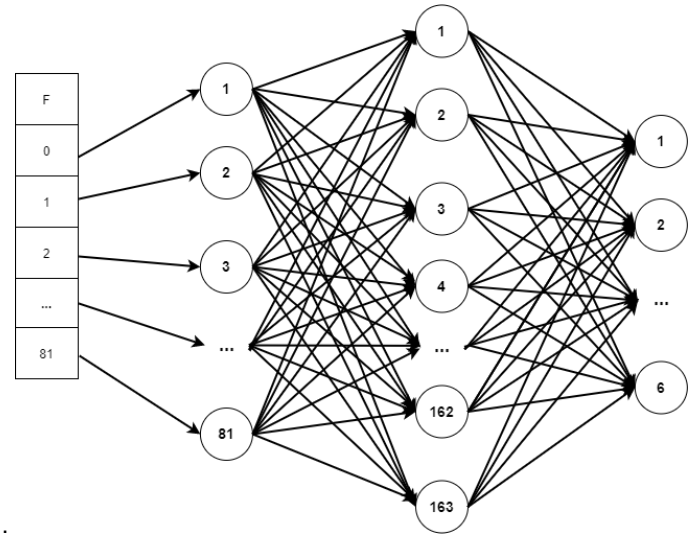

Figura 6. Representación de la estructura de la Red Neuronal Artificial.

Además del número de neuronas por cada capa, los otros parámetros que se deben de definir son:

- Función de Activación: La función de activación es una función matemática que se aplica a la salida de una neurona en una red neuronal. Su objetivo es introducir no linealidad en el modelo, lo que permite a la red aprender y representar funciones complejas que no se pueden aproximar con funciones lineales. Para este proyecto se hizo uso de la función de activación sigmoidal simétrica (`SIGMOID_SYM`)
- Criterio de Terminación: El criterio de terminación es una condición que se utiliza para determinar cuándo detener el proceso de entrenamiento de un modelo de aprendizaje automático. Se utilizaron los criterios de detención del entrenamiento una vez que alcance las `900` épocas o cuando la precisión alcanza una tolerancia de `0.0001`.

- Método de Entrenamiento: El método de entrenamiento se refiere al algoritmo utilizado para ajustar los pesos de una red neuronal durante el proceso de entrenamiento. El objetivo del entrenamiento es minimizar la función de pérdida o error para hacer que las predicciones del modelo se ajusten lo mejor posible a los valores reales del conjunto de entrenamiento. El método de entrenamiento del MLP que se decidió fue el método de retropropagación (`BACKPROP`).

El entrenamiento de la red neuronal se realiza mediante la función "`train()`" de un objeto ann que representa al MLP de la biblioteca de aprendizaje automático de OpenCV (`cv::ml::ANN_MLP`). Esta función se utiliza para entrenar el MLP utilizando los datos de entrenamiento y las etiquetas asociadas. A esta función de ejecución se le asignan tres parámetros, los cuales corresponden a la matriz de características para entrenamiento, un parámetro que describe que los datos de la matriz están organizados en filas, de forma que cada fila represente a una muestra y por último una matriz de etiquetas modificada, en la que se utiliza la codificación "one hot" para representar la clase a la que pertenece cada muestra.

Una vez terminado el entrenamiento se guardan los pesos asignados a cada neurona del MLP en un archivo .yml por medio de la función "`save()`" del mismo objeto ann que se ha utilizado. Guardar el modelo nos permite evitar tener que entrenar el MLP cada que se ejecuta el programa y además disponer de un modelo con el cual comparar otros modelos.

Etapa 3. Prueba de reconocimiento

En esta última etapa se realiza la evaluación del desempeño del modelo entrenado. En la Figura 7, se puede observar el diagrama a bloques del procedimiento.

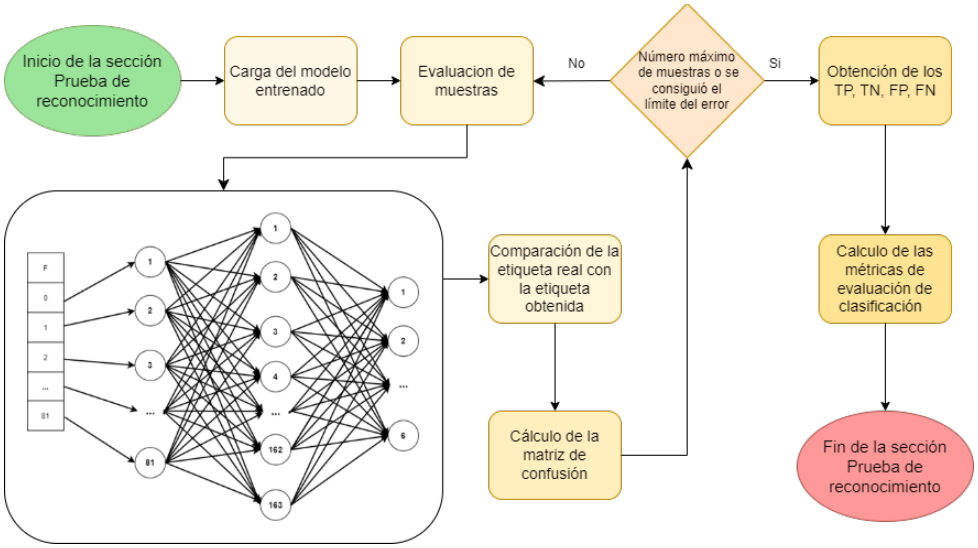

Figura 7. Diagrama de bloques que ilustra el procedimiento hecho dentro de la etapa de Prueba del Entrenamiento.

Evaluación de muestras

Una vez que el MLP ha sido entrenado correctamente, se procede a realizar pruebas evaluando las características de las muestras previamente almacenadas en el archivo de Valores Separados por Comas (CSV), por sus siglas en inglés, de Prueba. Para cada muestra, se obtiene una predicción de su etiqueta utilizando la función "predict()" de un objeto ANN de la biblioteca de aprendizaje automático de OpenCV. Posteriormente, se compara esta predicción con la etiqueta real almacenada en el archivo CSV de etiquetas de muestras para Prueba.

Este proceso se lleva a cabo en un ciclo for que permite evaluar las 12 muestras de prueba. La predicción obtenida se almacena en una variable, lo que facilitará su posterior verificación en una matriz de confusión.

Adicionalmente, también se guarda el valor real de la etiqueta obtenido del archivo CSV en otra variable para completar la matriz de confusión.

Cálculo de la matriz de confusión

Una matriz de confusión es una herramienta cuyo objetivo es evaluar el rendimiento de un modelo de clasificación al comparar las predicciones del modelo con los valores reales o verdaderos de las muestras. Para crear la matriz de confusión primero debe crearse una matriz cuadrada vacía de igual tamaño al número de clases, en este caso una matriz de 6 x 6. La matriz se rellena mediante comparaciones entre los valores predichos y los valores reales, los cuales son utilizados como coordenadas para registrar los resultados. Cuando la predicción coincide con el valor real, el contador en la coordenada correspondiente aumenta en uno, siendo esta una coordenada posicionada en la diagonal. Por otro lado, cuando la predicción no coincide, el contador en la coordenada (valor real, predicción) se incrementa.

Obtención de los valores TP, TN, FP y FN

Así como la matriz de confusión, los valores TP, TN, FP y FN (usaremos los acrónimos en inglés, los que se detallarán a continuación) nos permiten evaluar el rendimiento de un modelo de clasificación y medir su capacidad para predecir correctamente las distintas clases o categorías del problema.

- Los valores TP (True Positive): Representan el número de muestras que fueron correctamente clasificadas como positivas para una clase específica. Es decir, el modelo predijo correctamente que pertenecen a la clase en cuestión.
- Los valores TN (True Negative): Indican el número de muestras correctamente clasificadas como negativas para la clase en cuestión. Es decir, el modelo predijo correctamente que no pertenecen a la clase en cuestión.
- Los valores FP (False Positive): Muestra el número de muestras que fueron incorrectamente clasificadas como positivas para una clase específica. Es decir, el modelo predijo que pertenecen a la clase en cuestión, pero en realidad no es así.
- Los valores FN (False Negative): Indica el número de muestras que fueron incorrectamente clasificadas como negativas para una clase específica. Es decir, el modelo predijo que no pertenecen a la clase en cuestión, pero en realidad sí pertenecen.

Estos valores se calculan dentro del programa mediante la comparación de los valores reales y los predichos por el MLP, pero también pueden ser fácilmente calculados a mano con la ayuda de la matriz de confusión.

Cálculo de las métricas de evaluación de clasificación

Las métricas de evaluación de clasificación son medidas utilizadas para evaluar el rendimiento y la precisión de un modelo de clasificación en tareas donde se deben asignar etiquetas o categorías a los datos de entrada. Estas métricas proporcionan información sobre cómo el modelo se comporta al realizar predicciones en un conjunto de datos de prueba y ayudan a comprender cómo está clasificando correcta o incorrectamente las muestras. Las métricas de evaluación de clasificación utilizadas para evaluar el desempeño de nuestro modelo son las siguientes:

- Exactitud (*Accuracy*): Mide la proporción de predicciones correctas realizadas por el modelo en relación con el total de muestras en el conjunto de prueba. Es una métrica generalmente utilizada cuando todas las clases tienen aproximadamente el mismo número de muestras y se busca una visión general del rendimiento del modelo.

$$Acc = \frac{N_{TP} + N_{TN}}{N}$$

- Tasa de error (*Error Rate*) es una métrica de evaluación de clasificación que mide la proporción de predicciones incorrectas realizadas por un modelo en relación con el total de muestras en el conjunto de prueba.

$$E = \frac{N_{FP} + N_{FN}}{N}$$

- Precisión (*Precision*): Mide la proporción de predicciones positivas que son correctas en relación con todas las predicciones positivas realizadas por el modelo. Es útil cuando el costo de los falsos positivos es alto y se busca minimizar este tipo de error.

$$Pr = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

- Exhaustividad (*Recall*): Mide la proporción de muestras positivas que son correctamente identificadas por el modelo. Es útil cuando el costo de los falsos negativos es alto y se busca minimizar este tipo de error.

$$Re = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

- Puntuación F1 (*F1-Score*): Es una medida que combina tanto la precisión como la sensibilidad en una sola métrica. Es útil cuando se necesita un equilibrio entre ambas métricas y se busca una métrica única que resuma el rendimiento del modelo.

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re}$$

Resultados obtenidos

En esta sección, se presentan los resultados obtenidos mediante la implementación de nuestro modelo de clasificación de expresiones faciales utilizando los bosquejos, el descriptor de HOG y la ANN tipo MLP. Para evaluar el rendimiento del modelo, se utilizaron una serie de métricas de evaluación comúnmente empleadas en problemas de clasificación, tales como una matriz de confusión y demás parámetros derivados de ésta, donde el propósito es poder visualizar con facilidad si los bocetos y las características generadas por el descriptor de HOG funcionan de manera adecuada para el entrenamiento y la clasificación, o si existe alguna expresión que genere confusión en la clasificación.

A continuación, se muestran los resultados obtenidos de las seis particiones generadas con el método N-Fold Cross-Validation, en el cual se van rotando 60 muestras de entrenamiento con 12 muestras de prueba por toda la base de datos. En la Figura 8 se puede observar la matriz de confusión resultante de la suma de las 6 matrices generadas por las particiones, el cálculo de la exactitud (*accuracy*) de la esta matriz resultante es de 0.9722 o del 97.22%.

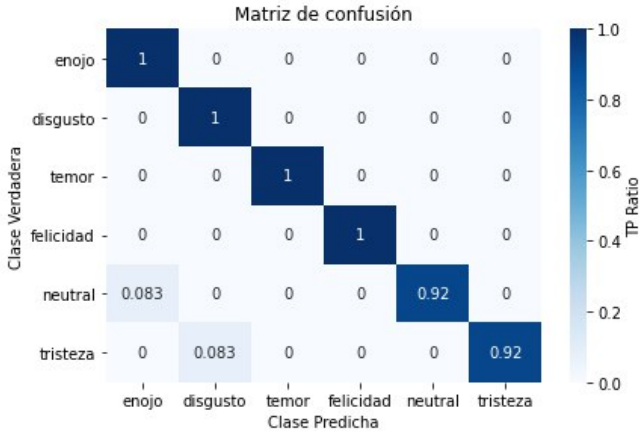

Figura 8. Matriz de confusión resultante.

Como se puede observar, el modelo obtenido genera muy buenos resultados de clasificación, en el cual solo existe una pequeña confusión en un pequeño número de muestras entre las expresiones neutral y enojo, así como tristeza y disgusto. En base a la matriz de confusión obtenida y a su exactitud, se puede decir que el resultado de la clasificación fue exitoso, ya que el modelo permite clasificar las seis diversas expresiones faciales.

En la Tabla 1, se pueden observar otras métricas clave obtenidas de la matriz de confusión de la Figura 8; dichas métricas posibilitan evaluar la capacidad del modelo para clasificar correctamente cada categoría de expresiones.

Tabla 1. *Métricas de evaluación de clasificación del modelo propuesto.*

Muestras	TP	TN	FP	FN	Exactitud	Tasa de Error	Precisión	Exhaustividad	F1-Score
Enojo	12	59	1	0	0.9861	0.0139	0.9231	1	1
Disgusto	12	59	1	0	0.9861	0.0139	0.9231	1	0.9600
Miedo	12	60	0	0	1	0	1	1	1
Felicidad	12	60	0	0	1	0	1	1	1
Neutral	11	60	0	1	0.9861	0.0139	1	0.9167	1
Tristeza	11	60	0	1	0.9861	0.0139	1	0.9167	0.9565

La exactitud (*accuracy*) mide el porcentaje de casos que el modelo ha acertado. La precisión (precision) permite medir la calidad del modelo en la clasificación (¿qué porcentaje de los que hemos dicho que son la clase positiva, en realidad lo son?), mientras que la exhaustividad (*recall*) permite conocer sobre la cantidad que el modelo es capaz de identificar (¿qué porcentaje de la clase positiva hemos sido capaces de identificar?). El F1 score se utiliza para combinar las medidas de precisión y exhaustividad en un solo valor, esto para poder comparar el rendimiento combinado de ambas métricas entre varias soluciones.

Cabe destacar que nuestro modelo ha demostrado un alto rendimiento en la clasificación de expresiones faciales, con valores elevados de precisión, exhaustividad y *F1-Score* para cada categoría. Resaltando que solo se generaron 2 falsos negativos (FN) en las particiones 5 y 6 respectivamente.

Conclusiones

En este estudio, hemos abordado el desafiante problema del reconocimiento de expresiones en imágenes faciales utilizando un enfoque basado en la creación de bocetos, para posteriormente extraer sus características con el descriptor HOG y realizar el reconocimiento con una red neuronal artificial de Perceptrón multicapa. A lo largo del proyecto, hemos logrado avances significativos en la identificación y clasificación de expresiones emocionales, lo que tiene implicaciones importantes en el campo de la visión por computadora y la inteligencia artificial. Nuestro modelo ha mostrado un alto rendimiento en la detección de emociones, alcanzando una exactitud del 97.22% en la clasificación de emociones en el conjunto de prueba. Además, al evaluar las métricas adicionales como Precisión, Exhaustividad y *F1-Score* para cada categoría emocional, hemos observado valores alentadores que respaldan la precisión y robustez del sistema.

La utilización de bocetos ha demostrado ser una poderosa herramienta de preprocesamiento, permitiendo resaltar las características más relevantes de una expresión facial, y gracias a esto representar de manera efectiva los patrones emocionales presentes en los rostros. Por otro lado, el descriptor HOG y la red neuronal artificial han demostrado su capacidad para realizar clasificaciones precisas en este contexto. La aplicación de nuestro sistema puede tener un impacto significativo en diversas áreas, como la interacción humano-máquina, donde las interfaces podrían adaptarse y responder a las emociones del usuario para proporcionar experiencias más personalizadas y empáticas. Además, en el análisis del comportamiento humano, esta tecnología podría ayudar a identificar signos tempranos de angustia o malestar en individuos.

Bibliografía/Referencias

(Bradski, 2000) Bradski, G., (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 4.

(Caroppo, 2020) Caroppo, A., Leone, A. and Siciliano, P. (2020). Comparison Between Deep Learning Models and Traditional Machine Learning Approaches for Facial Expression Recognition in Ageing Adults. Journal of Computer Science and Technology, 35, p. 1127-1146, https://doi.org/10.1007/s11390-020-9665-4

(Dalal, 2005) Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pp. 886-893 vol. 1, https://doi.org/10.1109/CVPR.2005.177

(Ebner, 2010) Ebner, N., Riediger, M., & Lindenberger, U. (2010). FACES—A database of facial expressions in young, middle-aged, and older women and men: Development and validation. Behavior research Methods, 42(1), 351-362. https://doi.org/10.3758/BRM.42.1.351

(Frank, 2001) Frank, M.G. (2001). Facial Expressions. International Encyclopedia of the Social & Behavioral Sciences, 5230–5234. https://doi.org/10.1016/b0-08-043076-7/01713-7

(Lekdioui, 2017) Lekdioui, K., Ruichek, Y., Messoussi, R., Chaabi, Y. and Touahni, R. (2017). Facial expression recognition using face-regions, International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), p.1-6, https://doi.org/10.1109/ATSIP.2017.8075517

(Lek, 2008) Lek, S. and Park, Y.S. (2008). Multilayer Perceptron, Encyclopedia of Ecology, Academic Press, p. 2455-2462, ISBN 9780080454054, https://doi.org/10.1016/B978-008045405-4.00162-2

(Shah, 2015) Shah, S. K. and Khanna, V. (2015). Facial Expression Recognition for Color Images using Gabor, Log Gabor Filters and PCA. International Journal of Computer Applications, 113(4), p. 42-46. https://doi.org/10.5120/19818-1642

(Suhaila, 2015) Suhaila N. M. and Loay E. G. (2015). Subject Independent Facial Emotion Classification Using Geometric Based Features. Research Journal of Applied Sciences, Engineering and Technology, 11. https://doi.org/10.19026/rjaset.11.2145

(Wang, 2019) Wang, Y., Li, M., Zhang C., Chen, H. and Lu, Y. (2019). Weighted-fusion feature of MB-LBPUH and HOG for facial expression recognition. Soft Computing, 24,p. 5859-5875, https://doi.org/10.1007/s00500-019-04380-x