UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

"IMPLEMENTACIÓN DE ALGORITMO DE DETECCIÓN Y SEGMENTACIÓN DE ROSTROS EN FOTOGRAFÍAS MEDIANTE PROCESAMIENTO DIGITAL DE SEÑALES."

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTOR:

SORAYA PAOLA OROZCO CALVA.

DIRECTOR:

ING. MARCELO FERNANDO VALDIVIEZO CONDOLO MG.SC.

LOJA – ECUADOR
2019

CERTIFICACIÓN

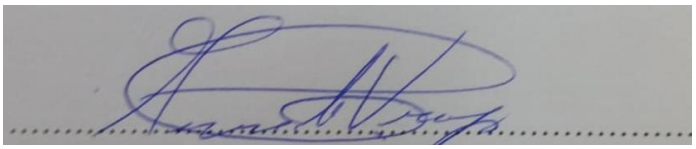
Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.

DIRECTOR DE TESIS

CERTIFICA:

Haber dirigido, asesorado, revisado y corregido el presente trabajo de tesis, en su proceso de investigación cuyo tema versa en "**IMPLEMENTACIÓN DE ALGORITMO DE DETECCIÓN Y SEGMENTACIÓN DE ROSTROS EN FOTOGRAFÍAS MEDIANTE PROCESAMIENTO DIGITAL DE SEÑALES**", previa a la obtención del título de **Ingeniera en Electrónica y Telecomunicaciones**, realizado por la señorita: **Soraya Paola Orozco Calva** , la misma que cumple con la reglamentación y políticas de investigación, por lo que autorizo su presentación y posterior sustentación y defensa.

Loja, 22 de agosto de 2019



Ing. Marcelo Fernando Valdiviezo Condolo, Mg. Sc.
DIRECTOR DEL TRABAJO DE TESIS

AUTORÍA

SORAYA PAOLA OROZCO CALVA, declaro ser la autora del presente trabajo de titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional – Biblioteca Virtual.

Firma: 

Cedula: 1150375226

Fecha: 24 de octubre de 2019

CARTA DE AUTORIZACIÓN

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

SORAYA PAOLA OROZCO CALVA, declaro ser autora de la tesis titulada: "**IMPLEMENTACIÓN DE ALGORITMO DE DETECCIÓN Y SEGEMENTACIÓN EN FOTOGRAFÍAS MEDIANTE PROCESAMIENTO DIGITAL DE SEÑALES**" como requisito para optar al grado de: **INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veinte y tres días del mes de octubre del dos mil diecinueve.

Firma: 

Autora: Soraya Paola Orozco Calva

Cédula: 1150375226

Dirección: Loja, (Motupe) **Correo Electrónico:** sporozcoc@unl.edu.ec

Teléfono: 072542628 **Celular:** 0984311090

Director de Tesis: Ing. Marcelo Fernando Valdiviezo Condolo, Mg.Sc.

Tribunal de Grado: Ing. Manuel Augusto Pesantez González, Mg.Sc.

Ing. Marco Augusto Suing Ochoa, Mg. Sc.

Ing. Ángel José Ordóñez Mendieta, Mg. Sc.

DEDICATORIA

A mi madre, Zenelia, que con su amor y consejos me ayudó en cada etapa de mi vida. A mi padre, Luis Alberto, por su apoyo constante en mis estudios.

A mi hermano, Luis Fernando, por su cariño y apoyo en todo momento.

A mi familia por acompañarme siempre y ser mi inspiración para cumplir cada una de mis metas.

A mis amigas de colegio por todas las alegrías compartidas y amigos de universidad por su apoyo en el transcurso de mi vida universitaria.

Soraya Paola

AGRADECIMIENTO

A Dios por bendecirme con una familia maravillosa y siempre ser la luz que guía mi camino. A mis padres por todo su amor y dedicación empleados en mi formación para ser una persona feliz y con valores.

A mi hermano por su afecto y apoyo en cada etapa de mi vida estudiantil.

Mi agradecimiento de manera especial al ing. Marcelo Valdiviezo que con su experiencia y paciencia supo orientarme a lo largo de todo este trabajo de titulación.

A mi compañero Bryan por su paciencia y ayuda en cada uno de los ciclos de la carrera y en el presente trabajo.

A mis amigos y demás compañeros que estuvieron apoyándome siempre para la culminación del presente trabajo.

A la planta docente de la carrera de Ingeniería en Electrónica y Telecomunicaciones por todas sus enseñanzas impartidas a lo largo de todos estos años de mi vida universitaria.

Soraya Paola

TABLA DE CONTENIDO

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

ACRÓNIMOS

AAM	Active Appearance Models, Modelos de Apariencia Activa
AdaBoost	Adaptive Boosting, Aprendizaje Adaptativo
ANN	Artificial Neural Network, Red Neuronal Artificial
API	Application Programming Interface, Interfaz de Programación de Aplicaciones
BioID	Biometric Identification, Identificación Biométrica
CNN	Convolutional Neural Network, Red Neuronal Convolucional
DFA	Discriminating Feature Analysis, Análisis de Características Discriminatorias
GUI	Graphical User Interface, Interfaz Gráfica de Usuario
HOG	Histogram Of Oriented Gradients, Histograma de Gradientes Orientados
ICA	Independent Component Analysis, Análisis de Componente Independientes
IEEE	Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos
IoU	Intersection over Union, Intersección sobre Unión
JVM	Java Virtual Machine, Máquina Virtual de Java
LDA	Lineal Discriminant Analysis, Análisis Discriminante Lineal
MIT	Massachusetts Institute of Technology, Instituto de Tecnología de Massachusetts
MTCNN	Multi-Task Cascaded Convolutional Neural Network, Red Neuronal Convolucional en Cascada Multitarea
NMS	Non-Maximum Suppression, Supresión No Máxima
OCR	Optical Character Recognition, Reconocimiento de Carácter Óptico

O-Net	Output Network, Red de Salida
OpenCV	Open Source Computer Vision Library, Librería de Visión por Computador de Código Abierto
PAC	Probably Approximately Correct Learning Model, Modelo de Aprendizaje
PCA	Principal Component Analysis, Análisis de Componentes Principales
P-Net	Proposal Network, Red de Propuestas
R-Net	Refine Network, Red de Refinamiento
XGBoost	Extra Gradient Boosting, Aprendizaje de Gradiente Extra
YOLO	You Only Look Once, Solo se Mira Una Vez

1. TÍTULO

“IMPLEMENTACIÓN DE ALGORITMO DE DETECCIÓN Y SEGMENTACIÓN DE ROSTROS EN FOTOGRAFÍAS MEDIANTE PROCESAMIENTO DIGITAL DE SEÑALES”

2. RESUMEN

En el presente trabajo se plantea la implementación de un algoritmo de detección y segmentación de rostros en fotografías a través de la investigación del estado del arte en este campo de la visión por computador; para una elección óptima del algoritmo a implementar, se realizó un análisis comparativo entre los algoritmos de detección Viola & Jones, MTCNN, YOLO. Adicionalmente se presentan los resultados obtenidos del desarrollo de un algoritmo basado en la arquitectura de red neuronal convolucional (CNN, *Convolutional Neural Network*).

La comparación entre los modelos mencionados se realizará tomando en cuenta dos casos de estudio puntuales: fotografías que contengan un solo rostro y fotografías que contengan varios rostros. Para ambos casos se utilizarán un conjunto de prueba formado por 15 y 25 imágenes respectivamente, con variaciones en la pose, etnia, maquillaje y oclusión.

En adición, cada fotografía fue procesada digitalmente haciendo uso de diferentes técnicas de normalización como: Ecualización del histograma, corrección gamma, escala y rotación, con la finalidad de analizar el comportamiento de cada algoritmo en diferentes escenarios.

Finalmente, en imágenes que contengan un solo rostro la tasa de detección del algoritmo a implementar fue calculada a través del uso de parámetros de evaluación como: Verdaderos positivos, falsos negativos y falsos positivos; mientras que en imágenes de varios rostros la tasa de detección será determinada por el parámetro F-Score calculado en cada imagen.

Palabras clave: Detección de Rostros, Segmentación, Viola & Jones, MTCNN, YOLO, CNN, Normalización.

ABSTRACT

In the present work the implementation of an algorithm of detection and segmentation of faces in photographs is proposed through the investigation of the state of the art in this area of computer vision; for an optimal choice of the algorithm to be implemented, a comparative analysis was performed between the Viola & Jones, MTCNN, YOLO detection algorithms. Additionally, the results obtained from the development of an algorithm based on the convolutional neural network (CNN) architecture are presented in this work.

The comparison between the mentioned models will be carried out taking into account two specific case studies: photographs that contain only one face and photographs that contain several faces. For both cases a test set consisting of 15 and 25 images will be used respectively, with variations in the pose, ethnicity, makeup and occlusion.

In addition, each photograph will be digitally processed using different normalization techniques such as: Histogram equalization, gamma correction, scale and rotation, in order to analyze the behavior of each algorithm in different settings.

Finally, in images that contain only one face, the detection rate of the algorithm to be implemented will be calculated through the use of evaluation parameters such as: True positives, false negatives and false positives; while in multi-face images the detection rate will be determined by the F-Score parameter calculated in each image.

Keywords: Face Detection, Segmentation, Viola & Jones, MTCNN, YOLO, CNN, Normalization.

3. INTRODUCCIÓN

La visión por computador se basa en el estudio del comportamiento humano al detectar y reconocer diferentes objetos que se encuentran presentes en su medio, entre ellos los rostros de varias personas, de la misma manera los algoritmos que existen en la actualidad esperan lograr un desempeño semejante observando una imagen cualquiera y detectando en ella todo lo que se pueda identificar de acuerdo a sus bases de datos de entrenamiento, por ejemplo se podría detectar no solo los rostros humanos sino también el género de la persona, su edad, su expresión facial, etc.

Sin embargo, al igual que en varios campos de la inteligencia artificial, la visión por computador presenta varias dificultades al intentar que un computador tenga las mismas capacidades intuitivas de un ser humano al detectar los rostros, por lo que se vuelve de gran relevancia el procesamiento digital de señales, siendo éste la base de las técnicas usadas en la actualidad para la detección y segmentación de rostros en imágenes.

Además, existen diferentes beneficios que el procesamiento de señales aporta, entre ellos la elevada precisión que se puede conseguir al implementar como entrada del sistema una imagen previamente normalizada.

Las técnicas de procesamiento digital permiten estudiar la información relevante de cualquier enfoque en el que se va a tratar una imagen, el presente caso de estudio se enfocará en los rostros, además estas técnicas optimizan los aspectos importantes de las imágenes, asumiendo trabajos como eliminación de ruido, segmentación, mejoramiento del contraste, detección de bordes, etc.

La implementación de distintos algoritmos de detección de rostros en varias imágenes es uno de los temas más estudiados en el área de la visión por computadora, debido a la variedad de aplicaciones que esta ofrece, tales como sistemas de seguridad y cámaras fotográficas donde además se utiliza detección de sonrisas, solo por mencionar algunas.

Dentro de los algoritmos que constituyen el estado del arte sobre procesamiento facial se encuentran los siguientes:

- Algoritmo de Viola & Jones, el cual se basa en imagen integral, filtros Haar, cascadas de clasificadores y el algoritmo AdaBoost.

- Algoritmo basado en búsqueda exhaustiva multiescala, ecualización de histogramas y redes neuronales.
- Algoritmo basado en modelos de apariencia activa (AAM, Active Appearance Models), localiza componentes faciales y se apoya en procesos iterativos de ajuste del modelo, métodos de gradiente descendente y descomposiciones mediante PCA (*Principal Component Analysis*).
- El algoritmo *CamShift*, el cual realiza un proceso de seguimiento por color. (García Mateos, 2007)
- El algoritmo *Mutli-task Cascaded Convolutional Networks*, el cual es una red neuronal basada en tres redes diferentes: P-Net, R-Net y O- Net.

En el campo de estudio de la visión artificial continúan desarrollándose varias técnicas que permitan una detección de objetos con prácticamente el 100% de acierto, por lo que esta rama de investigación sigue siendo un desafío, debido a esto se desarrolla el presente caso de estudio, demostrando el potencial que se genera al aplicar técnicas de procesamiento digital en las imágenes.

OBJETIVOS

GENERAL

Implementar un algoritmo que permita realizar detección del número de rostros existentes y la segmentación de los mismos en distintas fotografías almacenadas en una base de datos, mediante el estudio de técnicas que extraen información crítica relacionada a las características del rostro.

OBJETIVOS ESPECÍFICOS

- Seleccionar el método más eficiente utilizado en segmentación de imágenes mediante el estudio de distintos algoritmos desarrollados actualmente para determinar la información correspondiente al rostro humano.
- Realizar un pre-procesamiento sobre las imágenes para mejorar la información relevante del rostro aplicando procesamiento digital de señales.
- Implementar el algoritmo final a través del compilador Python para realizar la detección y segmentación de rostros en fotografías.

4. REVISIÓN DE LITERATURA

El análisis de imágenes describe un procedimiento en el cual se puede identificar y comprender las características principales y relevantes respecto al tema de investigación que se desee tratar, en el caso específico del presente trabajo la detección y segmentación de rostros.

El término detección facial trata de la ubicación del rostro de un ser humano dentro de un retrato establecido ya sea por una imagen obtenida a través de un dispositivo o por una base de datos realizada de un determinado grupo de personas. (Creación & Modelo, 2018)

Para realizar esta tarea se pueden utilizar diferentes lenguajes de programación que posean librearías y funciones que faciliten la detección del rostro.

4.1 Lenguajes de Programación

De acuerdo con los datos recopilados por sitios como GitHub ("The State of the Octoverse: machine learning - The GitHub Blog," 2019), compañías como TIOBE("Top 8 of the TIOBE index quite stable for the last 15 years," 2019) e instituciones como IEEE("The 2018 Top Programming Languages - IEEE Spectrum," 2018) e IBM ("The Most Popular Language For Machine Learning Is ... (IT Best Kept Secret Is Optimization)," 2016), los lenguajes más utilizados para realizar tareas de aprendizaje automático son las siguientes:

4.1.1 *Python*

Python es uno de los lenguajes de programación que ha ganado popularidad en los últimos años, posicionándose en el primer lugar de los lenguajes utilizados para realizar algoritmos de aprendizaje automático. Es un lenguaje de programación de código abierto, de alto nivel, interpretado y de propósito general; admite paradigmas de desarrollo orientado a objetos, imperativo y funcional.

Puede utilizarse en varios sistemas operativos como Windows, Mac OS y Linux, además se puede desarrollar software para aplicaciones de aprendizaje automático, comunicaciones de red, aplicaciones de escritorio con interfaz gráfica de usuario (GUI, Graphic User Interface), etc.

Al ser un lenguaje de código abierto, cualquier persona puede contribuir con el desarrollo y distribución del mismo, no se necesita pagar licencia para distribuir programas realizados con este lenguaje.

Para facilitar la programación, Python incluye varias estructuras de datos de alto nivel como listas, diccionarios, tuplas, cadenas de texto y conjuntos; así mismo su librería estándar incorpora distintas funciones básicas como manejar strings, funciones de nivel medio como manejar ficheros ZIP o CSV y funciones de alto nivel para programación criptográfica. (Fernández Montoro, 2012)

Dentro de las librerías más utilizadas para realizar proyectos con Python se encuentran:

- Numpy: utilizado para realizar operaciones matemáticas en datos multidimensionales, ayudando en gran medida en proyectos de aprendizaje automático y ciencia de datos
- Scipy: utilizado en computación científica
- Pandas: se utiliza para administrar conjuntos de datos
- Matplotlib: es una biblioteca de visualización, se utiliza en más del 40% de los proyectos de aprendizaje automático y ciencia de datos.
- Scikit-Learn: librería popular de aprendizaje automático que contiene implementaciones de una gran cantidad de algoritmos utilizados en este campo.
- TensorFlow: librería que permite entrenar, ejecutar y trabajar con redes neuronales.

4.1.2 *C, C++, C#*

El lenguaje C es de propósito general, muy empleado a nivel mundial para la creación de repositorios o bibliotecas de código. Se ha utilizado para el desarrollo de diversas aplicaciones: sistemas operativos, hojas de cálculo, gestores de bases de datos, etc. Es un lenguaje portable, es decir, es independiente del hardware. Los programas escritos en C son fácilmente trasportables a otros sistemas.

Está débilmente tipificado como un lenguaje de medio nivel. Dispone de las estructuras típicas de alto nivel y a su vez de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones a este lenguaje que

posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria y dispositivos periféricos. Estas características propician la realización de implementaciones óptimas y a la vez no es necesario un soporte intenso en tiempo de ejecución.(Massó Gómez, 2012)

Siendo uno de los lenguajes de programación más destacados, C # es compatible con distintos software, como aplicaciones integradas, aplicaciones con controladores de dispositivos, aplicaciones de servidor y cliente de alta calidad. Fue diseñado para ser simple, moderno y un lenguaje orientado a objetos. El lenguaje es adecuado para el desarrollo de software y la implementación en entornos distribuidos. (Raman et al., 2018)

Permite a los desarrolladores crear todo tipo de aplicaciones, las bibliotecas utilizadas para aprendizaje automático son:

- Agentes ML: permite que los juegos y las simulaciones sirvan como entornos para entrenar agentes inteligentes.
- ML.NET: un marco de aprendizaje automático de código abierto y multiplataforma para .NET.
- Accord.NET: proporciona aprendizaje automático, estadísticas, inteligencia artificial, visión por computadora y métodos de procesamiento de imágenes para .NET.(Heath, 2019)

C ++ es un lenguaje de programación de propósito general que tiene una función de programación esencial, orientada a objetos básica y común. Se considera como el lenguaje de programación más rápido, lo cual es muy importante para la ejecución de algoritmos de inteligencia artificial pesados.

Dentro de las librerías que se ocupan para realizar algoritmos de aprendizaje automático se encuentra OpenCV y TensorFlow (escrita en C / C ++ de bajo nivel).(Raman et al., 2018)

4.1.3 *JavaScript*

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como

texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Los navegadores más modernos disponibles actualmente incluyen soporte de JavaScript hasta la versión correspondiente a la tercera edición del estándar ECMA-262. La sintaxis[1] de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. (Eguíluz Pérez, 2012)

En el campo de aprendizaje automático, JavaScript ha tenido una gran influencia desde el año 2018 con proyectos como:

- Tensorflow.js: es ampliamente utilizado debido a su núcleo integral de algebra lineal y sus capas de aprendizaje profundo.
- Brain.js: se considera una biblioteca muy fácil de entender, por lo que está enfocada para principiantes en aprendizaje automático.
- ml5.js: facilita el acceso a algoritmos y modelos de aprendizaje automático en el navegador.
- Face-api.js: incluye modelos entrenados de detección y reconocimiento de rostros. (Shin, 2019)

4.1.4 *Java*

Es un lenguaje de programación de propósito general orientado a objetos, ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Debido a que C++ es conocido en gran parte de la comunidad de programadores, Java se diseñó para ser parecido a C++ y así facilitar su aprendizaje.

[1] Conjunto de reglas para escribir el código en los lenguajes de programación.(Eguíluz Pérez, 2012)

Java elimina muchas características de otros lenguajes, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el reciclador de memoria dinámica, el cual permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Java es tanto interpretado como compilado. Es decir, aproximadamente el 20% del código Java es interpretado por la JVM (Java Virtual Machine), pero es un 20% muy importante. Tanto la seguridad de Java como su habilidad para ser ejecutado en múltiples plataformas se deben a que los pasos finales de la compilación se manejan localmente. ("Principales Características de Java," 2019)

Además, Java es seguro; es decir, el código pasa varias pruebas antes de ejecutarse en una máquina, es decir atraviesa un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un verificador de teoremas para detectar fragmentos de código ilegal (código que falsea punteros, vulnera derecho de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto). ("Características de Java," 2019)

En cuanto al campo de aprendizaje automático, existen algunas aplicaciones desarrolladas en Java las cuales son:

- WEKA, que está dedicado al aprendizaje automático y la minería de datos
- JOONE para diseñar, entrenar y probar redes neuronales. ("Machine Learning Tools I recently discovered," 2018)

4.2 Captura

Consiste en obtener las distintas imágenes por medio de un sensor, y se pueden presentar en las siguientes condiciones:

- Ambiente controlado: se presenta en condiciones favorables para la captura de la imagen, es decir el rostro no se encuentre obstruido por otros objetos, exista buena iluminación, entre otros.
- Ambiente no controlado: es lo contrario al anterior, es decir no existe garantía de características favorables. (Vivas Imparato, 2014)

4.3 Imagen Digital

La forma en que se presenta una imagen digital es a través de una matriz bidimensional de números, en la cual cada celda representa un píxel. Para transformar una imagen continua en digital se realiza dos procesos conocidos como muestreo y cuantificación, en el muestreo se digitalizan los valores de las coordenadas, y en la cuantificación los valores de la amplitud.

La calidad de la imagen digital depende del número de muestras que se tomen en cuenta en los dos procesos de digitalización, dado que ésta es una aproximación de la imagen originalmente analógica. Las unidades de muestreo y cuantificación, corresponden a la resolución y al tamaño del espacio característico respectivamente. (Robledano-Arillo - Valentín Moreno-Pelayo - José Manuel Pereira-Uzal, 2016)

En la figura 1 se puede observar que la imagen superior se convierte en una imagen digital la cual se interpreta como una matriz de valores de intensidad en un determinado rango.

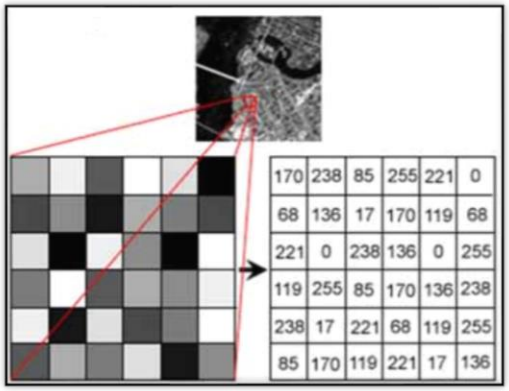


Figura 1. Imagen continua, imagen digital y matriz de valores de intensidad
Fuente: (Niclós Corts, 2011)

Cada píxel se asocia con un nivel de color para visualizar la imagen, los valores se encuentran entre 0 (negro) y 255 (blanco).

El ancho de la imagen está representando por el número de columnas, mientras que la altura se representa por el número de filas. Dependiendo del rango de valores que pueda tomar cada pixel se pueden diferenciar tres tipos de imágenes, siendo estos:

- Imágenes binarias: el píxel está compuesto de un bit por tanto solo puede tomar dos valores (0 = negro y 1=blanco). En la figura 2 se presenta una imagen de características binarias.

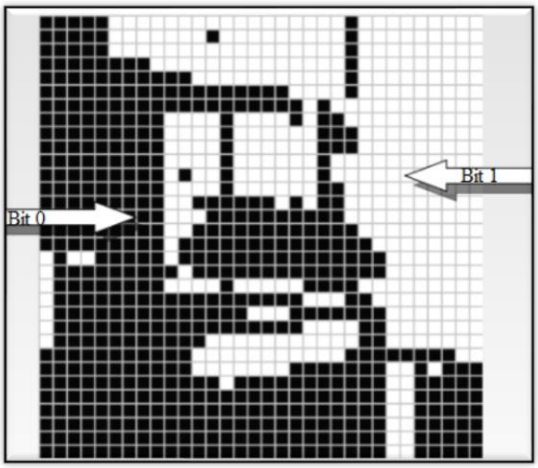


Figura 2. Imagen Binaria

Fuente: (Camps-Valls, 2012)

- Imágenes en escala de gris: el píxel toma 8 bits, es decir existen hasta 2^8 niveles de gris, por lo que su rango se encuentra entre 0 y 255. En la figura 3 se presenta una imagen en escala de grises con su correspondiente representación en el plano 3D; como se puede observar, entre más blanco es el píxel en 2D mayor su nivel de intensidad en su representación tridimensional.

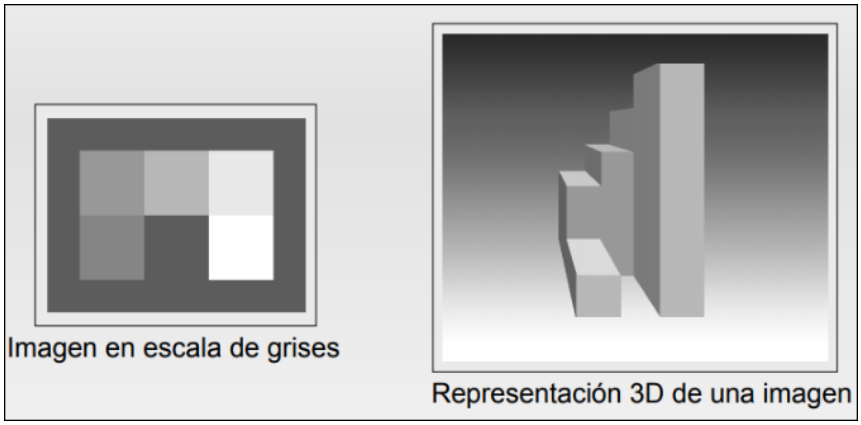


Figura 3. Imagen en escala de grises con su correspondiente representación en un plano 3D.

Fuente: (Camps-Valls, 2012)

- Imágenes en color: el píxel está cuantificado a partir de tres componentes independientes (Rojo, Verde, Azul) y 1 byte por cada valor, por tanto, existen 16.7 millones de valores (colores). En la figura 4 se presenta una imagen del plano RGB en la cual se puede apreciar los niveles de intensidad de cada canal y sus combinaciones para formar tonos secundarios.

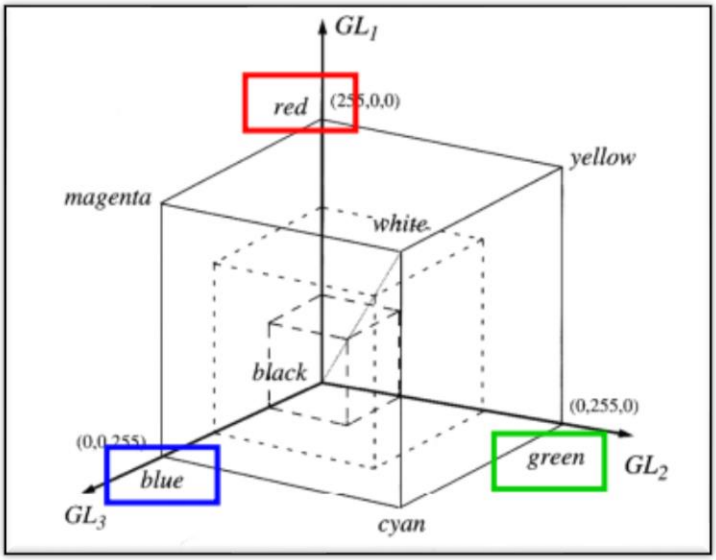


Figura 4. Plano RGB

Fuente: (Niclós Corts, 2011)

Las imágenes pueden también utilizar más o menos bits dependiendo de la profundidad de color que se desee en el análisis. (Párraga Párraga & Casa López, 2017)

4.4 Procesamiento de Imágenes

Se aplica técnicas de procesamiento de imágenes con el fin de intensificar las propiedades relevantes de una imagen para facilitar su posterior análisis, y se lo puede realizar tanto en el dominio espacial como en el frecuencial.

Las diferentes técnicas comprenden un conjunto de operaciones con la intención de optimizar la calidad visual de una imagen, reduciendo el contenido de información innecesaria presente en la misma o haciendo énfasis en las características de brillo y contraste. (Párraga Párraga & Casa López, 2017)

4.4.1 *Procesamiento Espacial*

En el dominio espacial, los valores de píxeles pueden modificarse de acuerdo con las reglas que dependen del valor de píxel original (procesos locales o puntuales).

Además, los valores de píxeles se pueden combinar o comparar con otros en su vecindad inmediata de varias maneras. Para un proceso de vecindad, cada píxel debe direccionarse muchas veces, y la velocidad de procesamiento se ralentiza en consecuencia. (Acuña & Farias Falkiewicz, 2018)

4.4.1.1 *Manipulación del Contraste*

La manipulación del contraste pretende lograr un uso adecuado de la pantalla a través de la expansión lineal del rango de contraste, estableciendo el valor del píxel más oscuro a negro, el valor más claro a blanco y cada valor restante a tonos de gris intercalados linealmente, de esta manera mejora la visibilidad de las características en la imagen. (Mello Román, 2017)

Una relación no lineal puede expandir una parte del rango de escala de grises mientras se comprime otra, esto se logra utilizando funciones de transferencia que de acuerdo a lo que se requiera destacar de la imagen, estas pueden ser diferentes y variar. (Russ & Brent Neal, 2017)

El principal objetivo de esta técnica es aumentar un conjunto dinámico de niveles de gris en la imagen, tal como se muestra en la figura 5 en la cual el contraste aplicado mejora la visibilidad de las características esenciales correspondientes a los distintos rostros. (Mello Román, 2017)

Figura 5. Mejora del contraste aplicada en diferentes rostros

Fuente: (Rivas Araiza et al., 2007)

Las funciones de transferencia para pantalla más comunes son aquellas en las que se relaciona el porcentaje de brillo visualizado con el valor del píxel almacenado, las mismas siguen relaciones matemáticas simples, como las curvas logarítmicas o de ley de potencia. (Russ & Brent Neal, 2017)

Se pueden utilizar varias funciones para manipular los valores del contraste, y éstas se pueden añadir a la expansión de contraste en caso de que la imagen no cubriese en un

principio todo el rango de negro a blanco, extendiendo el rango de escala original a uno de pantalla completa. En la figura 6 se muestran algunas de estas funciones. (Russ & Brent Neal, 2017)

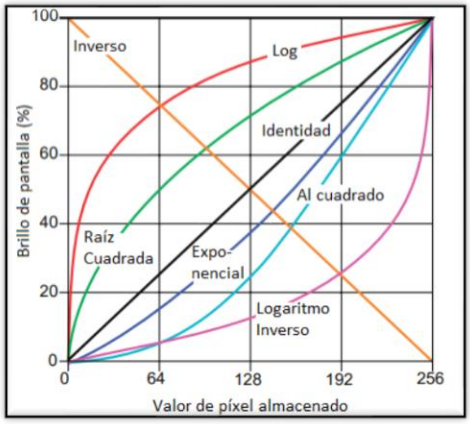


Figura 6. Ejemplos de funciones de transferencia para pantalla

Fuente: (Russ & Brent Neal, 2017)

4.4.1.2 *Corrección Gamma*

Una de las funciones más importantes para manipular el contraste de las imágenes es la función potencia, también conocida como corrección gamma; está función permite comprimir los brillos mostrados en el extremo brillante de la escala mientras se expanden aquellos en el extremo oscuro si se trata de una curva gamma aumentada. Este tipo de relación también puede convertir una imagen de una cámara con una respuesta lineal a la respuesta logarítmica más común. Una curva gamma reducida hace lo contrario, es decir los valores de intensidad oscuros se comprimen mientras se expanden los valores de intensidad claros. (Apolinario Casaño & Pinedo Lima, 2018)

Para realizar la técnica de corrección gamma en una imagen en escala de grises, se utiliza la siguiente fórmula: (Aranguren Zapata & Vela Asin, 2012)

$$Imagen_{nueva} = 255\left(\frac{Imagen_{entrada}}{255}\right)^{\gamma} \quad (1)$$

Donde γ es el factor de corrección gamma; si es menor a 1 los niveles de intensidad oscuros se convertirán en claros y, si es mayor a 1 los valores en tonos claros tomaran valores de intensidad de niveles oscuros.

4.4.1.3 *Ecualización del histograma*

La gráfica de un histograma convencional muestra la cantidad de píxeles que contienen cada uno de los 256 posibles valores de brillo almacenado en la imagen. Es decir, los picos dentro del histograma corresponden a los valores de brillo que se encuentran con frecuencia, los cuales por lo general corresponden a estructuras particulares que están presentes en la imagen y por el contrario los valles indican valores de brillo que son menos frecuentes. (Russ & Brent Neal, 2017)

De acuerdo con el párrafo anterior, al histograma se lo definiría como la representación de cada color de una imagen en su frecuencia relativa, si se pretende ecualizar el histograma, el resultado es la alteración de los valores de los píxeles de la imagen de manera global, ajustando el mismo de tal manera que sea en lo posible uniforme para todos los valores de intensidad. (Azueto-Ríos, Santiago-Godoy, Hernández-Gómez, & Hernández-Santiago, 2017)

Para realizar la ecualización del histograma en primer lugar se calculó el histograma acumulado de la imagen dado por:

$$H(i) = \sum_{k=0}^{i} h(k) \qquad (2)$$

Donde $h(k)$ es el histograma de la imagen e i son los niveles de intensidad de la imagen.

Dado que se necesita que el histograma sea uniforme, se utilizó la siguiente ecuación por cada nivel de gris presente en la imagen para lograrlo:

$$G(i') = (i' + 1)\frac{NM}{256} \qquad (3)$$

Donde $N\ y\ M$ son las dimensiones de la imagen, y *256* el número de niveles con los que se representa la imagen.

Asumiendo que $G(i') = H(i)$, se establece:

$$(i' + 1)\frac{NM}{256} = H(i) \quad \therefore \quad i' = \frac{256}{NM}H(i) - 1 \qquad (4)$$

Finalmente, se realizó la siguiente ecuación, debido a que los niveles de gris pueden ser únicamente valores enteros: (Esqueda Elizondo, 2005)

$$i_{nuevo} = Parte\ entera\left[\frac{256}{NM}H(i_{anterior}) - 1\right] \quad (5)$$

Como se observa en la figura 7, si se realiza una ecualización del histograma a la imagen se consigue mayor uniformidad en los niveles de brillo del histograma, es decir se pretende obtener la misma cantidad de píxeles para cada uno de los niveles de gris (0-255).

Una vez realizado este procedimiento se minimizan los cambios de luminosidad, aportando de manera positiva al bueno desempeño de los algoritmos de extracción de características. (Cazorla Martínez, 2016)

Figura 7. A la izquierda imagen original a escala de grises y a la derecha imagen resultante de la ecualización del histograma

Fuente: (Cazorla Martínez, 2016)

4.4.1.4 *Escala*

Para aplicar la técnica de escalado a una imagen se debe multiplicar cada uno de los píxeles de una imagen por un escalar, conservando la relación de aspecto de la imagen original.

El escalamiento se realiza de la siguiente forma:

$$B(x, y) = a \times A(x, y) \quad (6)$$

Cuando el escalar o constante es menor a 1, se reduce la resolución de la imagen y si es mayor a uno aumenta. (Esqueda Elizondo, 2005)

4.4.1.5 *Rotación*

Esta técnica permite rotar la imagen en varios ángulos, dependiendo de los requerimientos; la rotación generalmente se hace a partir del punto central de la imagen, pero en caso que se necesite un punto en particular se debe especificar el mismo como punto central del plano, de esta manera se lo podrá definir como eje de rotación.

4.4.1.6 *Filtrado Espacial*

La principal razón para aplicar filtros dentro de una imagen es para resaltar o atenuar ciertas características, las cuales se encuentran dentro de una banda de frecuencias espaciales (altas o bajas frecuencias), las operaciones que se llevan a cabo para realizar este procedimiento se definen como filtrado espacial. (Acuña & Farias Falkiewicz, 2018)

Dichas operaciones se realizan de manera directa en cada uno de los píxeles de la imagen, obteniendo información relevante al comparar todos los puntos de la imagen con un conjunto de píxeles que se encuentran de forma adyacente al pixel objeto, dependiendo del tipo de filtro aplicado, se podría actuar sobre un píxel en concreto, el cual ofrezca mejoras significativas sobre la imagen y permita un tratamiento más sencillo para su posterior análisis o bien se pueda determinar datos relevantes con los cuales se pueda trabajar de manera más precisa y rápida. (Castillo, Hernández, Inzunza, & Torres, 2013)

Dentro de los filtros espaciales se pueden encontrar dos tipos:

- Filtros lineales.- se basan en Kernels o máscaras de convolución. Dentro de estos filtros se encuentran los filtros paso bajo y los filtros paso alto
- Filtros no lineales.- utilizan técnicas que relacionan todos los valores de brillo que se le asignan al grupo de pixeles en la entrada de la imagen. Los filtros que corresponden a esta clasificación son los denominados filtros de la mediana y el filtro de máximo. (Acuña & Farias Falkiewicz, 2018)

Considerando el desempeño que se requiere del filtro, se pueden aplicar distintos procedimientos especiales en los cuales se pueda bien ocupar un marco exterior de ceros, repetir los valores de los bordes o a su vez no aplicar ninguno de ellos, en todo caso el tipo de filtro que se utilice va a depender del contenido del Kernel que se maneje en el análisis. (Castillo et al., 2013)

4.4.2 *Procesamiento Frecuencial*

En el análisis y procesamiento de imágenes, la representación de la adición de todas las señales periódicas que se encuentran en una imagen con distintas frecuencias se conoce como el procesamiento frecuencial (Esqueda, 2002). Dentro de las técnicas que se utilizan para el procesamiento frecuencial, existen unas cuantas que no se relacionan con propósitos de selección y mejora de características o rasgos importantes. Por ejemplo, algunos procedimientos se llevan a cabo para realizar la compresión de una imagen dada, reduciendo de esta manera la cantidad de datos y ofreciendo una mayor eficacia para su posterior almacenamiento o transmisión.

Cuando se aplica este tipo de procedimientos se requiere pasar del dominio de la frecuencia al dominio espacial para reconstruir y poder visualizar la imagen. Cuando se realiza este tipo de transformaciones se requiere que sean rápidas y con un efecto mínimo en la calidad de la imagen. Dentro de los parámetros que definen la calidad que se presenta en una imagen se encuentran los niveles de brillo, valores de color, ubicación de límites de las características y la introducción o eliminación de la textura, los cuales se ven afectados al momento de comprimir la imagen y como resultado se tiene, que entre más se comprima una imagen y se pierdan algunos datos, más afectada se verá la fidelidad de la imagen. (Russ & Brent Neal, 2017)

4.4.2.1 *Convolución*

Esta técnica permite comprender cómo diferentes sistemas de imágenes logran alterar o degradar las características de las mismas. La convolución es una operación que se realiza entre matrices, dando como resultado un valor real obtenido de la combinación de la multiplicación uno a uno de las matrices y sumando cada uno de estos productos, analíticamente:

Sean A y B dos matrices de la forma $\Re^n \times \Re^n$ con elementos de matriz (a_{ij}) y (b_{ij}) respectivamente, la convolución sería: (Martín Ortíz, 2013)

$$A \otimes B = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij} = z, \quad | \quad z \in \Re, \tag{7}$$

4.4.2.2 *Correlación*

El método de correlación sirve para obtener resultados en los cuales se pueda visualizar un parecido entre varios píxeles de una imagen, en el caso de que existan píxeles de esta naturaleza se dice que los mismos se encuentran correlacionados entre sí.

La correlación se diferencia de la convolución en el hecho de que en la correlación la máscara no se rota a diferencia de lo que sucede con la máscara en la convolución.

Sea el píxel de salida $g(x, y)$, su valor viene determinado por la suma ponderada de los píxeles vecinos, de esta manera la correlación está dada por: (Esqueda Elizondo, 2005)

$$g(x, y) = h(x, y) \circ f(x, y) = \sum_{i=-\infty}^{\infty} \sum_{\infty}^{\infty} f^*(i, j) h(x + i, y + j) \qquad (8)$$

4.5 Extracción de Características

La extracción de características es un procedimiento que se realiza para descubrir características relevantes que permitan relacionar la imagen de entrada con la presencia de un rostro en ella, detectándolos y clasificándolos en caso de estar presentes en la imagen, además se los etiqueta de acuerdo a la clase establecida con anterioridad durante el entrenamiento del algoritmo que se implementó para la detección de rostros.

En la figura 8 se muestra la manera en la que se ha clasificado los algoritmos que se utilizan para la detección de rostros. (Sánchez Salazar, 2017)

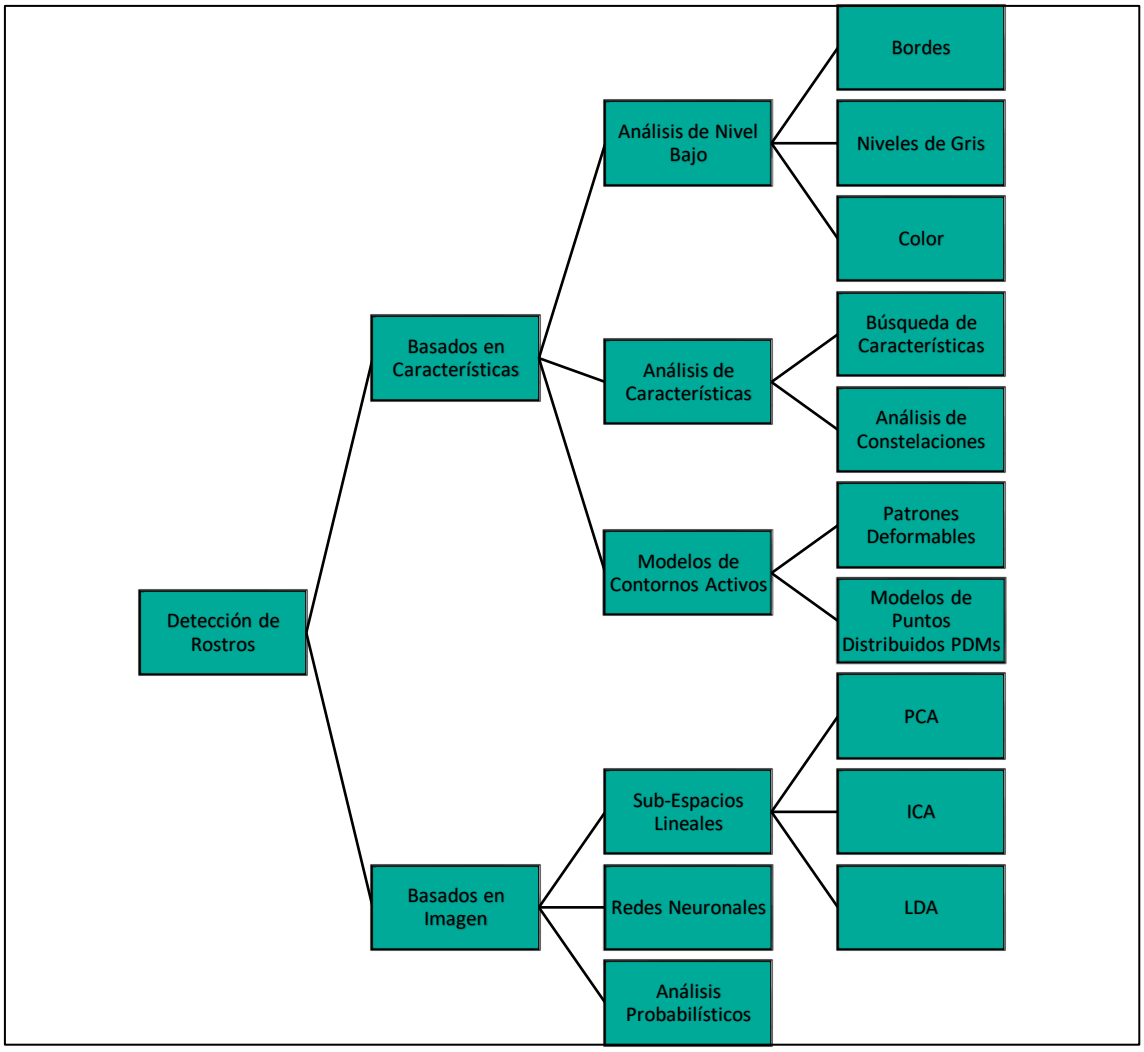


Figura 8. Clasificación de algoritmos para detección de rostros

Fuente: Autor

4.6 Algoritmos de Detección

4.6.1 *Máquinas de Soporte Vectorial (SVM, Support Vector Machine)*

Una SVM es un sistema de aprendizaje automático se utiliza cuando es necesario solucionar de manera óptima aquellos problemas relacionados con la clasificación y regresión. Las SVM se basan en la Teoría de Aprendizaje Estadístico, su eficiencia está basada en tres consideraciones entre las cuales se encuentran en primer lugar que constan de una base matemática muy sólida, en segundo lugar se encuentra el hecho de que pueden minimizar el riesgo de una clasificación equivoca haciendo uso de nuevas

muestras y la última consideración radica en que cuentan con algoritmos óptimos y fuertes herramientas de clasificación para conseguir una solución en un tiempo mínimo y de manera satisfactoria.

Las SVM se encuentran preparadas para clasificar todo tipo de muestras en dos posibles resultados: "positivos" o "negativos", que para el caso de detección facial se refiere a aquellas imágenes que corresponden a "rostros" y "no rostros" respectivamente. Para obtener estos resultados es necesario realizar un entrenamiento previo de la SVM.

El entrenamiento de la SVM se realiza mediante la obtención de varios ejemplos referentes a casos positivos o negativos. El proceso de aprendizaje comienza cuando se conoce el primer modelo de detección, para luego relacionarlo y compararlos con un conjunto de imágenes establecidas previamente para el entrenamiento. Cuando se realiza este procedimiento y se obtiene un resultado erróneo con una imagen entrenada, se dice que se obtiene un falso negativo en el caso de que la imagen entrenada como rostro no sea detectada y un falso positivo en el caso de que la imagen entrenada como no rostro sea detectada como que si lo es. Luego de esto se reentrena la SVM añadiendo tanto los falsos positivos como los falsos negativos al conjunto de datos que corresponde al lugar donde se produjo la detección errónea. Para conseguir una precisión óptima se debe reentrenar la SVM tantas veces como sea necesario hasta conseguir datos satisfactorios.

4.6.2 *DFA-SVM (Discriminating Feature Analysis – Support Vector Machine)*

Este algoritmo realiza varios procesos, en el cual el primero de ellos se basa en el método DFA que deriva un vector de características discriminatorias al combinar la imagen de entrada, su representación de wavelets Haar y sus proyecciones de amplitud. Mientras que las wavelets Haar producen una representación efectiva para la detección de objetos, las proyecciones de amplitud capturan las distribuciones simétricas verticales y las características horizontales de las imágenes de rostro humano. Segundo, el modelado de clases presenciales estima estadísticamente la función de densidad de probabilidad[2] (PDF,

[2] Una función de densidad de probabilidad, describe la probabilidad de ocurrencia de cada valor de una variable aleatoria (discreta o continua); un histograma es una representación común de una función de densidad de probabilidad discreta. (Llinás Solano, 2014)

Probability Density Function) de la clase de cara y define una medida basada en la distribución para clasificación facial y no facial. La clase de cara se modela como una distribución normal multivariable, y la medida basada en distribución luego separa los patrones de entrada en tres clases: la clase cara (patrones cerca de la clase cara), la clase no cara (patrones alejados de la clase cara), y La clase indecisa (patrones ni cerca ni lejos de la clase de cara). Finalmente, SVM junto con la medida basada en la distribución clasifica los patrones en la clase indecisa en cualquiera de las caras clase o la clase no cara. (Shih & Liu, 2006)

4.6.3 *HOG (Histograms of Oriented Gradients) – Navneet Dalal y Bill Triggs – 2005*

El método se basa en la evaluación de histogramas locales bien normalizados de orientaciones de gradiente de imagen en una grilla densa. La idea básica es que la apariencia y la forma del objeto local a menudo se pueden caracterizar de manera correcta por la distribución de gradientes de intensidad locales o direcciones de borde, incluso sin un conocimiento preciso de las posiciones correspondientes de gradiente o borde. En la práctica, esto se implementa dividiendo la ventana de imagen en pequeñas regiones espaciales (celdas), para cada celda, acumulando un histograma 1-D local de direcciones de gradiente u orientaciones de borde sobre los píxeles de la celda. Las entradas combinadas del histograma forman la representación. Para una mejor invariancia a la iluminación, sombreado, etc., también es útil contrastar-normalizar las respuestas locales antes de usarlas. Los bloques de descriptores normalizados son conocidos como histograma de descriptores de gradiente orientado (HOG). El mosaico de la ventana de detección con una cuadrícula densa (de hecho, superpuesta) de descriptores HOG y el uso del vector de características combinadas en un clasificador de ventana basado en SVM convencional proporciona el sistema de detección de rostros humanos.

4.6.4 *VJ Paul Viola y Michael Jones*

El algoritmo de Viola & Jones (Viola & Jones, 2001) utiliza como características la diferencia de intensidad que existe dentro de un conjunto determinado de pixeles. El procedimiento que sigue para determinar el valor de las características es sumar los valores de luminiscencia de la región blanca y restarlos de los valores de los pixeles de la región negra. Con las características resultantes se pueden formar hipótesis acerca de las

regiones existentes de una imagen. Además, este algoritmo utiliza una imagen integral para calcular de forma rápida el total de pixeles dentro de una imagen.

En la etapa final utiliza clasificadores en cascada, logrando de esta manera eliminar en las primeras etapas un número alto de ejemplos negativos con un coste muy bajo. (Escriche, 2017)

4.6.4.1 *Características de Haar*

Las características de Haar son similares a las wavelets[3] de Haar, se basan en la búsqueda de rasgos simples en una imagen y se refieren a la diferencia existente entre la suma de los píxeles dentro del rectángulo blanco y la suma de los píxeles en rectángulo gris. De esta manera la característica de importancia adquiere un valor numérico y su posición se determina por ventana de búsqueda. (Parra Barrero, 2015) La clasificación basada en características es más eficiente y más rápida que el procesado que se realiza en el análisis de píxeles.

De acuerdo al estudio de Viola & Jones se utilizan específicamente tres tipos de características:

- Características de dos rectángulos, representando bordes y calculadas de la diferencia entre las sumas de los píxeles contenidos en ambos rectángulos. Los rectángulos tienen el mismo tamaño y son adyacentes.
- Características de tres rectángulos, las que representan líneas y vienen dadas por la diferencia entre la suma de los rectángulos exteriores y el rectángulo central.
- Características de cuatro rectángulos, estás son de forma en X y se calculan a partir de la diferencia entre pares diagonales de rectángulos.

Se puede obtener nuevas características rotando 45 grados las características de dos y tres rectángulos, tal como se muestra en la figura 9.

[3] Señal oscilatoria de corta duración con energía finita concentrada en un intervalo de tiempo determinado, se utilizan para representar datos u otras funciones.

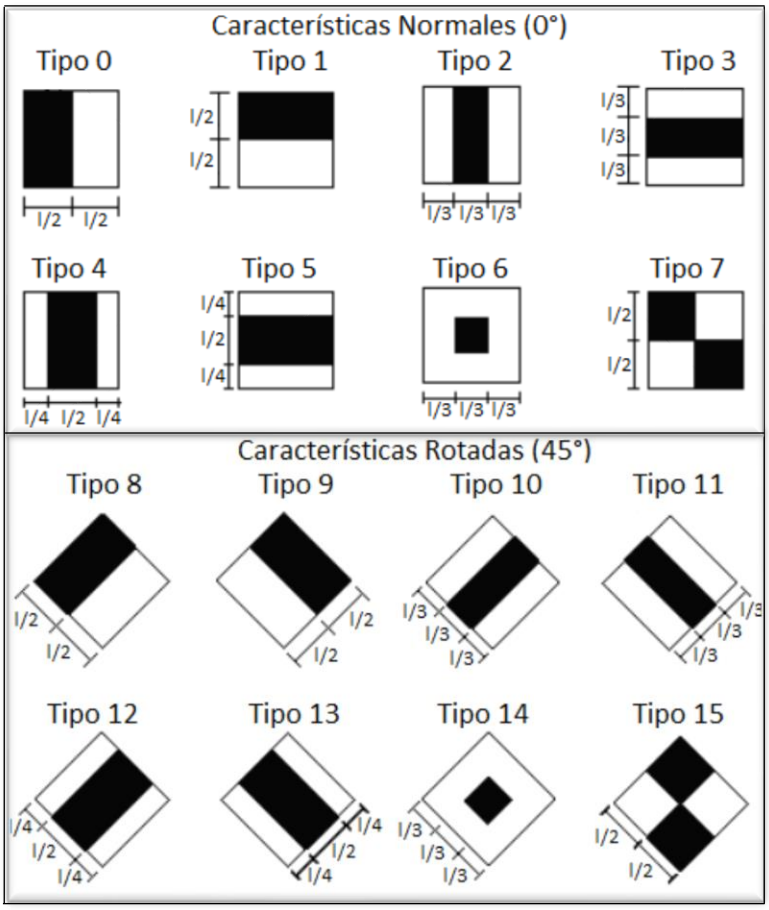


Figura 9. Clasificadores de Haar

Fuente: (Morales Navarro, et al. 2017)

4.6.4.2 *Imagen Integral*

Es una representación de la imagen original, permite extraer de forma rápida características de Haar a diferentes escalas o ubicaciones en un tiempo constante. Los valores que se presentan en esta imagen resultan de la suma generada por los distintos valores de intensidad presentes en un rectángulo de la imagen original, los mismo se consiguen haciendo uso de un algoritmo. (Espinoza & Jorquera, 2015)

La imagen integral en el punto (x, y) consiste en la suma de los píxeles ubicados en la parte superior y a la izquierda de dicho punto, se calcula mediante la siguiente ecuación:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (9)$$

Donde $ii(x, y)$ es la imagen integral en la ubicación de píxeles (x, y) , y los valores correspondientes a (x', y') son los valores de la imagen original. La imagen integral es altamente eficiente en el cálculo de áreas rectangulares.

Al utilizar imágenes integrales la ventaja que se presenta es que el tiempo de ejecución requerido para calcular distintas características en una imagen dada, se reduce debido a que la cantidad de operaciones es inferior a lo que serían con la imagen original, esto se evidencia en que el tiempo de ejecución de cálculo entre características iguales y distintas es aproximadamente igual. (Platero Plazas, 2015)

4.6.4.3 *Extracción de Características*

Para extraer las características de cada objeto en una imagen se usan patrones que permiten representar y describir dichas características. En este proceso se pretende encontrar atributos de los objetos que permitan asociarlos a determinada clase a través del procesamiento de sus medidas. (Sánchez Salazar, 2017)

En el enfoque de Viola & Jones, se aplica filtros Haar que como se explicó en la sección 4.6.4.1 permiten la extracción de características, estos filtros son selectivos en cuanto a orientación espacial y frecuencia, permiten modificaciones en orientación y escala, y pueden calcularse correctamente sobre la imagen integral.

Los filtros Haar generan características de borde, puntos y líneas a través de captura de contraste entre regiones con la finalidad de formar una codificación de diferencia de intensidades en la imagen. (López Aragoneses, 2016) La convolución de una imagen integral con un filtro *Haar* se muestra en la figura 10. (Guevara, Echeverry, & Urueña, 2008)

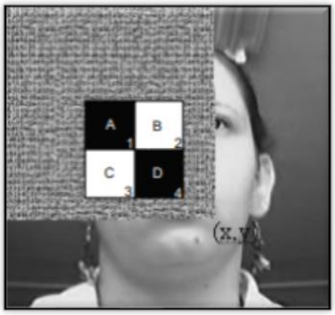


Figura 10. Convolución del filtro Haar con una imagen integral

Fuente: (Guevara et al., 2008)

4.6.4.4 *Clasificación*

Para ejecutar la etapa de clasificación se utiliza el método de Boosting[4] el cual se basa en combinar varias reglas de predicción débiles y prudentemente imprecisas para crear una regla de predicción con una precisión elevada. (Caballero Barriga, 2017)

Kearns y Valiant plantearon la idea de mejorar un algoritmo de aprendizaje débil en un algoritmo con mayor precisión fuerte usando PAC[5]. La OCR (*Optical Character Recognition*) fue la primera en experimentar con esos algoritmos mejorados.

Los tres algoritmos más importantes de Boosting son:

• *AdaBoost (Adaptive Boosting)*

• *Gradient Tree Boosting*

• *XGBoost* (Caballero Barriga, 2017)

- Clasificadores Haar en cascada

Un elemento esencial del detector Viola & Jones es el clasificador *Haar* en cascada. Se basan en árboles de decisión y se utilizan para verificar la presencia de un rostro en una imagen, recurriendo a clasificadores débiles para cada característica relevante a la presencia de un rostro, es decir características que se necesita detectar. Para realizar este procedimiento se hace uso de ventanas de diferentes tamaños y se las va deslizando a través de la imagen, para determinar si una de las ventanas contiene el objeto de interés (rostro) se necesita una cantidad pequeña de características. Utilizando una cascada de clasificadores en lugar de un solo clasificador fuerte se analizan varias ventanas que tengan probabilidad de contener el objeto de interés y se les aplica las características de Haar para distinguir dicho objeto, de esta manera se elimina una cantidad aproximada de 50% de ventanas que no contengan información de interés. Con este procedimiento para detectar características, se pueden aumentar de igual manera la tasa de falsas alarmas

[4] Algoritmo desarrollado para realizar investigaciones respecto al aprendizaje de máquina (Machine Learning), reduce la varianza a través se aprendizaje supervisado, reúne varios algoritmos débiles para formar un algoritmo fuerte.

[5] El Boosting inicialmente se conocía bajo el nombre de "PAC" (Probably Approximately Correct Learning Model), expuesto en 1984 por Leslie Valiant, convirtiéndose en el modelo más importante en la teoría del aprendizaje computacional. (Caballero Barriga, 2017)

como la de verdaderas reduciendo el número de etapas, aumentando la velocidad del detector. (Sánchez Salazar, 2017)

Con los clasificadores en cascada se agrupa las características Haar en distintas etapas en lugar de aplicarlas en una sola ventana, logrando un uso eficiente de los clasificadores débiles. Cada etapa deberá decidir de forma binaria si la muestra analizada se mantiene para la siguiente etapa o si se descarta inmediatamente, evitando la comparación con las características restantes en las ventanas. Adicionalmente, se debe establecer una relación equilibrada entre el número de clasificadores débiles y el umbral o tiempo de decisión. Es decir, si se establece umbrales de decisión cortos el detector final será muy rápido pero la precisión del mismo disminuirá; por el contrario, si se establece umbrales de decisión grandes la mayoría de las sub-ventanas deberá superar muchos nodos lo cual disminuye notablemente la velocidad de detección. (C. Zhang & Zhang, 2010).

En el trabajo de Viola y Jones para detección de rostros humanos se presentan datos de precisión del 95% con el uso de 200 funciones simples. Al utilizar una cascada de clasificador de Haar se podría detectar rostros humanos a una velocidad de al menos cinco cuadros por segundo. (Caballero Barriga, 2017)

4.6.5 *Redes Neuronales Artificiales*

Los algoritmos que se basan en redes neuronales contienen varias propiedades entre las que se encuentran las habilidades de aprender, adaptarse, agrupar y generalizar. Sus operaciones se realizan con un procesamiento en paralelo, constituido por muchas unidades de procesamiento sencillas llamadas neuronas, las capas ocultas de neuronas y las conexiones interneuronales. (Jiménez Ochoa, 2015)

El conocimiento de la red se adquiere con un proceso denominado entrenamiento; este proceso, durante la ejecución de las capas utilizadas, almacena las funciones que mejor describen las características faciales en una variable comúnmente conocida como peso. En el transcurso del entrenamiento estos pesos se irán descartando para evitar el sobreajuste de la red. Implícitamente, las neuronas de la red se activan siempre que se supera un umbral, el cual viene dado por una función de activación como: función escalón, tangente hiperbólica o sigmoide. La neurona puede ser lineal o no lineal dependiendo de la función utilizada. (Peña & Orellana, 2018)

Los modelos realizados con ANN (*Artificial Neural Network*) generalmente son conocidos como aprendizaje automático o *machine learning*, y a través del tiempo han evolucionado de la siguiente manera:

- En sus inicios se desarrolló una red conocida como Perceptrón, que es un modelo de una sola neurona, utilizado para clasificar patrones simples, como el resultado del comportamiento de una compuerta AND.
- A continuación, se crearon redes multicapa de perceptrones con retro-propagación (*backpropagation*) de errores, es decir se ajustaron los parámetros de la red de acuerdo a los errores que ocurrieron dentro de la neurona.
- Por último, se añadió el uso de autocodificadores para representar datos en una jerarquía, mejorando el comportamiento de las redes que se utilizan en redes multicapa. Esto es el principio del conocido aprendizaje profundo (*Deep Learning*). RNA-AP: Redes Neurales Artificiales con Aprendizaje Profundo. (Vázquez, Constable, & Nacional, 2018)

La forma en la que se organizan las neuronas en el interior de la red neuronal es decir la arquitectura está relacionada directamente con los algoritmos de aprendizaje utilizados durante el entrenamiento. De manera general las neuronas se agrupan en estructuras denominadas capas y el conjunto de las mismas conforma una red neuronal.

Las capas se clasifican en: entrada, ocultas y salida. La capa de entrada recibe los datos, las capas ocultas son las que proporcionan el aprendizaje a la red neuronal, modelando las características requeridas por la red y al final la capa de salida proporciona el resultado de dicho aprendizaje.

Se distinguen dos formas de clasificar las arquitecturas para realizar una red neuronal:

- Considerando la estructura
    - Redes monocapa: compuestas por una sola capa
    - Redes multicapa: compuestas de varias capas
- Considerando la manera en la que fluye los datos:
    - Redes unidireccionales (feedforward): la información fluye en un solo sentido

- Redes recurrentes (feedback): las capas se retroalimentan, es decir se puede recibir información de la capa de salida a la de entrada. (Larrañaga, Inza, & Moujahid Abdelmalik, 2005)

La arquitectura de un tipo de ANN más comúnmente utilizada es la de una red neuronal convolucional (CNN) consiste en múltiples capas y con retroalimentación de datos, es decir, es una red multicapa recurrente.

En la actualidad las técnicas más utilizadas para detección de rostros se basan en algoritmos de redes neuronales convolucionales ya que presenta un elevado porcentaje de aciertos, siendo superior al 95% en ciertos casos. Las redes neuronales durante la etapa de entrenamiento utilizan un grupo de imágenes de todo tipo de caras, es decir, de diferentes razas, tonos de piel, posición de labios, etc; y otro grupo de imágenes que no sean caras, de esta manera el sistema puede aprender cuáles imágenes corresponden a una cara y cuáles no. (Suquinagua León, 2019)

Para el presente trabajo se utilizó dos modelos de redes neuronales, los cuales se describen a continuación.

4.6.5.1 *MTCNN (Multi-task Cascaded Convolutional Networks)*

El modelo MTCNN integra un método de extracción de características para detección y alineación de rostros conjuntos, basado en una arquitectura CNN y un aprendizaje multitarea que consta de tres etapas, lo que permite elevar el rendimiento del algoritmo.

En la entrada del sistema se ingresa una imagen que se redimensiona en diferentes escalas, dando lugar a una pirámide de imágenes que se procesa en la primera etapa de la red llamada P-Net. A su vez, los datos obtenidos a la salida de la P-Net son los que alimentarán la segunda etapa denominada R-Net. Finalmente, la etapa nombrada O-Net utiliza como entrada los datos de salida de la R-Net. En la figura 11 se describe las arquitecturas de cada red, donde “MP” significa agrupación máxima y “Conv” significa convolución.

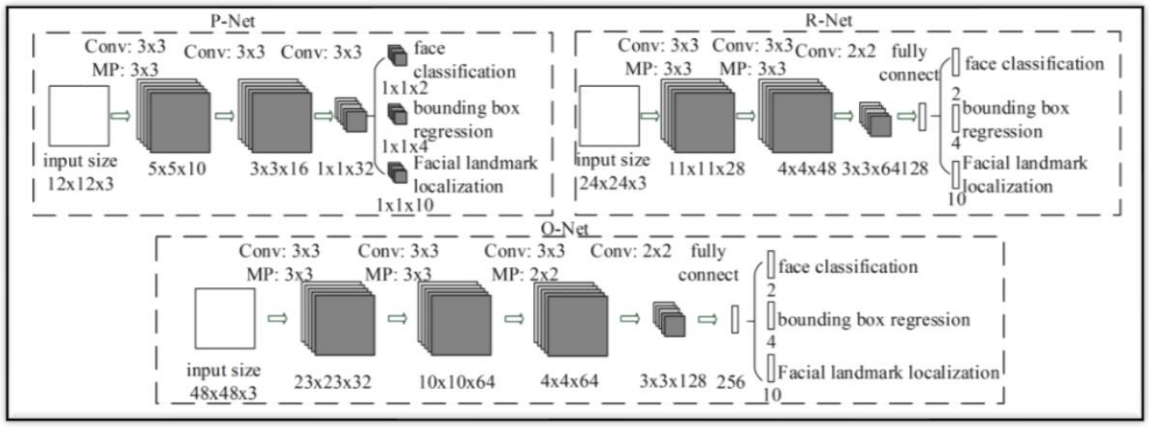


Figura 11. Arquitectura de MTCNN

Fuente: (K. Zhang, Zhang, Li, & Qiao, 2016)

Las tres etapas mencionadas anteriormente se detallarán a continuación:

- Etapa 1: P-Net (Red de Propuestas), retorna los recuadros con una probabilidad de contener un rostro dentro del umbral ($0{,}65 \leq \text{umbral} \leq 1$) establecido en la red. Además, se presentan los puntos correspondientes a los vértices del cuadro que delimita la ventana clasificada como rostro y se aplica supresión no máxima para minimizar el número de cuadros sobrepuestos en un mismo rostro. Las muestras con las que se entrena la P-Net se obtienen realizando cortes aleatorios en las imágenes de la base de datos WIDER FACE, tanto para imágenes de caras, no caras y partes de caras. Así mismo, para el entrenamiento de puntos faciales se utilizó la base de datos CelebA.
- Etapa 2: R-Net (Refine Network) adquiere los datos de la P-Net para disminuir la cantidad de recuadros clasificados erróneamente mediante regresión de cuadro delimitador y supresión no máxima. Las muestras para el entrenamiento de esta red se obtienen de la misma manera que en la P-Net, incluyendo sus falsos positivos, a través de un procedimiento denominado *hard sample mining*[6] con lo cual se mejora la precisión del modelo,
- Etapa 3: O-Net, permite detallar el rostro mediante sus puntos faciales[7]; añadiendo los 5 puntos principales de un rostro a los resultados obtenidos en las etapas

[6] Se refiere a un proceso de entrenamiento con mayor cantidad de datos, obtenidos de etapas previas (reentrenamiento). (K. Zhang et al., 2016)

[7] Corresponden a la ubicación de cada ojo, la punta de la nariz y las comisuras de los labios.

anteriores. Para el entrenamiento las muestras son obtenidas de la misma manera que en la R-Net.

Para reducir el coste computacional, la red implementa filtros con tamaño de 3 × 3, de esta manera además mejora el rendimiento, debido a la profundidad de los datos. (K. Zhang et al., 2016)

4.6.5.2 *YOLO (You Only Look Once)*

A grandes rasgos, YOLO es un algoritmo de aprendizaje profundo utilizado para detección de objetos en tiempo real, el cual se enfoca en mejorar la velocidad y el porcentaje de reconocimiento evitando la generación de regiones con posibilidad de contener un objeto.

El algoritmo está fundamentado en una sola red neuronal convolucional y se basa en procesar la imagen completa, obteniendo una única salida con todos los cuadros delimitadores que posean un objeto dentro de las clases conocidas por YOLO. Los valores que se obtienen en los cuadros delimitadores son: las coordenadas (x, y) que representan el centro del cuadro, el ancho, la altura y la confiabilidad de que exista un objeto dentro del cuadro, este último representa el IoU[8] entre el cuadro predicho y el cuadro verdadero.

La arquitectura de la red se fundamentó en el modelo de GoogLeNet[9] y está compuesta por 24 capas convolucionales, seguidas de 2 capas *Fully Connected*[10], utiliza capas de *Max Pooling* de 2 × 2, seguidos de capas convolucionales con filtros de 3 × 3.

Las primeras capas se encargan de extraer las características de la imagen y las *Fully Connected* extraen los puntos y las probabilidades. La arquitectura se observa en la figura 12.

[8] IOU (Intersección sobre Unión) representa una fracción entre 0 y 1, en donde la intersección se refiere al área sobrepuesta entre el cuadro delimitador predicho y el verdadero, y la unión se refiere al área total entre ambos cuadros. Idealmente, la IOU debe ser cercana a 1, indicando que el cuadro predicho es cercano al verdadero. (Redmon et al., 2016)

[9] Modelo de red profunda compuesta de 27 capas, cuya calidad se evalúa en el contexto de la clasificación y la detección El modelo agrega clasificadores auxiliares conectados en capas intermedias, esperando discriminación en las etapas inferiores del clasificador. La red se diseñó teniendo en cuenta la eficiencia computacional y la practicidad. (Szegedy et al., 2015)

[10] Capa totalmente conectada, es decir cada píxel se considera como una neurona que busca un patrón distinto en la red. Cada neurona de la siguiente capa se conecta con todas las neuronas de la capa anterior.

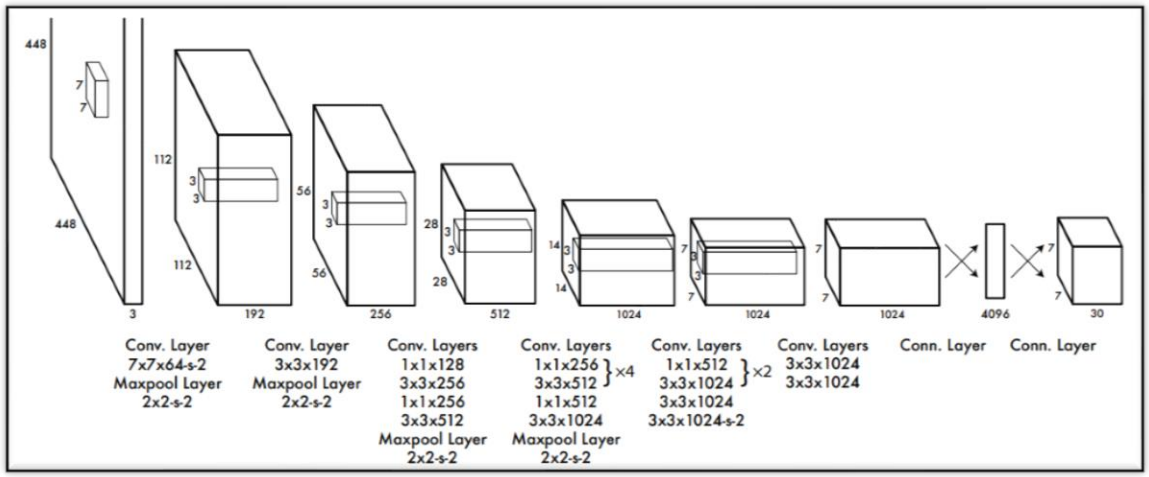


Figura 12. Arquitectura del Modelo YOLO

Fuente: (Redmon, Divvala, Girshick, & Farhadi, 2016)

La red durante su entrenamiento tuvo como parámetros: 135 épocas, *batch size*[11] de 64, *momentum*[12] de 0.9, *decay*[13] de 0.0005 y *dropout*[14] de 0.5. En las primeras 75 *epochs*[15] se colocó un *learning rate*[16] de 10^{-2}, luego 10^{-3} para las siguientes 30 y finalmente 10^{-4} para las que restaban. Las funciones de activación que se implementa en esta red son la función lineal para la capa final y la función rectificada lineal uniforme para todas las demás. (Redmon et al., 2016)

- *YOLOv3*

YOLOv3 es un modelo que se basa en las versiones anteriores de YOLO, capaz de detectar hasta 80 objetos diferentes en imágenes y video. Su red se conoce como Darknet-53, y se encuentra compuesta de 53 capas convolucionales, usando capas sucesivas de 3 × 3 y de 1 × 1 sucesivamente.

[11] Tamaño de imágenes a procesar número de imágenes a procesar en cada uno de los pasos
[12] Parámetro que acelera el optimizador en la dirección relevante y amortigua las oscilaciones
[13] Parámetro de permite que los cambios en el optimizador no cambien de manera radical, evitando pasar por alto un punto con resultados óptimos.
[14] Parámetro que evita que el modelo tenga un sobreajuste durante la etapa de entrenamiento
[15] Cantidad de veces que se ejecuta el modelo
[16] Parámetro que le informa a la red que tan rápido debe realizar ajustes para acercarse a una solución óptima

El entrenamiento de la red se lo realiza en imágenes completas, a múltiples escalas, aumentando los datos, usando *batch_normalization*[17] y todo tipo de parámetro estándar. Para obtener un mejor rendimiento utiliza clasificadores logísticos independiente en lugar de utilizar la función *softmax*, además para la función de pérdida utiliza *binary cross entropy*[18] para todas las clases.

YOLOv3 predice cuadros en 3 escalas diferentes, extrayendo sus características, similar a la definición de redes piramidales, de esta manera las predicciones realizadas para la tercera escala serán más precisas porque contarán con los cálculos realizados en las dos escalas anteriores. Las escalas se eligen de manera aleatoria, además se eligen 9 grupos para dividirlos de forma uniforme entre las 3 escalas. (Redmon & Farhadi, 2018)

4.7 Segmentación

La segmentación hace referencia a la división de la imagen en distintas regiones que contienen ciertas propiedades de interés. A estas regiones se les denomina objetos y el resultado obtenido es la separación de los mismos. (Sánchez Salazar, 2017)

Para segmentar una imagen se puede hacer uso de distintas características, como el color, forma o textura, con esto se forma una idea acerca del objeto que se desea detectar. (Escriche, 2017)

Para que una segmentación sea perfecta todos los píxeles que se asignen a la región de interés deberían ser parte del objeto correcto, pero esto no sucede en la práctica, por ello una vez realizada la segmentación, se dan a conocer las regiones y las discontinuidades entre las mismas. Después de esto las regiones se utilizan para encontrar información importante de los objetos que se encuentran en la imagen y que pueden ser parte del objeto correcto. (Jiménez Ochoa, 2015)

[17] Normaliza las activaciones de la capa anterior en cada batch, es decir mantiene cerca de 0 la activación media y cerca de 1 la desviación estándar.
[18] Pérdida utilizada cuando se maneja clasificadores logísticos y los objetivos son escalares.

4.7.1 *Técnicas de Segmentación*

- **Basados en Puntos. -** Se describen como una técnica en la cual se utiliza la diferencia que existe entre el píxel central de interés y sus vecinos, tomando en cuenta que cada píxel es un punto aislado siempre que tenga una diferencia significativa con los vecinos que lo rodean.
- **Basados en Crecimiento de Regiones. -** Para segmentar una imagen con esta técnica se parte del centro del objeto de interés y se va aumentando la región hasta que se encuentra con los puntos exteriores finales del mismo, es decir hasta determinar los bordes que lo limitan. Esta técnica se repite tantas veces como objetos de interés existan en la imagen.
- **Basados en Bordes**. - En esta técnica se identifica un objeto usando los bordes que lo delimitan, es decir se determina cada límite de los segmentos presentes en la imagen. (Párraga Párraga & Casa López, 2017)

5. MATERIALES Y MÉTODOS

Los algoritmos de detección y segmentación de imágenes que se pretenden implementar a lo largo de este proyecto de investigación tienen como objetivo fundamental la localización de rostros humanos en una imagen previamente conocida, cuyo formato puede ser jpg, jpeg, png, pgm, entre otros.

De manera general, se espera analizar los resultados obtenidos mediante una comparación del comportamiento de los diferentes modelos, es decir los falsos positivos, falsos negativos y verdaderos positivos de cada uno.

5.1 Lenguaje de Programación

En la actualidad existen varios lenguajes de programación utilizados en inteligencia artificial y en uno de sus campos de estudio que es visión por computador. El lenguaje utilizado dependerá de la aplicación que se desee ejecutar y la funcionalidad deseada. A continuación, en la tabla 1 se muestra un resumen de los lenguajes más utilizados en visión por computador de acuerdo a la información presentada en la recopilación bibliográfica:

Tabla 1. Lenguajes de Programación utilizados en aprendizaje automático (machine learning)

Lenguaje de Programación	Características
Python	Python es uno de los lenguajes de programación que ha ganado popularidad en los últimos años, posicionándose en el primer lugar de los lenguajes utilizados para realizar algoritmos de aprendizaje automático. Gracias a una comunidad de desarrolladores de aprendizaje automático existen muchas bibliotecas preconstruidas para el aprendizaje automático en Python. Estas son algunas de las bibliotecas importantes que ayudan a acelerar el desarrollo de inteligencia artificial: Scikit-Learn y TensorFlow.
C++	C ++ es uno de los lenguajes de programación más antiguos y populares. La mayoría de las plataformas de aprendizaje automático admiten C ++, incluido TensorFlow.
JavaScript	JavaScript es el lenguaje de programación de secuencias de comandos web más popular. Aunque sus aplicaciones en el aprendizaje automático han sido limitadas, los proyectos de alto perfil como Tensorflow.js de Google se basan en JavaScript.
Java	Es un lenguaje de programación de inteligencia artificial que puede ejecutarse en plataformas compatibles sin necesidad de recompilación. Algunas de las aplicaciones para inteligencia artificial son: WEKA y JOONE
C#	C# es un lenguaje de programación simple, flexible, orientado a objetos, seguro y de código abierto. Permite a los desarrolladores crear todo tipo de aplicaciones, las bibliotecas utilizadas para aprendizaje automático son: Agentes ML, ML.NET y Accord.NET

Debido a las características mencionadas anteriormente acerca de los lenguajes de programación se optó por utilizar Python, ya que posee varias bibliotecas que facilitan el desarrollo y la ejecución de los algoritmos de visión por computador utilizados para el tema de detección facial. En la figura 13 se puede observar que Python encabeza la lista de los lenguajes de programación seleccionados para realizar aprendizaje automático.

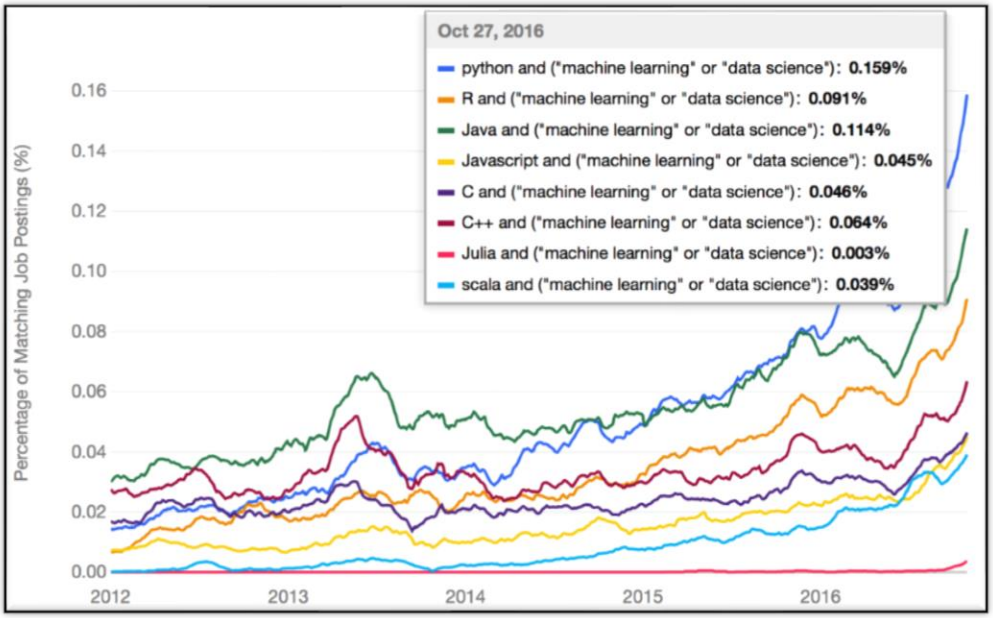


Figura 13. Lenguajes de programación más utilizados para realizar aprendizaje automático (machine learning)

Fuente: ("The Most Popular Language For Machine Learning Is ... (IT Best Kept Secret Is Optimization)," 2016)

Las tablas de valoración realizadas acerca de los lenguajes de programación se presentan en el Anexo 5.

5.2 Técnicas de normalización

De acuerdo con la información recopilada en la bibliografía existen diferentes técnicas para realizar pre-procesamiento en las imágenes, manipulando características presentes en las mismas como son: contraste, iluminación, escala, ángulo, compresión, etc.

Entre las técnicas elegidas para realizar el pre-procesamiento (normalización) de imágenes se utilizaron ecualización del histograma y corrección gamma, ya que suelen ser las más comunes y fáciles de ejecutar para acentuar las principales características correspondientes al rostro humano, logrando uniformidad en la iluminación y manipulando el contraste respectivamente.

Otra de las técnicas por las que se optó fue la de escala, pues se pretendía obtener una imagen con mayor resolución en la cual los algoritmos pudieran identificar de mejor manera los rostros existentes, mejorando así, su tasa de detección. Por último, se utilizó la técnica de rotación para forzar a los algoritmos a trabajar en circunstancias complicadas y constatar hasta qué punto son capaces de detectar un rostro de manera correcta.

5.3 Descripción General

Inicialmente se planteó el desarrollo de un algoritmo de detección y segmentación de imágenes basado en una red neuronal convolucional (CNN), el mismo que se entrenó con imágenes de distintas bases de datos tales como: MIT, ExtendedYaleB y BioID; este entrenamiento tiene como finalidad obtener pesos adecuados que permitan modelar las características que definen un rostro. Para evaluar el algoritmo clasificador se utilizó un proceso de ventana deslizante.

Las distintas etapas del algoritmo a implementar se realizaron utilizando la distribución de código abierto Anaconda, la cual a su vez desarrolla sus *scripts* en el lenguaje de programación Python.

En la figura 14 se muestra el procedimiento empleado en el desarrollo del algoritmo.

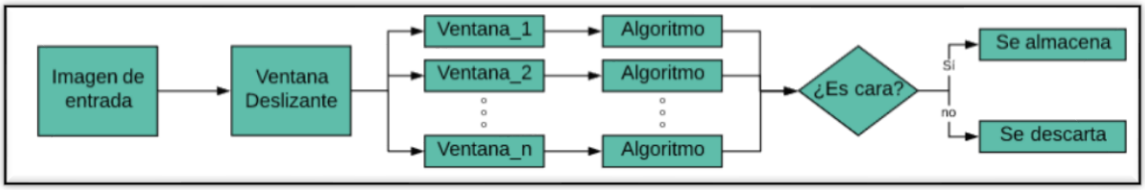


Figura 14. Esquema de funcionamiento del algoritmo desarrollado por el autor

Fuente: Autor

A continuación, se analizó el desempeño de tres modelos dedicados exclusivamente a detección de rostros: Viola & Jones, MTCNN y Yolo; además, las imágenes de entrada serán modificadas mediante una etapa de preprocesamiento, en la cual se aplicará ecualización del histograma, corrección gamma, escalado y rotación para determinar si existe una mejora en los resultados obtenidos.

Finalmente, para elegir el algoritmo (modelo) que se implementará, se tomará en cuenta la cantidad de rostros detectados y el tiempo empleado en detectar dichos rostros. Para visualizar con mayor claridad los datos obtenidos en la segmentación se creará un

directorio por cada imagen de entrada, dentro del cual se almacenará cada rostro detectado en un archivo de imagen diferente. En el apartado de representación se describirá detalladamente la organización de las imágenes.

5.3.1 *Recopilación Bibliográfica*

Como se mencionó en el apartado 4.5 concerniente a extracción de características, existen dos enfoques para el proceso de detección facial, dentro del mismo se pueden evidenciar varios detectores acordes a dicha clasificación; el primer enfoque, se encuentran los basados en características en los que es necesario normalizar los datos de entrada para lograr una detección óptima, además estos no ofrecen robustez ante distintos tipos de orientación u obstrucción en los que se encuentre el rostro; el segundo, basado en imágenes, presenta mejores respuestas en una imagen en la que el rostro se encuentre obstruido por algo y el entrenamiento para esta clase de algoritmos no requiere de datos que se hayan normalizado previamente

De acuerdo a la información establecida en el Anexo 6 se seleccionó como algoritmos de investigación Viola & Jones y ANN; dentro de este último se optó por utilizar los modelos: MTCNN (algoritmo recientemente desarrollado enfocado en detección facial y puntos faciales) y YOLO (algoritmo utilizado en detección de objetos).

Así mismo, para determinar la fiabilidad y eficiencia de los estándares del algoritmo desarrollado y de los algoritmos analizados es necesario realizar una recopilación de datos, comparando los porcentajes de precisión de dichos algoritmos enfocados a la detección y segmentación de rostros como son Viola & Jones con un porcentaje promedio de 88.8%, MTCNN con 89,8% y YOLO con 93,8%. Además, se debe tomar en cuenta las diferentes bases de datos utilizadas para entrenamiento, validación y prueba en cada uno de los algoritmos.

5.3.1.1 *Recopilación de datos*

En la última década se han realizado diferentes análisis de los algoritmos utilizados para detección y segmentación de imágenes, en los cuales se detallan los resultados adquiridos luego de realizar varias pruebas con distintas bases de datos en las que varía la cantidad de imágenes, incluso superando las 200000 en el caso de CelebA, estos conjuntos de

imágenes contienen distintos rostros humanos con diferentes ángulos de rotación, escala, expresión, pose, oclusión, iluminación, cantidad de rostros, e inclusive algunos rostros se encuentran maquillados. Estos resultados se encuentran entre el 80.5% y 99% de precisión.

5.3.1.2 *Análisis de Bases de Datos*

Como se explicó anteriormente se han realizado pruebas con distintas bases de datos, y de éstas también depende el porcentaje de precisión de los algoritmos desarrollados, por ello a continuación en la tabla 2 se detallan las bases de datos que se utilizaron para el desarrollo de los algoritmos planteados:

Tabla 2. Características de Bases de Datos

Nombre / Características	BioID	Extended Yale B	MIT	Wider Face	Celeb A
# Fotos	1 521	16 128	5 240	32 203	202 599
# **de caras**	1 521	16 128	5 240	393 703	202 599
Dimensión	384 x 286	168 x 192	200 x 200	1024 x Variada	Variada
Fondo	x		x		x
Rotación			x		
Iluminación	x	x	x	x	
Pose		x	x	x	x
Expresión				x	x
Escala	x			x	
Maquillaje				x	
Oclusión				x	

Para el proceso de aprendizaje de caras en el modelo desarrollado por la autora se decidió utilizar las bases de datos BioID (Universität Erlangen-Nürnberg (FAU), 2001), Extended

Yale B (Georghiades, Belhumeur, & Kriegman, 2001) y MIT (Massachusetts Institute of Technology, 2005) para la etapa de entrenamiento, validación y test. Además, las imágenes fueron escaladas previamente a una dimensión de 24 x 24 píxeles para reducir el tiempo de ejecución del algoritmo.

Por otro lado, para que el algoritmo sea capaz de distinguir las imágenes de no caras se trabajó con dos subconjuntos de la base de datos PICS, entre las cuales están las de *Manmade objects* con 72 imágenes de distintos objetos y PCA *images* con 91 imágenes clasificadas en distintos escenarios de la naturaleza, cada una de las imágenes cuenta con resolución de 256x256 píxeles. (University of Stirling, n.d.)

Con respecto a los otros modelos analizados, como lo son MTCNN y YOLO estos fueron entrenados con la base de datos WIDER FACE, además MTCNN también se entrenó con la base de datos CelebA para determinar los puntos faciales en los posibles rostros.

5.4 Fundamentos

5.4.1 *Bases de Detección Facial*

La detección es una de las principales tareas que se debe tomar en cuenta en la visión por computador, ya que para el desarrollo de la inteligencia artificial se debe alcanzar niveles de respuesta semejantes a los de un ser humano.

La detección facial se debe conseguir a pesar de que en una imagen se presenten varios rostros, con distintos niveles de iluminación, con varios ángulos o en lugares donde no exista un fondo controlado. Debido a esto entre los procesos que se debe seguir para lograr una detección óptima se encuentran: segmentar la imagen, extraer las características correspondientes a un rostro y verificar a través de un clasificador si dichas características son en realidad de un rostro humano.

5.4.2 *Algoritmos*

De acuerdo a la revisión bibliográfica y a lo planteado en el apartado 5.3.1 los algoritmos más utilizados para detección facial son los de Viola & Jones y Redes Neuronales Convolucionales.

5.4.2.1 *Red Neuronal Convolucional (CNN, Convolutional Neural Network)*

Las redes neuronales convolucionales son capaces de aprender a identificar características distintivas que definan un objeto, para este caso en particular los rostros, esto se logra a través de una estructura compuesta por varias capas ocultas con una jerarquía predeterminada, lo que involucra que las primeras capas detecten líneas o curvas y posteriormente las capas más profundas se especialicen hasta ser capaces de detectar un rostro.

Dado que la CNN necesita reconocer rostros en distintos ambientes, expresiones, poses, entre otras, deberá entrenarse con una gran cantidad de imágenes, en este caso la base de datos total cuenta con 23 765 imágenes de caras y 15 000 no caras; de esta manera el sistema será capaz de generalizar las características únicas correspondientes a los rostros.

El modelo fue desarrollado en una plataforma de código abierto utilizada para *machine learning*, llamada TensorFlow y desde aquí se llamó el API de Keras; es decir, se utilizó Keras para construir el modelo del sistema y se lo ejecutó sobre TensorFlow.

Primeramente, en el código se importó todas las librerías necesarias para manejar las imágenes y entrenar la red, dentro de estas se encuentran los bloques constructivos del modelo como son las capas de Convolución, Activación, MaxPooling, Optimizadores, Dropout, etc.

Luego se procedió a cargar todo el conjunto de datos en memoria para su posterior procesamiento, se creó etiquetas tanto para caras como para no caras, siendo la etiqueta 0 para caras y 1 para no caras; para obtener los conjuntos de entrenamiento, validación y prueba se dividió dicho conjunto en primer lugar en 80% para entrenamiento y 20% para prueba, a su vez el 80% de entrenamiento se divide en 80% entrenamiento y 20% validación.

Para realizar el preprocesamiento de las imágenes, se normalizó los valores de los píxeles para que contengan valores entre 0 y 1 únicamente; además, a las etiquetas se les aplicó un tipo de codificación conocido como *one hot encoding*[19], de esta manera la etiqueta 0

[19] Proceso por el cual una variable categórica, pasa a tener una forma de matriz binaria de acuerdo a las categorías que se proporcionen a un algoritmo de aprendizaje automático para realizar un trabajo de predicción.

se convirtió a 00 y la etiqueta 1 se convirtió a 01, este proceso se realiza para que la red neuronal incremente su eficiencia y la predicción sea más sencilla.

A partir de aquí se inició el desarrollo del modelo de la red; en primer lugar, se definió el *learning rate*, *epochs* y *batch size* con valores de $1e^{-3}$, 50 y 100 respectivamente; luego se planteó el modelo Secuencial de Keras, en el cual se agregaron cuatro capas de convolución con su respectiva función de activación LeakyReLU y su función de agrupamiento *Maxpooling*; así mismo, se agregó una capa Flatten en la que se aplanan todos los valores, una capa Dense, una capa Dropout con valor de 0,5 para descartar el 50% de los pesos en cada época y por ultimo una capa de salida con activación Softmax para que corresponda con la codificación *one hot encoding* de las etiquetas, este proceso se puede visualizar con mayor detalle en la figura 15.

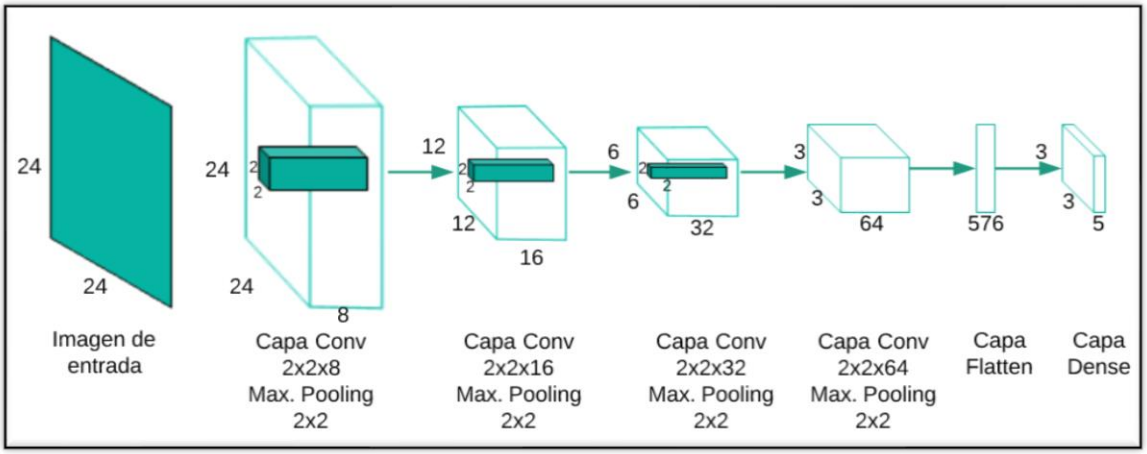


Figura 15. Arquitectura de red neuronal convolucional desarrollada por la autora

Fuente: Autor

Continuando con el desarrollo del modelo se planteó el optimizador Adam[20], la función de pérdida de entropía cruzada y una métrica para determinar el rendimiento del modelo, en este caso, la precisión; estos datos se colocaron en el bloque de compilación de la red. Para comenzar con el entrenamiento y validación del sistema se utilizará el bloque denominado fit(), en el que se colocarán los sets de datos de entrenamiento, validación, el tamaño del *batch* y la cantidad de *epochs*. Finalmente se guardó el modelo en un archivo de extensión .json y a su vez los pesos con los que se obtuvieron los mejores

[20] Método para optimización estocástica

resultados en un archivo de extensión .h5, este proceso se realizó para utilizar la red en un programa posterior sin necesidad de entrenarla nuevamente.

Para la ejecución de la red neuronal convolucional en una imagen de entrada cualquiera, se debe implementar en el modelo la definición de ventana deslizante, la cual se desplazará a través de la imagen obteniendo ventanas que serán utilizadas como entrada del sistema para predecir la existencia de un rostro dentro de las mismas, almacenándolo en caso de ser detectado como positivo.

5.4.2.2 *Viola & Jones*

El algoritmo de Viola & Jones se desarrolló para realizar una detección de rostros eficiente y rápida; es decir, su meta era conseguir una detección que se utilice en tiempo real, ya sea en cámaras, videos, etc., sin necesidad de tener una computadora con niveles de procesamiento muy elevados. Las tres etapas principales de este algoritmo para detectar un rostro se muestran en la figura 16.

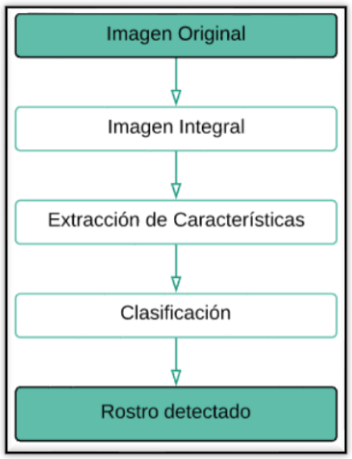


Figura 16. Etapas del algoritmo de detección Viola & Jones

Fuente: Autor

Para determinar el rendimiento del algoritmo se utilizó diferentes tipos de filtros, escalas y número de vecinos en las imágenes originales sin previa normalización; entre los filtros se encontraban los denominados *"profile face"* y "*frontal face"*, los cuales fueron desarrollados basándose en el modelo de Viola & Jones y que pueden ser implementados por la librería cv2 de Python; la diferencia fundamental entre ellos radica en la posición de los sujetos a detectar, siendo el primero enfocado a rostros de perfil y el segundo a rostros frontales.

En adición, el filtro *frontal face* a su vez tiene dos variantes que son "*default"* y "*alt",* todos estos parámetros se variaron en cada imagen hasta obtener los mejores resultados en cuanto a tasa de detección del algoritmo.

Simultáneamente, debido a que el algoritmo retorna cuatro puntos correspondientes al cuadro delimitador: la coordenada en el eje de las abscisas del vértice superior izquierdo, la coordenada en el eje de las ordenadas del mismo vértice, el ancho y alto. A partir de dichos puntos se graficó en cada imagen el cuadro delimitador de los rostros detectados.

Para finalizar el análisis de las imágenes sin normalizar se procedió a tabular los resultados tomando en cuenta también el tiempo de ejecución del algoritmo; estos resultados permitieron determinar los parámetros con los que el algoritmo presentaba una mayor eficiencia en distintas fotografías. Dichos parámetros serán utilizados en imágenes previamente normalizadas.

5.4.2.3 *MTCNN*

Este modelo de red neuronal está formado por 3 subredes primordiales conectadas en cascada como se muestra en la figura 17; la última subred denominada O-Net se modificó debido a que realizaba un procesamiento adicional sobre los datos para localizar puntos faciales dentro del rostro, proceso que no es de relevancia en este tema de investigación.

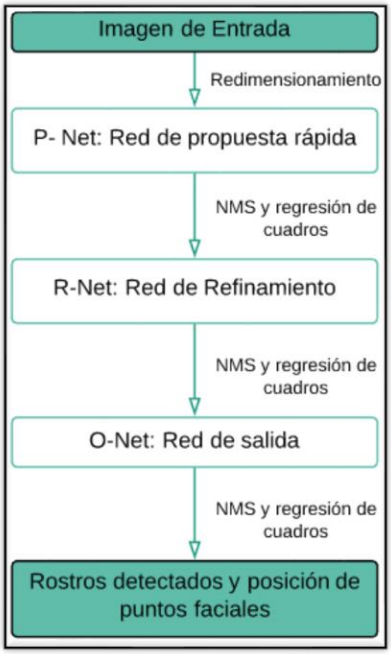


Figura 17. Etapas del modelo de detección MTCNN, NMS se refiere a supresión no máxima

Fuente: Autor

El modelo MTCNN se ejecuta a través de *Jupyter Notebook* en donde se carga la estructura del modelo y los pesos que se obtuvieron como resultado del proceso de entrenamiento.

Para realizar el proceso de segmentación se hace uso de las coordenadas del cuadro delimitador que el modelo obtiene como resultado; dichas coordenadas son graficadas a través de la librería OpenCV de Python, además estos cuadros son almacenados con formato de imagen en su respectiva carpeta.

Finalmente, se presenta la probabilidad de que en cada cuadro delimitador exista un rostro; esta probabilidad es el resultado del procesamiento efectuado por la red sobre las imágenes. Este modelo se utilizó tanto para las imágenes originales como para las normalizadas, el análisis que se efectuó se detallará más adelante en la etapa de alineación.

5.4.2.4 *YOLO FACE*

A diferencia de MTCNN, este modelo se ejecuta directamente desde el terminal de comandos de Anaconda, para lo cual es necesario acceder a la carpeta donde se encuentra el archivo .py del modelo y ejecutarlo desde aquí ingresando la ruta de la imagen de entrada y la ruta de salida. Las etapas principales que ejecuta este modelo se muestran en la figura 18.

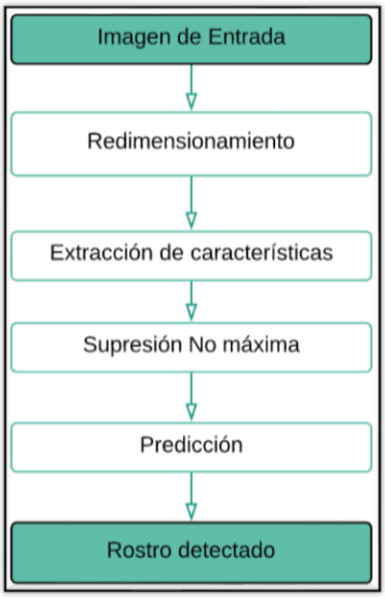


Figura 18. Etapas del modelo de detección YOLO

Fuente: Autor

Adicionalmente, se añadió líneas de comando en dicho archivo para realizar la segmentación de los cuadros delimitadores obtenidos como resultado de este modelo.

Al igual que el algoritmo de Viola & Jones y MTCNN, este modelo fue utilizado para imágenes sin normalizar y normalizadas; su análisis se explicará en la etapa de alineación.

5.5 Alineación

El análisis realizado en las imágenes normalizadas se desarrolló únicamente en los modelos Viola & Jones, MTCNN y YOLO; el modelo desarrollado por el autor no se utilizará para dicho análisis por motivos que se detallaran posteriormente en la sección de discusión.

En este apartado se indican los métodos de extracción de características utilizados por los diferentes modelos analizados para detección de rostros; iniciando con el método de Viola & Jones se debe mencionar que el entrenamiento del modelo se realizó utilizando solamente imágenes con rostros en posición frontal y con un nivel de iluminación adecuado para que las características de Haar puedan identificar con facilidad la diferencia de contraste en zonas específicas que determinan un rostro (frente, cavidad de los ojos, pómulos, etc.).

Las redes neuronales, en contraparte a los filtros Haar que están dados por valores conocidos, utilizan un kernel de valores aleatorios para determinar las características fundamentales que modelan un rostro, es por ello que los modelos MTCNN y YOLO a pesar de ser entrenados con la misma base de datos poseen diferentes pesos, por lo tanto, sus tasas de detección son diferentes.

Considerando la variación en los pesos y la estructura de cada red, existen rostros que solamente son detectados por uno u otro modelo debido a las características de detección en las que se especializó cada uno, entre las cuales se encuentran la forma del contorno del rostro, de la boca, de los ojos, la distancia entre ojos, distancia nariz – boca, etc.

Por otro lado, para calificar apropiadamente el rendimiento de los algoritmos en imágenes de varios rostros se utilizaron como métricas de error las siguientes:

- Precisión: Es la relación que existe entre el número de rostros detectados correctamente y todos los objetos detectados como rostros.

$$P = Precisión = \frac{VP(Verdadero\ Positivo)}{VP(Verdadero\ Positivo)+FP(Falso\ Positivo)} \quad (10)$$

- *Recall*: Es la relación que existe entre el número de rostros detectados correctamente y el total de rostros que verdaderamente existen en la imagen.

$$R = Recall = \frac{VP(Verdadero\ Positivo)}{VP(Verdadero\ Positivo)+FN(Falso\ Negativo)} \quad (11)$$

- FScore: Sean P y R la precisión y *recall*, respectivamente. El FScore es la media armónica de P y R.

$$FScore = 2\frac{P \times R}{P+R} \quad (12)$$

5.6 Representación

Para organizar adecuadamente las imágenes se desarrolló un algoritmo que se encarga de la creación de diferentes directorios; es decir, cuando el algoritmo detecta una imagen nueva en la ruta preliminarmente establecida, se crea un nuevo directorio (ubicado en el directorio fotos_ent) para dicha imagen, es decir, cada vez que se obtenga una fotografía nueva se irá creando un directorio con el nombre "foto_n", el valor de n se determinará por el número de fotografías que ya existan en el directorio aumentado su valor en uno, esto se realiza para que las fotos que puedan existir previamente no se eliminen.

En cuanto a las imágenes resultantes de la detección y segmentación de caras se ha creado previamente en el directorio de fotos de salida (fotos_sal), dos directorios denominados caras_detectadas y caras_segmentadas, en cada uno de estos se ubicarán respectivamente lo siguiente:

- La imagen original con un recuadro en cada uno de los rostros que detecte el algoritmo.
- Una imagen de cada rostro detectado, producto de la segmentación realizada en la imagen original, por ejemplo, si en una imagen se detectan 5 rostros, dentro del directorio caras_segmentadas y de acuerdo al nombre de la foto, se creará una imagen por cada uno de los 5 rostros detectados.

En la figura 19 se muestra un organigrama que detalla la distribución de los directorios creados por cada imagen.

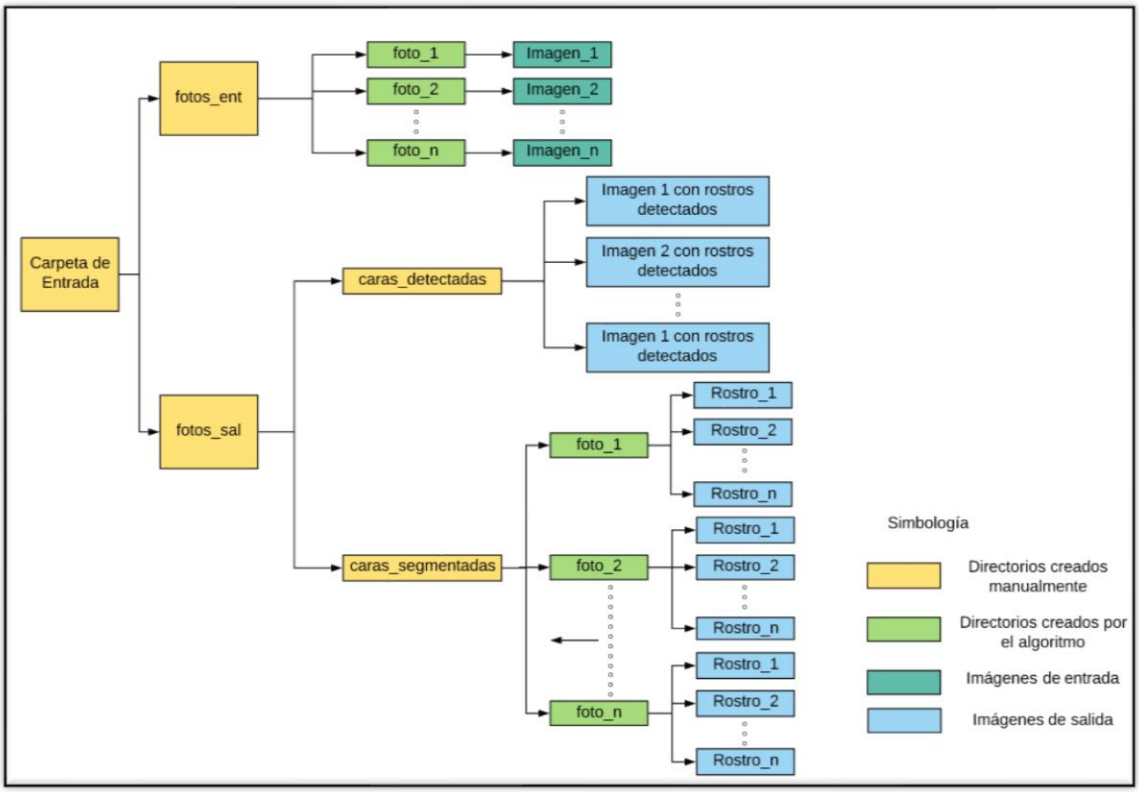


Figura 19. Directorios de las imágenes

Fuente: Autor

Una vez organizadas adecuadamente, en la etapa de ventana deslizante correspondiente al modelo desarrollado, las imágenes de entrada son procesadas para extraer su información, es decir sus niveles de intensidad en cada canal (RGB), para disminuir el cálculo computacional se ha decidido trabajar solamente con los niveles de intensidad en gris de la imagen. Además, cada ventana extraída de la etapa de ventana deslizante es redimensionada a 24×24 píxeles debido a que la entrada de la etapa clasificadora (CNN) necesita una imagen con estas características de dimensión y profundidad.

La reducción de dimensionalidad se efectúa de diferente manera en los modelos de Viola & Jones, MTCNN y YOLO.

En el modelo de Viola & Jones se necesita como entrada una matriz que posea los niveles de intensidad de gris de la imagen; en el caso de MTCNN y YOLO las imágenes no necesitan un procesamiento previo, es decir los modelos realizan cualquier redimensionamiento internamente.

6. RESULTADOS

Los presentes resultados se obtuvieron mediante una etapa de observación en los modelos Viola & Jones, MTCNN, YOLO y una red neuronal propuesta por la autora; la cual se ejecutó en una computadora con un procesador Inter Corei5 de 1,70 GHz de capacidad de procesamiento, 6 GB de memoria RAM, disco duro de 500 GB HDD sin tarjeta gráfica.

6.1 Algoritmo CNN

Los resultados obtenidos por el algoritmo desarrollado por la autora no fueron los esperados; es decir en imágenes que contengan varios rostros, el modelo detectó un gran número de falsos positivos a causa del *overfitting* existente en la red. En la figura 20 se muestran los parámetros de evaluación del rendimiento del modelo, a la izquierda se encuentra la precisión del entrenamiento obtenido en cada *epoch*; es necesario mencionar que desde *epochs* iniciales este porcentaje es elevado (93,63%) verificando de esta manera que la red se sobreajustó a los datos de entrenamiento. Así mismo, a la derecha se muestra la pérdida categórica del modelo en cada *epoch* la misma que es inversamente proporcional a la precisión.

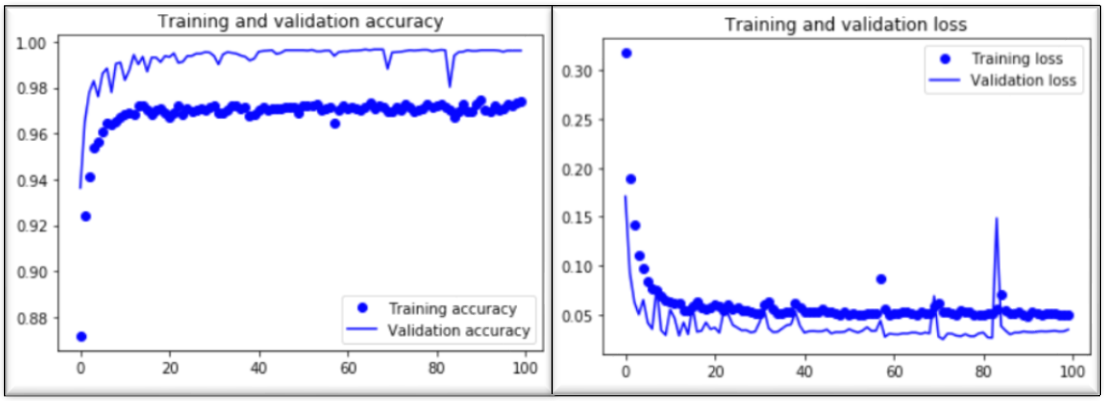


Figura 20. Parámetros de evaluación del modelo desarrollado por el autor

Fuente: Autor

A continuación, en las figuras 21 y 22 se muestran ejemplos de verdaderos positivos y falsos positivos detectados por este modelo respectivamente.

Figura 21. Verdaderos Positivos obtenidos en el modelo desarrollado por el autor

Fuente: Autor

Figura 22. Falsos positivos obtenidos en el modelo desarrollado por el autor

Fuente: Autor

6.2 Imágenes De Un Solo Rostro

6.2.1 *Imágenes Sin Normalización Previa*

En las tablas 3, 4 y 5 se muestran los resultados obtenidos en los modelos de Viola & Jones, MTCNN y YOLO respectivamente.

Tabla 3. Parámetros de análisis para el modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Esc.	Vecinos	Tipo Filtro	Tiempo ejecución (s)	N° Caras detectadas
1	1024 × 680	1	0	0	1.1	2	alt	0.459	1
2	1024 × 1539	1	0	0	1.2	2	alt	0.489	1
3	1024 × 879	1	0	0	1.2	2	profile	0.47	1
4	1024 × 1021	1	0	0	1.4	2	alt	0.234	1
5	1024 × 732	1	0	0	1.4	2	alt	0.159	1
6	1024 × 1538	1	0	0	1.4	2	alt	0.327	1
7	1024 × 1185	1	0	0	1.4	2	alt	0.275	1
8	1024 × 640	0	0	1	1.4	2	alt	0.167	0
9	1024 × 576	1	0	0	1.4	2	alt	0.125	1
10	1200 × 800	0	0	1	1.1	2	default	0.518	0
11	500 × 500	1	0	0	1.1	2	default	0.412	1
12	682 × 457	1	0	0	1.1	4	alt	0.271	1
13	640 × 391	1	0	0	1.3	3	alt	0.109	1
14	1140 × 530	0	0	1	1.3	2	alt	0.201	0
15	604 × 835	1	0	0	1.5	3	alt	0.133	1

Los parámetros que obtuvieron los mejores resultados en diferentes fotografías en cuanto a escala, vecinos y tipo de filtro son 1.4, 3 y alt respectivamente.

Tabla 4. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° de caras detectadas
1	1024 × 680	1	0	0	2.862	1
2	1024 × 1539	1	0	0	3.605	1
3	1024 × 879	1	0	0	2.886	1
4	1024 × 1021	1	0	0	3.169	1
5	1024 × 732	0	0	1	2.730	0
6	1024 × 1538	1	0	0	3.615	1
7	1024 × 1185	1	1	0	2.904	2
8	1024 × 640	0	0	1	2.449	0
9	1024 × 576	1	0	0	2.439	1
10	1200 × 800	0	0	1	2.519	0
11	500 × 500	0	1	1	2.474	1
12	682 × 457	1	1	0	2.103	2
13	640 × 391	1	1	0	2.563	2
14	1140 × 530	0	0	1	2.397	0
15	604 × 835	1	0	0	2.634	1

Tabla 5. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	1	0	0	4.840	1
2	1024 × 1539	1	0	0	2.029	1
3	1024 × 879	1	0	0	2.087	1
4	1024 × 1021	1	0	0	1.983	1
5	1024 × 732	1	0	0	2.144	1
6	1024 × 1538	1	0	0	2.022	1
7	1024 × 1185	1	0	0	2.142	1
8	1024 × 640	0	0	1	2.055	0
9	1024 × 576	1	0	0	2.316	1
10	1200 × 800	1	0	0	2.109	1
11	500 × 500	1	0	0	1.922	1
12	682 × 457	1	0	0	1.906	1
13	640 × 391	1	0	0	2.447	1
14	1140 × 530	1	0	0	1.937	1
15	604 × 835	1	0	0	2.286	1

Analizando las tablas mostradas anteriormente se determinó que en la imagen 8 ninguno de los modelos detectó el rostro presente en la fotografía debido a que éste tiene un ángulo de rotación cercano a 180°.

En adición, se puede observar que el algoritmo Viola & Jones presenta menor tiempo de ejecución en comparación con los modelos restantes, dicho tiempo no supera 0.518 segundos.

6.2.2 *Imágenes aplicando Ecualización del Histograma*

Los modelos MTCNN y YOLO obtuvieron mejores resultados con la ecualización de imágenes en blanco y negro, mientras que el modelo Viola & Jones con la ecualización de imágenes a color. Por esta razón se ha decidido presentar en las tablas 6, 7 y 8 los resultados de la ecualización más eficiente en cada modelo. En la figura 23 se muestra una imagen a la que se le ha aplicado ecualización del histograma, con los cuadros de detección correspondientes a cada modelo.

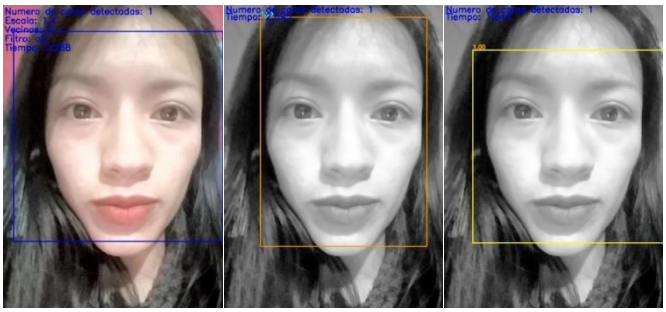


Figura 23. Imágenes normalizadas con ecualización del Histograma para los modelos Viola & Jones, MTCNN y YOLO de izquierda a derecha

Fuente: Autor

Tabla 6. Parámetros de análisis para el modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	0	0	1	0.301	0
2	1024 × 1539	0	0	1	0.675	0
3	1024 × 879	0	0	1	0.378	0
4	1024 × 1021	1	0	0	0.522	1
5	1024 × 732	1	0	0	0.275	1
6	1024 × 1538	1	0	0	0.622	1
7	1024 × 1185	1	0	0	0.535	1
8	1024 × 640	0	0	1	0.420	0
9	1024 × 576	1	0	0	0.182	1
10	1200 × 800	0	0	1	0.483	0
11	500 × 500	0	0	1	0.284	0
12	682 × 457	0	0	1	0.242	0
13	640 × 391	0	0	1	0.195	0
14	1140 × 530	0	0	1	0.322	0
15	604 × 835	1	0	0	0.268	1

Tabla 7. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° de caras detectadas
1	1024 × 680	1	0	0	2.808	1
2	1024 × 1539	0	0	1	4.005	0
3	1024 × 879	1	0	0	3.477	1
4	1024 × 1021	1	0	0	3.336	1
5	1024 × 732	1	0	0	3.020	1
6	1024 × 1538	1	2	0	4.164	3
7	1024 × 1185	1	0	0	3.532	1
8	1024 × 640	0	0	1	3.431	0
9	1024 × 576	1	0	0	2.974	1
10	1200 × 800	0	0	1	3.301	0
11	500 × 500	1	0	0	3.007	1
12	682 × 457	1	0	0	2.671	1
13	640 × 391	1	0	0	2.665	1
14	1140 × 530	0	0	1	3.522	0
15	604 × 835	1	0	0	2.695	1

Tabla 8. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	1	0	0	1.844	1
2	1024 × 1539	1	0	0	1.875	1
3	1024 × 879	1	1	0	1.844	2
4	1024 × 1021	1	0	0	2.015	1
5	1024 × 732	1	0	0	1.844	1
6	1024 × 1538	1	0	0	1.859	1
7	1024 × 1185	1	0	0	1.859	1
8	1024 × 640	0	0	1	1.875	0
9	1024 × 576	1	0	0	1.828	1
10	1200 × 800	1	0	0	1.876	1
11	500 × 500	1	0	0	1.859	1
12	682 × 457	1	0	0	2.078	1
13	640 × 391	1	0	0	1.862	1
14	1140 × 530	1	0	0	1.859	1
15	604 × 835	1	0	0	1.844	1

6.2.3 *Imágenes Escaladas en Dimensión*

Así mismo para analizar el rendimiento del algoritmo en imágenes con una gran dimensión se decidió escalar la imagen en un factor de 24; los resultados de este proceso para cada modelo se muestran en las tablas 9, 10 y 11. En la figura 24 se muestran los cuadros de detección de cada modelo en la imagen de prueba.

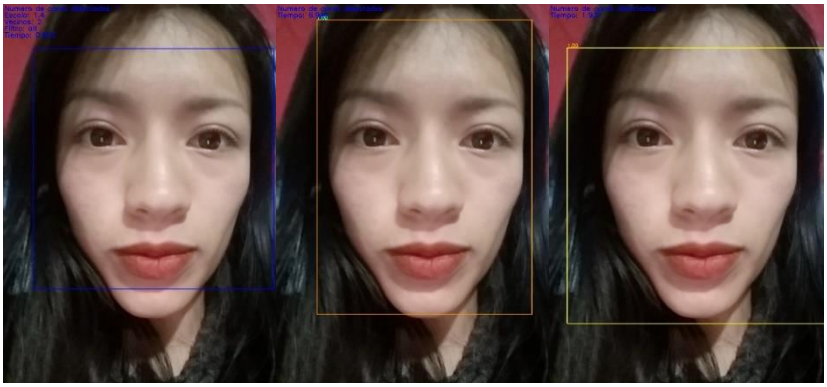


Figura 24. Imágenes normalizadas aplicando escalado en dimensión para los modelos Viola & Jones, MTCNN y YOLO de izquierda a derecha

Fuente: Autor

Tabla 9. Parámetros de análisis del modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	3072 × 2040	0	0	1	1.103	0
2	3072 × 4617	1	0	0	1.675	1
3	3072 × 2637	0	0	1	1.430	0
4	3072 × 3063	1	0	0	1.677	1
5	3072 × 2196	1	0	0	0.957	1
6	3072 × 4614	1	0	0	1.837	1
7	3072 × 3555	0	0	1	1.556	0
8	3072 × 1920	0	0	1	1.010	0
9	3072 × 1728	1	0	0	0.806	1
10	3600 × 2400	0	0	1	1.527	0
11	1500 × 1500	0	0	1	0.775	0
12	2046 × 1371	0	1	1	1.114	1
13	1920 × 1173	1	0	0	0.590	1
14	3420 × 1590	0	0	1	1.308	0
15	1812 × 2505	1	0	0	0.822	1

Tabla 10. Parámetros de análisis del modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras detectadas
1	3072 × 2040	0	0	1	6.106	0
2	3072 × 4617	1	0	0	14.325	1
3	3072 × 2637	1	0	0	9.348	1
4	3072 × 3063	1	0	0	9.707	1
5	3072 × 2196	0	0	1	9.227	0
6	3072 × 4614	1	0	0	14.280	1
7	3072 × 3555	1	1	0	11.971	2
8	3072 × 1920	0	1	1	7.243	1
9	3072 × 1728	1	0	0	6.294	1
10	3600 × 2400	0	0	1	9.071	0
11	1500 × 1500	0	1	1	4.448	1
12	2046 × 1371	1	0	0	5.087	1
13	1920 × 1173	1	1	0	4.261	2
14	3420 × 1590	0	1	1	8.363	1
15	1812 × 2505	1	0	0	6.947	1

Tabla 11. Parámetros de análisis del modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	3072 × 2040	1	0	0	1.962	1
2	3072 × 4617	1	0	0	2.081	1
3	3072 × 2637	1	0	0	2.000	1
4	3072 × 3063	1	0	0	2.000	1
5	3072 × 2196	1	0	0	1.953	1
6	3072 × 4614	1	0	0	2.062	1
7	3072 × 3555	1	0	0	2.015	1
8	3072 × 1920	1	0	0	2.140	1
9	3072 × 1728	1	0	0	1.937	1
10	3600 × 2400	1	0	0	2.000	1
11	1500 × 1500	1	0	0	1.875	1
12	2046 × 1371	1	0	0	1.906	1
13	1920 × 1173	1	0	0	2.109	1
14	3420 × 1590	1	0	0	2.143	1
15	1812 × 2505	1	0	0	1.937	1

Se puede observar que el tiempo de ejecución incrementó considerablemente debido al aumento en la dimensionalidad de la imagen, alcanzando en el modelo MTCNN valores exagerados de hasta 14, 325 segundos. Además, en este mismo modelo se incrementa el número de falsos positivos detectados debido a la existencia de varios conjuntos de píxeles que simulan ser un rostro.

6.2.4 *Imágenes Aplicando Corrección Gamma*

Se aplicaron diferentes factores de corrección gamma para determinar el rendimiento de los modelos con cada uno de ellos, en el presente trabajo, las imágenes se analizaron con factores de 0.25, 0.5, 5 y 10. A continuación en las tablas 12, 13 y 14 se mostrarán los resultados con el factor de corrección que obtuvo la tasa de detección más alta en cada modelo. Los resultados de los factores de corrección restantes se muestran en Anexo 3.

En la figura 25 se muestran los cuadros delimitadores que retorna cada modelo en caso de detectar un rostro en la imagen de prueba.

Figura 25. Imágenes normalizadas aplicando corrección gamma en los modelos Viola & Jones, MTCNN y YOLO de izquierda a derecha

Fuente: Autor

Tabla 12. Parámetros de análisis del modelo Viola & Jones con factor de corrección gamma igual a 5

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	0	0	1	0.397	0
2	1024 × 1539	1	0	0	0.442	1
3	1024 × 879	0	0	1	0.400	0
4	1024 × 1021	1	0	0	0.491	1
5	1024 × 732	1	0	0	0.365	1
6	1024 × 1538	1	0	0	0.471	1
7	1024 × 1185	1	0	0	0.480	1
8	1024 × 640	0	0	1	0.343	0
9	1024 × 576	1	0	0	0.245	1
10	1200 × 800	0	0	1	0.343	0
11	500 × 500	0	0	1	0.275	0
12	682 × 457	0	0	1	0.248	0
13	640 × 391	1	0	0	0.163	1
14	1140 × 530	0	0	1	0.386	0
15	604 × 835	0	0	1	0.249	0

Tabla 13. Parámetros de análisis del modelo MTCNN con factor de corrección gamma igual a 0.5

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° de caras detectadas
1	1024 × 680	0	0	1	2.693	0
2	1024 × 1539	1	0	0	3.372	1
3	1024 × 879	1	0	0	2.738	1
4	1024 × 1021	1	0	0	3.210	1
5	1024 × 732	0	0	1	2.840	0
6	1024 × 1538	1	0	0	3.856	1
7	1024 × 1185	1	0	0	3.322	1
8	1024 × 640	0	1	1	2.851	1
9	1024 × 576	1	0	0	2.692	1
10	1200 × 800	0	1	1	3.368	1
11	500 × 500	0	0	1	2.628	0
12	682 × 457	1	1	0	2.744	2
13	640 × 391	1	0	0	2.584	1
14	1140 × 530	0	0	1	3.146	0
15	604 × 835	1	0	0	2.499	1

Tabla 14. Parámetros de análisis del modelo YOLO con factor de corrección gamma igual a 0.5

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	1	0	0	1.962	1
2	1024 × 1539	1	0	0	1.906	1
3	1024 × 879	1	0	0	1.881	1
4	1024 × 1021	1	0	0	1.859	1
5	1024 × 732	1	0	0	1.844	1
6	1024 × 1538	1	0	0	1.859	1
7	1024 × 1185	1	0	0	2.156	1
8	1024 × 640	0	0	1	1.859	0
9	1024 × 576	1	0	0	1.859	1
10	1200 × 800	1	0	0	1.859	1
11	500 × 500	1	0	0	1.859	1
12	682 × 457	1	0	0	2.015	1
13	640 × 391	1	0	0	1.844	1
14	1140 × 530	0	0	1	1.844	0
15	604 × 835	1	0	0	1.859	1

6.2.5 *Imágenes Aplicando Rotación*

En las tablas 15, 16 y 17 se muestran los resultados al aplicar un ángulo de rotación de 45° en las imágenes; en la figura 26 se muestran, en caso de existir, los cuadros de detección de cada modelo.

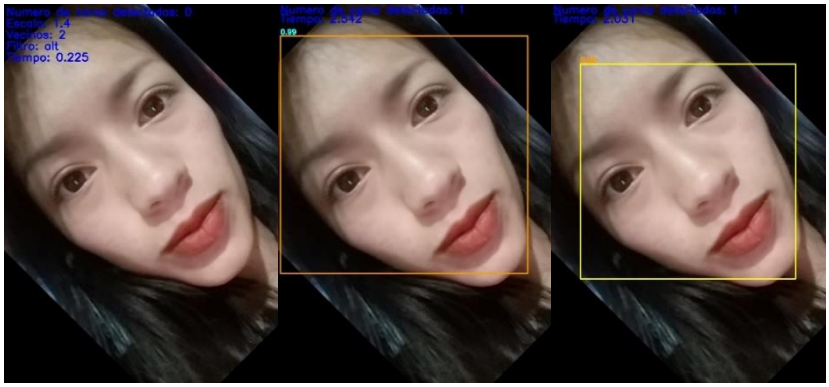


Figura 26. Imágenes normalizadas aplicando rotación

Fuente: Autor

Tabla 15. Parámetros de análisis del modelo Viola & Jones con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° de caras detectadas
1	1024 × 680	0	0	1	2.526	0
2	1024 × 1539	0	0	1	3.313	0
3	1024 × 879	1	0	0	2.632	1
4	1024 × 1021	0	0	1	3.336	0
5	1024 × 732	0	0	1	2.666	0
6	1024 × 1538	1	1	0	3.628	2
7	1024 × 1185	0	0	1	3.323	0
8	1024 × 640	0	0	1	2.497	0
9	1024 × 576	0	0	1	2.447	0
10	1200 × 800	0	0	1	2.701	0
11	500 × 500	0	0	1	2.361	0
12	682 × 457	1	0	0	2.170	1
13	640 × 391	0	0	1	2.084	0
14	1140 × 530	0	3	1	2.417	3
15	604 × 835	1	0	0	2.542	1

Tabla 16. Parámetros de análisis del modelo MTCNN con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	0	0	1	0.216	0
2	1024 × 1539	0	0	1	0.340	0
3	1024 × 879	0	0	1	0.284	0
4	1024 × 1021	0	0	1	0.328	0
5	1024 × 732	0	0	1	0.197	0
6	1024 × 1538	0	0	1	0.405	0
7	1024 × 1185	0	0	1	0.276	0
8	1024 × 640	0	0	1	0.213	0
9	1024 × 576	0	0	1	0.215	0
10	1200 × 800	0	0	1	0.301	0
11	500 × 500	0	0	1	0.176	0
12	682 × 457	0	0	1	0.266	0
13	640 × 391	0	0	1	0.130	0
14	1140 × 530	0	0	1	0.210	0
15	604 × 835	0	0	1	0.225	0

Tabla 17. Parámetros de análisis del modelo YOLO con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de ejecución (s)	N° Caras Detectadas
1	1024 × 680	1	0	0	1.890	1
2	1024 × 1539	1	0	0	1.890	1
3	1024 × 879	1	0	0	1.859	1
4	1024 × 1021	0	1	1	1.875	1
5	1024 × 732	1	0	0	2.015	1
6	1024 × 1538	1	0	0	1.859	1
7	1024 × 1185	1	0	0	1.861	1
8	1024 × 640	0	0	1	1.875	0
9	1024 × 576	0	0	1	1.828	0
10	1200 × 800	0	0	1	1.859	0
11	500 × 500	0	0	1	1.828	0
12	682 × 457	1	0	0	1.844	1
13	640 × 391	1	0	0	1.937	1
14	1140 × 530	0	0	1	4.672	0
15	604 × 835	1	0	0	2.031	1

Para el caso de imágenes con rotación mayor a 10°, el modelo de Viola & Jones no es capaz de detectar rostro alguno. Así mismo a partir de las tablas anteriores se puede apreciar una gran disminución en la tasa de detección de los modelos restantes.

A continuación, los datos obtenidos en los parámetros mencionados anteriormente en cada una de las tablas, se muestran de manera gráfica en las figuras 27, 28 y 29.

En la figura 27 se puede observar que el modelo YOLO posee una tasa de detección correcta más alta en todos los casos de análisis.

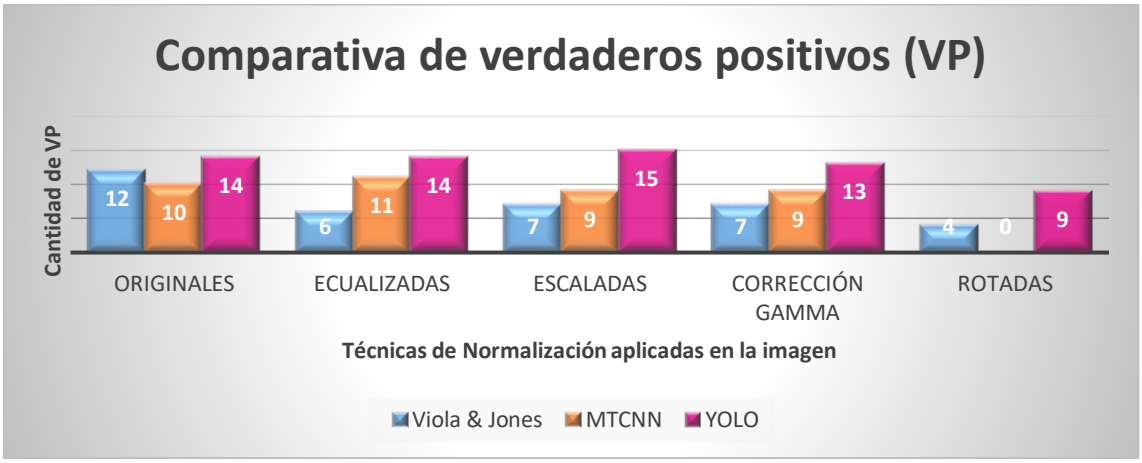


Figura 27. Tiempo de ejecución de cada modelo en los diferentes conjuntos de imágenes (originales y normalizadas)

Fuente: Autor

Por otro lado, en la figura 28 se puede apreciar que el modelo MTCNN obtuvo el mayor número de cuadros detectados incorrectamente como rostros; además, se debe mencionar que la tasa más alta se presentó en imágenes escaladas debido a la gran cantidad de píxeles que se procesan provocando que la información correspondiente al fondo pueda confundirse con las características de un rostro.

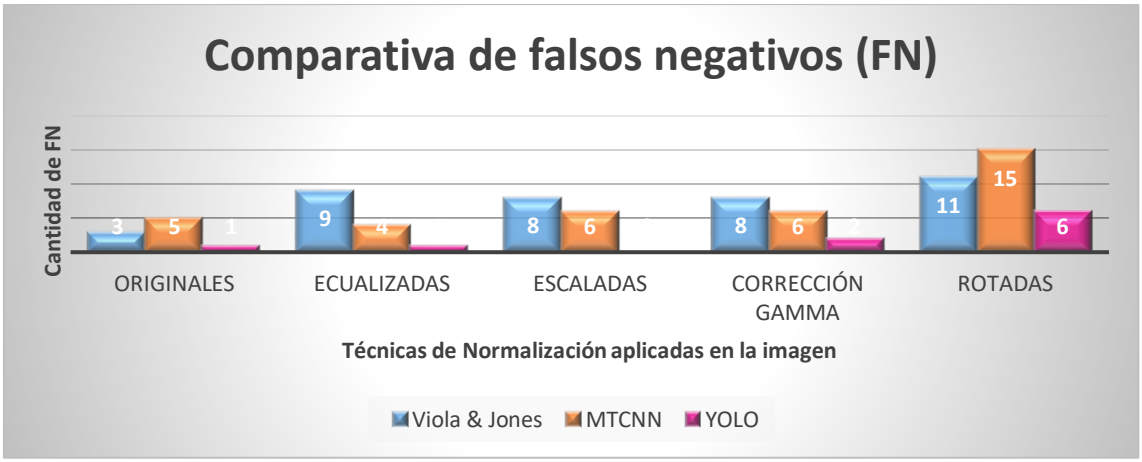


Figura 28. Tiempo de ejecución de cada modelo en los diferentes conjuntos de imágenes (originales y normalizadas)

Fuente: Autor

Por último, en la figura 29 se observa que el modelo que detecta una menor cantidad de rostros es Viola & Jones, detectando menos de la mitad del conjunto de imágenes en la mayoría de casos; sin embargo, cuando se ajustaron los parámetros de escala, filtro y número de vecinos en cada imagen se logró una tasa de detección aceptable. Por otro lado, el modelo MTCNN alcanza la tasa de falsos negativos más alta en imágenes con rotación.

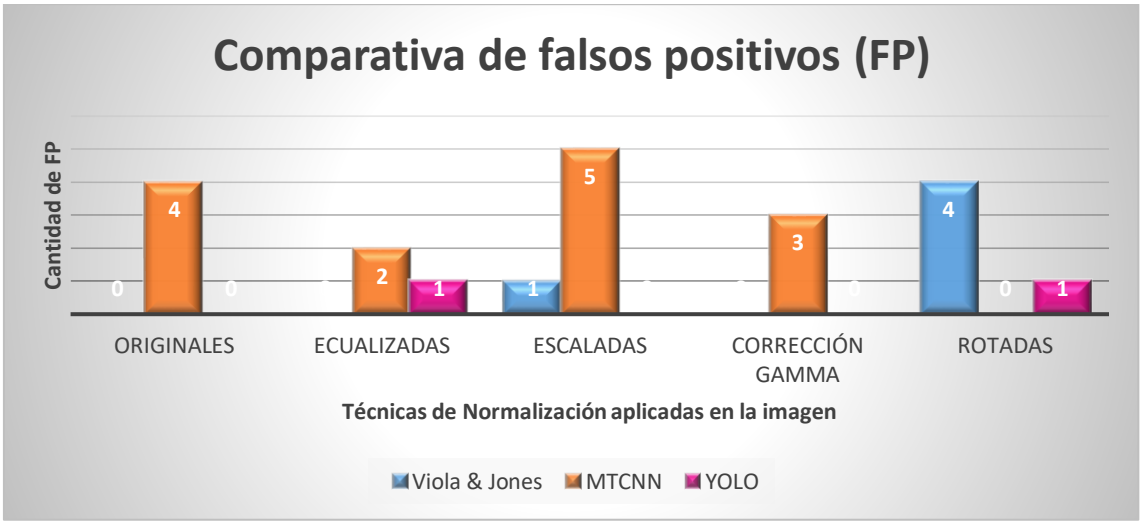


Figura 29. Tiempo de ejecución de cada modelo en los diferentes conjuntos de imágenes (originales y normalizadas)

Fuente: Autor

Para finalizar el estudio de las imágenes de un solo rostro, en la figura 30 se presenta una gráfica del tiempo total de ejecución que toma cada algoritmo para ejecutarse en los diferentes escenarios establecidos anteriormente, es decir, imágenes normalizadas y sin normalizar.

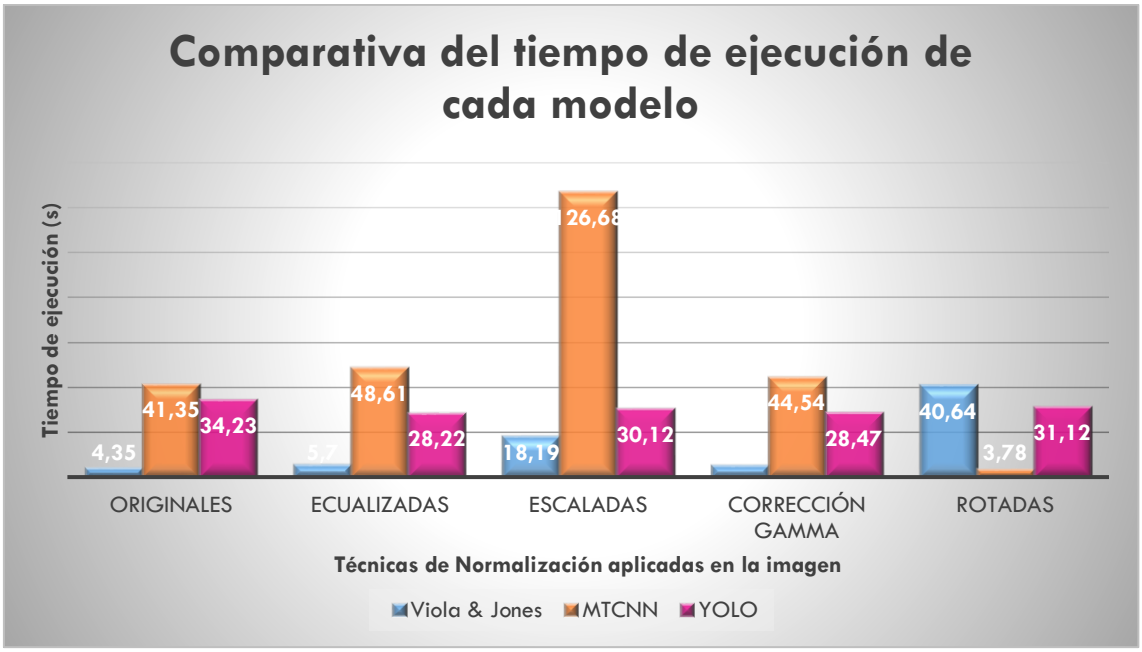


Figura 30. Tiempo de ejecución de cada modelo en los diferentes conjuntos de imágenes (originales y normalizadas)

Fuente: Autor

6.3 Imágenes De Varios Rostros

6.3.1 *Imágenes Sin Normalización Previa*

En las tablas 18, 19 y 20 se muestran los resultados obtenidos de la detección de rostros realizada por los modelos en imágenes con varios rostros y en diferentes ambientes.

Tabla 18. Parámetros de análisis para el modelo Viola & Jones

N°	Dim.	VP	FP	FN	Esc.	V.	Tipo Filtro	T. E. (s)	N° C. T.	P	R	FS
1	1024 × 1366	4	0	3	1.1	2	alt	1.196	7	1.00	0.57	0.73
2	1024 × 852	1	0	4	1.5	2	profile	0.514	5	1.00	0.20	0.33
3	1024 × 686	1	0	2	1.6	2	default	0.145	3	1.00	0.33	0.50
4	960 × 540	19	0	1	1.2	2	alt	0.267	20	1.00	0.95	0.97
5	960 × 540	7	0	1	1.3	2	default	0.290	8	1.00	0.88	0.94
6	1200 × 566	8	0	5	1.1	3	default	0.427	13	1.00	0.62	0.77
7	612 × 340	7	0	1	1.1	2	default	0.181	8	1.00	0.88	0.94
8	889 × 495	12	0	3	1.1	1	alt	0.544	15	1.00	0.80	0.89
9	760 × 450	7	1	4	1.1	4	alt	0.314	11	0.88	0.64	0.74
10	630 × 395	3	0	30	1.1	2	alt	0.221	33	1.00	0.09	0.17
11	1200 × 565	56	1	77	1.2	2	default	0.267	133	0.98	0.42	0.59
12	1600 × 800	1	2	5	1.1	2	alt	1.185	6	0.33	0.17	0.22
13	450 × 276	6	0	1	1.1	3	alt	0.127	7	1.00	0.86	0.92
14	1300 × 956	14	1	2	1.1	3	default	0.943	16	0.93	0.88	0.90
15	1300 × 957	2	0	9	1.1	3	alt	0.967	11	1.00	0.18	0.31
16	660 × 330	2	0	1	1.1	3	alt	0.197	3	1.00	0.67	0.80
17	1200 × 566	2	0	1	1.3	2	alt	0.275	3	1.00	0.67	0.80
18	259 × 194	0	0	7	1.3	1	default	0.272	7	0.00	0.00	0.00
19	300 × 168	2	0	5	1.1	1	alt	0.064	7	1.00	0.29	0.45
20	1280 × 722	25	0	2	1.1	2	alt	0.766	27	1.00	0.93	0.96
21	678 × 381	4	0	32	1.1	1	alt	0.213	36	1.00	0.11	0.20
22	635 × 290	7	0	3	1.1	2	alt	0.178	10	1.00	0.70	0.82
23	800 × 533	4	0	5	1.1	3	alt	0.322	9	1.00	0.44	0.61
24	945 × 612	8	0	8	1.1	3	alt	0.508	16	1.00	0.50	0.67
25	500 × 325	3	0	6	1.1	3	alt	0.224	9	1.00	0.33	0.50

De acuerdo con los resultados obtenidos para fotografías en las que se existe la presencia de varias caras, los parámetros a utilizarse para escala, vecinos y el tipo de filtro para las imágenes normalizadas serán de: 1.1, 3 y alt respectivamente.

Tabla 19. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C. T	P	R	FS
1	1024 × 1366	6	1	1	3.850	7	7	0.86	0.86	0.86
2	1024 × 852	3	1	2	3.273	4	5	0.75	0.60	0.67
3	1024 × 686	2	1	1	2.818	3	3	0.67	0.67	0.67
4	960 × 540	20	1	0	2.918	21	20	0.95	1.00	0.97
5	960 × 540	8	1	0	2.430	9	8	0.89	1.00	0.94
6	1200 × 566	11	0	2	2.489	11	13	1.00	0.85	0.92
7	612 × 340	8	0	0	2.149	8	8	1.00	1.00	1.00
8	889 × 495	8	1	7	2.669	9	15	0.89	0.53	0.66
9	760 × 450	7	0	3	2.201	7	11	1.00	0.70	0.82
10	630 × 395	3	0	30	2.222	3	33	1.00	0.09	0.17
11	1200 × 565	57	1	76	2.841	58	133	0.98	0.43	0.60
12	1600 × 800	4	2	2	4.382	6	6	0.67	0.67	0.67
13	450 × 276	6	0	1	2.011	6	7	1.00	0.86	0.92
14	1300 × 956	15	0	1	3.207	15	16	1.00	0.94	0.97
15	1300 × 957	6	0	5	3.242	6	11	1.00	0.55	0.71
16	660 × 330	1	0	2	2.041	1	3	1.00	0.33	0.50
17	1200 × 566	2	0	1	2.462	2	3	1.00	0.67	0.80
18	259 × 194	1	0	6	2.172	1	7	1.00	0.14	0.25
19	300 × 168	3	0	4	1.918	3	7	1.00	0.43	0.60
20	1280 × 722	26	1	1	3.369	27	27	0.96	0.96	0.96
21	678 × 381	14	0	22	2.424	14	36	1.00	0.39	0.56
22	635 × 290	8	0	2	2.007	8	10	1.00	0.80	0.89
23	800 × 533	4	0	5	2.525	4	9	1.00	0.44	0.61
24	945 × 612	8	0	8	2.729	8	16	1.00	0.50	0.67
25	500 × 325	6	0	3	2.601	6	9	1.00	0.67	0.80

Tabla 20. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	6	0	1	2.115	6	7	1.00	0.86	0.92
2	1024 × 852	4	0	1	1.996	4	5	1.00	0.80	0.89
3	1024 × 686	3	1	0	2.154	4	3	0.75	1.00	0.86
4	960 × 540	19	0	1	1.974	19	20	1.00	0.95	0.97
5	960 × 540	8	0	0	2.031	8	8	1.00	1.00	1.00
6	1200 × 566	12	0	1	1.953	12	13	1.00	0.92	0.96
7	612 × 340	8	0	0	1.890	8	8	1.00	1.00	1.00
8	889 × 495	15	0	0	2.000	15	15	1.00	1.00	1.00
9	760 × 450	9	0	1	1.906	9	11	1.00	0.90	0.95
10	630 × 395	33	1	0	1.941	34	33	0.97	1.00	0.98
11	1200 × 565	133	2	0	1.906	135	133	0.99	1.00	0.99
12	1600 × 800	4	0	2	2.0	4	6	1.00	0.67	0.80
13	450 × 276	7	0	0	1.875	7	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	2.031	16	16	1.00	1.00	1.00
15	1300 × 957	10	0	1	1.922	10	11	1.00	0.91	0.95
16	660 × 330	3	0	0	1.828	3	3	1.00	1.00	1.00
17	1200 × 566	3	0	0	1.859	3	3	1.00	1.00	1.00
18	259 × 194	7	1	0	1.844	8	7	0.88	1.00	0.94
19	300 × 168	7	0	0	1.954	7	7	1.00	1.00	1.00
20	1280 × 722	27	0	0	1.902	27	27	1.00	1.00	1.00
21	678 × 381	36	0	0	1.859	36	36	1.00	1.00	1.00
22	635 × 290	10	3	0	1.859	13	10	0.77	1.00	0.87
23	800 × 533	8	2	1	1.875	10	9	0.80	0.89	0.84
24	945 × 612	15	0	1	1.875	15	16	1.00	0.94	0.97
25	500 × 325	9	2	0	1.906	11	9	0.82	1.00	0.90

A continuación, en la figura 31 se muestra un resumen de las tablas obtenidas anteriormente tomando como métrica el parámetro *F-Score*; se puede apreciar que los valores de dicho parámetro predominan a lo largo de todas las imágenes en el modelo YOLO.

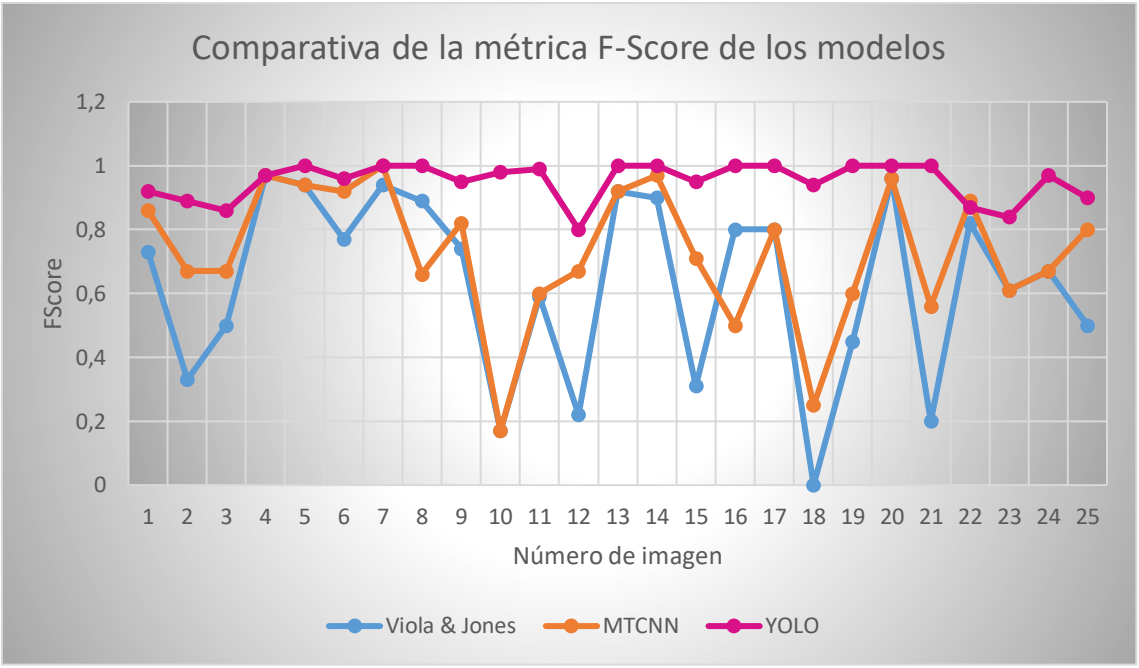


Figura 31. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO

Fuente: Autor

La imagen 18 es la que presenta un menor F-Score en el modelo Viola & Jones, debido a la reducida cantidad de píxeles existentes en cada rostro a pesar de que los mismos se encuentran en posición frontal.

En el modelo MTCNN el menor F-Score se presentó en la imagen 10.

Finalmente, en el modelo YOLO el menor F-Score se presentó en la imagen 12 debido al uso de maquillaje en la mayoría de rostros presentes en la imagen, sin embargo, este valor se encuentra en un rango aceptable.

6.3.2 *Imágenes aplicando Ecualización del Histograma*

Los modelos Viola & Jones y YOLO obtuvieron mejores resultados con la ecualización de imágenes a color, mientras que el modelo MTCNN con la ecualización de imágenes a blanco y negro. Por esta razón se ha decidido presentar en las tablas 21, 22 y 23 los resultados de la ecualización más eficiente.

Tabla 21. Parámetros de análisis para el modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	5	1	2	0.978	6	7	0.83	0.71	0.77
2	1024 × 852	0	0	5	0.630	0	5	0.00	0.00	0.00
3	1024 × 686	1	2	2	0.569	3	3	0.33	0.33	0.33
4	960 × 540	19	0	1	0.728	19	20	1.00	0.95	0.97
5	960 × 540	7	1	1	0.400	8	8	0.88	0.88	0.88
6	1200 × 566	8	0	5	0.574	8	13	1.00	0.62	0.77
7	612 × 340	7	0	1	0.198	7	8	1.00	0.88	0.94
8	889 × 495	11	0	4	0.378	11	15	1.00	0.73	0.84
9	760 × 450	5	1	5	0.314	6	11	0.83	0.50	0.62
10	630 × 395	3	0	30	0.337	3	33	1.00	0.09	0.17
11	1200 × 565	56	6	77	0.865	62	133	0.90	0.42	0.57
12	1600 × 800	2	3	4	1.319	5	6	0.40	0.33	0.36
13	450 × 276	6	0	1	0.740	6	7	1.00	0.86	0.92
14	1300 × 956	16	1	0	1.151	17	16	0.94	1.00	0.97
15	1300 × 957	3	2	8	1.157	5	11	0.60	0.27	0.37
16	660 × 330	2	0	1	0.347	2	3	1.00	0.67	0.80
17	1200 × 566	2	1	1	0.786	3	3	0.67	0.67	0.67
18	259 × 194	0	0	7	0.171	0	7	0.00	0.00	0.00
19	300 × 168	1	0	6	0.196	1	7	1.00	0.14	0.25
20	1280 × 722	24	0	3	1.206	24	27	1.00	0.89	0.94
21	678 × 381	3	0	33	0.353	3	36	1.00	0.08	0.15
22	635 × 290	5	0	5	0.268	5	10	1.00	0.50	0.67
23	800 × 533	4	1	5	0.483	5	9	0.80	0.44	0.57
24	945 × 612	9	1	7	0.628	10	16	0.90	0.56	0.69
25	500 × 325	3	0	6	0.280	3	9	1.00	0.33	0.50

Tabla 22. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	7	0	0	4.045	7	7	1.00	1.00	1.00
2	1024 × 852	3	0	2	3.265	3	5	1.00	0.60	0.75
3	1024 × 686	3	1	0	2.665	4	3	0.75	1.00	0.86
4	960 × 540	20	0	0	2.835	20	20	1.00	1.00	1.00
5	960 × 540	8	0	0	2.706	8	8	1.00	1.00	1.00
6	1200 × 566	12	0	1	3.204	12	13	1.00	0.92	0.96
7	612 × 340	8	0	0	2.431	8	8	1.00	1.00	1.00
8	889 × 495	14	0	1	2.968	14	15	1.00	0.93	0.96
9	760 × 450	9	0	1	2.723	9	11	1.00	0.90	0.95
10	630 × 395	11	0	22	2.475	11	33	1.00	0.33	0.50
11	1200 × 565	93	0	40	4.315	93	133	1.00	0.70	0.82
12	1600 × 800	5	0	1	4.621	5	6	1.00	0.83	0.91
13	450 × 276	6	0	1	2.422	6	7	1.00	0.86	0.92
14	1300 × 956	16	1	0	3.807	17	16	0.94	1.00	0.97
15	1300 × 957	10	0	1	3.827	10	11	1.00	0.91	0.95
16	660 × 330	3	0	0	2.254	3	3	1.00	1.00	1.00
17	1200 × 566	3	0	0	2.934	3	3	1.00	1.00	1.00
18	259 × 194	1	0	6	2.231	1	7	1.00	0.14	0.25
19	300 × 168	3	0	4	2.099	3	7	1.00	0.43	0.60
20	1280 × 722	27	0	0	3.429	27	27	1.00	1.00	1.00
21	678 × 381	23	0	13	2.587	23	36	1.00	0.64	0.78
22	635 × 290	8	0	2	2.557	8	10	1.00	0.80	0.89
23	800 × 533	5	0	4	2.664	5	9	1.00	0.56	0.72
24	945 × 612	9	0	7	2.977	9	16	1.00	0.56	0.72
25	500 × 325	7	0	2	2.326	7	9	1.00	0.78	0.88

Tabla 23. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	5	0	2	2.109	5	7	1.00	0.71	0.83
2	1024 × 852	4	0	1	1.916	4	5	1.00	0.80	0.89
3	1024 × 686	3	1	0	2.054	4	3	0.75	1.00	0.86
4	960 × 540	19	0	1	2.436	19	20	1.00	0.95	0.97
5	960 × 540	8	0	0	2.078	8	8	1.00	1.00	1.00
6	1200 × 566	12	0	1	1.859	12	13	1.00	0.92	0.96
7	612 × 340	8	0	0	1.906	8	8	1.00	1.00	1.00
8	889 × 495	15	0	0	1.844	15	15	1.00	1.00	1.00
9	760 × 450	10	0	0	1.844	10	11	1.00	1.00	1.00
10	630 × 395	33	2	0	1.844	35	33	0.94	1.00	0.97
11	1200 × 565	123	1	9	3.866	124	133	0.99	0.93	0.96
12	1600 × 800	5	0	1	1.922	5	6	1.00	0.83	0.91
13	450 × 276	7	0	0	1.875	7	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	3.781	16	16	1.00	1.00	1.00
15	1300 × 957	10	0	1	1.890	10	11	1.00	0.91	0.95
16	660 × 330	3	0	0	1.844	3	3	1.00	1.00	1.00
17	1200 × 566	3	0	0	1.844	3	3	1.00	1.00	1.00
18	259 × 194	7	1	0	1.858	8	7	0.88	1.00	0.94
19	300 × 168	6	1	1	4.328	7	7	0.86	0.86	0.86
20	1280 × 722	27	0	0	2.048	27	27	1.00	1.00	1.00
21	678 × 381	36	0	0	1.859	36	36	1.00	1.00	1.00
22	635 × 290	11	3	0	1.875	13	10	0.79	1.00	0.88
23	800 × 533	8	3	1	1.859	11	9	0.73	0.89	0.80
24	945 × 612	15	1	1	2.000	16	16	0.94	0.94	0.94
25	500 × 325	9	1	0	1.906	10	9	0.90	1.00	0.95

En la figura 32 se presenta una gráfica del parámetro F-Score correspondiente a cada imagen. Así mismo, se puede comprobar que YOLO continúa siendo el modelo predominante de acuerdo con estos valores.

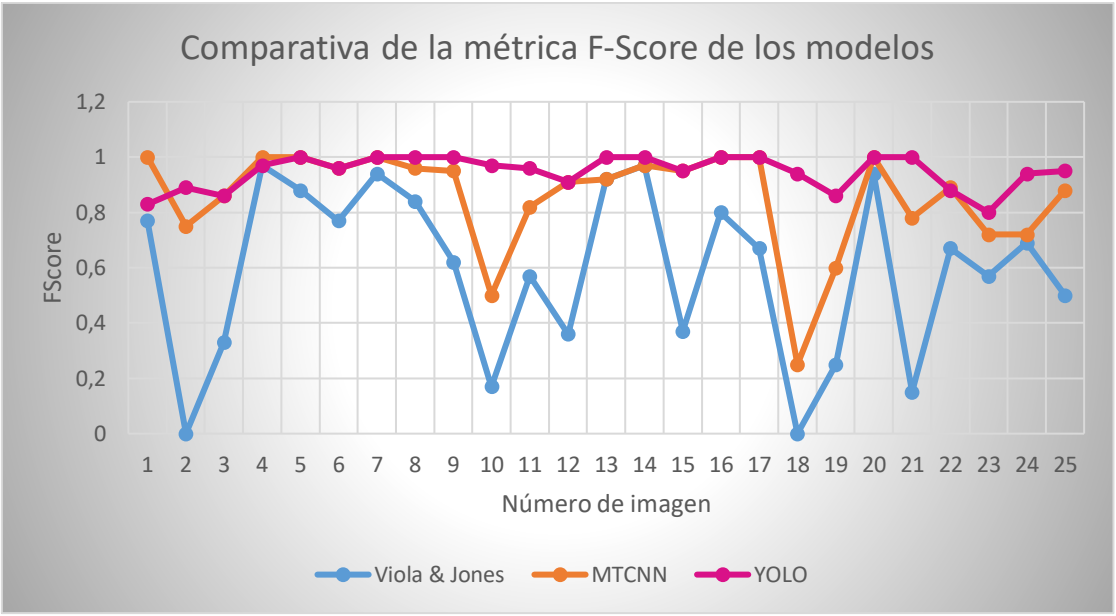


Figura 32. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO

Fuente: Autor

En el modelo de Viola & Jones se observa que tanto en la imagen 2 como en la 18 el valor de F-Score es 0, lo que significa que no ha tenido ninguna detección correcta. Esto se debe a que en la imagen 2 la mayoría de rostros se encuentran de perfil. Por otra parte, en la imagen 18 los rostros poseen un número reducido de píxeles, razón que impide su detección. De la misma forma, la imagen 18 también presenta el menor F-Score en MTCNN.

La imagen 23 posee el F-Score más bajo en el modelo YOLO, lo cual se debe a la dimensión que poseen algunos de los rostros presentes en la imagen y a la cantidad de falsos positivos detectados por el algoritmo.

6.3.3 *Imágenes Escaladas en Dimensión*

En las tablas 24, 25 y 26 se muestran los resultados obtenidos al aumentar la dimensión de las imágenes en un factor de 3.

Tabla 24. Parámetros de análisis para el modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	6	5	1	5.665	11	7	0.55	0.86	0.67
2	1024 × 852	0	2	5	3.949	2	5	0.00	0.00	0.00
3	1024 × 686	1	0	2	3.545	3	3	1.00	0.33	0.50
4	960 × 540	19	0	1	3.304	20	20	1.00	0.95	0.97
5	960 × 540	7	5	1	3.106	12	8	0.58	0.88	0.70
6	1200 × 566	9	3	4	3.484	12	13	0.75	0.69	0.72
7	612 × 340	7	1	1	1.331	8	8	0.88	0.88	0.88
8	889 × 495	14	0	1	2.526	14	15	1.00	0.93	0.96
9	760 × 450	8	7	2	2.362	15	11	0.53	0.80	0.64
10	630 × 395	8	2	25	1.647	10	33	0.80	0.24	0.37
11	1200 × 565	81	1	52	2.323	82	133	0.99	0.61	0.75
12	1600 × 800	3	5	3	7.645	8	6	0.38	0.50	0.43
13	450 × 276	6	2	1	0.947	8	7	0.75	0.86	0.80
14	1300 × 956	16	1	0	6.884	17	16	0.94	1.00	0.97
15	1300 × 957	3	6	8	6.996	9	11	0.33	0.27	0.30
16	660 × 330	2	0	1	1.172	2	3	1.00	0.67	0.80
17	1200 × 566	2	5	1	4.090	7	3	0.29	0.67	0.40
18	259 × 194	2	2	5	0.478	4	7	0.50	0.29	0.37
19	300 × 168	3	0	4	0.517	3	7	1.00	0.43	0.60
20	1280 × 722	26	1	1	5.305	27	27	0.96	0.96	0.96
21	678 × 381	8	3	28	1.796	11	36	0.73	0.22	0.34
22	635 × 290	9	1	1	1.536	10	10	0.90	0.90	0.90
23	800 × 533	5	3	4	2.461	8	9	0.63	0.56	0.59
24	945 × 612	13	2	3	3.648	15	16	0.87	0.81	0.84
25	500 × 325	7	1	2	1.087	8	9	0.88	0.78	0.83

Tabla 25. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	6	1	1	12.932	7	7	0.86	0.86	0.86
2	1024 × 852	3	0	2	8.967	3	5	1.00	0.60	0.75
3	1024 × 686	2	2	1	8.539	4	3	0.50	0.67	0.57
4	960 × 540	19	0	1	6.401	19	20	1.00	0.95	0.97
5	960 × 540	8	0	0	7.121	8	8	1.00	1.00	1.00
6	1200 × 566	9	2	4	7.499	11	13	0.82	0.69	0.75
7	612 × 340	8	0	0	4.722	8	8	1.00	1.00	1.00
8	889 × 495	6	0	9	6.204	6	15	1.00	0.40	0.57
9	760 × 450	5	0	5	4.446	5	11	1.00	0.50	0.67
10	630 × 395	10	0	23	5.370	10	33	1.00	0.30	0.46
11	1200 × 565	85	1	48	7.216	86	133	0.99	0.64	0.78
12	1600 × 800	4	2	2	14.661	6	6	0.67	0.67	0.67
13	450 × 276	6	0	1	3.593	6	7	1.00	0.86	0.92
14	1300 × 956	13	3	3	12.792	16	16	0.81	0.81	0.81
15	1300 × 957	5	1	6	13.686	6	11	0.83	0.45	0.58
16	660 × 330	1	0	2	3.642	1	3	1.00	0.33	0.50
17	1200 × 566	2	2	1	7.623	4	3	0.50	0.67	0.57
18	259 × 194	4	0	3	2.751	4	7	1.00	0.57	0.73
19	300 × 168	3	0	4	2.767	3	7	1.00	0.43	0.60
20	1280 × 722	27	0	0	8.676	27	27	1.00	1.00	1.00
21	678 × 381	23	0	13	6.228	23	36	1.00	0.64	0.78
22	635 × 290	8	0	2	3.675	8	10	1.00	0.80	0.89
23	800 × 533	4	0	5	7.416	4	9	1.00	0.44	0.61
24	945 × 612	10	2	6	8.588	12	16	0.83	0.63	0.72
25	500 × 325	7	0	2	4.194	7	9	1.00	0.78	0.88

Tabla 26. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	6	0	1	2.062	6	7	1.00	0.86	0.92
2	1024 × 852	4	0	1	2.001	4	5	1.00	0.80	0.89
3	1024 × 686	3	1	0	1.969	4	3	0.75	1.00	0.86
4	960 × 540	19	0	1	1.941	19	20	1.00	0.95	0.97
5	960 × 540	8	0	0	1.922	8	8	1.00	1.00	1.00
6	1200 × 566	12	0	1	1.953	12	13	1.00	0.92	0.96
7	612 × 340	8	0	0	1.890	8	8	1.00	1.00	1.00
8	889 × 495	15	0	0	1.922	15	15	1.00	1.00	1.00
9	760 × 450	9	0	1	1.890	9	11	1.00	0.90	0.95
10	630 × 395	33	1	0	1.875	34	33	0.97	1.00	0.98
11	1200 × 565	133	1	0	1.953	134	133	0.99	1.00	0.99
12	1600 × 800	4	0	2	2.062	4	6	1.00	0.67	0.80
13	450 × 276	7	0	0	1.859	7	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	2.062	16	16	1.00	1.00	1.00
15	1300 × 957	10	0	1	2.047	10	11	1.00	0.91	0.95
16	660 × 330	3	0	0	1.875	3	3	1.00	1.00	1.00
17	1200 × 566	3	0	0	1.953	3	3	1.00	1.00	1.00
18	259 × 194	7	1	0	1.969	8	7	0.88	1.00	0.94
19	300 × 168	6	1	0	1.845	7	7	0.86	1.00	0.92
20	1280 × 722	27	0	0	2.008	27	27	1.00	1.00	1.00
21	678 × 381	35	1	0	1.969	36	36	0.97	1.00	0.98
22	635 × 290	10	3	0	1.890	13	10	0.77	1.00	0.87
23	800 × 533	9	3	0	1.906	12	9	0.75	1.00	0.86
24	945 × 612	15	0	1	1.937	15	16	1.00	0.94	0.97
25	500 × 325	9	2	0	1.875	11	9	0.82	1.00	0.90

En la figura 33 se muestra una comparación del F-Score calculado en cada modelo para las distintas imágenes analizadas; el modelo YOLO continúa obteniendo mejores resultados.

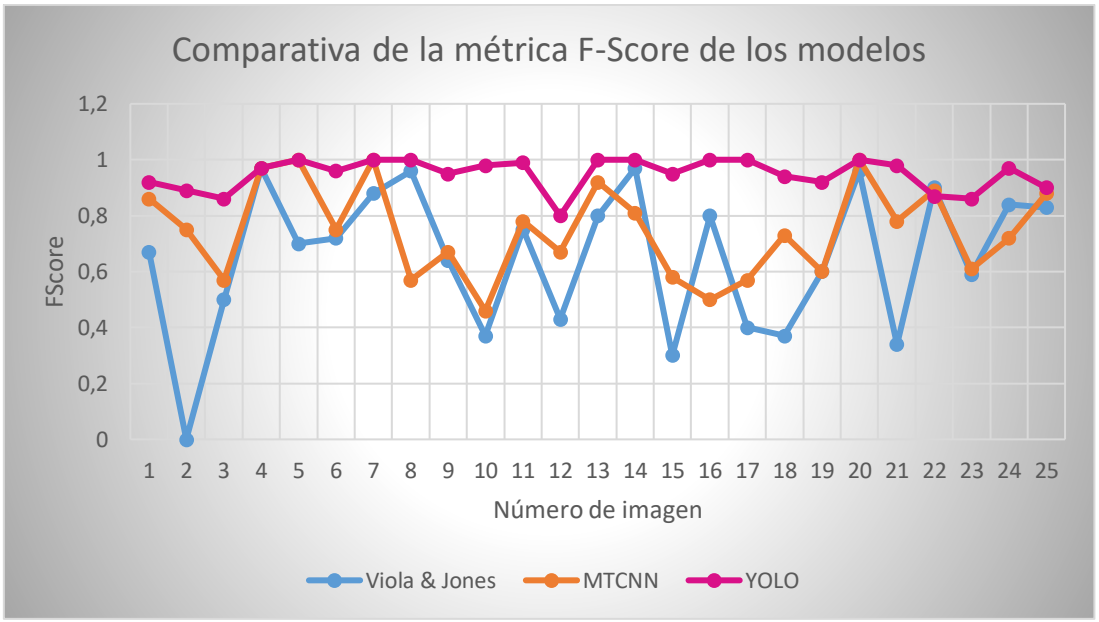


Figura 33. Análisis del parámetro F-Score en los modelos Viola & Jones. MTCNN y YOLO

Fuente: Autor

Analizando la figura se observa que la imagen 2 continúa presentando un F-Score de 0, por lo que se deduce que el problema radica en el tipo de filtro utilizado más no en la dimensión.

Para el caso de MTCNN se observa que el F-Score más bajo se encuentra en la imagen 10, debido a que el aumento de escala. a pesar de ser elevado, no es suficiente para detectar correctamente todos los rostros presentes en la imagen.

6.3.4 *Imágenes Aplicando Corrección Gamma*

Luego de los análisis realizados con diferentes factores de corrección gamma se eligió un factor de 0.5 para los modelos Viola & Jones y YOLO, en cambio para el modelo MTCNN se eligió un factor de 5. En las tablas 27, 28 y 29 se muestran los resultados correspondientes a estos factores.

Los resultados obtenidos con todos los factores de corrección gamma analizados en cada modelo se muestran en el Anexo 4.

Tabla 27. Parámetros de análisis del modelo Viola & Jones con factor de corrección gamma igual a 0.5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	3	2	3	1.086	5	7	0.60	0.50	0.55
2	1024 × 852	0	1	5	0.817	1	5	0.00	0.00	0.00
3	1024 × 686	1	0	2	0.655	1	3	1.00	0.33	0.50
4	960 × 540	18	1	2	0.653	19	20	0.95	0.90	0.92
5	960 × 540	7	2	1	0.685	9	8	0.78	0.88	0.83
6	1200 × 566	8	0	5	0.675	8	13	1.00	0.62	0.77
7	612 × 340	7	0	1	0.320	7	8	1.00	0.88	0.94
8	889 × 495	8	0	7	0.464	8	15	1.00	0.53	0.69
9	760 × 450	7	1	4	0.489	8	11	0.88	0.64	0.74
10	630 × 395	2	1	31	0.381	3	33	0.67	0.06	0.11
11	1200 × 565	12	0	121	0.420	12	133	1.00	0.09	0.17
12	1600 × 800	0	0	6	1.192	1	6	0.00	0.00	0.00
13	450 × 276	6	0	1	0.295	6	7	1.00	0.86	0.92
14	1300 × 956	16	0	0	1.192	16	16	1.00	1.00	1.00
15	1300 × 957	2	0	9	1.091	2	11	1.00	0.18	0.31
16	660 × 330	2	0	1	0.347	2	3	1.00	0.67	0.80
17	1200 × 566	2	0	1	0.717	2	3	1.00	0.67	0.80
18	259 × 194	0	0	7	0.227	0	7	0.00	0.00	0.00
19	300 × 168	2	0	5	0.097	2	7	1.00	0.29	0.45
20	1280 × 722	26	0	1	0.916	26	27	1.00	0.96	0.98
21	678 × 381	3	0	33	0.390	3	36	1.00	0.08	0.15
22	635 × 290	6	0	4	0.393	6	10	1.00	0.60	0.75
23	800 × 533	2	0	7	0.529	4	9	1.00	0.22	0.36
24	945 × 612	8	1	8	0.613	9	16	0.89	0.50	0.64
25	500 × 325	1	0	8	0.135	1	9	1.00	0.11	0.20

Tabla 28. Parámetros de análisis del modelo MTCNN con factor de corrección gamma igual a 5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	1	0	6	3.204	1	7	1.00	0.14	0.25
2	1024 × 852	3	0	2	2.716	3	5	1.00	0.60	0.75
3	1024 × 686	3	0	0	2.665	3	3	1.00	1.00	1.00
4	960 × 540	14	0	6	2.475	14	20	1.00	0.70	0.82
5	960 × 540	7	0	1	2.292	7	8	1.00	0.88	0.94
6	1200 × 566	6	0	7	2.650	6	13	1.00	0.46	0.63
7	612 × 340	7	0	1	2.355	7	8	1.00	0.88	0.94
8	889 × 495	6	0	9	2.519	6	15	1.00	0.40	0.57
9	760 × 450	5	1	5	2.402	6	11	0.83	0.50	0.62
10	630 × 395	0	0	33	1.905	0	33	0.00	0.00	0.00
11	1200 × 565	32	0	101	3.004	32	133	1.00	0.24	0.39
12	1600 × 800	3	2	3	3.502	5	6	0.60	0.50	0.55
13	450 × 276	3	0	4	2.018	3	7	1.00	0.43	0.60
14	1300 × 956	10	0	6	3.193	10	16	1.00	0.63	0.77
15	1300 × 957	5	0	6	2.924	5	11	1.00	0.45	0.62
16	660 × 330	1	0	2	2.324	1	3	1.00	0.33	0.50
17	1200 × 566	2	0	1	2.839	2	3	1.00	0.67	0.80
18	259 × 194	0	0	7	2.170	0	7	0.00	0.00	0.00
19	300 × 168	3	0	4	2.022	3	7	1.00	0.43	0.60
20	1280 × 722	23	0	4	2.830	23	27	1.00	0.85	0.92
21	678 × 381	8	0	28	2.297	8	36	1.00	0.22	0.36
22	635 × 290	7	0	3	2.240	7	10	1.00	0.70	0.82
23	800 × 533	4	0	5	2.377	4	9	1.00	0.44	0.61
24	945 × 612	7	0	9	2.597	7	16	1.00	0.44	0.61
25	500 × 325	6	0	3	2.448	6	9	1.00	0.67	0.80

Tabla 29. Parámetros de análisis del modelo YOLO con factor de corrección gamma igual a 0.5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	5	0	2	1.969	5	7	1.00	0.71	0.83
2	1024 × 852	3	0	2	3.968	3	5	1.00	0.60	0.75
3	1024 × 686	3	0	0	1.844	3	3	1.00	1.00	1.00
4	960 × 540	18	0	2	1.854	18	20	1.00	0.90	0.95
5	960 × 540	8	0	0	1.859	8	8	1.00	1.00	1.00
6	1200 × 566	12	0	1	1.969	12	13	1.00	0.92	0.96
7	612 × 340	8	0	0	1.828	8	8	1.00	1.00	1.00
8	889 × 495	14	0	1	1.844	14	15	1.00	0.93	0.96
9	760 × 450	9	0	1	1.859	9	11	1.00	0.90	0.95
10	630 × 395	33	2	0	1.828	35	33	0.94	1.00	0.97
11	1200 × 565	48	0	95	1.937	48	133	1.00	0.34	0.51
12	1600 × 800	2	0	4	1.875	2	6	1.00	0.33	0.50
13	450 × 276	7	0	0	1.859	7	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	1.859	16	16	1.00	1.00	1.00
15	1300 × 957	6	0	5	1.875	6	11	1.00	0.55	0.71
16	660 × 330	2	0	1	1.847	2	3	1.00	0.67	0.80
17	1200 × 566	3	0	0	1.875	3	3	1.00	1.00	1.00
18	259 × 194	7	2	0	1.584	9	7	0.78	1.00	0.88
19	300 × 168	7	0	0	1.844	7	7	1.00	1.00	1.00
20	1280 × 722	27	0	0	1.915	27	27	1.00	1.00	1.00
21	678 × 381	34	0	2	1.859	34	36	1.00	0.94	0.97
22	635 × 290	10	0	0	3.072	10	10	1.00	1.00	1.00
23	800 × 533	7	2	2	1.859	9	9	0.78	0.78	0.78
24	945 × 612	15	0	1	1.845	15	16	1.00	0.94	0.97
25	500 × 325	9	1	0	1.844	10	9	0.90	1.00	0.95

En la figura 34 se puede observar que el modelo YOLO posee un F-Score más alto en la mayoría de imágenes, exceptuando la imagen 12 en la cual el modelo MTCNN es ligeramente mayor, esta imagen continúa siendo el mayor reto para el modelo YOLO.

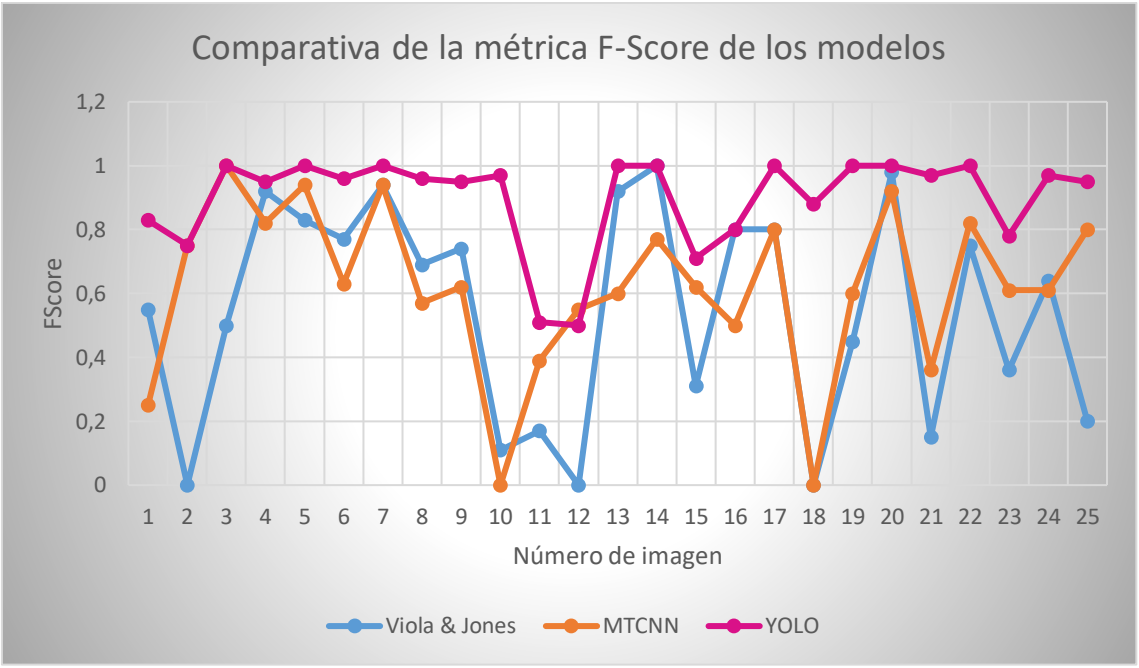


Figura 34. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO

Fuente: Autor

En el modelo Viola & Jones se presentan 3 casos en los cuales el F-Score es 0, además de las imágenes 2 y 18 que en casos anteriores han mostrado este mismo valor, en este caso también se une la imagen 12.

De la misma forma las imágenes 10 y 18 continúan presentando el F-Score más bajo en el modelo MTCNN llegando a 0 al aplicar normalización de corrección gamma.

6.3.5 *Imágenes aplicando rotación*

En las tablas 30, 31 y 32 se muestran los resultados al aplicar un ángulo de rotación de 45° respecto al centro de cada imagen.

Tabla 30. Parámetros de análisis del modelo Viola & Jones con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	0	0	7	1.333	0	7	0.00	0.00	0.00
2	1024 × 852	0	0	5	0.828	0	5	0.00	0.00	0.00
3	1024 × 686	0	0	3	0.717	0	3	0.00	0.00	0.00
4	960 × 540	0	0	20	0.435	0	20	0.00	0.00	0.00
5	960 × 540	0	0	8	0.583	0	8	0.00	0.00	0.00
6	1200 × 566	0	0	13	0.628	0	13	0.00	0.00	0.00
7	612 × 340	0	0	8	0.286	0	8	0.00	0.00	0.00
8	889 × 495	0	0	16	0.516	0	15	0.00	0.00	0.00
9	760 × 450	0	0	11	0.530	0	11	0.00	0.00	0.00
10	630 × 395	0	0	33	0.304	0	33	0.00	0.00	0.00
11	1200 × 565	0	0	133	0.499	0	133	0.00	0.00	0.00
12	1600 × 800	0	0	6	0.975	0	6	0.00	0.00	0.00
13	450 × 276	0	0	7	0.260	0	7	0.00	0.00	0.00
14	1300 × 956	0	1	16	1.094	1	16	0.00	0.00	0.00
15	1300 × 957	0	0	11	0.932	0	11	0.00	0.00	0.00
16	660 × 330	0	0	3	0.313	0	3	0.00	0.00	0.00
17	1200 × 566	0	0	3	0.596	0	3	0.00	0.00	0.00
18	259 × 194	0	0	7	0.147	0	7	0.00	0.00	0.00
19	300 × 168	0	0	7	0.174	0	7	0.00	0.00	0.00
20	1280 × 722	0	0	27	0.742	0	27	0.00	0.00	0.00
21	678 × 381	0	0	36	0.348	0	36	0.00	0.00	0.00
22	635 × 290	0	0	10	0.248	0	10	0.00	0.00	0.00
23	800 × 533	0	0	9	0.475	0	9	0.00	0.00	0.00
24	945 × 612	0	0	16	0.570	0	16	0.00	0.00	0.00
25	500 × 325	0	0	9	0.252	0	9	0.00	0.00	0.00

Tabla 31. Parámetros de análisis del modelo MTCNN con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	0	0	7	3.213	0	7	0.00	0.00	0.00
2	1024 × 852	1	0	4	2.800	1	5	1.00	0.20	0.33
3	1024 × 686	1	1	2	2.841	2	3	0.50	0.33	0.40
4	960 × 540	5	1	13	2.501	6	20	0.83	0.28	0.42
5	960 × 540	1	1	6	2.728	2	8	0.50	0.14	0.22
6	1200 × 566	0	1	9	2.867	1	13	0.00	0.00	0.00
7	612 × 340	0	0	7	2.254	0	8	0.00	0.00	0.00
8	889 × 495	0	1	11	2.663	0	15	0.00	0.00	0.00
9	760 × 450	0	0	6	2.309	0	11	0.00	0.00	0.00
10	630 × 395	0	0	26	2.313	0	33	0.00	0.00	0.00
11	1200 × 565	1	0	99	2.486	1	133	1.00	0.01	0.02
12	1600 × 800	0	2	4	3.840	2	6	0.00	0.00	0.00
13	450 × 276	2	0	5	2.215	2	7	1.00	0.29	0.45
14	1300 × 956	1	1	15	3.372	2	16	0.50	0.06	0.11
15	1300 × 957	0	2	8	3.401	2	11	0.00	0.00	0.00
16	660 × 330	0	0	2	2.249	0	3	0.00	0.00	0.00
17	1200 × 566	0	1	3	3.063	1	3	0.00	0.00	0.00
18	259 × 194	0	0	7	2.070	0	7	0.00	0.00	0.00
19	300 × 168	0	0	7	2.038	0	7	0.00	0.00	0.00
20	1280 × 722	8	0	17	3.149	8	27	1.00	0.32	0.48
21	678 × 381	0	0	33	2.707	0	36	0.00	0.00	0.00
22	635 × 290	0	0	6	2.155	0	10	0.00	0.00	0.00
23	800 × 533	0	0	9	2.684	0	9	0.00	0.00	0.00
24	945 × 612	3	2	9	3.208	5	16	0.60	0.25	0.35
25	500 × 325	0	0	6	2.211	0	9	0.00	0.00	0.00

Tabla 32. Parámetros de análisis del modelo YOLO con ángulo de rotación de 45°

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	N° C.	P	R	FS
1	1024 × 1366	1	0	6	1.875	1	7	1.00	0.14	0.25
2	1024 × 852	1	1	4	2.078	2	5	0.50	0.20	0.29
3	1024 × 686	3	0	0	1.859	3	3	1.00	1.00	1.00
4	960 × 540	13	0	5	1.882	13	20	1.00	0.72	0.84
5	960 × 540	5	1	2	1.859	6	8	0.83	0.71	0.77
6	1200 × 566	3	1	6	1.844	4	13	0.75	0.33	0.46
7	612 × 340	2	1	5	1.846	3	8	0.67	0.29	0.40
8	889 × 495	6	3	5	1.875	9	15	0.67	0.55	0.60
9	760 × 450	0	0	6	1.844	0	11	0.00	0.00	0.00
10	630 × 395	0	1	26	1.906	1	33	0.00	0.00	0.00
11	1200 × 565	1	1	99	1.859	2	133	0.50	0.01	0.02
12	1600 × 800	0	0	4	1.859	0	6	0.00	0.00	0.00
13	450 × 276	4	0	3	1.844	4	7	1.00	0.57	0.73
14	1300 × 956	8	3	8	1.875	11	16	0.73	0.50	0.59
15	1300 × 957	4	0	4	1.859	4	11	1.00	0.50	0.67
16	660 × 330	1	0	1	1.906	1	3	1.00	0.50	0.67
17	1200 × 566	0	0	2	1.859	0	3	0.00	0.00	0.00
18	259 × 194	1	0	6	1.035	1	7	1.00	0.14	0.25
19	300 × 168	0	0	7	0.174	0	7	0.00	0.00	0.00
20	1280 × 722	10	2	7	2.140	12	27	0.83	0.59	0.69
21	678 × 381	3	6	30	1.845	9	36	0.33	0.09	0.14
22	635 × 290	0	0	6	1.859	0	10	0.00	0.00	0.00
23	800 × 533	1	0	8	1.984	1	9	1.00	0.11	0.20
24	945 × 612	4	0	5	1.845	4	16	1.00	0.44	0.61
25	500 × 325	3	0	3	1.859	3	9	1.00	0.50	0.67

En la figura 35 se puede observar que para el modelo de Viola & Jones el F-Score en todas las imágenes es 0 debido a que este modelo solo ha sido entrenado para detectar caras en posición frontal sin ángulo de rotación.

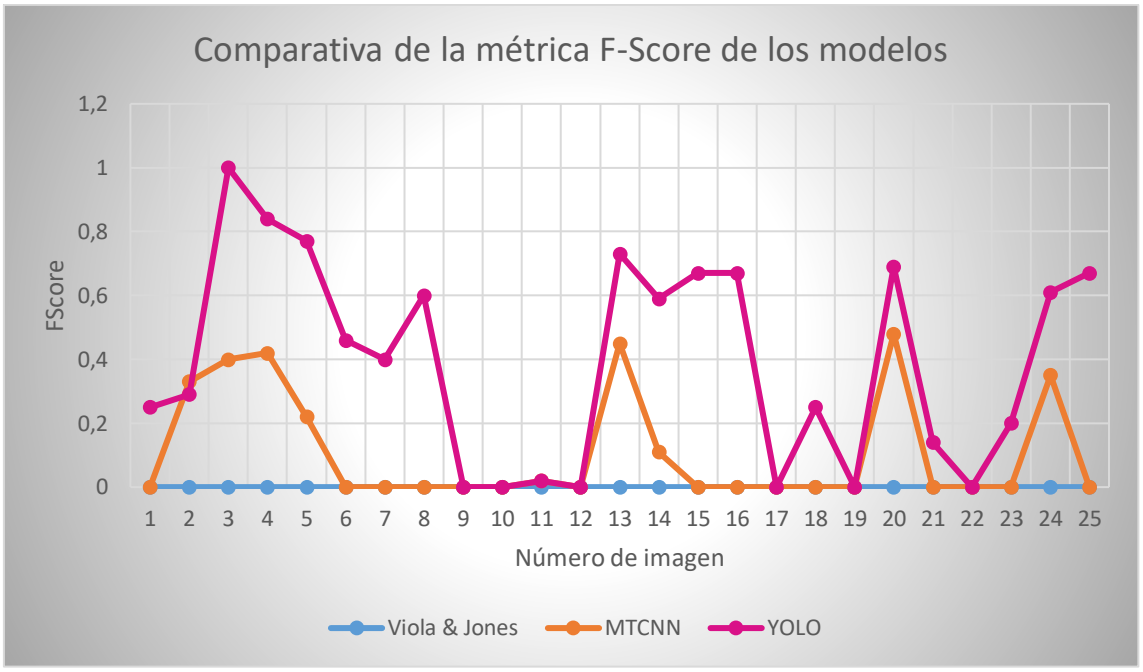


Figura 35. Análisis del parámetro F-Score en los modelos Viola & Jones. MTCNN y YOLO

Fuente: Autor

En general, se puede aseverar que todos los modelos presentan dificultades al detectar rostros con ciertos ángulos de rotación, siendo los resultados incluso menores con ángulos mayores.

A continuación, en la figura 36 se presenta una gráfica que muestra el tiempo total que toma cada uno de los algoritmos para ejecutarse en los diferentes escenarios establecidos anteriormente, es decir, imágenes normalizadas y sin normalizar.

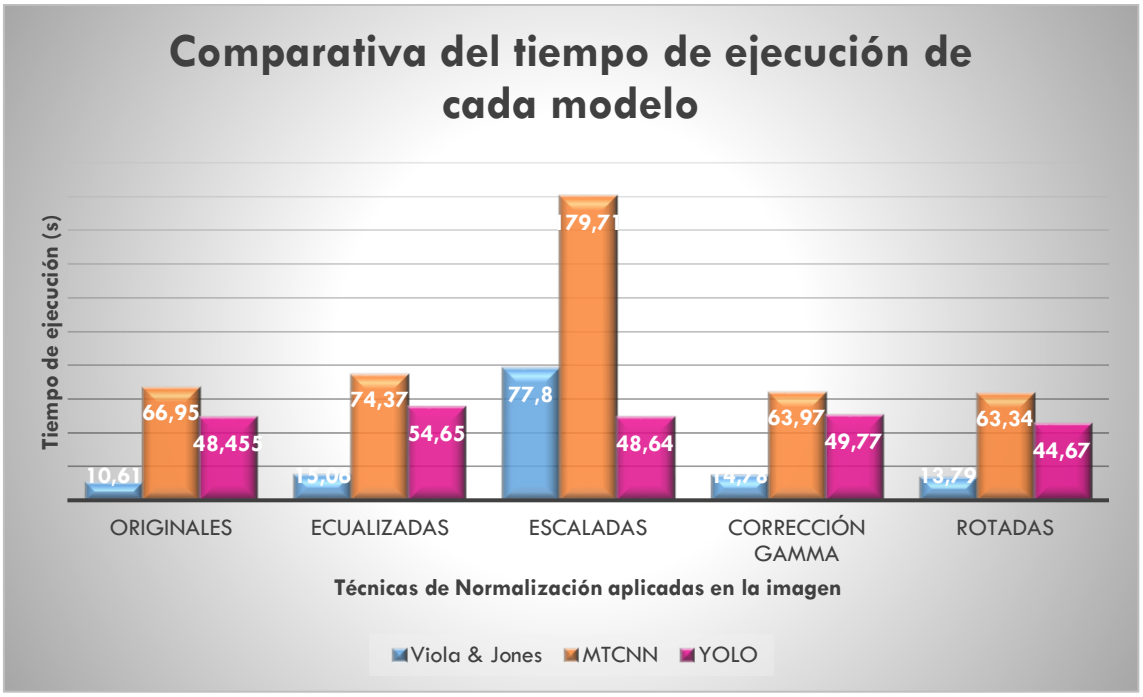


Figura 36. Tiempo de ejecución de cada modelo en los diferentes conjuntos de imágenes (originales y normalizadas)

Fuente: Autor

7. DISCUSIÓN

7.1 Algoritmo CNN

El entrenamiento de un modelo de red neuronal es un proceso que puede tardar días e incluso semanas en completarse, esto depende de la complejidad del modelo que se esté desarrollando. los parámetros más relevantes que influyen en el entrenamiento pueden ser la dimensión y el conjunto de las imágenes de entrada. el *learning rate*, el número de *epochs*, la profundidad de la red (número de capas), el número de canales de la imagen (RGB o Gris) y la cantidad de salidas de la red.

Los parámetros para desarrollar el algoritmo se escogieron de tal forma que la red no demore más de 30 minutos aproximadamente en terminar el proceso de entrenamiento, esto con el fin de realizar varias pruebas y determinar los pesos con los que el algoritmo alcance una precisión más elevada y una tasa de error baja.

Cabe recalcar que las imágenes de rostros detectadas correctamente por el algoritmo contienen parte del fondo del ambiente, esto ocurre debido a que en la fase de entrenamiento se utilizó imágenes que presentaban estas características. Además, debido al sobreajuste producido por la red, este algoritmo obtuvo una gran cantidad de falsos negativos lo que hizo imposible la implementación del mismo.

7.2 Algoritmo Viola & Jones

De acuerdo con los resultados, el algoritmo de detección desarrollado a partir de características Haar no cumple con los niveles de eficiencia suficientes; a pesar de que se utilizaron diferentes parámetros en cada imagen para alcanzar una tasa de detección elevada no se consiguieron resultados satisfactorios en la mayoría de los casos analizados.

Tomando en cuenta que el filtro utilizado es frontal face, los rostros de perfil o con ángulos de rotación notables no fueron detectados, y a pesar de los intentos por incorporar varios filtros con el fin de mejorar la tasa de detección, solamente se consiguió elevar en gran número la cantidad de falsos negativos.

7.3 Algoritmo MTCNN

A partir de los resultados obtenidos se puede determinar que el modelo MTCNN tarda aproximadamente 3 segundos en detectar rostros, esto se debe a su arquitectura, la cual

está conformada por tres redes neuronales conectadas en cascada, por lo que para obtener el resultado final de la detección se debe esperar la ejecución de las dos primeras etapas.

Por otra parte, según los resultados mostrados en imágenes de un solo rostro, el algoritmo MTCNN presenta deficiencias al momento de detectar imágenes que contengan retratos pintados de rostros.

7.4 Algoritmo YOLO

De acuerdo con el análisis desarrollado se pudo notar que en el modelo YOLO, en algunos casos, los cuadros delimitadores contienen además del rostro información relacionada con el fondo de la imagen que se puede considerar como ruido pues no aporta información relevante; así mismo se presentaron casos en los que en el cuadro delimitador era muy pequeño por lo que eliminaba información relevante del rostro, como punta de nariz, filos de ojos, etc.

8. CONCLUSIONES

- A partir del análisis de los resultados obtenidos en los distintos modelos se pudo determinar que el modelo Viola & Jones consiguió el tiempo de ejecución más bajo, el cual en la mayoría de casos no superaba un segundo, el modelo MTCNN presentó el tiempo de ejecución más alto en todos los casos analizados siendo este aproximadamente tres segundos y el modelo YOLO un tiempo intermedio de 1.5 segundos aproximadamente; sin embargo, la tasa de detección obtenida por el modelo YOLO en la mayoría de los casos fue significativamente mayor en comparación con los otros modelos, por lo que se concluyó que este modelo es el más eficiente para detección y segmentación de rostros dentro de los modelos de detección analizados en la presente investigación.
- El pre-procesamiento realizado en el conjunto de imágenes de prueba. normalizando parámetros de iluminación, dimensión y rotación tanto en imágenes con un rostro y varios rostros no tuvo impacto alguno en la tasa de detección del modelo YOLO debido a que este cuenta con una fase de normalización dentro de su arquitectura; no obstante, la tasa de detección del modelo MTCNN aumentó al normalizar las imágenes utilizando ecualización del histograma; en fotografías de un solo rostro se disminuyó la cantidad de falsos negativos de 3 a 1 mientras que en fotografías de varios rostros el F-Score aumentó en todos los casos. De la misma manera, según las gráficas analizadas en imágenes con varios rostros, la detección del modelo Viola & Jones también mejoró los valores de F-Score en todos los casos.
- Como se mencionó en Análisis de Bases de Datos. las redes neuronales al ser entrenadas con imágenes en distintas condiciones de iluminación, oclusión, escala, etc., poseen una tasa de detección más elevada que el algoritmo de Viola & Jones el cual solo detecta caras frontales utilizando el filtro *frontal face* implementado en esta investigación.
- La tasa de detección disminuye notablemente al aplicar rotación en imágenes de uno y varios rostros, llegando a ser cero en la mayoría de los casos; lo cual, en redes neuronales, se puede corregir al añadir imágenes rotadas en la etapa de entrenamiento, y en el modelo Viola & Jones utilizando filtros Haar con distintos ángulos de rotación.

- Por las razones mencionadas anteriormente, se decidió implementar el modelo YOLO en el lenguaje de programación Python ya que es un lenguaje de código abierto cuyo aprendizaje es relativamente cómodo y posee librerías desarrolladas para evaluar algoritmos relacionados al campo de inteligencia artificial; la ejecución del modelo se realizará desde la terminal de comandos y almacenando los resultados de la detección y segmentación en los directorios creados.
- El algoritmo desarrollado por la autora como modelo base de comparación, no alcanzó los estándares de precisión establecidos en el estado del arte, los cuales son superiores al 85%, debido al sobreajuste existente en la red producido por la robustez de la arquitectura, la cual no se logró incrementar por falta de recursos computacionales.

9. RECOMENDACIONES

- Para obtener resultados precisos en la detección de los rostros. se recomienda entrenar la red neuronal con bases de datos en las cuales las imágenes contengan únicamente las características del rostro; es decir, descartando toda la información correspondiente al fondo de la imagen.
- Para escoger una base de datos que optimice los resultados, ésta debe poseer rostros con diferentes expresiones, tonos de piel, niveles de oclusión, maquillaje, rotación, iluminación, escala, pose, con la finalidad de incluir el mayor número de características posibles dentro del sistema.
- Debido al elevado número de tareas computacionales que debe ejecutar el CPU cuando se está entrenando una red neuronal. se recomienda paralelizar el proceso de entrenamiento trabajando en clusters de GPU, o contar con GPU's físicos como los modelos GTX 1060(6GB) o la GTX 1050 Ti (4GB) para desarrollar modelos básicos y los modelos RTX 2080 O GTX1080 para realizar modelos más complejos; de esta manera se acelera en gran medida la ejecución de la etapa de entrenamiento, que para redes más robustas tardaría varias semanas en completarse utilizando únicamente el procesador del computador.
- Para realizar detecciones en las cuales el tiempo de ejecución sea un factor crítico, se recomienda utilizar el modelo de Viola & Jones eligiendo el filtro de detección adecuado acorde al escenario de implementación.
- Como se mencionó anteriormente. los modelos de redes neuronales necesitan un elevado tiempo de entrenamiento para obtener el conjunto de pesos que describan adecuadamente las características de un rostro; es por esto que, para realizar el análisis del comportamiento de los modelos en distintas condiciones de iluminación, escala, etc., se recomienda utilizar modelos pre-entrenados cuyo conjunto de pesos haya sido calculado previamente.
- Para mejorar la tasa de detección en rostros que presenten rotación se debería incluir filtros de características Haar con distintos ángulos de rotación en la etapa de extracción de características del modelo Viola & Jones.
- En el caso de modelos de redes neuronales, la detección de rostros con ángulos de rotación se puede mejorar agregando imágenes que presenten estas características en las bases de datos que a su vez se incluirán en la etapa de entrenamiento y validación.

10. BIBLIOGRAFÍA

Acuña, L. N., & Farias Falkiewicz, M. M. (2018). *APLICACIÓN ANDROID PARA EL PROCESAMIENTO DE IMÁGENES HISTOLÓGICAS Y SOPORTE PARA EL ACOPLAMIENTO DE UN SMARTPHONE A UN MICROSCOPIO*. Retrieved from https://rdu.unc.edu.ar/bitstream/handle/11086/6527/Informe Proyecto Integrador Acuña - Falkiewicz.pdf?sequence=1

Apolinario Casaño, R. E., & Pinedo Lima, C. (2018). *Diseño de un sistema de detección de humo en ambientes cerrados aplicando técnicas de procesamiento digital de imágenes* (FACULTAD DE INGENIERÍA Y ARQUITECTURA ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA). Retrieved from http://www.repositorioacademico.usmp.edu.pe/bitstream/usmp/4335/1/apolinario_pinedo.pdf

Aranguren Zapata, A., & Vela Asin, T. (2012). *Sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos*. Universidad Peruana de Ciencias Aplicadas.

Azueto-Ríos, Á., Santiago-Godoy, R., Hernández-Gómez, L. E., & Hernández-Santiago, K. A. (2017). *Implementación de un sistema de imagenología infrarroja para la detección vascular del antebrazo y mano Implementation of an infrared imaging system for vascular detection in the forearm and hand*. *38*(2), 479–491. https://doi.org/10.17488/RMIB.38.2.4

Caballero Barriga, E. R. (2017). *APLICACIÓN PRÁCTICA DE LA VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE ROSTROS EN UNA IMAGEN, UTILIZANDO REDES NEURONALES Y ALGORITMOS DE RECONOCIMIENTO DE OBJETOS DE LA BIBLIOTECA OPENCV* (UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS). Retrieved from http://repository.udistrital.edu.co/bitstream/11349/6104/1/CaballeroBarrigaEdisonRene2017.pdf

Camps-Valls, G. (2012). *Procesado de Datos*.

Características de Java. (2019). Retrieved October 22, 2019, from http://dis.um.es/~bmoros/Tutorial/parte2/cap2-5.html

Castillo, R., Hernández, J. M., Inzunza, E., & Torres, J. P. (2013). *Procesamiento Digital de Imágenes Empleando Filtros Espaciales*. Retrieved from http://www.iiis.org/CDs2013/CD2013SCI/CISCI_2013/PapersPdf/CA780XP.pdf

Cazorla Martínez, R. (2016). *Software para la Detección y el Reconocimiento de Rostros*. Retrieved from https://ddd.uab.cat/pub/tfg/2016/tfg_49339/Software_para_la_deteccio_n_y_el_reconocimiento_de_caras.pdf

Creación, R. Y., & Modelo, D. E. L. (2018). ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO FACIAL 3D MEDIANTE SISTEMA DE VIDEO APLICADO A LA SEGURIDAD USANDO INTELIGENCIA ARTIFICIAL ” (ESCUELA SUPERIOR

POLITÉCNICA DE CHIMBORAZO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA). Retrieved from http://dspace.espoch.edu.ec/bitstream/123456789/9198/1/108T0245.pdf

Eguíluz Pérez, J. (2012). *Introducción a JavaScript*. Retrieved from www.librosweb.es

Escriche, D. (2017). *Detección automática de personas en secuencias de vídeo*. 51. Retrieved from https://upcommons.upc.edu/bitstream/handle/2117/104413/121627.pdf?sequence=1&isAllowed=y

Espinoza, D. E., & Jorquera, P. I. (2015). *Reconocimiento Facial* (Pontificia Universidad Católica de Valparaíso). Retrieved from http://opac.pucv.cl/pucv_txt/txt-1000/UCD1453_01.pdf

Esqueda Elizondo, J. J. (2005). *Fundamentos de Procesamiento de Imágenes*. Retrieved from https://s3.amazonaws.com/academia.edu.documents/39190415/FundamentosDeProcesamientoDeImagenes.pdf?response-content-disposition=inline%3B filename%3DFundamentos_de_procesamiento_de_imagenes.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYY

Esqueda, J. (2002). *Fundamentos de Procesamiento de Imágenes INSTITUTO TECNOLÓGICO DE CIUDAD MADERO*. Retrieved from https://s3.amazonaws.com/academia.edu.documents/39190415/FundamentosDeProcesamientoDeImagenes.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1542866986&Signature=%2BdtC%2B9fB9GcCIRhgunj7Mq63LnY%3D&response-content-disposition=inline%3B filename%3DFunda

Fernández Montoro, A. (2012). *Python 3 al descubierto*. Retrieved from www.rclibros.es

García Mateos, G. (2007). *Procesamiento de Caras Humanas Mediante Integrales Proyectivas* (Universidad de Murcia). Retrieved from http://dis.um.es/~ginesgm/files/inv/tesis.pdf

Georghiades, A., Belhumeur, P., & Kriegman, D. (2001). The Extended Yale Face Database B. Retrieved August 6, 2019, from http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html

Guevara, M., Echeverry, J., & Urueña, W. (2008). *Detección de Rostros en Imágenes Digitales usando Clasificadores en Cascada*. *1*. Retrieved from http://www.redalyc.org/articulo.oa?id=84903801

Heath, N. (2019). GitHub: The top 10 programming languages for machine learning - TechRepublic. Retrieved October 21, 2019, from https://www.techrepublic.com/article/github-the-top-10-programming-languages-for-machine-learning/

Jiménez Ochoa, M. G. (2015). *Desarrollo de un sistema de visión artificial para la detección de aglomeración de personas en un semáforo* (Universidad Nacional de

Loja). Retrieved from https://dspace.unl.edu.ec/jspui/bitstream/123456789/11225/1/Jiménez Ochoa, Magaly Gabriela.pdf

Larrañaga, P., Inza, I., & Moujahid Abdelmalik. (2005). *Tema 8. Redes Neuronales*. Retrieved from http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

Llinás Solano, H. (2014). *Introducción a la Teoría de la probabilidad* (Universidad del Norte, Ed.). Retrieved from https://www.ecoeediciones.com/wp-content/uploads/2015/08/Introduccion-a-la-teoria-de-la-probabilidad-Vista-preliminar-del-libro.pdf

López Aragoneses, L. (2016). *Detección de la saliencia visual dinámica: hacia el modelado de la atención* (Universidad Carlos III de Madrid). Retrieved from https://e-archivo.uc3m.es/bitstream/handle/10016/27207/TFG_Laura_Lopez_Aragoneses.pdf?sequence=1

Machine Learning Tools I recently discovered. (2018). Retrieved October 22, 2019, from https://jwork.org/home/node/61

Martín Ortíz, M. (2013). *Procesamiento Digital de Imágenes* (p. 114). p. 114. Retrieved from http://mmartin.cs.buap.mx/notas/PDI-MM-Rev.2013.pdf

Massachusetts Institute of Technology. (2005). MIT- CBCL FACE RECOGNITION DATABASE. Retrieved August 6, 2019, from http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html

Massó Gómez, W. (2012). Utilizando Bibliotecas de Código C en Aplicaciones C#. *Revista Telem@tica*, *11*(2), 36–45. Retrieved from http://revistatelematica.cujae.edu.cu/index.php/tele

Mello Román, J. C. (2017). *MEJORA DE CONTRASTE UTILIZANDO MORFOLOGÍA MATEMÁTICA MULTIESCALA PARA IMÁGENES EN ESCALA DE GRISES E IMÁGENES EN COLOR* (Universidad Nacional de Asunción). Retrieved from http://www.conacyt.gov.py/sites/default/files/Tesis_JC_Mello.pdf

Morales Navarro, N. A., Méndez Sántiz, G., Castillejos Lara, E., Brindis Velázquez, O., & Velasco Bermúdez, S. (2017, December). Sistema de visión para detectar y alarmar el estado de cansancio de un conductor. *7*, 119–129. https://doi.org/ISSN 2007-9400

Niclós Corts, R. (2011). *Calibración y corrección radiométrica*. Retrieved from file:///D:/TODO EL MÁSTER DE TELEDETECCIÓN/MATERIAS POR PROFESOR/P. de Imagenes. Raquel Niclós/Tema 4/4_1_Niclos_Calibracion_2011.pdf

Parra Barrero, E. (2015). *Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal* (Escuela Técnica Superior de Ingeniería). Retrieved from

http://bibing.us.es/proyectos/abreproy/90325/fichero/TrabajoFinGrado.pdf

Párraga Párraga, J. V., & Casa López, L. F. (2017). *Desarrollo de un algoritmo que permita paralelizar imágenes 3D sobre tecnología NVIDIA y OPENCL* (UNIVERSIDAD POLITÉCNICA SALESIANA). Retrieved from https://dspace.ups.edu.ec/bitstream/123456789/14117/1/UPS - ST003109.pdf

Peña, M., & Orellana, J. (2018). *Red neuronal para clasificación de riesgo en cooperativas de ahorro y crédito*. *13*, 121–124. https://doi.org/10.24133/cctespe.v13i1.710

Platero Plazas, D. C. (2015). *RECONOCIMIENTO DE IMÁGENES FACIALES ORIENTADO A CONTROLES DE ACCESO Y SISTEMAS DE SGURIDAD* (Universidad Distrital "Francisco José de Caldas"). Retrieved from http://repository.udistrital.edu.co/bitstream/11349/7359/1/PlateroPlazasDonovanCamilo2015.pdf

Principales Características de Java. (2019). Retrieved October 22, 2019, from http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm

Raman, K. J., Azman, A., Ibrahim, S. Z., Yogarayan, S., Abdullah, M. F. A., Razak, S. F. A., … Sonai Muthu, K. (2018). *A Comparability Study on Driver Fatigue Using C#, C++ and Python*. Retrieved from http://journal.utem.edu.my/index.php/ijhati/article/viewFile/3811/2704

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. Retrieved from http://pjreddie.com/yolo/

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. Retrieved from https://pjreddie.com/yolo/.

Rivas Araiza, E. A., Mendiola Santibañez, J. D., Herrera Ruiz, G., González Gutiérrez, C. A., Perea, M. T., & Moreno, G. J. R. (2007). *Mejora de Contraste y Compensación en Cambios de la iluminación*. *10*(4), 357–371. Retrieved from http://www.scielo.org.mx/pdf/cys/v10n4/v10n4a4.pdf

Robledano-Arillo - Valentín Moreno-Pelayo - José Manuel Pereira-Uzal, J. (2016). *Aproximación experimental al uso de métricas objetivas para la estimación de calidad cromática en la digitalización de patrimonio documental gráfico*. https://doi.org/10.3989/redc.2016.2.1249

Russ, J. C., & Brent Neal, F. (2017). *The Image Processing* (Séptima; Taylor & Francis Group, Ed.). Boca Raton.

Sánchez Salazar, C. R. (2017). *Análisis de dos algoritmos para detección de rostros e implementación de uno de ellos utilizado en analítica de video para sistemas de videovigilancia* (ESCUELA POLITÉCNICA NACIONAL FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA). Retrieved from https://bibdigital.epn.edu.ec/bitstream/15000/17471/1/CD-7971.pdf

Shih, P., & Liu, C. (2006). Face detection using discriminating feature analysis and

Support Vector Machine. *Pattern Recognition*, *39*, 260–276. https://doi.org/10.1016/j.patcog.2005.07.003

Shin, J. (2019). Top Javascript Machine Learning libraries in 2019. Retrieved October 22, 2019, from https://towardsdatascience.com/top-javascript-machine-learning-libraries-in-2019-cb63b95bdd10

Suquinagua León, G. E. (2019). *Sistema de Notificación de Agenda a través de Reconocimiento Facial: Prototipo para Docentes a Tiempo Completo de la Carrera de Computación de la Universidad Católica de Santiago de Guayaquil* (UNIVERSIDAD CATÓLICA DE SANTIAGO DE GUAYAQUIL). Retrieved from http://192.188.52.94:8080/bitstream/3317/12595/1/T-UCSG-PRE-ING-CIS-220.pdf

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). *Going Deeper with Convolutions*. Retrieved from https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

The 2018 Top Programming Languages - IEEE Spectrum. (2018). Retrieved October 21, 2019, from https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

The Most Popular Language For Machine Learning Is ... (IT Best Kept Secret Is Optimization). (2016). Retrieved October 19, 2019, from https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en

The State of the Octoverse: machine learning - The GitHub Blog. (2019). Retrieved October 21, 2019, from https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/

Top 8 of the TIOBE index quite stable for the last 15 years. (2019). Retrieved October 21, 2019, from https://www.tiobe.com/tiobe-index/?fbclid=IwAR3-4495rarqBT8FLoIm_SE74O8HdIAXRHbHyEX7A4Rtr5bq8I107I12AWE

Universität Erlangen-Nürnberg (FAU). (2001). The BioID Face Database. Retrieved August 6, 2019, from https://www.bioid.com/facedb/

University of Stirling. (n.d.). Psychological Image Collection at Stirling (PICS). Retrieved August 6, 2019, from http://pics.psych.stir.ac.uk/

Vázquez, J. C., Constable, L., & Nacional, T. (2018). *RNA-AP: Redes Neurales Artificiales con Aprendizaje Profundo*. Retrieved from http://sedici.unlp.edu.ar/bitstream/handle/10915/77385/Documento_completo.pdf?sequence=1

Viola, P., & Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features*. https://doi.org/10.1109/CVPR.2001.990517

Vivas Imparato, A. A. (2014). *Desarrollo de un sistema de reconocimiento facial* (Escuela

Técnica Superior de Ingeniería y Sistemas de Telecomunicación). Retrieved from http://oa.upm.es/33506/1/TFG_abdon_alejandro_vivas_imparato.pdf

Zhang, C., & Zhang, Z. (2010). *A Survey of Recent Advances in Face Detection*.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*. Retrieved from https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf

11. ANEXOS

ANEXO 1: MANUAL DEL PROGRAMADOR

Algoritmo de Ventana deslizante

```
import numpy as np      #Permite operar con matrices de manera simple
from keras.models import load_model   #Permite cargar el modelo de CNN
from keras.models import model_from_json  #  Permite leer el modelo de CNN
from PIL import Image, ImageDraw      #Permite trabajar con imágenes
json_file = open('model2.json')   #Abriendo el modelo de CNN
loaded_model_json = json_file.read()  #Leyendo el archivo de la red
json_file.close()
loaded_model = model_from_json(loaded_model_json) #Cargando la arquitectura de la CNN
loaded_model.load_weights('pesos2.h5') #Cargando los pesos de la CNN

 #Creación de la función detection con parámetros de entrada nombre de la imagen, paso y tamaño de la ventana deslizante
def detection(image,paso, tamano):
    image = Image.open(image)   #Abriendo la imagen
    image = image.convert('L')  #Convirtiendo la imagen a escala de grises
    a,b = image.size            #Obteniendo las dimensiones de la imagen
    cont_foto=0                 #Inicializando contador en 0
    clase=[]                    #Inicializando arreglo clase vacío
    puntos= []                  #Inicializando arreglo puntos vacío
    for c in range(tamano, a, tamano): #Tamaño de ventana deslizante
        for y1 in range(0, b-tamano, paso): #Recorriendo los píxeles correspondientes al alto de la imagen
            y2= y1+c
            for x1 in range(0, a-tamano, paso): #Recorriendo los píxeles correspondientes al ancho de la imagen
                x2 = x1 +c
                caja = (x1, y1, x2, y2)        #Puntos en el que se encuentra la ventana
                region = image.crop(caja)      #Recortando la ventana para analizarla
                cont_foto=cont_foto+1          #Aumentado en 1 el contador
                region2 = region.resize((24,24), Image.ANTIALIAS) #Redimensionando la ventana de análisis a un tamaño de 24x24
                region2 = np.array(region2).reshape(-1,24,24,1) #Redimensionando la forma de la matriz de la ventana de imagen
                clasificacion= loaded_model.predict(region2)    #Resultado de la predicción obtenido en la CNN
                clase= clasificacion.argmax()                   #Clase resultante
                #En caso de que la ventana fuera detectada como cara, se guarda la imagen en un directorio previamente creado
                if (clase != 1):
                    region.save('C:/Users/Soraya/Documents/Computer_Detection/pruebas1/crop_{}'.format(cont_foto)+'.jpg')
                    puntos.append(caja) #Se guarda los puntos de las ventanas detectadas como caras
    return
print (detection('prueba1.jpg', 25, 25)) #Se ejecuta la función detection en la imagen llamada prueba1
```

Red Neuronal Convolucional (CNN,Convolutional Neural Networks)

Importando Librerías

```
import numpy as np   #Permite operar con matrices de manera simple
# Fija las semillas aleatorias para la reproducibilidad
np.random.seed(64)
import os            #Permite manipular estructura de directorios
import re            #Permite realizar operaciones de expresiones regulares
import matplotlib.pyplot as plt  #Realiza graficos en 2D
%matplotlib inline
from sklearn.model_selection import train_test_split #Divide el conjunto de imágenes
```

```
import keras        #Librería utilizada para realizar CNN
from keras.utils import to_categorical   #Permite codificar las etiquetas
from keras.models import Sequential,Input,Model  #Modelo utilizado para la CNN
#Capas utilizadas para la creación de la CNN
from keras.layers import Dense, Dropout, Flatten, Activation
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras import optimizers
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint
from keras.layers.core import Dense
from keras.models import model_from_json
```

```
Using TensorFlow backend.
```

Cargando conjunto de Imágenes (dataset)

```
#Conocer el directorio actual y unirlo con el nombre de la carpeta que contiene el dataset
dirname = os.path.join(os.getcwd(), 'database')
imgpath = dirname + os.sep #Ruta en la que se encuentra el dataset
#Inicializando arreglos vacios
images = []
directories = []
dircount = []
prevRoot='' #Inicializando variable en vacío
cant=0 #Inicializando contador en cero

print("leyendo imagenes de ",imgpath)

for root, dirnames, filenames in os.walk(imgpath): #Generando nombres de directorios en dataset
    for filename in filenames: #Recorriendo los subdirectorios
        if re.search("\.(jpg|jpeg|png|bmp|tiff|pgm|PGM)$", filename): #Busca solo formatos de imágenes
            cant=cant+1                             #El contador va aumentando cada que se encuentre una imagen
            filepath = os.path.join(root, filename) #Ruta de imagen
            image = plt.imread(filepath,0)          #Leyendo imagen
            images.append(image) #Agregando imagen al arreglo images
            b = "Leyendo..." + str(cant)
            print (b, end="\r")
            if prevRoot !=root:           #Se ejecuta siempre que exista un subdirectorio
                print(root, cant)
                prevRoot=root             #La variable vacía toma el valor de la ruta del directorio del dataset
                directories.append(root)  #Agregando subdirectorios encontrados en el arreglo directories
                dircount.append(cant)     #Agregando imágenes encontrados en cada subdirectorio en el arreglo dircount
                cant=0                    #El contador vuelve a ser cero
dircount.append(cant)       #Agrega los valores del contador al arreglo dircount
dircount = dircount[1:]     #Selecciona solo los valores correspondientes a la cantidad de imágenes
dircount[0]=dircount[0]+1   #Ubica en la posición cero del arreglo la cantidad de imágenes
#Presenta en pantalla la cantidad de subdirectorios encontrados, cantidad de imágenes en cada uno y el total
print('Directorios leidos:',len(directories))
print("Imagenes en cada directorio", dircount)
print('suma Total de imagenes en subdirs:',sum(dircount))
```

```
leyendo imagenes de  C:\Users\Soraya\Documents\Tesis_Soraya_Orozco\Computer_Detection\CNN\database\
C:\Users\Soraya\Documents\Tesis_Soraya_Orozco\Computer_Detection\CNN\database\caras 1
C:\Users\Soraya\Documents\Tesis_Soraya_Orozco\Computer_Detection\CNN\database\no_caras 23765
Directorios leidos: 2
Imagenes en cada directorio [23766, 13613]
suma Total de imagenes en subdirs: 37379
```

Creando etiquetas

```
labels=[] #Inicializando un arreglo vacio, para conocer la cantidad de imágenes que se etiquetan
indice=0  #Inicializando variable en 0, necesaria para definir las clases presentes en la red
for cantidad in dircount: #Recorre el arreglo dircount
    for i in range(cantidad): #Recorre la variable cantidad que toma su valor de dircount
        labels.append(indice) #Agrega al arreglo labels las etiquetas de acuerdo a la cantidad de imágenes
    indice=indice+1           #Aumenta la variable indice para cambiar el valor de etiqueta de acuerdo a las clases
print("Cantidad etiquetas creadas: ",len(labels)) #Presenta en pantalla la cantidad de etiquetas creadas
```

```
Cantidad etiquetas creadas:  37379
```

```
etiquetas_caras=[] #Inicializando un arreglo vacio
indice=0           #Inicializando variable en 0
for directorio in directories: #Recorre la variable directories correspondiente a la ruta de subdirectorios
    name = directorio.split(os.sep)   #Divide la ruta en cada palabra que contenga
    print(indice , name[len(name)-1]) #Presenta en pantalla solo el nombre del subdirectorio presente en la ruta
    etiquetas_caras.append(name[len(name)-1]) #Agrega al arreglo los nombres de los subdirectorios
    indice=indice+1 #Aumenta la variable indice para cambiar el valor de etiqueta de acuerdo a las clases
```

```
0 caras
1 no_caras
```

```
y = np.array(labels) #Conversión de lista en arreglo
X = np.array(images, dtype=np.float32) #Conversión de lista a arreglo
classes = np.unique(y)  # Encuentra los números únicos de las etiquetas de entrenamiento (train)
nClasses = len(classes) #Determina la cantidad de clases
#Presenta en pantalla el número total de clases y las clases
print('Número total de clases: ', nClasses)
print('Clases: ', classes)
```

```
Número total de clases:  2
Clases:  [0 1]
```

Creando conjuntos de train y test

```
train_X,test_X,train_Y,test_Y = train_test_split(X,y,test_size=0.2, random_state=13) #Diviendo conjunto en 80% train y 20% test
train_X = train_X.reshape(-1,24,24,1)   #Redimensionando la forma de las imágenes de train
test_X = test_X.reshape(-1,24,24,1)     #Redimensionando la forma de las imágenes de test
#Presentando en pantalla la forma de los datos de train y test
print('Forma de los datos de train: ', train_X.shape, train_Y.shape)
print('Forma de los datos de test: ', test_X.shape, test_Y.shape)
```

```
#Presentando la primera imagen de los datos de train
fig = plt.figure(figsize = (3,3)) #Tamaño de la figura
plt.subplot(121)                  #Posición de la imagen de acuerdo a los ejes
plt.imshow(train_X[0], cmap='gray') #Presentando imagen
plt.axis('off')                   #Apagando los ejes
plt.title("Clase: {}".format(train_Y[0])) #Título de la imagen

# Presentando la primera imagen de los datos de train
plt.subplot(122)                  #Posición de la imagen de acuerdo a los ejes
plt.imshow(test_X[0], cmap='gray') #Presentando imagen
plt.axis('off')                   #Apagando los ejes
plt.title("Clase: {}".format(test_Y[0])) #Título de la imagen
```

```
Text(0.5, 1.0, 'Clase: 1')
```

Preprocesando las imagenes

```
train_X = train_X.astype('float32') #Estableciendo el tipo de datos de la imagen como float de 32bits
test_X = test_X.astype('float32')   #Estableciendo el tipo de datos de la imagen como float de 32bits
train_X = train_X / 255.    #Valores de los píxeles entre 0 y 1
test_X = test_X / 255.      #Valores de los píxeles entre 0 y 1
```

Codificación de las etiquetas

```
train_Y_one_hot = to_categorical(train_Y) #Codificación activa de las etiquetas de train
test_Y_one_hot = to_categorical(test_Y)   #Codificación activa de las etiquetas de test

#Presentando en pantalla las etiquetas originales y nuevas
print('Etiquetas originales:', train_Y[0], ',' , test_Y[0])
print('Etiquetas codificadas:', train_Y_one_hot[0], ',' , test_Y_one_hot[0])
```

```
Etiquetas originales: 0 , 1
Etiquetas codificadas: [1. 0.] , [0. 1.]
```

Creando conjuntos de train y validación

```
#Mezclando los datos y dividiendo los grupos de train y validación
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2, random_state=13)
#Presentando en pantalla la forma de los datos de train y validación
print(train_X.shape,valid_X.shape,train_label.shape,valid_label.shape)
```

```
(23922, 24, 24) (5981, 24, 24) (23922, 2) (5981, 2)
```

Creando modelo de CNN

```
#Declaración variables con los parámetros de configuración de la red
INIT_LR = 5e-3 #Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001
epochs = 100 #Cantidad de iteraciones completas al conjunto de imagenes de entrenamiento
batch_size = 100 #Cantidad de imágenes que se toman a la vez en memoria
nClases = 2  #Cantidad de clases
```

```
face_detection_model = Sequential()  #Modelo Sequential para realizar CNN de manera simple
#Primera capa convolucional con 8 filtros, un kernel de 2x2, activación LeakyReLU y capa de agrupamiento de 2x2
face_detection_model.add(Conv2D(8,2,data_format='channels_last',kernel_initializer = 'he_normal', padding='same', input_shape=(24
face_detection_model.add(LeakyReLU(alpha=0.1))
face_detection_model.add(MaxPooling2D((2, 2), strides=2))
#Segunda capa convolucional con 16 filtros, un kernel de 2x2, activación LeakyReLU y capa de agrupamiento de 2x2
face_detection_model.add(Conv2D(16,2,data_format='channels_last',kernel_initializer = 'he_normal', padding='same'))
face_detection_model.add(LeakyReLU(alpha=0.1))
face_detection_model.add(MaxPooling2D((2, 2), strides=2))
#Tercera capa convolucional con 32 filtros, un kernel de 2x2, activación LeakyReLU y capa de agrupamiento de 2x2
face_detection_model.add(Conv2D(32,2,data_format='channels_last',kernel_initializer = 'he_normal', padding='same'))
face_detection_model.add(LeakyReLU(alpha=0.1))
face_detection_model.add(MaxPooling2D((2, 2), strides=2))
#Cuarta capa convolucional con 64 filtros, un kernel de 2x2 y activación LeakyReLU
face_detection_model.add(Conv2D(64,2,data_format='channels_last',kernel_initializer = 'he_normal', padding='same'))
face_detection_model.add(LeakyReLU(alpha=0.1))

face_detection_model.add(Flatten()) #Aplana los datos de entrada
face_detection_model.add(Dense(5, init='he_normal')) #La salida será un arreglo de (*,5)
face_detection_model.add(Dropout(0.6)) #Se descarta el 60% de pesos aleatorios
face_detection_model.add(Dense(nClasses, activation='softmax')) #La salida será un arreglo de (*,2)
optim = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)#Optimizador estocástico Adam
```

```
C:\Users\Soraya\Anaconda3\envs\TESIS\lib\site-packages\ipykernel_launcher.py:17: UserWarning: Update your `Dense` call to the K
eras 2 API: `Dense(5, kernel_initializer="he_normal")`
```

```
face_detection_model.summary() #Resumen de las capas establecidas en la red
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 24, 24, 8)         40
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 24, 24, 8)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 8)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 16)        528
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 12, 12, 16)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 32)          2080
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 6, 6, 32)          0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 3, 3, 64)          8256
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 3, 3, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2885
_________________________________________________________________
dropout_2 (Dropout)          (None, 5)                 0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 12
=================================================================
Total params: 13,801
Trainable params: 13,801
Non-trainable params: 0
_________________________________________________________________
```

```
model_json = face_detection_model.to_json() #Estableciendo una variable en la que se pase el formato de la red a .json
with open("model2.json", "w") as json_file: #Nombre del archivo .json
    json_file.write(model_json)             #Guardando el modelo
```

```
#Estableciendo como parámetro de pérdida de la red la pérdida de entropía cruzada
face_detection_model.compile(loss=keras.losses.categorical_crossentropy, optimizer= optim, metrics=['accuracy'])
filepath="pesos2.h5" #Nombre del archivo en el que se guardan los pesos de la red
#Se guardaran los pesos que obtengan los mejores parámetros durante la validación de la red
checkpoint = ModelCheckpoint(filepath, verbose=1, save_best_only=True,save_weights_only=True, mode='auto')
```

Entrenamiento del modelo de CNN

```
#Entrenamiento del modelo de acuerdo a los parametros establecidos anteriormente
face_detection_train = face_detection_model.fit(train_X, train_label, batch_size=batch_size,epochs=epochs,verbose=1,validation_da
```

```
Train on 23922 samples, validate on 5981 samples
Epoch 1/100
23922/23922 [==============================] - 8s 318us/step - loss: 0.3175 - acc: 0.8717 - val_loss: 0.1707 - val_acc: 0.9363

Epoch 00001: val_loss improved from inf to 0.17067, saving model to pesos2.h5
Epoch 2/100
23922/23922 [==============================] - 7s 308us/step - loss: 0.1897 - acc: 0.9244 - val_loss: 0.0928 - val_acc: 0.9651

Epoch 00002: val_loss improved from 0.17067 to 0.09275, saving model to pesos2.h5
```

Evaluación de la CNN

```
test_eval = face_detection_model.evaluate(test_X, test_Y_one_hot, verbose=1)
```

```
7476/7476 [==============================] - 1s 135us/step
```

```
#Presenta en pantalla los parámetros obtenidos de la evaluación de CNN
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.026822406033681922
Test accuracy: 0.9965222043873729
```

```
#Presentado en una gráfica los resultados obtenidos a lo largo del entrenamiento de la red
accuracy = face_detection_train.history['acc'] #Precisión obtenida en el entrenamiento durante cada época
val_accuracy = face_detection_train.history['val_acc'] #Precisión obtenida en la validación durante cada época
loss = face_detection_train.history['loss']  #Pérdida obtenida en el entrenamiento durante cada época
val_loss = face_detection_train.history['val_loss'] #Pérdida obtenida en la validación durante cada época
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy') #Gráfica de la precisión durante el entrenamiento
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy') #Gráfica de la precisión durante la validación
plt.title('Training and validation accuracy') #Título de la gráfica de precisión
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')  #Gráfica de la pérdida durante el entrenamiento
plt.plot(epochs, val_loss, 'b', label='Validation loss')  #Gráfica de la pérdida durante la validación
plt.title('Training and validation loss')     #Título de la gráfica de pérdida
plt.legend()
plt.show()
```

Creando directorios para imágenes de entrada

```
import os      #Permite manipular y crear estructuras de directorios
import shutil  #Permite realizar operaciones con archivos
import re      #Permite realizar operaciones de expresiones regulares
path = 'C:/Users/Soraya/Documents/Computer_Detection1/fotos/' #Ruta de la que se obtendran las imágenes
def obt_foto(path):            #Creación de la función obt_foto
    os.chdir(path)             #Cambia de directorio de trabajo
    archivos = os.listdir('.') #Enumera todos los archivos que están contenidos en la ruta
    #Creación de arreglos vacíos y variables iniciadas en 0
    fotos = []         #Arreglo para fotografías encontradas
    cont_foto = 0      #Contador de fotografías
    num_actual = []    #Arreglo utilizado para determinar el número de directorio nuevo
    for i in archivos: #Recorre los archivos encontrados en la ruta
        if re.search("\.(jpg|jpeg|png|bmp|tiff|pgm|PGM|JPG)$", i): #Se ejecuta solo si el archivo encontrado es una imagen
            #Ruta para directorios de imágenes de entrada
            path2 = 'C:/Users/Soraya/Documents/Tesis_Soraya_Orozco/Computer_Detection/fotos_ent/'
            os.chdir(path2)                             #Cambia de directorio de trabajo al de directorio de entrada
            cont_foto = cont_foto+1                     #Aumenta el contador en 1
            nuevo = path2 + 'foto_{}'.format(cont_foto) #Se establece la ruta de directorio para una imagen nueva
            if (os.path.exists(nuevo)):                 #Se ejecuta en caso de que exista el directorio anterior
                num_actual = os.listdir ('.') #Enumera los archivos que se encuentran en el directorio de fotos de entrada
                cont_foto = cont_foto + len(numero_actual) #El contador pasa a ser uno más a la cantidad de directorios enumerado
                nuevo = path2 + 'foto_{}'.format(cont_foto)#Se establece una ruta de directorio para una imagen nueva
            os.mkdir(nuevo) #Se crea el nuevo directorio
            os.rename(path+i, path+'{}.jpg'.format(cont_foto)) #La imagen toma el nombre del directorio creado
            fotos.append(i) #Se agrega al directorio fotos las imágenes encontradas
            os.chdir(path)  #Cambia al directorio del que se adquieren las imágenes
            shutil.move('{}.jpg'.format(cont_foto),nuevo) #Mueve la imagen encontrada al directorio nuevo
            #Ruta para directorios de imágenes de salida
            path3 ='C:/Users/Soraya/Documents/Tesis_Soraya_Orozco/Computer_Detection/fotos_sal/caras_segmentadas'
            os.chdir(path3) #Cambia al directorio de imágenes de salida
            d_cs = path3 + 'foto_{}'.format(cont_foto) #Se establece una ruta de directorio de salida para una imagen nueva
            os.mkdir(d_cs)  #Se crea el nuevo directorio
            os.chdir(path)  #Se regresa al directorio de adquisición de imágenes
    #En caso de no existir imágenes en la ruta se presenta en pantalla un mensaje indicando esto
    if len (fotos) ==0:
        print ('No existen imágenes en la ruta')
    return len(fotos)  #Si el programa se ejecuta de manera correcta retorna la cantidad de fotografías encontradas
print (obt_foto(path))
```

ANEXO 2: IMÁGENES DE ANÁLISIS

- Imágenes de un solo rostro

Figura 37. Imagen 1 e imagen 2 de izquierda a derecha

Figura 38. Imagen 3 e imagen 4 de izquierda a derecha

Figura 39. Imagen 5 e imagen 6 de izquierda a derecha

Figura 40. Imagen 7 e imagen 8 de izquierda a derecha

Figura 41. Imagen 9

Figura 42. Imagen 10 e imagen 11 de izquierda a derecha

Figura 43. Imagen 12 e imagen 13 de izquierda a derecha

Figura 44. Imagen 14 e imagen 15 de izquierda a derecha

- Imágenes de varios rostros

Figura 45. Imagen 1 e imagen 2 de izquierda a derecha

Figura 46. Imagen 3 e imagen 4 de izquierda a derecha

Figura 47. Imagen 5 e imagen 6 de izquierda a derecha

Figura 48. Imagen 7 e imagen 8 de izquierda a derecha

Figura 49. Imagen 9 e imagen 10 de izquierda a derecha

Figura 50. Imagen 11

Figura 51. Imagen 12

Figura 52. Imagen 13 e imagen 14 de izquierda a derecha

Figura 53. Imagen 15 e imagen 16 de izquierda a derecha

Figura 54. Imagen 17 e imagen 18 de izquierda a derecha

Figura 55. Imagen 19 e imagen 20 de izquierda a derecha

Figura 56. Imagen 21 e imagen 22 de izquierda a derecha

Figura 57. Imagen 23 e imagen 24 de izquierda a derecha

Figura 58. Imagen 25

ANEXO 3: NORMALIZACIÓN EN IMÁGENES DE UN SOLO ROSTRO

- Ecualización del Histograma

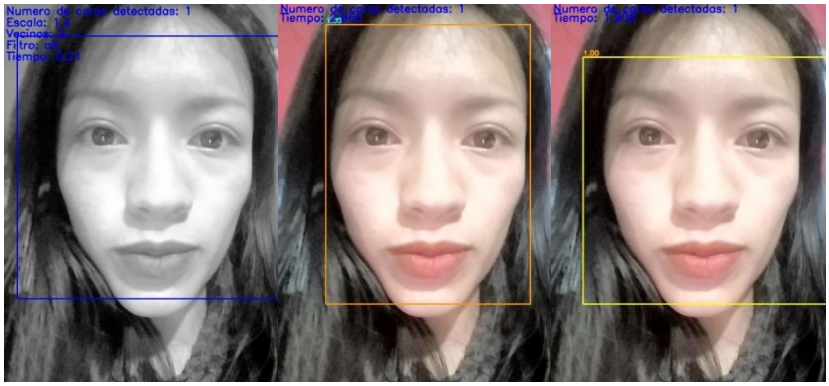


Figura 59. Imágenes normalizadas con ecualización del Histograma para los modelos Viola & Jones, MTCNN y YOLO de izquierda a derecha

Tabla 33. Parámetros de análisis del modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	1	0	0	0.564	0
2	1024 × 1539	0	0	1	0.515	0
3	1024 × 879	0	0	1	0.211	0
4	1024 × 1021	1	0	0	0.312	1
5	1024 × 732	0	0	1	0.231	0
6	1024 × 1538	1	0	0	0.449	1
7	1024 × 1185	1	0	0	0.451	1
8	1024 × 640	0	0	1	0.370	0
9	1024 × 576	1	0	0	0.232	1
10	1200 × 800	0	0	1	0.395	0
11	500 × 500	0	0	1	0.160	0
12	682 × 457	0	0	1	0.256	0
13	640 × 391	0	0	1	0.126	0
14	1140 × 530	0	0	1	0.314	0
15	604 × 835	1	0	0	0.210	1

Tabla 34. Parámetros de análisis del modelo MTCNN.

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° de caras detectadas
1	1024 × 680	1	0	0	2.625	1
2	1024 × 1539	0	0	1	3.778	0
3	1024 × 879	0	0	1	3.045	0
4	1024 × 1021	1	0	0	3.358	1
5	1024 × 732	0	0	1	2.923	0
6	1024 × 1538	1	1	0	3.668	2
7	1024 × 1185	1	0	0	3.353	1
8	1024 × 640	0	0	1	2.834	0
9	1024 × 576	1	0	0	2.515	1
10	1200 × 800	1	1	0	3.324	2
11	500 × 500	1	0	0	2.739	1
12	682 × 457	0	0	1	2.409	0
13	640 × 391	1	0	0	2.739	1
14	1140 × 530	0	0	1	3.168	0
15	604 × 835	1	0	0	2.865	1

Tabla 35. Parámetros de análisis del modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	1	0	0	1.844	1
2	1024 × 1539	1	0	0	1.867	1
3	1024 × 879	1	1	0	1.875	2
4	1024 × 1021	1	0	0	1.875	1
5	1024 × 732	1	0	0	2.004	1
6	1024 × 1538	1	0	0	1.859	1
7	1024 × 1185	1	0	0	1.875	1
8	1024 × 640	0	0	1	1.859	0
9	1024 × 576	1	0	0	1.875	1
10	1200 × 800	1	0	0	2.031	1
11	500 × 500	1	0	0	1.859	1
12	682 × 457	1	0	0	1.875	1
13	640 × 391	1	0	0	1.844	1
14	1140 × 530	1	0	0	1.859	1
15	604 × 835	1	0	0	1.906	1

- Corrección Gamma con factores de 0.25, 0.5, 5 y 10

Figura 60. Imágenes normalizadas con un factor de corrección gamma de 0.25 para los modelos Viola & Jones. MTCNN y YOLO de izquierda a derecha

Tabla 36. Parámetros de análisis del modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	0	0	1	0.264	0
2	1024 × 1539	1	0	0	0.312	1
3	1024 × 879	0	0	1	0.404	0
4	1024 × 1021	0	0	1	0.397	0
5	1024 × 732	0	0	1	0.266	0
6	1024 × 1538	1	0	0	0.442	1
7	1024 × 1185	1	0	0	0.449	1
8	1024 × 640	0	0	1	0.178	0
9	1024 × 576	1	0	0	0.211	1
10	1200 × 800	0	0	1	0.420	0
11	500 × 500	0	0	1	0.254	0
12	682 × 457	0	0	1	0.226	0
13	640 × 391	0	0	1	0.190	0
14	1140 × 530	0	0	1	0.358	0
15	604 × 835	0	0	1	0.200	0

Tabla 37. Parámetros de análisis del modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° de caras detectadas
1	1024 × 680	0	0	1	3.288	0
2	1024 × 1539	1	0	0	3.533	1
3	1024 × 879	0	0	1	2.503	0
4	1024 × 1021	0	0	1	3.070	0
5	1024 × 732	0	0	1	2.632	0
6	1024 × 1538	1	0	0	4.210	1
7	1024 × 1185	1	0	0	3.296	1
8	1024 × 640	0	0	1	3.237	0
9	1024 × 576	0	0	1	2.196	0
10	1200 × 800	0	0	1	3.587	0
11	500 × 500	0	0	1	2.375	0
12	682 × 457	1	0	0	2.689	1
13	640 × 391	0	0	1	2.355	0
14	1140 × 530	0	0	1	2.914	0
15	604 × 835	1	0	0	2.626	1

Tabla 38. Parámetros de análisis del modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	1	0	0	1.844	1
2	1024 × 1539	1	0	0	2.000	1
3	1024 × 879	1	1	0	1.875	2
4	1024 × 1021	1	0	0	1.882	1
5	1024 × 732	1	0	0	1.859	1
6	1024 × 1538	1	0	0	2.015	1
7	1024 × 1185	1	0	0	1.875	1
8	1024 × 640	0	0	1	1.984	0
9	1024 × 576	1	0	0	1.859	1
10	1200 × 800	1	0	0	1.859	1
11	500 × 500	0	0	1	1.906	0
12	682 × 457	1	0	0	1.844	1
13	640 × 391	0	0	1	1.859	0
14	1140 × 530	0	0	1	1.859	0
15	604 × 835	1	0	0	1.875	1

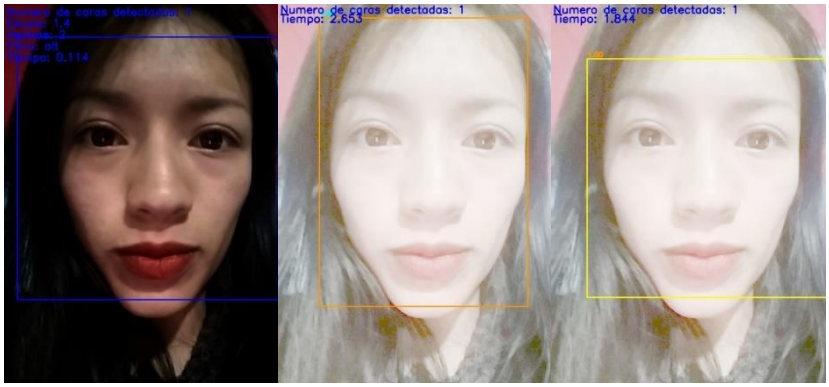


Figura 61. Imágenes normalizadas con un factor de corrección gamma de 0.5 para el modelo Viola & Jones, y factor de corrección 5 para los modelos MTCNN y YOLO de izquierda a derecha

Tabla 39. Parámetros de análisis del modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	0	0	1	0.150	0
2	1024 × 1539	1	0	0	0.221	1
3	1024 × 879	0	0	1	0.239	0
4	1024 × 1021	1	0	0	0.206	1
5	1024 × 732	0	0	1	0.145	0
6	1024 × 1538	1	0	0	0.354	1
7	1024 × 1185	1	0	0	0.259	1
8	1024 × 640	0	0	1	0.163	0
9	1024 × 576	1	0	0	0.145	1
10	1200 × 800	0	0	1	0.243	0
11	500 × 500	0	0	1	0.093	0
12	682 × 457	0	0	1	0.111	0
13	640 × 391	0	0	1	0.080	0
14	1140 × 530	0	0	1	0.172	0
15	604 × 835	1	0	0	0.114	1

Tabla 40. Parámetros de análisis del modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° de caras detectadas
1	1024 × 680	1	0	0	2.787	1
2	1024 × 1539	0	0	1	3.520	0
3	1024 × 879	0	0	1	3.149	0
4	1024 × 1021	0	0	1	2.913	0
5	1024 × 732	0	0	1	2.538	0
6	1024 × 1538	1	0	0	3.560	1
7	1024 × 1185	1	0	0	2.899	1
8	1024 × 640	0	0	1	2.222	0
9	1024 × 576	0	0	1	2.649	0
10	1200 × 800	0	0	1	2.657	0
11	500 × 500	0	0	1	2.190	0
12	682 × 457	1	0	0	2.559	1
13	640 × 391	1	0	0	2.575	1
14	1140 × 530	0	0	1	2.335	0
15	604 × 835	1	0	0	2.653	1

Tabla 41. Parámetros de análisis del modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	1	0	0	1.845	1
2	1024 × 1539	1	0	0	1.882	1
3	1024 × 879	1	0	0	1.857	1
4	1024 × 1021	1	0	0	1.860	1
5	1024 × 732	1	0	0	1.859	1
6	1024 × 1538	1	0	0	1.875	1
7	1024 × 1185	1	0	0	1.875	1
8	1024 × 640	0	0	1	1.844	0
9	1024 × 576	0	1	1	1.875	1
10	1200 × 800	0	0	1	1.859	0
11	500 × 500	1	0	0	1.859	1
12	682 × 457	1	0	0	1.859	1
13	640 × 391	1	0	0	1.844	1
14	1140 × 530	1	0	0	1.937	1
15	604 × 835	1	0	0	1.844	1

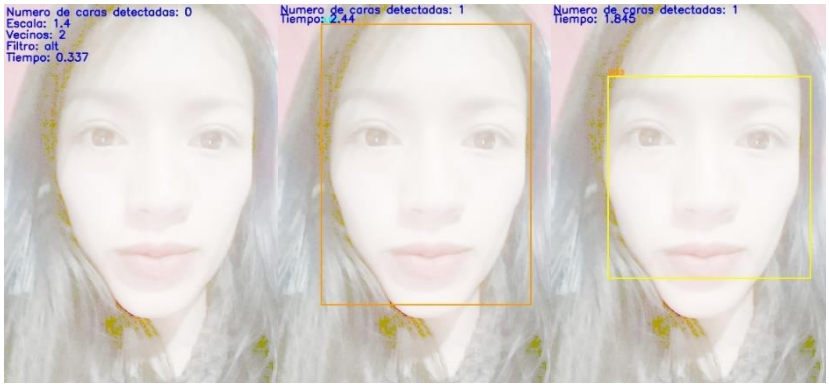


Figura 62. Imágenes normalizadas con un factor de corrección gamma de 10 para los modelos Viola & Jones. MTCNN y YOLO de izquierda a derecha

Tabla 42. Parámetros de análisis del modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	0	0	1	0.240	0
2	1024 × 1539	0	0	1	0.314	0
3	1024 × 879	0	0	1	0.347	0
4	1024 × 1021	0	0	1	0.401	0
5	1024 × 732	1	0	0	0.256	1
6	1024 × 1538	1	0	0	0.349	1
7	1024 × 1185	1	0	0	0.419	1
8	1024 × 640	0	0	1	0.334	0
9	1024 × 576	1	0	0	0.231	1
10	1200 × 800	0	0	1	0.335	0
11	500 × 500	0	0	1	0.277	0
12	682 × 457	0	0	1	0.265	0
13	640 × 391	1	0	0	0.074	1
14	1140 × 530	0	0	1	0.140	0
15	604 × 835	0	0	1	0.337	0

Tabla 43. Parámetros de análisis del modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° de caras detectadas
1	1024 × 680	0	0	1	2.396	0
2	1024 × 1539	0	0	1	3.408	0
3	1024 × 879	0	0	1	2.852	0
4	1024 × 1021	0	0	1	2.588	0
5	1024 × 732	0	0	1	2.328	0
6	1024 × 1538	0	0	1	2.979	0
7	1024 × 1185	1	0	0	3.007	1
8	1024 × 640	0	0	1	2.484	0
9	1024 × 576	0	0	1	2.258	0
10	1200 × 800	0	0	1	2.584	0
11	500 × 500	0	0	1	2.149	0
12	682 × 457	0	0	1	2.270	0
13	640 × 391	0	0	1	2.039	0
14	1140 × 530	0	0	1	2.629	0
15	604 × 835	1	0	0	2.440	1

Tabla 44. Parámetros de análisis del modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de ejecución	N° Caras Detectadas
1	1024 × 680	1	0	0	2.015	1
2	1024 × 1539	1	0	0	1.875	1
3	1024 × 879	1	0	0	1.937	1
4	1024 × 1021	1	0	0	1.859	1
5	1024 × 732	1	0	0	1.844	1
6	1024 × 1538	1	0	0	2.047	1
7	1024 × 1185	1	0	0	1.844	1
8	1024 × 640	0	0	1	1.844	0
9	1024 × 576	0	1	1	1.859	1
10	1200 × 800	0	0	1	1.875	0
11	500 × 500	1	0	0	1.969	1
12	682 × 457	1	0	0	1.859	1
13	640 × 391	1	0	0	1.859	1
14	1140 × 530	1	0	0	1.844	1
15	604 × 835	1	0	0	1.845	1

ANEXO 4: NORMALIZACIÓN DE IMÁGENES CON VARIOS ROSTROS

- Ecualización del Histograma

Tabla 45. Parámetros de análisis para el modelo Viola & Jones

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	4	1	3	1.123	5	0.80	0.57	0.67
2	1024 × 852	0	0	5	0.933	0	0.00	0.00	0.00
3	1024 × 686	1	1	2	0.775	2	0.50	0.33	0.40
4	960 × 540	19	0	1	0.816	19	1.00	0.95	0.97
5	960 × 540	7	1	1	0.529	8	0.88	0.88	0.88
6	1200 × 566	8	0	5	0.727	8	1.00	0.62	0.77
7	612 × 340	7	0	1	0.387	7	1.00	0.88	0.94
8	889 × 495	11	0	4	0.626	11	1.00	0.73	0.84
9	760 × 450	6	2	4	0.501	8	0.75	0.60	0.67
10	630 × 395	3	1	30	0.407	4	0.75	0.09	0.16
11	1200 × 565	61	2	72	0.733	63	0.97	0.46	0.62
12	1600 × 800	2	3	4	1.371	5	0.40	0.33	0.36
13	450 × 276	6	1	1	0.290	7	0.86	0.86	0.86
14	1300 × 956	16	0	0	1.223	16	1.00	1.00	1.00
15	1300 × 957	3	2	8	1.015	5	0.60	0.27	0.37
16	660 × 330	2	0	1	0.344	2	1.00	0.67	0.80
17	1200 × 566	2	1	1	0.720	3	0.67	0.67	0.67
18	259 × 194	0	0	7	0.165	0	0.00	0.00	0.00
19	300 × 168	1	0	6	0.109	1	1.00	0.14	0.25
20	1280 × 722	25	1	2	0.881	26	0.96	0.93	0.94
21	678 × 381	3	0	33	0.465	3	1.00	0.08	0.15
22	635 × 290	6	0	4	0.337	6	1.00	0.60	0.75
23	800 × 533	4	1	5	0.469	5	0.80	0.44	0.57
24	945 × 612	9	0	7	0.662	9	1.00	0.56	0.72
25	500 × 325	3	1	6	0.296	4	0.75	0.33	0.46

Tabla 46. Parámetros de análisis para el modelo MTCNN

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	7	1	0	4.221	8	0.88	1.00	0.94
2	1024 × 852	2	0	3	3.313	2	1.00	0.40	0.57
3	1024 × 686	3	1	0	3.700	4	0.75	1.00	0.86
4	960 × 540	20	1	0	2.818	21	0.95	1.00	0.97
5	960 × 540	8	0	0	2.847	8	1.00	1.00	1.00
6	1200 × 566	12	1	1	3.012	13	0.92	0.92	0.92
7	612 × 340	8	1	0	2.830	9	0.89	1.00	0.94
8	889 × 495	10	1	5	3.072	11	0.91	0.67	0.77
9	760 × 450	9	1	1	2.443	10	0.90	0.90	0.90
10	630 × 395	6	1	27	2.222	7	0.86	0.18	0.30
11	1200 × 565	66	0	67	3.986	66	1.00	0.50	0.67
12	1600 × 800	5	2	1	4.382	7	0.71	0.83	0.77
13	450 × 276	5	0	2	2.452	5	1.00	0.71	0.83
14	1300 × 956	15	0	1	3.885	15	1.00	0.94	0.97
15	1300 × 957	6	0	5	3.683	6	1.00	0.55	0.71
16	660 × 330	2	0	1	2.503	2	1.00	0.67	0.80
17	1200 × 566	3	0	0	3.097	3	1.00	1.00	1.00
18	259 × 194	0	0	7	2.346	0	0.00	0.00	0.00
19	300 × 168	3	0	4	2.154	3	1.00	0.43	0.60
20	1280 × 722	26	0	1	3.659	26	1.00	0.96	0.98
21	678 × 381	20	0	16	3.049	20	1.00	0.56	0.72
22	635 × 290	7	0	3	2.146	7	1.00	0.70	0.82
23	800 × 533	4	0	5	3.217	4	1.00	0.44	0.61
24	945 × 612	8	1	8	3.490	9	0.89	0.50	0.64
25	500 × 325	6	0	3	2.685	6	1.00	0.67	0.80

Tabla 47. Parámetros de análisis para el modelo YOLO

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	5	0	2	2.099	5	1.00	0.71	0.83
2	1024 × 852	3	0	2	1.844	3	1.00	0.60	0.75
3	1024 × 686	3	0	0	1.859	3	1.00	1.00	1.00
4	960 × 540	19	0	1	1.874	19	1.00	0.95	0.97
5	960 × 540	8	0	0	1.844	8	1.00	1.00	1.00
6	1200 × 566	9	0	4	3.924	9	1.00	0.69	0.82
7	612 × 340	8	0	0	1.844	8	1.00	1.00	1.00
8	889 × 495	14	0	1	1.875	14	1.00	0.93	0.96
9	760 × 450	10	0	0	1.859	10	1.00	1.00	1.00
10	630 × 395	30	0	3	1.844	30	1.00	0.91	0.95
11	1200 × 565	48	0	85	1.828	48	1.00	0.36	0.53
12	1600 × 800	4	0	2	3.749	4	1.00	0.67	0.80
13	450 × 276	7	0	0	1.859	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	1.984	16	1.00	1.00	1.00
15	1300 × 957	9	0	2	1.875	9	1.00	0.82	0.90
16	660 × 330	3	0	0	1.860	3	1.00	1.00	1.00
17	1200 × 566	3	0	0	1.969	3	1.00	1.00	1.00
18	259 × 194	7	1	0	1.844	8	0.88	1.00	0.94
19	300 × 168	4	0	3	1.844	4	1.00	0.57	0.73
20	1280 × 722	27	0	0	1.858	27	1.00	1.00	1.00
21	678 × 381	34	0	2	1.844	34	1.00	0.94	0.97
22	635 × 290	10	0	0	1.859	10	1.00	1.00	1.00
23	800 × 533	8	0	1	1.828	8	1.00	0.89	0.94
24	945 × 612	11	0	5	1.859	11	1.00	0.69	0.82
25	500 × 325	9	1	0	1.844	10	0.90	1.00	0.95

- Corrección Gamma con factores de 0.25. 0.5. 5 y 10

Tabla 48. Parámetros de análisis para el modelo Viola & Jones con factor de corrección gamma de 0.25

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	3	2	3	0.963	5	0.60	0.50	0.55
2	1024 × 852	0	1	5	0.657	1	0.00	0.00	0.00
3	1024 × 686	0	0	3	0.526	0	0.00	0.00	0.00
4	960 × 540	18	0	2	0.508	18	1.00	0.90	0.95
5	960 × 540	7	1	1	0.464	8	0.88	0.88	0.88
6	1200 × 566	3	0	10	0.464	3	1.00	0.23	0.37
7	612 × 340	5	0	3	0.229	5	1.00	0.63	0.77
8	889 × 495	6	0	9	0.458	6	1.00	0.40	0.57
9	760 × 450	2	1	9	0.364	3	0.67	0.18	0.28
10	630 × 395	0	0	33	0.313	0	0.00	0.00	0.00
11	1200 × 565	2	0	131	0.317	2	1.00	0.02	0.04
12	1600 × 800	0	1	6	0.866	1	0.00	0.00	0.00
13	450 × 276	6	0	1	0.255	6	1.00	0.86	0.92
14	1300 × 956	15	1	1	0.969	16	0.94	0.94	0.94
15	1300 × 957	1	0	10	0.781	1	1.00	0.09	0.17
16	660 × 330	0	0	3	0.254	0	0.00	0.00	0.00
17	1200 × 566	0	0	3	0.588	0	0.00	0.00	0.00
18	259 × 194	0	0	7	0.475	0	0.00	0.00	0.00
19	300 × 168	1	0	6	0.117	1	1.00	0.14	0.25
20	1280 × 722	22	1	5	0.799	23	0.96	0.81	0.88
21	678 × 381	1	0	35	0.294	1	1.00	0.03	0.06
22	635 × 290	5	0	5	0.296	5	1.00	0.50	0.67
23	800 × 533	0	0	9	0.426	0	0.00	0.00	0.00
24	945 × 612	7	0	9	0.545	7	1.00	0.44	0.61
25	500 × 325	1	0	8	0.275	1	1.00	0.11	0.20

Tabla 49. Parámetros de análisis para el modelo MTCNN con factor de corrección gamma de 0.25

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	4	1	3	4.177	5	0.60	0.50	0.55
2	1024 × 852	0	0	5	3.008	0	0.00	0.00	0.00
3	1024 × 686	1	0	2	2.419	1	0.00	0.00	0.00
4	960 × 540	12	0	8	2.545	12	1.00	0.90	0.95
5	960 × 540	4	0	4	2.456	4	0.88	0.88	0.88
6	1200 × 566	1	1	12	2.682	2	1.00	0.23	0.37
7	612 × 340	0	0	8	2.072	0	1.00	0.63	0.77
8	889 × 495	3	0	12	2.464	3	1.00	0.40	0.57
9	760 × 450	0	0	10	2.848	0	0.67	0.18	0.28
10	630 × 395	4	0	29	2.447	4	0.00	0.00	0.00
11	1200 × 565	0	0	133	2.744	0	1.00	0.02	0.04
12	1600 × 800	0	1	6	3.639	0	0.00	0.00	0.00
13	450 × 276	3	0	4	2.184	3	1.00	0.86	0.92
14	1300 × 956	1	1	15	3.972	2	0.94	0.94	0.94
15	1300 × 957	0	1	11	2.880	1	1.00	0.09	0.17
16	660 × 330	0	0	3	2.865	0	0.00	0.00	0.00
17	1200 × 566	1	1	2	2.924	2	0.00	0.00	0.00
18	259 × 194	0	0	7	2.049	0	0.00	0.00	0.00
19	300 × 168	1	0	6	2.357	1	1.00	0.14	0.25
20	1280 × 722	15	0	12	2.995	15	0.96	0.81	0.88
21	678 × 381	0	0	36	2.404	0	1.00	0.03	0.06
22	635 × 290	2	0	8	2.382	2	1.00	0.50	0.67
23	800 × 533	0	0	9	2.691	0	0.00	0.00	0.00
24	945 × 612	3	0	13	2.912	3	1.00	0.44	0.61
25	500 × 325	0	0	9	1.990	0	1.00	0.11	0.20

Tabla 50. Parámetros de análisis para el modelo YOLO con factor de corrección gamma de 0.25

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	2	0	4	1.892	2	1.00	0.33	0.50
2	1024 × 852	1	0	4	1.886	1	1.00	0.20	0.33
3	1024 × 686	2	0	1	1.992	2	1.00	0.67	0.80
4	960 × 540	18	0	2	2.050	18	1.00	0.90	0.95
5	960 × 540	7	0	1	1.859	7	1.00	0.88	0.94
6	1200 × 566	6	0	7	1.875	6	1.00	0.46	0.63
7	612 × 340	6	0	2	1.859	6	1.00	0.75	0.86
8	889 × 495	9	0	6	1.875	9	1.00	0.60	0.75
9	760 × 450	1	0	9	1.859	1	1.00	0.10	0.18
10	630 × 395	22	0	11	1.844	22	1.00	0.67	0.80
11	1200 × 565	2	0	131	1.860	2	1.00	0.02	0.04
12	1600 × 800	0	0	6	1.859	0	0.00	0.00	0.00
13	450 × 276	7	0	0	1.844	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	1.875	16	1.00	1.00	1.00
15	1300 × 957	4	0	7	1.859	4	1.00	0.36	0.53
16	660 × 330	1	0	2	1.844	1	1.00	0.33	0.50
17	1200 × 566	3	0	0	1.844	3	1.00	1.00	1.00
18	259 × 194	6	2	1	1.844	8	0.75	0.86	0.80
19	300 × 168	3	0	4	1.847	3	1.00	0.43	0.60
20	1280 × 722	26	0	1	1.980	26	1.00	0.96	0.98
21	678 × 381	3	0	33	2.032	3	1.00	0.08	0.15
22	635 × 290	8	0	2	1.859	8	1.00	0.80	0.89
23	800 × 533	2	0	7	1.844	2	1.00	0.22	0.36
24	945 × 612	12	0	4	1.859	12	1.00	0.75	0.86
25	500 × 325	8	0	1	1.828	8	1.00	0.89	0.94

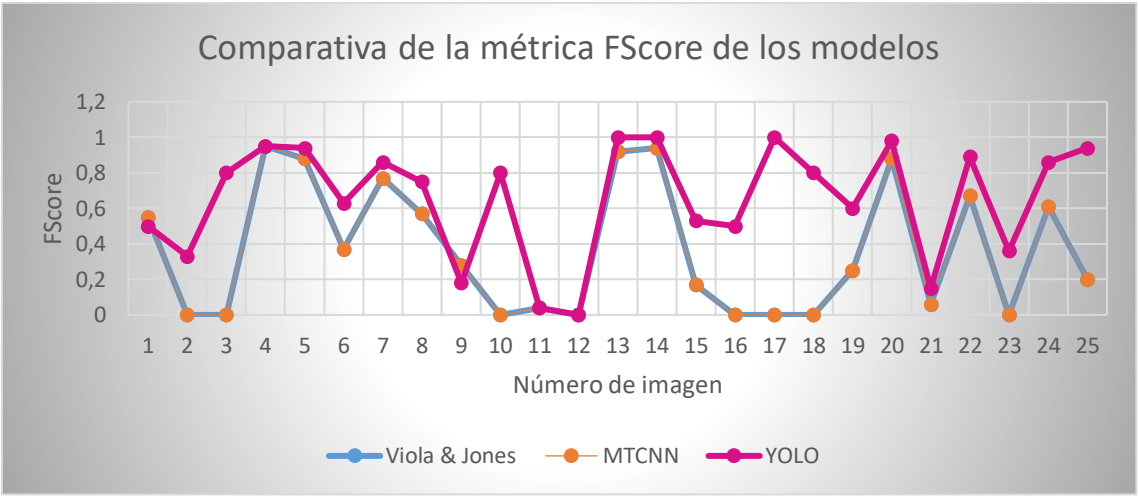


Figura 63. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO en imágenes normalizadas aplicando corrección gamma con un factor de 0.25

Tabla 51. Parámetros de análisis para el modelo Viola & Jones con factor de corrección gamma de 5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	2	1	4	0.592	3	0.67	0.33	0.44
2	1024 × 852	0	0	5	0.698	0	0.00	0.00	0.00
3	1024 × 686	1	0	2	0.768	1	1.00	0.33	0.50
4	960 × 540	18	0	2	0.547	18	1.00	0.90	0.95
5	960 × 540	7	1	1	0.595	8	0.88	0.88	0.88
6	1200 × 566	6	0	7	0.765	6	1.00	0.46	0.63
7	612 × 340	7	0	1	0.385	7	1.00	0.88	0.94
8	889 × 495	9	0	6	0.571	9	1.00	0.60	0.75
9	760 × 450	6	1	4	0.621	7	0.86	0.60	0.71
10	630 × 395	0	0	33	0.347	0	0.00	0.00	0.00
11	1200 × 565	58	3	75	0.701	61	0.95	0.44	0.60
12	1600 × 800	2	1	4	1.382	3	0.67	0.33	0.44
13	450 × 276	4	1	3	0.266	5	0.80	0.57	0.67
14	1300 × 956	14	0	2	0.944	14	1.00	0.88	0.94
15	1300 × 957	2	1	9	1.218	3	0.67	0.18	0.28
16	660 × 330	2	0	1	0.305	2	1.00	0.67	0.80
17	1200 × 566	2	1	1	0.642	3	0.67	0.67	0.67
18	259 × 194	0	0	7	0.258	0	0.00	0.00	0.00
19	300 × 168	1	0	6	0.171	1	1.00	0.14	0.25
20	1280 × 722	24	1	3	0.964	25	0.96	0.89	0.92
21	678 × 381	3	0	33	0.388	3	1.00	0.08	0.15
22	635 × 290	4	0	6	0.388	4	1.00	0.40	0.57
23	800 × 533	3	0	6	0.514	3	1.00	0.33	0.50
24	945 × 612	7	1	9	0.692	8	0.88	0.44	0.59
25	500 × 325	4	1	5	0.416	5	0.80	0.44	0.57

Tabla 52. Parámetros de análisis para el modelo MTCNN con factor de corrección gamma de 0.5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	5	0	2	3.684	5	1.00	0.71	0.83
2	1024 × 852	2	0	3	2.930	2	1.00	0.40	0.57
3	1024 × 686	2	0	1	2.828	2	1.00	0.67	0.80
4	960 × 540	17	0	3	2.730	17	1.00	0.85	0.92
5	960 × 540	8	1	0	2.838	9	0.89	1.00	0.94
6	1200 × 566	5	0	8	2.869	5	1.00	0.38	0.55
7	612 × 340	6	0	2	2.376	6	1.00	0.75	0.86
8	889 × 495	7	1	8	3.716	7	0.88	0.47	0.61
9	760 × 450	2	0	8	2.704	2	1.00	0.20	0.33
10	630 × 395	6	0	27	2.710	6	1.00	0.18	0.31
11	1200 × 565	6	0	127	2.674	6	1.00	0.05	0.10
12	1600 × 800	3	0	3	4.358	3	1.00	0.50	0.67
13	450 × 276	4	1	3	2.328	5	0.80	0.57	0.67
14	1300 × 956	8	0	8	3.773	8	1.00	0.50	0.67
15	1300 × 957	1	0	10	3.542	1	1.00	0.09	0.17
16	660 × 330	1	0	2	2.396	1	1.00	0.33	0.50
17	1200 × 566	2	0	1	2.977	2	1.00	0.67	0.80
18	259 × 194	0	0	7	2.437	0	0.00	0.00	0.00
19	300 × 168	3	0	4	2.134	3	1.00	0.43	0.60
20	1280 × 722	26	1	1	3.486	26	0.96	0.96	0.96
21	678 × 381	3	0	33	2.576	3	1.00	0.08	0.15
22	635 × 290	6	1	4	2.508	7	0.86	0.60	0.71
23	800 × 533	3	0	6	3.149	3	1.00	0.33	0.50
24	945 × 612	7	0	9	3.149	7	1.00	0.44	0.61
25	500 × 325	6	0	3	2.250	6	1.00	0.67	0.80

Tabla 53. Parámetros de análisis para el modelo YOLO con factor de corrección gamma de 5

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	3	0	4	1.859	3	1.00	0.43	0.60
2	1024 × 852	2	0	3	1.859	2	1.00	0.40	0.57
3	1024 × 686	3	0	0	1.859	3	1.00	1.00	1.00
4	960 × 540	18	0	2	1.849	18	1.00	0.90	0.95
5	960 × 540	8	0	0	1.844	8	1.00	1.00	1.00
6	1200 × 566	7	0	6	3.953	7	1.00	0.54	0.70
7	612 × 340	8	0	0	1.844	8	1.00	1.00	1.00
8	889 × 495	15	0	0	1.845	15	1.00	1.00	1.00
9	760 × 450	9	0	1	1.828	9	1.00	0.90	0.95
10	630 × 395	4	0	32	2.015	4	1.00	0.11	0.20
11	1200 × 565	9	0	124	1.860	9	1.00	0.07	0.13
12	1600 × 800	3	0	3	1.875	3	1.00	0.50	0.67
13	450 × 276	7	0	0	1.859	7	1.00	1.00	1.00
14	1300 × 956	16	0	0	2.062	16	1.00	1.00	1.00
15	1300 × 957	10	0	1	1.859	10	1.00	0.91	0.95
16	660 × 330	1	0	2	1.844	1	1.00	0.33	0.50
17	1200 × 566	3	0	0	1.844	3	1.00	1.00	1.00
18	259 × 194	7	0	0	1.828	7	1.00	1.00	1.00
19	300 × 168	3	0	4	1.828	3	1.00	0.43	0.60
20	1280 × 722	27	0	0	1.847	27	1.00	1.00	1.00
21	678 × 381	34	0	2	1.844	34	1.00	0.94	0.97
22	635 × 290	10	0	0	1.859	10	1.00	1.00	1.00
23	800 × 533	4	0	5	2.015	4	1.00	0.44	0.61
24	945 × 612	13	0	3	1.844	13	1.00	0.81	0.90
25	500 × 325	9	0	0	1.859	9	1.00	1.00	1.00

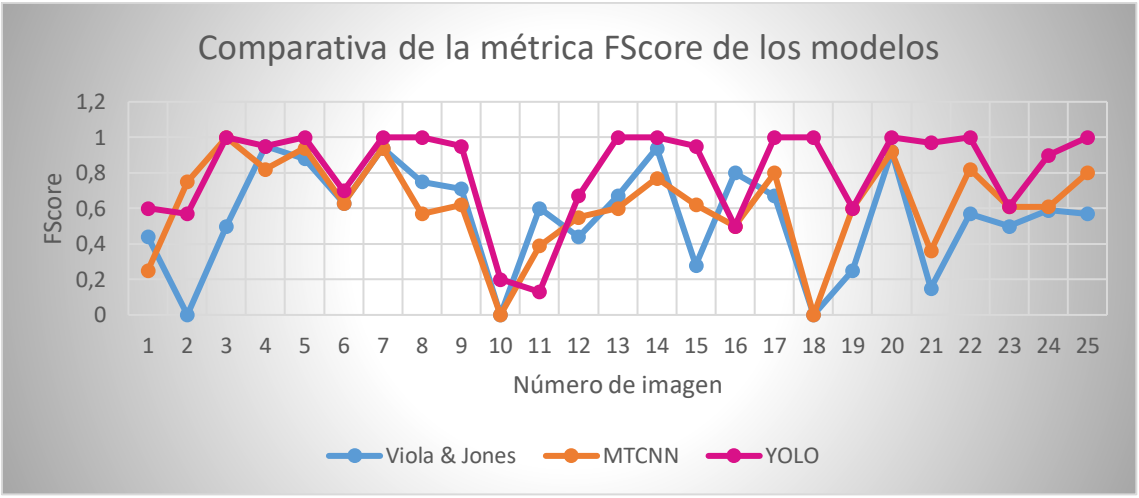


Figura 64. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO en imágenes normalizadas aplicando corrección gamma con un factor de 5, 0.5 y 5 respectivamente.

Tabla 54. Parámetros de análisis para el modelo Viola & Jones con factor de corrección gamma de 10

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	0	0	7	0.451	0	0.00	0.00	0.00
2	1024 × 852	0	0	5	0.545	0	0.00	0.00	0.00
3	1024 × 686	1	0	2	0.749	1	1.00	0.33	0.50
4	960 × 540	10	0	10	0.553	10	1.00	0.50	0.67
5	960 × 540	7	1	1	0.516	8	0.88	0.88	0.88
6	1200 × 566	7	0	6	0.581	7	1.00	0.54	0.70
7	612 × 340	7	0	1	0.303	7	1.00	0.88	0.94
8	889 × 495	10	1	5	0.545	11	0.91	0.67	0.77
9	760 × 450	5	2	6	0.553	7	0.71	0.45	0.55
10	630 × 395	0	0	33	0.235	0	0.00	0.00	0.00
11	1200 × 565	31	2	102	0.839	33	0.94	0.23	0.37
12	1600 × 800	3	2	3	1.272	5	0.60	0.50	0.55
13	450 × 276	0	1	7	0.303	1	0.00	0.00	0.00
14	1300 × 956	10	0	6	0.858	10	1.00	0.63	0.77
15	1300 × 957	2	0	9	1.134	2	1.00	0.18	0.31
16	660 × 330	1	0	2	0.277	1	1.00	0.33	0.50
17	1200 × 566	1	1	2	0.543	2	0.50	0.33	0.40
18	259 × 194	0	0	7	0.277	0	0.00	0.00	0.00
19	300 × 168	1	0	6	0.131	1	1.00	0.14	0.25
20	1280 × 722	15	0	12	0.642	15	1.00	0.56	0.72
21	678 × 381	2	0	34	0.477	2	1.00	0.06	0.11
22	635 × 290	3	0	7	0.354	3	1.00	0.30	0.46
23	800 × 533	3	0	6	0.494	3	1.00	0.33	0.50
24	945 × 612	6	1	10	0.658	7	0.86	0.38	0.53
25	500 × 325	1	1	8	0.395	2	0.50	0.11	0.18

Tabla 55. Parámetros de análisis para el modelo MTCNN con factor de corrección gamma de 10

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	0	0	7	2.642	0	0.00	0.00	0.00
2	1024 × 852	0	0	5	2.585	0	0.00	0.00	0.00
3	1024 × 686	0	0	3	2.312	0	0.00	0.00	0.00
4	960 × 540	4	0	16	2.642	4	1.00	0.20	0.33
5	960 × 540	4	0	4	2.456	4	1.00	0.50	0.67
6	1200 × 566	4	0	9	2.657	4	1.00	0.31	0.47
7	612 × 340	5	0	3	2.103	5	1.00	0.63	0.77
8	889 × 495	3	0	12	2.533	3	1.00	0.20	0.33
9	760 × 450	3	0	7	2.200	3	1.00	0.30	0.46
10	630 × 395	0	0	33	1.820	0	0.00	0.00	0.00
11	1200 × 565	2	0	131	2.955	2	1.00	0.02	0.04
12	1600 × 800	3	0	3	3.214	3	1.00	0.50	0.67
13	450 × 276	0	0	7	1.875	0	0.00	0.00	0.00
14	1300 × 956	5	1	11	3.032	6	0.83	0.31	0.45
15	1300 × 957	1	0	10	3.056	1	1.00	0.09	0.17
16	660 × 330	1	0	2	2.243	1	1.00	0.33	0.50
17	1200 × 566	0	0	3	2.620	0	0.00	0.00	0.00
18	259 × 194	0	0	7	1.864	0	0.00	0.00	0.00
19	300 × 168	0	0	7	1.852	0	0.00	0.00	0.00
20	1280 × 722	4	0	23	2.744	4	1.00	0.15	0.26
21	678 × 381	3	0	33	2.160	3	1.00	0.08	0.15
22	635 × 290	1	0	9	2.220	1	1.00	0.10	0.18
23	800 × 533	3	0	6	2.388	3	1.00	0.33	0.50
24	945 × 612	5	0	11	2.599	5	1.00	0.31	0.47
25	500 × 325	3	0	6	2.094	3	1.00	0.33	0.50

Tabla 56. Parámetros de análisis para el modelo YOLO con factor de corrección gamma de 10

N°	Dimensión	VP	FP	FN	Tiempo de Ejecución (s)	N° Caras Detectadas	P	R	FS
1	1024 × 1366	0	0	7	1.869	0	0.00	0.00	0.00
2	1024 × 852	1	0	5	2.078	1	1.00	0.17	0.29
3	1024 × 686	3	0	0	2.580	3	1.00	1.00	1.00
4	960 × 540	16	0	4	1.884	16	1.00	0.80	0.89
5	960 × 540	8	0	0	1.859	8	1.00	1.00	1.00
6	1200 × 566	5	0	8	2.054	5	1.00	0.38	0.55
7	612 × 340	8	0	0	1.828	8	1.00	1.00	1.00
8	889 × 495	14	0	1	1.927	14	1.00	0.93	0.96
9	760 × 450	8	0	2	1.859	8	1.00	0.80	0.89
10	630 × 395	0	0	33	1.828	0	0.00	0.00	0.00
11	1200 × 565	2	0	131	1.859	2	1.00	0.02	0.04
12	1600 × 800	1	0	5	1.875	1	1.00	0.17	0.29
13	450 × 276	5	0	2	1.844	5	1.00	0.71	0.83
14	1300 × 956	15	0	1	1.922	15	1.00	0.94	0.97
15	1300 × 957	7	0	4	1.859	7	1.00	0.64	0.78
16	660 × 330	1	0	2	1.844	1	1.00	0.33	0.50
17	1200 × 566	2	0	1	1.875	2	1.00	0.67	0.80
18	259 × 194	2	0	5	1.844	2	1.00	0.29	0.45
19	300 × 168	1	0	6	1.844	1	1.00	0.14	0.25
20	1280 × 722	23	0	4	1.865	23	1.00	0.85	0.92
21	678 × 381	27	1	9	1.828	28	0.96	0.75	0.84
22	635 × 290	6	0	4	5.281	6	1.00	0.60	0.75
23	800 × 533	3	0	6	1.844	3	1.00	0.33	0.50
24	945 × 612	9	0	5	1.844	9	1.00	0.64	0.78
25	500 × 325	8	0	1	1.906	8	1.00	0.89	0.94

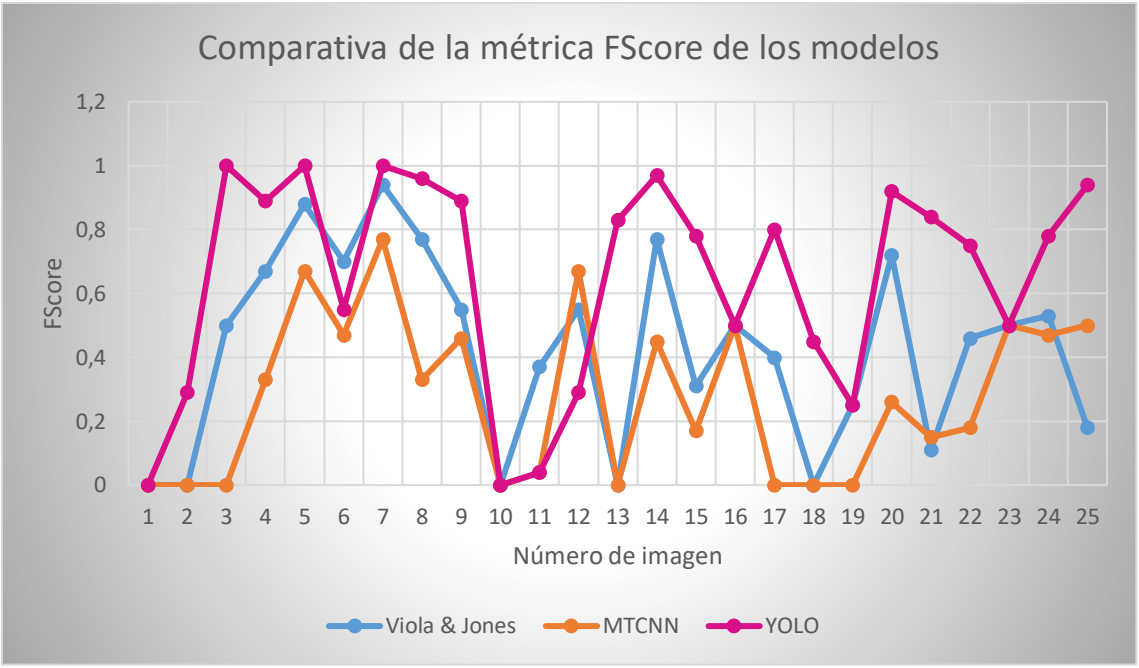


Figura 65. Análisis del parámetro F-Score en los modelos Viola & Jones, MTCNN y YOLO en imágenes normalizadas aplicando corrección gamma con un factor de 10

ANEXO 5: TABLA DE VALORACIÓN DE LENGUAJES DE PROGRAMACIÓN

Lenguaje / Características	Python	JavaScript	Java	C, C#, C++
Código Abierto	x	x	x	x
Interpretado	x	x		
Multiplataforma	x	x	x	x
Orientada a Objetos	x	x	x	
Alto Nivel	x	x	x	
Top 1 según GitHub	x			
Top 1 según IEEE Spectrum	x			
Top 1 según TIOBE			x	
Facilidad de sintaxis	x			
Tipado dinámico	x			
Facilidad de aprendizaje	x			
Gran cantidad de modelos de ML* implementados	x			
Herramientas para creación de GUI	x	x	x	
Disponibilidad para trabajar con Google Colab	x			
Recolector de basura	x	x	x	
Mayor Velocidad				x
Mayor cantidad de librerías para ML*	x			
Mayor Seguridad			x	
Mayor Comunidad de Programadores				x
Gran cantidad de Libros con base técnica			x	x
Total	**16/20**	**7/20**	**9/20**	**5/20**

*ML significa Machine Learning

ANEXO 6: TABLA DE VALORACIÓN DE ALGORITMOS DE DETECCIÓN

Algoritmo / Características	Viola & Jones	HOG	SVM	DFA-SVM	ARN
Mayor Precisión					x
Menor Complejidad matemática	x				x
Mayor Velocidad	x				
Mayor Cantidad de VP					x
Menor Cantidad de FP	x				
Menor Cantidad de FN					x
Trabajos específicos para detección facial	x			x	x
Repercusión en trabajos relacionados con detección	x		x		
Implementadas en Python	x				
Uso de clasificadores simples	x	x			
Tolerancia a cambios en la rotación (ángulo)			x	x	x
Tolerancia a cambios de iluminación				x	x
Trabajan en RGB		x			x
Facilidad de entrenamiento					x
Mayor cantidad de datos de entrenamiento					x
Total	**7/20**	**2/20**	**2/20**	**3/20**	**10/20**

ANEXO 7: TRABAJOS FUTUROS

La detección de rostros en sus etapas iniciales se realizaba sobre imágenes previamente capturadas, este proceso aportó en gran medida al estudio de diferentes técnicas de detección tales como Viola & Jones, HOG, DFA-SVM, etc.; sin embargo, debido a las necesidades actuales de diferentes entidades con respecto a seguridad, control de acceso, conteo de personas, etc., es prioritario profundizar en la investigación de técnicas de detección que permitan detectar rostros en tiempo real, las cuales brindarán una solución rápida y eficaz a los problemas mencionados anteriormente.

Así mismo, la detección de rostros puede ser aplicada para reconocer emociones a través del análisis de expresiones faciales, lo cual permite mejorar diferentes procesos en el sector comercial, educativo, medicinal e incluso en la construcción de relaciones sólidas entre padres, hijos, amigos, etc. Por ejemplo, en el sector educativo se puede realizar un estudio estadístico con los gestos faciales manifestados por cada estudiante, estableciendo parámetros de progreso y de esta manera optimizando el proceso de enseñanza y convivencia.

Además, el algoritmo seleccionado en el presente trabajo puede formar parte de una plataforma integra de visión por computador, en la que se incluya dispositivos electrónicos para la adquisición de imágenes o vídeo y un sistema que permita detectar las expresiones faciales o realizar reconocimiento facial una vez detectado el rostro.

En el sector comercial, un sistema de detección facial en conjunto con un sistema de reconocimiento puede implementarse para realizar retiros de dinero en cajeros automáticos, la etapa de reconocimiento facial se encargará de verificar los datos biométricos del usuario, mientras que la etapa de detección facial se encargará de analizar los gestos que realiza con el fin de evitar posibles robos.

En adición, un sistema de detección facial puede ser utilizado para realizar un análisis de mercado; es decir, al detectar el rostro de una persona se puede determinar la dirección en la que está observando diferentes anuncios de publicidad, lo cual permite evaluar sus preferencias en varios sectores comerciales.

La detección facial en fotografías puede mejorar el control de asistencia de alumnos a eventos realizados o programados dentro de la malla curricular; es decir, al finalizar cada

evento se puede tomar una fotografía de todos los asistentes la cual será procesada en un sistema de detección y determinará el porcentaje de asistencia; posteriormente se puede acoplar un sistema de reconocimiento facial para determinar la identidad de cada asistente.

Para controlar la cantidad de personas que ingresan a un lugar específico como, por ejemplo: laboratorios, ascensores, salas de computo, sala de profesores, etc.; se puede implementar un sistema de detección facial, el cual almacene en una variable la cantidad de caras detectadas y que al superar cierto umbral (aforo) emita una alarma que advierta de este inconveniente.

Finalmente, en el caso de videoconferencias se puede utilizar un sistema de detección facial que determine la posición del rostro, y realice un seguimiento del mismo, de forma que la cámara ajuste su campo de visión adecuadamente o en su defecto, que el computador emita una alarma en caso de que el usuario salga del campo de visión.