PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito parcial para obtener el Título de INGENIERO DE SISTEMAS

Diseño e implementación de un estimador de edad, género y raza con imágenes de rostros a través de Inteligencia Artificial

Por

Br. Ender José Peña Sánchez

Tutor: Yaneth Moreno, M.Sc

Mérida, Agosto de 2021

Resumen

La presente investigación tiene como objetivo el diseño e implementación de un prototipo de procesador de imágenes de rostros para la estimación de tres características faciales: el género, la raza y la edad. El estudio se llevó a cabo a través de la metodología CRISP-DM. Se realizó un procesamiento previo de los datos, los cuales consistían en 23708 imágenes de rostros. Se desarrollaron modelos de *Machine Learning*, extrayendo previamente características a través de descriptores locales; y *Deep Learning* mediante la técnica de *Transfer Learning* en una red neuronal con una estructura *Multitask*. Para ambos se diseñó y ejecutó un esquema de entrenamiento y calibración de parámetros de los algoritmos, a fin de obtener el mejor desempeño posible. Los resultados obtenidos muestran que ambos enfoques de la Inteligencia Artificial son técnicas apropiadas para el desarrollo del prototipo planteado en esta investigación. Para la implementación fue añadida una funcionalidad de detección facial, y se utilizó el modelo de *Deep Learning*, que demostró en términos generales un mejor desempeño. Este prototipo puede ser utilizado como procesador de imágenes para detección y análisis facial.

Palabras clave: Inteligencia artificial, Aprendizaje automático, Redes neuronales.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La Inteligencia Artificial (IA) es una vertiente que ha venido adquiriendo fuerza en los últimos años y que ha revolucionado la tecnología. Esta busca poder crear entidades con capacidades similares a los seres humanos y que nos sorprende continuamente cuando vemos las innovaciones dadas a conocer por grandes empresas a lo largo del mundo. Este auge tecnológico y el avanzado proceso de globalización que se desarrolla cada día hace que la generación e intercambio de información, por parte de los seres humanos, sea inconmensurablemente grande. Se habla entonces de *Big Data*, la cual está siendo impulsada por la IA, debido a que esta ha desarrollado la capacidad de procesar grandes cantidades de datos dando como resultado la creación de sistemas más robustos, modernos y capaces de solucionar problemas cada vez más complejos.

El *Machine Learning* (ML) o Aprendizaje Automático y el *Deep Learning* (DL) o Aprendizaje Profundo son dos áreas de la Inteligencia Artificial que han permitido construir sistemas que aprenden a hacer alguna tarea o conjunto de tareas y funciones. Ambas areas abordan los problemas e implementan soluciones de manera distinta, pero así mismo ambas han dado resultados increíbles que han hecho aumentar el interés en los sistemas inteligentes.

El análisis de imágenes es una de las categorías más estudiadas por el ML y el DL, particularmente imágenes de rostros. Esto ha ido evolucionando continuamente, desde la detección de personas en una imagen o incluso video (un video puede ser visto como una secuencia de cuadros-imágenes) hasta el reconocimiento de esta (es decir,

determinar su identidad), y en última instancia, conocer además otras características o atributos que puedan determinarse a partir del análisis de estas imágenes. La información que puede aportar el rostro de una persona es aplicada en ámbitos como aeropuertos, cámaras de seguridad, redes sociales, ciencias forenses, por nombrar algunos, y son utilizados para distintos fines. Sin embargo, aún existen limitaciones en cuanto a qué tanta información puede obtenerse con estas tecnologías y cuán relevante es para que cumpla con los fines esperados y con un rendimiento aceptable. Esta investigación se enfoca en el análisis de imágenes de rostros para determinar tres atributos de una persona: la edad, el género y la raza; a través de la aplicación de diferentes técnicas de IA.

1.1. Antecedentes

El estudio de imágenes faciales se ha incrementado en los últimos años y se ha expandido logrando la implementación de distintas tecnologías innovadoras. Algo comúnmente realizado es invertir grandes esfuerzos en procesamiento de las imágenes previo a la aplicación de técnicas de IA para construir sistemas que detecten, reconozcan o incluso estimen y clasifiquen rostros según algún criterio. Sobre esto, Dharavath, Talukdar, y Laskar (2014) en su investigación titulada *"Improving Face Recognition Rate with Image Preprocessing"* exponen un estudio de las técnicas de preprocesamiento de imágenes más usadas, tales como el recortado, redimensionado, normalización y filtrado. Además de aplicar estas técnicas, posteriormente utilizan extracción de características para el reconocimiento de rostros. Demostrando cómo la aplicación de métodos de preprocesamiento como primer paso en el estudio del reconocimiento facial puede aumentar notablemente el desempeño.

Otro de los elementos más considerados en el estudio de imágenes en el campo de la IA, particularmente en el ML, es la representación que se le da a dichas imágenes como *Características*. En este sentido, múltiples investigaciones se han llevado a cabo para estudiar distintas estrategias para representar imágenes. En el trabajo titulado *"Feature Extraction and Representation for Face Recognition"* Sarfraz, Hellwich, y Riaz (2010) se estudió una de las más relevantes técnicas de extracción de características usadas en reconocimiento de rostros en 2D, haciendo comparaciones

entre las holísticas y las locales. Realizaron pruebas con tres descriptores: *Eigenfaces* o Caras propias, Gabor, y los Histogramas de Patrones Locales Binarios (LBPH, por sus siglas en inglés), haciendo notar que la existencia de condiciones variantes en las imágenes que no sean resueltas previamente, como pose, desalineaciones, iluminación, etc., puede afectar el rendimiento de la tarea de reconocimiento. Además exponen la necesidad de implementar métodos más sofisticados y solucionar estas variaciones para obtener mejores resultados.

Por otro lado Dalal y Triggs (2005), en su obra *"Histograms of Oriented Gradients for Human Detection"*, estudiaron la aplicación de Histogramas de Gradientes Orientados (HOG, por sus siglas en inglés) normalizados como características de imágenes para la detección de personas. Demostraron que este método da muy buenos resultados, especialmente cuando se hace un estudio adecuado de los parámetros del descriptor y se aplica normalización de contraste para obtener mayor robustez ante variaciones de iluminación y sombras.

La estimación de características como la edad, género y raza han sido estudiadas últimamente como complemento y profundización de los avances obtenidos en la detección y reconocimiento de rostros. En la investigación *"Age and Gender Estimation of Unfiltered Faces"*, de Eidinger, Enbar, y Hassner (2014), se llevó a cabo un estudio para la estimación de edad y género estableciendo un proceso que consiste en varios pasos. En primer lugar, la detección del rostro, luego los rostros detectados son alineados, para posteriormente representar estos píxeles a través de LBPH. Finalmente, construyen una Máquina de Vectores de Soporte (SVM, por sus siglas en inglés), con *kernel* lineal al que, adicionalmente, le aplican la técnica *Dropout*, usada en redes neuronales para evitar problemas de sobreajuste.

En las anteriores investigaciones descritas, la corriente de IA empleada es el ML; sin embargo, el DL está siendo cada vez más usado para el análisis de imágenes. Tal es el caso de la investigación de Levi y Hassner (2015), *"Age and Gender Classification using Convolutional Neural Networks"*, donde se desarrolló la implementación de una Red Neuronal Convolucional (CNN, por sus siglas en inglés) para estimar la edad y género de imágenes de rostros. Propusieron una red de estructura sencilla y añadieron

al conjunto de datos de entrenamiento versiones recortadas de las imágenes, logrando buenos resultados en la predicción de ambas características, aún contando con poca data para entrenar el modelo.

Las arquitecturas de las redes neuronales han sido variadas y se han planteado estrategias alternativas que han dado excelentes resultados. Ito, Kawai, Okano, y Aoki (2018), en su investigación titulada *"Age and Gender Prediction from Face Images Using Convolutional Neural Network"*, plantearon un esquema predictivo de edad y género utilizando CNN y haciendo uso del enfoque *multi-task* o multi-tarea. Esto consiste en construir una red neuronal cuya capa base (capa de convoluciones) es compartida con un conjunto de capas para cada etiqueta (edad y género), lo que permite realizar múltiples tareas (o predicciones) de manera simultánea. Además demostraron que este enfoque mejora el desempeño de los modelos en cuanto a exactitud de las predicciones y el tiempo de ejecución, respecto a los esquemas *single-task* o aquellos en donde se realiza una sola tarea.

Otra estrategia interesante es la desarrollada en el trabajo *"A deep analysis on age estimation"*, propuesta por Huerta, Fernández, Segura, Hernando, y Prati (2015), en el cual plantean un modelo de estimación de edad abordando dos vías: por una parte, a través de la fusión de descriptores basados en textura y apariencia local y el uso de estos junto a un clasificador; y por otro lado mediante redes neuronales. Realizaron evaluaciones de ambos métodos variando las configuraciones de los hiperparámetros de los modelos.

Esta estrategia de fusión de características y variaciones de hiperparámetros, además de otras técnicas, se observan también en el trabajo titulado *"Gender Classification Based on Fusion of Different Spatial Scale Features Selected by Mutual Information From Histogram of LBP, Intensity, and Shape"*, Tapia y Perez (2013), donde se aplica un proceso de selección de características basado en Información Mutua (MI, por sus siglas en inglés) para mejorar la clasificación de género en imágenes faciales. Además emplean la fusión de distintos descriptores de intensidad, forma y textura, así como LBP en diferentes escalas y radios. Haciendo diferentes variaciones de prueba, demostraron que las estrategias planteadas mejoran las técnicas comúnmente usadas donde se emplean en su mayoría, uno u otro descriptor y utilizando una escala en específico.

Más recientemente, también se han implementado plataformas en tiempo real. Azarmehr, Laganiere, Lee, Xu, y Laroche (2015) en su investigación *"Real-time Embedded Age and Gender classification in unconstrained video"*, presentaron un *framework* completo para clasificación de edad y género en videos en tiempo real. Se hace uso de alineamiento de rostros, normalización de iluminación y además se aplica el descriptor de características locales de textura LBP para múltiples resoluciones. Para mejorar el rendimiento emplean una técnica de reducción de dimensionalidad usando Análisis Discriminante Mejorado (EDA, por sus siglas en inglés) y un SVM no lineal.

1.2. Planteamiento del Problema

Los seres humanos tenemos una habilidad notablemente buena para reconocer rostros. Generalmente, podemos recordar a alguien a quien no hemos visto durante años, inmediatamente después de observar su rostro, aun cuando el peinado o el bronceado sean distintos de cómo lo recordamos.

No obstante, cuando se trata de emular estas capacidades de reconocimiento facial a través de la IA, surgen grandes dificultades. Cuando un sistema computacional intenta analizar rostros deberá hacerlo estudiando imágenes que usualmente están expuestas a variedad de condiciones ambientales, tales como: iluminación, pose, ruido, etc. Entrenar un modelo de IA que sea robusto frente a este y otros inconvenientes no es tan sencillo y suele requerir de las técnicas apropiadas, suficientes datos y experiencia. Con esto en mente, es razonable asumir que la dificultad para analizar con mayor profundidad el rostro humano se incrementa, ya que aumentaría la complejidad del sistema de IA.

La mayoría de los sistemas inteligentes implementados hoy día consisten en la detección de rostros y de su reconocimiento. Tal es el caso de las aplicaciones de seguridad de alguna empresa, donde se compara si el rostro analizado es de un trabajador de la misma (en otras palabras, si pertenece a la base de datos de la plataforma). Sin embargo, es escasa la implementación de tecnologías que permitan el estudio de algo más que la existencia o no de un rostro en una imagen y de determinar quién es la persona que aparece en ella.

La edad, el género o la raza de una persona no son fáciles de determinar, puede haber múltiples parámetros que influyan en la determinación de estos rasgos que pueden ser complejos y variados. Es por esto que se propone el desarrollo de un sistema de análisis facial, que a través de las herramientas que la IA puede ofrecer, permita el procesado de imágenes para la estimación de estas características.

1.3. Justificación

La detección y reconocimiento facial está cada vez siendo más usado, gracias a que la tecnología permite realizar tareas de manera más rápida y, en algunos casos, más eficiente que los seres humanos. En un entorno donde se emplee la imagen del rostro de una persona para obtener alguna información y satisfacer algún fin, como ya se ha implementado en muchas áreas, ser capaz de adquirir información adicional relevante no trae más que beneficios en pro de mejorar el desempeño de estas tecnologías. En los sistemas de cámaras de vigilancia como los existentes en los aeropuertos, centros comerciales, entre otras instalaciones; obtener estas características puede resultar muy útil para el caso de situaciones fuera de lo común que pongan en riesgo la seguridad de las personas. Por otro lado, las investigaciones criminales y forenses pueden valerse de cualquier cualidad de la víctima para acercarlos a una investigación exitosa. También en redes sociales y sistemas de interacción humano-computadora se pueden implementar tecnologías más eficaces y personalizables al poder determinar con mayor precisión el perfil del usuario. Otro ámbito a destacar es el análisis migratorio, donde se podrían obtener estadísticas interesantes para analizar el flujo de personas entre países según sus características demográficas.

Esta investigación busca estudiar el uso de la IA para profundizar el análisis facial y determinar atributos adicionales de los rostros tales como: la edad, el género y la raza, a través de una imagen del rostro. Esto puede ser utilizado para identificar con mayor rapidez y precisión a una persona, además de proporcionar una mejor descripción de esta, dar recomendaciones personalizadas o recabar información útil, entre otros.

1.4. Objetivos

El objetivo general de este trabajo es desarrollar un prototipo de procesador de imágenes de rostros a través de Inteligencia Artificial para la estimación de la edad, el género y la raza.

Para esto se deben cumplir los siguientes objetivos específicos:

1. Recopilar la data a utilizar para el entrenamiento de los modelos.
2. Analizar la data para la identificación del acondicionamiento necesario.
3. Realizar el preprocesamiento pertinente de los datos.
4. Determinar el conjunto de técnicas y herramientas en el campo de la Inteligencia Artificial que servirán de base para el estudio.
5. Diseñar los modelos utilizando las técnicas y herramientas escogidas.
6. Realizar los entrenamientos y pruebas de desempeño.
7. Implementar el sistema de procesamiento de imágenes.

1.5. Metodología

En el desarrollo de proyectos en áreas como Ciencia de Datos e Inteligencia Artificial, es vital hacer uso de procedimientos que permitan realizar un estudio ordenado, claro y preciso de los datos y las herramientas empleadas para su procesamiento, así como facilitar su interpretación y resultados. Una de las metodologías más usadas es la llamada CRISP-DM (*Cross Industry Standard Process for Data Mining*), tal como se expone en (Shearer, 2000) consta de varias fases, en cada una de las cuales hay tareas o pasos que llevar a cabo. Estas fases son las siguientes:

Fase 1: Comprensión del problema:

En esta fase se clarifican los problemas, metas y recursos. Se analiza el problema en cuestión y las dificultades que supone. Se define el criterio para determinar el

éxito del proceso en base a una perspectiva de negocio. Una vez definida la meta, se analizan qué datos están disponibles para el estudio, así como sus restricciones. En esta investigación no se toman en cuenta objetivos de negocio como tal ya que tiene un propósito investigativo; en su lugar se analiza el problema desde una perspectiva general.

Fase 2: Comprensión de los datos

Esta fase implica darle una mirada detallada y minuciosa a los datos a utilizar. Se hace una exploración a través de tablas y gráficos que permitan un entendimiento amplio y profundo de estos y así evitar problemas en la fase siguiente. Las tareas son:

- **Recolección**: adquirir los datos seleccionados para el estudio.
- **Descripción:** principalmente bajo un enfoque de calidad y cantidad, así como el formato de los datos.
- **Exploración de los datos:** a través de tablas, gráficos, y cualquier otra herramienta de visualización. Este análisis permite entender cómo abordar el problema para lograr el objetivo planteado, así como ayudar a determinar las transformaciones de los datos que podrían ser necesarias más adelante.
- **Verificación de calidad:** en esta tarea debe determinarse si hay errores, valores faltantes o inconsistencias en los datos que puedan entorpecer el proceso de desarrollo.

Fase 3: Preparación de los datos

Este es uno de los aspectos más importante en la minería de datos, así como el paso que suele consumir más tiempo. Tiene como tareas:

- **Selección de los datos**: aquí se escogen los datos a utilizar. Generalmente hay dos formas de hacerlo: ya sea por ítems (muestras) o por atributos/características.

- **Limpieza de los datos:** en esta tarea deben tratarse los problemas encontrados en la tarea de verificación de la fase anterior.

- **Construcción de nuevos datos:** en caso de ser necesario, se pueden aplicar transformaciones a los datos, resultando en la construcción de datos nuevos, como generar nuevas muestras, o extracción de características.

- **Formatear datos:** algunos modelos requieren que los datos se presenten en cierta forma o estructura, por lo que puede ser necesario cambiar el formato de presentación de estos.

Fase 4: Modelado

En esta fase se seleccionan una o varias técnicas de modelado, se diseñan y construyen los modelos requeridos y se realiza una calibración de sus parámetros a los valores óptimos, lo cual implica realizar múltiples iteraciones. A continuación se detallan las tareas:

- **Seleccionar las técnicas de modelado:** en este paso se escogen los algoritmos a utilizar, se debe tomar en cuenta la naturaleza del problema. Deben plantearse las herramientas a emplear para la creación de los modelos a construir.

- **Generar un diseño de pruebas:** aquí se determina la estrategia con la que se evaluará el desempeño del modelo: las métricas a considerar, la cantidad y forma en que se realizarán las pruebas.

- **Construir la configuración de los modelos:** se prueban distintas configuraciones de los parámetros de los modelos para determinar cuáles son las que producen mejores resultados. También se pueden realizar descripciones de los modelos, que incluyan detalles sobre dificultades de ejecución, comportamientos encontrados durante los entrenamientos, etc.

- **Interpretación de los modelos:** se analizan e interpretan los resultados, se busca por posibles patrones encontrados.

Fase 5: Evaluación

En esta fase se estudian los resultados para determinar si cumplen con las metas establecidas, se analizan conclusiones o inferencias resultantes del proceso.

Fase 6: Implementación/Despliegue

La puesta en producción del modelo dependiendo de las necesidades y objetivos. Los resultados del desarrollo pueden implicar la implementación o integración para algún fin, o la presentación de resultados y proceso de análisis de datos en una organización. En la figura 1.1 se muestra el ciclo de vida de la metodología planteada.

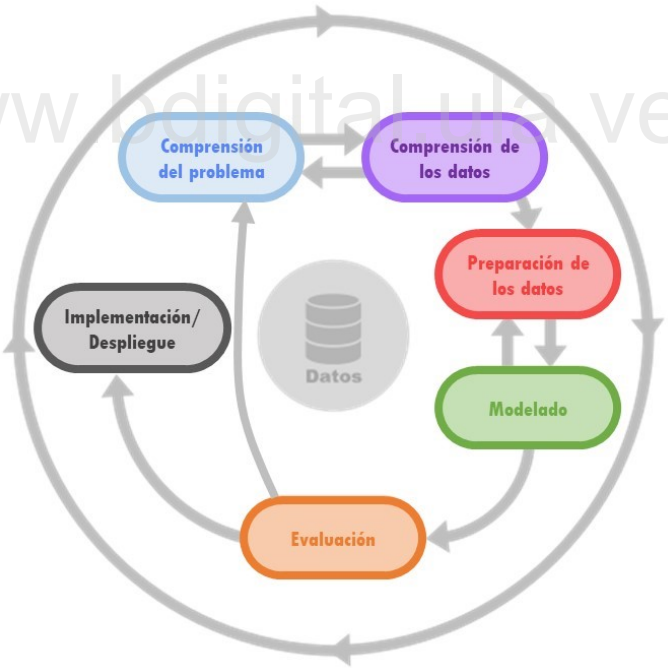

Figura 1.1: Ciclo de vida de la metodología CRISP DM. Modificado de [1]

[1] https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_crispdm_ddita/clementine/crisp_help/crisp_overview.html

1.6. Alcance

En este trabajo se presenta el desarrollo de un sistema de procesamiento de imágenes de rostros, estimando tres características o atributos de estos: el género, la raza y la edad. Esto a través de técnicas de Inteligencia Artificial, y mediante el uso de datos que servirán como recurso base para los entrenamientos de los modelos a diseñar. Para esto se considera necesario llevar a cabo un conjunto de pasos, que incluye todo el procesado de los datos, así como la elección y utilización de diferentes herramientas de IA para el desarrollo de diversos modelos. El prototipo a desarrollar debe ser capaz de detectar uno o varios rostros en una imagen y para cada una de estas estimar las tres características descritas anteriormente.

Capítulo 2

Marco Referencial

2.1. Inteligencia Artificial

Una de las ciencias más recientes, cuyo nombre fue establecido en 1956, es un área de estudio sumamente amplia. Poole y Mackworth (2010) la definen como el campo que estudia la síntesis y análisis de agentes (algo que actúa en un entorno, que hace algo) computacionales que actúan inteligentemente. Para ellos, un agente actúa inteligente cuando:

- Lo que hace es apropiado para sus circunstancias y sus objetivos, teniendo en cuenta las consecuencias a corto y largo plazo de sus acciones.
- Es flexible a entornos y objetivos cambiantes.
- Aprende de la experiencia.
- Toma las decisiones adecuadas dadas sus limitaciones perceptivas y computacionales.

Asimismo, Russell Stuart y Norvig (2009) agrupan los tipos de Inteligencia Artificial en cuatro categorías, según el enfoque dado por distintos autores en años anteriores, siendo estas:

1. **Sistemas que piensan como humanos**

   "La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje..." (Bellman, 1978)

2. **Sistemas que actúan como humanos**

   Aquí entran aquellos sistemas que intentan emular la manera de actuar de los seres humanos, especialmente en tareas que, por ahora, las personas hacen mejor que las máquinas. (Rich y Knight, 1991)

3. **Sistemas que piensan racionalmente**

   Aquellos sistemas que tratan de imitar el pensamiento racional del ser humano, a través del estudio de los cálculos que hacen posible, percibir, razonar y actuar. Winston (1992)

4. **Sistemas que actúan racionalmente**

   Estos buscan emular el comportamiento del ser humano, están relacionados con conductas inteligentes en artefactos. (Nilsson, 1998)

2.2. Visión Artificial

Permite el estudio de imágenes, también llamada visión por computadora (del inglés *computer vision*). La visión artificial aparece en el intento de dotar a las máquinas de un sistema de visión para automatizar el proceso de percepción visual mediante el tratamiento de imágenes digitales. (Martinsanz y García, 2008). Esta disciplina se apoya en campos como la geometría, la estadística, la física y otras disciplinas.

2.2.1. Detección Facial

Es un área de la detección de objetos que consiste en determinar la presencia y ubicación de rostros en una imagen o secuencia de imágenes. Debe diferenciarse la detección del reconocimiento de rostros, el primero es un paso previo del segundo.

"Se podría decir que la detección facial es un proceso que responde a la pregunta ¿Dónde está la persona?, mientras que el reconocimiento facial puede responder a la pregunta de ¿Quién es esa persona?". (Sousa y Mora, 2016, p. 5)

Si bien la detección de rostros es una tarea trivial para la visión humana, es un desafío para la visión artificial debido a las variaciones en la escala, ubicación, orientación, pose, expresión facial, condición de luz y diversas características de apariencia (por ejemplo, presencia de anteojos, vello facial, maquillaje, etc.) (Chauhan, 2014).

Múltiples métodos han sido diseñados para la detección de rostros, probablemente el más utilizado es el propuesto por (Viola y Jones, 2001), sin embargo, otro algoritmo muy utilizado es el basado en el Histograma de Gradientes Orientados (HOG, por sus siglas en inglés), a pesar de que es computacionalmente más lento en la detección de rostros, proporciona mejores resultados para casos donde hay variaciones como presencia de rostros con lentes, o con distintas poses.

2.2.2. Histograma de Gradientes Orientados

La apariencia y la forma de objetos a menudo se pueden caracterizar bastante bien por la distribución de gradientes de intensidad local o direcciones de borde. Este algoritmo se implementa dividiendo la ventana de imagen en pequeñas regiones espaciales también llamadas "celdas", acumulando cada celda a un histograma *1-dimensional* local de direcciones de gradiente u orientaciones de borde sobre los píxeles de la celda. A continuación se detallan los pasos del algoritmo descritos por (Dalal y Triggs, 2005) para construir el descriptor.

Cálculo de gradientes

Para el cálculo de los gradientes, lo más común es aplicar la máscara de derivada discreta 1-Dimensional en una o ambas direcciones horizontal y vertical, los autores determinaron que usar máscaras derivativas más grandes disminuye el rendimiento. Este método requiere filtrar los datos de color o intensidad de la imagen con los siguientes núcleos de filtro:

$$[-1, 0, 1] \text{ y } [-1, 0, 1]^T \tag{2.1}$$

Intervalos de orientación

En este paso cada píxel calcula un voto ponderado para un canal de histograma basado en la orientación en función de los valores encontrados en el cálculo del gradiente, y los votos se acumulan en contenedores de orientación sobre regiones espaciales locales llamados celdas. Las celdas pueden ser rectangulares o radiales. Los contenedores de orientación están espaciados uniformemente sobre $0° - 180°$ (gradiente "sin signo") o $0° - 360°$ (gradiente "con signo"). Empíricamente, se ha demostrado que los mejores resultados se obtienen al utilizar gradientes sin signo. Para el voto, los autores determinaron que la mejor función es la de la magnitud en sí del gradiente en el píxel, en vez de su cuadrado, su raíz cuadrada o una forma recortada de la magnitud, que también fueron consideradas en la práctica.

En la figura 2.1 se observa un ejemplo de una imagen y su división en celdas de 8×8, así como la magnitud (intensidad) y la dirección (sin signo) de los gradientes de cada píxel de una las celdas.

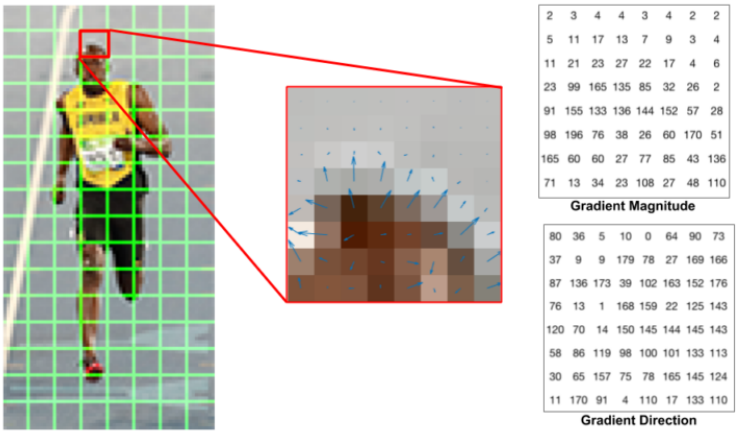

Figura 2.1: Intervalos de Orientación, Recuperado de [1]

[1] https://www.learnopencv.com/histogram-of-oriented-gradients/

Normalización y Bloques de descriptores

Para una mejor invariancia a la iluminación, sombras, etc., también es útil normalizar el contraste de los histogramas locales antes de usarlos. Esto se puede hacer acumulando una medida de "energía" del histograma local en regiones espaciales algo más grandes ("bloques") y utilizando los resultados para normalizar todas las celdas del bloque. El descriptor HOG es entonces el vector concatenado de los componentes de los histogramas de celdas normalizados de todas las regiones de bloques. El proceso general se observa en la figura 2.2.

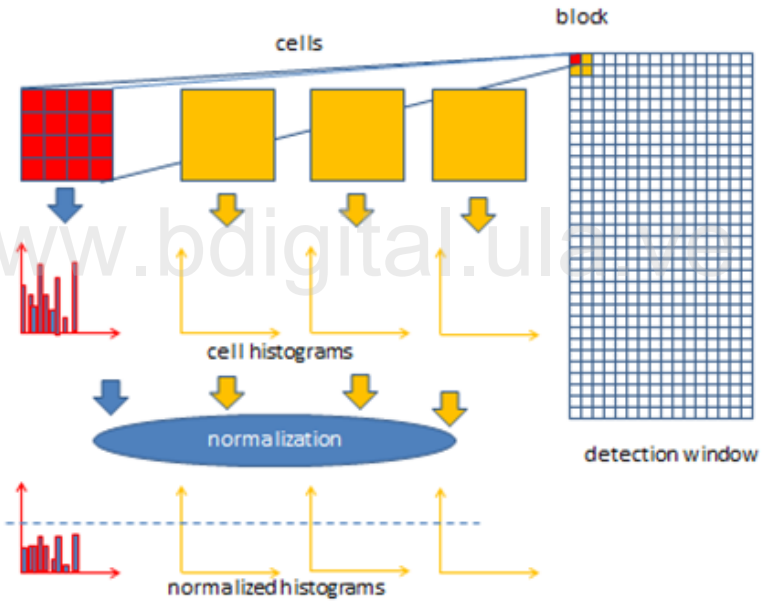

Figura 2.2: Esquema general del algoritmo HOG Recuperado de [2]

[2] https://software.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/computer-vision/feature-detection-functions/histogram-of-oriented-gradients-hog-descriptor.html

Esquemas de normalización de bloques.

Cuatro métodos de normalización fueron propuestos por los autores. Sea v el vector no normalizado que contiene todos los histogramas en un bloque dado, $\|v\|_k$ la k-norma para $k = 1, 2$, y ϵ una constante pequeña. Los esquemas son:

- $$norma\ L2 : v = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \tag{2.2}$$
- L2-hys: Norma L2 limitando los valores de v a 0,2 y renormalizando.
- $$norma\ L1 : v = \frac{v}{\|v\|_1 + \epsilon} \tag{2.3}$$
- $$sqrt\ L1 : v = \sqrt{\frac{v}{\|v\|_1 + \epsilon}} \tag{2.4}$$

2.2.3. Extracción de Características

En los sistemas de análisis de imágenes, la extracción de características[2] es el primer paso crucial que permite el procesamiento simbólico del contenido de la imagen. El objetivo es encontrar una representación específica de los datos que pueda resaltar información relevante. Esta representación se puede encontrar maximizando un criterio o puede ser una representación predefinida. (Sarfraz et al., 2010).

Esto implica reducir la cantidad de recursos necesarios para describir un gran conjunto de datos (por ejemplo, píxeles de imágenes), es decir, reducir la dimensionalidad de estos para tener conjuntos de datos más manejables para su procesamiento, además de evitar problemas de redundancia de información, lo que origina la creación de **descriptores visuales**, un descriptor es un conjunto valores que detallan

[2]En Machine Learning y Reconocimiento de patrones una característica (feature en inglés) es una propiedad individual medible o característica de un fenómeno que se observa. (Bishop, 2006)

las características visuales de los contenidos dispuestos en imágenes, tales como la forma, el color, la textura o el movimiento, entre otros. Sobre la representación de imágenes a través de estos descriptores (Arista-Jalife et al., 2017) destacan que:

> En el caso de clasificación de imágenes, dos imágenes idénticas deben poseer los mismos descriptores, dos imágenes similares deben poseer dos descriptores con poca distancia numérica entre ellos, y, por ende, dos imágenes enteramente diferentes deben poseer una amplia distancia numérica (p. 210).

2.2.4. Enfoques de extracción de características: global y local

Existen dos enfoques para el estudio de características de imágenes, según cómo aborden el problema, estos son:

Características globales

La extracción de características globales (también llamadas holísticas) se basa en procesar la cara completa y luego representarla en forma genérica. Estos métodos extraen rasgos de un rostro sin considerar las áreas del rostro de donde provienen. El objetivo de estos enfoques es reducir la dimensionalidad de los datos de características extraídos de la imagen del rostro, lo que permite utilizar datos más distintivos para representar un rostro. (Pasandi, 2014).

Características locales

Las características locales están relacionadas con bordes, esquinas y otras estructuras dentro de una imagen. Dado que estos enfoques extraen información considerando las partes importantes dentro de la imagen, se asignan mejor a los problemas de reconocimiento facial en comparación con las técnicas de representación global (Pasandi, 2014).

Se basa en describir una parte o región de la imagen a través de alguna transformación que depende del descriptor a utilizar. El resultado final permite representar

el contenido de la imagen subyacente de una manera que debería producir una solución única siempre que se encuentre el mismo contenido, así como también debe tener cierto grado de invariancia con respecto a las variaciones que se encuentran comúnmente, como la translación, la escala y rotaciones (Sarfraz et al., 2010). Este enfoque también es llamado simplemente *Enfoque basado en características*.

2.2.5. Descriptores Locales

A continuación se detallan los descriptores locales utilizados en esta investigación.

Patrones Binarios Locales

Conocido como LBP (por sus siglas en ingles) es un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen mediante el umbral de la vecindad de cada píxel y considera el resultado como un número binario. Debido a su poder discriminativo y simplicidad computacional, el operador de textura LBP se ha convertido en un enfoque popular en varias aplicaciones.

La versión original del descriptor fue propuesta por (Ojala et al., 1996), consiste en extraer un patrón de un bloque 3×3 de una imagen. En este método, el píxel central se utiliza como umbral cuando se compara con sus vecinos. Si el valor de este píxel central es mayor que el valor del vecino, establecemos el valor del vecino como "1", de lo contrario, se establece en "0". Como tenemos 8 vecinos en un bloque 3×3, LBP produce un número binario de 8 dígitos. Este número se convertirá en un número decimal. Luego se divide la imagen en varias regiones y se extrae el histograma de valores de LBP dentro de cada parche. Estos histogramas caracterizan el patrón visual dentro de cada región. Cada uno de estos se normaliza, de tal manera que la suma individual de cada uno de ellos es 1. Como último paso, se concatenan los histogramas de todas las regiones para construir el vector de características para la imagen completa de la cara. (Pasandi, 2014).

En la figura 2.3 se muestra el procedimiento.

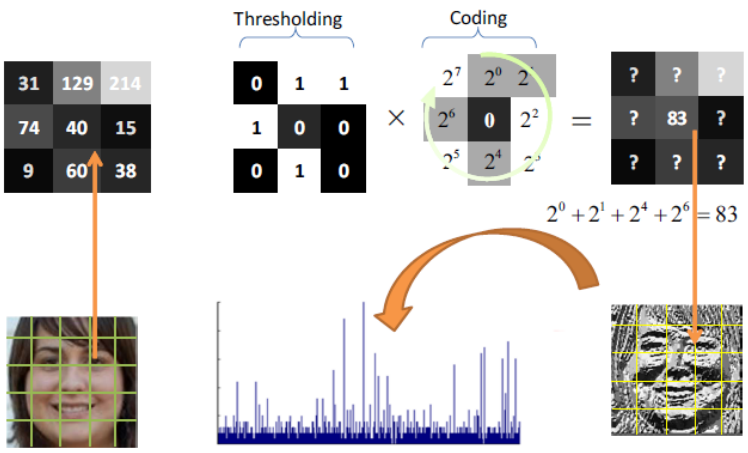

Figura 2.3: Procedimiento para construir el descriptor LBP, por (Pasandi, 2014).

Más tarde se generalizó el método para vecindades de distintos tamaños. Formalmente, se tiene que para cada píxel en (x_c, y_c) el operador LBP viene dado por:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(i_p - i_c)2^p \tag{2.5}$$

donde i_c e i_p son, respectivamente, valores de nivel de gris (intensidad) del píxel central y de los P píxeles circundantes en la vecindad del círculo con radio R.

la función $s(x)$ está definida como:

$$s(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \tag{2.6}$$

Según la definición anterior, el operador básico de LBP es invariante a las transformaciones monótonas de escala de grises que preservan el orden de intensidad de píxeles en los vecindarios locales. El histograma de etiquetas de LBP calculado sobre una región se puede explotar como descriptor de textura.(Huang et al., 2011).

En la figura 2.4 se observan distintos ejemplos del descriptor LBP, el circulo de la izquierda denota un operador con una vecindad de $P = 8$ puntos y un radio $R = 1$,

mientras que en el circulo central y el de la derecha se observa una vecindad de 12 puntos y radio 2, y una vencindad de 16 puntos y radio 4, respectivamente.

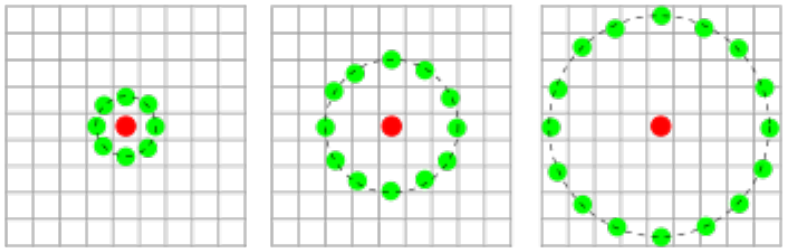

Figura 2.4: Ejemplos de vecindades del LBP, recuperado de [3]

Patrones Uniformes Esta es una extensión del operador original propuesta más adelante por (Ojala et al., 2002) que se puede utilizar para reducir la longitud del vector de características e implementar un descriptor simple invariante de rotación. Un LBP se denomina uniforme si contiene como máximo dos transiciones bit a bit de 0 a 1 o viceversa cuando el patrón de bit se recorre circularmente. Por ejemplo, los patrones 00000000 (0 transiciones), 01110000 (2 transiciones) y 11001111 (2 transiciones) son uniformes, mientras que los patrones 11001001 (4 transiciones) y 01010010 (6 transiciones) no lo son. Al computar el histograma, se utilizan patrones uniformes de modo que haya una etiqueta separada para cada patrón uniforme y todos los patrones no uniformes se etiqueten con una sola etiqueta. (Pietikäinen, 2010). Usando esta técnica se puede reducir la longitud de un vector desde 256 a 59 espacios. A estos se les llama Patrones Binarios Locales Uniformes (uLBP, por sus siglas en inglés), y se define de la siguiente manera

$$BP^{U2}_{P,R}(x_c, y_c) = \begin{cases} \sum_{p=0}^{P-1} s(i_p - i_c)2^p & \text{if } U(LBP_{N,R}) \leq 2 \\ N(N-1)+3 & \text{otro caso} \end{cases} \tag{2.7}$$

donde el operador U es una medida de uniformidad definida por (Ojala et al., 2002) que calcula el número de transiciones bit a bit de 0 a 1, o de 1 a 0.

[3] https://en.wikipedia.org/wiki/File:Lbp_neighbors.svg

Histograma de Gradientes Orientados

Este descriptor ya fue explicado en la sección 2.2.2. No sólo se usa para la detección de objetos, sino también como descriptor de características para la clasificación de imágenes, y también es empleado para tal fin en este trabajo.

2.3. Aprendizaje Automático

El aprendizaje automático (ML, por sus siglas en inglés) es un área de la inteligencia artificial que permite implementar sistemas que "aprenden" automáticamente. Tal como lo plantea (Mueller y Massaron, 2016), el aprendizaje automático utiliza una variedad de algoritmos que aprenden iterativamente de los datos para mejorar, describir datos y predecir resultados. A medida que los algoritmos ingieren datos de entrenamiento, es posible producir modelos más precisos basados en esos datos. Un modelo de aprendizaje automático es el resultado generado cuando entrena su algoritmo con datos. Después del entrenamiento, cuando proporcione un modelo con una entrada, se le dará una salida.

Hay tres tipos principales de Aprendizaje Automático:

1. **Aprendizaje Supervisado:** El aprendizaje supervisado está destinado a encontrar patrones en los datos que se puedan aplicar a un proceso de análisis. Estos datos tienen características etiquetadas que definen el significado de los datos. Cuando las etiquetas son continuas, se trata de un modelo de **regresión**, por otro lado si las etiquetas son discretas, se habla entonces de un modelo de **clasificación**.

2. **Aprendizaje no Supervisado:** En este caso no se dispone de etiquetas, se hace uso de algoritmos que encuentren patrones en los datos y que permitan agrupar/asociar estos datos para entender el significado detras de estos.

3. **Aprendizaje por refuerzo:** Esta técnica es descrita por (Mueller y Massaron, 2016) como un modelo de aprendizaje conductual. El algoritmo recibe información del análisis de los datos para guiar al usuario hacia el mejor resultado. El sistema aprende mediante prueba y error. Por lo tanto, una secuencia de

decisiones acertadas dará como resultado que el proceso se "refuerce" porque resuelve mejor el problema en cuestión.

2.3.1. Selección de Características

Al implementar un modelo de ML, es común que las variables o características tengan una alta dimensionalidad, y es posible que muchas de estas no aporten mucho a la tarea de predecir la variable de salida (etiqueta) o que sean redundantes. La selección de características permite seleccionar un subconjunto de las variables a fin de obtener aquellas más relevantes o útiles y mejorar el modelo de Aprendizaje Automático. Esto trae múltiples beneficios potenciales, tal como lo exponen (Guyon y Elisseeff, 2003), facilita la visualización y comprensión de datos, reduce los requisitos de medición y almacenamiento, reducir los tiempos de entrenamiento, y se encarga de la alta dimensionalidad de los datos para mejorar el rendimiento de la predicción. Además, ellos agrupan los métodos de selección de características en tres categorías.

1. **Métodos de envoltura:** Son considerados los mejores en términos de precisión, pero al costo de requerir mayor cantidad de recursos computacionales. Consisten en tomar distintos subconjuntos de los datos y entrenarlos con algún modelo de Aprendizaje Automático, para posteriormente comparar el rendimiento logrado con cada subconjunto. En la siguiente figura se observa este esquema.

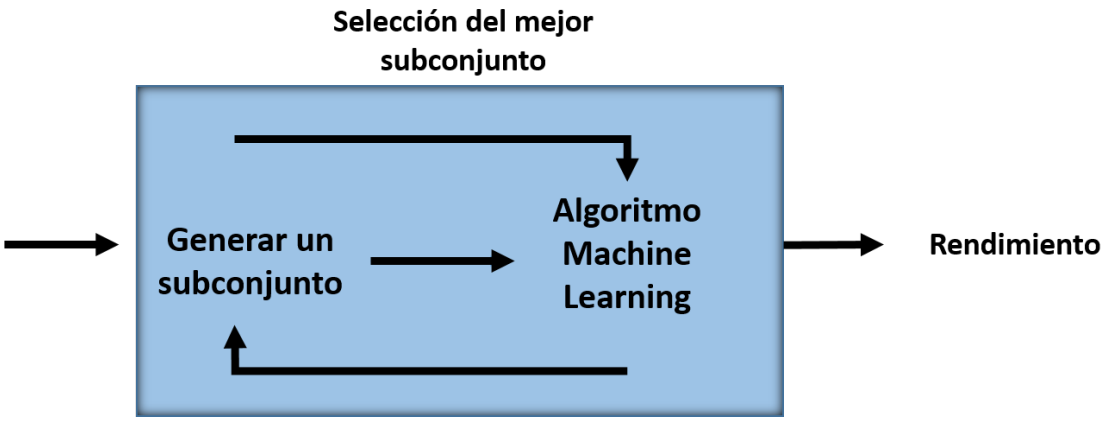

Figura 2.5: Métodos de envoltura

2. **Métodos de filtro:** Esté método es independiente del algoritmo de Aprendizaje automático a utilizar, aquí se utilizan medidas estadísticas para establecer puntajes sobre los subconjuntos de las variables y según estos puntajes clasificar las variables y determinar la correlación de estas con la etiqueta. La siguiente figura muestra lo planteado.

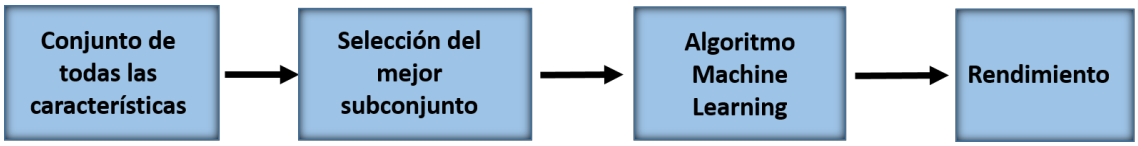

Figura 2.6: Métodos de filtro

3. **Métodos integrados:** Los métodos integrados incorporan la selección de variables como parte del proceso de entrenamiento. Pueden ser más eficientes en varios aspectos: hacen un mejor uso de los datos disponibles al no necesitar dividir los datos de formación en un conjunto de formación y validación; llegan a una solución más rápido al evitar volver a entrenar un predictor desde cero para cada subconjunto de variables investigado. (Guyon y Elisseeff, 2003). La figura 2.7 muestra en general el proceso descrito.

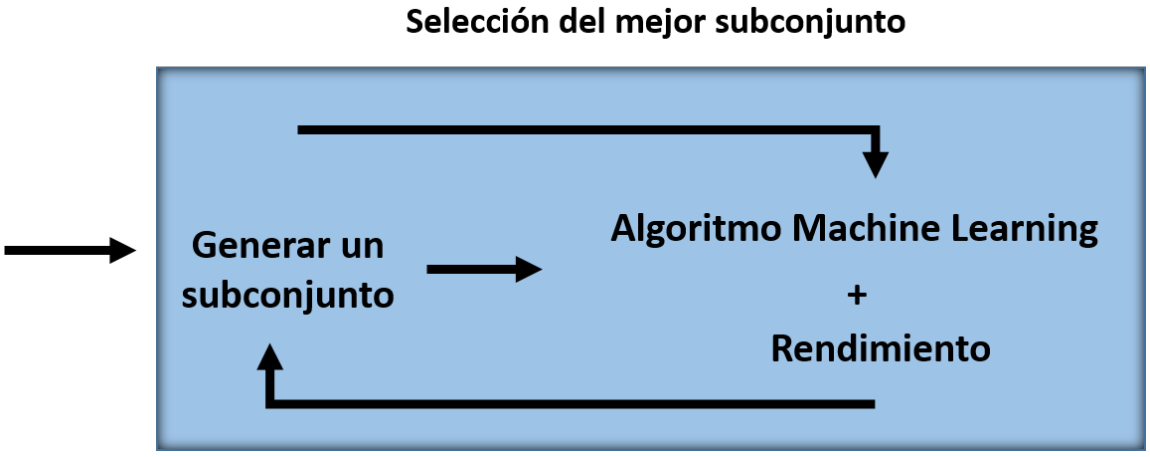

Figura 2.7: Métodos integrados

2.3.2. Máquinas de Vectores de Soporte

Es uno de los algoritmos de ML más usados, las Máquinas de Vectores de Soporte o *Support Vector Machines* (SVM) son la creación del matemático Vladimir Vapnik y de algunos de los sus colegas que trabajaban en los laboratorios de AT&T en la década de 1990 (como Boser, Guyon y Cortes).

Tal como lo describe (Mueller y Massaron, 2016), la estrategia de un SVM es bastante simple: dado un conjunto de datos (pueden verse como puntos sobre un plano), se traza una línea que construya un hiperplano tal que separe los puntos pertenecientes a clases distintas. La línea de separación deseada es la que tiene el margen más grande (el espacio vacío entre los límites de las clases). Este algoritmo coloca la línea de separación en el medio del margen, conocido como margen máximo o hiperplano de margen óptimo, y los ejemplos cerca de la línea (que forman parte de los límites) son vectores de apoyo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

El principal hiperparámetro de este algoritmo es el valor de regularización C, este controla la compensación entre la penalización de la variable de holgura (clasificaciones erróneas) y el ancho del margen. Mientras mayor sea C, máyor será la regularización (mayor penalización o restricción ante errores de clasificación). Un ejemplo de un hiperplano de un SVM se observa en la figura 2.8.

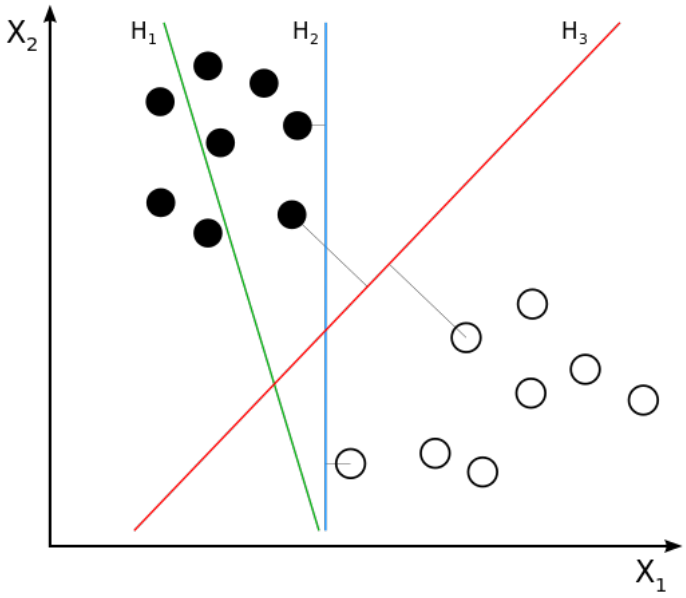

Figura 2.8: Hiperplano de un SVM. Recuperado de [5]. H1 no separa las clases. H2 las separa, pero solo con un margen pequeño. H3 las separa con el margen máximo (el obtenido por el SVM)

2.4. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN por sus siglas en inglés) son un tipo de red neuronal ampliamente utilizado en el campo de la IA debido a sus increíbles resultados en los últimos años. Son análogas a las redes neuronales tradicionales en el sentido de que están compuestas por neuronas que se optimizan a sí mismas mediante el aprendizaje. Cada neurona seguirá recibiendo una entrada y realizará una operación (como un producto escalar seguido de una función no lineal).
La única diferencia notable entre las CNN y las redes neuronales tradicionales, es que las primeras se utilizan principalmente en el campo del reconocimiento de patrones dentro de las imágenes. Estas permiten codificar características específicas de las imágenes en la arquitectura, lo que hace que la red sea más adecuada para tareas centradas en imágenes, mientras que también reduce aún más los parámetros

[5] https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg

necesarios para configurar el modelo. Las CNN están compuestas de tres tipos de capas: convolución, agrupación o *pooling*, y capas densas o *fully-connected*.

2.4.1. Capas de convolución

La convolución es una operación matemática, en la que una función se aplica de alguna manera a otra función. El resultado se puede entender como una «mezcla» de las dos funciones. Estas capas requieren algunos componentes, que son datos de entrada, un filtro y un mapa de características. Si se tiene como entrada una imagen en color, que se compone de una matriz de píxeles en 3D. Esto significa que la entrada tendrá tres dimensiones (altura, ancho y profundidad) que corresponden a RGB en una imagen. Por otro lado, se tiene el "detector de características" llamado *kernel* o filtro, que se moverá a través de los campos receptivos de la imagen, verificando si la característica está presente. Este proceso se conoce como convolución.

El *kernel* es una matriz bidimensional (2-D) de pesos, que representa una parte de la imagen. Lo más común es que estos sean una matriz 3x3. Este filtro se aplica a un área de la imagen y se calcula un producto escalar entre los píxeles de entrada y el filtro. Este producto escalar se alimenta luego a una matriz de salida. Posteriormente, el filtro se desplaza con cierto paso a lo largo de la imagen, repitiendo el proceso hasta que el filtro ha barrido toda la imagen. La salida final de la serie de productos escalares de la entrada y el filtro se conoce como mapa de características, mapa de activación o característica convolucionada. En la figura 2.9 se ilustra este procedimiento.

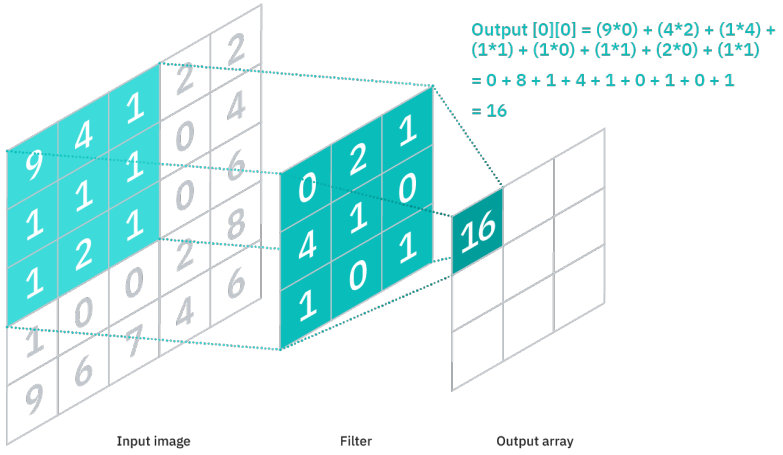

Figura 2.9: Operación de convolución de una CNN [6]

El valor de los pesos permanece constante durante todo el barrido de la imagen, sin embargo, este, al igual que otros parámetros, se ajustan durante el entrenamiento a través del proceso de retropropagación y descenso de gradiente. Ahora bien, hay tres hiperparámetros que afectan el tamaño del volumen de la salida que deben configurarse antes de que comience el entrenamiento de la red neuronal, estos son:

1. El número de filtros o *kernels*: Afecta la profundidad de la salida. Por ejemplo, tres filtros distintos producirían tres mapas de características diferentes, creando una profundidad de tres.

2. El paso o *stride*: Es la distancia, o número de píxeles, que el *kernel* se mueve sobre la matriz de entrada. Si bien los valores de paso de dos o más son raros, una paso más grande produce una salida menor.

3. *Zero padding*: Se usa generalmente cuando los filtros no se ajustan a la imagen de entrada. Esto establece todos los elementos que quedan fuera de la matriz de entrada en cero, produciendo una salida mayor o de igual tamaño. Hay tres tipos de *padding*:

    - *Valid padding*: esto también se conoce como sin relleno. En este caso, la última convolución se descarta si las dimensiones no se alinean.

[6] https://www.ibm.com/cloud/learn/convolutional-neural-networks

- *Same padding*: este relleno asegura que la capa de salida tenga el mismo tamaño que la capa de entrada
- *Full padding*: aquí se aumenta el tamaño de la salida agregando ceros al borde de la entrada.

Después de cada operación de convolución, una CNN aplica una transformación de unidad lineal rectificada (ReLU) al mapa de características, lo que introduce la no linealidad en el modelo. Esta viene dada como sigue:

$$f(x) = max(0, x) \tag{2.8}$$

Donde x es la entrada de la neurona.

2.4.2. Capas de agrupación o *pooling*

También conocida como submuestreo, lleva a cabo una reducción de dimensionalidad, lo que reduce el número de parámetros en la entrada. Similar a la capa convolucional, la operación de agrupación barre un filtro a través de toda la entrada, pero la diferencia es que este filtro no tiene ningún peso. En cambio, el kernel aplica una función de agregación a los valores seleccionados en cada paso, llenando la matriz de salida. Hay dos tipos principales de agrupación:

1. *Max pooling*: Se selecciona el píxel con el valor máximo y es enviado a la matriz de salida para cada barrido que hace el filtro sobre la imagen.

2. *Average pooling*: En este caso se calcula el valor promedio de los píxeles para cada barrido sobre la imagen.

Estas capas ayudan a reducir la complejidad, mejorar la eficiencia y limitar el riesgo de sobreajuste (*overfitting*). En la práctica, se utilizan numerosas capas de convolución y agrupamiento, cada vez se van extrayendo características o patrones más y más complejos.

2.4.3. Capas densas o *fully connected*

Tal como su nombre lo indica, en estas capas todas las neuronas están conectadas con la capa anterior. Estas capas son las encargadas de realizar la clasificación o estimación de la salida, en base a las características extraídas a través de las capas anteriores y sus diferentes filtros. Utilizan diferentes funciones de activación dependiendo de la naturaleza de la salida. Para tareas de clasificación binaria, se hace uso de la función sigmoide, aunque también puede usarse la función softmax con 2 salidas, ya que de hecho, esta es una generalización de la función sigmoide. Para casos multiclase, se usa la función softmax. Estas funciones permiten determinar la probabilidad de que una muestra pertenezca a una u otra clase, y este es el criterio para determinar la predicción realizada por la CNN. La función sigmoide viene dada por la ecuación 2.9 y su gráfica se observa en la figura 2.10

$$P(t) = \frac{1}{1 + e^{-t}} \tag{2.9}$$

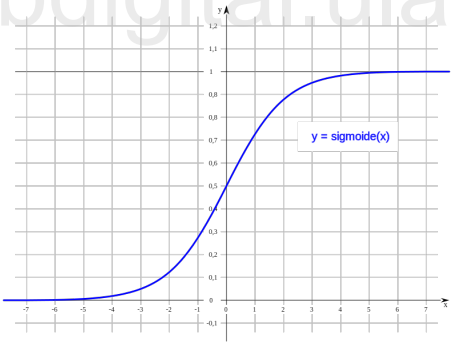

Figura 2.10: Función sigmoide

Como se ha visto, el ciclo de vida de una red neuronal comienza por aplicar capas consecutivas de convoluciones y agrupamiento, para adquitir patrones en las imágenes que serán usadas como vector de características por las capas densas para llevar a cabo la tarea de clasificación y generar la salida correspondiente a través de las funciones de activación. La figura 2.11 muestra el ejemplo de un esquema global de una CNN.

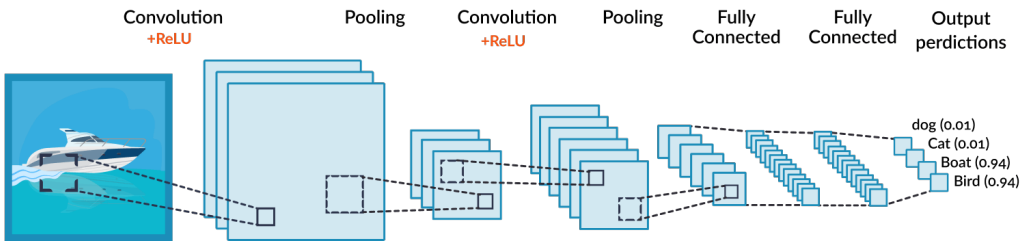

Figura 2.11: Arquitectura de una CNN. [7]

[7] https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/

Capítulo 3

Comprensión y Preparación de los datos

Como ya se mencionó anteriormente, la fase 1 de la metodología CRISP DM consiste en un análisis desde una perspectiva de negocio, este trabajo tiene un propósito académico que no tomó en cuenta objetivos de negocio, en su lugar, se considera el problema desde una perspectiva general, en la cual se desean determinar características en imágenes de rostros a través de la construcción de modelos de clasificación basados en técnicas de IA, donde todas las características se consideran igualmente importantes para el estudio, y donde no se tienen consideraciones o exigencias externas.

En este capítulo se llevan a cabo las fases 2 y 3 de la metodología planteada, que son el estudio o comprensión de los datos y la preparación de estos. Primero se realiza un análisis completo de las características y peculiaridades del dataset utilizado, así como el enfoque propuesto para abordar el estudio como resultado de este análisis. Posteriormente se describe el procesamiento previo pertinente para preparar los datos para la fase siguiente.

3.1. Comprensión de los datos

3.1.1. Recopilación y descripción

Dataset

Los datos empleados en esta investigación pertenecen al dataset UTKFace [1], este consiste en 23708 imágenes de rostros (un rostro por imágen) con anotaciones de género, raza y edad. Las imágenes tienen gran variedad en pose, expresión facial, iluminación, etc. Fue utilizado el conjunto de datos cuyos rostros están previamente recortados y alineados. Todas las imágenes son de 200x200 píxeles. Cada archivo de imágen tiene en su nombre las anotaciones o etiquetas y estas vienen codificadas en números enteros, para el género un 0 o 1, indicando masculino o femenino, respectivamente. La raza es un número del 0 al 4, donde 0 indica caucásico, 1 africano/afroamericano, 2 asiático, 3 indio (de la India), y 4 es denotado como otros, agrupando aquí a latinos, personas de Medio Oriente (Turquía, Arabia Saudita, Egipto, etc). La edad es un entero en un rango de 0 a 116 años.

3.1.2. Exploración

La visualización de los datos ayuda a comprender y abordar el enfoque de estudio de una mejor manera. En la figura 3.1 se muestran algunas imágenes del dataset.

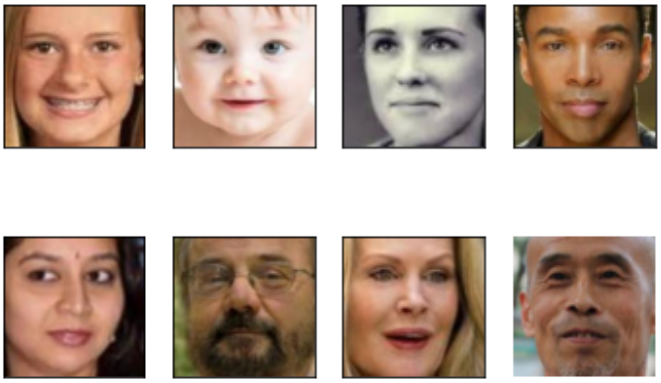

Figura 3.1: Imágenes del UTKFace dataset.

[1] https://susanqq.github.io/UTKFace/

Los porcentajes de cada etiqueta y el número de muestras respectivo presente en los datos se muestran en las figuras 3.2, 3.3, y 3.4.

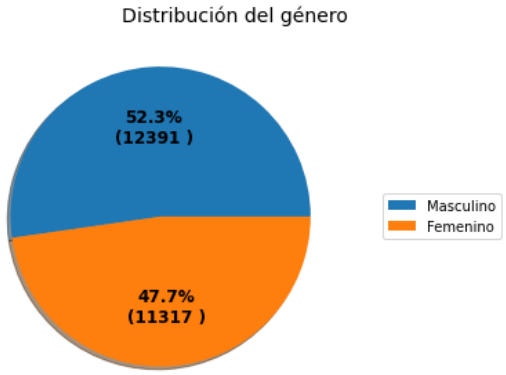

Figura 3.2: Distribución del género

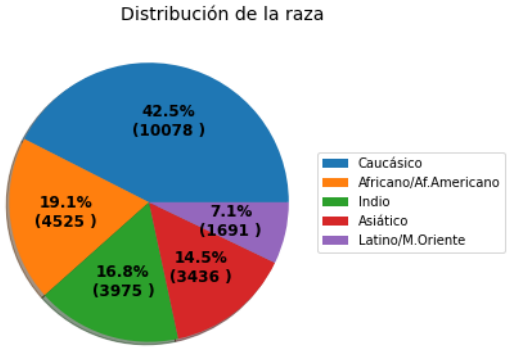

Figura 3.3: Distribución de la raza

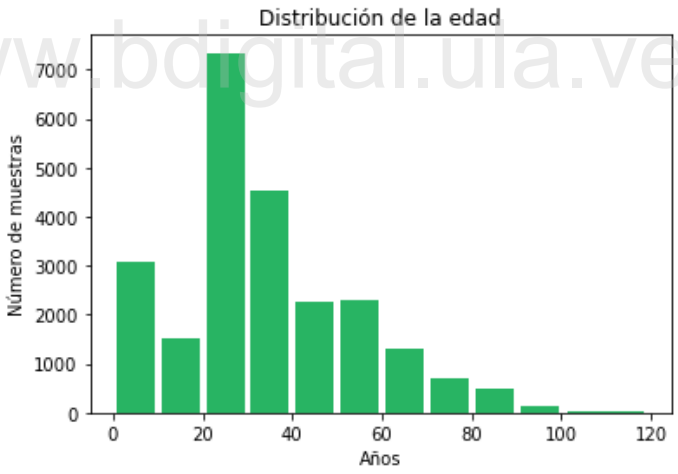

Figura 3.4: Distribución de la edad

Tal como se observa, los datos están balanceados para el género. Por otro lado en la raza hay una predominancia grande de los caucásicos, ocupando un 42.5 % de los datos, mientras que la clase de latinos/Medio Oriente ocupa apenas el 7.1 %. La distribución de edad muestra que en los datos hay mayormente personas entre 20 y 40 años, y pocas muestras para personas mayores de 70 años.

El desbalanceo de clases o mayoría de muestras de un rango determinado de valores de una variable sobre el resto en un conjunto de datos puede ser problemático ya que puede conducir a que los algoritmos enfoquen su desempeño únicamente en aquellas clases para las que se dispone de más datos, y por lo tanto pudieran no aprender a reconocer bien las clases o muestras minoritarias. Esto es algo que debe tenerse simpre presente a la hora de desarrollar y evaluar modelos predictivos.

3.1.3. Verificación de calidad

Un análisis manual de las imágenes fue llevado a cabo para determinar algunos errores o inconsistencias como existencia de imágenes que no contuvieran rostros o etiquetas evidentemente erróneas. Se encontraron 5 imágenes que no contenían rostros, sino ojos o nada en sí, estas se muestran a continuación:

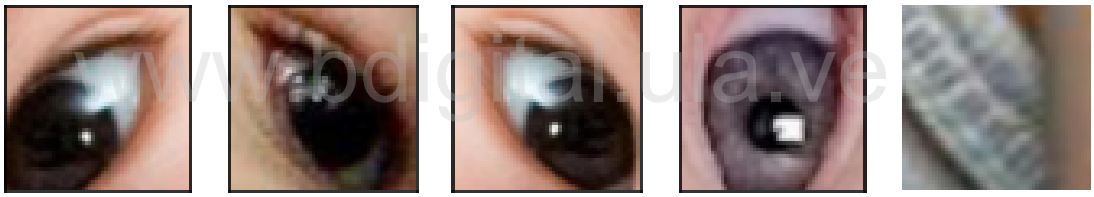

Figura 3.5: Imágenes erróneas del UTKFace

También fueron encontradas 12 imágenes que contenían rostros de adultos con etiquetas de edad de niño.

3.2. Preparación de los datos

3.2.1. Selección y limpieza

Todas las etiquetas fueron utilizadas para el estudio. Fueron solucionados los problemas encontrados en la tarea de verificación anteriormente descrita, eliminando las imágenes con las inconsistencias y errores especificados, en total se eliminaron 17 imágenes, quedando entonces un dataset con 23691 elementos.

3.2.2. Construcción de los datos

En esta tarea se llevaron a cabo procedimientos para generar nuevas estructuras de representación de los datos como preámbulo a la fase de modelado.

Modificación de las etiquetas

Como ya se mencionó anteriormente, los datos presentan cierto desbalanceo en las etiquetas de raza y edad. Esto puede abordarse de múltiples maneras. Algo que comúnmente se hace es la modificación de pesos (o "importancia") de las clases durante la fase de entrenamiento de los modelos, y consiste en darle mayor relevancia a las predicciones de la clase minoritaria, comúnmente.

Otra estrategia es unir una o más clases para formar una cuyo tamaño sea más representativo en relación con el de las demás. Esto se hizo para abordar el desbalanceo en el caso del atributo de la raza. Se unieron los grupos 3 y 4, pertenecientes a personas de la India, Latinoamericanos y del Medio Oriente, el grupo 2 (asiáticos) tiene menos muestras que el grupo 4 (indios), pero pueden haber más similitudes entre el grupo latinos y personas del Medio Oriente con personas de la India, que con personas asiáticas, y esto pudiera dificultar el desempeño de los modelos. En consecuencia, la distribución resultante se observa en la figura 3.6.

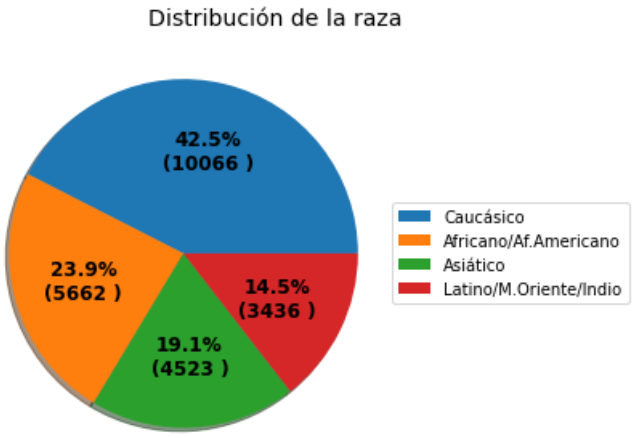

Figura 3.6: Distribución modificada de la raza

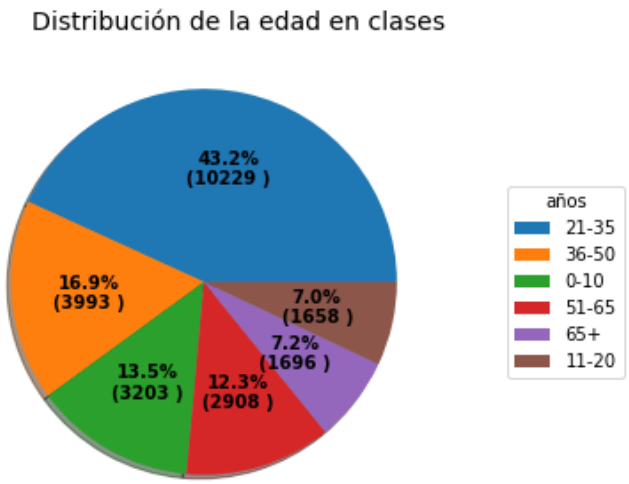

Figura 3.7: Distribución de la edad en clases

Para el caso de la edad, hay dos formas de enfocar su estimación: puede verse como un problema de regresión (estimar un valor numérico continuo) o de clasificación (estimar la clase a la que pertenece según un rango de edades). En este trabajo se trata como clasificación, las edades son agrupadas en clases siguiendo el siguiente esquema:

Clasificación de las edades	
0 - 10 años	Clase 0
11 - 20 años	Clase 1
21 - 35 años	Clase 2
36 - 50 años	Clase 3
51 - 65 años	Clase 4
66+ años	Clase 5

Cuadro 3.1: Esquema de clasificación de las edades

La distribución de la edad de las clases construídas queda como se ilustra en la figura 3.7

El criterio de distribución de las clases se hizo considerando agrupar las imáge-

nes en rangos de edades en los que las características faciales fueran similares, sin embargo sus efectos serán estudiados más adelante.

Creación de las características

En el campo de *Machine Learning*, cuando se trata de imágenes, es común representar los datos de alguna manera codificada que sea más eficiente (en tamaño y contenido). Utilizar imágenes en bruto puede resultar muy costoso computacionalmente, por ejemplo, una simple imágen 96x96 pixeles contenida en un vector 1-dimensional resultará en un arreglo de 9216 valores, y eso si sólo tiene 1 canal; es decir, una imágen en blanco negro. Como ya se habló anteriormente, existen descriptores que permiten codificar información de una imagen para extraer información útil y en una presentación más compacta. Para eso se hizo uso del Histograma de Gradientes Orientados (HOG) y el Histograma de Patrones Locales Binarios (LBP).

Múltiples estrategias se han llevado a cabo para crear características de imágenes a partir de estos descriptores, desde extraerlos a imágenes en distintas escalas hasta la variación parámetros de los mismos, así como la fusión de varios descriptores para mejorar resultados.

En primer lugar las imágenes son convertidas a escalas de grises, y se crean 3 conjuntos de imágenes de resoluciones 32x32, 64x64 y 96x96 píxeles. A continuación se describe diseño de los métodos para la creación de características.

Vector de características HOG: Para este descriptor se aplica normalización global, se varían los tamaños de celda en 8x8 y 16x16 píxeles, para cada uno se extrae el descriptor en las 3 escalas planteadas. La figura 3.8 muestra el resultado de la aplicación del descriptor para ambos valores de celda.

Los dos vectores de características serán la fusión de las 3 resoluciones para la celda 8x8 y para la celda 16x16.
Para un tamaño de celda de 8x8 pixeles, se define el vector de características con la siguiente ecuación:

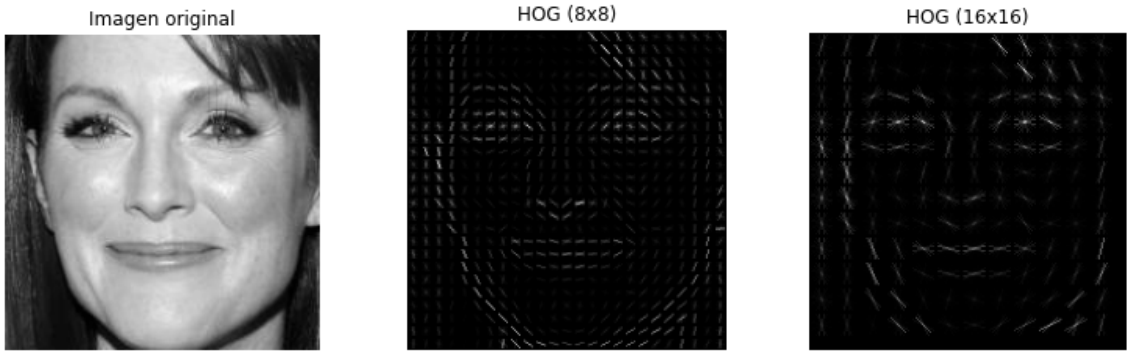

Figura 3.8: Descriptor HOG. Tamaño de celda 8x8 y 16x16.

$$HOG_A = HOG^{(32x32)}_{(8,8)} + HOG^{(64x64)}_{(8,8)} + HOG^{(96x96)}_{(8,8)} \tag{3.1}$$

Análogamente, el vector de celdas de tamaño 16x16 píxeles se construye como sigue:

$$HOG_B = HOG^{(32x32)}_{(16,16)} + HOG^{(64x64)}_{(16,16)} + HOG^{(96x96)}_{(16,16)} \tag{3.2}$$

Vector de características LBP: Para este descriptor se recomienda primero dividir la imagen en celdas, y a cada una de estas sub-imágenes obtenidas, se le extrae el descriptor para finalmente unir todos los vectores en uno solo. Se escogió una división de 4x4 celdas. En (Tapia y Perez, 2013) se extrae para 8 tamaños de radios, de 1 a 8, basado en esa estrategia, en esta investigación se hará sólo para los valores de $R = 1, 2, 4, 6$ y 8. Por lo tanto se obtienen 5 vectores LBP para una resolución en particular. El número de vecinos p se mantiene fijo en 8. Para ilustrar el efecto de aplicar el descriptor LBP a una imágen, se muestra en la figura, el descriptor LBP para los 5 radios considerados, cabe destacar que para la construcción del descriptor como se dijo anteriormente, primero se dividió la imágen en 4x4 celdas.

Los vectores de características para las resoluciones de 32x32, 64x64 y 96x96 píxeles son denotados como LBP_A, LBP_B, y LBP_C, respectivamente, y vienen dados por las siguientes ecuaciones:

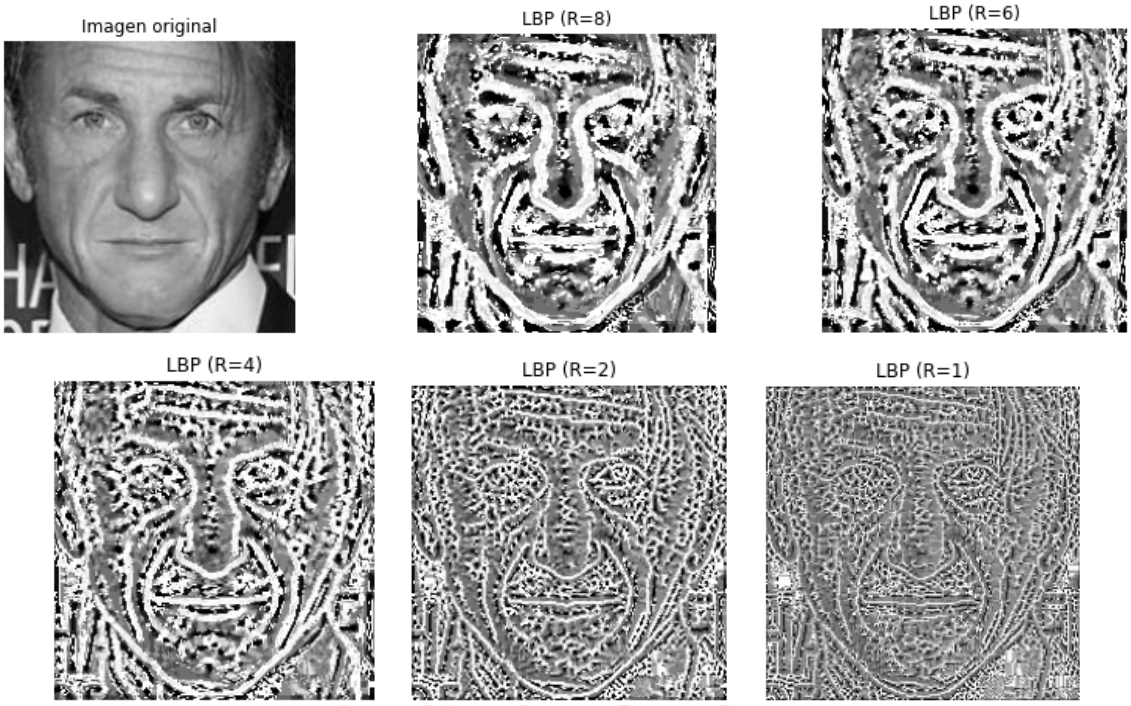

Figura 3.9: Descriptor LBP. Tamaños de R=1,2,4,6,8.

$$LBP_A = LBP_{8,1}^{(32x32)} + LBP_{8,2}^{(32x32)} + LBP_{8,4}^{(32x32)} + LBP_{8,6}^{(32x32)} + LBP_{8,8}^{(32x32)} \quad (3.3)$$

$$LBP_B = LBP_{8,1}^{(64x64)} + LBP_{8,2}^{(64x64)} + LBP_{8,4}^{(64x64)} + LBP_{8,6}^{(64x64)} + LBP_{8,8}^{(64x64)} \quad (3.4)$$

$$LBP_C = LBP_{8,1}^{(96x96)} + LBP_{8,2}^{(96x96)} + LBP_{8,4}^{(96x96)} + LBP_{8,6}^{(96x96)} + LBP_{8,8}^{(96x96)} \quad (3.5)$$

En la siguiente tabla se muestran las longitudes de todos los vectores resultantes.

Vector descriptor	Tamaño
HOG_A	560
HOG_B	2240
LBP_A	4276
LBP_B	4704
LBP_C	4720

Cuadro 3.2: Vectores de características y sus tamaños

Capítulo 4

Modelado y Pruebas

4.1. Técnicas de modelado

En esta sección de describen los enfoques planteados para el desarrollo del sistema de procesamiento de imágenes.

4.1.1. Machine Learning: Máquina de Vectores de Soporte (SVM) con *kernel*

Para la construcción de este modelo se consideró el siguiente *pipeline* conformado por tres pasos, primero se estandarizaron los datos de entrada en el rango $[0, 1]$, esto suele ayudar a una convergencia más rápida de los algoritmos. Luego se aplicó un método de filtrado como selección de características basado en el ANOVA f-test o análisis de varianza, este test permite determinar la variación de las características respecto a las etiquetas. Se asume que características con mayor varianza son las que tendrán mayor poder discriminativo a la hora de clasificar una u otra clase. Luego se seleccionaron las K características que obtuvieron un mayor puntaje, y serán las K características utilizadas para entrenar el estimador o algoritmo. Es importante recordar además que, esto no sólo permite seleccionar aquellas características más **relevantes** para la clasificación, sino que también reduce la dimensionalidad de los datos, al disminuir el tamaño del vector de características, por lo que permite tiempos más cortos de entrenamiento y reducción de datos almacenados en memoria.

Kernel

Como ya se dijo anteriormente, el estimador o algoritmo utilizado fue el SVM, como recurso adicional, se hizo uso de un *kernel*. La función del *kernel* es tomar datos como entrada y transformarlos en la forma requerida, su uso es conveniente cuando los datos son linealmente no separables o tienen una gran dimensión. Para la implementación del modelo de ML, se utilizó un SVM con *kernel* de Funcion de Base Radial (RBF, por sus siglas en inglés), y viene dado por la siguiente expresión:

$$k(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2) \tag{4.1}$$

donde $\gamma > 0$.

El hiperparámetro γ define hasta dónde llega la influencia de un solo ejemplo de entrenamiento, con valores bajos significa "lejos" y valores altos significa "cerca". Un SVM con *kernel* RBF fue construido para cada una de las tres etiquetas. La figura 4.1 muestra el esquema del modelo general para las 3 etiquetas en estudio.

Para la construcción y entrenamiento de los modelos, se utilizó la librería de Python para *Machine Learning*, **Scikit Learn** [1].

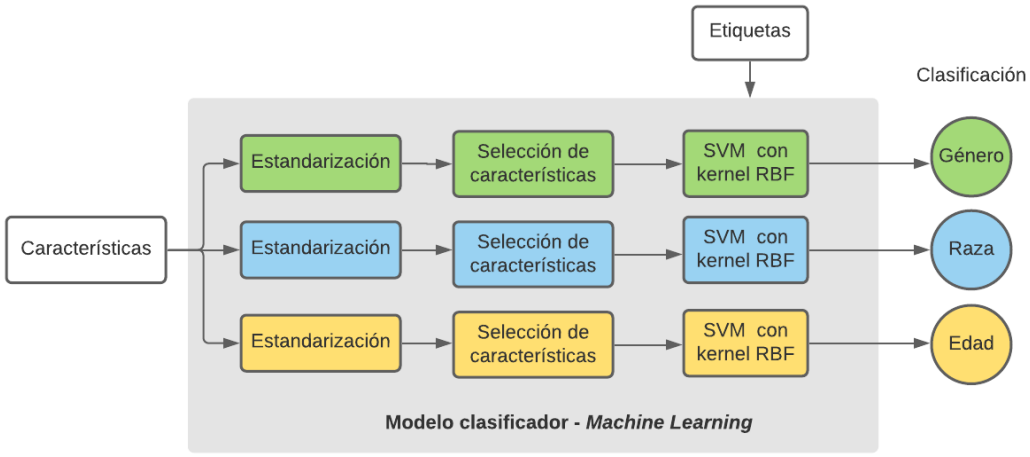

Figura 4.1: Modelo clasificador - *Machine Learning*

[1] https://scikit-learn.org/stable/

4.1.2. *Deep Learning*: Red Neuronal Convolucional (CNN)

El entrenamiento de redes neuronales puede requerir mucho tiempo, y suelen necesitar de grandes cantidades de datos para generar buenos resultados. En esta investigación se hace uso del *Transfer Learning* o aprendizaje por transferencia, como lo expone (West et al., 2007), este enfoque se centra en almacenar el conocimiento adquirido mientras se resuelve un problema y se lo aplica a un problema diferente pero relacionado; en otras palabras, aplicar *Transfer Learning* en el ámbito de las redes neuronales consiste en tomar una red pre-entrenada (comúnmente con una cantidad masiva de datos) para una tarea específica, y usar su configuración para un problema que guarde alguna relación con el que fue originalmente entrenado.

Multi-task learning

Si bien se puede construir un modelo (red neuronal) para cada etiqueta, trabajos como el de (Ito et al., 2018) han demostrado que se obtienen buenos resultados al construir una única arquitectura que tenga como salida varias etiquetas, además que se logra reducir la complejidad de la red haciendo uso compartido de las capas de convolución, y por lo tanto, de las características extraídas. Este esquema es conocido como *multi-task learning* o multi-tarea, ya que "aprenden" múltiples tareas simultáneamente.

Para la creación de la CNN, se hizo uso de la API de Python, **Keras**[2], con **Tensorflow**[3] como *backend*, estas herramientas además proporcionan varias arquitecturas de CNN pre-entrenadas con el conjunto de datos Imagenet[4], el cual consiste en 1281167 imágenes pertenecientes a 1000 clases, las cuales incluyen objetos, animales, etc. Una red entrenada con estos datos, puede servir de base para el caso específico estudiado en esta investigación.

Se utilizó la arquitectura de la red *Xception* pre-entranada sobre el conjunto de datos Imagenet, esta red propuesta por (Chollet, 2017) consiste en 36 capas convolucionales que forman la base de extracción de características y una capa final de clasificación. Esta última capa es omitida y en su lugar se crean varias capas densas

[2] https://keras.io/
[3] https://www.tensorflow.org/
[4] http://image-net.org/

que constituyen el clasificador para cada etiqueta, formando entonces la estructura de la red *multi-task*, al final de cada clasificador se hace uso de una función de activación *softmax*, asignando como salida un número de neuronas igual al número de clases; dos para el género, cuatro para la raza y 6 para la edad. La función de coste empleada fue la entropía cruzada, la suma de las pérdidas de cada etiqueta constituye la función de coste de toda la red. Se emplearon imágenes en una resolución de $96x96$ píxeles.

El modelo planteado para la red neuronal se observa en la figura 4.2.

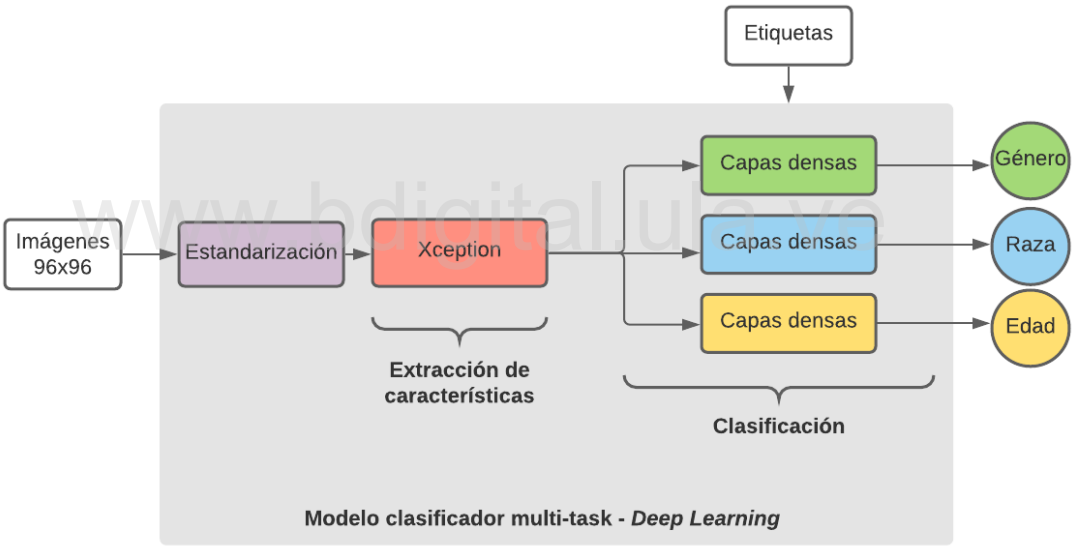

Figura 4.2: Modelo clasificador *multi-task* - *Deep Learning*

4.2. Diseño de pruebas

A continuación se describen las estrategias planteadas para el entrenamiento y validación de los modelos.

4.2.1. Métricas de evaluación de desempeño

Es vital determinar bajo qué criterio se considerará al modelo como "bueno", "malo", o "regular". Para problemas de clasificación la métrica más empleda es *accuracy* o exactitud. No obstante para los modelos finales se utilizarán también otras métricas como recall o exhaustividad, precisión, score F1 o valor F1, y la matriz de confusión, que sirven para tener un panorama más amplio de los resultados.

Sea un problema de clasificación binaria con clases positiva y negativa

TP= True Positive (la clase positiva se predijo correctamente)
TN = True Negative (la clase negativa se predijo correctamente)
FP = False Positive (la clase positiva se predijo como negativa)
FN = False Negative (la clase negativa se predijo como positiva)
Las métricas a utilizar se describen a continuación.

1. **Accuracy:** Indica el número de elementos clasificados correctamente en comparación con el número total de elementos.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

2. **Recall:** Es la proporción de casos positivos que fueron correctamente identificados.

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

3. **Precision:** Representa el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos.

$$precision = \frac{TP}{TP + FP} \tag{4.4}$$

4. **F1-*score***: Es la combinación de las métricas de precisión y recall y sirve de

compromiso entre ellas.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.5}$$

5. **Matriz de confusión:** Como su nombre lo indica es una matriz en la que cada columna representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Son muy útiles porque permiten ver si el algoritmo está confundiendo dos clases. En la siguiente imagen se observa una matriz de confusión para el caso de dos clases, sin embargo, la idea se extiende para más clases.

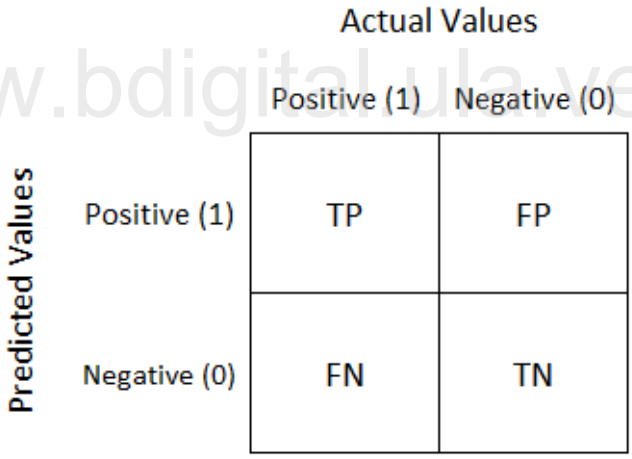

Figura 4.3: Matriz de confusion

4.2.2. Separación de datos de entrenamiento y validación

Para el entrenamiento de los modelos se utiliza un 80 % de los datos, y para validación un 20 %, estos últimos son utilizados únicamente para evaluar los modelos.

4.2.3. Esquemas de entrenamiento

SVM

Una técnica muy utilizada que permite evaluar modelos y evitar el sobreajuste es la validación cruzada, la cual consiste en dividir los datos de entrenamiento en k carpetas, para cada carpeta, un modelo es entrenado usando $k-1$ carpetas, y dejando una para pruebas. El resultado de las k carpetas es promediado. Este procedimiento se observa en la figura 4.4.

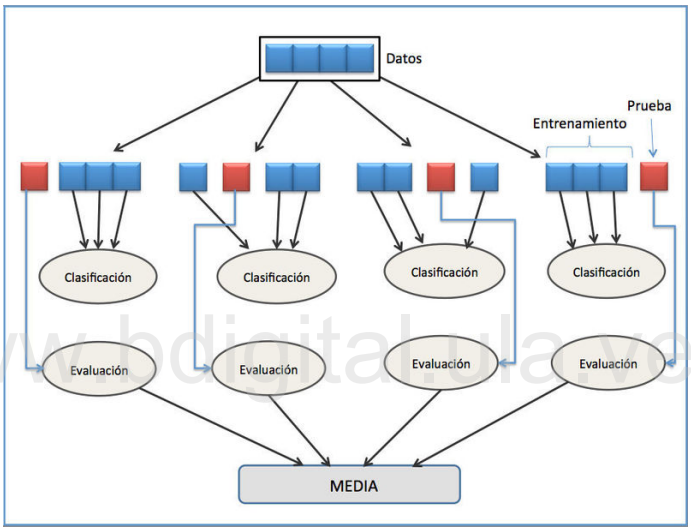

Figura 4.4: Validación cruzada. Recuperado de[5]

Se diseñó un procedimiento para el entrenamiento con los distintos vectores de características construidos. En primer lugar, para cada una de las etiquetas se entrenaron cinco modelos, correspondientes a los cinco descriptores, esto conservando los valores por defecto de los parámetros del modelo establecidos por **Scikit Learn**: (C=1, y $\gamma = 1/$número de características).
Debido a que se ha demostrado anteriormente que la fusión de descriptores resulta beneficioso para el desempeño de los modelos, se tomaron para cada descriptor el modelo con el mejor desempeño, y se fusionaron para crear un vector de características final. Para este se hizo una optimización de los parámetros del SVM con kernel RBF, a fin de buscar el rendimiento óptimo del modelo. Finalmente, el modelo con

[5] https://commons.wikimedia.org/w/index.php?curid=17617674

el mejor desempeño en cada una de las etiqutas fue entrenado sobre los datos de validación. Este procedimiento se muestra en la figura 4.5

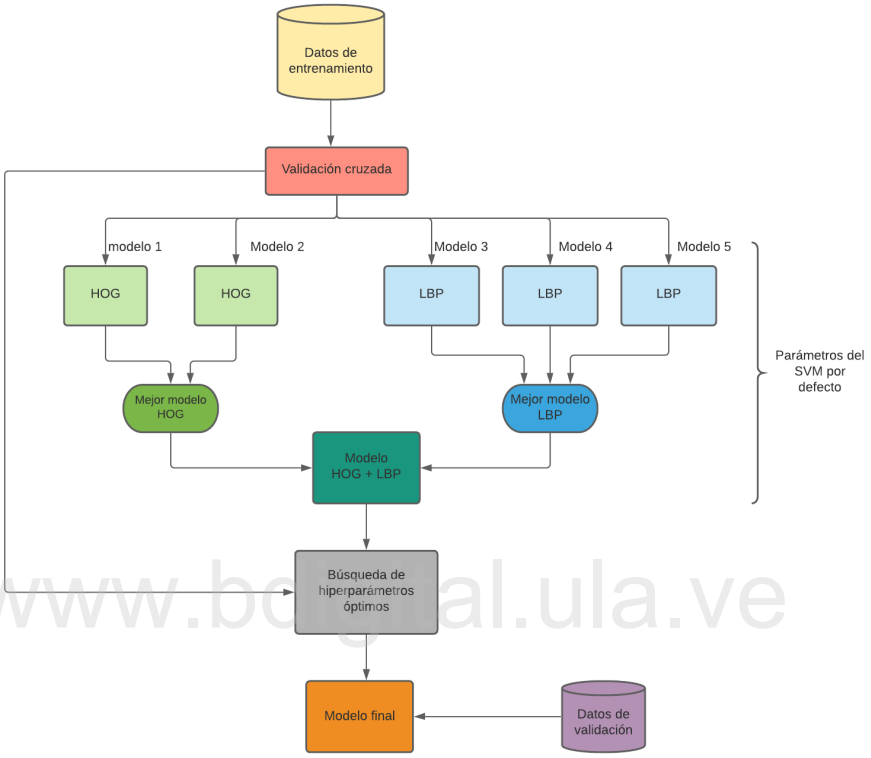

Figura 4.5: Esquema de entrenamiento del SVM

CNN

El procedimiento empleado para el entrenamiento de la red neuronal convolucional basado en *Transfer Learning* se realizó tomando como referencia las indicaciones descritas por la documentación de Keras[6] y Tensorflow[7]. Básicamente consiste en **dos etapas**, la **primera** consiste en el entrenamiento de la capa superficial de la red, donde se "congelan"(los parámetros se hacen no entrenables, por lo tanto no cambian) las capas de convoluciones y se entrenan las capas de clasificación. Durante

[6] https://keras.io/guides/transfer_learning/
[7] https://www.tensorflow.org/guide/keras/transfer_learning

esta etapa se realizó una optimización de parámetros para determinar el rendimiento óptimo del modelo. La **segunda** etapa consiste en la sintonía fina o *fine tuning*, que consiste descongelar todo el modelo o parte de él, y entrenarlo con una taza de aprendizaje pequeña. Esta etapa permite que las capas de extracción de características (convoluciones) se adapten a los datos actuales. El modelo resultante se entrenó con los datos de validación. El esquema descrito se muestra en la figura 4.6.

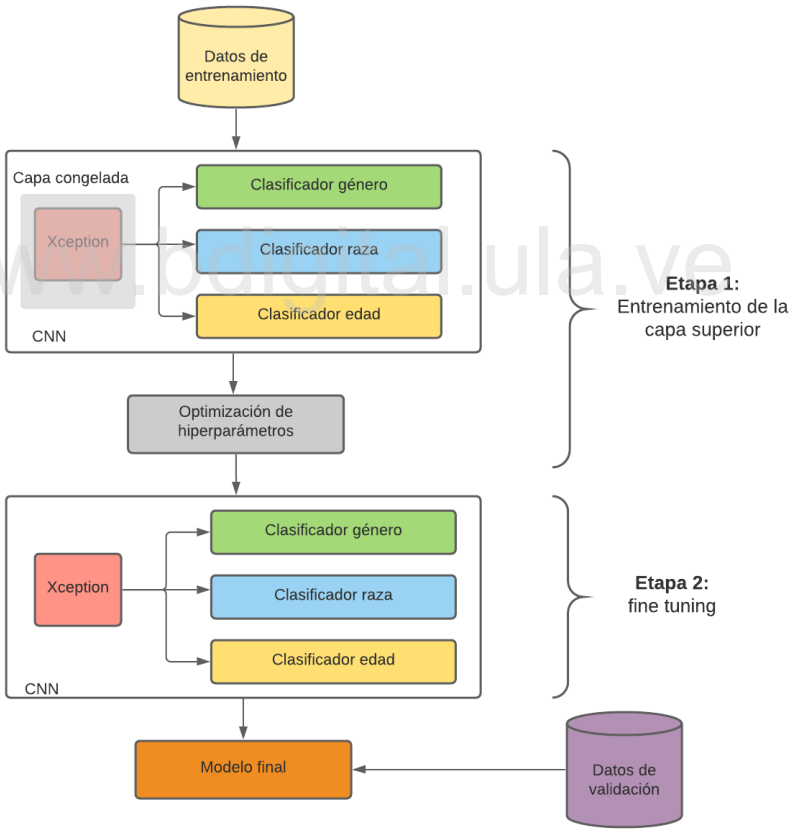

Figura 4.6: Esquema de entrenamiento de la CNN

4.3. Construcción de los modelos

4.3.1. SVM

Para la primera parte de los entrenamientos, se aplica ANOVA, dejando un 25 % de las características de cada descriptor creado anteriormente.

	Género	Raza	Edad
HOG_A	83.65 %	68.53 %	61.61 %
HOG_B	**86.24 %**	**73.21 %**	**64.72 %**
LBP_A	83.57 %	70.5 %	61.47 %
LBP_B	86.847 %	77.48 %	65.82 %
LBP_C	**87.27 %**	**77.5 %**	**67.01 %**

Cuadro 4.1: Entrenamientos con diferentes descriptores. Métrica: *Accuracy*

Como se observa en la tabla 4.1, los descriptores HOG_B y LBP_C producen el mejor desempeño, y ocurre además para las tres etiquetas. El género es la etiqueta con el mejor *accuracy*, seguido por la raza y por último la edad.

En consecuencia, el vector final, común en todas las etiquetas, será el siguiente:

$$HOG + LBP = HOG_B + LBP_C \tag{4.6}$$

Este vector tiene un tamaño de 6960.

Optimización de hiperparámetros

Una vez obtenido el vector de características final, se procedió a realizar una optimización de hiperparámetros de los modelos. Los hiperparámetros a variar serán el número de características utilizados resultantes de aplicar el método estadístico ANOVA, se fijarán 2 valores para las pruebas, un 10 % del tamaño original (696) y un 20 % (1392). También C y γ del SVM. En la tabla 4.2 se muestran todos los hiperparámetros.

Los modelos construidos y los hiperparámetros a evaluar fueron utilizados para los entrenamientos, en las siguientes tablas se muestran los resultados promediados

Parámetro	Valores	Descripción
γ	0.0001, 0.001, 0.005	hiperparámetro del kernel RBF
C	1, 10, 100	hiperparámetro de regularización del SVM
K	696, 1392	tamaño del vector de características

Cuadro 4.2: Hiperparámetros a optimizar

de realizar validación cruzada de $k = 5$ carpetas para cada modelo. La métrica empleada es *accuracy*.

K = 696

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	85.99 %	87.96 %	85.4 %
	10	86.72 %	88.66 %	86.2 %
	100	87.82 %	**89.65 %**	86.2 %

K = 1392

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	87.72 %	89.46 %	59.4 %
	10	88.54 %	**89.81 %**	63.2 %
	100	88.53 %	**89.81 %**	63.2 %

Cuadro 4.3: Optimización de parámetros para el género

K = 696

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	75.73 %	79.13 %	74-87 %
	10	77.58 %	79.75 %	75.69 %
	100	77.31 %	**79.76 %**	75.69 %

K = 1392

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	77.62 %	80.26 %	37.47 %
	10	78.56 %	80.8 %	41.27 %
	100	77.22 %	**80.96 %**	41.27 %

Cuadro 4.4: Optimización de parámetros para la raza

K = 696

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	63.51 %	66.83 %	63.9 %
	10	65.82 %	66.62 %	64.99 %
	100	64.54 %	**66.63 %**	64.99 %

K = 1392

		gamma		
		0.0001	**0.001**	**0.005**
C	**1**	65.44 %	67.98 %	43.03 %
	10	67.04 %	**68.78 %**	43.5 %
	100	63.9 %	**68.78 %**	43.5 %

Cuadro 4.5: Optimización de parámetros para la edad

Los resultados muestran que, para todas las etiquetas, los modelos obtienen un mejor desempeño para $\gamma = 0{,}001$. Por otro lado, es notable para $\gamma = 0{,}005$ el desempeño de los modelos disminuye considerablemente, e incluso no se muestran cambios

aún cuando el hiperparámetro C aumenta (mayor regularización), un ejemplo de esto es en la tabla 4.3, el *accuracy* se mantiene en 63,2 % para $C = 10$ y $C = 100$, esto se debe a que un valor relativamente alto de γ hizo que el área de influencia de las muestras fuera muy reducida, por lo que el modelo tiene sobreajuste y no puede generalizar correctamente, aún cuando se varíe C.

Se observa también que para $K = 1392$ se obtienen resultados ligeramente mejores que para $K = 696$, aún cuando el primer caso tiene el doble de características que el segundo, esto sugiere que la selección de características empleada pudo capturar aquellas más relevantes para predecir la etiqueta.

La etiqueta género obtuvo el mejor e idéntico desempeño para $C = 10$ y $C = 100$, con $\gamma = 0{,}001$ utilizando $K = 1392$.

Para la etiqueta raza el modelo con mayor accuracy se obtiene para $C = 100$, $\gamma = 0{,}001$ y $K = 1392$.

Por último, la etiqueta edad análogamente al género, obtuvo el mejor desempeño para $C = 10$ y $C = 100$ con $\gamma = 0{,}001$ y $K = 1392$.

En la tabla 4.6 se muestra la combinación óptima de hiperparámetros para cada etiqueta, para el caso en que se obtuvo el mismo puntaje, se tomó $C = 10$.

	Etiquetas		
	Género	**Raza**	**Edad**
C	10	100	10
gamma	0.001	0.001	0.001
K	1392	1392	1392
Accuracy	89.81 %	80.96 %	68.78 %

Cuadro 4.6: Hiperparámetros óptimos de cada etiqueta

Validación de los resultados

Los modelos fueron entrenados con los hiperparámetros óptimos empleando todos los datos de entrenamiento, y posteriormente se probaron con los datos de validación. A continuación se muestran los resultados para cada etiqueta.

Género

Reporte de Clasificación - Género

	Precision	Recall	F1-score	Accuracy	Muestras
Masculino	89.36 %	91.20 %	90.27 %	89.91 %	2432
Femenino	90.52 %	88.56 %	89.53 %		2307

Cuadro 4.7: Métricas del clasificador del género

		Predicción	
		Masculino	**Femenino**
Clase real	**Masculino**	2218	214
	Femenino	264	2043

Cuadro 4.8: Matriz de confusión del clasificador del género

Raza:

Reporte de Clasificación - Raza

	Precision	Recall	F1-score	Accuracy	Muestras
Caucásico	82.17 %	89.36 %	85.61 %	82.3 %	2001
Africano/ Af.Americano	87.29 %	82.54 %	84.85 %		882
Asiático	88.32 %	82.87 %	85.51 %		712
Latino/M.Oriente/ Indio	74 84 %	69.41 %	72.02 %		1144

Cuadro 4.9: Métricas del clasificador de la raza

		Predicción			
		Caucásico	**Africano/ Af.Americano**	**Asiático**	**Latino/ M.Oriente/ Indio**
Clase real	**Caucásico**	1788	35	36	142
	Africano/ Af.Americano	56	728	11	87
	Asiático	70	14	590	38
	Latino/M.Oriente/ Indio	262	57	31	794

Cuadro 4.10: Matriz de confusión del clasificador de la raza

Edad:

Reporte de clasificación - Edad

	Precision	Recall	F1-score	Accuracy	Muestras
0 - 10	93.66 %	88.62 %	91.07 %	69.32 %	650
11 - 20	57.93 %	30.84 %	40.25 %		308
21 - 35	71.13 %	89.76 %	79.36 %		2031
36 - 50	51.13 %	35.63 %	41.99 %		828
50 - 65	54.76 %	50.59 %	52.59 %		591
65 +	71.9 %	59.52 %	65.12 %		331

Cuadro 4.11: Métricas del clasificador de la edad

		Predicción					
		0 - 10	**11 - 20**	**21 - 35**	**36 - 50**	**51 - 65**	**66 +**
Clase real	**0 - 10**	576	33	37	2	1	1
	11 - 20	33	95	177	2	0	1
	21 - 35	2	32	1823	143	29	2
	36 - 50	1	2	413	295	108	8
	51 - 65	2	2	98	125	299	65
	66 +	1	0	15	10	108	197

Cuadro 4.12: Matriz de confusión del clasificador de la edad

4.3.2. CNN

Se construyó la red neuronal planteada, al final del modelo base *Xception* se utilizó una capa de *average pooling* global, para los clasificadores de cada etiqueta se añadieron capas densas, después de la primera capa de cada etiqueta se añade una capa de normalización, y una capa de *Dropout* de 0.5 después de cada capa densa para evitar problemas de sobreajuste. Se tomó un tamaño de *batch size* de 32, y se utilizó un 20 % de los datos de entrenamiento para validación al final de cada época. Se empleó el criterio de detención temprana o *early stopping* con paciencia de tres épocas, es decir, si la pérdida no cambia durante tres épocas, el entrenamiento se detiene y se mantienen los mejores pesos obtenidos (los que arrojan el valor mínimo

para la función de coste). En la figura 4.7 se observa la estructura de la red neuronal convolucional.

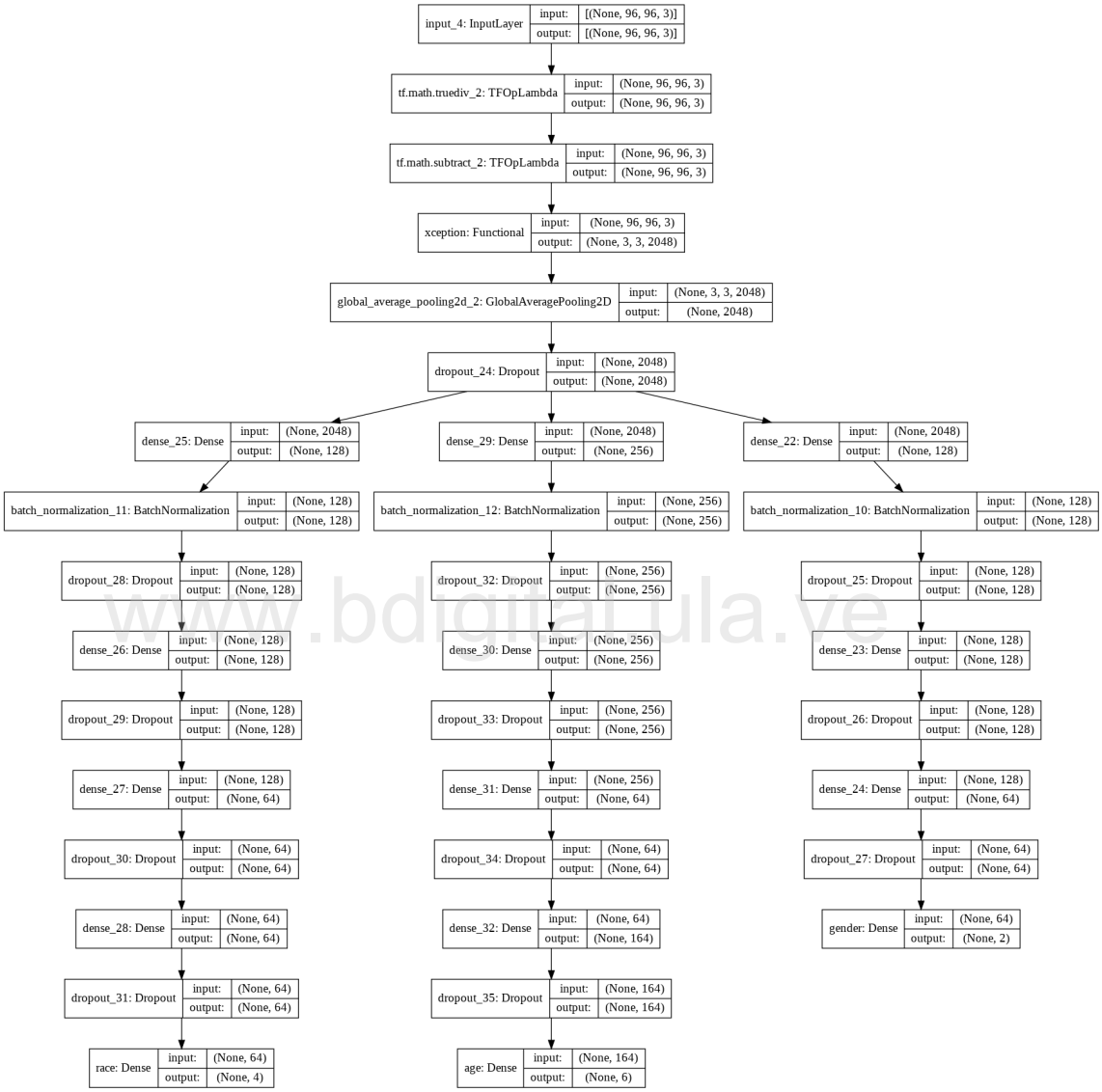

Figura 4.7: Red neuronal convolucional *multi-task*

Optimización de hiperparámetros

Los hiperparámetros a variar fueron la tasa de aprendizaje y el optimizador, en la tabla 4.13 se muestran los valores y opciones consideradas. Como ya se dijo

anteriormente, esto se hizo para la primera etapa del entrenamiento, que corresponde al entrenamiento de la capa superior del modelo. Como este se trata de un modelo con múltiples salidas, se toma como criterio de selección la función de coste, ya que esta toma en cuenta la pérdida de cada una de las etiquetas.

Hiperparámetro	Valores/Opciones	Descripción
Tasa de aprendizaje	0.01, 0.001, 0.0001	Determina el tamaño del paso en cada iteración
Optimizador	Adam, Adadelta, SGD	Algoritmo utilizado para actualizar los parámetros de la red

Cuadro 4.13: Hiperparámetros a optimizar para la CNN

La red fue entrenada durante 15 épocas . En la tabla 4.14 se observan los resultados. La mejor combinación de hiperparámetros se obtiene con el optimizador Adam y una tasa de aprendizaje de 0.001.

		Optimizador		
		Adam	**Adadelta**	**SGD**
Taza de aprendizaje	**0.01**	4.1	4.29	3.35
	0.001	**3.14**	4.94	3.83
	0.0001	3.45	5.44	4.57

Cuadro 4.14: Hiperparámetros óptimos para la CNN

La curva de aprendizaje fue registrada con los hiperparámetros óptimos, esto se visualiza en la figura 4.8, donde se observa que el entrenamiento se detiene en la época 14. La tabla 4.15 muestra los resultados para el *accuracy* de cada etiqueta.

	Género	Raza	Edad
Accuracy	83.62 %	57.79 %	58.45 %

Cuadro 4.15: Resultados del entrenamiento de la capa superior

Los resultados de la primera etapa del entrenamiento muestran que sólo entrenando la capa de clasificación creada, se logra diferenciar bastante bien la etiqueta del género, alcanzando más del 80 % de accuracy. Por otro lado las etiquetas de la raza y la edad no superan el 60 %.

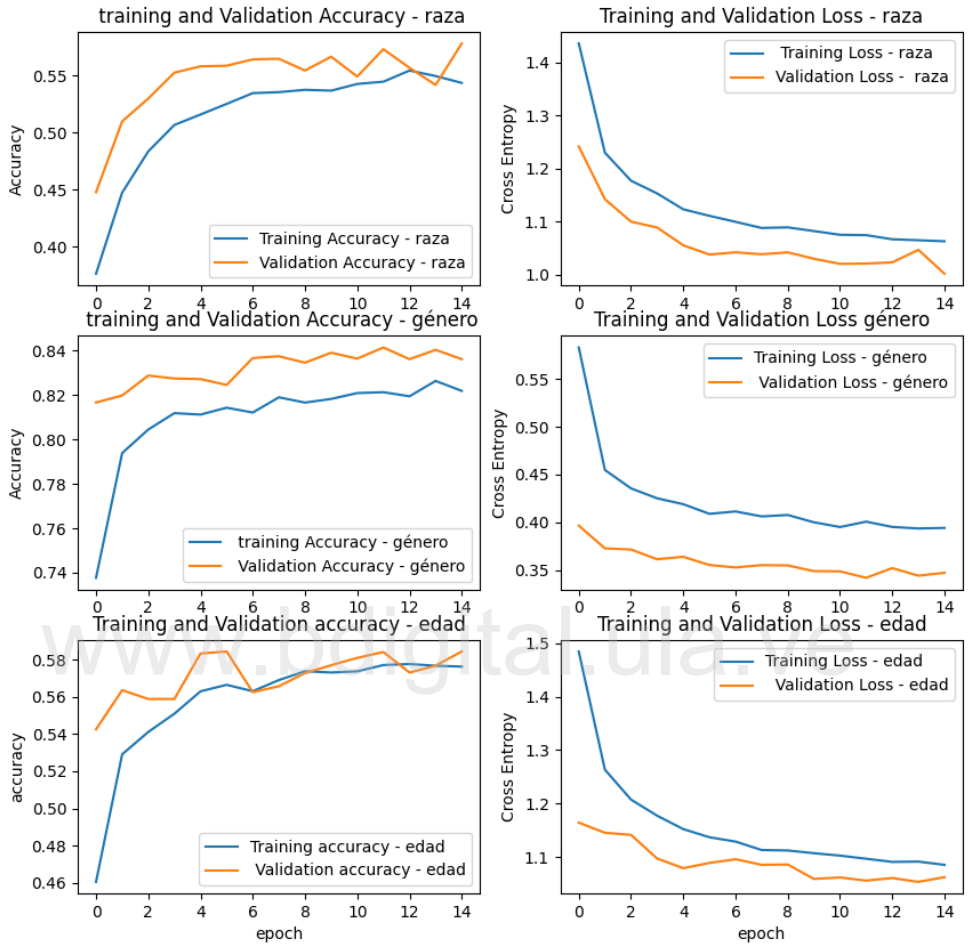

Figura 4.8: Curvas de aprendizaje del entrenamiento de la capa superior

Fine tuning

Una vez obtenidos los hiperparámetros que minimizan la función de coste durante el entrenamiento de la capa superior del modelo, se procedió a entrenar todo el modelo con una taza de aprendizaje 10 veces más pequeña, esto para que los cambios en los pesos que ya tenía anteriormente el modelo base fueran lo suficientemente pequeños como para evitar que destruyeran lo aprendido. En la tabla 4.16 se detallan

los resultados para cada etiqueta, la curva de aprendizaje se observa en la figura 4.9.

	Género	Raza	Edad
Accuracy	91.66 %	83.65 %	70.22 %

Cuadro 4.16: Resultados del entrenamiento - *fine tuning*

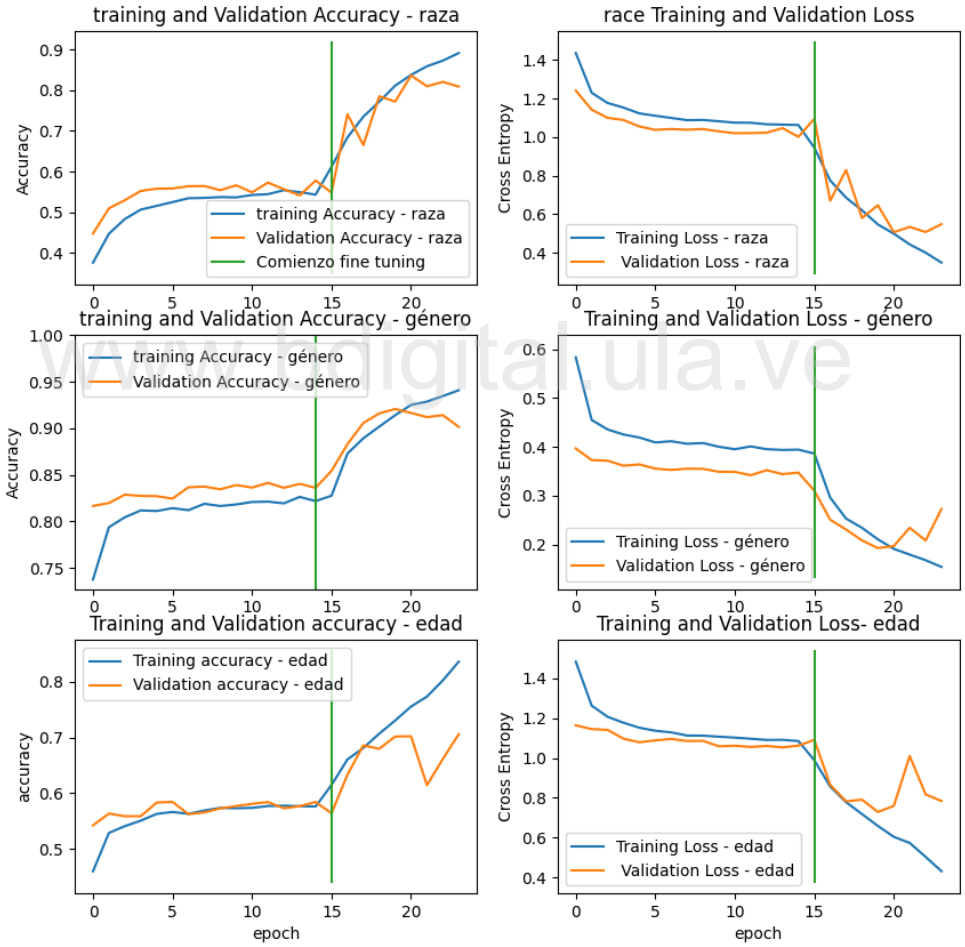

Figura 4.9: Curvas de aprendizaje - *fine tuning*

Se observa que la segunda etapa del entrenamiento permite aumentar el desempeño en todas las etiquetas, y lo hace durante las primeras 5 épocas (hasta la época 20), luego la pérdida del modelo deja de disminuir, y este empieza a sobreajustarse a los datos de entrenamiento, en consecuencia el desempeño con los datos de validación deja de mejorar. En contraste con la etapa anterior, se observa que los nuevos pesos la capa de convoluciones se adaptaron a los nuevos datos y permitieron aumentar el desempeño del modelo en cada una de las etiquetas.

Validación de los resultados

El modelo fue entrenado con los datos validación, arrojando los siguientes resultados para cada etiqueta.

Género

Reporte de Clasificación - Género

	Precision	**Recall**	**F1-score**	**Accuracy**	**Muestras**
Femenino	89.03 %	94.88 %	91.86 %	91.2 %	2481
Masculino	93.94 %	87.16 %	90.42 %		2258

Cuadro 4.17: Métricas del clasificador del género - CNN

		Predicción	
		Masculino	**Femenino**
Clase real	**Masculino**	2354	127
	Femenino	290	1968

Cuadro 4.18: Matriz de confusión del clasificador del género - CNN

Raza

Reporte de Clasificación - Raza					
	Precision	**Recall**	**F1-score**	**Accuracy**	**Muestras**
Caucásico	86.36 %	88.53 %	87.43 %	82.99 %	2066
Africano/ Af.Americano	89.83 %	81.81 %	85.63 %		896
Asiático	83.57 %	88.37 %	85.90 %		679
Latino/M.Oriente/ Indio	70.93 %	70.22 %	70.57 %		1098

Cuadro 4.19: Métricas del clasificador de la raza - CNN

		Predicción			
		Caucásico	**Africano/ Af.Americano**	**Asiático**	**Latino/ M.Oriente/ Indio**
Clase real	**Caucásico**	1829	22	56	159
	Africano/ Af.Americano	31	733	13	119
	Asiático	38	3	600	38
	Latino/M.Oriente/ Indio	220	58	49	771

Cuadro 4.20: Matriz de confusión del clasificador de la raza - CNN

Edad

Reporte de clasificación - Edad					
	Precision	**Recall**	**F1-score**	**Accuracy**	**Muestras**
0 - 10	80.79 %	95.61 %	87.58 %	69.26 %	638
11 - 20	51.02 %	32.05 %	39.37 %		312
21 - 35	78.54 %	80.56 %	79.54 %		2022
36 - 50	48.62 %	34.94 %	40.66 %		807
50 - 65	53.35 %	64.24 %	58.29 %		632
65 +	68.36 %	77.74 %	72.75 %		328

Cuadro 4.21: Métricas del clasificador de la edad - CNN

		Predicción					
		0 - 10	**11 - 20**	**21 - 35**	**36 - 50**	**51 - 65**	**66 +**
Clase real	**0 - 10**	610	21	7	0	0	0
	11 - 20	91	100	117	1	2	1
	21 - 35	43	67	1629	223	60	0
	36 - 50	8	8	278	282	225	6
	51 - 65	1	0	42	72	406	111
	66 +	2	0	1	2	68	255

Cuadro 4.22: Matriz de confusión del clasificador de la edad - CNN

4.4. Interpretación de los modelos

Los resultados observados en la validación de cada uno de los modelos pueden analizarse en cada etiqueta por separado.
Para el **género**, se observa un *accuracy* de casi un 90 % para el SVM, y valores similares para *precision* y *recall*, por lo que el modelo es capaz de diferenciar adecuadamente entre una u otra clase, la matriz de confusión muestra que el modelo confunde apenas un 10 % de las muestras con la clase opuesta, para la CNN se ve una ligera mejora y se obtuvo un 91.2 %, de igual forma este modelo es capaz de diferenciar bastante bien rostros de uno u otro género.

La **raza** obtuvo un *accuracy* de 82.3 % para el SVM y 82.99 % para la CNN. La clase con el desempeño más deficiente fue la 4 (Latino/M.Oriente/Indio), donde para el SVM se obtuvo un *recall* de 69.41 %, es decir, el modelo pudo "reconocer" casi un 70 % de los rostros de la clase 4 como tal, y en consecuencia, pasó por alto casi un 30 %. La *precision* indica que, cuando predijo un rostro como perteneciente a la clase 4, estaba en lo correcto un 74.84 %. Vemos que para esta clase se tiene un *recall* menor respecto a la precision, lo que indica que hay más Falsos Negativos que Falsos Positivos, es decir, comete mayores errores al predecir que "un rostro no pertenece a la clase 4", cuando en realidad sí. El *recall* y *precisión* de la CNN son muy similares, 70.93 % y 70.22 % respectivamente, por lo que este modelo tiene un comportamiento más equilibrado en los Falsos Negativos y Falsos Positivos que se producen. Observando la matriz de confusión de ambos modelos, vemos que esto ocurre principalmente con la clase 0, esto es, confundir a la clase caucásico con la

clase latino/M.Oriente/Indio. El resto de clases en ambos modelos tienen métricas que están en un intervalo de 80 % a 90 %, con valores similares en las métricas.
Finalmente la **edad**, que es la etiqueta con mayor número de clases, obtuvo un *accuracy* de 69.32 % para el SVM y 69.26 % para la CNN. El desempeño es variado para distintas clases, vemos que para los rostros de 0 – 10 años (clase 0) se tiene un *F-score* de 91.07 % para el SVM, *precision* es la métrica con mayor valor, obteniendo un 93.66 %, lo que sugiere que el modelo clasifica correctamente un rostro con una edad de 0-10 años más del 90 % de las veces. En contraste, ocurre lo contrario para la CNN, donde es el *recall* el que obtuvo el mayor valor, con un 95.61 %. Esto nos dice que, el modelo SVM es más "preciso" al reconocer esta clase, o lo que es lo mismo, los rostros que clasifica como pertenecientes a esta, en su mayoría lo son, sin considerar si omite algunas muestras de esta clase; mientras que el modelo CNN es más "sensible" o "exhaustivo", es decir, trata de pasar por alto la menor cantidad posible de muestras de esta clase, aun cuando esto implica incluir muestras de otras clases. Para el rango de edades entre 21-35 años (clase 2) con el SVM se obtuvo un *F1-score* de 79.36 %, la precision (71.13 %) es casi un 20 % menor que el *recall* (89.76 %), lo que indica que reconoce un buen número de muestras de la clase 3, pero también incluye muestras de otras clases, la matriz de confusión corrobora esto al mostrar 413 muestras clasificadas como pertenecientes a la clase 3, cuando en realidad pertenecían a la clase 4, el siguiente rango de edades. El mismo patrón se observó en la CNN, donde el *recall* de la clase 2 supera casi en un 20 % a la precisión, y donde el modelo tuvo problemas para diferenciar entre rostros de 21-35 y 35-60 años. En las clases 1, 3 y 4 se tiene que de igual forma para ambos modelos, existen algunas muestras que son clasificadas erróneamente entre clases adyacentes.
Por otro lado, es de notar que la matrices de confusión muestran que son muy pocos los rostros clasificados incorrectamente en rangos de edades alejados entre sí, por ejemplo, predecir un rostro como 0 – 10 años cuando en realidad está en el rango 51-65 años o 66+ años.

Capítulo 5

Evaluación e Implementación

Este capítulo abarca las dos fases finales de la metodología, donde se evalúan los modelos finales construidos, y se lleva a cabo la implementación del prototipo de procesador de imágenes.

5.1. Evaluación

En esta fase se realizó un análisis acerca de los resultados obtenidos así como una comparación de los modelos construidos.

5.1.1. Evaluación de los resultados

En la tabla 5.1 se muestra el *accuracy* para cada etiqueta de los dos modelos propuestos, así como el tiempo que le toma a cada modelo realizar la clasificación de una imagen.

Desempeño de los modelos propuestos

	Género	**Raza**	**Edad**	**Tiempo de clasificación (1 imagen)**
SVM	89.91 %	82.3 %	**69.32 %**	0.68 s
CNN	**91.2 %**	**82.99 %**	69.26 %	0.36 s

Cuadro 5.1: Desempeño de los modelos propuestos

Como se observa, el modelo SVM sólo supera al modelo CNN en la etiqueta edad, esta obtuvo el menor índice de rendimiento en ambos modelos, sobre esto hay que tener en cuenta que la edad es un atributo muy complejo de determinar basándose sólo en el rostro de una persona, pues depende de muchos factores ambientales, e incluso para una persona es difícil de estimar efectivamente. No obstante, los modelos fueron capaces de capturar muy bien diferencias entre personas jóvenes y ancianas; por otro lado, tuvieron problemas al clasificar personas en edades de clases adyacentes, lo cual es entendible, ya que como se ha dicho, el patrón de envejecimiento es muy variado entre las personas y, por ejemplo, una persona de 35 años, bien pudiera aparentar físicamente 40 o 30 años. Las dos etiquetas restantes tuvieron buenos resultados, el error en los modelos principalmente estuvo presente al clasificar razas, sin embargo, esta etiqueta obtuvo más del 80 % de aciertos, lo que sugiere que los modelos pudieron capturar patrones que diferenciaron bastante bien una clase de otra. Por otro lado, al SVM le toma casi el doble de tiempo realizar la clasificación de una imagen respecto a la CNN.

Otro aspecto a considerar para la evaluación de estos modelos son las diferencias que se hicieron notar durante el desarrollo de los mismos, tomando en consideración el caso de análisis de imágenes como fue el caso de esta investigación. Esto se describe en la tabla 5.2.

5.2. Implementación

Sistema procesador de imágenes

Una vez obtenido el modelo que hará la clasificación de las etiquetas, se integró a un sistema de procesamiento de imágenes, para esto se construyó una sencilla aplicación web utilizando Flask[1], framework de Python y Jinja[2], un motor de plantillas web para este mismo lenguaje. Se diseñó una Interfaz de Usuario para adjuntar la imagen y posteriormente ser enviada al servidor creado por Flask, que contiene la CNN que procesará dicha imagen, donde como primer paso se realiza un pre-procesamiento a

[1] https://flask.palletsprojects.com/en/2.0.x/
[2] https://jinja.palletsprojects.com/en/3.0.x/

SVM (Machine learning)	CNN (Deep learning)
1.- La extracción de características se realiza como paso previo al entrenamiento del modelo, a través de técnicas que permiten extraer y codificar información a partir de píxeles en bruto, todo este proceso supone un paso de preprocesamiento.	1.- La extracción de características es realizada por la misma red, a través de las capas de convoluciones, encargadas de extraer patrones en los datos, esto no supone trabajo extra, ya que se realiza durante el entrenamiento de la red.
2.- El entrenamiento de los algoritmos requiere de menos recursos computacionales. Sin embargo, el uso de kernels aumenta los tiempos de entrenamiento.	2.- El entrenamiento de redes neuronales sin GPU puede tomar tiempos considerablemente largos.
3.- El modelo general está conformado por 3 modelos individuales, cada uno se encarga de la clasificación de una etiqueta	3.- Es un único modelo multitarea que realiza la clasificación para todas las etiquetas simultáneamente

Cuadro 5.2: Tabla comparativa de los modelos SVM y CNN

la misma, que consiste en la misma operación realizada durante la construcción del modelo (CNN), esto es la estandarización de los píxeles de la imagen, ya que este es el formato que sabe reconocer el modelo como entrada (bajo el cual fue entrenado). Luego se realiza un proceso de detección y alineamiento de rostros, esto último a través de la detección de los 68 puntos faciales. Tanto para la detección como el alineamiento de los rostros se hizo uso de la librería dlib[3], particularmente su API para Python. Los rostros detectados son entonces enviados al modelo clasificador (CNN), el cual arroja el correspondiente resultado y finalmente es mostrado a través de la UI. Para el manejo de las imágenes e impresión de los resultados se utilizó la librería OpenCV[4]. En el diseño de la UI se utilizó Bootstrap[5]. El esquema descrito se observa en la figura 5.1.

La figura 5.2 muestra el flujo de procesamiento de una imagen hasta generar el resultado. Por otro lado, la Interfaz de Usuario de la aplicación web creada, y el resultado de procesar una imagen se obervan en las figuras 5.3 y 5.4, respectivamente.

[3] http://dlib.net/
[4] https://opencv.org/
[5] https://getbootstrap.com/

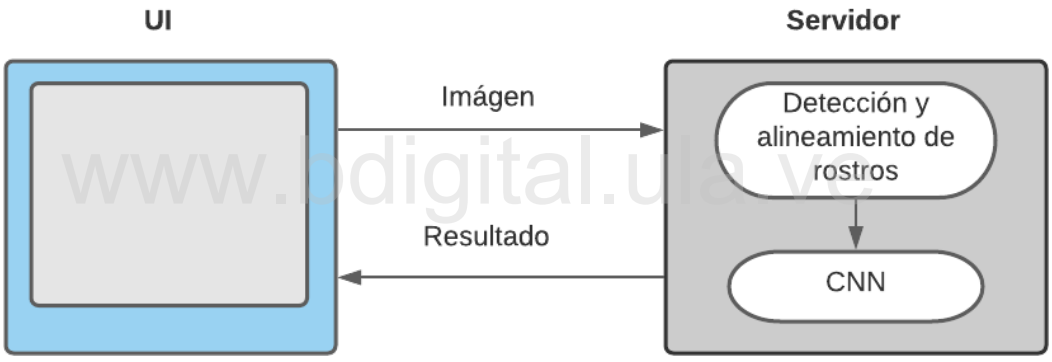

Figura 5.1: Esquema de la aplicación web del sistema de procesamiento de imágenes

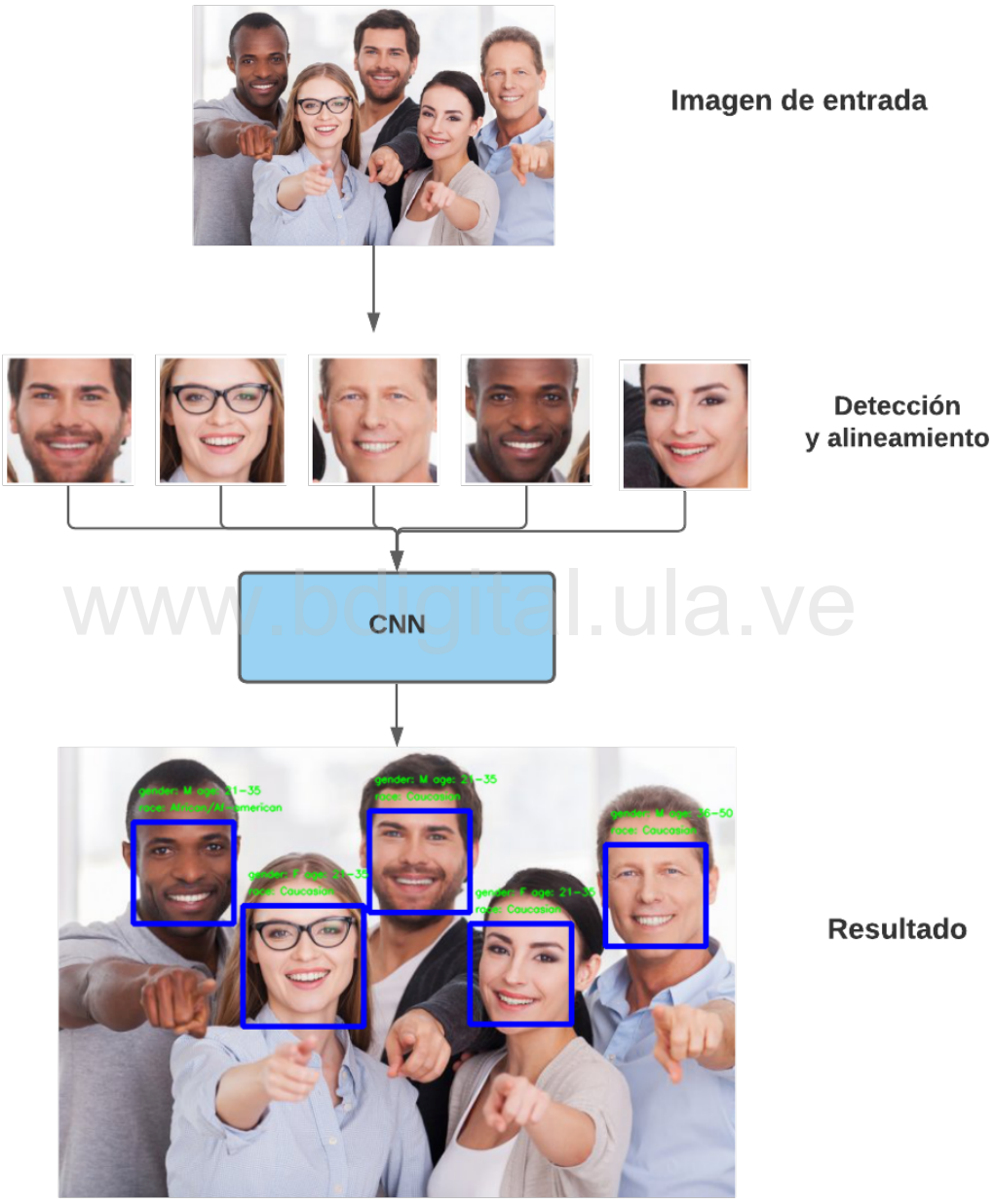

Figura 5.2: Sistema de procesamiento de imágenes

Figura 5.3: UI de la aplicación web

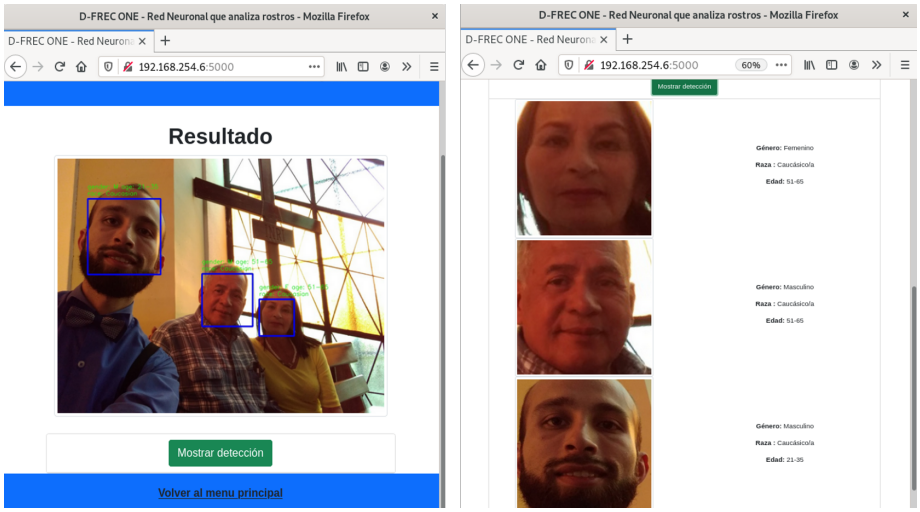

Figura 5.4: Muestra del resultado - UI de la aplicación web. A la izquierda se observa el análisis a la imágen completa y a la derecha los resultados por cada rostro.

Capítulo 6

Conclusiones y Recomendaciones

La investigación realizada llevó a cabo el desarrollo e implementación de un prototipo de procesador de imágenes de rostros para la estimación de características faciales. Para esto se hizo uso de la metodología CRISP-DM, que es ampliamente utilizada en el ámbito de análisis de datos, los cuales son imprescindibles para construir modelos de Inteligencia Artificial. En primer lugar, se realizó un análisis y procesamiento de los datos, a fin de entender sus características y prepararlos para usarlos como materia prima para los entrenamientos. Una vez los datos fueron procesados, se determinaron las técnicas y herramientas en el campo de IA que serían aplicados en el desarrollo de este proyecto. Con esto en mente, se diseñaron los modelos y se realizaron entrenamientos con los datos previamente procesados, se emplearon técnicas de Machine Learning y Deep Learning, los modelos de ambas temáticas fueron analizados individualmente y luego en una visión global. Ambos enfoques requirieron de un estudio de parámetros de los algoritmos e iterativas pruebas y evaluaciones. Para estas pruebas se diseñó un esquema para plantear cómo iban a ser llevadas a cabo para evaluar el desempeño de los modelos. Las mismas demuestran que se logró a través del procesamiento de datos, utilización de herramientas, y el empleo de técnicas de IA, el desarrollo de modelos que permiten estimar el género, raza y edad en imágenes de rostros. Finalmente, se implementó un sistema de procesamiento de imágenes teniendo como base el modelo que obtuvo el mejor desempeño, el cual fue la Red neuronal convolucional (CNN). Este esquema puede detectar uno o varios rostros presentes en una imagen y estimar las características ya

mencionadas. Dicho prototipo diseñado y desarrollado en esta investigación puede ser utilizado como procesador de imágenes para detección y análisis facial.

Como recomendaciones para este trabajo, se proponen las siguientes:

1. Para modelos de *Machine Learning*, utilizar otros descriptores visuales para la extracción de características y verificar su eficacia.
2. Emplear métodos de selección de características más complejos, como el método de Información Mutua (MI).
3. Emplear arquitecturas de redes neuronales que hayan sido entrenadas previamente con grandes cantidades de datos de rostros, para que el proceso de aprendizaje por transferencia sea mucho más efectivo.
4. Añadir características adicionales, como pueden ser emociones, por supuesto disponiendo primero de datos etiquetados con dichas características.
5. Utilizar Datasets con mayor cantidad de muestras y de ser posible, con clases más balanceadas.
6. Emplear una configuración de procesamiento distribuído, para agilizar los tiempos de entrenamiento y así poder utilizar más datos y modelos más complejos (mayor número de características, redes neuronales más profundas, etc).

Apéndice A: Instalación y configuración de Anaconda

Anaconda es una excelente herramienta para crear entornos virtuales y gestionar paquetes y librerías de Python cuando se trata de Ciencia de Datos y *Machine Learning*.

Instalación

1. Descargar el Instalador para Linux.

2. Luego, vaya al directorio donde se encuentra el archivo y ejecute, para Python 3.7:

   ```
   $ bash ~/Downloads/Anaconda3-2020.02-Linux-x86_64.sh
   ```

   Y para Python 2.7:

   ```
   $ bash ~/Downloads/Anaconda2-2019.10-Linux-x86_64.sh
   ```

3. El instalador se muestra por pantalla y debe seguir las instrucciones

Creación de un entorno virtual

Para crear un entorno virtual y especificar algunos paquetes a instalar en él:

```
$ conda create -n InteligenciaArtificial python=3.7 spyder
```

De esta manera se creará un entorno virtual llamado InteligenciaArtificial con Python en su versión 3.7 y el IDE Spyder (PyCharm también es ampliamente recomendado). Para activar el entorno creado:

```
$ conda activate InteligenciaArtificial
```

Instalación de librerías

A continuación se listan las librerías necesarias para el desarrollo de los algoritmos utilizados en este proyecto, algunos deben usarse en una versión específica a fin de evitar problemas de compatibilidad con otras librerías o con la versión de Python de la que dispone el entorno creado.

- OpenCV
  ```
  $ conda install -c conda-forge opencv
  ```

- Scikit-image
  ```
  $ pip3 install -U scikit-image==0.16.2
  ```

- Scikit-learn
  ```
  $ pip3 install -U scikit-learn==0.22.1
  ```

- Tensorflow
  ```
  $ pip3 install tensorflow==2.3.0
  ```

- Dlib
  ```
  $ conda install -c conda-forge dlib
  ```

- Flask

```
$ pip3 install Flask==1.1.2
```

Bibliografía

Arista-Jalife, A., Calderón-Auza, G., Fierro-Radilla, A., y Nakano, M. (2017). Clasificación de imágenes urbanas aéreas: Comparación entre descriptores de bajo nivel y aprendizaje profundo. *Información tecnológica*, *28*(3), 209–224.

Azarmehr, R., Laganiere, R., Lee, W.-S., Xu, C., y Laroche, D. (2015). Real-time embedded age and gender classification in unconstrained video. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Bellman, R. (1978). *An introduction to artificial intelligence: Can computers think?*. Thomson Course Technology.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Chauhan, M., Mayank y Sakle (2014). Study & analysis of different face detection techniques. *International Journal of Computer Science and Information Technologies*, *5*(2), 1615–1618.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 1251–1258).

Dalal, N., y Triggs, B. (2005). Histograms of oriented gradients for human detection. En *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, (pp. 886–893 vol. 1).

Dharavath, K., Talukdar, F. A., y Laskar, R. H. (2014). Improving face recognition

rate with image preprocessing. *Indian Journal of Science and Technology*, *7*, 1170—1175.

Eidinger, E., Enbar, R., y Hassner, T. (2014). Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, *9*(12), 2170–2179.

Guyon, I., y Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157–1182.

Huang, D., Shan, C., Ardabilian, M., Wang, Y., y Chen, L. (2011). Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(6), 765–781.

Huerta, I., Fernández, C., Segura, C., Hernando, J., y Prati, A. (2015). A deep analysis on age estimation. *International Association for Pattern Recognition (IAPR)*, (pp. 1—11).

Ito, K., Kawai, H., Okano, T., y Aoki, T. (2018). Age and gender prediction from face images using convolutional neural network. En *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, (pp. 7–11). IEEE.

Levi, G., y Hassner, T. (2015). Age and gender classification using convolutional neural networks. En *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) workshops*.
URL https://osnathassner.github.io/talhassner/projects/cnn_agegender

Martinsanz, G., y García, J. (2008). *Visión por computador: imágenes digitales y aplicaciones*. Alfaomega.
URL https://books.google.com.mx/books?id=0odHPwAACAAJ

Mueller, J. P., y Massaron, L. (2016). *Machine learning for dummies*. John Wiley & Sons.

Nilsson, N. J. (1998). *Artificial intelligence: a new synthesis*. Morgan Kaufmann.

Ojala, T., Pietikäinen, M., y Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, *29*(1), 51–59.

Ojala, T., Pietikainen, M., y Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, *24*(7), 971–987.

Pasandi, M. E. M. (2014). *Face, Age and Gender Recognition Using Local Descriptors*. Ph.D. thesis, Citeseer.

Pietikäinen, M. (2010). Local Binary Patterns. *Scholarpedia*, *5*(3), 9775. Revision #188481.

Poole, D. L., y Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Rich, E., y Knight, K. (1991). Artificial intelligence, chapter the frame problem. *Internation Edition,*.

Russell Stuart, J., y Norvig, P. (2009). *Artificial intelligence: a modern approach*. Prentice Hall.

Sarfraz, M. S., Hellwich, O., y Riaz, Z. (2010). *Feature Extraction and Representation for Face Recognition*. IntechOpen.

Shearer, C. (2000). The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, *5*(4), 13–22.

Sousa, K., y Mora, C. (2016). Sistema de seguridad basado en reconocimiento facial utilizando una raspberry pi. Trabajo de grado, Universidad Central de Venezuela.

Tapia, J. E., y Perez, C. A. (2013). Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of lbp, intensity, and shape. *IEEE Transactions on Information Forensics and Security*, *8*(3), 488–499.

Viola, P., y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, (pp. I–I). IEEE.

West, J., Ventura, D., y Warnick, S. (2007). Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, *1*(08).

Winston, P. H. (1992). Artificial intelligence. *Addison-Wesley*.