ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

“Reconocimiento facial: técnicas tradicionales y técnicas de aprendizaje profundo, un análisis”

TRABAJO DE TITULACIÓN

Previo a la obtención del Título de:
MAGISTER EN CIENCIAS DE LA COMPUTACIÓN

Presentado por:
Shendry Balmore Rosero Vásquez

GUAYAQUIL – ECUADOR

AÑO: 2019

RESUMEN

El Reconocimiento Facial se ha convertido en los últimos años en la piedra angular de las propuestas de modelos que intentan de manera automatizada identificar un rostro, aplicación que derivaría en el desarrollo de múltiples sistemas orientados a la seguridad (control de acceso), búsqueda y rescate de personas desaparecidas entre las más importantes por el momento. El problema de la mayoría, tanto de las propuestas como de las evaluaciones de las mismas, radica en la necesidad de equipos de grandes prestaciones al igual que gran cantidad de imágenes que conformen los datasets de entrenamiento, no en vano uno de los modelos evaluados en la presente tesis “Facenet”, recurrió al entrenamiento de una red neuronal con un dataset de 3 millones de rostros y 9000 clases, algo que con equipos que regularmente se encuentran en los laboratorios de cada universidad, sería difícil de alcanzar, por ello se analizaron los resultados de cuatro modelos que pueden ejecutarse en computadores de bajas prestaciones y que puedan generar resultados satisfactorios que podrían implementarse en aplicaciones de seguridad y biométricos para reconocimiento de rostros. Los modelos evaluados fueron escogidos en función de tres grandes clasificaciones: Modelos tradicionales, APIS de reconocimiento de objetos fácilmente adaptables a identificación de rostros y un modelo basado en redes neuronales convolucionales. De estos modelos, el modelo basado en redes neuronales convolucionales presentó mejores resultados a la hora de identificar rostros, tanto por su facilidad de implementación como por los resultados obtenidos en los experimentos, mismos que destacaron el uso de un grupo de control para descartar falsos positivos debido a la naturaleza del entrenamiento de los modelos.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

A continuación, se presentan los antecedentes relacionados con el reconocimiento facial y la justificación del estudio propuesto. El reconocimiento facial es una técnica de visión por computador que intenta simular la habilidad humana de detectar y reconocer un rostro basado en la reunión de sus características y posibilidad de compararlo con una base de datos preestablecida.

1.Introducción

1.1. Antecedentes y justificación

Antecedentes

En los últimos años el reconocimiento facial ha cobrado especial importancia sobre otros métodos de análisis de imágenes, esto se debe al desarrollo de la tecnología y aplicaciones relacionadas, así como también al esfuerzo de científicos en este campo por simular la particularidad humana de poder reconocer rostros.

Sin embargo de esto, el reconocimiento facial o su estudio a través del análisis de imágenes en el área de visión por computador, no es una tarea nueva, se tiene conocimiento que los primeros esfuerzos por simular esta característica hasta ahora única del ser humano, aparecen en la década de los 60s a través de Woodrow Wilson Bledsoe quien actualmente es considerado el padre del reconocimiento facial [1]. Bledsoe allanó el camino de las técnicas actuales con la implementación de matrices físicas para la marcación de coordenadas que permitan definir las características del rostro humano. Lo que posteriormente en los 70s se conocería como marcadores faciales.

Posterior a esto, y a finales de los 80s Sirovich y Kirvic [2] emplean técnicas de algebra lineal lo que se conocería como el enfoque *eigenface* para demostrar que bastan 100 características para representar un conjunto de imágenes faciales. Esto fue ampliado por Turk y Pentland [3] lo que se convirtió en la primera forma automatiza de detección de rostros en imágenes.

A partir de estos trabajos, las agencias de gobierno de muchos países empiezan a acumular datasets con el propósito de generar aplicaciones de reconocimiento facial que pudieran ser implementadas tanto en aplicaciones comerciales como en logística de seguridad, para ese entonces se pueden distinguir dos grandes grupos de técnicas: técnicas que en este momento se describen como técnicas tradicionales, eigenfaces, fisherfaces o alternativas a estas como histogramas locales de patrones binarios, y otro grupo de reciente interés, a aquellas técnicas basadas en la implementación de aprendizaje profundo. El aprendizaje profundo aparece como una alternativa actual debido a dos factores: (i) el uso de redes convolucionales y (ii) la habilidad de los equipos actuales de manejar gran cantidad de información para procesamiento [4].

Justificación

Dados los antecedentes mencionados es fácil llegar a pensar que las técnicas de aprendizaje profundo han comenzado a desplazar a las técnicas tradicionales, lo cual no necesariamente se cumple por una diversidad de factores entre ellos la necesidad de encontrar un método que pueda lograr mayor efectividad con menores recursos, incluso resolver los clásicos problemas de iluminación y posicionamiento del rostro. Si bien existen evaluaciones relacionados con el reconocimiento facial [5]–[9], se debe considerar que al momento no existen estudios que hayan hecho un análisis comparativo desde una misma perspectiva a estos los grupos de técnicas analizados en esta tesis.

1.2. Objetivos

1.2.1. Objetivo General

Evaluar distintas propuestas de reconocimiento facial mediante la comparación de técnicas tradicionales y modernas basadas en aprendizaje profundo.

1.2.2. Objetivos Específicos

- Evaluar el rendimiento y aplicación del API de TensorFlow.
- Generar un pipeline de trabajo basado en redes neuronales convolucionales e implementadas en TensorFlow que sirva como base para la definición de reconocimiento facial.
- Evaluar el rendimiento y aplicación del TensorFlow Object Detection API.
- Evaluar una propuesta alternativa al API de Tensor Flow mediante el detector Keras RetinaNet API.
- Comparar las técnicas anteriores con técnicas tradicionales, las cuales serán utilizadas como elemento de control.

1.3. Alcance

El presente trabajo pretende servir como instrumento base para futuros estudios relacionados con el reconocimiento facial, a través de la creación de un pipeline de trabajo basado en redes neuronales.

1.4. Organización del documento

El capítulo 1 presenta la organización del documento de tesis, así como también los elementos constitutivos de la misma; El capítulo 2 contiene los fundamentos teóricos de los modelos examinados mientras que el capítulo 3 contiene la metodología de experimentación y la evaluación de los seis modelos analizados;

y en el Capítulo 4 se presentan los resultados obtenidos. Al final se muestran las conclusiones y las referencias utilizadas en la presente tesis.

CAPÍTULO 2

Hasta hace poco la habilidad de reconocer un rostro era una característica restringida a la capacidad humana; sin embargo, estudios del tema sobre el desarrollo de herramientas automatizadas que permitan al computador adquirir mencionada habilidad datan de los años 60. La diversidad de métodos existentes radica en la implementación de la caracterización del rostro y el tratamiento comparativo del mismo. Al momento existen muchas variantes de implementación de soluciones que van desde sencillos modelos matemáticos hasta la aplicación complejas redes neuronales, esta sección contiene los fundamentos teóricos de los modelos examinados. Es decir, un análisis teórico que va desde el tratamiento de imágenes pasa por la definición de los métodos tradicionales y finaliza con la mención de aquellos métodos que aplican aprendizaje profundo.

2. Marco Teórico

2.1. Procesamiento de Imágenes

Una imagen digital es una representación bidimensional de una imagen como un conjunto de elementos finitos de una matriz, en la que cada elemento se le denomina píxel. Cada píxel representa un valor numérico para definir una escala de gris y color en la imagen. La digitalización entonces es un proceso que intenta aproximar una escena real desde una perspectiva digital.

El procesamiento de imágenes entonces se orienta a desarrollar dos funciones: (i) mejorar la información de interpretación de la imagen y (ii) procesamiento de los datos extraídos de la imagen que permitan a su vez su almacenamiento, transmisión y representación en equipos de manera independiente [10]. En la siguiente sección se desarrolla la definición de imágenes para una mejor comprensión.

2.1.1. Imágenes Digitales

Como ya se mencionó, una imagen es una representación bidimensional de un conjunto finito de valores digitales, mencionados valores digitales se denominan píxeles (Figura 2.1)

La Figura 2.1 Muestra el proceso general de captura de imágenes digitales en la que el objeto del mundo real luego de ser capturado por el dispositivo adopta una representación matricial, en la que cada posición (*x,y*) representa un valor numérico.

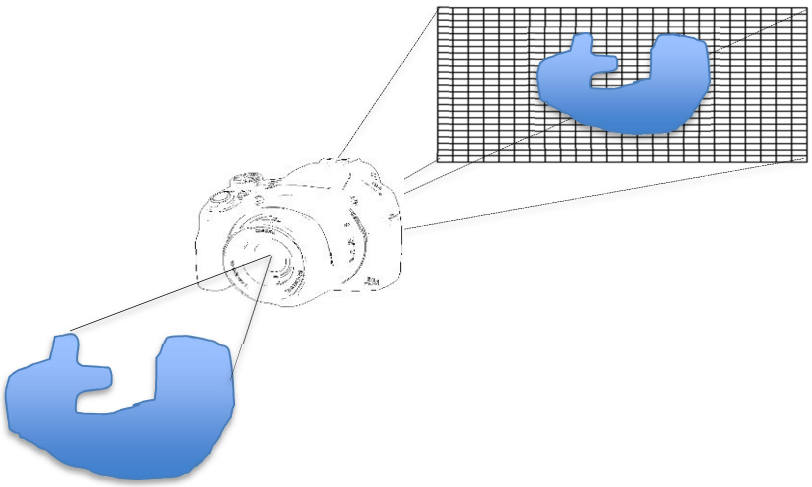


Figura 2.1: Proceso de captura de imágenes a través del uso de una cámara.

Cada valor numérico contiene un valor de intensidad denominado píxel, que representa niveles de grises, colores, opacidades, etc.

Gonzalez y Woods [10] consideran que el procesamiento digital de imágenes contiene un conjunto de tareas claves como lo muestra la Figura 2.2. Estas etapas son presentadas seguidamente.

2.1.2. Etapas del procesamiento de imágenes

Una vez que la captura analógica de la imagen se ha dado (Figura 2.1) el procesamiento de imágenes requiere de un proceso cuyas etapas (Figura 2.2) se listan a continuación:

- Adquisición de imágenes.
- Mejoramiento de la imagen.
- Restauración de la imagen.
- Procesamiento morfológico de la imagen.
- Segmentación.
- Reconocimiento de objetos.
- Representación y descripción de objetos.

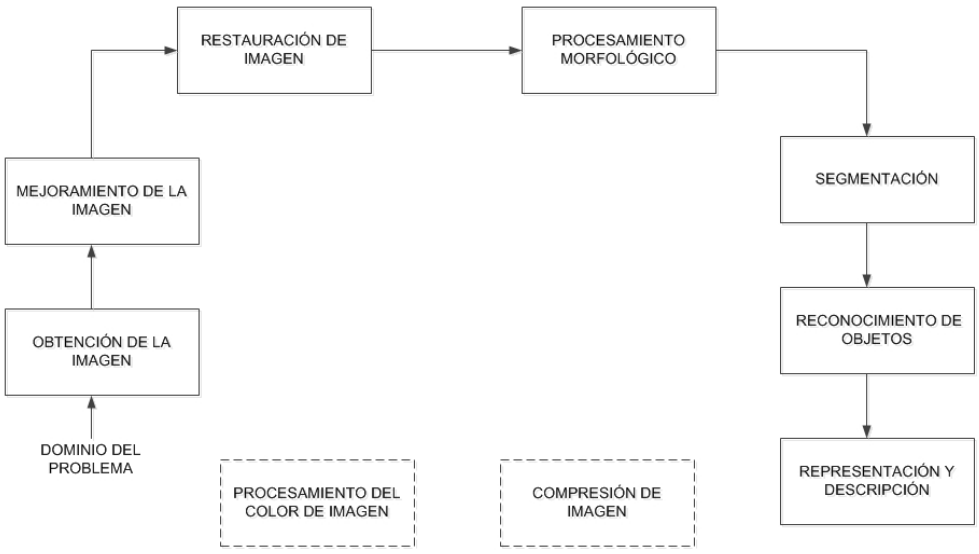


Figura 2.2: Etapas claves en el procesamiento digital de imágenes.

2.2 . Técnicas y modelos de Reconocimiento facial

Actualmente uno de los campos más éxitos del procesamiento digital de imágenes es el campo de reconocimiento facial, campo de especial atención en la mayor parte de aplicaciones (ver Tabla 2.1) que se vienen desarrollando desde cerca de 30 años atrás [8], el objetivo del reconocimiento facial descansa en el hecho de analizar un conjunto de imágenes o videos e identificar los rostros que aparezcan en los objetos de estudio comparándolos con una base de datos de rostros previamente identificados.

Áreas de aplicación	**Aplicaciones específicas**
Entretenimiento	Juegos de Video, realidad virtual, programas de entrenamiento interacción humano-robot, interacción humano-computador.
Tarjetas inteligentes	Licencias de conducir, identificaciones, control de draude.
Seguridad	control parental, acceso a dispositivos personales, seguridad de aplicaciones, seguridad de bases de datos, acceso a internet, registros médicos.
Vigilancia	videos de vigilancia control de cámaras en circuitos cerrados, accesos y seguridades físicas, seguimiento y control.

Tabla 2.1: Áreas de aplicación del reconocimiento facial, gentileza de ACM Computing Surveys, Vol. 35, No. 4, December 2003.

El reconocimiento facial es una técnica biométrica que pretende medir características únicas relacionadas con un rostro a través de medios electrónicos como lo señaló en su momento Woodrow W. Bledsoe [11] quien en los 60s conjuntamente con Helen Chan y Charles Bisson, de Panoramic Research, investigaron la forma en la que algoritmos de computadora puedan reconocer rostros humanos. Según Bledsoe: "*el problema de reconocimiento se ve dificultado por la gran variabilidad en la rotación y la inclinación de la cabeza, la intensidad y el ángulo de iluminación, la expresión facial, el envejecimiento*".

Algunos otros intentos de reconocimiento facial por parte de la máquina han permitido poca o ninguna variabilidad en estas cantidades. El método de correlación (o coincidencia de patrones) de datos ópticos no procesados, que a menudo utilizan algunos investigadores, es seguro que fallará en los casos en que

la variabilidad sea grande. En particular, la correlación es muy baja entre dos imágenes de la misma persona con dos rotaciones de cabeza diferentes.

Desde la perspectiva de Woodrow Bledsoe, una primera solución a este problema fue denominada problema hombre-máquina debido a que el ser humano extraía las coordenadas de un conjunto de características en la imagen capturada, las cuales serían utilizadas por el computador para establecer un reconocimiento, para ello se hacía uso de un GRAFACON o Tabla RAND. El personal a cargo debía extraer las coordenadas de características determinantes del rostro, así como también las del centro de sus pupilas, esquinas internas de ojos, esquinas externas de ojos y así en lo sucesivo. Con esas coordenadas un listado de al menos 20 distancias tales como ancho de bocas y distancia entre ojos (pupila a pupila) debían ser calculadas, se procesaron cerca de 40 imágenes por hora, lo que permitía generar una base de datos en la que el nombre de la persona en la imagen estaba asociado con el listado de distancias calculadas y almacenadas dentro de un computador. Posteriormente en la fase de reconocimiento, el conjunto de distancias era comparada con la correspondiente distancia de cada fotografía, generando una distancia entre la imagen y el registro en la base de datos, en busca de los registros de menor distancia en término de cercanía.

Bledsoe consideraba que la explicación precedente era susceptible de fallas debido a que era complejo encontrar dos imágenes que mantengan los mismos parámetros en cuanto a rotación de cabeza, inclinación y escala, esta última relacionada con los parámetros de captura de las cámaras. Por lo que era obligatorio desarrollar una normalización de las distancias que representaban el rostro en una posición central. Para lograr esta normalización era necesario que los programas de computadora intentaran determinar la inclinación (*tilt*), desviación (*lean*) y rotación (*rotation*) de un rostro, posterior a ello mediante la utilización de los ángulos determinados, la computadora deshacía el efecto de estas transformaciones en las distancias calculadas. Para calcular estos ángulos, la computadora debe conocer la geometría tridimensional de la cabeza. Debido a que las cabezas reales no estaban disponibles, Bledsoe (1964) utilizó una cabeza estándar derivada de mediciones en siete cabezas.

Trabajos similares de reconocimiento continuaron en el Stanford Research Institute, principalmente desarrollados por Peter Hart. En los experimentos realizados sobre una base de datos de alrededor de 2000 fotografías el computador superó sistemáticamente a los humanos en tareas de reconocimiento [12] [13].

Actualmente, investigadores de Google publicaron un artículo, Facenet [9], el cual utiliza los conceptos de redes neurales convolucionales considerando los píxeles como características de la imagen, en lugar de extraerlas manualmente. De acuerdo con las mediciones realizadas por los mismos investigadores, se pudo obtener una precisión del 99.63% sobre el dataset "LWF[1]" [15].

2.2.1. Proceso simplificado en el reconocimiento facial

Una forma simple de tratar de entender el proceso involucrado en el reconocimiento de rostros por parte de un dispositivo electrónico radica en que la aplicación sea capaz de trabajar en tres etapas claramente definidas, una vez que se ha logrado capturar la imagen de un sujeto digitalmente:

- Detección del rostro
- Extracción de características
- Reconocimiento del rostro

Detección del rostro

El enfoque principal de este paso es determinar si los objetos que aparecen en una imagen digital corresponden a un rostro, lo que implica también reconocer en dónde se encuentra ubicado dicho rostro, esto implica resolver un problema de alineación, debido a que los rostros en las fotografías pueden o no estar alienados es necesario recurrir a un método que permita una estandarización de la ubicación.

[1] Un conjunto de imágenes denominadas *Labeled Faces in the Wild* que contienen más de 13000 fotografías de rostros coleccionados en la web.

Según Yang, D. J. Kriegman, and N. Ahuja [10], se define como detección de rostros al proceso en el que dada una imagen cualquiera, se puede determinar la existencia o no de una faz humana, de encontrarse, el proceso debe retornar la localización exacta de esta faz en la imagen. Los retos en el proceso de detección de caras se relacionan con los siguientes factores:

- Postura: tanto una cara como la posición de la cámara influyen en la captura realizada, que puede generar mayor o menor dificultad al momento de encontrar y ubicar una faz dentro de una imagen.
- Componentes estructurales: la presencia de rasgos generales en un rostro tales como lentes, gafas, bigotes.
- Expresiones Faciales.
- Oclusiones.
- Orientación de la imagen.
- Condición de la imagen.

Varias décadas atrás ya se había analizado el problema de localizar los limites faciales en determinadas fotografías, los cuáles se concentraban en determinar la ubicación de la cabeza y su relación con el background existente debido a que muchos algoritmos basados en métodos globales como los de la transformada de Hough no eran concluyentes [11], en las siguiente secciones se analizarán dos métodos específicos de detección de rostros al igual que sus pro y contras.

Extracción de características

Después que el rostro ha sido detectado, se procede a extraer ciertas zonas del rostro (*patches*) la diversidad de métodos existentes para el reconocimiento de rostros se enfoca directamente en las zonas extraídas como elementos de comparación, lo cual por la cantidad de información en píxeles genera una gran desventaja e información de procesamiento, lo que obliga a buscar métodos que permitan la reducción de la dimensionalidad de las características a encontrar. Una segunda desventaja del uso de estos patches, es la diferencia que existe en las capturas de imágenes relacionadas con diferencia de alineación de cámaras, diferencias entre expresiones faciales, condiciones de iluminación entre las más

importantes, las que conllevan a posibles oclusiones que obligan a reducir información, reducir dimensionalidad, trabajar con características predominantes solamente y eliminación de ruido. Luego de este proceso los patches son transformados en arreglos de dimensiones fijas.

Reconocimiento del rostro

El proceso de transformación de ciertas partes del rostro en arreglos de dimensiones fijas permite obtener una representación facial y reconocer rostros de manera automatizada. Este paso requiere de un conjunto de imágenes que juntamente con las características extraídas formarán una base de datos, el siguiente paso tiene que ver con el ingreso o captura de una imagen que contenga una cara, extraer sus características dominantes y compararlas contra la base de datos almacenada, este proceso se denomina identificación del rostro o reconocimiento facial.

Se debe distinguir la identificación de la validación. La identificación tiene que ver con la probabilidad de que la imagen de un rostro pertenezca a determinada persona, mientras que la validación se relaciona con el proceso en el que, dada una identificación, se desea que el sistema empleado indique que tan cierta o falsa fue la suposición de la identificación. La Figura 2.3 muestra el proceso simplificado desarrollado a partir de [18] de lo que representa la identificación de rostros.

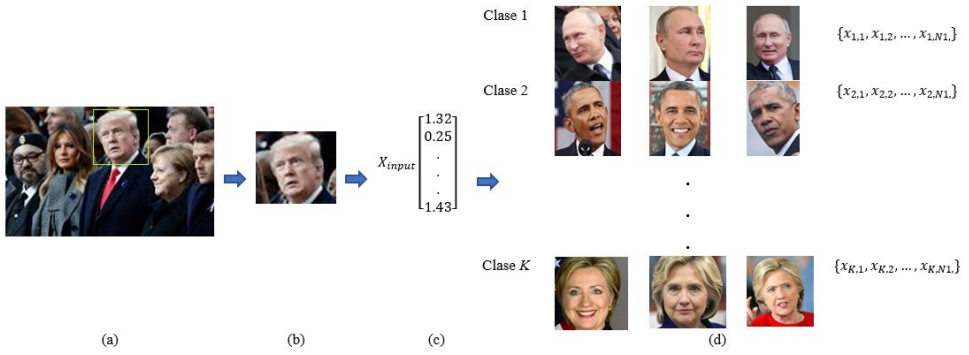


Figura 2.3: Proceso simplificado de identificación de rostros, en la que (a) Inicia el proceso de reconocimiento con la detección del rostro, para este caso se enmarca en el recuadro verde, (b) obtención del parche y alineación del rostro, (c) detección de características y generación del vector de características, (d) proceso de comparación del vector de entrada con la base de datos de vectores características para determinar la coincidencia con mayor probabilidad.

Todo proceso de reconocimiento involucra un conjunto de datos para la construcción de categorías y la comparación de similaridades entre los datos de prueba y el conjunto de datos clasificados. Como el reconocimiento facial es una derivación específica del reconocimiento de patrones la notación empleada para esta técnica al igual que en *Machine Learning,* se basa en representaciones matriciales y vectoriales, diferenciándose el aprendizaje supervisado $\{X_n^T, Y_n^T\}$ (cada muestra del data set de entrenamiento) del aprendizaje no supervisado por la notación simple $\{X_n^T\}$ [12], la notación matricial según la literatura examinada utiliza mayúsculas mientras que la notación vectorial minúsculas, la consulta de entrada es denotada como X sin el indicador T que representa el data set de entrenamiento.

2.2.2. Técnicas tradicionales

En esta sección se revisarán 3 de las técnicas más utilizadas en reconocimiento facial, las cuales didácticamente brindan un punto de partida en el análisis matemático del estudio de rostros mediante visión por computador. Estas técnicas fueron evaluadas haciendo uso de sus respectivas implementaciones en OpenCV.

OpenCV es una librería de visión por computador iniciada por INTEL en 1999 que se enfoca en procesamiento de imágenes e incluye implementaciones de algoritmos de visión por computador, disponibles para C, C++, Python y últimamente Android, está liberada bajo licencia BSD (*Berkeley Software Distribution*) por lo que puede ser utilizada en productos tanto académicos como comerciales siempre y cuando se garantice la referencia al productor original. En cuanto a reconocimiento facial, OpenCV trabaja actualmente con tres algoritmos:

- *Eigenfaces*
- *FisherFaces*
- Histogramas de patrones locales binarios o LBPH (*Local Binary Patterns Histograms*)

Como se mencionó en la sección anterior, el proceso de reconocimiento de rostros es una tarea sencilla para los seres humanos, pero ¿qué tan complejo es este

proceso para el computador?, para poder responder a este interrogante se debe analizar que el proceso de reconocimiento abarca el extraer características significativas de una imagen (rostro), ubicarlas en una representación útil y realizar algún tipo de clasificación en ellas.

El reconocimiento facial basado en las características geométricas de un rostro es probablemente el enfoque más intuitivo para el reconocimiento facial. Uno de los primeros sistemas automáticos de reconocimiento facial fue descrito por Takeo Kanade [13] en el que se utilizaron puntos marcadores (posición de los ojos, oídos, nariz, ...) para construir un vector de características (distancia entre los puntos, ángulo entre ellos, ...). El reconocimiento se realizó calculando la distancia euclidiana entre los vectores de características de una sonda y la imagen de referencia. Aunque el método propuesto por Kanade puede considerarse robusto contra los cambios en la iluminación, presenta un inconveniente debido a que el registro preciso de los puntos marcadores es complejo, incluso con los algoritmos más modernos. Otros trabajos sobre el reconocimiento facial geométrico fueron desarrollados y presentados por Poggio y Brunelli [14] en 1992, en los que se utilizó un vector de características de 22 dimensiones, pese a esto los experimentos en grandes conjuntos de datos han demostrado que solo las características geométricas no contienen suficiente información para el reconocimiento facial.

Es por ello que aparecen alternativas como Eigenfaces [15] cuyo método se basa en una propuesta holística para el reconocimiento de rostros, en la que la imagen de una cara se encuentra entre la imagen de alta dimensión y se encuentra una representación de dimensión inferior, punto donde la clasificación es relativamente fácil. El subespacio de dimensión inferior se encuentra mediante el Análisis de Componentes Principales (PCA), que identifica los ejes con la varianza máxima. Desafortunadamente los ejes con varianza máxima no contienen necesariamente ninguna información discriminatoria, por lo que en ocasiones una clasificación se vuelve imposible. Por lo tanto, en 1997 se aplicó una proyección específica de clase con un Análisis Discriminante Lineal al reconocimiento facial como se lo

puede verificar en [16]. La idea básica es minimizar la varianza dentro de una clase, mientras se maximiza la varianza entre las clases al mismo tiempo.

Propuestas más recientes incluyen varios métodos para una extracción de características locales. Los cuales, para evitar la alta dimensionalidad de los datos de entrada, solo se describen las regiones locales de una imagen, las características extraídas son más robustas contra la oclusión parcial, la iluminación y el tamaño pequeño de la muestra. Los algoritmos utilizados para la extracción de una característica local son Gabor Wavelets [17], Discrete Cosinus Transform [18] y Local Binary Patterns [19].

Eigenfaces

Uno de los principales problemas con imágenes es la alta-dimensionalidad de estas. Las imágenes bidimensionales *p x q* (escala de grises) abarcan un espacio vectorial *m = pq-dimensiones,* de tal forma que una imagen de 100x100 formará un espacio vectorial de 10.000 elementos. De los cuales no todos los elementos son necesarios para el estudio, Karl Pearson en 1901 y Harold Hotelling en 1933, realizaron sendos estudios sobre Análisis de Componentes Principales o PCA de manera de convertir un conjunto de variables posiblemente correlacionadas en un conjunto más pequeño de variables no correlacionadas. La idea es que un conjunto de datos de alta dimensión a menudo se describe mediante variables correlacionadas y, por lo tanto, solo unas pocas dimensiones significativas representan la mayor parte de la información. El método PCA encuentra las direcciones con la mayor variación en los datos, llamados componentes principales.

Fisherfaces

Si bien mediante el uso de PCA que es el núcleo del método Eigenfaces, se encuentra una combinación lineal de características que maximiza la varianza total en los datos y es una forma poderosa de representarlos, existe un problema, este método no considera clase alguna y, por lo tanto, pierde mucha información discriminatoria cuando se ejecutan los componentes. Este problema se maximiza

con imágenes en la que las zonas características están marcadas por zonas de exposición a luz. Los componentes identificados por un PCA no necesariamente contienen información discriminatoria, por lo que las muestras proyectadas se reconocen juntas lo que hace imposible una clasificación

Como alternativa, el Análisis Discriminante Lineal realiza una reducción de dimensionalidad específica de clase y fue inventado por el estadístico Sir R. A. Fisher, y publicado en el año 1936 (*The use of multiple measurements in taxonomic problems).* Con el fin de encontrar la combinación de características que separa mejor entre las clases, el Análisis Discriminante Lineal maximiza la proporción de dispersión entre clases y dentro de las clases, en lugar de maximizar la dispersión global. Es decir, las mismas clases deben agruparse estrechamente, mientras que las diferentes clases están lo más lejos posible entre sí en la representación de la dimensión inferior.

Histogramas de patrones locales binarios

Las propuestas anteriores tratan a los datos (imágenes) como vectores a los que se les reduce la dimensionalidad, con la idea de mantener información de utilidad, por una parte, el enfoque *eigenface* maximiza la dispersión total, lo que puede llevar a problemas si la varianza es generada por una fuente externa, porque los componentes con una varianza máxima en todas las clases no son necesariamente útiles para la clasificación. Por otra parte, al aplicar un análisis de discriminante lineal se conserva cierta información discriminativa mediante el método de *fisherfaces*. Esto nos permite entender la problemática de tener diferentes condiciones de luz por cada rostro de cada persona o en su defecto diferentes condiciones gestuales en cada rostro, la idea general con estos métodos es que a mayor cantidad de información debería mejorar el índice de reconocimiento, sin embargo, de la literatura investigada, para obtener buenos rangos de reconocimiento se necesitan al menos 8 imágenes con +/- 1 imagen por cada persona, en este caso *fisherfaces* falla. Un análisis detallado de estas mediciones se pueden encontrar en [20]. La mayor parte de investigaciones se concentran en extraer características locales de las imágenes. Sin embargo, se debe entender que la idea es no ver la imagen completa como un vector de alta

dimensión, sino describir solo las características locales e importantes de un objeto. Esto representará una imagen de baja dimensionalidad implícitamente. El problema aquí es que los métodos antes señalados no son inmunes a problemas de iluminación o variaciones de posición. Al igual que SIFT [21], la metodología de patrones binarios locales tiene sus raíces en el análisis de textura 2D. La idea básica de los patrones binarios locales es resumir la estructura local en una imagen comparando cada píxel con su vecindario. Toma un píxel como centro y umbral contra sus vecinos. Si la intensidad del píxel central es mayor-igual que su vecino, denótelo con 1 y 0 si no. Terminará con un número binario para cada píxel, al igual que 11001111. Por lo tanto, con 8 píxeles circundantes, obtendrá 2 ^ 8 combinaciones posibles, denominadas Patrones Binarios Locales (LBP) o, en ocasiones, denominados simplemente códigos LBP. El primer operador de LBP descrito en la literatura en realidad usó un vecindario fijo de 3 x 3 como lo muestra la Figura 2.4.

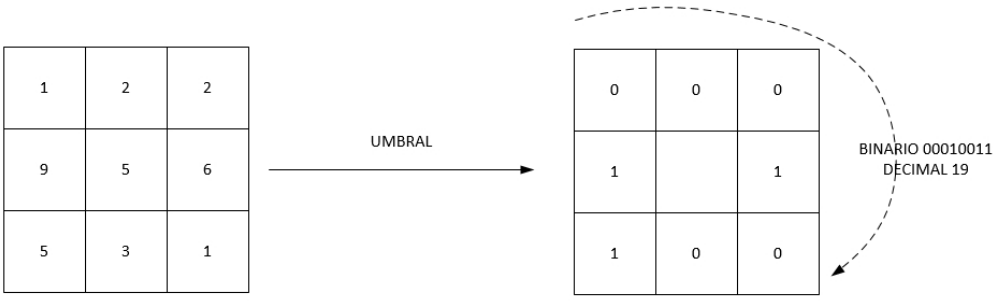


Figura 2.4: Ventana de 3x3 para operadores binarios de patrones.

2.2.3. Técnicas basadas en aprendizaje profundo

Considerando que un proceso de identificación de un rostro se resume en: detección del rostro y su enmarcado, obtención del parche con el marco adecuado, alineación del rostro, detección de características y generación del vector de características, es posible la adecuación de las modernas técnicas de detección de objetos mediante aprendizaje profundo al reconocimiento facial.

Esto permite que enfoques más apropiados para el reconocimiento de rostros sean las extensiones de los clasificadores de imágenes. Para ello se debe comprender que el aprendizaje profundo [9] (analizado con más detalle en las siguientes secciones), aplica múltiples capas de procesamiento para el aprendizaje de representaciones de datos con múltiples niveles de extracción de características. Técnica que desde este punto de vista desde el 2014 ha cambiado el panorama general de las técnicas de reconocimiento facial [27] debido principalmente al uso de redes neuronales convolucionales, una red convolucional es un tipo de red neuronal artificial con una variación del concepto de perceptrón multicapa y basada en matrices bidimensionales, lo que facilita su uso de manera efectiva en clasificación y segmentación de imágenes. Estas características las hacen especialmente adecuadas para el uso en la detección de objetos, especialmente para la detección y reconocimiento de rostros.

Bajo este contexto el investigador tiene dos opciones: (i) generar una arquitectura que permita entrenar de manera adecuada una red para el reconocimiento de rostros o (ii) utilizar una arquitectura pre-entrenada para la extracción de características y que estás puedan utilizarse como insumo en el desarrollo propio de aplicaciones para reconocimiento facial, esta segunda opción presenta algunas ventajas con respecto a procesos en los cuáles no se está dispuesto a incurrir en altos costos computacionales.

En esta sección se explicará brevemente lo que es una arquitectura y modelo pre-entrenado, clasificadores y la diferencia entre generar aprendizaje desde cero y la obtención de transferencia de aprendizaje.

Introducción a las arquitecturas de redes profundas

De acuerdo con Xu y Vidal en [28] los modelos pre-entrenados que al momento sobresalen son:

- Redes VGG.
- *Resnet.*
- *Inception.*

- *Xception.*
- *MobilNet*

Las redes VGG por su significado inglés Visual Geometry Group[2], mantienen el arquetipo clásico de una red convolucional, esto es, convoluciones, agrupamientos, capas de activación y capas de clasificación totalmente conectadas.

Resnet surge como una alternativa para la implementación de redes que perdían efectividad a medida que aumentaban su profundidad, esto se debía a lo que se denominaba el problema de fuga del gradiente. *Resnet* resuelve este problema de ineficiencia de capas con lo que se denominó: "conexión de acceso directo de identidad" que omite una o más capas (ver detalle en la Figura 2.9).

Inception, a diferencia de *resnet*, que se concentra en la profundidad de las redes, se enfoca en la extensión de una red, es decir disminuir el costo computacional de procesar redes de gran tamaño, el objetivo era aumentar el tamaño de las redes neuronales sin afectar o exceder la capacidad computacional de los equipos de procesamiento. Esto se logró con dos elementos destacables: el primero en la capa "*Inception module*" que consiste en una convolución de 5×5, una 3×3 y *max-pool*, en la que la selección de las características más importantes se deja a la capa siguiente. Y el segundo, con el uso de circunvoluciones, que permitía resolver la complejidad computacional, mediante la implementación de convoluciones 1×1 para "filtrar" la profundidad de las salidas. Estas circunvoluciones tienen en cuenta un solo valor a la vez, por medio de múltiples canales, al mismo tiempo que pueden extraer información espacial para comprimirla en una dimensión más pequeña.

Xception, a diferencia de las redes convolucionales, presenta una variación: en lugar de clasificar los datos de entrada en múltiples grupos comprimidos, los mapea separadamente para ejecutar una convolución 1×1 en profundidad y poder capturar correlaciones *cross-channel*. Lo que se conoce como "*depthwise*

[2] Desarrollado por la University de Oxford para el ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2014.

separable convolution" que es una convolución espacial (*depthwise convolution*) realizada de manera separada para cada canal, seguida de una convección 1×1 (*pointwise convolution*) entre canales. Por lo tanto, es una búsqueda de correlaciones primero en un espacio bidimensional y luego en un espacio unidimensional. Esta asignación 2D + 1D es más fácil de aprender que una completamente 3D.

Mobilnet, es una arquitectura para aplicaciones de visión basadas en dispositivos móviles y embebidas para componentes computacionales de bajo rendimiento propuesto por Google. Se basa en el uso de "*depthwise separable convolutions*" para la reducción del número de parámetros al comparar la red con las convoluciones normales en una misma profundidad [29].

Modelos Pre-entrenados

Gracias a las publicaciones y liberación de Google con respecto a APIs para el reconocimiento de objetos, se pueden encontrar modelos pre-entrenados con distintos valores de efectividad, entre los más comunes se encuentran:

- SSD con *Inception* V2.
- SSD *Multibox (Single Shot Multibox Detector) con MobileNets*.
- R-FCN (*rety fully connected*) con *Resnet 101.*
- *Faster* R-CNN con *resnet* 101.
- *Faster* R-CNN con Inception Resnet v2

El SSD (*Single Shot Multibox Detector*) es un modelo para la detección de objetos [30] , el R-FCN es un modelo de red basada en regiones para el mejoramiento de la precisión en la detección de objetos [31], al igual que R-CNN entendida como una red neuronal convolucional o CNN (*Convolutional neural network*) por sus siglas en inglés de regiones para la extracción de características.

Clasificadores

En los últimos años y de acuerdo con la literatura revisada, existe aún cierta discrepancia en la forma como las arquitecturas de clasificación deben ser

identificadas (arquitecturas, redes, modelos, clasificadores, etc). Esto es, por la forma como indistintamente muchos investigadores abordan el problema de trabajar con *deep learning*, lo que si está claro es que existen dos formas de entrenar una red: la primera desde cero (*from scratch*), y la segunda mediante transferencia de aprendizaje (*transfer learning),* el presente trabajo se enfoca en esta segunda opción debido a que permite la reutilización de modelos pre-entrenados para la resolución de nuevos problemas. En este sentido uno de los elementos importantes para lograr el objetivo de la presente investigación, son los modelos y arquitecturas seleccionadas con información de base, lo que en el presente trabajo se ha denominado *clasificadores,* si bien el análisis de cuál es el mejor clasificador está fuera de esta tesis, se seleccionó la base de *inception network* y su modelo incremental, debido a que muchos de los algoritmos de reconocimiento facial los utilizan (o a su variación), permitiendo un análisis más equilibrado y que se centraría en la diferenciación de librerías más que en los modelos utilizados.

Inception network

Se entiende como clasificador la estructura de red que permitirá discriminar un objeto de otro, la “Inception network” fue un importante hito en el desarrollo de clasificadores CNN, antes de la *Inception network* las CNN más populares simplemente apilaban capas de convolución cada vez más profundas, con la esperanza de obtener un mejor rendimiento. Existen algunas versiones de este tipo de red, las más comunes son:

- Inception v1.
- Inception v2 and Inception v3.
- Inception v4 and Inception-ResNet.

Inception v1

Trata de resolver el problema de que las partes predominantes en una imagen pueden tener mucha variación en cuanto a tamaño y forma de imagen a imagen, lo que implica que la elección del tamaño correcto del kernel para la operación de

convolución, sea una tarea complicada, un kernel de gran tamaño es adecuado para información globalmente distribuida mientras que un kernel pequeño es adecuado para información distribuida localmente. Las redes profundas son propensas al sobreajuste (*overfitting*). También es difícil pasar las actualizaciones de gradiente a través de TFODAPI la red. Y el apilamiento ingenuo de grandes operaciones de convolución es computacionalmente costoso. La solución planteada en *Inception* v1 es tener filtros de distintos tamaños que operen al mismo nivel, la red esencialmente se convertirá en una red más amplia que profunda. La Figura 2.5 muestra una versión “sencilla” del módulo *Inception* que realiza una convolución sobre una entrada con tres tamaños diferentes de filtros como 1x1, 3x3, 5x5. Además, también se realiza la agrupación máxima. Las salidas se concatenan y se envían al siguiente módulo de inicio [52].

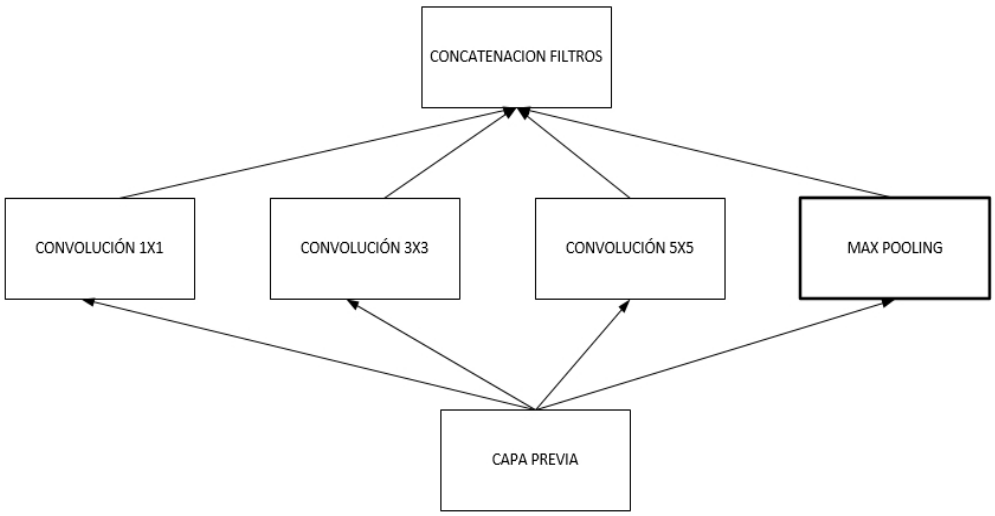


Figura 2.5: Inception naive desde "goin deeper convolution".

De lo explicado anteriormente, las redes neuronales profundas son computacionalmente costosas. Para reducir su complejidad, los autores limitan el número de canales de entrada agregando una convolución extra 1x1 antes de las convoluciones 3x3 y 5x5. Esto puede parecer contrario a la intuición, las convoluciones 1x1 son mucho más baratas que las convoluciones 5x5, eso sin

mencionar que el número reducido de canales de entrada también ayuda. Se debe tomar en cuenta que, sin embargo, la convolución 1x1 se introduce después de la capa de agrupación máxima, en lugar de antes.

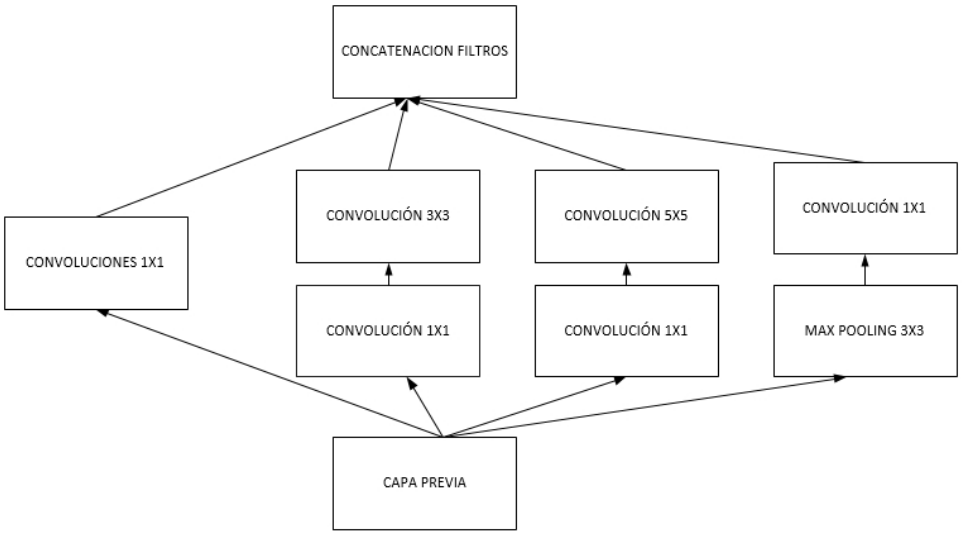


Figura 2.6: Módulo Inception con reducción de dimensionalidad.

Usando el módulo de inicio de dimensión reducida (Figura 2.6), se construyó una arquitectura de red neuronal que posteriormente se conocería como GoogLeNet (*Inception* v1) [52].

GoogLeNet tiene 9 de estos módulos de inicio apilados linealmente, 22 capas de profundidad (27 si se incluyen las capas agrupadas). Utiliza la agrupación promedio global al final del último módulo inicial. Se trata de un clasificador bastante profundo. Al igual que con cualquier red muy profunda, está sujeta al problema gradiente de desaparición (*vanishing gradient problem*).

Para evitar que la parte media de la red se "extinga", los autores introdujeron dos clasificadores auxiliares. Básicamente, aplicaron la función *softmax* a las salidas de dos de los módulos iniciales y calcularon una pérdida auxiliar (*auxiliary loss*) sobre las mismas etiquetas. La función de pérdida total (*total loss function*) es una suma ponderada de la pérdida auxiliar (*auxiliary loss*) y la pérdida real (*real*

loss). El valor de peso utilizado en “Going deeper with convolutions” [52] fue de 0,3 para cada pérdida auxiliar, una captura de ejemplo lo muestra la Figura 2.7.

```
# The total loss used by the inception net during training.
total_loss = real_loss + 0.3 * aux_loss_1 + 0.3 * aux_loss_2
```

Figura 2.7: Captura de los datos de salida presentado por C. Szegedy, W. Liu, P. Sermanet et al. En “Going to deeper convolutions”

De acuerdo con lo presentado por los autores, la pérdida auxiliar se utiliza específicamente para propósitos de entrenamiento y se ignora durante la inferencia.

Inception v2

Tanto el modelo de clasificador *Inception V2* e *Inception V3* fueron introducidos en el paper “*Rethinking the Inception Architecture for Computer Vision*” [53]. La idea principal de los autores fue presentar un número de actualizaciones que mejorasen la precisión y una reducción de la complejidad computacional. En este tipo de modelos se pudo apreciar la existencia de lo que se denomina “Cuello de botella representativo” o “*representational bottleneck*“, la idea es reducir este cuello de botella considerando que, las redes neuronales funcionan mejor cuando las circunvoluciones no alteran drásticamente las dimensiones de la entrada. Reducir demasiado las dimensiones puede causar la pérdida de información, lo que se conoce como un "cuello de botella representativo"

Usando métodos inteligentes de factorización, las convoluciones pueden hacerse más eficientes en términos de complejidad computacional.

Para solucionar este problema los autores sugieren factorizar la convolución 5x5 a dos operaciones de convolución 3x3 para mejorar la velocidad de cálculo. Aunque esto puede parecer contrario a la intuición, pero de acuerdo a C. Szegedy, W.Liu, P. Sermanet et al [53], una convolución 5x5 es 2.78 veces más cara que una convolución 3x3. Por lo tanto, el hecho de apilar dos convoluciones 3x3 conduce a un aumento en el rendimiento como lo muestra la Figura 2.8.

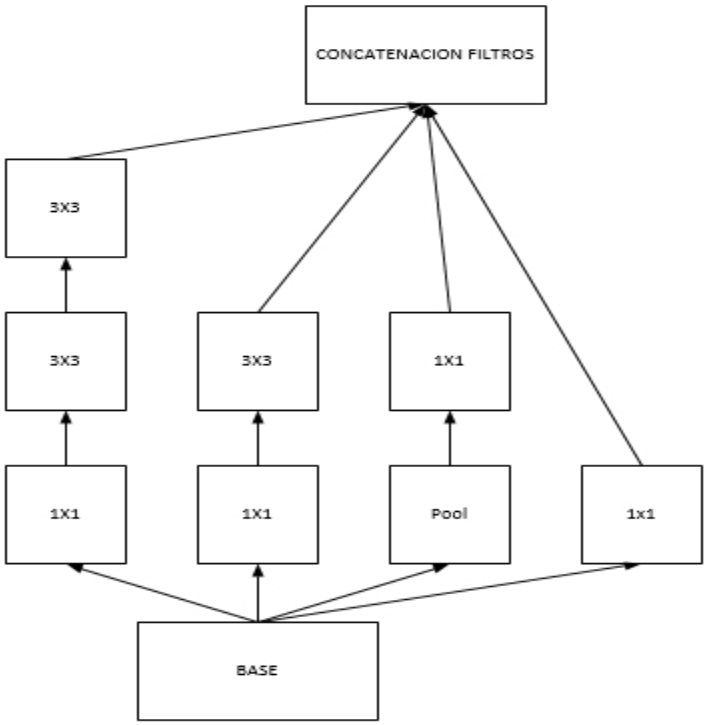


Figura 2.8: La convolución 5x5 más a la izquierda del antiguo módulo de inicio, ahora se representa como dos convoluciones 3x3. Fuente “Rethinking the Inception Architecture for Computer Vision”.

Adicionalmente, se factorizan las convoluciones del tamaño de filtro *nxn* a una combinación de convoluciones *1xn* y *nx1*. Esto quiere decir una convolución *3x3* es equivalente a realizar primero una convolución *1x3* y luego realizar una convolución *3x1* en salida. De acuerdo a la literatura consultada los autores descubrieron que este método es un 33% más barato que la única convolución de *3x3.*

Inception V3

De acuerdo a la literatura relacionada, los autores (en los casos predecesores) determinaron que los clasificadores auxiliares no contribuían hasta el final del proceso de entrenamiento, cuando la precisión estaba cerca de la saturación Argumentaron que funcionan como regularizaciones, especialmente si tienen

operaciones de BatchNorm[3] [51] o Dropout[4] [54]. Por lo que proponen mejorar el Inception v2 con TFODAPIs las mejoras de este, incluidas:

- RMSProp *Optimizer*: debido a los problemas de convergencia en los rangos de aprendizaje *Stochastic Gradient Descent* (SGD) o Gradiente descendiente estocástico, el RMSProp al igual otros métodos intentan sobrellevar este problema ajustando automáticamente el tamaño del paso para que se encuentre en la misma escala que los gradientes; esto es, a medida que el gradiente promedio se reduce, el coeficiente en la actualización de SGD se hace más grande para compensar.
- Convoluciones 7x7 optimizadas
- Uso de *BatchNorm* en Clasificadores auxiliares
- *Label Smoothing*: Un componente de regularización añadido a la fórmula de pérdida (*loss)* que previene que la red confíe en determinada clase

Inception V4

Tanto *Inception V4* como *Inception-ResNet* se introducen en el paper "*Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*" [55] aquí se introdujo lo que se denominaría "Bloques de reducción" especializados, que se utilizan para cambiar el ancho y la altura de la cuadrícula. Las versiones anteriores no tenían explícitamente bloques de reducción, pero se implementó la funcionalidad.

Inception-ResNet V1 y V2

Basada en el rendimiento de ResNet [56] fue propuesta como un hibrido para el modelo Inception, actualmente existen dos versiones V1 que básicamente tiene un costo computacional similar a Inception V3 mientras que V2 tiene un costo computacional similar a Inception V4.

[3] Normalización en lotes (*Batch Normalization*) es una técnica para mejorar el rendimiento y la estabilidad de redes neuronales artificiales, introducida en el 2015.
[4] El Dropout es una técnica de regularización que permite la reducción del overfitting.

ResNet

De acuerdo a lo consultado y en base al teorema de aproximación universal [57] que indica:

> *El teorema de Aproximación Universal establece que una sola capa intermedia es suficiente para aproximar, con una precisión arbitraria, cualquier función con un número finito de discontinuidades, siempre y cuando las funciones de activación de las neuronas ocultas sean no lineales (Hornik)*

Entonces, dada una capacidad suficiente, se sabe que una red de avance con una sola capa es suficiente para representar cualquier función. Sin embargo, la capa puede ser masiva y la red es propensa a un ajuste excesivo de los datos. Debido a esto se tiende a pensar que las redes obligatoriamente deben profundizar.

Considerando esto y considerando que toda red CNN desde AlexNet[5] [58] es cada vez más profunda (5 capas convolucionales), la red VGG (19 capas convolucionales) y GoogleNet (Inception V1 con 22 capas respectivamente). Sin embargo, la profundización de una red no se logra por el apilamiento de capas. Para entender este concepto, se debe entender que una red profunda es difícil de entrenar por la optimización del gradiente descendiente, que al tratarse de una retro-propagación este gradiente puede resultar infinitamente pequeño, el resultado una red profunda degradada.

Previa a la aparición de ResNet [56], el problema del desvanecimiento del gradiente era tratado mediante la adición de una pérdida auxiliar en una capa intermedia como supervisión extra, aunque en si no solucionaba el problema de la degradación. Por ello la idea central de ResNet es introducir una “conexión de acceso directo a la identidad” que permite el salto entre una o más capas, como lo muestra la Figura 2.9.

[5] AlexNet se considera una de las CNN mejor rendimiento, siendo su artículo, uno de los mayormente citados hasta la actualidad.

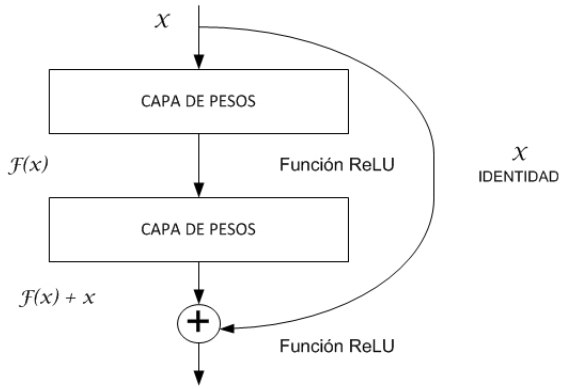


Figura 2.9: Bloque de construcción de aprendizaje residual.

En este sentido K. He, X. Zhang, S. Ren et al. Indican que la apelación de capas no debería degradar el rendimiento de la red, debido a que ya que se podrían apilar las asignaciones de identidad (capa que no hace nada) en la red actual, y la arquitectura resultante funcionaría igual. Por tanto, un modelo más profundo no debe producir un error de entrenamiento más alto que sus contrapartes menos profundas. Los mismos autores plantean la hipótesis de que dejar que las capas apiladas se ajusten a un mapeo residual es más fácil que permitir que se ajusten directamente al mapeo subyacente deseado. Por aquello, ResNet no fue la primera en utilizar conexiones de acceso directo. De la literatura explorada, se conoce que *Highway Network* [59] introdujo conexiones de acceso directo cerradas. Estas puertas parametrizadas controlan cuánta información se permite que fluya a través del acceso directo. Se puede encontrar una idea similar en la celda de memoria corta a largo plazo (LSTM) [60], en la que hay una puerta olvidada parametrizada que controla la cantidad de información que fluirá al siguiente paso de tiempo. Por lo tanto, muchos investigadores consideran que *ResNet* es un caso especial de *Highway Network.*

Contrario a esto, los experimentos y resultados obtenidos indican que *Highway Network* no supera a *ResNet*.

En definitiva, *ResNet* refina el bloque residual y propone una variante de activación previa del bloque residual, en la que los gradientes pueden fluir a través de las

conexiones de acceso directo a cualquier otra capa anterior sin problema. Se demostró entonces con experimentos que ahora pueden entrenar una *ResNet* profunda de 1001 capas para superar a sus contrapartes menos profundas. Debido a sus resultados convincentes, *ResNet* se convirtió rápidamente en una de las arquitecturas más populares en varias tareas de visión artificial.

Variantes de ResNet

- *ResNeXt*
- Cnn densamente conectado
- Red profunda con Profundidad Estocástica

2.3. Aprendizaje profundo (deep learning)

De acuerdo a Yann LeCun, Yoshua Bengio y Geofrey Hinton en el 2015 [23], el Deep Learning se ha convertido en una técnica que permite que los modelos computacionales compuestos de múltiples capas de procesamiento puedan "aprender" representaciones de datos con múltiples niveles de abstracción, como técnica ha permitido mejorar significativamente el estado del arte de reconocimiento visual de objetos por lo que se plantea su estudio en la presente investigación.

Deep Learning permite descubrir complejas estructuras en grandes volúmenes de datos mediante la utilización del algoritmo "*backpropagation*" o propagación en reversa por su traducción literal en español, algoritmo que indica cómo un computador debería cambiar sus parámetros internos los cuales son utilizados para procesar la representación en cada capa desde su representación en un capa previa. Las redes convolucionales profundas han traído avances en el procesamiento de imágenes, video, habla y audio, mientras que las redes recurrentes se han orientado más al desarrollo de datos secuenciales como texto y voz.

Un aspecto importante de lo expuesto por LeCun Bengio y Hinton [32], es que las capas mencionadas no son diseñadas por expertos humanos, sino que estas capas son aprendidas desde los datos mediante procedimientos generales de aprendizaje.

El aprendizaje profundo o deep learning (DL) o por su traducción desde el inglés, es un subcampo del "*Machine Learning*" (ML/Aprendizaje de máquina) el cual es a su vez es un subcampo de la Inteligencia artificial conocido como AI, por sus siglas del inglés (ver Figura 2.10).

De la literatura examinada se puede argumentar que el campo de aplicación en el cual el DL se ha enfocado logrando excepcionales resultados se encuentra alrededor de lenguaje hablado, lenguaje natural, visión incluso juegos por computador, mientras que el clásico aprendizaje de máquina siegue dominando campos como la regresión lineal o arboles de decisión [24].

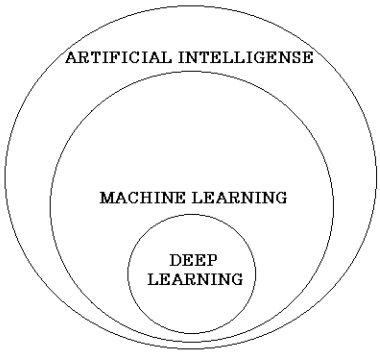


Figura 2.10: Representación de la relación del Deep Learning como subconjunto del Machine Learning dentro de la Inteligencia Artificial. Recuperado de Towards Datacience.

En este sentido es oportuno indicar que la Inteligencia Artificial (AI) tiene como meta proveer al usuario de un conjunto de algoritmos que puedan resolver cierta clase de problemas, mismos que los humanos pueden resolver de manera automática e intuitiva pero que para el computador constituye un reto tecnológico, cuyo enfoque está en las inferencias, planificación, heurística, etc. Un ejemplo de algoritmos de ML son las conocidas *Artificial Neural Networks* (ANNs) que aprenden de los datos y su función especial es el reconocimiento de patrones,

basado en forma similar a como el cerebro humano trabaja. El *deep learning* pertenece a este tipo de algoritmos y desde ese punto de vista, tiene ya más de 60 años de edad con respecto a técnicas que actualmente se pueden implementar [25] [26].

2.3.1. Historia de las redes neuronales y deep learning

Según la literatura relacionada con el tema, se puede descubrir que el *Deep Learning* tiene sus primeras apariciones en 1940 enfocada desde diversas perspectivas como Cibernética, Conexionismo [27] [28] [29] y las ya mencionadas *Artificial Neural Networks* (ANN), estas últimas y pese a que las ANNs, se inspiran en la estructura del cerebro humano para emular su funcionamiento y como sus neuronas interactúan unas con otras, se debe especificar que no significa que se trate de un modelo real de un cerebro humano, por el contrario las ANNs inspiradas en la estructura cerebral real presentan un paralelismo entre la idea básica de cerebro y como este puede comportarse ante ideas de aprendizaje.

Sus primeras apariciones teóricas datan de 1943 en las publicaciones de McCulloch y Pitts quienes estudiaron las características neuronales de las actividades nerviosas y su relación entre ellas para la determinación de actividades lógicas [30], tanto así que su primera representación funcional trataba de un clasificador binario, que consistía en una suma de las señales de entrada, multiplicadas por valores de pesos escogidos, debido a que los pesos que se utilizaron para identificar la etiqueta de la clase a discriminar debían ser introducidos por un humano, no fue considerada como aprendizaje automático. Más adelante en la década de los 50, específicamente en el año 1957, el psicólogo Frank Rosenblat inventa lo que sería denominada una red de tipo Perceptrón. Contrario a la representación de McCulloch y Pitts, Rosenblatt estimaba que la herramienta de análisis más apropiada en una red neuronal era la teoría de probabilidades, y esto lo llevó a una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias [31], este modelo no necesitaba de la intervención humana. El modelo Perceptrón como lo muestra la Figura 2.11 y la

Figura 2.12, así como su funcionamiento básico es utilizado hoy en día incluso para redes neuronales muy profundas.

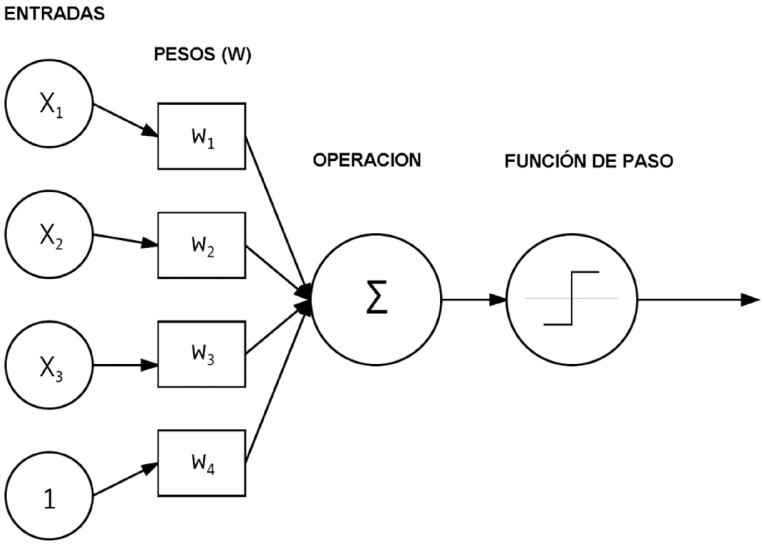


Figura 2.11: Red Neuronal, ejemplo simple de la arquitectura de red del perceptrón con un número determinado de entradas, procesamiento de pesos como suma y una función de predicción.

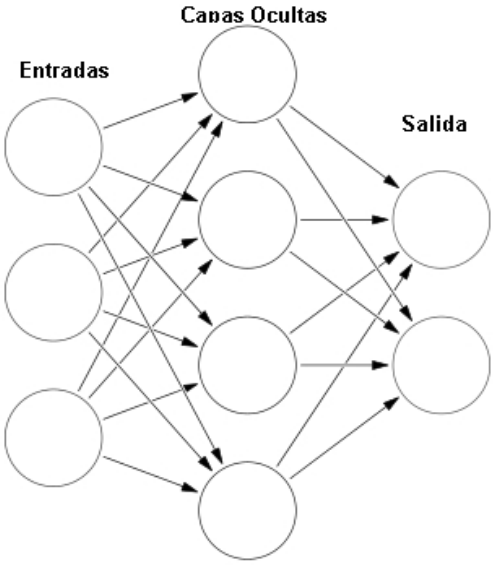


Figura 2.12: Estructura simple de una red neuronal, que muestra el conjunto de entradas, sus capas ocultas y las posibles salidas.

Aunque las técnica basadas en Perceptrón generaron gran impacto en la comunidad investigadora, fue en 1969 cuando una publicación de Minsky y Papert de 1969, lograron afianzar estos estudios debido a su aporte denominado: *"Perceptron: An Introduction to Computional Geometry*", en esta investigación [40] se demostró que un perceptrón con una función de activación lineal e independiente de su profundidad, trabajaba como un simple clasificador lineal, por tanto incapaz de resolver problemas no lineales. Sumado a esto los recursos con los que se contaba en aquel entonces dificultaban aún más el estudio de redes neuronales[6]. Esta idealización de las redes neuronales resultó en un duro golpe contra las mismas, hasta que en 1974 Paul Werbos quien se cree su inventor, desarrolló la teoría de redes denominadas de "retropropagación" o *backpropagation*, la misma que fue expuesta en su disertación doctoral en Harvard denominada "*Beyond Regression*" y reinventada en 1982 por David Parker del Stanford Linear Accelerator Center (Centro de Aceleración Lineal de Stanford).

El algoritmo de *backpropagation* habilita un *feedforward* multi capa en redes neuronales a ser entrenadas, esto combinado con funciones de activación no lineales, permitió a los investigadores resolver problemas no lineales, abriendo un abanico de opciones al estudio de las redes neuronales. En sí el algoritmo de retropropagación no es más que un método de cálculo de gradiente que se aplica en el entrenamiento de redes neuronales artificiales en el que se emplea un ciclo de propagación y adaptación de dos fases. Su inicio parte de la aplicación de un "estímulo" patrón como entrada a la red, éste se propaga desde la primera capa por las capas subsecuentes, hasta la generación de una salida. La salida es comparada con un conjunto de salidas deseadas a fin de calcular el error para cada una de las salidas. Estos errores se propagan en reversa desde la capa de salida, por las neuronas de las capas ocultas, la clave de esta propagación se encuentra en que las neuronas de la capa oculta solo reciben una fracción de la señal total del error, una contribución relativa que haya aportado cada neurona a la salida original. Este proceso a manera de ciclo se repite, capa por capa, hasta

[6] Hoy en día esta es una discusión que se mantiene acerca de lo que un perceptrón puede o no puede hacer.

que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total [33].

Se puede decir entonces que la retropropagación permite eficientemente entrenar una red neuronal de manera que aprenda de sus errores, aun hoy en día la cantidad de capas ocultas (Figura 2.12) resulta un elemento importante relacionado con el rendimiento de los equipos que se utilizan para entrenar redes.

Una variación importante y actualmente mejorada de las redes anteriormente mencionadas es el *deep learning*, que ha permitido tratar redes con mayor número de capas ocultas y con aprendizajes jerárquicos (los conceptos simples parten de capas bajas mientras que los conceptos abstractos se ubican en capas superiores) un ejemplo de ello es el trabajo sobre las denominadas Redes Neuronales Convolucionales o *Convolutional Neural Network* (CNN) aplicadas al reconocimiento de caracteres desarrollado en 1988 por Y. LeCun, B. Boser, J. Denker et al [42].

Las CNNs se utilizan en distintas aplicaciones del procesamiento y análisis de imágenes, donde se las considera el estado del arte en la visión por computador.

Campos del aprendizaje de máquina

Por lo general los algoritmos de Machine Learning pueden ser clasificados en dos grandes grupos: algoritmos supervisados y algoritmos no supervisados (Figura 2.13).

Algoritmos Supervisados

Este tipo particular de algoritmos tiene como característica la necesidad de un conjunto de entradas que generan un conjunto específico de salidas, el objetivo de estos algoritmos es aprender patrones que pueden ser utilizados para automáticamente conectar datos de entrada con su específica proyección de datos de salida.

Algoritmos no Supervisados

Para el caso de algoritmos no supervisados, estos tratan de descubrir automáticamente características específicas sin ninguna pista de como los datos de entrada trabajan.

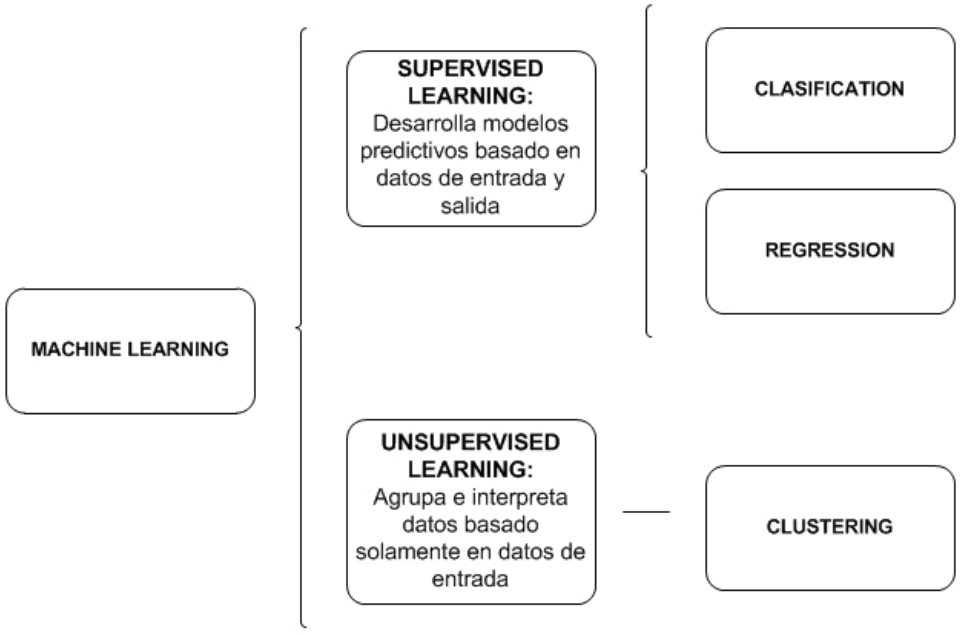


Figura 2.13: Clasificación general de los algoritmos de machine learning.

Para el caso de tratamiento de imágenes, el Machine Learning intenta identificar patrones que podrían permitir organizar grupos de imágenes y discriminarlas de otro grupo de imágenes. Las técnicas anteriormente utilizadas para estos propósitos requerían el preprocesamiento de imágenes (algo similar a lo que ahora se utiliza en d*eep learning*) mediante la extracción de características las mismas que luego de ser extraídas eran devueltas como un vector al que se le podían realizar determinadas operaciones para cuantificar sus características, entre las técnicas de extracción de características más destacadas se encuentran técnicas de codificación de texturas (Local Binary Patterns) [22], Haralick textura [34], formas (Hu Moments [35], Zernike Moments [36]), y color (color moments,

color histograms, color correlograms [37]), técnicas de detección de puntos clave o *keypoints* (FAST [38], Harris [39], DoG, entre otros) y descriptores locales invarientes (SIFT [21], SURF [40], BRIEF [41], ORB [42], etc.) así como también métodos de histogramas de gradientes orientados o [43]

Redes Neuronales Profundas (Deep Neural Networks)

Al igual que sus predecesoras, las redes profundas son algoritmos modelados a partir de los modelos y estructuras de los cerebros humanos con el objeto de reconocer patrones, su funcionamiento se basa en el uso del perceptrón, etiquetado y agrupamiento de datos crudos. La gran mayoría de patrones son elementos numéricos que deben ser almacenados en vectores como una representación del mundo real, estos elementos numéricos representan imágenes, sonido, texto, etc. Aun hoy en día no existe consenso en la comunidad científica sobre la profundidad (*how deeply is*) de una red neuronal profunda, aunque muchos científicos consideran a las redes neuronales profundas como grandes redes neuronales, este concepto se concibe de las Charlas de Andrew Ng[7], en donde indica que actualmente existen los equipos de cómputo lo suficientemente rápidos como para procesar y entrenar grandes volúmenes de datos en redes neuronales, al punto que el incremento de la cantidad de información, genera un incremento en la precisión de los datos a clasificar, lo que no necesariamente es el mismo comportamiento de los algoritmos de aprendizajes tradicionales como: *logistic regression, support vector machine, decision trees*, etc, lo que Andrew plasma en la Figura 2.14.

[7] Coursera y Jefe Científico de Baidu Research, fundó formalmente “Google Brain” que finalmente resultó en la producción de tecnologías de aprendizaje profundo en una gran cantidad de servicios de Google.

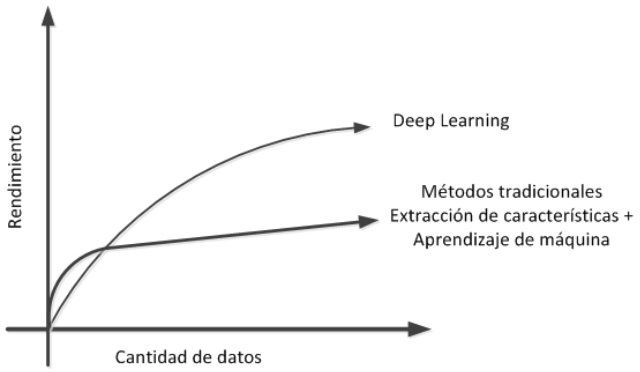


Figura 2.14: Visión que ofrece las charlas de Andrew Ng en el 2015 sobre el rendimiento del Deep Learning Vs Técnicas tradicionales de extracción de características medidas en base a la cantidad de datos.

Esta relación (Figura 2.14) hace que una red neuronal profunda se considere en realidad redes neuronales con gran cantidad de datos a procesar, por lo que un análisis de la arquitectura a diseñar permitiría determinar cuán profunda o no es la red por desarrollar, considerando los siguientes parámetros:

- La arquitectura por desarrollar trata de una red específica tal como una Red Neuronal Convolucional/*Convolutional Neural Networks* (CNN), Redes Neuronales Recurrentes/*Recurrent Neural Networks* (RNN) o en su defecto Redes Neuronales de Memoria a Largo Plazo/*Long Short-Term Memory* (LSTM).
- La red posee al menos una profundidad mayor a 2 (*Deep Learning*) o la red posee una profundidad mayor que 10 (*Very Deep Learning*).

Neurona y su estructura

Para entender cómo funciona una red neuronal multicapa, se debe analizar sus principales elementos constitutivos como la neurona. La unidad básica de procesamiento en una red neuronal es la neurona, a menudo llamada nodo o unidad que es el elemento que recibe las entradas de otros nodos, o en sud efecto de fuentes externas para el procesamiento de salidas. Cada entrada tiene asociada un peso *weight* (*w*) el cual es asignado en base a su importancia relativa

con otras entradas. El nodo aplica una determinada función *f* para una suma ponderada de sus entradas como lo muestra la Figuras 2.15 y Figura 2.16.

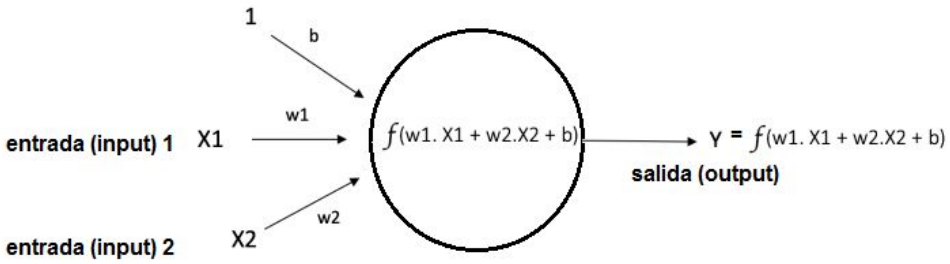


Figura 2.15: Estructura de una neurona, con entradas, una función de ponderación y una salida, la Figura muestra los pesos asignados a cada entrada y como lo función permitiría el procesamiento de datos para su salida, la entrada designada por b se denomina bias.

La función de activación no lineal, por lo general representa una operación matemática representada por una función conocida como función de activación, existen diversos ejemplos de funciones de activación que se pueden encontrar en la práctica [44], entre ellas:

- Sigmoidea: Toma una entrada de valor real y la traslada a un valor entre 0 y 1
- tanh: Toma una entrada de valor real y la traslada a un valor de -1 y 1
- ReLU: Cuyas siglas significan unidad lineal rectificada. Toma una entrada de valor real y la pone en un umbral de cero (reemplaza los valores negativos por cero)

La tabla 2.2: muestra una interpretación matemática de las funciones enumeradas

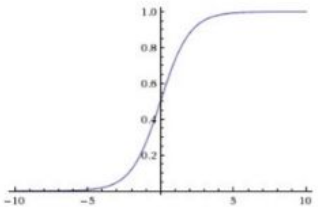
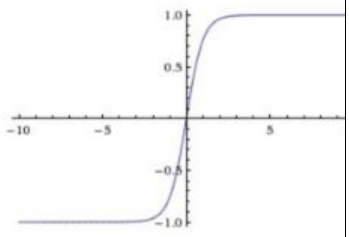
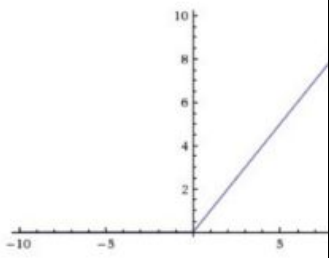
Nombre	Función Matemática	Interpretación	Gráfica
Sigmoidea	$\sigma(x) = 1 / (1 + \exp(-x))$	entrada de valor real y la traslada a un valor entre 0 y 1	
tanh	$\tanh(x) = 2\sigma(2x) - 1$	entrada de valor real y la traslada a un valor de -1 y 1	
ReLU	$f(x) = \max(0, x)$	entrada de valor real y la pone en un umbral de cero (reemplaza los valores negativos por cero)	

Tabla 2.2: Interpretación de las funciones de activación generalmente utilizadas en redes neuronales, presentado en http://cs231n.github.io/neural-networks-1/ como parte de Spring 2018 Assignments

Previamente se había mencionado una entrada adicional denominada *bias* cuya función es la de proveer a cada nodo un valor de entrenamiento constante.

Red Neuronal de pre-alimentación (*Feedforward Neural Network*)

La Red Neuronal de Pre-alimentación se considera la primera y más simple clase de red neuronal artificial [45] [46], contiene múltiples neuronas o nodos dispuestos en capas. Los nodos de capas adyacentes tienen conexiones o bordes entre ellos y todas esas conexiones tienen pesos asociados con ellos (ver Figura 2.16)

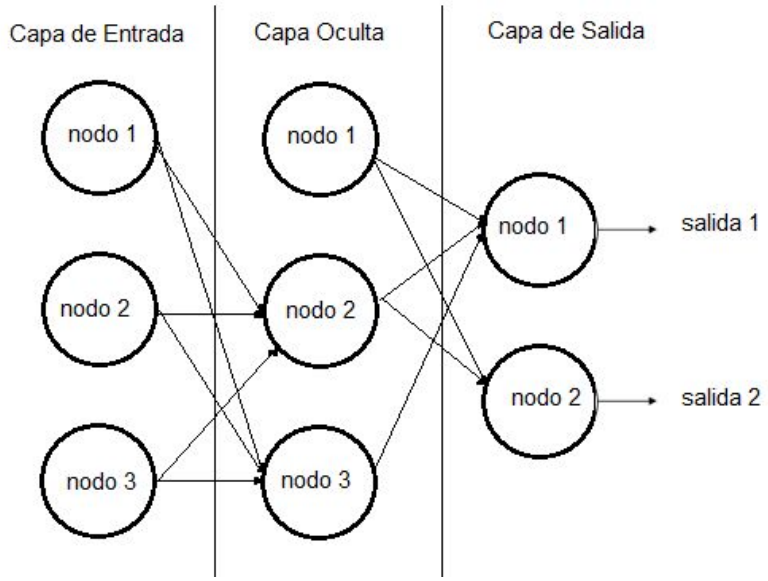


Figura 2.16: Ejemplo del feedforward representado a nivel de nodos entre los cuales se encuentran los nodos de entrada, nodos ocultos y nodos de salida

La Figura 2.16, muestra los distintos nodos que componen la red, como son; nodos de entrada, que proveen información desde el mundo exterior a la red y que en conjunto se conocen como capa de entrada (*Input Layer*), en esta etapa ningún procesamiento se lleva a cabo, su tarea principal es trasladar datos a los nodos ocultos. Los nodos ocultos por su parte no tienen conexión con el mundo exterior de ahí su nombre de “ocultos”, estos realizan procesamiento de datos y transfieren información desde los nodos de entrada hacia los nodos de salida, quienes a su vez se conocen como la capa de salida y son los responsables de transferir información desde las capas ocultas al mundo exterior.

En las redes Feed-Forward, la información se mueve en una sola dirección, en avanzada, desde los nodos de entrada pasando por la capa oculta, hasta llegar a la capa de salida, no consta de ciclos o lazos [47] a diferencia de las redes neuronales recurrentes donde la información forma ciclos. Las configuraciones de red de este tipo, más conocidas son:

- Perceptrón de capa simple: Es considerada la forma más simple de feed-forward neural network, debido a que no contiene capas ocultas [48].

- Perceptrón multicapa: Como su nombre lo sugiere contiene múltiples capas ocultas, se conocen como Multi Layer Perceptrón (MLP), a diferencia de los perceptrones de capa simple que solo puede aprender funciones lineales, los perceptrones multicapa pueden también aprender de funciones non lineales. Un ejemplo de este tipo de configuración lo muestra la Figura 2.17.

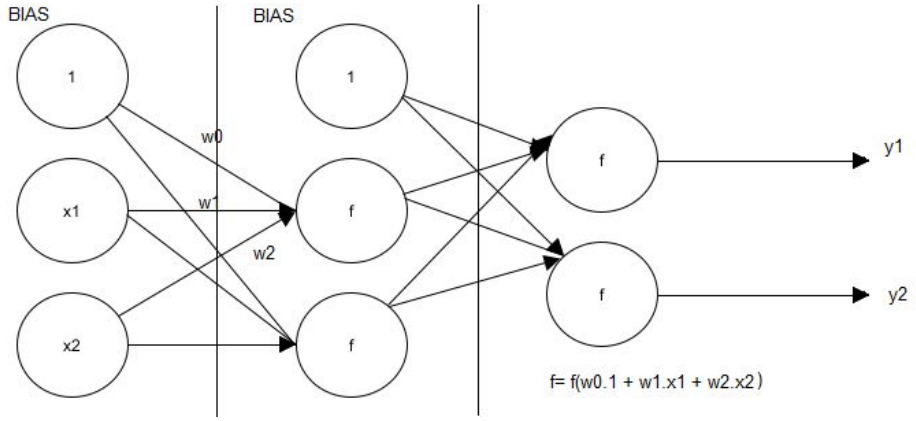


Figura 2.17: Perceptrón multicapa, con una capa oculta

Entrenamiento de una MLP a través del algoritmo de retro-propagación (*Back-Propagation*)

Los procesos a través del cual un perceptrón multi capa aprende se denominan algoritmo de retro-propagación de errores en el que se trata de un esquema de entrenamiento supervisado en el que el MLP técnicamente aprende de sus errores, la ANN contiene nodos en diferentes capas, las conexiones entre nodos de capas adyacentes tienen “pesos” (*weights*) asociados a cada uno, la meta del aprendizaje es asignar de manera correcta los pesos para cada uno de esos enlaces. En aprendizaje supervisado, el conjunto de datos de entrenamiento es etiquetado, lo que significa que para determinado grupo de datos de entrada se conocen los datos esperados de salida. Y algoritmo de retro-propagación, los pesos asignados a cada enlace se asignan de manera aleatoria, para cada entrada en el conjunto de datos de entrenamiento la ANN es activada y su salida

se puede observar, esta salida es comparada con la salida esperada que es conocida y el error se propaga en reversa hacia la capa anterior, este error es anotado y su peso es ajustado consecuentemente, este proceso se repite hasta que el error cumple con un determinado umbral. Una vez que el algoritmo termina, se puede decir que la red ha aprendido por lo que se encuentra lista para nuevas entradas de datos. Se dice entonces que la red aprende de varias muestras y de sus errores.

Para tareas de clasificación, la función que generalmente es utilizada es la Softmax [44], como la función de activación en la capa de salida de un MLP, que asegura que las salidas son valores de probabilidad y que alcanza un valor de "1". La función Softmax toma un vector de puntuaciones arbitrarias de valores reales y lo lleva a un vector de valores entre cero y uno que alcanzan en suma 1. Entonces, *Probability (Pass) + Probability (Fail) = 1.*

Pasos de entrenamiento

- *Forward Propagation*: todos los pesos en la red son aleatoriamente asignados (Figura 2.18) entonces la red toma la primera muestra de entrenamiento como entrada, la neurona de salida V del nodo en consideración puede ser calculada de acuerdo a *f* como función de activación sigmoidea, en la que $V = f\ (1 * w1 + X1 * w2 + X2 * w3)$ [58] considere que $X1$ y $X2$ pueden serlos primeros datos de un *dataset* del cual se quiere realizar un aprendizaje y posterior predicción, de manera similar los otros nodos de las capas ocultas son calculados, la salida de los nodos de la capa oculta sirven como entrada de la capa de salida, esto permite calcular probabilidades de salida desde los nodos previos en la capa de salida. Las probabilidades de salida calculada se evalúan para determinar cuán lejos probabilísticamente se encuentran, esta distancia es considerada por la red como una salida incorrecta, dando paso al *back-propagation* y actualización de pesos.

- Entrada a la red= [$X1$, $X2$]
- Salida deseada desde la red (objetivo)= [1,0]

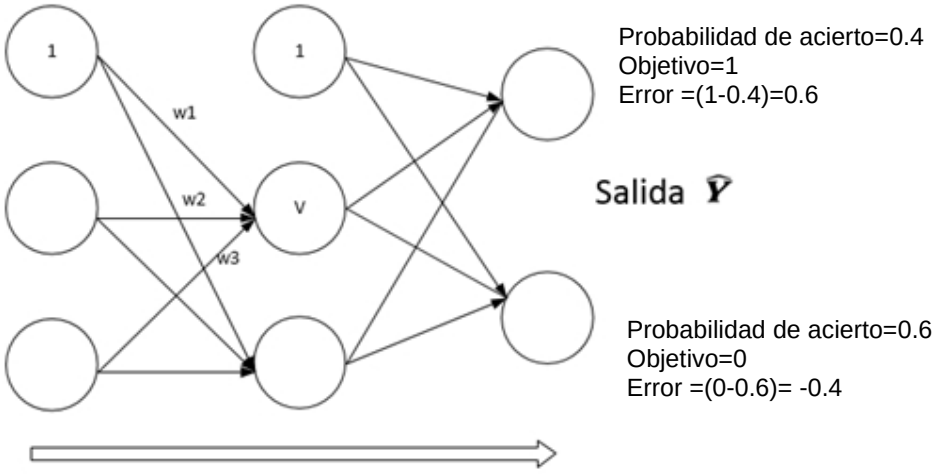


Figura 2.18: Forward Propagation, pesos y cálculo de salida con probabilidades de acierto o fallas

- *Back Propagation*: Se calcula el error de salida y se propaga mencionado error en reversa hacia las capas interiores utilizando el algoritmo de retro-propagación para el cálculo de gradientes[8], posterior a esto se utiliza un método de optimización por lo general el "gradiente descendente" para ajustar todos los pesos en la red con el objetivo de reducir el error en la capa de salida (ver Figura 2.19). Para este caso los nuevos pesos del nodo posterior al *backpropagation* en análisis son $w4, w5, w6$.

[8] Variación de una magnitud en función de la distancia, a partir de la línea en que esta variación es máxima en las magnitudes cuyo valor es distinto en los diversos puntos de una región del espacio.

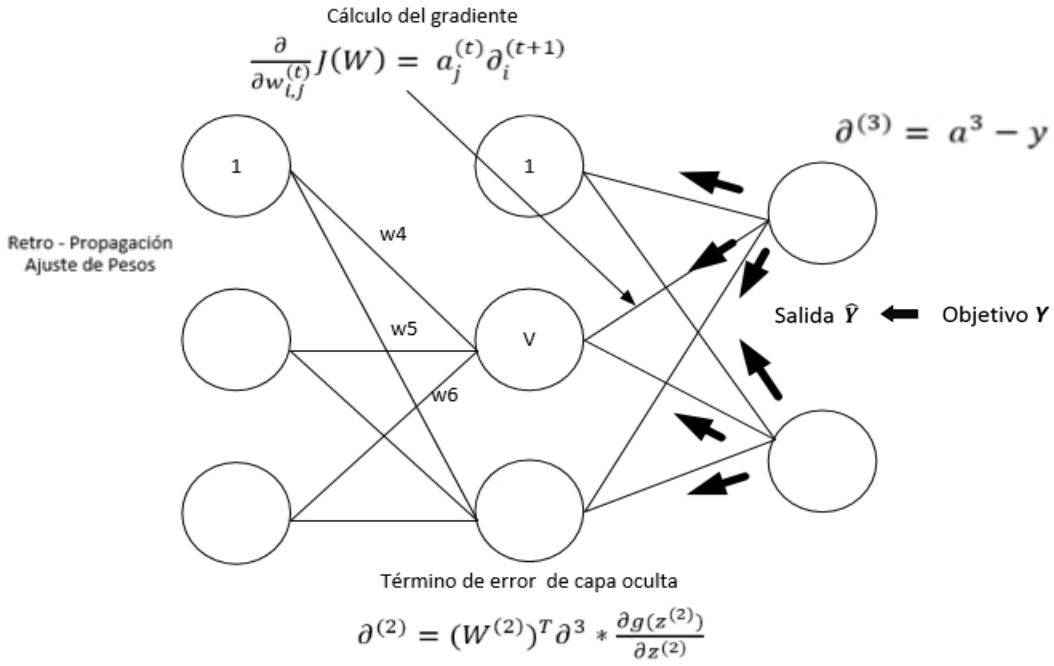


Figura 2.19: Back-Propagation y actualización de pesos en una MLP.

Con el nuevo conjunto de pesos y una vez ingresados los mismos valores de entrada, la red debería tener un rendimiento mejor que el anterior debido a que los pesos se ajustaron de forma que se minimice el error en la predicción, como lo muestra la Figura 2.20. Por lo que se puede deducir que la red se ha entrenado correctamente en su primera muestra de entrenamiento. Este proceso debe repetir para cada una de las muestras del dataset [49].

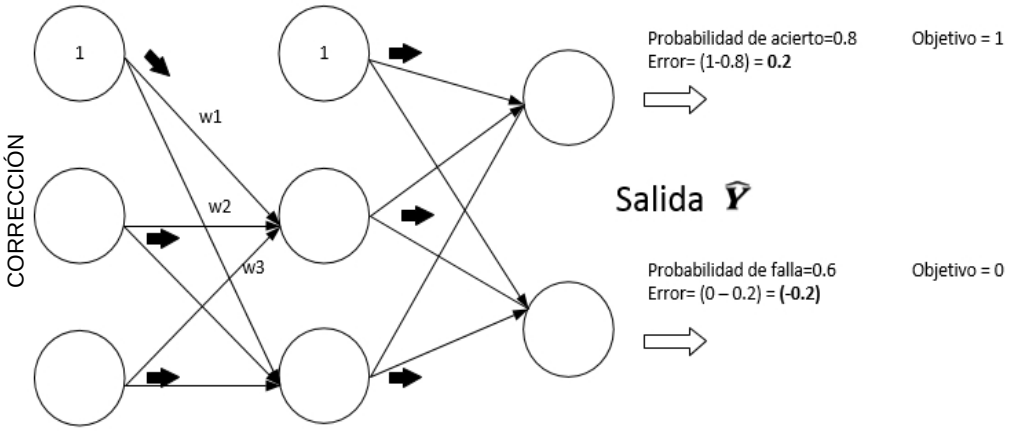


Figura 2.20: Efecto de aprendizaje de la red, en el que se puede evidenciar que las probabilidades de acierto o falla mejoran.

2.4. Redes neuronales convolucionales

Para entender una red neuronal convolucional o CNN por sus siglas en inglés (*Convolutional Neural Network*) se debe tomar como base las redes neuronales tradicionales, debido a que una CNN es un tipo de red neuronal artificial (sección anterior) con una diferenciación en el perceptrón multicapa. En las redes neuronales tradicionales, cada neurona en la capa de entrada (*input layer*) es conectada a cada neurona de salida (*output layer*) en cada siguiente capa, esto es lo que se denomina *fully connected layer* (FC) o capa totalmente conectada. En cambio en una CNN, no se utilizan las capas FC hasta la últimas capas en la red. Por lo tanto, se puede entender una CNN, como una red neuronal que se intercambia en una capa "convolucional" especializada en lugar de capa "totalmente conectada" para al menos una de las capas en la red [50]. Una función de activación no lineal (ReLU), es entonces aplicada a la salida de esas convoluciones y el proceso de "convolución/activación" continua[9] hasta que finalmente alcanza el final de la red y se aplica una o dos capas FC, en dónde se puede obtener la salida final clasificada. Cada capa en una CNN aplica un

[9] Este proceso se junta a una mezcla de otros tipos de capas, lo que ayuda a reducir el ancho y la altura del volumen de entrada al igual que ayuda en la reducción del sobreajuste (overfitting).

diferente conjunto de filtros y combina los resultados, alimentando la salida en la siguiente capa de la red. Durante el entrenamiento, una CNN aprende automáticamente los valores de esos filtros, es esta característica lo que las convierte en ideales para el estudio de visión artificial, y en la clasificación/segmentación de imágenes, entre otras aplicaciones.

Para complementar este concepto, se debe entender que en temas relacionados con *deep learning,* una convolución es una operación que utiliza multiplicaciones de matrices seguidas de una suma. En la literatura especializada se le denota por el operador * y se define matemáticamente sobre una imagen de entrada *I* bidimensional y un kernel *K* bidimensional definido como:

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I\,(i-m, j-n) K\,(m,n) \tag{2.1}$$

A pesar de (2.22), la mayoría de las librerías de ML y DL usan una función simplificada de correlación cruzada como:

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I\,(i+m, j+n) K\,(m,n) \tag{2.2}$$

Debido a que matemáticamente hablando una imagen es una matriz[10], en términos de matrices, una imagen puede considerarse como una gran matriz y un *kernel* o matriz convolucional como una matriz pequeña que será utilizada para operar sobre la principal (imagen) en función de generar desenfoque, nitidez, detección de bordes, y otras funciones de procesamiento de imágenes. Particularmente esta matriz pequeña, se desplaza desde la parte superior de la matriz (imagen) y se desliza de arriba-abajo, e izquierda a derecha, aplicando una operación matemática (convolución) sobre cada coordenada *x,y* de la imagen original. Normalmente los núcleos (*kernel*) se definen a mano para obtener

[10] Matemáticamente una imagen es una matriz y a diferencia de estas para el caso específico de imágenes RGB, dicha matriz contiene un componente adicional como lo es la profundidad para el color rojo, verde y azul.

diferentes funciones de procesamiento de imágenes como desenfoque (suavizado promedio, suavizado gaussiano, etc.), la detección de bordes (Laplaciano, Sobel, Scharr, Prewitt, etc.) y el afilado, todas estas operaciones son formas de núcleos definidos a mano y que están diseñados específicamente para realizar una función particular.

Núcleos o Kernels

Como se explicó en la sección anterior, una imagen puede ser considerada como una matriz mayor y un kernel como una matriz menor, la idea entonces es deslizar el *kernel* a lo largo de toda la imagen original como lo muestra la Figura 2.21.

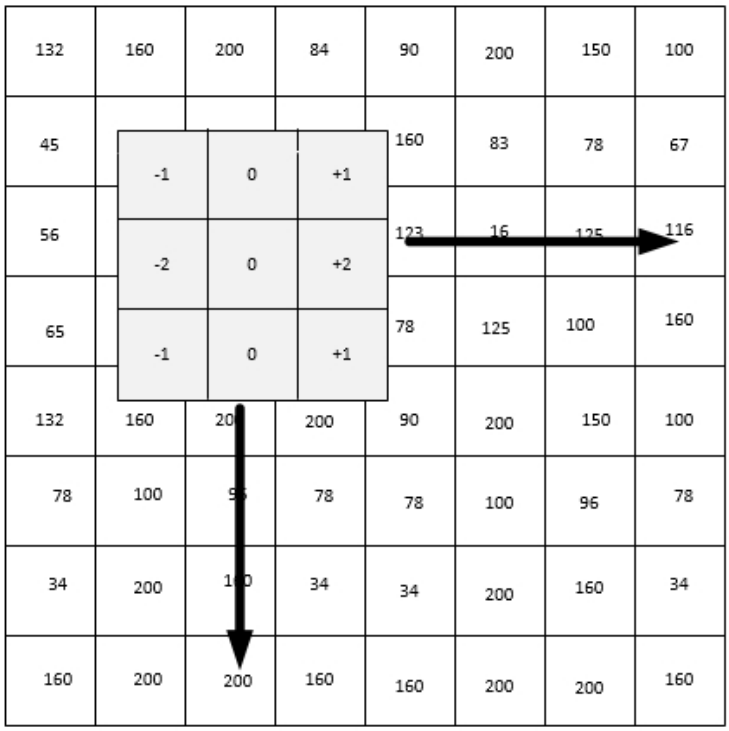


Figura 2.21: Proceso de convolución entre la imagen (matriz grande) y el kernel (matriz pequeña).

El proceso, en cada coordenada *(x, y)* de la imagen original, se detiene y examina el vecindario de los píxeles ubicados en el centro del núcleo de la imagen. Con ese vecindario de píxeles, se ejecuta una convolución del kernel y se obtiene un

solo valor de salida. El valor de salida se almacena en la imagen de salida en las mismas *(x, y)* coordenadas que el centro del núcleo.

Convolución

Para el caso de procesamiento de imágenes, las convoluciones requieren de tres componentes: la imagen de entrada desde luego, el kernel que se aplicará a la imagen de entrada y un espacio de almacenamiento para la imagen de salida resultado de la convolución, el proceso a seguir consistirá en:

- Seleccionar una coordenada *(x,y)* de la imagen original.
- Colocar el centro del kernel en esta coordenada.
- Tomar un elemento de multiplicación de la región de imagen de entrada y el kernel, luego sumar los valores de estas operaciones de multiplicación en un solo valor. La suma de estas multiplicaciones se denomina la salida del kernel.
- La salida del Kernel deberá ser colocada en las mismas coordenadas *(x,y)* iniciales de la imagen de salida.
- Una vez aplicada la convolución, se colocará el pixel en la coordenada *(i,j)* de la imagen de salida.

2.4.1. Funcionamiento de una red convolucional y deep learning

La aplicación de filtros convolucionales, activaciones de funciones no lineales, agrupamientos (*pooling*), propagación en reversa (*backpropagation*), permite que una red CNN sea capaz de aprender filtros que puedan detectar bordes y estructuras en niveles bajos de la red y luego utilizarlos como estructuras de construcción, eventualmente detectando objetos de alto nivel en las capas más profundas de la red.

Como se ha mencionado en las secciones anteriores una red neuronal acepta una imagen de entrada (si es el caso) y la procesa a través de una serie de capas ocultas, por lo general a través de funciones de activación no lineales. Cada capa oculta está también compuesta de una serie de neuronas, en las que cada

neurona está totalmente conectada a todas las neuronas de la capa anterior. La última capa de una red neuronal está también totalmente conectada y presenta la clasificación final de salida de la red. Pero en el caso específico de imágenes, las redes neuronales trabajan sobre las intensidades del pixel en bruto, lo que implica un pobre re-escalamiento de la imagen a medida que aumenta el tamaño de la imagen, y poca precisión, contrario a esto las CNN toman ventaja de aquello al presentar una estructura más sensible a cambios.

Existen muchos tipos de capas utilizadas para construir CNNs pero entre las más conocidas se encuentran:

- Convolucional.
- Activación (ACT/ReLU).
- *Pooling* (Pool).
- *Fully-connected* (FC).
- *Batch Normalization* (BN).
- *Dropout* (DO).

Al apilar un conjunto de estas capas, de manera específica se obtiene una CNN, un ejemplo lo muestra la Figura 2.22.

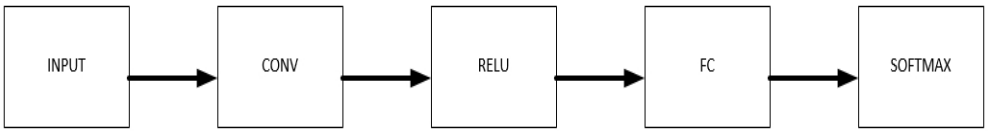


Figura 2.22: Apilamiento de capas en una red neuronal simple que acepta una entrada, aplica una convolución, luego aplica una capa de activación, posteriormente una capa totalmente conectada y finalmente un clasificador Softmax.

De las capas de la Figura 2.22, solamente CONV y FC contienen parámetros que son aprendidos durante el proceso de entrenamiento, la capa de activación y *dropout* no son consideradas capas en si.

Capas convolucionales

La capa convolucional (CONV) es el bloque principal de una CNN, los parámetros de la capa CONV consisten en un conjunto de filtros de aprendizaje *K* (*kernels)*, donde cada filtro tiene un ancho y altura, y son por lo general cuadradas, estos filtros son pequeños (espacio dimensional) pero se extienden a través de toda la profundidad del volumen. Para entradas para la CNN, la profundidad es el número de canales en la imagen, mientras que para volúmenes más profundos en la red, la profundidad será el número de filtros aplicados en la capa anterior (ver Figura 2.23).

La Figura 2.23 muestra, el proceso en el que cada uno de los *K* núcleos se desliza a través de la región de entrada, calculando una multiplicación por elementos, sumando y luego almacenando el valor de salida en un mapa de activación bidimensional.

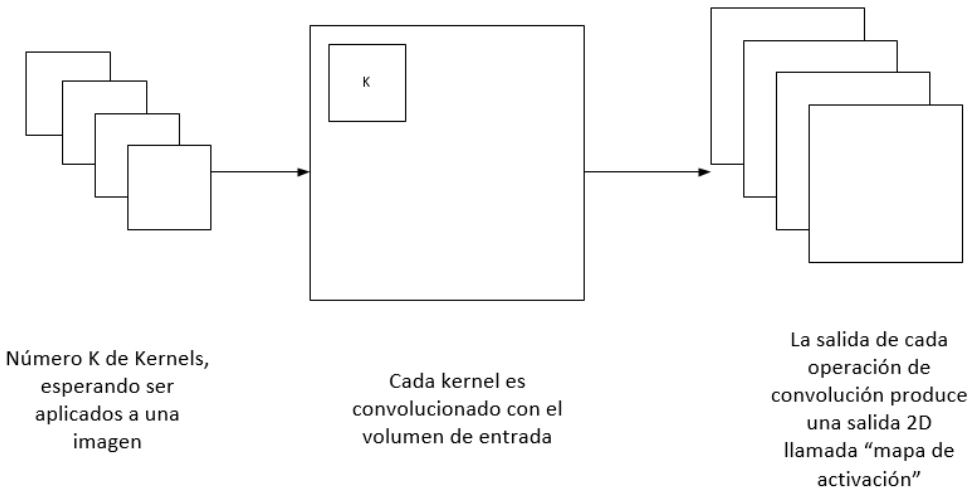


Figura 2.23: Proceso convolucional mediante el cual se obtiene el mapa de activación.

Posterior a la aplicación de los *K* filtros al volumen de entrada, se obtienen *K* mapas de activación bidimensionales, entonces los mapas de activación K son

apilados a lo largo de la dimensión de profundidad de la matriz, para formar el volumen de salida final (Figura 2.24).

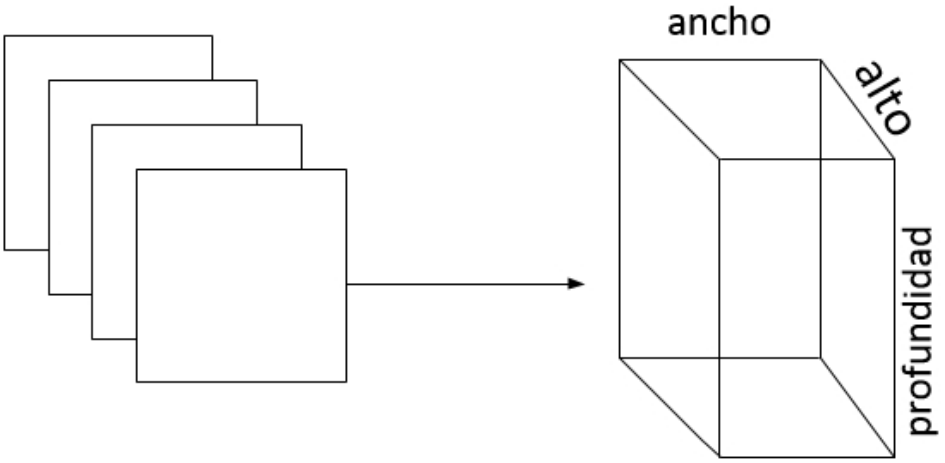


Figura 2.24: La Figura muestra el proceso posterior a obtener los mapas de activación K, donde se apilan juntos para formar el volumen de entrada a la siguiente capa en la red.

Cada entrada en el volumen de salida es, por lo tanto, una salida de una neurona que “mira” solo una pequeña región de la entrada. De esta manera, la red “aprende” los filtros que se activan cuando ven un tipo específico de función en una ubicación espacial dada en el volumen de entrada.

En las capas más bajas de la red, los filtros pueden activarse cuando ven regiones de borde o de esquina. Luego, en las capas más profundas de la red, los filtros pueden activarse en presencia de características de alto nivel, como partes de la cara, la pata de un perro, el capó de un automóvil, etc. El concepto de convolucionar un filtro pequeño con un volumen de entrada grande ® tiene un significado especial en las redes neuronales convolucionales, específicamente, la conectividad local y el campo receptivo de una neurona. Cuando se trabaja con imágenes por lo general es poco práctico conectar las neuronas en el volumen actual a todas las neuronas en el volumen anterior, simplemente hay demasiadas

conexiones y demasiados pesos, lo que hace imposible entrenar redes profundas en imágenes con grandes dimensiones espaciales.

Por el contrario, para una CNN se elige conectar cada neurona solo a una región local del volumen de entrada; se llama al tamaño de esta región local el campo receptivo (o simplemente, la variable F) de la neurona.

En general, esto permitiría explicar la conectividad de las neuronas en un volumen de entrada sin explicar el ordenamiento del volumen de salida, para lo cual es necesario comprender conceptos como: *52onfi* (profundidad), *stride* (paso), y *zero-padding* (relleno a cero)

Depth

La profundidad de un volumen de salida controla el número de neuronas en la capa CONV que conecta a una región local con el volumen de entrada. Cada filtro produce un mapa de activación que se "activa" en presencia de bordes orientados o manchas o color. Para una capa CONV dada, la profundidad del mapa de activación será *K*, o simplemente la cantidad de filtros que estamos aprendiendo en la capa actual. El conjunto de filtros que "miran" la misma *(x; y)* ubicación de la entrada, se llama la columna de profundidad.

Stride

Dado el concepto de ventana deslizante explicado anteriormente sobre una imagen, que se ha denominado convolución y en el cual solamente se toma un pixel a la vez, permite aplicar este mismo principio a CNNs, es decir para cada paso, se crea una nueva columna de profundidad alrededor de la región local de la imagen donde se está convolucionando cada uno de los *K* filtros con la región y que almacena la salida en un volumen tridimensional. Por lo genera cuando se crea la capa CONV, se utiliza un stride (paso) de tamaño S de, ya sea S=1 ó S=2.

Zero-Padding

Como se pudo observar previamente, es necesario "rellenar" los bordes de una imagen para conservar el tamaño de la imagen original cuando se aplica una

convolución; lo mismo ocurre con los filtros dentro de una CNN. Utilizando el "*zero-padding"* se puede "rellenar" la entrada a lo largo de los bordes de modo que nuestro tamaño de volumen de salida coincida con el tamaño de volumen de entrada. La cantidad de relleno a aplicar está controlada por el parámetro P.

Esta técnica es especialmente crítica cuando se comienza a observar las arquitecturas CNN profundas que aplican múltiples filtros CONV uno encima del otro.

Capas de activación

Después de cada capa CONV en una CNN, se aplica una función de activación no lineal, como ReLU, ELU o cualquiera de sus otras variantes (ver Tabla 2.2). Por lo general, se denomina capas de activación como RELU en los diagramas de red, ya que las activaciones ReLU son las más comunes. De uso común, también se puede indicar ACT; para cualquier caso se está aplicando una función de activación dentro de la arquitectura de la red.

Las capas de activación no son técnicamente "capas" (debido al hecho de que no se aprenden parámetros / pesos dentro de una capa de activación) y, a veces, se omiten de los diagramas de arquitectura de la red, ya que se supone que una activación sigue inmediatamente a una convolución.

Capa Pooling

Su principal objetivo es la reducción de información espacial, existen dos métodos para reducir el tamaño de un volumen de entrada: las capas CONV con un stride > 1, y las capas *POOL*. De acuerdo a la literatura revisada, es común insertar capas *POOL* entre capas *CONV* consecutivas en arquitecturas CNN.

Capas totalmente conectadas

Las neuronas en las capas *FC* están completamente conectadas a TFODAPIs las activaciones en la capa anterior, como es el estándar para redes neuronales de avance. Las capas de *FC* siempre se colocan al final de la red (es decir, no aplica una capa *CONV*, luego una capa *FC*, seguida de otra capa *CONV*.

Es común usar una o dos capas *FC* antes de aplicar el clasificador de softmax, como lo muestra la 54onfiguración:

INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC => FC

Batch Normalization

Se utilizan para normalizar las activaciones de un determinado volumen de entrada antes de pasar a la siguiente capa en la red [51].

Dropout

También es una forma de normalización cuyo objetivo es el de prevenir el *overfitting* al aumentar la precisión de prueba

CAPÍTULO 3

En los últimos años las técnicas de reconocimiento facial han cobrado significativa importancia, debido tanto al desarrollo de aplicaciones interesadas en esta área como a la disponibilidad de tecnologías que han permitido dicho desarrollo. Sin embargo, el reconocimiento facial aún debe superar obstáculos como cambios de iluminación, calidad de imágenes capturadas y posición específica del rostro a reconocer, factores que siguen marcando distancia con la forma en la que el cerebro humano logra identificar a una persona, esto a su vez ha generado una variedad de técnicas de reconocimiento con el objetivo lograr mejores resultados, técnicas que independientemente de su implementación, exigen datasets robustos, estandarizados y pre-procesados para facilitar la tarea de entrenamiento y posterior reconocimiento.

En la actualidad es posible separar a dos grandes grupos de técnicas de reconocimiento facial: las primeras son aquellas que se enmarcan en técnicas tradicionales y las segundas son aquellas que aplican cierto nivel de *machine learning,* específicamente las relacionadas con aprendizaje profundo. Dentro de las técnicas tradicionales, se ha realizado un levantamiento previo de información sobre los métodos con mayor relevancia, por lo que la presente tesis incluye en este estudio a las propuestas que implementan eigenfaces, fisherfaces e histogramas de patrones binarios locales. Por otra parte, en cuanto a las técnicas sobre aprendizaje profundo, se tomó en consideración bibliotecas de redes neuronales de código abierto que trabajen sobre redes pre-entrenadas como Resnet 101, Retinanet e Inception las cuales se implementaron sobre APIs de *TensorFlow,* y *Keras* utilizados para reconocimiento de objetos pero cuyos pipelines fueron adaptados para reconocimiento facial específicamente para este estudio, adicionalmente fue posible evaluar también dentro de estas técnicas, la implementación de *Facenet*, considerada así por los resultados mencionados en el paper "*FaceNet: A Unified Embedding for Face Recognition and Clustering*" [71]

que lo colocan como una técnica con mejores resultados sobre otras implementaciones.

En las siguientes secciones se analizará comparativamente cada una de las técnicas mencionadas para posteriormente presentar los resultados de esta evaluación.

3. Metodología y Datos

3.1. Generación del dataset

La información recolectada para la generación del dataset de trabajo para la presente tesis fue descargada de redes sociales desde agosto del 2018 hasta diciembre del mismo año, lo que permitió trabajar con 41 clases y lograr un dataset de 857 imágenes aleatorias por individuo, es decir sin posición, resolución, tamaño ni condiciones de iluminación estandarizadas fuera del preprocesamiento. Cada clase corresponde a una carpeta etiquetada con el nombre de su correspondiente dueño para procesos de validación posterior. Este dataset ha sido codificado como Dataset UP para propósitos de referencia y corresponde al denominado grupo A. Paralelamente a ello se generó un dataset de control al cuál se le denominó grupo B, que consta de fotografías de los mismos individuos y ajeno a todo proceso de entrenamiento con una resolución de 240x300 pixeles con tamaño y condiciones de luz homogéneas entre fotografías.

3.2. Metodología

Tanto para la evaluación de técnicas tradicionales como para la evaluación de técnicas basadas en aprendizaje profundo, fue necesario trabajar en flujo de trabajo que consiste en distintas fases de trabajo como fase 1: preprocesamiento, el cual varía dependiendo del método a evaluar y que se explicará en detalle en la sección de análisis correspondiente al método evaluado, fase 2: entrenamiento, similar al caso anterior, detallado en su respectiva sección y evaluación del método. Para la evaluación del método de reconocimiento de rostros, fue considerado la utilización de dos grupos de imágenes denominadas grupo A que

son fotografías utilizadas para el entrenamiento de la red; y grupo B, fotografías de los mismos individuos pero de distinto dataset ajeno a todo proceso de entrenamiento, del grupo A se tomaron aleatoriamente 10 fotografías en 5 ejecuciones distintas con una resolución de 160x160 por imagen, Para el grupo B se tomaron en cuenta las 10 fotografías seleccionadas aleatoriamente del grupo anterior con mejor resolución (240x300) como elemento de control.

Para la evaluación de cada grupo se definieron dos parámetros: (i) *ACIERTOS* como la validez del reconocimiento expresado en que la foto corresponda al sujeto etiquetado; y, (ii) *FALLAS* a toda expresión de la implementación en la que el rostro no corresponde al sujeto etiquetado.

3.3. Análisis de técnicas tradicionales de reconocimiento facial

Eigenfaces

A continuación, se presenta el análisis de reconocimiento facial mediante *EigenFaces* código disponible en la web y cuya implementación se basa en el método de *Turk y Pentland* [15], para su funcionamiento fue necesario la instalación de librerías como *Numpy, Matplotlib y Opencv* (cv2). El flujo de trabajo toma las consideraciones base de matemáticas de Numpy tales como preprocesamiento de imágenes, construcción de *eigenfaces*, representación de imágenes *eigenfaces*, reconocimiento de rostros mediante algoritmos *K near neighbors (K-nn)* o vecinos cercanos, y finalmente la evaluación de rendimiento. Este último paso fue analizado con el procedimiento detallado en la sección 3.2. Para propósitos de simplificación y redacción el dataset de pruebas fue etiquetado como “Dataset UP”.

Pasos y Metodología

1. Carga de imágenes y respectiva conversión de cada una de ellas en arreglos susceptibles de ingresar a una matriz *Nump.* Uno de los principales problemas encontrados en esta propuesta es la necesidad de normalización de imágenes,

esto es: las imágenes que constituyen el conjunto de entrenamiento deben mantener las mismas condiciones de iluminación y deben normalizarse de forma que ojos y la boca estén alineados en todas las imágenes. Este inconveniente dada la propuesta de un dataset no estandarizado eliminaba fotografías que ocasionaban un problema en la fase de cálculo. Adicionalmente las fotografías se vuelven a muestrar a una resolución común.

2. Cálculo de promedios, una vez que las fotografías ahora son representadas por un espacio vectorial (concatenación de filas de pixeles) y almacenadas en una matriz (cada columna es una imagen), es necesario calcular la imagen promedio la cual debe ser restada en la imagen original en la matriz formada, un ejemplo del resultado lo muestra la Figura 3.1.

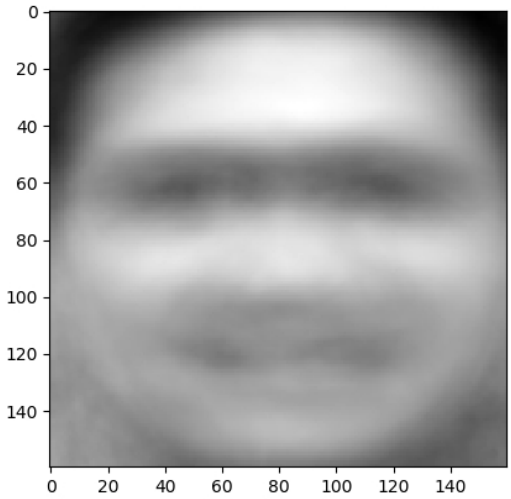


Figura 3.1: Gráfica con el vector promedio obtenido del Dataset UP, posterior al proceso de vectorización de imágenes.

2. Cálculo con imágenes normalizadas

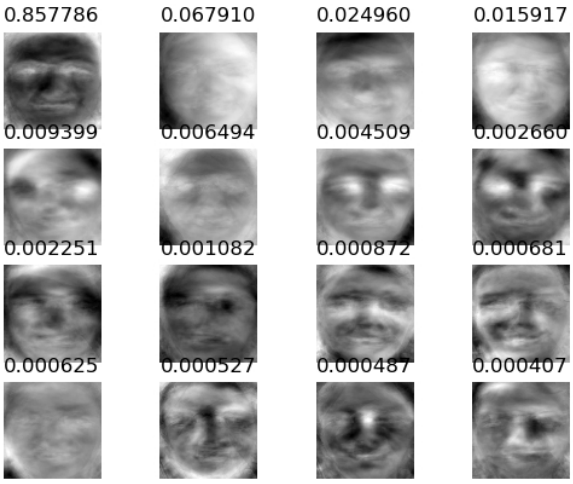


Figura 3.2: Conjunto de 16 imágenes eigenfaces del Dataset UP

3. Cálculo de matriz de covarianza reducida para evitar saturaciones por tamaño de matriz en general
4. Cálculo de los eigenvalues y eigenvectors
5. Proyección de imagen en el eigenspace y reconstrucción mediante la utilización de K eigenfaces con (3.1)

$$W = v' \quad \text{y} \quad X = V \tag{3.1}$$

6. Reconocimiento de imagen proyectada en el *eigenspace*, la imagen es considerada como la combinación lineal de las *eigenfaces.* Para el reconocimiento de un rostro se aplica el algoritmo K-nn para encontrar el arreglo más cercano a la base de datos almacenada. Para el caso de evaluación se consideró aleatoriamente una imagen de un subconjunto diferente de imágenes y una imagen aleatoria del mismo subconjunto de entrenamiento.

Al momento de la escritura del presente informe, el eigenface se considera uno de los métodos tradicionales más populares para reconocimiento facial, debido a que los *K eigenvectors* podrían contener la cantidad de variedad de rostros pero que

también mantiene aún el problema de la variación de luz tanto del rostro como del entorno, por lo que la literatura investigada propone analizar técnicas basadas en redes neuronales debido a que las condiciones de iluminación podrían ser alimentadas a la red como elementos del dataset.

Los resultados de agrupamiento por similitud se pueden apreciar en la Figura 3.3.

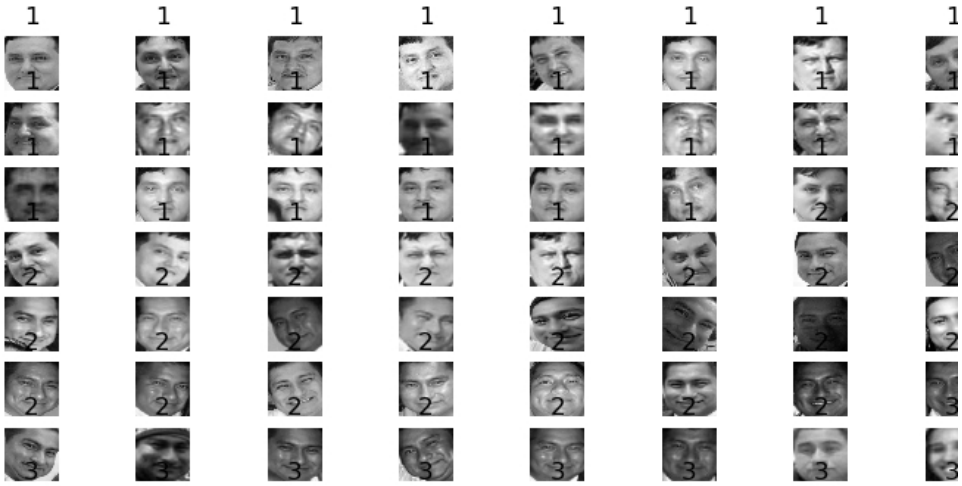


Figura 3.3: Agrupamiento de imágenes por similitud, como resultado de la etapa de clasificación de características.

Las ventajas de implementar un método de reconocimiento facial basado en eigenfaces descansa en lo que menciona [72], de las cuales se consideraron más importantes, las relacionadas con:

- Independencia de la geometría facial.
- Simplicidad.
- Velocidad de reconocimiento con respecto a otros métodos.
- Mejora tasa de éxito en reconocimiento.

Si embargo, para propósitos de la presente tesis, el preprocesamiento de imágenes no estandarizadas conlleva un esfuerzo y costo computacional

adicional, el cual obligaba en muchos casos desestimar fotografías que no pudieron ser normalizadas tanto por posición como por iluminación. Con este antecedente se buscaron alternativas como fisherfaces y histogramas de patrones binarios locales.

Fisherfaces

Si bien *eigenfaces* es considerada como la más exitosa técnica en el ámbito de reconocimiento facial, en la práctica la necesidad de normalización de imágenes la condena a mantener un poco probabilidad de ser implementada en situaciones prácticas del mundo real. Como una alternativa más flexible puede considerarse las implementaciones basadas en fisherfaces que es una mejora del eigenface que a diferencia de este (uso de PCA) utiliza el Análisis discriminante de Fisher o "*Fisher's Linear Discriminant Analysis"* abreviado como FLDA/LDA.

Fisherfaces, es una técnica para reconocimiento facial que considera la luz e incluye el análisis de expresiones faciales. Tal como se lo mencionó, utiliza el discriminante de Fisher para proyectar datos de manera que su dispersión se considere óptima en cuanto a clasificación [73]. Ahí la principal diferencia con eigenfaces, en la que los PCA busca una reducción de los mejores vectores que describan los datos, LDA busca los vectores que ofrezcan mejor discriminación entre clases, en el presente estudio alivió de manera significativa la necesidad de normalización de imágenes.

Pasos y metodología.

A continuación, se resumen los pasos relacionados a esta técnica que de manera general se enmarcan en:

- Definición de matriz de varianza entre clases con (3.2) y definición de la varianza en cada clase con (3.3), en la que μ_i representa la imagen promedio de la clase X_i, $|X_i|$ el número de puntos al interior de la clase X_i y μ el promedio de todas las imágenes.

$$S_B = \sum_{i=1}^{c} |X_i| \, (\mu_i - \mu)(\mu_i - \mu)^T \tag{3.2}$$

$$S_W = \sum_{i=1}^{c} \sum_{X_k \in X_i} (X_k - \mu_i)(X_k - \mu_i)^T \tag{3.3}$$

- Encontrar la matriz de proyección, mediante la resolución de un problema de optimización para maximizar S_By minimizar S_W cuyo resultado es una matriz de vectores propios, que implica la reducción de la dimensionalidad de las imágenes.
- Aplicar LDA para reducir el número de clases y agrupar las imágenes según la clase a la que pertenezcan.

Pese a lo simple que puede parecer el método, en la práctica demanda de un gran costo computacional por los cálculos a ser implementados. De ahí que en la presente tesis también se consideró una tercera alternativa como método tradicional de reconocimiento facial, esta vez basado en histogramas de patrones binarios locales.

Histogramas de patrones binarios locales

Utilizado inicialmente para la descripción de texturas, basa su idea en el uso de descripciones locales sobre regiones de un rostro, debido a que dichas regiones pueden aportar más información que otras. Para ello la captura de la imagen facial es dividida en regiones, a cada región se le aplica un histograma con el que se puede obtener un operador denominado LBPH, entonces las descripciones locales son concatenadas para generar una descripción general.

Pasos y metodología.

El método en general asigna etiquetas a cada uno de los píxeles de la imagen tomando en consideración la distribución de sus vecinos, por lo que es necesario:

- Generar una máscara de tamaño determinado que recorra la imagen de manera iterativa seleccionando cada píxel y sus vecinos.
- El píxel central es comparado de manera ordenada con cada uno de los vecinos.
- Se asigna un 1 cada vez que el píxel central sea menor que el píxel comparado, de lo contrario se asignará un 0.
- El resultado (binario) es transformado a número decimal contado en el histograma para formar la descripción (ver 3.4)

$$H_i = \sum_{i=1}^{c} I\,[L \quad (x,y) = i], i = 0, \dots, n-1 \qquad (3.4)$$

$$I(x)\begin{cases}1, & \text{s} \quad x \text{ e} \quad \text{v} \quad . \\ 0, & \text{o} \quad \text{v} \quad .\end{cases}$$

Experimento

Una vez que fue posible generar un flujo de trabajo para eigenfaces/fisherfaces y LBPH, se procedió a probar el reconocimiento de rostros, considerando el Dataset UP (sección 3.1) metodología de la sección 3.2, de lo cual se obtuvieron los siguientes resultados como lo detalla la tabla 3.1.

		Ejecución del código									
		1era		**2da**		**3Era**		**4ta**		**5ta**	
Técnica.	Grupo	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas
Eigenfaces	A	0	10	0	10	0	10	0	10	0	10
	B	0	10	0	10	0	10	0	10	0	10
Fisherfaces	A	0	10	0	10	0	10	0	10	0	10
	B	0	10	0	10	0	10	0	10	0	10
LBPH	A	6	4	5	5	7	3	6	4	5	5
	B	1	9	0	10	2	8	1	9	9	10

Tabla 3.1: Tabla de pruebas sobre algoritmos tradicionales de reconocimiento facial.

En donde se puede observar que los aciertos en reconocimiento promedio mediante (3.5), fue aproximadamente del 58% para Local *Binary Patterns Histograms*.

$$P = \frac{\sum_{i=1}^{n} C_{\ i}}{n} \tag{3.5}$$

Donde P, es el porcentaje de aciertos, Cai es la cantidad de aciertos por ejecución del código y n el número de ejecuciones realizadas, en este caso $n = 5$.

Merece destacar que las eigenfaces y fisherfaces no tuvieron aciertos en la identificación de los rostros aun cuando se trabajó con el mismo dataset.

3.4. Análisis de técnicas basadas en aprendizaje profundo para reconocimiento facial

3.4.1. Tensorflow Object Detection

Pese a que el desarrollo de aplicaciones que sean capaces de localizar e identificar objetos en una misma escena capturada por una fotografía sigue considerándose un reto, el API de Tensorflow para el reconocimiento de objetos permitiría construir, entrenar y desarrollar modelos para la detección de objetos y aprovechar esta característica para el reconocimiento de rostros. A continuación, se presenta el esquema de trabajo realizado para poder reconocer rostros a través del *Tensorflow Object Detection API*, para propósitos de simplificación en lo posterior se hará referencia al acrónimo *TFODAPI* para referirse a dicho API.

Instalación y desarrollo

TFODAPI, requiere de las siguientes librerías Protobuf 3.0.0, Python-tk, Pillow 1.0, lxml tf Slim, Jupyter notebook, Matplotlib, Tensorflow (>=1.9.0), Cython, contextlib2, cocoapi o versiones superiores.

TensorFlow Object Detection API es un marco de trabajo de código abierto construido sobre TensorFlow, por lo cual es necesario descargar el API desde https://github.com/tensorflow/models/archive/master.zip[11]

[11] Hasta el momento del desarrollo del presente trabajo, el sitio se encuentra disponible y en constante mantenimiento y contribución.

Adicionalmente se trabajó sobre COCO API[12] que es un dataset diseñado para detección de objetos, segmentación, detección de puntos clave de personas, segmentación de elementos y la generación de subtítulos.

Una de las características que hacen atractivo a *TFODAPI* es la forma cómo calcula la pérdida por entrenamiento y la forma como se minan los elementos negativos.

Al igual que la mayoría de las arquitecturas, el Tensorflow Object Detection basa sus características de detección en las siguientes características: en primer lugar, *feature extraction* mediante un conjunto de bloques convolucionales y luego la detección con bloques secuenciales de detección. Los mapas de características se reducen de tamaño después de cada bloque convolucional.

Cada uno de los bloques de detección tiene 3 ramas: generación de caja, clasificación y corrección de localización. El primero es responsable de recortar rectángulos (cuadros) de las diversas relaciones de aspecto centradas en nodos de cuadrícula regulares sobre el mapa de características. El segundo es responsable de predecir los puntajes de confianza de *C + 1* (C para todas sus clases + 1 para el fondo) para cada caja generada. El tercero es responsable del ajuste de las posiciones para todas las cajas generadas.

Las salidas de cada bloque de detección se recolectan en la última capa donde ocurre el filtrado inteligente de las predicciones.

La pérdida utilizada para el entrenamiento es una suma de las pérdidas de Clasificación y Localización, esta última calculada solo para cajas con la misma etiqueta de clase que la verdad básica.

Para la presente tesis la Figura 3.4 detalla el procedimiento abreviado desarrollado para la adaptación del modelo e implementación de un reconocedor facial basado en *TFODAPI*.

[12] http://cocodataset.org/

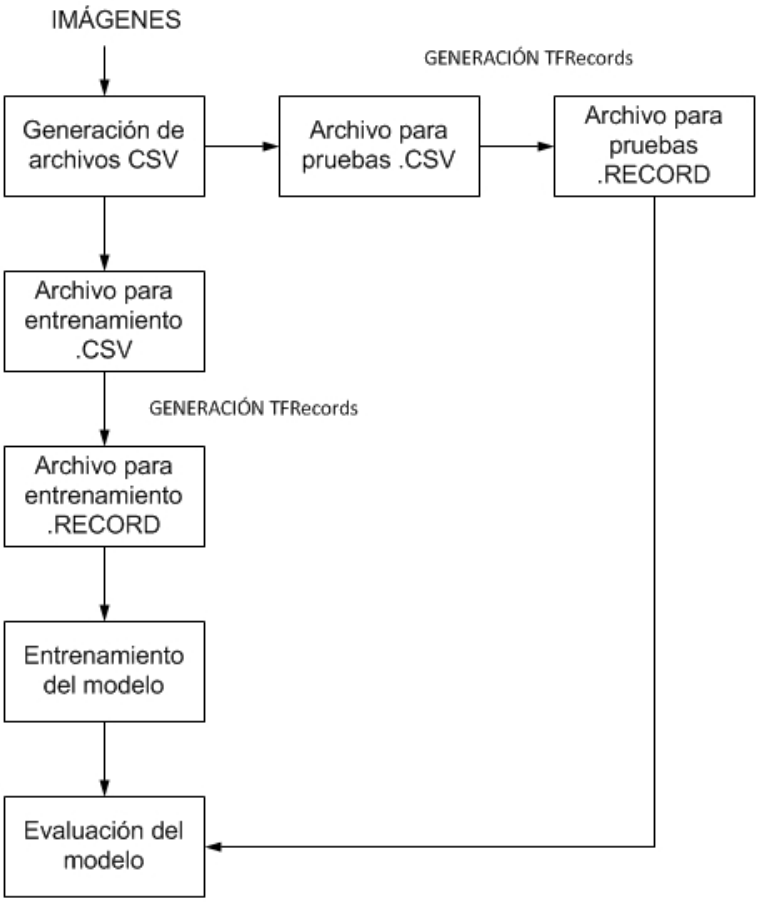


Figura 3.4: Diagrama de bloques del API tensorflow para detección de objectos.

La generación de archivos .CSV como preprocesamiento, se realizó mediante un conjunto de scripts en Python lo que permitió no solo obtener los archivos base, sino que permitió descartar imágenes que, por falla de captura no reúna los aspectos matemáticamente importantes para el entrenamiento del modelo (iluminación, detección de características y tamaño entre los más importantes). Esta adaptación es lo que se denominó flujo *TFODAPI* y se muestra en la Figura 3.5.

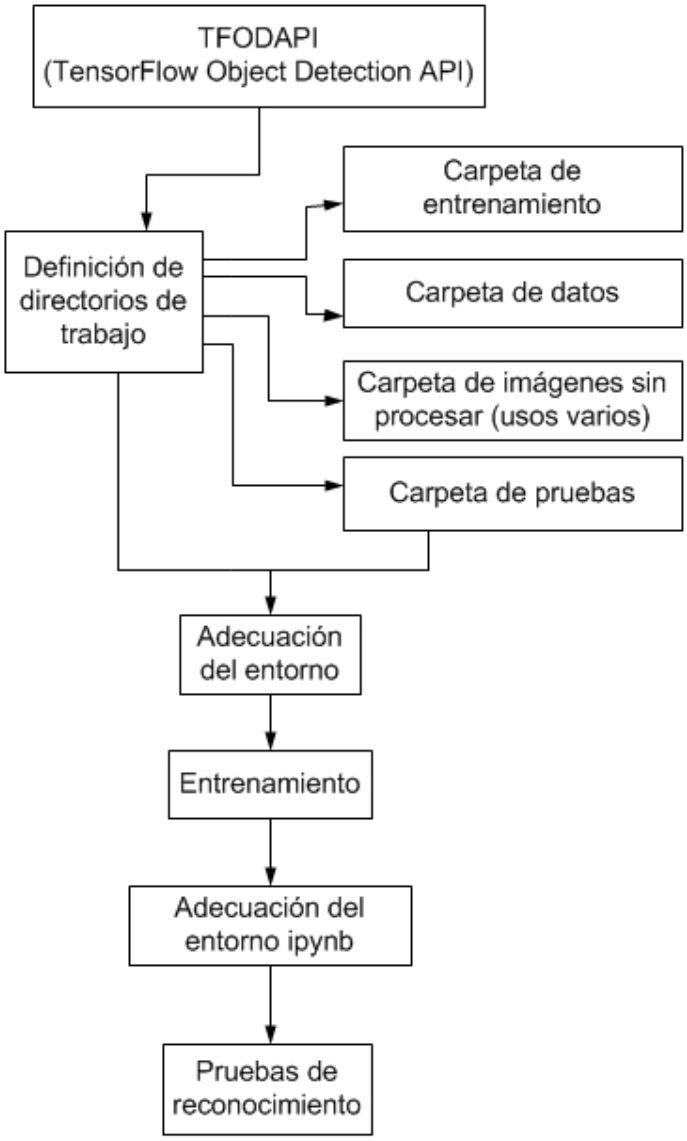


Figura 3.5: Detalle del flujo del TFODAPI aplicado (Tensorflow Object Detection API) para la prueba de reconocimiento facial.

Metodología y resultados

Tomando en consideración el flujo general del *TFODAPI* (Figura 3.4 y Figura 3.5), el Dataset UP (sección 3.1) y metodología (sección 3.2), se muestran resultados obtenidos en la tabla 3.2.

	Ejecución del código									
	1era		2da		3era		4rta		5ta	
	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas
Grupo A	10	0	10	0	10	0	10	0	10	0
Grupo B	0	10	0	10	0	10	0	10	0	10

Tabla 3.2: Valores de aciertos o fallas por ejecución del script de reconocimiento facial en el que las fotografías contaban con al menos un 98% de probabilidades de acertar.

Un ejemplo de la ejecución del programa lo muestra la Figura 3.6.

Figura 3.6: Ejemplos de la captura del resultado de reconocimiento facial implementado con el TFODAPI.

De acuerdo con las pruebas realizadas y considerando el porcentaje de predicción por fotografía expuesta por el código del API de tensorflow Object Detection, se calcula una probabilidad de precisión promedio del 98% de que los sujetos evaluados correspondan a cada etiquetado de prueba, mientras que en el promedio de aciertos calculados con (3.5), se pudo obtener un 100% de aciertos.

Estos dos valores, tanto el porcentaje de predicción como el promedio calculado contrastan con otras pruebas realizadas [74] sobre el detector utilizado, cuya documentación refiere que la confianza declarada para cada objeto detectado (recordar que se realizó una adaptación para reconocimiento facial) supera el 45%.

Para que estos resultados sean comparables, se manejó valores umbrales por defecto del API entre 0.5 y 0.3, adicionalmente un *score_threshold* de 1.e-8 a 0.2, bajo los mismos lineamientos de "*object_detection_tutorial.ipynb*" para los dos grupos de fotografías evaluados.

Un elemento interesante de esta prueba radica en el hecho de que al utilizar el grupo B del cual el modelo no tuvo entrenamiento previo, toda evaluación realizada generó un resultado negativo. El sustento de esta prueba se puede observar en los datos proporcionados por Mei Wang y Weihong Deng [27] quienes indican que: “En términos de protocolo de entrenamiento, el modelo de FR se puede evaluar bajo sujetos dependientes o configuraciones independientes según las identidades de prueba aparezcan en el conjunto de entrenamiento. En términos de tareas de prueba, el rendimiento del modelo de reconocimiento puede ser evaluado bajo verificación de rostros, identificación de rostros ajustados, ajustes de identificación de rostros abiertos”. De ahí que esta misma metodología se aplica al API de Keras y facenet como elemento de control.

Hasta el momento en que se desarrolla la presente tesis se desconoce de modelos y métodos que apliquen la detección de objetos al reconocimiento facial, sin embargo, de la literatura revisada se puede extrapolar por lo menos dos ventajas:

- A diferencia de otras aplicaciones para detección de objetos en una escena en la que los modelos pre-entrenados resultaban ineficientes [75], la aplicación del API de tensorflow para reconocimiento facial se implementó sin ningún tipo de afinamiento y con casos de éxito del 100%.
- Pese a que no existe un referente exacto sobre reconocimiento facial mediante el *TFODAPI*, los resultados obtenidos se asemejan a pruebas similares realizadas con el mismo detector [30] [76] , en las que se obtuvieron éxitos entre el 0.5 y 0.95% de casos.

3.4.2. Implementación Keras del retinanet object detection

De manera similar al modelo explicado en la sección 3.4.1. es bueno indicar que los detectores de objetos que ostentan mayor precisión se basan en enfoques de dos etapas (común en R-CNN), esto es la aplicación de un clasificador para objetos relativamente dispersos, a diferencia de detectores de una etapa que se aplican en un muestreo denso y regular de posibles ubicaciones de objetos, los cuales tienen el potencial de ser más rápidos y más simples, no así más precisos,

es por esta clara diferencia que basado en la implementación Keras del *Retinanet Object Detection* de [77] se adecuó este API para la detección de rostros.

Su funcionamiento se basa en el detector *Retinanet* que a su vez es una red única y unificada compuesta por una red troncal y dos subredes específicas. La columna principal es responsable de calcular un mapa de características convolucionales sobre una imagen de entrada completa. La primera subred realiza la clasificación de objetos convolucionales en la salida del backbone; la segunda subred realiza una regresión por cuadro de límite convolucional. Las dos subredes presentan un diseño simple que sus autores proponen como una red detección densa en una etapa. En este contexto se trabajó con el dataset de 857 imágenes (41 individuos distintos), el pipeline generado para este modelo se puede apreciar en la Figura 3.7.

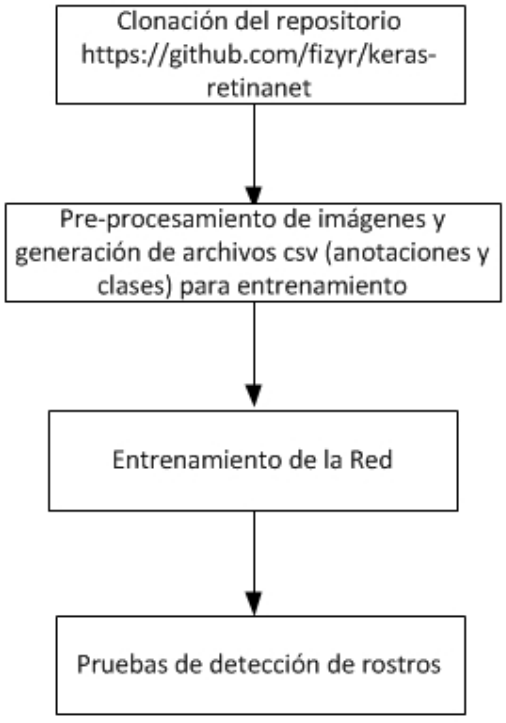


Figura 3.7: Proceso de pruebas de reconocimiento facial basado en Keras RetinaNet.

El procedimiento de entrenamiento de "*keras-retinanet* " funciona con modelos de entrenamiento y debido a que solo contienen las capas necesarias para el entrenamiento (valores de regresión y clasificación) el modelo es ligero de entrenar. Pese a esto con un computador Core I7 de 8 Gb de RAM con procesador gráfico NVIDIA GEFORCE se logró obtener un promedio de entrenamiento de 5 horas con 15 minutos, debido a la naturaleza de la propuesta no se planificó pruebas en equipos de mayores capacidades sin embargo se hizo pruebas en equipos con GPU 1080 ti, 32GB de RAM, e i7 como procesador en los que se consiguió promedios de 1 hora de entrenamiento.

Metodología y resultados

Para mantener la consistencia del experimento se utilizó los dos grupos de imágenes (grupo A y grupo B) declarados previamente, de lo cual se obtuvieron los siguientes resultados detallados en la Tabla 3.3.

	Ejecución									
	1era		**2da**		**3era**		**4ta**		**5ta**	
	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas
Grupo A	1	9	2	8	1	9	1	9	1	9
Grupo B	3	7	2	8	1	9	1	9	1	9

Tabla 3.3: Resultados de la ejecución de código de reconocimiento en el que se puede apreciar la cantidad de aciertos y fallas por ejecución y grupos de pruebas.

En promedio, la precisión de reconocimiento según la predicción marcada por rostro evaluado es de alrededor del 85%, mientras que en el promedio de aciertos calculados basado en (3.5), es del 12%, un ejemplo de la ejecución del modelo lo muestra la Figura 3.8.

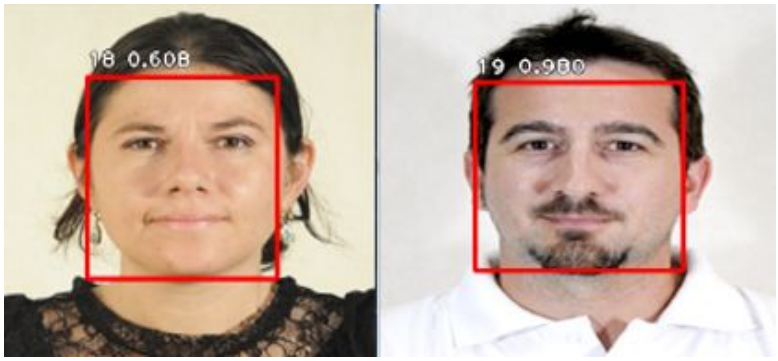


Figura 3.8: Ejemplo de la ejecución del código de reconocimiento para casos positivos, el cuadro muestra la parte de detección del rostro, el número de la persona y la precisión.

De manera similar a lo expuesto en la sección anterior, hasta el momento de la redacción de la presente tesis, no existen datos de la evaluación de este modelo con respecto a reconocimiento facial, mas de las pruebas realizadas por los proponentes del uso de *retinanet* sobre la identificación de objetos [77] , no supera una precisión del 41% en la demarcación de los mismos. Con este antecedente y con los resultados de las pruebas realizadas, se puede argumentar que el modelo de implementación Keras del *Retinanet Object Detection,* tiene en base al dataset evaluado, menor precisión que el *TFODAPI.*

3.4.3. Facenet y openface

Facenet es una implementación basada en tensorflow como lo describe “FaceNet: A Unified Embedding for Face Recognition and Clustering” [9]. Al igual que implementa ideas de "Deep Face Recognition" [61], mediante la inspiración de OpenFace [62]. Facenet utiliza como dataset CASIA-WebFace[13] para entrenamiento. De acuerdo a David SandBerg (su autor) se utilizaron 453453 imágenes sobre 10575 identidades para su posterior detección, el mismo Sandberg explica que obtuvo mejores resultados en el rendimiento del modelo con el dataset VGGFace2. El cuál contiene alrededor de 3.3 millones de rostros y alrededor de 9000 clases.

[13] El sitio web dispone de varios datasets que se pueden utilizar de forma abierta citando su origen, mismos que pueden ser ubicados en http://www.cbsr.ia.ac.cn/

Las fases de trabajo de *FaceNet* son: pre-pocesamiento, entrenamiento y evaluación de rendimiento, para el propósito de esta investigación, se diseñó un pipeline que permite evaluar la precisión de lo propuesto por Sandberg el cual se detalla en la Figura 3.9.

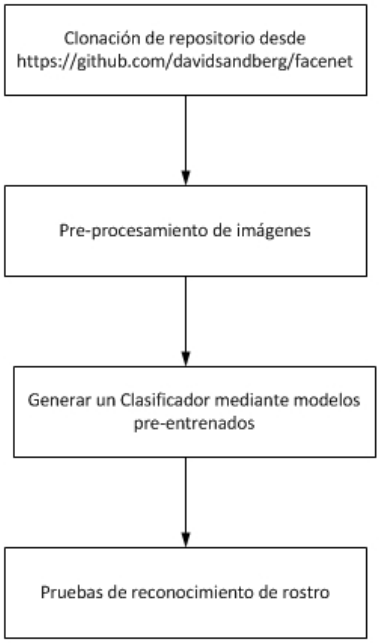


Figura 3.9: Pipeline para pruebas de reconocimiento con Facenet

Preprocesamiento

Consistió en detectar el rostro y recortarlo de manera estándar (160 x 160) para cada una de las identidades y fotografías de las clases. La mayoría de implementaciones de reconocimiento facial han recurrido a la aplicación de librerías como *dlib* [63], sin embargo de las pruebas que se han realizado para la implementación de *faceNet*; dlib como detector de rostros, adolece de algunos problemas relacionados con oclusiones parciales, siluetas, movimientos de cabeza y objetos relacionados con el rostro, lo que hace que en los conjuntos de entrenamiento se reduzcan considerablemente por el número de imágenes ignoradas, según las palabras de Sandberg en su mismo repositorio. Pese a esto,

en las pruebas simples realizadas con *dlib* para la presente tesis, se determinó que no existe diferencias significativas en la detección de rostros por lo que el pipeline propuesto puede aplicarse tanto para dlib como para "*multi-task cnn*" [64] siendo este último el más utilizado por David Sandberg debido a que resuelve los problemas de oclusión e iluminación en la detección de rostros mediante la aplicación de redes convolucionales en cascada.

Adicionalmente K. Zhang, Z.Zhang, Z. Li et al. en [80] proponen para *facenet* una estrategia de minería de muestras duras en línea que mejora aún más el rendimiento en la práctica. El método en sí de acuerdo a su autores, logra una precisión superior a las técnicas modernas en puntos de referencia FDDB[14] y WIDER FACE[15] para la detección de rostros, y el punto de referencia AFLW [65] para la alineación de rostros, al tiempo que mantiene el rendimiento en tiempo real. Facenet presenta dos implementaciones de pre-procesamiento, una realizada con Matlab/Caffe la cual está fuera del alcance de esta investigación y otra con TensorFlow misma que se utilizó en el pipeline de evaluación del presente trabajo.

Entrenamiento y generación de clasificadores

De acuerdo con lo planteado por David SandBerg los mejores resultados obtenidos dependían de la aplicación de la función "*softmax loss*", el modelo *Inception-Resnet-v1* y la *Multi-Task CNN*. Para el caso de generar el clasificador se utilizó una red pre-entrenada (20180408-102900) con una arquitectura *Inception Resnet V1* que permitiría obtener una precisión de 0.9905 con un dataset *CASIA-WebFace*.

[14] Face Detection Data Set and Benchmark, un DataSet de regiones faciales diseñado para estudiar el problema de la detección facial sin restricciones, se lo puede encontrar en http://vis-www.cs.umass.edu/fddb/

[15] http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/

Metodología y resultados

De manera similar a los experimentos anteriores se utilizaron dos grupos (A y B) para pruebas y control respectivamente como lo detalla la tabla 3.4.

	Ejecución									
	1era		**2da**		**3era**		**4ta**		**5ta**	
	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas	Aciertos	Fallas
Grupo A	7	3	6	4	4	6	8	2	7	3
Grupo B	3	7	1	9	1	9	2	8	2	8

Tabla 3.4: Resultados de las pruebas realizadas con Facenet, se destaca que los fallos son falsos positivos.

A pesar de que en el repositorio de David Sandberg se estima una precisión de 0.99650+-0.00252 con el dataset LFW y *CASIA-WebFace*, de las pruebas obtenidas en la presente investigación se alcanzó una efectividad de alrededor del 60% basado en (3.6) para aciertos de las imágenes del dataset de entrenamiento.

$$E = \frac{\sum_{i=1}^{n} A_i}{N} \tag{3.6}$$

Donde E es la efectividad del reconocimiento evaluado, A_i es el número de veces que el modelo acertó en reconocer la imagen correctamente y Np el número de ejecuciones. En el caso del grupo de control, se obtuvo un significativo 10%, que permitirá un análisis posterior una vez se comparen los resultados en general con los otros modelos evaluados.

Es necesario explicar que la diferencia en cuanto a precisión con el dataset LFW se debe a:

- Diferencias sustanciales de LFW/*CASIA-WebFace*, con el tamaño del Dataset UP, utilizado en la evaluación del modelo.
- LFW/*CASIA-WebFace*, son considerados dataset estandarizados mientras que el Dataset UP es un dataset con un conjunto de imágenes diversas sin estandarización, esto implica que el preprocesamiento de imágenes en el pipeline pudo haber afectado la calidad de las mismas, por

ende afectar la recopilación de las características de los rostros. Pese a esto comparativamente los resultados son mejores para los grupos A y B con respecto a los otros modelos y técnicas.

- Similar al punto anterior LWF/*CASIA-WebFace*, fue recopilado con condiciones similares de iluminación, mientras que el Dataset UP, fue desarrollado con imágenes bajo diferentes condiciones de iluminación.

Estas perspectivas, permiten especular que un modelo basado en facenet/openface podría generar resultados totalmente distintos a la precisión expresado por sus autores para el caso de reconocimientos en tiempo real. Lo que no necesariamente implica que el modelo sea un modelo de mala calidad, por el contrario, en las pruebas realizadas en esta tesis, sus resultados tuvieron una ligera ventaja con respecto a los otros modelos evaluados.

Comparación General

Se procedió a comparar distintos modelos de reconocimiento facial, los cuales basan sus decisiones de reconocimiento en redes pre-entrenadas cuya estructura de clasificación e identificación es diferente, un resumen de los modelos comparados en el presente análisis se detalla en la Tabla 3.5.

Modelo Evaluado	**Arquitectura**
EigenFaces	Arreglos Numéricos
Fisherfaces	Arreglos Numéricos
LBPH	Arreglos Numéricos
TFODAPI	Resnet 101
Keras Retinanet	Retinanet
Facenet	Inception/ResnetV1

Tabla 3.5: Resumen de redes pre-entrenadas comparadas en el reconocimiento de rostros.

Por lo tanto, de las pruebas realizadas se determinó que, para el caso de modelos tradicionales de reconocimiento facial, se impone los algoritmos basados en *local binary patterns histograms (LBPH)* por sobre los *eigenfaces* y *fisherfaces*.

En el caso de técnicas que implementan aprendizaje profundo, si bien el API de *tensorflow* muestra ventaja sobre los modelos basados en *API Keras RetinaNet* y *openface (facenet)*, como lo muestra la Tabla 3.6. Es necesario destacar que Facenet tuvo mejores resultados con el grupo B, tanto por una ligera ventaja en aciertos como por una menor cantidad de fallas, lo que se podría considerar como un modelo destinado a ser implementado en aplicaciones de tiempo real.

	Grupo A		**Grupo B**	
	Aciertos	**Fallas**	**Aciertos**	**Fallas**
EigenFaces	0	10	0	10
Fisherfaces	0	10	0	10
LBPH	5.8	4.2	1	9
TFODAPI	10	0	0	10
Keras ODA	1.2	8.8	1.6	8.4
Facenet	6.4	3.6	1.8	8.2

Tabla 3.6: Cuadro comparativo de resultados de ejecución de modelos de reconocimiento facial.

Las Figuras 3.10 – 3.13 mejoran la perspectiva del análisis numérico presentado.

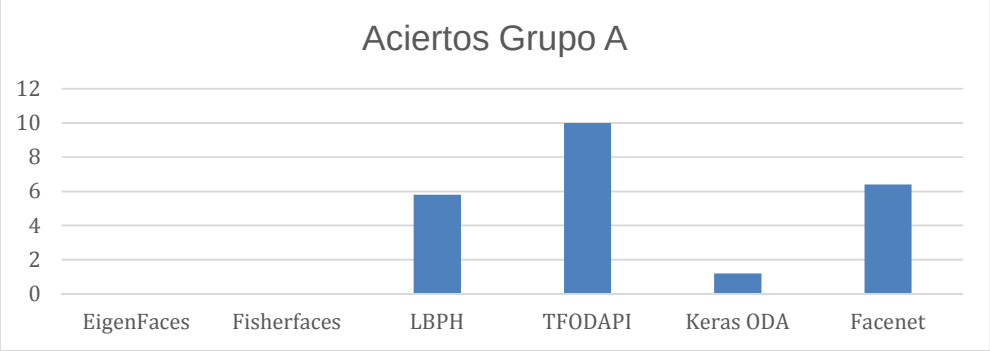


Figura 3.10: Representación gráfica de los aciertos presentados en la prueba de reconocimiento facial.

En contraste con estos resultados, para el grupo B (Dataset con imágenes que no guardan relación con los entrenamientos de los modelos) se puede observar que (Figura 3.13) *facenet* tiene el menor número de falsos positivos/fallas al momento de reconocer e identificar rostros.

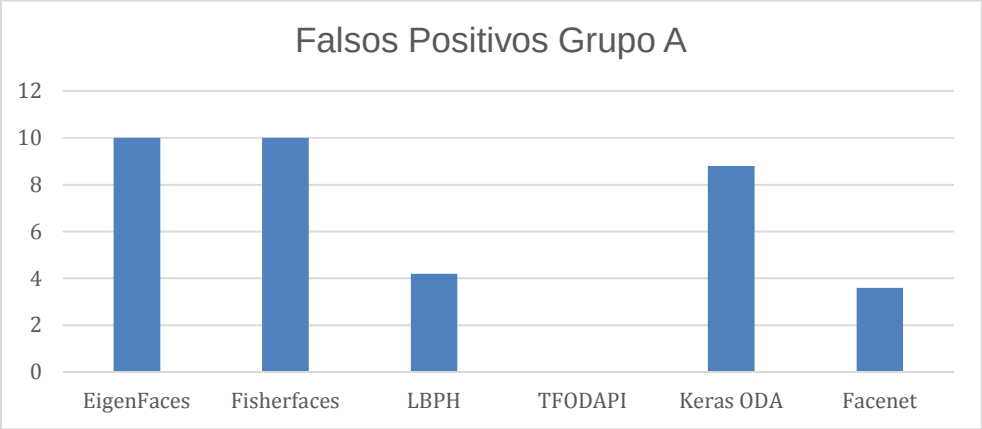


Figura 3.11: Gráfica que representa la cantidad de falsos positivos obtenidos en las pruebas de reconocimiento facial por modelo.

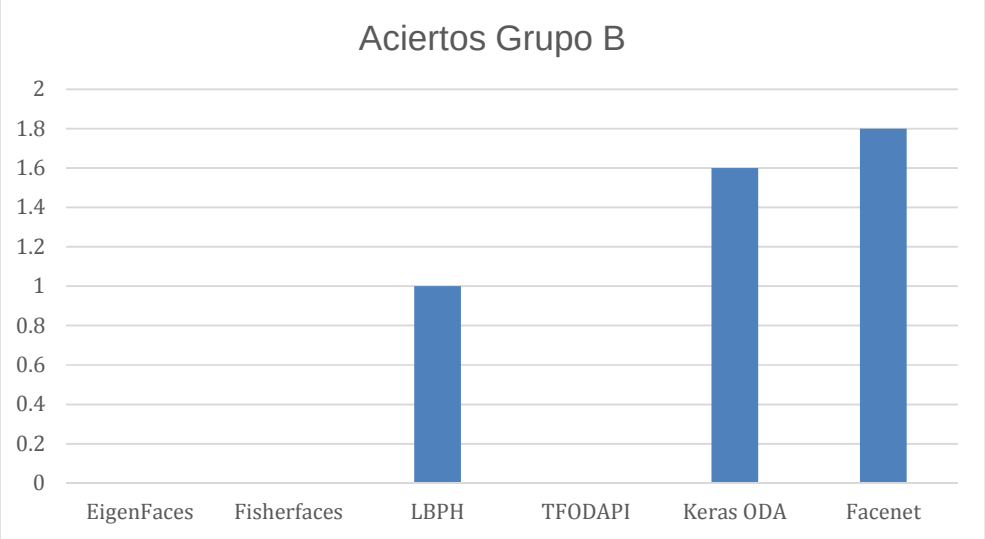


Figura 3.12: Gráfica que representa la cantidad de aciertos obtenidos en las pruebas de reconocimiento facial por modelo para el grupo de imágenes no utilizadas en entrenamiento.

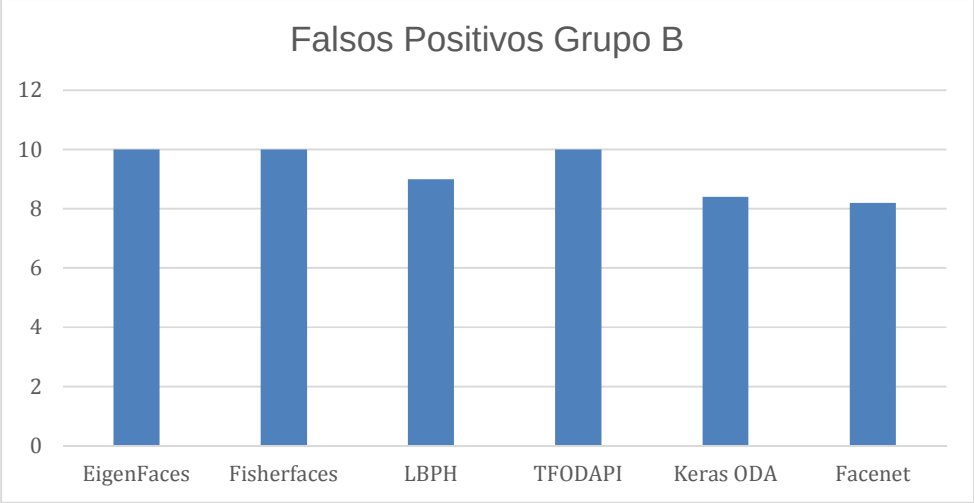


Figura 3.13: Gráfica que representa la cantidad de falsos positivos obtenidos en las pruebas de reconocimiento facial por modelo en el grupo B.

CAPITULO 4

Por el momento es posible clasificar las técnicas de reconocimiento facial en dos grandes grupos, los basados en técnicas tradicionales y los que se basan en la aplicación de técnicas aprendizaje profundo, para la presente tesis se escogió 3 modelos tradicionales basados en eigenfaces, fisherfaces e histogramas de patrones locales binarios, mientras en que en cuanto a la aplicación de técnicas de aprendizaje profundo se escogió al API de detección de objetos de tensorflow bajo resnet 101, el detector retinatet mediante Keras y Facenet con openface que trabaja sobre inception/resnetV1.

4. Resultados y Trabajo Futuro

4.1. Resultados

A continuación, se presentan los resultados de la evaluación de los experimentados realizados sobre los 6 modelos evaluados y que se desarrollaron en Python. Mencionados experimentos fueron realizados sobre la base de un Dataset de 857 imágenes recopiladas de distintas fuentes en línea de los docentes de la Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, esto es debido a que se quiere evaluar la precisión de un modelo basado en la capacidad de reconocer e identificar un rostro de una imagen cuya captura represente el único dato disponible del individuo a reconocer.

Esto incrementó de acuerdo a la literatura examinada, la complejidad de la capacidad de poder identificar un rostro humano debido a la poca estandarización de imágenes, diferencias en cuanto a luminancia y crominancia de las mismas, y las resoluciones y objetos adicionales extraños al rostro en evaluación, generando en el reconocimiento facial un verdadero desafío, contrario a las pruebas que los mismos desarrolladores de los modelos realizaron, en las que se empleaban Datasets debidamente estandarizados para tal efecto.

4.1.1. Experimento: Grupo A

Se denominó grupo A al conjunto de 10 fotografías tomadas aleatoriamente del grupo de fotografías que sirvieron para entrenar los distintos modelos, entre los que constan: Modelos Tradicionales (EigenFaces, Fisherfaces, Histogramas de Patrones Binarios), API de TensorFlow para reconocimiento de objetos que fue acondicionado para reconocer rostros, API de Keras para reconocimiento de objetos acondicionado para reconocimiento facial y finalmente una implementación denominada Facenet que presenta los distintos beneficios de redes neuronales convolucionales. Su elección obedece no solo a la forma de detección de objetos sino a que actualmente de acuerdo a la literatura investigada, se consideran los modelos sobresalientes sobre reconocimiento de objetos y que permiten al momento constituirse como el estado del arte en reconocimiento facial.

4.1.2. Experimento: Grupo B (control)

De acuerdo con la literatura establecida para el presente estudio, se determinó que los modelos más eficientes en cuanto a reconocimiento facial se encontraban del lado de técnicas en las que fuera posible un entrenamiento no convencional de las imágenes. Ante este precedente, se volvió predecible el comportamiento de los modelos evaluados a tal punto que los modelos basados en técnicas de aprendizaje inteligente no supervisado superaban ampliamente a las técnicas tradicionales, por eso fue necesario repetir los experimentos con un dataset de control, el cual estaba conformado por un conjunto de imágenes de 240x 230 de resolución, que fueron obtenidas del proceso de carnetización llevado a cabo en la Universidad Estatal Península de Santa Elena. Las imágenes contienen en general un individuo con una captura frontal con variaciones de luminancia y crominancia típicas de una fotografía de estudio, lo que resulta en una fotografía de fácil identificación del representado para el ser humano.

4.1.3. Comparativas

La Tabla 4.1 muestra los resultados de los experimentos realizados sobre la ejecución de los modelos en dos grupos de imágenes.

		Grupo A		Grupo B	
		Aciertos	**Errores**	**Aciertos**	**Errores**
Tradicional	**EigenFaces**	0%	100%	0%	100%
	Fisherfaces	0%	100%	0%	100%
	LBPH	60%	40%	10%	90%
Aprendizaje Profundo	**TFODAPI**	100%	0%	0%	100%
	Keras ODA	12%	90%	20%	80%
	Facenet	60%	40%	20%	80%

Tabla 4.1: Resultados de los experimentos sobre imágenes de los Grupos A y B, los detalles se presentan en el texto precedente.

Si bien estos resultados, pueden diferir de muchos estudios similares como lo muestra la Tabla 4.2, es necesario tomar en cuenta algunos factores que pudieron haber incidido en los resultados.

		Porcentaje de éxito según literatura relacionada	**Referencia**	**Porcentaje de éxito según evaluaciones realizadas**
Tradicional	**EigenFaces**	92-100%	[72]	0%
	Fisherfaces	85.75%	[82]	0%
	LBPH	88.73%	[8]	60%
Aprendizaje Profundo	**TFODAPI**	45% (valor promedio de detección de objetos)	[74]	100%
	Keras ODA	40.8% (valor promedio de detección de objetos)	[77]	10%
	Facenet	99.63%	[83]	60%

Tabla 4.2: Comparativa de trabajos relacionados sobre experimentos similares, se tomó los valores declarados como promedio por los autores de las respectivas implementaciones

Factores como:

- Las imágenes fueron recolectadas de distintas fuentes sin ninguna estandarización de crominancia y luminancia.
- El equipo utilizado para entrenamiento de las redes neuronales es un equipo que si bien contiene un GPU que permitiría procesamiento paralelo, en la mayoría de los casos se trabajó directamente con CPU, con 8Gb de RAM, por lo que fue necesario reducir la cantidad de etapas y tamaños de los procesos por lotes, lo que podría verse reflejado en la calidad de los modelos de entrenamiento.
- Los tiempos de entrenamiento no fueron considerados dentro del estudio.

Los elementos previos reflejan la realidad de los entornos en los que futuras aplicaciones de reconocimiento facial pudieran implementarse, por lo que sólo se consideró dentro del experimento las veces que el algoritmo acertó/falló en identificar un determinado rostro.

Sin afectar a lo enunciado previamente, se puede entender que el pipeline desarrollado para *facenet*, presenta la mayor cantidad de aciertos y la menor cantidad de errores en relación con los demás métodos evaluados.

4.2. Trabajos futuros

Considerando las diferencias entre las evaluaciones realizadas en la presente tesis y los resultados expuestos por los respectivos autores de los modelos evaluados, se plantean como posibles trabajos futuros:

- Estudio con énfasis en la evaluación del API de Tensor Flow Vs Facenet bajo las similares condiciones de la investigación precedente; esto es, imágenes que no superan los 300 x 400 de resolución en promedio y seleccionadas indistintamente de fuentes públicas como redes sociales, considerando para este caso el aumento de imágenes por clase, al igual que el uso de GPU en la valoración de la duración del preprocesamiento de imágenes. Adicionalmente, el estudio comparativo de los modelos inferenciales utilizados para la transferencia de aprendizaje, a fin de determinar el mejor modelo y red entrenada para condiciones hostiles de evaluación.
- En base a lo anterior y ante la compra por parte la Facultad de equipos de vigilancia *IP eye 380*, se plantea a futuro el desarrollo de una aplicación en tiempo real que permita la identificación de personas, en el que el diseño experimental se base en la cantidad de falsos positivos y negativos encontrados por unidades de tiempo, para ello se plantea la generación de un registro de aciertos y una selección de 5 grupos cada uno de 10 personas previamente identificadas, los cuales posarán indistintamente delante de la cámara IP.

CONCLUSIONES

En el presente trabajo se analizaron 2 tendencias en cuanto a formas de reconocimiento facial lo que implicó la elección de 6 modelos, tres modelos tradicionales y tres modelos sobre aprendizaje profundo, con el objetivo de identificar el método que pueda resultar de mayor precisión sobre Datasets simples y sin ninguna estandarización, a través de experimentación simple y directa sobre equipos de baja capacidad. A diferencia de las pruebas que muestra la literatura actual para las que se han desarrollado conjuntos de imágenes adaptadas a las condiciones a probar y que se desarrollan sobre equipos de grandes prestaciones. Bajo este contexto y al final del desarrollo de la presente tesis se concluye:

1. Los modelos basados en aprendizaje profundo mostraron mejor precisión al momento de identificar rostros que pertenecen al dataset de entrenamiento, lo que brinda en promedio una precisión del 60% contra el +/- 90% estimado en las publicaciones actuales, esto es justificable por el tipo y calidad de dataset utilizado.

2. Se evidenciaron mejores resultados cuando la cantidad de imágenes dentro de cada clase mostraba al menos un conjunto de 10 imágenes por clase, donde cada imagen debería contener diferencias sustanciales con respecto al resto de imágenes de su respectiva clase (individuo).

3. No se obtuvieron resultados significativos al aumentar el número de clases (individuos) de prueba, por lo que puede inferirse que el número de clases no afecta al proceso de identificación sino al proceso de individuos a identificar.

4. Los modelos basados en la detección de objetos y que fueron adaptados al reconocimiento facial fueron superiores solo para los casos en los que las imágenes consideradas en los experimentos pertenecían al conjunto de imágenes de entrenamiento, para estos casos aumentar el tamaño del Dataset podría mejorar la precisión de identificación, pero resulta poco aplicable a condiciones en las que se requiera mayor inferencia sobre el rostro a identificar.

BIBLIOGRAFÍA

[1] G. Williams, “Breve historia del reconocimiento facial”, *web*, 2018. [Online]. Available: https://www.facefirst.com/blog/breve-historia-del-reconocimiento-facial/. [Accessed: 21-Dec-2018].

[2] F. Dornaika, Ed., *Advances in Face Image Analysis: Theory and Applications*. BENTHAM SCIENCE PUBLISHERS, 2016.

[3] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, Jan. 1991.

[4] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition,” in *Procedings of the British Machine Vision Conference 2015*, 2015.

[5] J. A. Stick, R. F. Slocombe, F. J. Derksen, and E. A. Scott, “face recognition survey,” *Am. J. Vet. Res.*, vol. 44, no. 11, pp. 2123–2132, 1983.

[6] D. G. Lowe, “Object Recognition from Local Scale-Invariant Features,” *Proc. Int. Conf. Comput. Vision-Volume 2 - Vol. 2. ICCV ’99*, pp. 35–40, 1999.

[7] M. H. Yang, D. J. Kriegman, and N. Ahuja, “Detecting faces in images: A survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

[8] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, “Local binary patterns and its applications on facial image: A survey,” *IEEE Trans. Syst. Man, Cybern. C Appl. Rev.*, vol. 41, no. 6, pp. 765–781, 2011.

[9] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” *Nature*, vol. 521, no. 7553, p. 800, 2016.

[10] R. Gonzalez and R. E. Woods, “Digital Image Processing.” Prentice Hall, p. 797.

[11] J. Tucker, “How facial recognition technology came to be,” 2014.

[12] J. H. Munson, R. O. Duda, and P. E. Hart, “Experiments with Highleyman’s Data,” *IEEE Trans. Comput.*, vol. C-17, no. 4, pp. 399–401, Apr. 1968.

[13] R. O. Duda and P. E. Hart, “Experiments in the recognition of hand-printed text, part II,” in *Proceedings of the December 9-11, 1968, fall joint computer conference, part II on - AFIPS ’68 (Fall, part II)*, 1968, p. 1139.

[14] F. Schroff and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering.”

[15] M. Kawulok, M. E. Celebi, and B. Smolka, *Advances in Face Detection and Facial Image Analysis*. Cham: Springer International Publishing, 2016.

[16] M. Yang, D. J. Kriegman, S. Member, and N. Ahuja, “Detecting faces in images a survey.pdf,” vol. 24, no. 1, pp. 34–58, 2002.

[17] J. Hu, H. Yan, and M. Sakalli, “Locating head and face boundaries for head-shoulder images,” *Pattern Recognit.*, vol. 32, no. 8, pp. 1317–1333, 1999.

[18] W.-L. Chao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face Recognition, A Literature Survey,” 2003.

[19] X. Wang, “Face Recognition,” *SpringerReference*, no. 1, pp. 1–5, 2011.

[20] T. Kanade, “Picture Processing System by Computer Complex and Recognition of human Faces,” *Dep. Inf. Sci. Kyoto Univ.*, 1973.

[21] R. Brunelli and T. Poggio, “Face recognition through geometrical features,” Springer, Berlin, Heidelberg, 1992, pp. 792–800.

[22] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[23] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, “Face Recognition by Elastic Bunch Graph Matching * †,” CRC Press, 1999.

[24] K. Messer *et al.*, “Performance Characterisation of Face Recognition Algorithms and Their Sensitivity to Severe Illumination Changes.”

[25] T. Ahonen, A. Hadid, and M. Pietikäinen, “Face Recognition with Local Binary Patterns.”

[26] A. M. Martinez and A. C. Kak, “PCA versus LDA,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, 2001.

[27] M. Wang and W. Deng, “Deep Face Recognition: A Survey.”

[28] J. Xu and P. Vidal, “Deep neural networks an introduction,” *on line*, 2018. [Online]. Available: https://www.deeplearningitalia.com/guia-para-arquitecturas-de-redes-profundas/. [Accessed: 05-Nov-2018].

[29] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017.

[30] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” Dec. 2015.

[31] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection via Region-based

Fully Convolutional Networks," May 2016.

[32] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] G. Seif, "Deep Learning vs Classical Machine Learning – Towards Data Science," 2018. [Online]. Available: https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa. [Accessed: 01-Dec-2018].

[34] A. Rosebrock, "Deep Learning for Computer Vision with Python Starter Bundle 1st Edition (1.2.2)."

[35] J. Geake, "Neuromythologies in education," *Educ. Res.*, vol. 50, no. 2, pp. 123–133, 2008.

[36] M. L. Novo, Á. Alsina, J. M. Marbán, and A. Berciano, "Connective intelligence for childhood mathematics education," *Comunicar*, vol. 25, no. 52, pp. 29–39, 2017.

[37] D. H. Freedman and D. H. Freedman, *Los hacedores de cerebros : cómo los científicos están perfeccionando las computadoras, creando un rival del cerebro humano*. Editorial Andres Bello, 1996.

[38] W. S. McCulloch and W. H. Pitts, "originally published in: Bulletin of Mathematical Biophysics, Vol. 5, 1943, p. 115-133," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[39] F. Rosenblatt, "A probabilistic model for information storage and organization in the brain.," *Procedimentos Distrib. Energ. Elétrica no Sist. Elétrico Nac. – PRODIST*, vol. 65, no. 6, pp. 386–408, 1957.

[40] M. Minsky and S. Papert, "R69-13 Perceptrons: An Introduction to Computational Geometry," *IEEE Trans. Comput.*, vol. C-18, no. 6, pp. 572–572, 1969.

[41] D. C. Ruiz and G. Q. Gavino, "Aplicación del algoritmo Backpropagation de redes neuronales para determinar los niveles de morosidad en los alumnos de la Universidad Peruana Unión construction of the respective model and its validation . The methodology for this work was the CRISP - DM," pp. 21–31, 2011.

[42] Y. LeCun *et al.*, "Backpropagation Applied to Handwirtten Zip Code

Recognition.” .

[43] R. M. Haralick and K. Shanmugam, “Textural Features for Image Classification,” *IEEE Trans. Syst. Man, Cybern. SMC-3.6 (Nov. 1973*, 1973.

[44] M.-K. Hu, “Visual Pattern Recognition by Moment Invariants,” *Ire Trans. Inf. Theory*, pp. 66–70, 1962.

[45] A. Khotanzad and Y. H. Hong, “Invariant Image Recognition by Zernike Moments,” *S V Sound Vib.*, vol. 8, no. 12, pp. 28–30, 1990.

[46] J. Huang, S. Ravi Kumar, M. Mitra, W. Zhu, and R. Zabih, “Image Indexing Using Color Correlograms,” *Nouv. Rev. Fr. Hematol.*, vol. 12, no. 6, pp. 871–888.

[47] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. II, pp. 1508–1515, 2005.

[48] C. Harris and M. Stephens, “A combined corner and edge detector,” pp. 147–151.

[49] H. Bay, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Comput. Vis. Image Underst*, 2008.

[50] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” *Proc. 11th Eur. Conf. Comput. Vis. Part IV. ECCV’10*, 2010.

[51] E. Rublee, V. Rabaud, and K. Konolige, “ORB : an efficient alternative to SIFT or SURF About local feature and matching Motivation oFAST – Oriented FAST BRIEF (Calonder et al . 2010),” *Proc. 2011 Int. Conf. Comput. Vision. ICCV ’11. Washington, DC*, pp. 1–5, 2011.

[52] B. Triggs and N. Dalal, “Histograms of Oriented Gradients for Human Detection,” *Comput. Vis. Adv. Res. Dev.*, 2007.

[53] E. S. Teaching, “CS231n Convolutional Neural Networks for Visual Recognition,” *http://vision.stanford.edu/teaching/cs231n/*, 2018. [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 10-Nov-2018].

[54] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015.

[55] P. Tahmasebi and A. Hezarkhani, “Application of a Modular Feedforward

Neural Network for Grade Estimation," *Nat. Resour. Res.*, vol. 20, no. 1, pp. 25–32, 2011.

[56] D. B. Pagès and A. Bonafonte Cavez, "Recurrent Neural Networks for Speaker Dependent Language Modeling Work completed in M*Modal and supervised by Center for Language and Speech Technologies and Applications (TALP)," 2014.

[57] M. Humphrys, "Single-layer Neural Networks (Perceptrons)," *https://www.computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html*. [Online]. Available: https://www.computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html. [Accessed: 10-Nov-2018].

[58] A. Menshawy, *Deep learning by example : a hands-on guide to implementing advanced machine learning algorithms and neural networks*. .

[59] R. Tadeusiewcz, M. Bernacki, and P. Włodarczyk, "Principles of training multilayer neural netwrok using backpropagation," *Editado 2005*, 1992. [Online]. Available: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html. [Accessed: 15-Jun-2018].

[60] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*. 2015.

[61] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015.

[62] C. Szegedy *et al.*, "Going deeper with convolutions."

[63] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision."

[64] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," 2014.

[65] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning."

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition."

[67] K. Hornik, "Approximation capabilities of multilayer feedforward networks,"

Neural Networks, vol. 4, no. 2, pp. 251–257, Jan. 1991.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.”

[69] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway Networks,” May 2015.

[70] S. Hochreiter and J. J. Urgen Schmidhuber, “LONG SHORT-TERM MEMORY,” 1997.

[71] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” Mar. 2015.

[72] M. Çarıkçı and F. Özen, “A Face Recognition System Based on Eigenfaces Method,” *Procedia Technol.*, vol. 1, pp. 118–123, 2012.

[73] C. Esparza, C. Tarazona, E. Sanabria, and D. Velazco, “RECONOCIMIENTO FACIAL BASADO EN EIGENFACES, LBHP Y FISHERFACES EN LA BEAGLEBOARD-xM,” *Rev. Tecnológica Colomb. Av.*, vol. 10, no. 20, 2015.

[74] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” Nov. 2016.

[75] P. Mustamo, “Pirkko Mustamo Object detection in sports: TensorFlow Object Detection API case study,” 2018.

[76] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “DSSD : Deconvolutional Single Shot Detector,” Jan. 2017.

[77] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” Aug. 2017.

[78] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, “OpenFace: A general-purpose face recognition library with mobile applications,” 2016.

[79] D. E. King, “Dlib-ml: A Machine Learning Toolkit,” 2009.

[80] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[81] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof, “Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization *.”

[82] C.-Y. Zhang and Q.-Q. Ruan, “Short Paper: Face Recognition Using L-Fisherfaces *,” 2010.

[83] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.