Escuela Politécnica Superior de Jaén
Máster Universitario en Ingeniería Mecatrónica

**UNIVERSIDAD DE JAÉN**
*Centro de Estudios de Postgrado*

Trabajo Fin de Máster

# USO DE VISIÓN ARTIFICIAL CON ALGORITMOS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DEL COVID EN RADIOGRAFÍAS DE PACIENTES

**Alumno/a:** **Guzmán Calanche, Andrés Gabriel**

Tutor/a primero: Prof. D. Pedro González García

Tutor/a segundo: Prof. D. Antonio Jesús Rivera Rivas

Dpto.: Informática

**Junio, 2021**

Universidad de Jaén
Centro de Estudios de Postgrado
Departamento de Informática

Don Pedro González García y Don Antonio Jesús Rivera Rivas, tutores del Proyecto Fin de Carrera titulado “Uso de Visión Artificial con algoritmos de aprendizaje profundo para la detección del La COVID-19 en radiografías de pacientes”, que presenta Andrés Gabriel Guzmán Calanche, autorizan su presentación para defensa y evaluación en la Universidad de Jaén.

Jaén, Julio de 2021

El alumno:

Firmado por GUZMAN CALANCHE ANDRES GABRIEL - Y6888793P el día

Andrés Gabriel Guzmán Calanche

Los tutores:

GONZALEZ GARCIA PEDRO - 26210879X Firmado digitalmente por GONZALEZ GARCIA PEDRO - 26210879X Fecha: 2021.07.06 09:40:00 +02'00'

Pedro González García

RIVERA RIVAS ANTONIO JESUS - 26015863B Firmado digitalmente por RIVERA RIVAS ANTONIO JESUS - 26015863B Fecha: 2021.07.06 09:25:24 +02'00'

Antonio Jesús Rivera Rivas

# Agradecimentos

Primero quiero agradecer la oportunidad que me dieron el Profesor Pedro González García y el Profesor Antonio Jesús Rivera  Rivas de ser mis tutores en este trabajo. Gracias por su guía, su apoyo y sus enseñanzas. Este trabajo no hubiera sido posible sin ustedes.

También quiero agradecer a mi familia. Gracias por su apoyo y su comprensión en esta aventura de vida. Este logro es de ustedes.

Agradezco a mi Madre por su guía, su apoyo y sus sabios consejos que me han permitido llegar tan lejos.

Y te agradezco a ti Papá, este será el primer grado que no podremos celebrar juntos. Esto no es un adiós, sino un hasta luego.

# Índice general

# Ìndice de Figuras

# Índice de Fórmulas

# 1. Introducción

Uno de los eventos notables más recientes, ha sido el impacto de la pandemia en el mundo moderno. Siendo de alcance global y de rápida dispersión. Razón por la cual ha surgido la necesidad de crear mecanismos asociados a un diagnóstico rápido y preciso que permita servir de apoyo al creciente número de víctimas.

En este sentido, se han comenzado a realizar varias aplicaciones en aprendizaje profundo para la detección de casos de La COVID-19 a través del análisis de imágenes de radiografías. Sin embargo, esto conlleva a otro problema y es como optimizar el proceso en el uso de las técnicas de predicción.

Muchas veces el manejo de imágenes de distintas fuentes y resoluciones impacta de forma negativa en los cálculos predictivos al aumentar el tiempo de procesamiento y el uso de recursos asociados.

En consecuencia, se realiza el estudio de aplicación de técnicas de reducción de dimensionalidad en imágenes en algoritmos de clasificación supervisada para detección de La COVID-19. Específicamente en los procesos de detección de la enfermedad en el uso de algoritmos de *Machine Learning*.

## 1.1. Motivación

La facilidad de propagación de la pandemia sumado a la novedad de esta sumió en un caos de servicios de asistencia médica sin precedentes a nivel mundial. Donde se presentaron situaciones de escaseo de los recursos necesarios para un diagnóstico eficaz de la enfermedad.

Este hecho generó la gran pregunta, si era posible hacer una detección de forma automática de la enfermedad permitiendo así realizar un diagnóstico eficaz, con alcance mayor que apoyara la gestión de los doctores los cuales no daban abasto para responder al problema.

Adicionalmente, La COVID-19 se diagnóstica como una neumonía atípica lo cual puede generar muchas confusiones al personal médico que no tenga la suficiente preparación al respecto.

Todo lo anterior motivó a muchos de los especialistas de *Machine Learning* a desarrollar scripts para la detección de La COVID-19 analizando las imágenes de pacientes enfermos. Al comenzar la generación de scripts, se presentaron varios problemas, como por ejemplo el uso de recursos de computadora, tiempo de ejecución del programa, por mencionar los más impactantes.

Basado en lo anterior, el presente trabajo, se realiza la aplicación de técnicas de reducción de dimensionalidad en las imágenes que van a ser analizadas en algoritmos de clasificación de forma de poder agilizar el procesamiento y obtención de un resultado con buenos niveles de precisión y menos tiempo de procesamiento.

## 1.2. Objetivos

Para una mejor comprensión del trabajo, se muestra a continuación el objetivo general y los objetivos específicos del trabajo.

### 1.2.1. Objetivo General

Evaluar el efecto de aplicar técnicas de reducción de dimensionalidad en imágenes para determinar cómo afecta el proceso de clasificación y determinar el punto óptimo de reducción según los casos de estudio.

### 1.2.2. Objetivos Específicos

- Revisión de técnicas y métodos de Visión Artificial con algoritmos de Aprendizaje Profundo.
- Análisis de técnicas de reducción de dimensionalidad aplicables en Aprendizaje Profundo.
- Desarrollo de alternativas para la detección en imágenes aplicadas al problema de detección la COVID-19 en radiografías.

- Validación de las propuestas mediante la experimentación con conjuntos de datos relacionados con el problema.

## 1.3. Estructura de la memoria

Este capítulo explica cómo está organizada la memoria y un resumen de cada capítulo que la integra, para que el lector rápidamente pueda conocer de qué trata cada capítulo.

### 1.3.1. Capítulo1. Introducción

Se explican las ideas iniciales que permiten contextualizar el trabajo y dar los conceptos necesarios para entender en que consiste el presente trabajo.

### 1.3.2. Capítulo2. Estado del arte

Este capítulo explica varios conceptos, definiciones y técnicas de aprendizaje profundo relevantes para el trabajo que permitan generar los antecedentes necesarios y facilitar la comprensión del desarrollo y de los resultados obtenidos.

### 1.3.3. Capítulo3. Herramientas

Revisión de técnicas y herramientas relevantes para el trabajo. Ventajas y desventajas de estas, cual fue el proceso de análisis para determinar la elección adecuada y justificación de uso. Así como la metodología para el desarrollo del trabajo y la obtención de los resultados.

### 1.3.4. Capítulo 4. Conjuntos de imágenes para la identificación de COVID-19

. Aquí se analiza las diferentes bases de datos de imágenes encontradas, sus características y la elección final de la base de datos utilizada para el trabajo.

#### 1.3.5. Capítulo 5. Técnicas de reducción desarrolladas. Experimentación y resultados.

Desarrollo de los modelos para clasificación junto con aplicación de las técnicas de reducción de dimensionalidad seleccionadas para el trabajo. Junto con los resultados obtenidos y comentarios sobre los mismos.

#### 1.3.6. Capítulo 6. Conclusiones y Recomendaciones

Una vez realizado los diferentes desarrollos con los modelos, se discuten los resultados obtenidos, se dan observaciones y comparaciones de las diferentes técnicas utilizadas. Además de ofrecer recomendaciones para futuros trabajos.

#### 1.3.7. Capítulo 7. Anexos

Anexos con la vinculación para tener acceso a toda la información relacionada con el trabajo.

##### 1.3.7.1. Anexo A. Planificación temporal del proyecto.

En el primer anexo se muestra la planificación temporal inicial realizada del proyecto incluyendo algunas consideraciones en la ejecución del proyecto.

##### 1.3.7.2. Anexo B. Ficheros y conjuntos de datos.

En este apartado se anexan varios enlaces donde se tiene acceso a los distintos cuadernos generados como parte del trabajo. Así como el conjunto de datos utilizado y los distintos conjuntos de datos generados con las técnicas de reducción de dimensionalidad.

# 2. Estado del arte

Este capítulo se habla sobre visión artificial, redes neuronales, aprendizaje profundo (en inglés llamado *Deep Learning*) y aprendizaje de máquina (*Machine Learning*). Además de las técnicas de clasificación.

## 2.1. La covid-19

La COVID-19 es la enfermedad causada por el nuevo coronavirus conocido como SARS-CoV-2. La OMS tuvo noticia por primera vez de la existencia de este nuevo virus el 31 de diciembre de 2019, al ser informada de un grupo de casos de «neumonía vírica» que se habían declarado en Wuhan (República Popular China).

Comenzó el 31 de diciembre de 2019, cuando fue informada a la Organización Mundial de la Salud (OMS) de casos de neumonía de etiología desconocida en Ciudad de Wuhan, provincia de Hubei, China. 3 días después se publicó la secuencia del genoma para soporte inmediato de la salud pública a través de los recursos de la comunidad en línea virological.org. Seguido de otros cuatro genomas depositados el 12 de enero del 2020.

Las secuencias del genoma sugieren la presencia de un virus estrechamente relacionado con el SARS-CoV y ya para el 11 de febrero de 2020, la OMS nombró

la nueva neumonía inducida por coronavirus como Enfermedad por coronavirus 2019 (la COVID-19).

El período de incubación, es decir el tiempo que transcurre desde que una persona se infecta por el virus hasta que presenta síntomas, oscila en general entre los 4 y los 7 días, en el 95 % de las ocasiones es menor a 12.5 días. Los límites extremos se han establecido entre 2 y 14 días después del contagio, aunque se han reportado casos inusuales de hasta 24 días. Sus condiciones más graves son en personas mayores de 60 años

- **Síntomas:** Los síntomas más habituales de la COVID-19 son la fiebre, la tos seca y el cansancio. Otros síntomas menos frecuentes que afectan a algunos pacientes son los dolores y molestias, la congestión nasal, el dolor de cabeza, la conjuntivitis, el dolor de garganta, la diarrea, la pérdida del gusto o el olfato y las erupciones cutáneas o

cambios de color en los dedos de las manos o los pies. Estos síntomas suelen ser leves y comienzan gradualmente. Algunas de las personas infectadas sólo presentan síntomas leves.

- **Propagación:** La principal forma de propagación de la COVID-19 es a través de las gotículas respiratorias expelidas por alguien que tose o que tiene otros síntomas como fiebre o cansancio. Muchas personas con LA COVID-19 presentan sólo síntomas leves. Esto es particularmente cierto en las primeras etapas de la enfermedad. Es posible contagiarse de alguien que solamente tenga una tos leve y no se sienta enfermo.

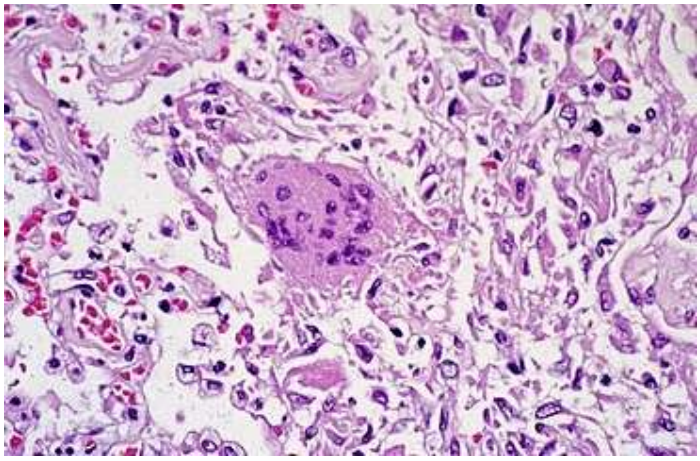

**Ilustración 2.1.1. Patología de un tejido pulmonar debido al SARS (Wikipedia, s.f.)**

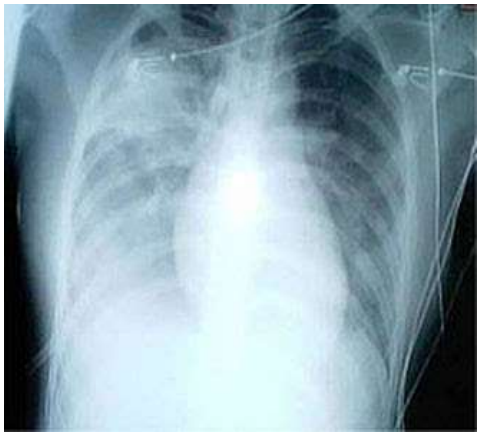

**Ilustración 2.1.2 Radiografía de pulmones opacos SARS (Wikipedia, s.f.)**

En el momento actual, se dispone de dos pruebas de detección de infección activa, una prueba rápida de detección de antígenos y una detección de ARN viral mediante una RT-PCR o una técnica molecular equivalente. La realización de una u otra, o una secuencia de ellas, dependerá del ámbito de realización, la disponibilidad y de los días de evolución de los síntomas.

## 2.2. Visión Artificial

La visión artificial por computador es la capacidad de la máquina para ver el mundo que la rodea. De forma más precisa, es la capacidad para deducir la estructura y las propiedades del mundo tridimensional a partir de imágenes bidimensionales.

Desde un punto de vista ingenieril, un sistema de visión artificial es un sistema autónomo que hace las veces en algunas de las tareas del sistema de visión humano. El conjunto de tareas o información que un sistema de visión artificial puede llegar a realizar o extraer van desde una detección de objetos en una imagen hasta la interpretación tridimensional de escenas complicadas.

La visión por computador es una disciplina en pleno auge y de continuo interés en el campo científico-técnico, gracias a que existe un alto número de aplicaciones posibles y a la importante función que desempeña. Algunos de estos campos son la robótica, los procesos de inspección automática, navegación de vehículos, análisis de imágenes médicas, etc.

La información visual es energía luminosa procedente del entorno. Para poder utilizar esta información es necesario que se transforme a un formato en la que pueda ser procesada. El ojo humano es el encargado de realizar esta misión en la visión humana, mientras que en la visión artificial esta tarea es llevada a cabo por una cámara. Esta cámara convierte la energía luminosa en impulsos eléctricos, que de esta manera puede ser muestreada y digitalizada para su procesamiento en un ordenador.

Una manera simple de comprender este sistema se bases en los propios sentidos, como los ojos que se usan para comprender el mundo que los rodea y donde la visión artificial trata de producir ese mismo efecto en máquinas.

Aunque la capacidad visual es uno de los pilares de la inteligencia humana. Su uso en la robótica supone también un importante avance en la inteligencia artificial. Para nosotros, la

percepción visual es algo innato y cotidiano, pero, la visión artificial es muy compleja y conlleva muchas dificultades. Entre las principales dificultades, destacan:

- **Mundo tridimensional:** mientras que las imágenes que se obtienen con una cámara son bidimensionales, el mundo que nos rodea no. Es necesario realizar las transformaciones correspondientes para obtener valores correctos.

- **Zonas de interés:** se necesita extraer elementos de información sutiles en imágenes complejas, por lo que entre tanta información es necesario reconocer formas, colores, etc.

- **Carácter dinámico de las escenas:** el mundo está vivo, por lo que en las imágenes que se toman muchos elementos están en movimiento. Por otro lado, otros factores como luminosidad, contraste, foco, etc. pueden marcar una importante diferencia, y por desgracia, estos factores son variables, no se pueden controlar.

La visión artificial resulta de gran utilidad en diferentes áreas de aplicación, tanto en acciones repetitivas como peligrosas. Entre varias se puede mencionar:

- Inspección y ensamblaje industria.
- Apoyo en el diagnóstico médico.
- Exploración espacial.

Hoy en día, los sistemas de visión basado en el PC ofrecen un aumento de productividad, robustez y fiabilidad. Además de una alta eficiencia y la capacidad de realizar tareas de inspección más sofisticadas.

Por norma general se distinguen tres procesos, los cuales se solapan en alguna ocasión:

- **Procesamiento:** Conlleva manipular las imágenes vistas como señales digitales para extraer la información.

- **Análisis:** Está pensado para determinar ciertas estructuras elementales tales como bordes o regiones, y las relaciones existentes entre ellas.

- **Aplicaciones:** Tratan de solucionar los problemas relacionados con algunas situaciones del mundo real.

La visión puede ser, probablemente, el mecanismo sensorial de percepción más importantes en el ser humano. No obstante, no quiere decir que sea el único, ya que una incapacidad visual no implica que ciertas actividades mentales no se puedan desarrollar.

El interés de los métodos de procesamiento de imágenes digitales se fundamenta en dos áreas de aplicación. La primera trata de mejorar la calidad de la percepción humana. La segunda tiene como fin procesar los datos que hay en la escena para que las máquinas puedan tener una percepción automática.

El concepto de visión artificial surge de intentar dotar a las máquinas de un sistema de visión. En gran cantidad de sistemas, la visión se complementa con otros mecanismos sensoriales como pueden ser detectores de alcance o proximidad. Es necesario integrar a posteriori todos los sensores para obtener el proceso de percepción global por completo. Las técnicas de procesamiento se utilizan hoy en día para resolver una gran diversidad de problemas. Generalmente no están relacionados, pero requieren del mismo modo métodos que sean capaces de realzar y extraer la información que hay en las imágenes para ser analizadas e interpretadas por los seres humanos.

### 2.2.1. Similitudes y diferencias entre la visión artificial y la visión humana

A continuación, se va a realizar una comparativa para definir las principales diferencias y similitudes entre la visión artificial y la visión humana.

La visión humana puede definirse como un sistema integrado, de forma genérica, por dos ojos, el nervio óptico y el cerebro. El ojo, que se puede apreciar en la Ilustración 2.1, es una lente que forma una imagen óptica y detecta la imagen con su retina. Desde aquí, a través del nervio óptico, la información es transmitida al cerebro para que se procese y se extraiga la información necesaria.

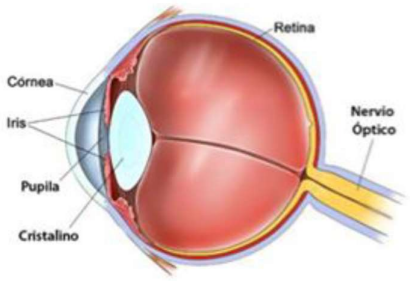


**Ilustración 2.2.1 Ojo humano (Comparativa de visión humana y visión artificial., 2014)**

Respecto lo anterior, se presentan una serie de similitudes en un sistema de visión artificial: una cámara con una lente recoge la imagen y se transmite la secuencia de vídeo a un ordenador que analiza la información, la cual se puede emplear para el fin que se desee.

La similitud entre un sistema de visión artificial y el sistema de visión humano es muy alta. Por ejemplo, el ojo está identificado con la cámara.

El ojo humano es compuesto por una gran cantidad de órganos y partes bien diferenciadas. Muchas de ellas, salvando la comparativa biológica-mecánica, se corresponden con elementos de un sistema de visión automático como se muestra en la Ilustración 2.2.

- El iris de un ojo se puede decir que es el diafragma de una cámara digital. Ambos se encargan del control de cantidad de luz que entra al sensor.
- El cristalino realiza la función de la óptica. Consigue un enfoque correcto y un objeto nítido.
- La retina es similar al sensor. Determina la luminosidad que debe tener cada píxel de la imagen, es decir, se encargan de generar la imagen que realmente se ve.
- El nervio óptico hace de bus de conexión, ya que conecta la cámara con el sistema de procesamiento.
- El lóbulo parietal del cerebro es parecido al sistema de procesamiento de datos. Esto quiere decir que en un sistema automático de visión se aplica un algoritmo ya implementado que determina si la imagen cumple o no una serie de requisitos.

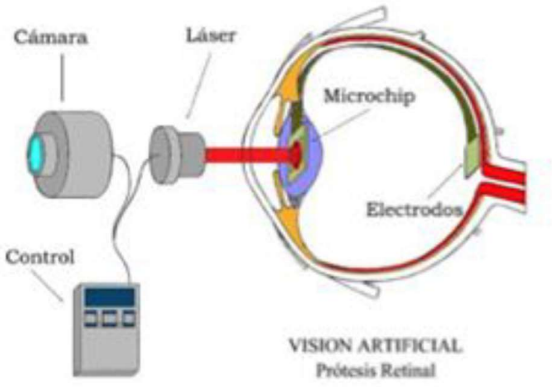


**Ilustración 2.2.2 Correspondencia ojo humano con sistema de visión (Comparativa de visión humana y visión artificial., 2014)**

Actualmente no es posible proveer a un sistema de una visión parecida a la humana. Para justificarlo, basta con decir que la medicina no es capaz de saber con certeza el funcionamiento del sistema de visión humano.

Tanto la detección de objetos a una cierta velocidad, como diferenciar la superficie de estos son actividades que la visión humana realiza a diario. Es realmente complicado aplicar estos procesos en máquinas, por lo que se concluye que el cerebro humano tiene una gran capacidad de cálculo.

El desarrollo de la visión artificial parte de la visión humana, como punto de inicio. Casi todos los sistemas disponen una arquitectura modelada, hasta cierto punto, como la visión humana. En una persona el ojo es el encargado de detectar una imagen, la información es extraída por una parte del cerebro, otra parte de este acepta esta información y ordena las operaciones a realizar. En un sistema artificial, la cámara o el sensor hacen de ojo, un procesador que se ha construido y que está programado para el análisis de la imagen, procesa la imagen obtenida de la cámara y un actuador acepta la información que recibe el procesador y dirige las acciones necesarias.

Es imposible que las capacidades de la visión humana puedan ser imitadas con el mismo grado de calidad por un sistema de visión artificial. El ojo humano puede detectar 100 millones de píxeles en el espectro de la luz, mientras que una cámara normal puede llegar a detectar unos 250.000 elementos. Pero la tecnología avanza y poco a poco se van creando cámaras tan potentes que se van acercando al funcionamiento del ojo humano.

No obstante, para muchas aplicaciones es suficiente con una cantidad justa de píxeles que puede proporcionar cualquier cámara. También es cierto que, si se quiere un aumento de píxeles, es necesario un incremento del tiempo de cómputo, el cual es en muchas ocasiones un factor determinante para dar validez a una aplicación.

Aun así, no hay sistemas que sean capaces de imitar con las mismas garantías de visión a la humana. Por ejemplo, no hay un sistema de visión capaz de conducir un coche con tráfico y eventos no esperados. Pero para uso en la industria no es necesario llegar a tal extremo, ya que la mayoría de las tareas de visión artificial son mucho más sencillas. El fin último de estos sistemas suele ser hacer trabajos sencillos y repetitivos, los cuales pueden llegar a fatigar la vista humana.

Que una tarea sea monótona y repetitiva, hace que la efectividad de un humano en la inspección no sea superior al 80 %. Mientras que un sistema automático de visión implementado de manera correcta puede llegar a una efectividad del 99.7 %.

Por este motivo surge la visión artificial, quedando bastante claro que no se puede imitar a un sistema de visión humana, pero puede llegar a superarlo cuando las tareas de repetición o peligrosas pueden disminuir la efectividad humana. Como resumen se destacan los siguientes postulados:

- La visión humana puede detectar más detalles.
- La visión humana maneja tareas más complicadas.
- La visión artificial tiene mejor respuesta en tareas de repetición.
- La visión artificial puede trabajar con luz visible o sin ella.

### 2.2.2. Imágenes digitales

La visión artificial está fundamentada en el tratamiento de imágenes. Las instantáneas pueden ser fotos o ser extraídas de un vídeo obtenido a partir de cualquier tipo de cámara: cámara web, cámara de vídeo digital, etc.

Las imágenes han de ser tratadas para llegar a obtener el máximo número de conclusiones posibles para conseguir el objetivo final del proyecto. La forma de tratar las imágenes depende de las herramientas de las que dispone el ser humano, desde algoritmos básicos hasta los más complejos que puedan llegar a existir. También puede ayudar la tecnología existente para facilitar el procesamiento como ordenadores, programas, etc.

### 2.2.3. Formación de las imágenes digitales

El punto inicial de todo sistema de percepción visual es la imagen. El proceso para captar la imagen implica que se tenga que utilizar algún sensor para obtener la representación bidimensional de la escena.

En el ser humano los ojos son los encargados de adquirir la imagen. Estos transforman la información visual en impulsos nerviosos que a continuación va a utilizar el cerebro para interpretar la imagen. La estructura del ojo puede variar de unos humanos a otros.

La visión humana hace que se vean las imágenes como una representación continua. No obstante, para procesar una imagen en un ordenador, es necesaria una transformación

previa de la imagen. Esta imagen continua debe ser transformada de alguna forma en un conjunto de números que pueda manipular el ordenador. Este proceso es conocido como digitalización de la imagen. Para crear una imagen, hay que considerar una unidad básica: el píxel.

El píxel es la menor unidad homogénea que forma parte de una imagen digital. Si se amplía lo suficiente una imagen digital, se pueden observar los píxeles de la imagen. Aparecen como pequeños cuadros en color o en escala de gris.

La formación de las imágenes es debida a una sucesión de píxeles. El orden marca la coherencia de la información presentada siendo su conjunto una matriz de información para el uso digital como se ve en la Ilustración 2.2.3.


**Ilustración 2.2.3 Matriz de píxeles**

Cada píxel queda codificado por un conjunto de bits que tienen una longitud determinada, la cual es llamada la profundidad de color. Por ejemplo, un píxel puede codificarse con un byte (8 bits), admitiendo 256 variaciones: 28. En las imágenes reales se utilizan tres bytes para definir un color, es decir, en total se pueden representar 224 colores (16.777.216).

Para transformar el valor de un píxel a un color, es necesario conocer la profundidad, el brillo del color y el modelo de color que se está usando. En el modelo de color RGB (*Red-Green-Blue*), se forman los colores en función del porcentaje que cada uno de ellos represente. Por ejemplo, para obtener el color violeta es necesario mezclar el rojo y el azul, pudiendo obtener diferentes tonalidades variando las proporciones de los colores primarios. En el modelo RGB se suelen utilizar 8 bits para indicar la proporción de cada uno de los

colores primarios. Así, cuando uno de ellos vale 0 no interviene en la mezcla, mientras que cuando vale 255 (máximo valor para 8 bits) interviene con su máxima tonalidad. El modelo RGB es el utilizado por la mayor parte de los dispositivos.

Para expresar el tamaño de la imagen, se utiliza el concepto de Megapíxeles, el cual equivale a un millón de píxeles (en esta medida de computación no se usa la base binaria de 1024). Esta unidad es empleada con gran frecuencia para indicar la resolución de las cámaras de fotos digitales. Si una cámara tiene una resolución de 2048 x 1536 píxeles, tiene una resolución de 3,1 Megapíxeles (2048 x 1536 = 3.141.728).

Al realizar fotos, el tamaño de estas queda definido por el número de megapíxeles que tenga la cámara y por el tamaño de las impresiones que se pueden realizar. Pero hay que tener en cuenta que cada Megapíxel se distribuye en un área y, por tanto, la diferencia entre 7 y 8 megapíxeles es menos representativa que entre 3 y 4, debido a que no se trata de una medida exponencial.

La resolución de una imagen queda definida como el número de píxeles que la describen. Se suele medir en píxeles por pulgada (ppi), definiendo la calidad de la imagen, lo que condiciona también la cantidad de memoria que va a ocupar. Por ejemplo, si una imagen tiene una resolución de 72ppi, implica que tiene 5.184 píxeles en una pulgada cuadrada (72 píxeles ancho x 72 píxeles alto).

Se puede deducir que, a mayor resolución de una imagen, tiene más cantidad de píxeles que la describen. Es evidente que, a mayor resolución, mejor representación se va a obtener utilizando un dispositivo de salida adecuado, ya que pueden proporcionar un mayor detalle descriptivo y una transición de color más suave.

El tamaño de una imagen en píxeles expresa su tamaño expresado en píxeles horizontales y verticales. Se puede obtener de manera sencilla si se conoce el tamaño de impresión y la resolución de la imagen. Para ello basta con multiplicar el alto y el ancho por la resolución. Por ejemplo, una imagen de 9 x 12 pulgadas escaneada a 300ppi, tiene una dimensión de 2.700 x 3.600 píxeles.

Estos conceptos están relacionados y dependen mutuamente, aunque se refieren a características diferenciadas y no se deben confundir. El tamaño de la imagen indica sus dimensiones una vez impresa, mientras que el tamaño del archivo indica la cantidad de memoria necesaria para almacenarla.

Lógicamente la resolución de la imagen condiciona estos dos conceptos. Dado que la resolución de la imagen es fija, al aumentar el tamaño de esta se reduce la resolución y viceversa.

Si se reduce la resolución manteniendo el tamaño, se eliminan píxeles y se tiene una descripción menos precisa de la misma además de unas transiciones de color más abruptas. El tamaño del archivo que genera la imagen es proporcional a su resolución, de modo que si se modifica este parámetro se modifica el tamaño del archivo.

Es éste, por tanto, un elemento importante para decidir la resolución de una imagen ya que plantea un compromiso a la hora de capturar toda la información que se necesita de la misma y mantener el tamaño del archivo. Todo va a depender del uso final de la ilustración.

### 2.2.4. De números a colores

Para transformar la información numérica de un píxel en un color, hay que saber el modelo de color y su brillo, así como la profundidad del color. En este punto es donde aparece el modelo de color RGB, el cual permite crear un color compuesto por 3 colores básicos, dependiendo de la saturación de cada uno.

En los dispositivos gráficos se define el concepto de profundidad de color: que es la cantidad de bits con la que se codifican los píxeles. De este modo, cada píxel se puede modificar con un byte (8 *bits*), de modo que puede tener 256 variaciones de color. De forma general las imágenes utilizan 3 *bytes* para definir un color, es decir, se pueden representar 224 colores.

La mayor parte de los dispositivos que se utilizan en un ordenador utilizan el modelo RGB. Está basado en la adición de los 3 colores primarios, donde se puede representar un color por medio de la adición entre ellos.

Además, las escalas de grises y negros tienen sus variaciones, muy semejantes a los demás tonos. También existe el color blanco que es la máxima saturación de los 3 colores primarios, y el negro como la ausencia de todos ellos. En las Ilustraciones 2.4 y 2.5 se pueden ver dos ejemplos de escalas de grises.

En la Ilustración 2.6 se puede observar el valor de los píxeles en niveles de gris en una imagen. La saturación de los grises se representa en una escala de 0 a 255, representando cada elemento de la matriz a un píxel.

**Ilustración 2.2.4 Escala de grises 1 (aloj.us)**

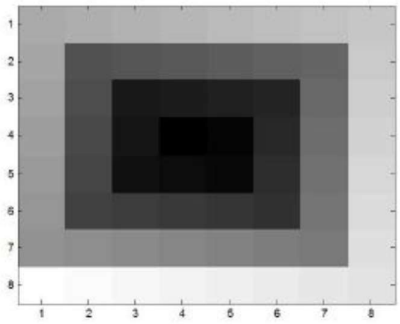


**Ilustración 2.2.5 Escala de grises 2 (aloj.us)**

En el modelo RGB, cada color primario tiene su propia escala para representar un color específico. El problema sería tener 3 elementos de 3 diferentes matrices para cada uno de los posibles colores. Si se sabe que los 3 elementos de las distintas matrices se adicionan, cómo distinguir al color amarillo (255, 255, 0) del cyan (0, 255, 255).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 173 | 177 | 181 | 185 | 189 | 193 | 197 | 201 |
| 169 | 85 | 89 | 93 | 97 | 101 | 105 | 205 |
| 165 | 81 | 29 | 33 | 37 | 41 | 109 | 209 |
| 161 | 77 | 25 | 5 | 9 | 45 | 113 | 213 |
| 157 | 73 | 21 | 17 | 13 | 49 | 117 | 217 |
| 153 | 69 | 65 | 61 | 57 | 53 | 121 | 221 |
| 149 | 145 | 141 | 137 | 133 | 129 | 125 | 225 |
| 255 | 253 | 249 | 245 | 241 | 237 | 233 | 229 |

**Ilustración 2.2.6 Valor de los píxeles en niveles de gris (aloj.us)**

### 2.2.5. Aplicaciones varias de la visión artificial

En muchas ocasiones la visión por computador no es la mejor solución a los problemas. Por ejemplo, la conducción por carretera de un vehículo, algo que corresponde a

una solución humana. Pero como ya se dijo, las soluciones humanas tienden a ser inexactas, lentas y sin rigor, por lo que, en muchas ocasiones, especialmente en la industria, la visión artificial es la alternativa más y mejor considerada.

También cabe diferenciar aquellas aplicaciones en las que la visión artificial constituye una herramienta por sí sola y aquellas en las que es una parte de un sistema multisensorial. El primer caso se refiere a las aplicaciones donde la visión es el único sensor. El segundo es referido a la navegación en robótica, donde la visión es una capacidad sensorial más para la percepción del entorno que rodea al robot. Se van a analizar varios campos de aplicación en las siguientes secciones.

### 2.2.6. Navegación en robótica

La visión es un elemento de un sistema con más sensores. La información que procede de la visión se valida, compara e integra con el resto de la información que proporcionan otros sensores.

Para la navegación en robótica se recurre generalmente a técnicas de visión estereoscópica para tratar de reconstruir la escena en 3-D. El uso del movimiento basado en la visión constituye un gran recurso.

### 2.2.7. Biología, Geología y Meteorología

En el campo de la biología se puede distinguir entre aplicaciones microscópicas y macroscópicas.

En una imagen microscópica se pueden encontrar una gran cantidad de organismos, que, utilizando técnicas de segmentación, pueden ser aislados para identificarlos mediante propiedades tipo excentricidad, tamaño, color, etc. O simplemente para contar el número de microorganismos presentes en una imagen.

En imágenes macroscópicas se pueden utilizar las regiones para identificar diferentes tipos de texturas en vegetales o características de diferentes áreas naturales por su color o el crecimiento de ciertas especies por diferencia de imágenes. También puede ser interesante identificar el grado de floración de un determinado cultivo.

La visión artificial también es usada en geología para, por ejemplo, detectar movimientos de terrenos captando dos imágenes en diferentes momentos de tiempo para comprobar la variación mediante una diferencia de imágenes (bajo similares condiciones de iluminación).

En meteorología se utilizan las técnicas de detección y predicción del movimiento para observar la evolución de ciertas masas nubosas u otros fenómenos meteorológicos a través de imágenes recibidas vía satélite. También puede resultar interesante hallar la cota de nieve.

### 2.2.8. Medicina

En la rama médica hay muchas aplicaciones en las que está presente el procesamiento de imágenes y a menudo está orientado a detectar dolencias o enfermedades. Algunas de las técnicas utilizadas son las radiografías, resonancias magnéticas, etc.

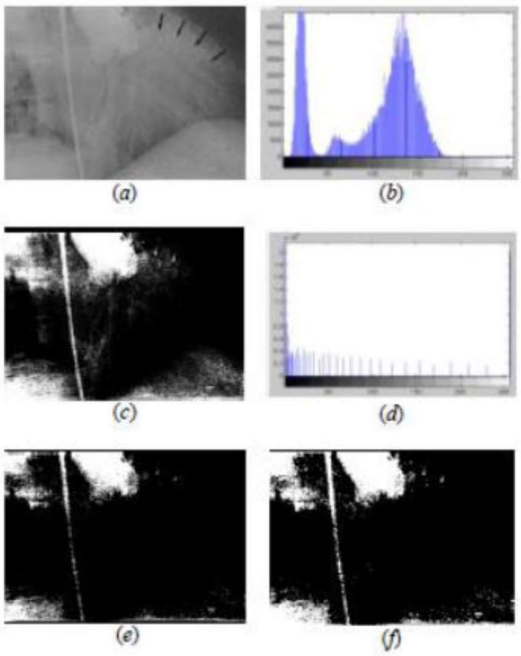


**Ilustración 2.2.7 Sistema de inspección en medicina (Alejo, 2014)**

Un claro ejemplo es el de una radiografía que se observa en la Ilustración 2.7, en la que la imagen presenta baja calidad (a) y se puede extraer información sobre las manchas blancas que aparecen en la misma. En (b) se muestra su histograma de frecuencias. Se

modifica el histograma mediante el aumento del contraste y gamma, obteniendo la imagen en (c) y su histograma en (d). Pero todavía se puede extraer más información, si se binariza con un umbral a placer se obtiene (e) y mediante un proceso de rellenado de huecos se consigue (f). A partir de esta imagen se extraen las manchas para obtener el área de las diferentes regiones etiquetadas.

En el campo médico se pueden desarrollar otras aplicaciones como el movimiento de las paredes cardiacas a partir de imágenes de resonancia magnética.

### 2.2.9. Identificación de construcciones, infraestructuras y objetos en escenas de exterior

Es posible detectar la presencia de algunas regiones a través de la segmentación, utilizando imágenes de satélite o tomadas desde el aire. Se puede utilizar para detectar la presencia de construcciones como edificios, carreteras, etc. a través de técnicas de extracción de bordes o contornos combinados con la segmentación.

Por ejemplo, se puede dar el caso de reconstrucciones de tejados de casas urbanas, que mediante una base de datos se puede partir de ella y elegir modelos para reconstruirlo. Otro ejemplo consiste en la detección de carreteras usando el método de contornos deformables. Para ello la transformada de Hough puede ser muy útil.

### 2.2.10. Reconocimiento y clasificación

A partir del tamaño y el recuento, se puede desarrollar una aplicación para clasificar objetos. Por ejemplo, para contar monedas en función de su área, perímetro, número de Euler, etc. tras haberlas binarizado.

Otra aplicación destacada es el reconocimiento de caras de personas utilizando perfiles de intensidad. Otras aplicaciones presentes en el estado del arte proponen una lectura automática de datos del DNI, así como reconocer objetos basados en el color.

También es posible el reconocimiento de huellas dactilares, utilizando un conjunto de máscaras. También se pueden mencionar el reconocimiento de caracteres usando aplicaciones OCR.

### 2.2.11. Inspección y control de calidad

La inspección se puede realizar para verificar ciertas características o verificar dimensiones en objetos manufacturados.

La inspección se refiere a la verificación de si un objeto cumple con ciertos criterios. Para ello se compara un objeto con modelos que describan características buscadas.

Uno de los fines de los controles de calidad es detener la producción si se detecta que el sistema genera productos que no cumplen con estándares globales. Por ejemplo, en la fabricación de galletas se puede detectar cuando las galletas no son redondas utilizando la transformada de Hough.

Otra aplicación es la inspección de placas de circuitos impresos, donde hay numerosas pistas y permite verificar la ubicación de los componentes, chequear las pistas, etc.

### 2.2.12. Cartografía

Utilizando imágenes estereoscópicas, aéreas o de satélite se pueden obtener elevaciones del terreno usando técnicas de correspondencia basadas en el área. Se utiliza para la elaboración de catastros en zonas rurales, utilizando imágenes aéreas que permiten una fácil identificación de las parcelas y sus delimitaciones tras el tratamiento mediante técnicas de extracción de bordes y regiones. Si los sensores están perfectamente calibrados se puede llegar a determinar la superficie real de las parcelas basándose en el área de las imágenes medidas en píxeles.

### 2.2.13. Representación digital de una imagen.

Para obtener una imagen en visión artificial, es necesario el uso de algún tipo de sensor que la proporcione. A partir del sensor se obtiene una representación en dos dimensiones de la escena.

Se podría representar una imagen como una función bidimensional de intensidad luminosa o nivel de gris, f(x, y) donde x e y denotarían las coordenadas espaciales bidimensionales y el valor de f en cada punto (x, y) sería proporcional a la luminosidad de la imagen en ese punto.

Sin embargo, un ordenador no puede trabajar con una imagen que tenga una función continua, por lo que hay que muestrearla y digitalizarla. El muestreo aproxima la escena real a una imagen integrada por una matriz de MxN puntos, cuyos elementos representan el nivel de gris en cada punto de la imagen, o los valores en las correspondientes matrices de color RGB.

Cada imagen está compuesta por una matriz de píxeles, donde al brillo de cada píxel se le conoce como nivel de gris del mismo. De forma genérica, los valores de nivel de gris varían entre 0 y 255, donde los extremos son el blanco y el negro. Todos los demás valores representan distintas intensidades de gris.

Por norma general, 0 es el color negro y 255 el blanco, pudiendo diferenciar 256 variaciones de negro a blanco como se observa en la Ilustración 4.1.

En función de la aplicación y los requerimientos necesarios, es mejor un tamaño de imagen u otro. Debido a ello, hay algunas aplicaciones donde las imágenes tienen muy baja resolución, mientras que otras necesitan una alta resolución. Para el presente trabajo es conveniente que las imágenes tengan alta resolución, para poder aplicar la técnica SURF. También podría funcionar con resoluciones bajas, pero los resultados no serían los esperados.

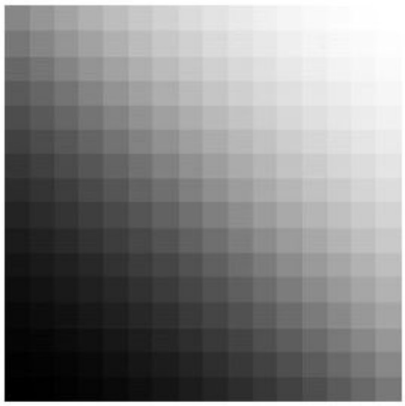

**Ilustración 2.2.8 Diferentes tonalidades de gris (aloj.us)**

Una imagen se traduce a una matriz de datos, es decir, un conjunto ordenado de elementos reales. Las imágenes se almacenan como matrices, donde cada elemento se corresponde con un píxel.

Para el almacenamiento de imágenes, varios programas trabajan con números en punto flotante con precisión simple de 32 *bits*. Aunque para reducir la memoria se puede trabajar con matrices de enteros de 8 *bits* con signo, enteros de 16 bits con signo y enteros con signo de 32 *bits*.

En cuanto al tipo de imágenes, existen 3 tipos básicos de imágenes y entre ellas difieren por la forma en la que son interpretados los elementos de la matriz de datos.

#### 2.2.13.1. Imagen binaria

Cada píxel asume uno de los valores discretos. En esencia estos dos valores se corresponden con blanco y negro. Una imagen binaria se almacena como una matriz bidimensional de 0 (negro) y 1 (blanco). Cada pixel se corresponde con 1 *bit*.

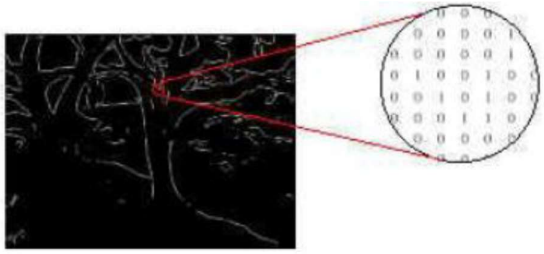

**Ilustración 2.2.9 Imagen binaria (Tutorial Opencv)**

Una imagen binaria puede considerarse como un tipo especial de imagen de intensidad, que contiene sólo el blanco y el negro.

Una imagen binaria puede ser almacenada en una matriz double o uint8. Sin embargo, una matriz uint8 es preferible, ya que utiliza mucha menos memoria. En la Ilustración 4.2 se muestra un ejemplo de imagen binaria.

#### 2.2.13.2. Imagen de intensidad

También llamada imagen en escala de grises. Consiste en una matriz de datos cuyos valores representan intensidades dentro de un mismo rango. La matriz puede ser *double*, en cuyo caso contiene los valores en el intervalo [0, 1], o de tipo unit8, en cuyo caso el rango es

[0, 255]. Los elementos de la matriz representan diferentes intensidades o niveles de gris, donde el 0 representa el negro y el 255 el blanco. Cada píxel se corresponde con 1 byte, de ahí los 255 niveles permitidos. En la Ilustración 4.3 se muestra un ejemplo de una imagen en escala de grises.

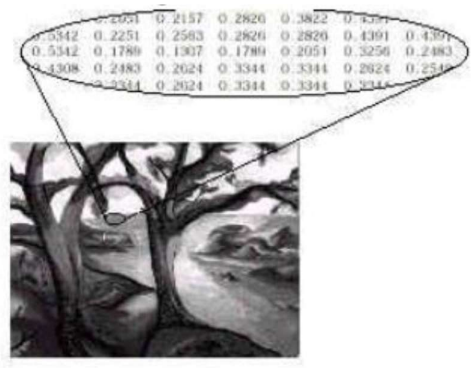

**Ilustración 2.2.10 Imagen en escala de grises (Tutorial Opencv)**

### 2.2.13.3. Imagen de color

También llamada imagen en formato RGB. Representa cada color del píxel como un conjunto de tres valores, que representan las intensidades de rojo, verde y azul que componen el color. Cada píxel se codifica con 3 bytes (uno por color), siendo el número total de colores posibles del orden de 16,7 millones. Esto es lo que se llama una imagen multicanal, en este caso de 3 canales.

También existen imágenes RGBA de cuatro canales, donde el cuarto canal indica el nivel de transparencia de un píxel, con aplicaciones en visión nocturna o imágenes por satélite. En la Ilustración 4.4 se muestra un ejemplo de imagen a color.

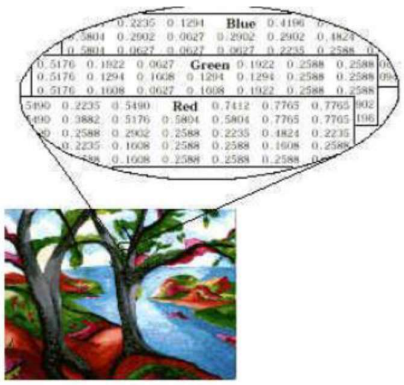


**Ilustración 2.2.11 Imagen de color (Tutorial Opencv)**

### 2.2.14. Etapas en el procesamiento de una imagen

Las etapas en el desarrollo de un proyecto dependen del objetivo de este. Aunque se pueden definir una serie de etapas de forma global para el procesamiento de una imagen como se observa en la Ilustración 4.5.

Para un sistema de seguimiento visual, las etapas de segmentación, extracción de características y reconocimiento no suelen ser llevadas a cabo, mientras que sí se requiere la implantación de otras técnicas específicas. Analizando cada una de las etapas de manera individual, aparecen las siguientes definiciones:

- **Adquisición de la imagen:** El objetivo es obtener una imagen digital. Para este trabajo, la imagen es un vehículo en el que se ve la matrícula.
- **Procesamiento previo:** Se realizan operaciones sobre las imágenes para llevarla a un punto para obtener información de ellas en etapas venideras.
- **Detección de contornos:** Está presente prácticamente en todos los procesamientos de imágenes. Una vez se tienen los contornos, se trabaja para obtener la información que se quiere.
- **Segmentación:** Obtiene todas las regiones donde los píxeles comparten algún atributo. Estas regiones son los objetos de interés en la imagen.
- **Extracción de características:** Estas características pueden presentarse en forma de líneas, círculos, elipses, formas determinadas, etc. pero también pueden tenerse como características locales de la imagen, es decir, puntos que atesoren multitud de información que pueda relacionarse con patrones externos.
- **Localización y/o reconocimiento de objetos:** Corresponde a la localización de un objeto. Muchas veces consiste en detectar formas o Ilustraciones con cierta forma. Junto con la localización suele aparecer el reconocimiento de estos objetos.
- **Interpretación de la escena:** Una vez obtenida la información en las etapas anteriores se procede a interpretar la escena, considerando para ello la relación entre los objetos simples previamente reconocidos y localizados, así como un cierto conocimiento sobre restricciones y reglas que rigen el mundo real. Esta etapa está estrechamente ligada a la inteligencia artificial, estando aún en sus primeros pasos y resultados definitivos.


**Ilustración 2.2.12 Etapas en el procesamiento de una imagen**

En un sentido más amplio, las etapas se pueden agrupar en dos niveles: visión de bajo nivel y análisis de la escena.

La visión de bajo nivel incluye las primeras etapas de procesamiento. El segundo nivel es el análisis de la escena, cuya misión es coger las características que ha obtenido el primer nivel y hacer una descripción de la escena a un nivel superior.

En el presente trabajo se aplican procedimientos de visión artificial relativamente novedosos que no están incluidos en las técnicas clásicas de visión. Esta tecnología trata de obtener características locales de la imagen, para obtener un modelo dentro de ella.

#### 2.2.14.1. Adquisición de la imagen

Las imágenes digitales son señales discretas, cuyo origen suele ser una señal continua. Para verlo de manera más clara, una cámara digital obtiene imágenes del mundo real, el cual es continuo. Existen dos etapas en el proceso de obtención de imágenes:

- **Captura:** Se utiliza un dispositivo, que suele ser óptico, con el cual se obtiene la información relativa a una escena.
- **Digitalización:** Se transforma la información obtenida, que es una señal con componentes continua, en la imagen digital, que es una señal con todas sus componentes discretas.

#### 2.2.14.2. Modelos de captura de imágenes

Se distingue entre dos dispositivos para la captura de una imagen. El primero es el dispositivo pasivo, el cual está basado en el principio de cámara oscura. El segundo se trata de un dispositivo activo, que se basa en el escaneo.

Los dispositivos basados en cámara aventajan a los basados en escaneo en cuanto a la velocidad. También son más simples y se asemejan más al sistema visual humano. Debido al gran avance de la tecnología, los modelos de cámara han acabado igualando, e incluso superando, a los de escaneo en cuanto a calidad de imagen obtenida se refiere.

Esta clasificación no incluye todas las formas posibles para la creación de imágenes, como pueden ser las imágenes sintéticas.

##### 2.2.14.2.1. Cámara oscura

Se puede usar para obtener imágenes de escenas tridimensionales (mundo real) y proyectarlas en un plano bidimensional. Ejemplos de este tipo son las cámaras de fotos y las de vídeo. Este modelo también se usa para obtener imágenes de elementos bidimensionales. También es posible obtener una representación en tres dimensiones si se usan dos o más cámaras para obtener diferentes perspectivas.

##### 2.2.14.2.2. El escaneo

En este caso existe un elemento activo (normalmente haz de láser) que recorre la escena que se quiere capturar. De este modo, son necesarios dos dispositivos, el emisor del haz y el receptor. Estos dispositivos se usan con diferentes fines.

#### 2.2.14.2.3. Digitalización

Se trata del proceso de paso del mundo continuo al mundo discreto (analógico a digital). En la digitalización se distinguen dos procesos: muestreo y cuantificación.

- **Muestreo:** El muestreo consiste en la medición a intervalos respecto de alguna variable (tiempo o espacio). El parámetro fundamental es la frecuencia de muestreo, que representa el número de veces que se mide un valor por unidad de cambio. El número de muestras por unidad de espacio sobre el objeto original lleva al concepto de resolución espacial de la imagen, que se define como la distancia, sobre el objeto original, entre dos píxeles adyacentes.
- **Cuantificación:** La cuantificación consiste en la discretización de los posibles valores de cada píxel. Los niveles de cuantificación son potencias de 2 para facilitar el almacenamiento en el computador.

### 2.2.15. Operaciones generales. Técnicas clásicas

A continuación, se van a exponer una serie de operaciones generales que se pueden realizar sobre las imágenes con el fin de modificarlas.

#### 2.2.15.1. Escalado de una imagen

El escalado es una transformación que se realiza a una imagen cuando se multiplican las coordenadas de cada píxel por un factor de escala. Si el factor está entre 0 y 1 la imagen se reducirá, mientras que si es mayor que 1 la imagen se aumentará.

El factor de escala puede ser diferente para ambas coordenadas. Se puede aplicar un factor sobre la coordenada horizontal y otro diferente sobre la vertical. Este escalado modifica la forma de las Ilustraciones y se conoce como escalado anisotrópico.

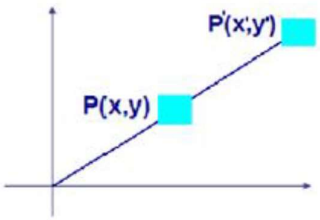


**Ilustración 2.2.13 Escalado de una imagen (Alejo, 2014)**

Las coordenadas de los píxeles de la imagen escalada (x', y') vienen dadas por las siguientes ecuaciones.

$$x` = s_x \cdot x$$

**Fórmula 2.2.1 Ecuación de escalada en x`**

$$y` = s_y \cdot y$$

**Fórmula 2.2.2 Ecuación de escalada en y`**

Si $s_x = s_y$ se producirá un escalado isotrópico. Si $s_x \neq s_y$ el escalado es anisotrópico.

### 2.2.15.2. Modificaciones en el color de una imagen. Escala de grises

Una imagen en escala de grises está representada por medio de una matriz bidimensional de m x n elementos, mientras que una imagen de color RGB es representada por una matriz tridimensional m x n x p. m y n denotan las dimensiones de la matriz y p representa el plano, que para RGB puede ser 1 para rojo, 2 para verde y 3 para azul.

La operación que permite transformar una imagen RGB en escala de grises obtiene las tres matrices. La imagen resultante en escala de grises tiene las dos dimensiones de la imagen y a mayores una dimensión más, donde se guarda la matriz unidimensional de los niveles de gris. Por tanto, la imagen inicial en color RGB será igual que la resultante en escala de grises salvo por el color.


**Ilustración 2.2.14 Imagen convertida a escala de grises**

### 2.2.15.3. Binarización: Binarización automática

Las imágenes digitales se componen utilizando un amplio rango de valores de intensidad, a las que se las denomina imágenes de nivel de gris. Normalmente el rango de niveles de gris es 256 (8 bits por píxel), pero ello no quiere decir que tenga que ser siempre así. De modo que puede variar dependiendo de la aplicación.

Por otro lado, hay aplicaciones que no necesitan tantos niveles de gris, ya que las imágenes pueden estar representadas por pocos niveles. En caso de reducir el nivel de gris hasta llegar a tener 2 valores, se tiene una imagen binaria. Este tipo de imagen se muestra en la Ilustración 2.15.


**Ilustración 2.2.15 Imagen normal/binarizada**

Donde los píxeles en negro equivalen a 0 y los píxeles en blanco a 1 (255), que se almacenan en la memoria del ordenador. Con este tipo de imágenes se consigue reducir la representación y el procesamiento de la imagen al mínimo.

Reducir la representación de la imagen hace que se aproveche la memoria y la potencia computacional. Además, se pueden obtener las propiedades geométricas y topológicas de manera rápida de los objetos que componen la imagen.

En un ordenador el procesamiento de imágenes binarias es mucho más rápido, lo que ha provocado que se empleen estas imágenes para una gran mayoría de las aplicaciones. Hoy en día, se ha mejorado de forma exponencial la potencia computacional para tratar imágenes en niveles de gris, se siguen utilizando las imágenes binarias para las aplicaciones. Esto se debe a que con las imágenes binarias se consigue una aplicación más simple y robusta.

Es preciso definir qué es la binarización: es el proceso mediante el cual se convierte una imagen en escala de grises en una imagen binaria. Este proceso es necesario, ya que no existen cámaras que sean capaces de obtener imágenes binarias, lo máximo que consiguen son imágenes en niveles de gris.

Si se tienen imágenes en niveles de gris bien contrastadas, se puede realizar una binarización con poco procesamiento, además de ser rápido y fiable. En ocasiones se utilizan técnicas de retroiluminación para obtener imágenes bien contrastadas que se puedan reducir a binarias y quede representada claramente la información deseada.

La forma para obtener una imagen binaria es considerar el bit más significativo del nivel de gris de cada píxel. Significa fijar el umbral en el punto medio de la escala de grises, que sirve como referencia. Si el valor del píxel es menor que el umbral, ese píxel valdrá 0, mientras que si es mayor valdrá 1.

Aunque binarizar siguiendo esta técnica no es la forma más correcta, siendo descartada en muchas ocasiones. El rango dinámico de una imagen no siempre se extiende a todo el rango de niveles de gris posible y aun así, muchas veces es más adecuado fijar el umbral de binarización en otro punto distinto del valor medio de la escala de grises.

Cuando una imagen está bien contrastada, apenas se pierde información. Muchas veces esta operación permite separar los objetos del fondo.

Para realizar una correcta binarización, la etapa clave es la elección del umbral, que es la referencia para separar el fondo de los objetos. La separación se haría de forma ideal si se conociese la distribución de los píxeles oscuros y claros. En general estas distribuciones no son conocidas de antemano, pero con el histograma de la imagen se obtiene la información del número de pixeles asociados a cada nivel de gris que resulta muy valiosa.

Si las distribuciones de los niveles de gris asociados a los píxeles claros y oscuros están muy separadas, entonces el histograma será bimodal. En este caso, el umbral óptimo es aquel que divida ambos niveles de gris, estando situado en el valle del histograma.

Por el contrario, cuando las distribuciones comienzan a estar más solapadas la elección del umbral como un valor perteneciente al valle empieza a ser difícil, puesto que el valle ya no aparece tan claro. En este tipo de imágenes la binarización no proporcionará buenos resultados.

Para establecer el umbral de binarización en una aplicación industrial pueden utilizarse dos estrategias: preestablecer un umbral fijo para todas las imágenes capturadas o bien calcular dinámicamente en cada imagen el umbral idóneo.

La opción más sencilla es establecer un umbral determinado y fijo para realizar la binarización. El valor más idóneo del umbral puede obtenerse analizando previamente los histogramas de las imágenes obtenidas en la aplicación. No es un mal método debido a que el coste computacional para determinar el umbral es nulo, pero debe ser implantado únicamente en entornos muy controlados, con una iluminación muy estable. La más mínima variación en la iluminación en la escena origina cambios en los niveles de gris de la imagen que invalidaría la idoneidad del umbral preestablecido.

Cuanta más anchura tenga el valle en el histograma, más fiable va a ser trabajar con un umbral fijo porque existirá un mayor margen de error. La anchura del valle puede absorber las oscilaciones en los niveles de gris que pueda haber por variaciones en la luz y hacer que el sistema funcione correctamente aun cuando existan perturbaciones. Una buena configuración del sistema de iluminación permitirá maximizar la distancia entre los dos picos que aparecen en el histograma correspondiente al objeto y al fondo.

Otra forma más ortodoxa que trabajar con un umbral fijado y mucho más robusta ante cambios en la iluminación, es calcular a partir de la información del histograma de cada imagen cuál es el valor más idóneo como umbral. El objetivo de esta técnica es obtener para cada imagen en escala de grises, la mejor imagen binaria localizando el umbral óptimo. De este modo se pueden evitar problemas que aparecen al tener un umbral fijo como puede ser el desplazamiento del histograma debido a cambios en la iluminación. Este tipo de binarización se conoce con el nombre de binarización adaptativa. Este tipo de binarización analiza los diferentes niveles de gris de la imagen y halla el umbral óptimo para cada zona de la imagen. En la parte derecha de la Ilustración 2.16 se muestra una imagen con una binarización con umbral fijo y en la parte izquierda se muestra con la binarización adaptativa.


**Ilustración 2.2.16 Diferentes binarizaciones**

Aunque estos algoritmos hallan el umbral más adecuado para cada imagen, hay que recordar que para obtener buenos resultados es necesario trabajar con imágenes que tengan un histograma bimodal. En imágenes con histogramas relativamente planos tampoco se puede obtener una buena representación con dos niveles de gris, aunque se emplee una búsqueda automática del umbral.

A continuación, se expone un algoritmo sencillo no optimizado de elección de un umbral que funciona bien siempre que el histograma presente dos modos. Se calculan el valor de nivel de gris medio y su varianza para cada uno de los dos modos del histograma, uno entre 0 y k y el otro en k y N. El algoritmo busca iterativamente el valor m para el que la suma ponderada de las varianzas el mínimo.

Sea una imagen I de N niveles de gris, el pseudocódigo para la binarización automática es la siguiente:

- Calcular el histograma de I.
- Para cada valor k del nivel de gris, generar dos poblaciones dividiendo el histograma en k.
- Calcular las varianzas de los grupos generados.
- Calcular la suma ponderada de estas dos varianzas (ponderar con el porcentaje de píxeles de su población).
- Guardar este valor de la suma en un vector de N elementos.
- Buscar el índice m correspondiente a la suma mínima de las varianzas.
- Binarizar la imagen en el umbral m.

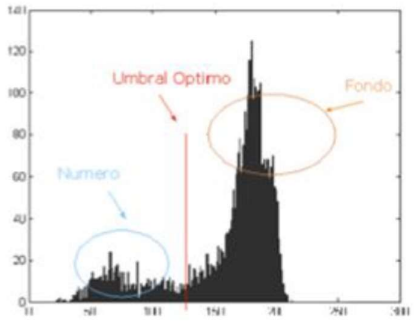


**Ilustración 2.2.17 Histograma con el umbral óptimo (Tutorial Opencv)**

Teniendo un histograma bimodal como el de la Ilustración 2.17, se observa de manera clara como el umbral óptimo es el que separa las dos crestas del gráfico, ya que está separando entre dos intensidades bien contrastadas y diferenciadas.

#### 2.2.15.4. Etiquetado. Extracción de características

En este apartado se va a exponer en qué consiste el etiquetado y cómo se realiza. A partir de las etiquetas creadas, se van a comentar las características que se pueden extraer de las mismas.

##### 2.2.15.4.1. Etiquetado

La mayoría de las aplicaciones en visión artificial están destinadas o requieren encontrar y clasificar cada uno de los elementos que encuentran dentro de la imagen. La forma de hacerlo es empleando el etiquetado, el cual asigna un mismo valor a todos los píxeles que forman parte de un mismo objeto en una imagen.

Con ello se podrá contar más adelante los elementos, ver si alguno no es apto, obtener características de cada uno de ellos, etc.

El empleo de esta técnica permitirá en pasos futuros poder aplicar otros algoritmos a la imagen, para poder procesar correctamente sus elementos como encontrar la orientación, ejes de inercia, etc.

Hay que tener claro que no se devuelve ninguna imagen modificada, ya que sólo se pretende obtener información de la imagen de entrada. Con esta información luego se trabajará. Como imagen tal, no se va a trabajar con el valor de cada pixel sino con la etiqueta que lleve, es decir, en esta función es indiferente el color del píxel, su nivel de gris, etc. Lo que importa es saber a qué objeto pertenece (fondo o alguno de los objetos).

Para un correcto funcionamiento de este proceso, es necesario trabajar con una imagen binaria y los elementos no deben estar conectados entre sí, ya que esos elementos conexionados se tomarán como uno solo y esta operación no daría sus frutos. Para eliminar dichas conexiones se emplean operaciones morfológicas, más concretamente apertura o cierre. Se puede decir que éste es uno de los casos en los que una imagen binaria aporta información que, ni por asomo, podría suministrar una imagen en color o escala de grises.

La principal precaución a tener en cuenta es el diferenciar a partir de una imagen binaria cual es el fondo y cuáles son los objetos. Unos corresponderán a partes blancas y otros a partes negras, pero hay que identificarlo. Los objetos siempre han de corresponder a partes blancas para poder etiquetarlos. En el caso de que no coincida, se realizará una inversión de la imagen.

Para realizar el algoritmo, en primer lugar, hay que enumerar cada píxel que compone la imagen como se muestra en la Ilustración 2.18.

A continuación, el algoritmo agrupa las etiquetas por proximidad, de modo que a cada grupo de píxeles conectados les asigna un valor, el que sea el menor de todos ellos (Ilustración 2.19).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 58 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

**Ilustración 2.2.18 Enumeración de los píxeles (Etiquetado y extracción de características.)**

El etiquetado necesita recorrer la imagen varias veces para clasificar debidamente las etiquetas mínimas. Se necesitará una variable (etiqueta) de valor inicial 1 y que deba ir incrementándose según se encuentre píxeles de valor 0 (negro).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 45 | 49 | 50 |
| 51 | 52 | 53 | 54 | 45 | 56 | 58 | 45 | 59 | 60 |
| 61 | 62 | 63 | 64 | 45 | 45 | 45 | 45 | 69 | 70 |
| 71 | 72 | 73 | 74 | 45 | 45 | 45 | 45 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

**Ilustración 2.2.19 Enumeración de los píxeles de un objeto (Etiquetado y extracción de características.)**

Es necesario que no exista ningún tipo de ruido en el etiquetado, ya que provocaría que hubiese más etiquetas de las que realmente hay que tener, lo cual provocaría errores. Esto es debido a que los píxeles aislados se consideran como objetos. Para que este problema no aparezca, se utiliza alguna de las operaciones morfológicas ya mencionadas.

#### 2.2.15.5. Extracción de características

La visión humana es capaz de identificar con casi perfecta claridad los objetos, y es fácil comparar tamaños entre varios objetos y asimilar características. En este flanco la visión artificial para conseguir emular a la humana se basa en las características de ésta, y la supera cuando las tareas son repetitivas y monótonas. Debido al cansancio humano se provoca un cúmulo de imprecisiones que favorecen el desarrollo de tecnologías basadas en la visión.

Una vez realizado el etiquetado, se pueden extraer una serie de características de cada objeto. Con estas características se puede decidir si un elemento posee los parámetros buscados, algo que ofrece de manera automática unos resultados similares a los que ofrece el cerebro humano.

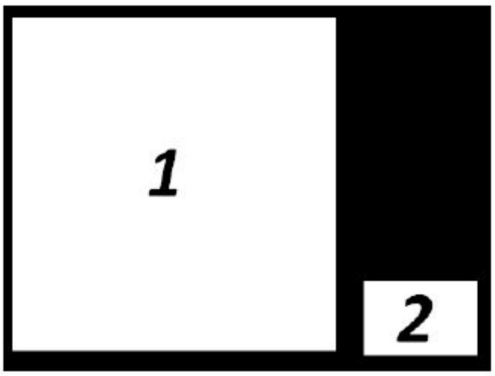


**Ilustración 2.2.20 Etiquetas de una imagen (Etiquetado y extracción de características.)**

Entre todas las características que se pueden medir, se pueden destacar las siguientes: área, centro de gravedad, ejes de inercia, número de Euler, perímetro y número de agujeros.

### 2.2.16. Otras operaciones frecuentemente utilizadas

En este apartado se han incluido dos técnicas de tratamiento para imágenes binarias. Estas operaciones son la eliminación de regiones según su área, y el rellenado de ilustraciones. Ambas técnicas son muy útiles, ya que acomodan la imagen y la llevan a una situación muy parecida, pero con más posibilidades de realizar un algoritmo robusto y que logre el objetivo.

#### 2.2.16.1. Eliminación de regiones según su área

Existe la posibilidad de diferenciar varios objetos o regiones independientes en una imagen binaria. Avanzando un poco más, se puede llegar a eliminar regiones según su área. Así, por ejemplo, si se tiene una imagen binaria formada por varios objetos o etiquetas, se pueden eliminar todas aquellas que no lleguen a un mínimo o umbral que se estipule.

De este modo, se trata de una herramienta muy útil para el filtrado de imágenes, ya que elimina de forma eficiente el ruido del tipo sal y pimienta, además de objetos menores que aparecen en la binarización y no representan ningún objeto deseado.

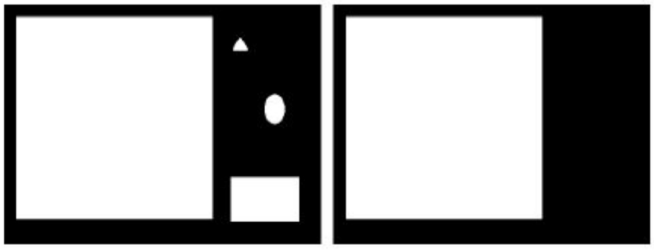

**Ilustración 2.2.21 Filtrado de imagen por área (Tratamiento de imágenes opencv)**

### 2.2.16.2. Rellenado de Ilustraciones

Otra herramienta muy eficaz para procesar imágenes es eliminar los agujeros presentes en ellas. El agujero se puede definir como una región de puntos negros independientes que están completamente rodeados por puntos blancos.

El objetivo principal de esta técnica es que las ilustraciones independientes sean más sólidas. Con ello se puede erosionar y eliminar con más margen, ya que se obtiene el contorno deseado con mayor facilidad.

**Ilustración 2.2.22 Rellenado de Ilustraciones (Tratamiento de imágenes opencv)**

### 2.2.16.3. Detección de contornos

Una de las técnicas más útiles es extraer los bordes en las imágenes. Un borde se puede definir como una frontera entre dos regiones cuyos niveles de gris son significativos.

La extracción de contornos a veces puede ser muy útil si se ejecuta en función de un determinado nivel de gris. La detección de contornos es usada en multitud de aplicaciones de visión artificial. Por ejemplo, a la hora de analizar las formas de ciertos objetos como por

ejemplo detectar fallos de imprenta en el etiquetado, analizando las formas que aparezcan en las etiquetas y permitiendo con un simple cambio de umbral la detección de distintos logotipos o formas en una misma etiqueta.

Para determinar los bordes es necesario el uso del gradiente, con el cual se detectan todas las posibles direcciones del píxel en el que está.

$$\overline{gradl} = \left(\frac{dl}{dx}, \frac{dl}{dy}\right)$$

$$|gradl| = \sqrt{\left(\frac{dl}{dx}\right)^2 + \left(\frac{dl}{dy}\right)^2}$$

**Fórmula 2.2.3 Gradiente para dirección del pixel**

En la práctica lo que se hace es aplicar máscaras o filtros a la imagen. Estas máscaras son matrices que van recorriendo cada píxel de la imagen y realizando una operación determinada.

Como ya se comentó, un borde es la frontera entre dos regiones con niveles de gris significativamente distintos como se aprecia en la Ilustración 2.23.

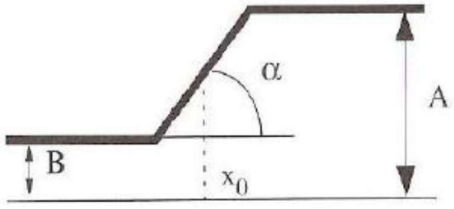


**Ilustración 2.2.23 Dos niveles de gris (Imágenes en escala de grises.)**

Donde A denota un nivel de gris alto; mientras que B denota un nivel de gris bajo.

Esta explicación es para una variación continua del nivel de gris. Pero en tratamiento digital de imágenes se tienen variables discretas. Ya que lo que se quiere es detectar cambios en la luminosidad, aparecen una serie de dificultades a la hora de detectar el píxel del contorno y entre las más distinguidas están las siguientes.

- Superficies con distinta orientación.
- Efectos de iluminación: existencia de sombras y reflejos que se pueden detectar como bordes por error.
- Cambios de textura.

#### 2.2.16.4. Transformada de Hough. Detección de rectas

La transformada de Hough es un algoritmo utilizado para el reconocimiento de patrones en imágenes que permite encontrar ciertas formas dentro de una imagen, como líneas, círculos y cualquier curva que siga una ecuación determinada dentro de una imagen.

Debido a que las imágenes son bidimensionales, para derivar se precisa del gradiente. En cada punto de la imagen el gradiente será diferente, alcanzando valores muy elevados en contornos. Este gradiente apunta en la dirección del cambio de intensidad.

La versión más simple consiste en encontrar líneas. Su modo de operación es principalmente estadístico y consiste en averiguar para cada punto si forma parte de una línea. De este modo se averiguan las posibles líneas de las que puede formar parte el punto. Eso se realiza para todos los puntos de la imagen y al final se determina que líneas fueron las que más puntos posibles tuvieron, siendo éstas las líneas en la imagen. Para aplicar la transformada de Hough, es necesario obtener primero una imagen binaria de los píxeles que forman parte de la frontera del objeto. Normalmente esta imagen es el resultado de un operador de contornos.

Un problema de este método surge cuando se quiere representar una recta que se aproxima a posiciones verticales (singularidad). En este caso, tanto la pendiente como la ordenada en el origen tienden a infinito. Para evitar este problema, lo que se hace es utilizar la representación normal de recta.

**Ilustración 2.2.24 Líneas detectadas por Hough (Muñoz M, 2014)**

Aunque este método está generalizado para la líneas rectas, también se puede aplicar para detectar cualquier forma que tenga la función g(x,c), donde x en un vector de coordenadas y c un vector de coeficientes. Esto permite hallar, entre otros, los puntos geométricos de los puntos de una circunferencia.

#### 2.2.16.5. Reconocimiento de caracteres de la imagen

Una vez que la imagen ha sido tratada y se tiene en la perspectiva deseada, hay que realizar la extracción de caracteres. Es decir, reconocer las letras y los números que componen la placa de la matrícula. Para ello se implementa un algoritmo OCR (*Optical Character Recognition*).

El reconocimiento de caracteres se puede realizar de diversas formas, tales como operaciones XOR con unas plantillas, extracción de características (número de euler, número de agujeros, etc.), etc.

Para reconocer los caracteres, se pueden crear algoritmos propios o utilizar, como en este trabajo, una librería ya existente. Para utilizar esta librería se crea un programa, el cual hace llamadas a las API de dicha librería. La librería utilizada es Tesseract, que es un motor OCR, el cual contiene una gran cantidad de funcionalidades.

La extracción de los caracteres va a permitir guardarlos en un archivo para que se puedan utilizar de la forma que el usuario desee.

Para poder aplicar un OCR, es importante que la calidad de la imagen donde se encuentran los caracteres sea alta, es decir, sin ruidos ni elementos extraños. También hay que tener en cuenta que la tipografía de texto a reconocer no debe ser poco común, ya que sería muy complicado detectar los caracteres.

Para su reconocimiento, se parte de la premisa de que los caracteres ya están segmentados, es decir, etiquetados. También es necesario que la imagen contenga sólo dos niveles de gris; esté binarizada.

Aun así, las imágenes reales no son perfectas, de modo que el OCR se encuentra con varios problemas:

- El dispositivo que toma la imagen puede introducir niveles de gris al fondo que no pertenecen a la imagen original.

- La resolución de los dispositivos puede introducir ruido en la imagen, afectando los píxeles que han de ser procesados.
- La distancia entre caracteres, que no siempre es la misma, puede producir errores de reconocimiento.
- La conexión de dos o más caracteres por píxeles comunes también puede producir errores.

Todos los algoritmos OCR tienen como finalidad diferenciar texto de una imagen cualquiera. Para hacerlo se basan en cuatro etapas, que se muestran en la Ilustración 2.25.

- **Binarización.** Los algoritmos OCR parten de una imagen binaria, de modo que es necesario convertir la imagen de entrada (color, escala de grises, etc.) en una imagen en blanco y negro. Al realizar esto, se conservan las propiedades esenciales de la imagen. El resultado de este proceso es una imagen en blanco y negro donde quedan marcados de manera clara los contornos de los caracteres y símbolos de la imagen.
- **Segmentación de la imagen.** Es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación implica la detección mediante el etiquetado de los contornos o regiones de la imagen. Para ello se basa en la información de intensidad o información espacial. No existe un único método genérico para realizar la segmentación que sea lo suficientemente eficaz para el análisis de un texto. Aunque las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.
- **Adelgazamiento de componentes.** Una vez etiquetados los componentes de la imagen, se les tiene que aplicar un proceso de adelgazamiento para cada uno de ellos. Este procedimiento consiste en ir borrando sucesivamente los puntos de los contornos de cada componente de forma que se conserve su tipología. La eliminación de los puntos ha de seguir un esquema de barridos sucesivos para que la imagen siga teniendo las mismas proporciones que la original y conseguir que no se deforme.
- **Comparación con patrones.** En esta etapa se comparan los caracteres obtenidos con anterioridad con unos teóricos que están almacenados en una base de datos. Para realizar esta comparación existen diversos procedimientos: método de proyección, métodos estadísticos, métodos estructurales, etc.

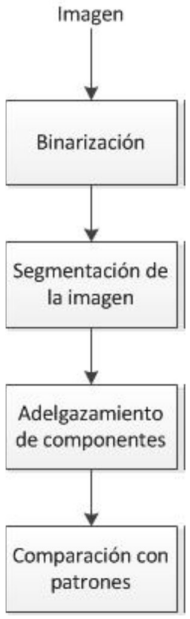


**Ilustración 2.2.25 Etapas de un algoritmo OCR**

## 2.3. Aprendizaje Automático

Aprendizaje Automático (*Machine Learning* en inglés) es una rama de la informática que estudia el diseño de algoritmos con la capacidad de “aprender”. Un subcampo sería el *Deep Learning* (aprendizaje profundo), que es una serie de técnicas que hacen uso de las redes neuronales artificiales profundas, es decir con más de una capa oculta, para imitar computacionalmente la estructura y funcionamiento del cerebro.

El aprendizaje automático es un campo de la Inteligencia Artificial que permite a las máquinas la capacidad de aprender. Esto elimina la necesidad de haber sido programados específicamente para resolver un problema concreto. (Géron, 2019)

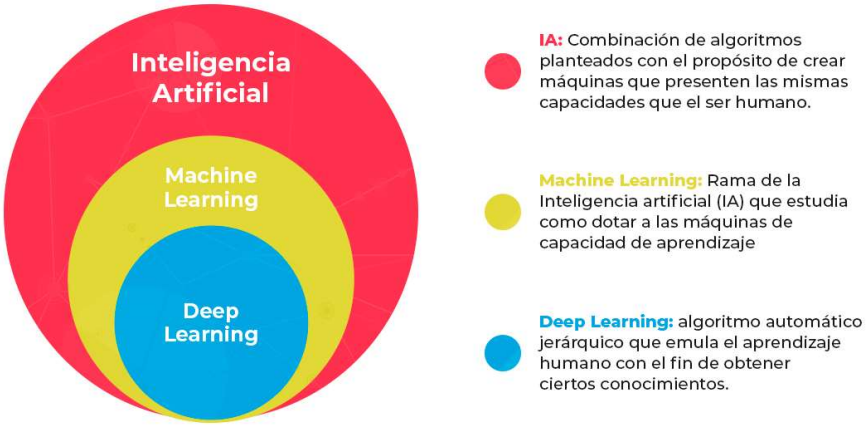


**Ilustración 2.3.1 Estructuración IA (¿Que es Machine Learning?, s.f.)**

Estos sistemas basados en ejemplos utilizan un conjunto de datos para aprender y se someten a un proceso de entrenamiento, etapa en la cual el algoritmo debe ser capaz de extraer las características más relevantes de los datos de entrenamiento para, posteriormente, poder generalizar de forma apropiada. Por tanto, se trata de una inducción del conocimiento.

La fase de entrenamiento es muy importante en este tipo de algoritmos y para ser realizado de forma satisfactoria el conjunto de datos de entrenamiento debe cumplir los siguientes requisitos:

- **Ser significativo:** El conjunto de datos debe ser lo suficientemente grande.
- **Ser representativo:** El conjunto de datos debe tener variedad suficiente en los datos. Si disponemos de unos datos desbalanceados corremos el riesgo de que el algoritmo se especialice en un subconjunto y no tendrá buena capacidad de generalización.

### 2.3.1. Tareas del aprendizaje automático

En la práctica existen diferentes tipos de tareas y cada una de ellas tiene sus propias características que son las que definen su tipo y cómo debe ser resuelto. De forma general estas tareas pueden ser clasificadas como *tareas predictivas* y *descriptivas*. Por tanto, el aprendizaje automático tiene que ser capaz de resolver distintos tipos de tareas. (Buduma, 2017)

### 2.3.2. Tareas predictivas

Se caracterizan fundamentalmente porque cada instancia de entrenamiento contiene la solución al problema. Dentro de las tareas predictivas se puede destacar la regresión y la clasificación. (soulmachine, 2017)

### 2.3.3. Regresión

Un problema es de regresión cuando necesitamos obtener como solución un valor continuo. De forma matemática se puede definir como la necesidad de encontrar una función *f* tal que $f:\mathbb{R}n \rightarrow \mathbb{R}$.

Un ejemplo de problema de regresión puede ser la predicción de la nota numérica que un alumno puede sacar en una determinada asignatura en función de sus datos personales, como puede ser la edad, gustos musicales, etc.

Una vez generado el modelo de regresión por un algoritmo de aprendizaje automático, es necesario evaluarlo. Las métricas utilizadas para regresión son las siguientes:

$$EMA = \frac{1}{n}\sum_{t=1}^{n}|e_t| \, , donde \; e_t = \; original_t - \; predicción_t$$

**Fórmula 2.3.1 Error Medio Absoluto (EMA)**

$$ECM = \frac{1}{n}\sum_{t=1}^{n}|e_t^2| \, , donde \; e_t = \; original_t - \; predicción_t$$

**Fórmula 2.3.2 Error Cuadrático Medio (ECM)**

$$RECM = \sqrt{\frac{1}{n}\sum_{t=1}^{n}|e_t^2|} \quad , donde \; e_t = \; original_t - \; predicción_t$$

**Fórmula 2.3.3 Raíz del Error Cuadrático Medio (RECM)**

- El ECM es diferenciable, lo que ayuda a encontrar los valores mínimos y máximos utilizando los métodos matemáticos de manera más efectiva.
- El ECM destaca grandes errores entre los pequeños.
- El RECM como se puede ver es la raíz cuadrada del ECM. Es fácil de interpretar en comparación con el ECM y utiliza valores absolutos más pequeños, lo que es útil para los cálculos informáticos.

### 2.3.4. Clasificación

Un problema es de clasificación cuando la solución se trata de un conjunto de valores discretos definidos previamente. De forma matemática puede ser definido como la necesidad de encontrar una función *f* tal que $f:\mathbb{R}n \rightarrow \{1,\ldots,k\}$.

Si el número total de clases es de exactamente dos, se conoce como clasificación binaria o biclase. Por el contrario, si el número de clases es superior a dos, se conoce con el nombre de clasificación multiclase.

Un ejemplo de clasificación podría ser un sistema de concesión de crédito, las clases en este caso serían dos: el crédito es concedido y el crédito no es concedido. Los atributos serían los datos de la persona que solicita el crédito, como podría ser su sueldo, edad, etc.

Es importante destacar, ya que se utilizará en el desarrollo del presente proyecto, que todo problema de clasificación puede ser convertido en un problema de regresión y viceversa, por lo que es posible resolver problemas de clasificación con algoritmos de regresión modificando el planteamiento del problema.

Las métricas utilizadas para medir su rendimiento son:

#### 2.3.4.1. Matriz de Confusión

Para evaluar el modelo de clasificación generado por un algoritmo de aprendizaje automático se utiliza el porcentaje de acierto o el porcentaje de error que se comete en el conjunto de prueba. (Géron, 2019)

$$PorcentajeAcierto = \frac{a}{n}$$

**Fórmula 2.3.4 Porcentaje de Acierto**

$$PorcentajeError = \frac{f}{n}$$

**Fórmula 2.3.5 Porcentaje de Error**

Donde:

- *a* es el número total de aciertos
- *f* es el número total de fallos
- *n* es el número total del conjunto de datos

Hay que tener en cuenta que estos resultados pueden ser engañosos, ya que por ejemplo en un hipotético dominio biclase donde el porcentaje de aparición de la primera clase es del 90% y la segunda clase solo se da en el 10% de los casos, si el modelo generado clasifica todas las entradas en la primera clase, se obtendría un 90% de acierto que es un porcentaje muy elevado y, sin embargo, este modelo sería inútil. Por tanto, para analizar más profundamente el resultado de un modelo se suele utilizar la matriz de confusión.


**Ilustración 2.3.2 Matriz de Confusión binaria. (Barrios, s.f.)**

#### 2.3.4.2. Métrica de Exactitud:

Esta métrica indica el número de elementos clasificados correctamente en comparación con el número total de artículos. Tiene limitaciones ya que no funciona bien con las clases desequilibradas que pueden tener muchos elementos de la misma clase en incluir algunas otras clases. (Géron, 2019)

$$PR = \frac{TP + TN}{TP + TN + FN + FP}$$

**Fórmula 2.3.6 El porcentaje de aciertos total**

Donde:

- TP= Total Positivo.
- TN= Total Negativo.
- FP= Falso Positivos.
- FN= Falsos Negativos.

#### 2.3.4.3. Métrica de Exhaustividad/ Sensibilidad:

Muestra la cantidad de verdaderos positivos que el modelo ha clasificado en función del número total de valores positivos.

$$TPR = \frac{TP}{TP + FN}$$

**Fórmula 2.3.7 El porcentaje de aciertos de la clase positivo**

#### 2.3.4.4. Métrica de Precisión:

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

$$TNR = \frac{TP}{FP + TP}$$

**Fórmula 2.3.8 El porcentaje de aciertos de la clase negativo.**

#### 2.3.4.5. Puntuación F1:

Esta métrica es la combinación de las métricas de precisión y exhaustividad y sirve de compromiso entre ellas. La mejor puntuación F1 es igual a 1 y la peor a 0.

$$F1 = 2 * \frac{precisión * exhaustividad}{precisión + exhaustividad}$$

**Fórmula 2.3.9 Puntuación F1**

## 2.4. Tipos de aprendizaje

Para poder aprender, las redes neuronales se sirven de un algoritmo de aprendizaje, formado por un conjunto de reglas que permiten a la red aprender a partir de los datos suministrados mediante la modificación de los pesos sinápticos de las conexiones entre neuronas y el umbral.

Los tipos de aprendizaje se dividen en tres:

- **Aprendizaje Supervisado:** Se introducen unos valores de entrada a la red, y los valores de salida que se generan se comparan con los valores de salida correctos. Si existe alguna diferencia se ajusta la red en consecuencia.

- **Aprendizaje de refuerzo:** Se introducen valores de entrada y lo único que se le indica a la red es si las salidas proporcionadas por esa son correctas o no.

- **Aprendizaje no Supervisado:** Para este tipo de aprendizaje no hay ningún tipo de guía, así que la red lo único que hace es reconocer patrones en los datos de entada y crear distintas categorías partiendo de estos patrones. De esta forma, cuando entra algún dato, tras el entrenamiento, la red podrá clasificarlo en una las categorías generadas.

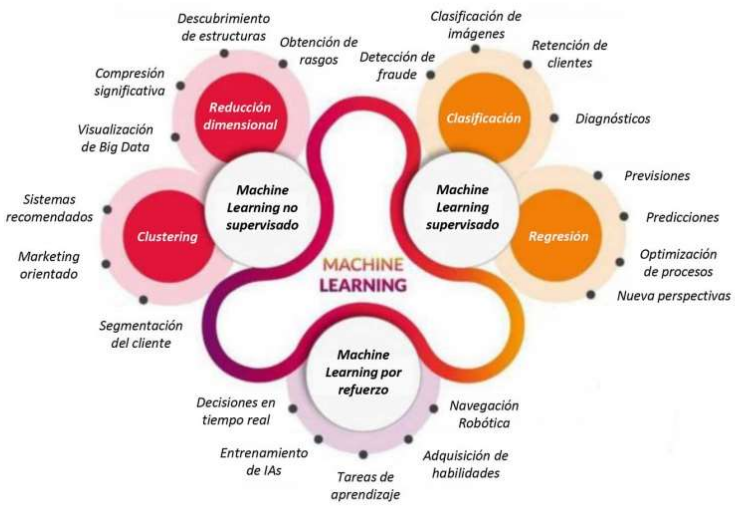


**Ilustración 2.4.1 Organización de Aprendizaje Automático (Canales Sectoriales, s.f.)**

Utilizando la matriz de confusión, se pueden obtener las siguientes tasas que proporcionan mucha información acerca del modelo generado:

## 2.5. Redes Neuronales

Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida.

El objetivo principal de este modelo es aprender modificándose automáticamente a si mismo de forma que puede llegar a realizar tareas complejas que no podrían ser realizadas mediante la clásica programación basada en reglas. De esta forma se pueden automatizar funciones que en un principio solo podrían ser realizadas por personas.

### 2.5.1. Componentes de las neuronas y modelo matemático

De forma similar a una neurona bilógica, una neurona artificial imita la estructura de un árbol, con un nodo de entrada y un nodo de salida, conectados entre ellos. (Géron, 2019)

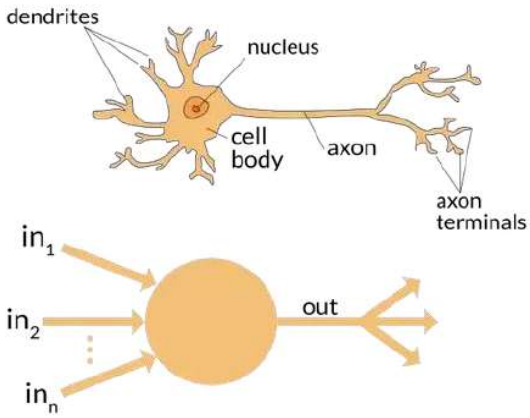


**Ilustración 2.5.1 Comparación de neurona y modelo matemático (Quereda, 2018)**

Existen los siguientes seis componentes en las neuronas artificiales:

- Nodos de entrada: Cada nodo va asociado a un valor numérico, que puede ser cualquier número real.
- Conexiones: Las conexiones entre nodos llevan asignados un peso, que también es un número real.
- Sumatorio: Teniendo en cuenta los valores de los nodos de entrada y los pesos se conmuta la suma ponderada.
- Función de activación o transferencia: La cual se encarga de devolver una salida a partir de un valor de entrada, en este caso, el sumatorio obtenido previamente. Las funciones que se utilizan buscan que las derivadas sean simples, reduciendo de este modo el coste computacional. Pueden ser funciones de activación lineal o funciones de activación no lineales.
- Nodo de salida: Valor asociado al valor de salida de la función de activación.
- *Bias* (sesgo); Es un parámetro crítico adicional que permite mover la función de activación hacia la derecha o a la izquierda, dependiendo de ello el éxito del aprendizaje.

Los perceptrones funcionan con valores umbral, que se utilizan con la finalidad de clasificar. Por ejemplo, una salida por encima del valor umbral indicaría que se instancia pertenece a una clase, y una salida con un valor por debajo del umbral, significa que esa entrada forma parte de la otra clase (Géron, 2019)

### 2.5.2. Perceptrón Multicapa (MLP):

Los perceptrones de varias capas, también conocidos como “redes neuronales *feed-forward*”, consisten en un conjunto de neuronas que se encuentran organizadas en distintas capas, normalmente dos o tres, pero no hay límite.

Dichas neuronas están conectadas entre ellas de forma que la salida de una capa sirve como entrada de la siguiente. Usualmente, los perceptrones de varias capas están totalmente conectados (*fully connected*) lo que quiere decir que cada perceptrón de una capa específica se une a cada perceptrón de la capa siguiente.

La arquitectura más utilizada actualmente sería:

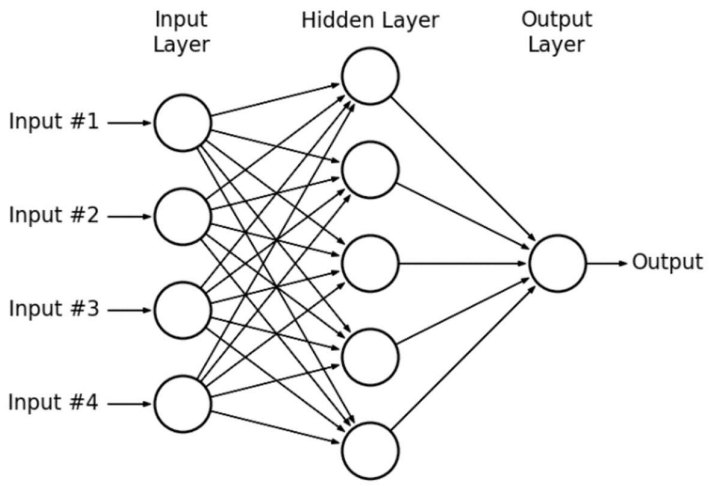


**Ilustración 2.5.2 Modelo de red (Quereda, 2018)**

- Una primera capa de entrada, que recibe la información del exterior.
- Una serie de capas ocultas (intermedias), las cuales son las encargadas de realizar el trabajo de la red.

- Una capa de salida, que proporciona el resultado.
- El número de capas y neuronas dependerá del tipo de aplicación a la cual se vaya a destinar la red neuronal.

### 2.6. Redes Neuronales Recurrentes (RNN).

Una red neuronal recurrente se ve bastante similar a una red neuronal tradicional, excepto que se agrega un estado de memoria a las neuronas.

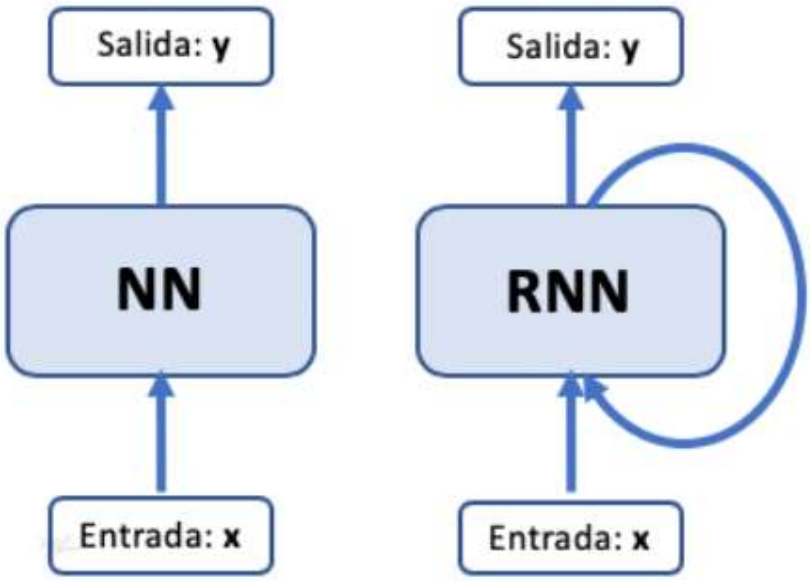


**Ilustración 2.5.3 Red Secuencial y Red Recurrente. (Rivera, 2018)**

- **Aplicaciones de RNN**

RNN tiene múltiples usos, especialmente cuando se trata de predecir el futuro. En el sector financiero, RNN puede ser útil para predecir los precios de las acciones o el signo de la dirección del mercado de valores (es decir, positivo o negativo).

RNN es útil para un coche autónomo, ya que puede evitar un accidente automovilístico anticipando la trayectoria del vehículo.

RNN es ampliamente utilizado en análisis de texto, subtítulos de imágenes, análisis de sentimientos y traducción automática. Por ejemplo, se puede utilizar una revisión de película para entender la sensación que el espectador percibió después de ver la película. Automatizar esta tarea es muy útil cuando la compañía cinematográfica no tiene tiempo suficiente para revisar, etiquetar, consolidar y analizar las revisiones. La máquina puede hacer el trabajo con un mayor nivel de precisión.

- **Limitaciones de RNN**

En teoría, se supone que RNN lleva la información hasta el tiempo. Sin embargo, es bastante difícil propagar toda esta información cuando el paso de tiempo es demasiado largo. Cuando una red tiene demasiadas capas profundas, se vuelve imposible de entrenar. Este problema se llama: problema de gradiente de fuga. Si recuerda, la red neuronal actualiza el peso usando el algoritmo de descenso de gradiente. Los gradientes se vuelven más pequeños cuando la red avanza hacia abajo a las capas inferiores.

En conclusión, los gradientes permanecen constantes, lo que significa que no hay espacio para mejorar. El modelo aprende de un cambio en el degradado; este cambio afecta a la salida de la red. Sin embargo, si la diferencia en el gradiente es demasiado pequeña (es decir, los pesos cambian un poco), la red no puede aprender nada y, por lo tanto, la salida. Por lo tanto, una red que enfrenta un problema de gradiente de fuga no puede converger hacia una buena solución.

- **Mejora LSTM**

Para superar el problema potencial del gradiente de fuga que enfrenta RNN, tres investigadores, Hochreiter, Schmidhuber y Bengio mejoraron el RNN con una arquitectura llamada Memoria Larga a Corto Plazo (LSTM). En resumen, LSMT proporciona a la red información relevante del pasado a tiempo más reciente. La máquina utiliza una mejor arquitectura para seleccionar y llevar la información a tiempo posterior.

## 2.7. Máquinas de Vector de Soporte (SVM).

El método de clasificación-regresión Máquinas de Vector Soporte *(Vector Support Machines*, SVMs) fue desarrollado en la década de los 90, dentro de campo de la ciencia computacional. Si bien originariamente se desarrolló como un método de clasificación binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión. SVMs ha resultado ser uno de los mejores clasificadores para un amplio abanico de situaciones, por lo que se considera uno de los referentes dentro del ámbito de aprendizaje estadístico y *machine learning*.

Las Máquinas de Vector Soporte se fundamentan en el *Maximal Margin Classifier*, que, a su vez, se basa en el concepto de hiperplano. A lo largo de este ensayo se introducen por orden cada uno de estos conceptos.

### 2.7.1. Hiperplano y *Maximal Margin Classifier*

En un espacio p-dimensional, un hiperplano se define como un subespacio plano y afín de dimensiones $p-1p-1$. El término afín significa que el subespacio no tiene por qué pasar por el origen. En un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, es decir, una recta. En un espacio tridimensional, un hiperplano es un subespacio de dos dimensiones, un plano convencional. Para dimensiones $p>3p>3$ no es intuitivo visualizar un hiperplano, pero el concepto de subespacio con $p-1p-1$ dimensiones se mantiene.

La definición matemática de un hiperplano es bastante simple. En el caso de dos dimensiones, el hiperplano se describe acorde a la ecuación de una recta:

$$\beta 0+\beta 1x_1+\beta 2x2 =0$$

**Fórmula 2.5.1 Ecuación Hiperplano**

Dados los parámetros β0, β1 y β2, todos los pares de valores $\mathbf{x}=(x_1,x_2)$ para los que se cumple la igualdad son puntos del hiperplano. Esta ecuación puede generalizarse para p-dimensiones:

$$\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp=0$$

**Fórmula 2.5.2 Ecuación Hiperplano para varias dimensiones**

y de igual manera, todos los puntos definidos por el vector ($\mathbf{x}=x_1,x_2,...,x_p$) que cumplen la ecuación pertenecen al hiperplano.

Cuando **x** no satisface la ecuación:

$$\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp<0$$

**Fórmula 2.5.3 Ecuación Hiperplano X<0**

o bien

$$\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp>0$$

**Fórmula 2.5.4 Ecuación Hiperplano X > 0**

El punto **x** cae a un lado o al otro del hiperplano. Así pues, se puede entender que un hiperplano divide un espacio p-dimensional en dos mitades. Para saber en qué lado del hiperplano se encuentra un determinado punto **x**, solo hay que calcular el signo de la ecuación.

La siguiente imagen muestra el hiperplano de un espacio bidimensional. La ecuación que describe el hiperplano (una recta) es 1+2x1+3x2=0. La región azul representa el espacio en el que se encuentran todos los puntos para los que $1+2x_1+3x_2>0$ y la región roja el de los puntos para los que $1+2x_1+3x_2<0$.

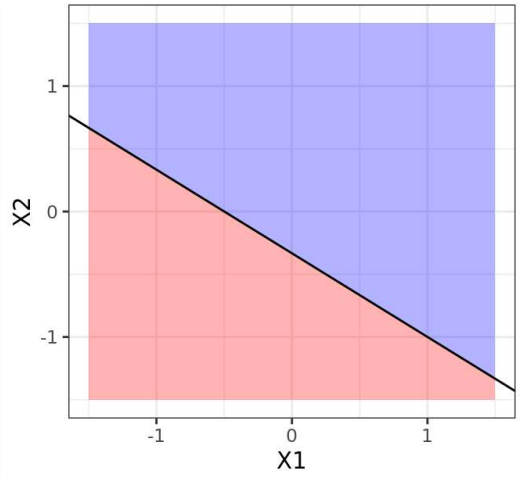


**Ilustración 2.5.4 Hiperplano de separación. (Rodrigo, 2017)**

### 2.7.2. Clasificación binaria empleando un hiperplano

Cuando se dispone de nn observaciones, cada una con predictores y cuya variable respuesta tiene dos niveles (de aquí en adelante identificados como (+1 y −1), se pueden emplear hiperplanos para construir un clasificador que permita predecir a que grupo pertenece una observación en función de sus predictores. Este mismo problema puede abordarse también con otros métodos (regresión logística, LDA, árboles de clasificación…) cada uno con ventajas y desventajas.

Para facilitar la comprensión, las siguientes explicaciones se basan en un espacio de dos dimensiones, donde un hiperplano es una recta. Sin embargo, los mismos conceptos son aplicables a dimensiones superiores.

### 2.7.3. Casos perfectamente separables linealmente

Si la distribución de las observaciones es tal que se pueden separar linealmente de forma perfecta en las dos clases (+1+y −1), entonces, un hiperplano de separación cumple que:

$$\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp>0, \text{ si } yi=1$$

**Fórmula 2.5.5 Caso si y=1**

$$\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp<0, \text{ si } yi=-1$$

**Fórmula 2.5.6 Caso si y=-1**

Al identificar cada clase como +1+1 ó −1−1, y dado que multiplicar dos valores negativos resultan en un valor positivo, las dos condiciones anteriores pueden simplificarse en una única:

$$yi\cdot(\beta 0+\beta 1x1+\beta 2x2+...+\beta pxp)>0, \text{ para } i=1...n$$

**Fórmula 2.5.7 Caso para i= 1 ..n**

Bajo este escenario, el clasificador más sencillo consiste en asignar cada observación a una clase dependiendo del lado del hiperplano en el que se encuentre. Es decir, la observación **x^*** se clasifica acorde al signo de la función $f(\mathbf{x^*})=\beta 0+\beta 1x*1+\beta 2x*2+...+\beta px$. Si $f(x*)$ es positiva, la observación se asigna a la clase +1, si es negativa, a la clase −1. Además, la magnitud de $f(x*)$ permite saber cómo de lejos está la observación del hiperplano y con ello la confianza de la clasificación.

La definición de hiperplano para casos perfectamente separables linealmente resulta en un número infinito de posibles hiperplanos, lo que hace necesario un método que permita seleccionar uno de ellos como clasificador óptimo.

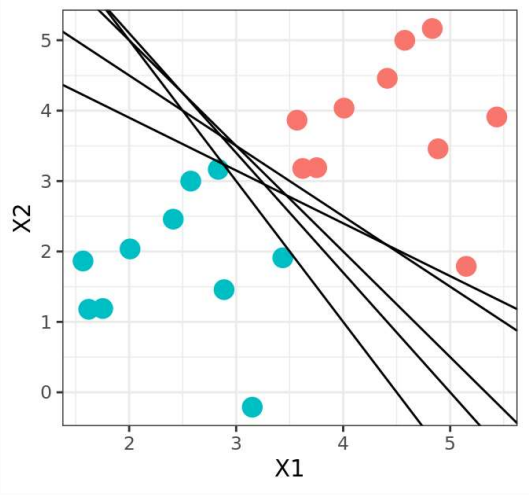


**Ilustración 2.5.5 Posibles hiperplanos de separación. (Rodrigo, 2017)**

La solución a este problema consiste en seleccionar como clasificador óptimo al que se conoce como *maximal margin hyperplane* o hiperplano óptimo de separación, que se corresponde con el hiperplano que se encuentra más alejado de todas las observaciones de entrenamiento. Para obtenerlo, se tiene que calcular la distancia perpendicular de cada observación a un determinado hiperplano. La menor de estas distancias (conocida como margen) determina como de alejado está el hiperplano de las observaciones de entrenamiento. El *maximal margin hyperplane* se define como el hiperplano que consigue un mayor margen, es decir, que la distancia mínima entre el hiperplano y las observaciones es lo más grande posible. Aunque esta idea suena razonable, no es posible aplicarla, ya que habría infinitos hiperplanos contra los que medir las distancias. En su lugar, se recurre a métodos de optimización.

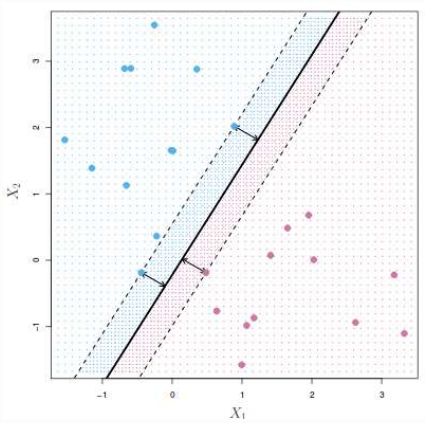


**Ilustración 2.5.6 Imagen *maximal margin* (Rodrigo, 2017)**

La imagen anterior muestra el *maximal margin hyperplane* para un conjunto de datos de entrenamiento. Las tres observaciones equidistantes respecto al *maximal margin hyperplane* se encuentran a lo largo de las líneas discontinuas que indican la anchura del margen. A estas observaciones se les conoce como vectores soporte, ya que son vectores en un espacio p-dimensional y soportan (definen) el *maximal margin hyperplane*. Cualquier modificación en estas observaciones (vectores soporte) conlleva cambios en el *maximal margin hyperplane*. Sin embargo, modificaciones en observaciones que no son vector soporte no tienen impacto alguno en el hiperplano.

### 2.7.4. casos cuasi-separables linealmente

El *maximal margin hyperplane* descrito en el apartado anterior es una forma muy simple y natural de clasificación siempre y cuando exista un hiperplano de separación. En la gran mayoría de casos reales, los datos no se pueden separar linealmente de forma perfecta, por lo que no existe un hiperplano de separación y no puede obtenerse un *maximal margin hyperplane*.

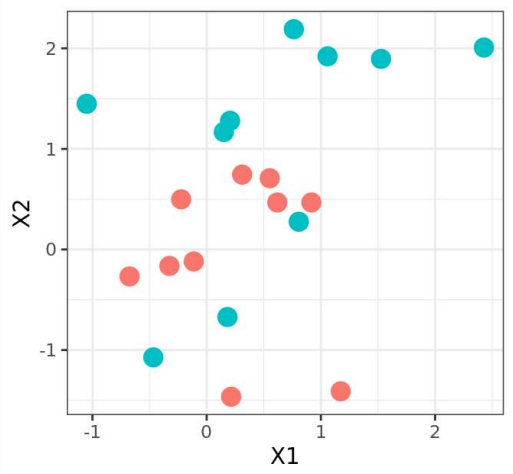


**Ilustración 2.5.7 Clases no separables linealmente. (Rodrigo, 2017)**

Para solucionar estas situaciones, se puede extender el concepto de *maximal margin hyperplane* para obtener un hiperplano que casi separe las clases, pero permitiendo que cometa unos pocos errores. A este tipo de hiperplano se le conoce como *Support Vector Classifier o Soft Margin*.

### *2.7.5. Support Vector Classifier o Soft Margin SVM*

El *Maximal Margin Classifier* descrito en la sección anterior tiene poca aplicación práctica, ya que rara vez se encuentran casos en los que las clases sean perfecta y linealmente separables. De hecho, incluso cumpliéndose estas condiciones ideales, en las que exista un hiperplano capaz de separar perfectamente las observaciones en dos clases, esta aproximación sigue presentando dos inconvenientes:

- Dado que el hiperplano tiene que separar perfectamente las observaciones, es muy sensible a variaciones en los datos. Incluir una nueva observación puede suponer cambios muy grandes en el hiperplano de separación (poca robustez).
- Que el *maximal margin hyperplane* se ajuste perfectamente a las observaciones de entrenamiento para separarlas todas correctamente suele conllevar problemas de *overfitting*.

Por estas razones, es preferible crear un clasificador basado en un hiperplano que, aunque no separe perfectamente las dos clases, sea más robusto y tenga mayor capacidad predictiva al aplicarlo a nuevas observaciones (menos problemas de *overfitting*). Esto es exactamente lo que consiguen los clasificadores de vector soporte, también conocidos como *soft margin classifiers o Support Vector Classifiers*. Para lograrlo, en lugar de buscar el margen de clasificación más ancho posible que consigue que las observaciones estén en el lado correcto del margen; se permite que ciertas observaciones estén en el lado incorrecto del margen o incluso del hiperplano. (Chollet, 2018)

La siguiente imagen muestra un clasificador de vector soporte ajustado a un pequeño set de observaciones. La línea continua representa el hiperplano y las líneas discontinuas el margen a cada lado. Las observaciones 2, 3, 4, 5, 6, 7 y 10 se encuentran en el lado correcto del margen (también del hiperplano) por lo que están bien clasificadas. Las observaciones 1 y 8, a pesar de que se encuentran dentro del margen, están en el lado correcto del hiperplano, por lo que también están bien clasificadas. Las observaciones 11 y 12, se encuentran en el lado erróneo del hiperplano, su clasificación es incorrecta. Todas aquellas observaciones que, estando dentro o fuera del margen, se encuentren en el lado incorrecto del hiperplano, se corresponden con observaciones de entrenamiento mal clasificadas.

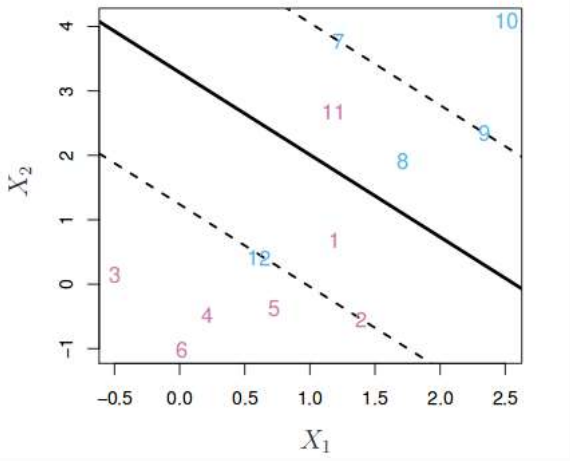


**Ilustración 2.5.8 Imagen clasificador vector soporte (Rodrigo, 2017)**

La identificación del hiperplano de un clasificador de vector soporte, que clasifique correctamente la mayoría de las observaciones a excepción de unas pocas, es un problema de optimización convexa. Si bien la demostración matemática queda fuera del objetivo de esta introducción, es importante mencionar que el proceso incluye un hiperparámetro de *tuning* CC. CC controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste. Si C=∞C=∞, no se permite ninguna violación del margen y por lo tanto, el resultado es equivalente al *Maximal Margin Classifier* (teniendo en cuenta que esta solución solo es posible si las clases son perfectamente separables). Cuando más se aproxima CC a cero, menos se penalizan los errores y más observaciones pueden estar en el lado incorrecto del margen o incluso del hiperplano. CC es a fin de cuentas el hiperparámetro encargado de controlar el balance entre *bias* y varianza del modelo. En la práctica, su valor óptimo se identifica mediante *cross-validation*.

El proceso de optimización tiene la peculiaridad de que solo las observaciones que se encuentran justo en el margen o que lo violan influyen sobre el hiperplano. A estas observaciones se les conoce como vectores soporte y son las que definen el clasificador obtenido. Esta es la razón por la que el parámetro CC controla el balance entre bias y varianza. Cuando el valor de CC es pequeño, el margen es más ancho, y más observaciones violan el margen, convirtiéndose en vectores soporte. El hiperplano está, por lo tanto, sustentado por más observaciones, lo que aumenta el *bias* pero reduce la varianza.

Cuando mayor es el valor de CC, menor el margen, menos observaciones serán vectores soporte y el clasificador resultante tendrá menor *bias* pero mayor varianza.

Otra propiedad importante que deriva de que el hiperplano dependa únicamente de una pequeña proporción de observaciones (vectores soporte), es su robustez frente a observaciones muy alejadas del hiperplano. Esto hace al método de clasificación vector soporte distinto a otros métodos tales como *Linear Discrimiant Analysis* (LDA), donde la regla de clasificación depende de la media de todas las observaciones.

### 2.7.6. Clasificador de Vector Soporte

El *Support Vector Classifier* descrito en los apartados anteriores consigue buenos resultados cuando el límite de separación entre clases es aproximadamente lineal. Si no lo es, su capacidad decae drásticamente. Una estrategia para enfrentarse a escenarios en los que la separación de los grupos es de tipo no lineal consiste en expandir las dimensiones del espacio original.

El hecho de que los grupos no sean linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. Las imágenes siguientes muestran como dos grupos, cuya separación en dos dimensiones no es lineal, sí lo es al añadir una tercera dimensión.

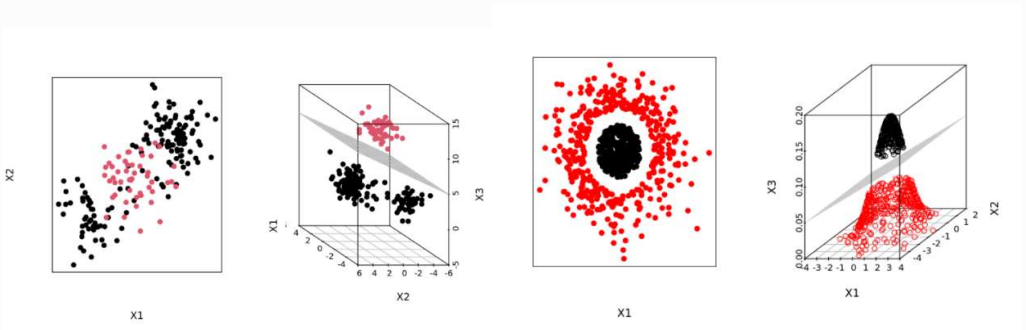


**Ilustración 2.5.9 Separación agregando una nueva dimensión. (Rodrigo, 2017)**

El método de Máquinas Vector Soporte (SVM) se puede considerar como una extensión del *Support Vector Classifier* obtenida al aumentar la dimensión de los datos. Los límites de separación lineales generados en el espacio aumentado se convierten en límites

de separación no lineales al proyectarlos en el espacio original.

#### 2.7.6.1. Aumento de la dimensión, kernels

Una vez definido que las Máquinas de Vector Soporte siguen la misma estratégica que el *Support Vector Classifier*, pero aumentando la dimensión de los datos antes de aplicar el algoritmo, la pregunta inmediata es ¿Cómo se aumenta la dimensión y qué dimensión es la correcta?

La dimensión de un conjunto de datos puede transformarse combinando o modificando cualquiera de sus dimensiones. Por ejemplo, se puede transformar un espacio de dos dimensiones en uno de tres aplicando la siguiente función:

$$f(x_1,x_2)=(x^2_1, \sqrt{2}x_1x_2,x^2_2)$$

**Fórmula 2.5.8 Transformación de 2 a 3 dimensiones**

Esta es solo una de las infinitas trasformaciones posibles, ¿Cómo saber cuál es la adecuada? Es aquí donde los kernel entran en juego. Un kernel (K) es una función que devuelve el resultado del producto punto entre dos vectores realizado en un nuevo espacio dimensional distinto al espacio original en el que se encuentran los vectores. Aunque no se ha entrado en detalle en las fórmulas matemáticas empleadas para resolver el problema de optimización, esta contiene un producto punto. Si se sustituye este producto punto por un kernel, se obtienen directamente los vectores soporte (y el hiperplano) en la dimensión correspondiente al kernel. Ha esto se le suele conocer como kernel *trick*, porque, con solo una ligera modificación del problema original, gracias a los kernels, se puede obtener el resultado para cualquier dimensión. Existen multitud de kernels distintos, algunos de los más utilizados son:

### 2.7.6.2. Kernel lineal

$$K(\mathbf{x},\mathbf{x'})=\mathbf{x}\cdot\mathbf{x'}$$

**Fórmula 2.5.9 Kernel Lineal**

Si se emplea un Kernel lineal, el clasificador *Support Vector Machine* obtenido es equivalente al *Support Vector Classifier*.

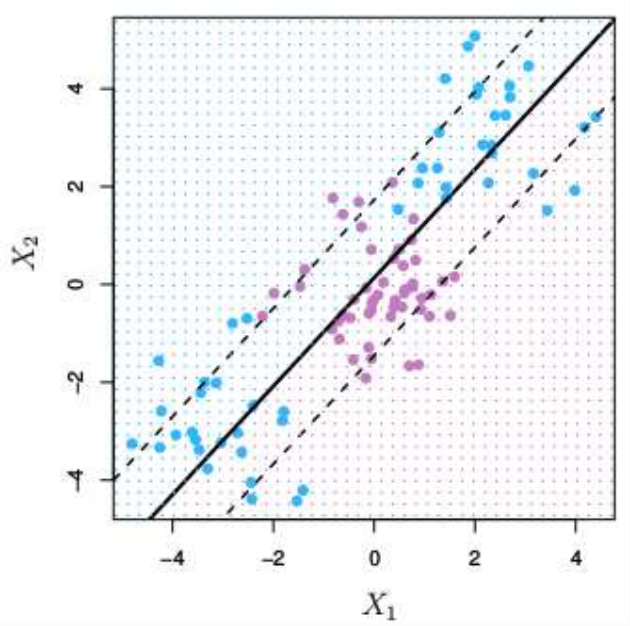


**Ilustración 2.5.10 Imagen SVM con kernel lineal (Rodrigo, 2017)**

### 2.7.6.3. Kernel polinómico

$$K(\mathbf{x},\mathbf{x'})=(\mathbf{x}\cdot\mathbf{x'}+c)^{d}$$

**Fórmula 2.5.10 Kernel Polinómico**

Cuando se emplea d=1 y c=0, el resultado es el mismo que el de un kernel lineal. Si d>1, se generan límites de decisión no lineales, aumentando la no linealidad a medida que aumenta d. No suele ser recomendable emplear valores de dd mayores 5 por problemas de *overfitting*.

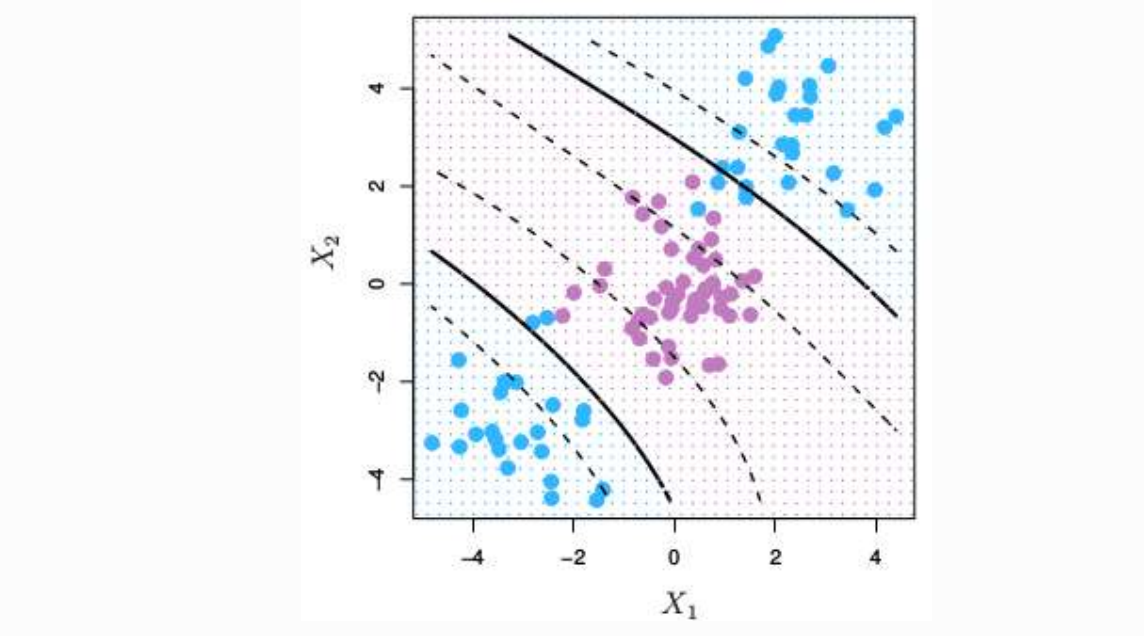


Ilustración **2.5.11 Imagen SVM con una kernel polinómico de grado 3 (Rodrigo, 2017)**

### 2.7.6.4. Gaussian Kernel (RBF)

$$K(\mathbf{x},\mathbf{x'})=\exp(-\gamma||\mathbf{x}-\mathbf{x'}||^2)$$

**Fórmula 2.5.11 Kernel Gausiano**

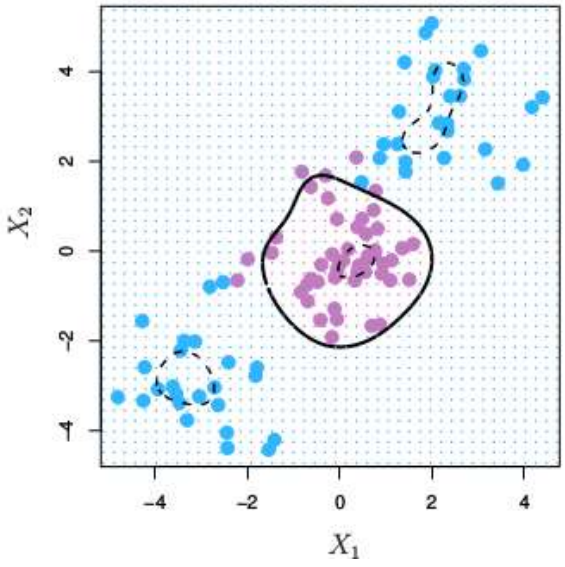


**Ilustración 2.5.12 Imagen SVM con una kernel radial (Rodrigo, 2017)**

El valor de γγ controla el comportamiento del kernel, cuando es muy pequeño, el modelo final es equivalente al obtenido con un kernel lineal, a medida que aumenta su valor, también lo hace la flexibilidad del modelo.

Los kernels descritos son solo unos pocos de los muchos que existen. Cada uno tiene una serie de hiperparámetros cuyo valor óptimo puede encontrarse mediante validación cruzada. No puede decirse que haya un kernel que supere al resto, depende en gran medida de la naturaleza del problema que se esté tratando. Este kernel tiene dos ventajas: que solo tiene dos hiperparámetros que optimizar (γγ y la penalización CC común a todos los SVM) y que su flexibilidad puede ir desde un clasificador lineal a uno muy complejo.

### 2.7.7. *Support Vector Machines* para más de dos clases

El concepto de hiperplano de separación en el que se basan los SVMs no se generaliza de forma natural para más de dos clases. Se han desarrollado numerosas estrategias con el fin de aplicar este método de clasificación a situaciones con k>2-clases, de entre ellos, los más empleados son*: one-versus-one, one-versus-all* y DAGSVM.

#### *2.7.7.1. One-versus-one*

Supóngase un escenario en el que hay K > 2 clases y que se quiere aplicar el método de clasificación basado en SVMs. La estrategia de *one-versus-one* consiste en generar un total de K(K-1)/2 SVMs, comparando todos los posibles pares de clases. Para generar una predicción se emplean cada uno de los K(K-1)/2 clasificadores, registrando el número de veces que la observación es asignada a cada una de las clases. Finalmente, se considera que la observación pertenece a la clase a la que ha sido asignada con más frecuencia. La principal desventaja de esta estrategia es que el número de modelos necesarios se dispara a medida que aumenta el número de clases, por lo que no es aplicable en todos los escenarios.

Si a la función svm() recibe como variable respuesta un factor con más de dos niveles, realiza automáticamente una clasificación multi-clase empleando el método *one-versus-one*.

#### *2.7.7.2. One-versus-all*

Esta estrategia consiste en ajustar K SVMs distintos, cada uno comparando una de las K clases frente a las restantes K-1 clases. Como resultado, se obtiene un hiperplano de clasificación para cada clase. Para obtener una predicción, se emplean cada uno de los K clasificadores y se asigna la observación a la clase para la que la predicción resulte positiva. Esta aproximación, aunque sencilla, puede causar inconsistencias, ya que puede ocurrir que más de un clasificador resulte positivo, asignando así una misma observación a diferentes clases. Otro inconveniente adicional es que cada clasificador se entrena de forma no balanceada. Por ejemplo, si el set de datos contiene 100 clases con 10 observaciones por clase, cada clasificador se ajusta con 10 observaciones positivas y 990 negativas.

#### 2.7.7.3. DAGSVM

DAGSVM (*Directed Acyclic Graph* SVM) es una mejora del método *one-versus-one*. La estrategia seguida es la misma, pero consiguen reducir su tiempo de ejecución eliminando comparaciones innecesarias gracias al empelo de una *directed acyclic graph* (DAG). Supóngase un set de datos con cuatro clases (A, B, C, D) y 6 clasificadores entrenados con cada posible par de clases (A-B, A-C, A-D, B-C B-D, C-D). Se inician las comparaciones con el clasificador (A-D) y se obtiene como resultado que la observación pertenece a la clase A, o lo que es equivalente, que no pertenece a la clase D. Con esta información se pueden excluir todas las comparaciones que contengan la clase D, puesto que se sabe que no pertenece a este grupo. En la siguiente comparación se emplea el clasificador (A-C) y se predice que es A. Con esta nueva información se excluyen todas las comparaciones que contengan C. Finalmente solo queda emplear el clasificador (A-B) y asignar la observación al resultado devuelto. Siguiendo esta estrategia, en lugar de emplear los 6 clasificadores, solo ha sido necesario emplear 3. DAGSVM tiene las mismas ventajas que el método *one-versus-one* pero mejorando mucho el rendimiento.

### 2.7.8. Coste computacional de SVM

Debido a la utilización de kernels, en el ajuste de un SVM participa una matriz n x n, donde n es el número de observaciones de entrenamiento. Por esta razón, lo que más

influye en el tiempo de computación necesario para entrenar un SVM es el número de observaciones, no el de predictores.

## 2.8. Aprendizaje Profundo

El aprendizaje profundo (o llamado *Deep Learning* en inglés) es una rama del aprendizaje automático. A diferencia de los algoritmos tradicionales de aprendizaje automático, muchos de los cuales tienen una capacidad finita de aprendizaje independientemente de cuántos datos adquieran, los sistemas de aprendizaje profundo pueden mejorar su rendimiento al poder acceder a un mayor número de datos, o lo que es lo mismo, hacer que la máquina tenga más experiencia. Una vez que las máquinas han conseguido suficiente experiencia mediante el aprendizaje profundo, pueden ponerse a trabajar para realizar tareas específicas como conducir un coche, detectar hierbajos en un campo de cultivo, detectar enfermedades, inspeccionar maquinaria para identificar errores, etc. (Chollet, 2018)

Las redes de aprendizaje profundo aprenden mediante la detección de estructuras complejas en los datos que reciben. Al crear modelos computacionales compuestos por varias capas de procesamiento, las redes pueden crear varios niveles de abstracción que representen los datos.


**Ilustración 2.8.1 Diferencia entre el aprendizaje profundo y el aprendizaje máquina clásico. (Gomez, 2019)**

Por ejemplo, un modelo de aprendizaje profundo conocido como «redes neuronales convolucionales», se puede entrenar como un gran número (de millones) de imágenes; por ejemplo, las que contienen gatos. Este tipo de red neuronal normalmente aprende de los píxeles que contienen las imágenes que adquiere. Puede clasificar grupos de píxeles que representan las características de un gato, con grupos de características como las garras, las orejas y los ojos, lo que indicaría la presencia de un gato en la imagen.

El aprendizaje profundo es totalmente distinto del aprendizaje automático convencional. En este ejemplo, un experto en dominios necesitaría dedicar mucho tiempo a diseñar un sistema de aprendizaje automático convencional que detecte las características que representan un gato. Con el aprendizaje profundo, todo lo que necesita es ofrecer al sistema un gran número de imágenes de gatos, tras lo cual el sistema aprende de forma autónoma las características que representan un gato.

En muchas tareas, como la visión por ordenador, el reconocimiento de voz, la traducción automática y la robótica, el rendimiento de los sistemas de aprendizaje profundo supera enormemente el de los sistemas de aprendizaje automático convencional. Esto no significa que crear sistemas de aprendizaje profundo sea relativamente fácil en comparación con los sistemas de aprendizaje automático convencional. Si bien el reconocimiento de características es autónomo en el aprendizaje profundo, hay que ajustar miles de hiperparámetros (botones) para que el modelo de aprendizaje profundo sea realmente efectivo.

### 2.8.1. Ejemplos de casos prácticos de aprendizaje profundo

- **Robótica:**

Muchos de los adelantos recientes que se han producido en la robótica se han producido por avances en la IA y el aprendizaje profundo. Por ejemplo, gracias a la IA, los robots pueden percibir y responder a su entorno. Esta capacidad aumenta la variedad de funciones que pueden realizar: desde desplazarse por plantas de almacenes a ordenar y manejar objetos de diferente naturaleza, frágiles o mezclados. Algo tan simple como elegir una fresa es una tarea muy sencilla para una persona, pero ha sido sumamente difícil conseguir que lo haga un robot. A medida que progrese la IA, esos avances irán mejorando las cosas que puede hacer los robots.

El desarrollo que se está produciendo en la IA significa que podemos esperar que los robots del futuro se puedan usar cada vez más como ayudantes de humanos. No se utilizarán

para comprender preguntas y responderlas, como sucede hoy en día; sino que podrán actuar según gestos y comandos de voz, e incluso anticiparse al próximo movimiento de un trabajador. Actualmente, los robots colaborativos ya trabajan junto a personas, pero realizando las tareas que mejor se adaptan a cada uno por separado.

- **Agricultura:**

La IA tiene el potencial de revolucionar la agricultura. En este momento, el aprendizaje profundo permite a los granjeros implantar equipos que pueden ver y distinguir entre cultivos y malas hierbas. Gracias a esta capacidad, las máquinas recolectoras pueden rociar de forma selectiva los herbicidas sobre las malas hierbas sin tocar el resto. La maquinaria agrícola que utiliza la visión por ordenador con aprendizaje profundo puede incluso optimizar cada una de las plantas de un terreno al rociar selectivamente herbicidas, fertilizantes, fungicidas, insecticidas y productos químicos. Además de reducir el uso de herbicidas y mejorar la producción agrícola, el aprendizaje profundo se puede extender a otras operaciones agrícolas, como la aplicación de fertilizantes, el riego y la recolección.

- **Exploración médica y servicios sanitarios:**

El aprendizaje profundo ha sido particularmente eficaz en las exploraciones médicas, debido a la disponibilidad de datos de gran calidad y la capacidad que tienen las redes neuronales convolucionales de clasificar imágenes. Por ejemplo, el aprendizaje profundo puede ser tan efectivo como un dermatólogo al clasificar los cánceres de piel, o incluso más. Varios proveedores ya han recibido la aprobación de la FDA para aplicar algoritmos de aprendizaje profundo con la finalidad de emitir diagnósticos, incluido el análisis de imágenes para oncología y enfermedades de la retina. El aprendizaje profundo también está realizando avances significativos en la mejora de la calidad de los servicios sanitarios al predecir eventos médicos a partir de los datos de los historiales médicos.

- **El futuro del aprendizaje profundo:**

En la actualidad, existen varias arquitecturas de red neuronal optimizadas para determinados tipos de entradas y tareas. Las redes neuronales convolucionales son muy buenas clasificando imágenes. Otra forma de arquitectura de aprendizaje profundo utiliza redes neuronales recurrentes para procesar datos secuenciales. Tanto los modelos de redes

neuronales convolucionales como las recurrentes realizan lo que se conoce como aprendizaje supervisado, lo que significa que se tienen que proveer con grandes cantidades de datos para poder aprender. En el futuro, los tipos más sofisticados de IA usarán el aprendizaje no supervisado. Se está dedicando una gran cantidad de investigación a la tecnología de aprendizaje no supervisado y semi supervisado.

El aprendizaje de refuerzo es un paradigma ligeramente diferente al aprendizaje profundo en el que un agente aprende por ensayo y error en un entorno simulado únicamente de recompensas y castigos. Las extensiones de aprendizaje profundo en este dominio se conocen como aprendizaje de refuerzo profundo (DRL). Se ha producido un considerable progreso en este campo, como se demuestra en los programas de DRL que han logrado vencer a humanos en el antiguo juego del Go.

Diseñar arquitecturas de redes neuronales que solucionen problemas es increíblemente difícil y es incluso más complejo cuando hay que ajustar muchos hiper parámetros y se pueden elegir más funciones de pérdida para optimizarlas. Se ha realizado una gran cantidad de investigaciones para aprender arquitecturas de redes neuronales en buen estado de forma autónoma. Aprender a aprender, o lo que se conoce también como «meta aprendizaje» o «AutoML», sigue progresando constantemente.

Las redes neuronales artificiales actuales se basaban en el entendimiento que se tenía en los años 50 de cómo los cerebros humanos procesan la información. La neurociencia ha progresado bastante desde entonces y las arquitecturas de aprendizaje profundo son ahora tan sofisticadas que parecen mostrar estructuras como células de red, presentes en cerebros neuronales biológicos que se usan para la navegación. Tanto la neurociencia como el aprendizaje profundo pueden beneficiarse entre sí de la «polinización cruzada» de ideas y es muy probable que estos campos se fundan en uno en algún momento.

Ya no usamos ordenadores mecánicos y, en algún momento, tampoco usaremos los digitales. En su lugar, usaremos una nueva generación de ordenadores cuánticos. Se han producido varios avances en la computación cuántica en los últimos años y los algoritmos de aprendizaje pueden, de hecho, beneficiarse de la increíble cantidad de computación disponible que los ordenadores cuánticos pueden ofrecer. También se podrían usar algoritmos de aprendizaje para comprender el resultado que se obtiene de los ordenadores cuánticos probabilísticos. El aprendizaje automático cuántico es una rama muy activa del aprendizaje automático y, con la primera conferencia internacional sobre aprendizaje automático cuántico celebrada en 2018, se pronostica un buen comienzo.

## 2.9. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNNs) son un tipo de Redes Neuronales muy efectivas en las tardes de reconocimiento y clasificación.

La diferencia respecto al resto es que se asume que la entrada será una imagen, lo que permite codificar una serie de propiedades en la arquitectura. El resultado es la asignación de una probabilidad a cada una de ls distintas categorías. La suma de todas las probabilidades debería ser 1.

Las cuatro principales operaciones realizadas en una red convolucional son:

- Convolución.
- Funciones de Activación
- *Pooling* o Submuestreo
- *Clasificación (Fully Connected Layer)*

### 2.9.1. Convolución:

La finalidad de este paso es extraer características de la imagen de entada (generando un mapa de características) utilizando un conjunto de filtros que se pueden aprender.

La red neuronal aprende los valores de los filtros a aplicar por si sola durante el proceso de entrenamiento, pero hay que especificarle una serie de parámetros como el número de filtros, el tamaño del filtro, la arquitectura de la red, entren otros. (Chollet, 2018)

El Mapa de Características (*Feature Map*) se controla mediante tres parámetros:

- **Profundidad (*Depth*):** Corresponde al número de filtros utilizados en la operación de convolución. Por ejemplo, si aplicamos 3 filtros, se obtienen 3 mapas de profundidad distintos que se guardan en una matriz, la cual tendrá una profundidad de 3.
- **Paso (*Stride*):** Número de pixeles por los que movemos el filtro sobre la matriz de entada. Cuanto mayor es el número de pasos, se producen mapas de características más pequeños.

- ***Zero-padding*:** Este parámetro nos permite conservar el tamaño original de entrada. Se utiliza para añadir un borde de pixeles con valor de cero alrededor de la imagen de entrada.

### 2.9.2. Funciones de Activación:

La función de activación se encarga de devolver una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida en un rango determinado como (0,1) o (-1,1).

Se buscan funciones que las derivadas sean simples, para minimizar con ello el coste computacional.

- **Tipos de función de activación:**

  - ***Sigmoid* – Sigmoide**

La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0. (Buduma, 2017)

$$f(x) = \frac{1}{1 - e^{-x}}$$

**Fórmula 2.8.1 Función Sigmoide**

Características de la función sigmoide:

- Satura y mata el gradiente.
- Lenta convergencia.
- No está centrada en el cero.
- Esta acotada entre 0 y 1.
- Buen rendimiento en la última capa.

- **Tanh – *Tangent Hyperbolic* – Tangente hiperbólica**

La función tangente hiperbólica transforma los valores introducidos a una escala (-1,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1. (Buduma, 2017)

$$f(x) = \frac{2}{1 - e^{-2x}} - 1$$

**Fórmula 2.8.2 Función tangente hiperbólica**

Características de la función tangente hiperbólica:

- Muy similar a la sigmoide
- Satura y mata el gradiente.
- Lenta convergencia.
- Centrada en 0.
- Esta acotada entre -1 y 1.
- Se utiliza para decidir entre uno opción y la contraria.
- Buen desempeño en redes recurrentes.

- **ReLU – *Rectified Lineal Unit***

La función ReLU transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran. (Buduma, 2017)

$$f(x) = \max(0, x) = \begin{cases} 0 \; para \; x < 0 \\ x \; para \; x \geq 0 \end{cases}$$

**Fórmula 2.8.3 Función ReLU**

Características de la función ReLU:

- Activación *Sparse* – solo se activa si son positivos.
- No está acotada.

- Se pueden morir demasiadas neuronas.
- Se comporta bien con imágenes.
- Buen desempeño en redes convolucionales.

### o **Leaky ReLU – *Rectified Lineal Unit***

La función Leaky ReLU transforma los valores introducidos multiplicando los negativos por un coeficiente rectificativo y dejando los positivos según entran.

$$f(x) = \begin{cases} 0\, para\, x < 0 \\ a \cdot x\, para\, x \geq 0 \end{cases}$$

**Fórmula 2.8.4 Función *Leaky ReLU***

Características de la función *Leaky ReLU:*

- Similar a la función ReLU.
- Penaliza los negativos mediante un coeficiente rectificador.
- No está acotada.
- Se comporta bien con imágenes.
- Buen desempeño en redes convolucionales.

### o **Softmax – *Rectified Lineal Unit***

La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

$$f(z) = \frac{e^{zj}}{\sum_{k=1}^{k} e^{zk}}$$

**Fórmula 2.8.5 Función Softmax**

Características de la función Softmax:

- Se utiliza cuando queremos tener una representación en forma de probabilidades.
- Esta acotada entre 0 y 1.
- Muy diferenciable.
- Se utiliza para para normalizar tipo multiclase.
- Buen rendimiento en las últimas capas.

La idea de esta operación es introducir la no linealidad en la red convolucional, ya que normalmente los datos que queremos que la red aprenda no son lineales. Existen otras funciones no lineales que también se pueden utilizar en lugar de esta, pero en la mayoría de las pruebas es la que mejor funciona.

### 2.9.3. Capa de reducción o *pooling*:

Se suele colocar después de la capa convolucional y reduce la dimensión de cada mapa de características (sin afectar la profundidad de este), manteniendo la información más relevante.

La capa de reducción puede ser de distintos tipos: *max-pooling, average-pooling, glblas Max-pooling, globalAverage-pooling, sum-pooling*.

La más utilizada es *max-pooling* ya que se ha demostrado que es la que mejor funciona. Este método divide la imagen de entrada en un conjunto de rectángulos d ellos cuales queda con el máximo valor de cada uno.

En el caso de *average-pooling* sería el mismo procedimiento, pero, en este caso, obteniendo la media de cada rectángulo.

### 2.9.4. Clasificación (*Fully Connected Layer*):

Antes de esta capa, se utiliza Flatten, que realiza un aplanamiento para convertir datos multidimensional en un vector de características 1D para poder ser utilizado por esta la capa clasificadora densamente conectada.

Se trata de una capa típica d ellos perceptrones multicapa que utiliza la función de activación softmax en la capa de salida.

Como el nombre indica, cada neurona de la capa anterior se encuentra conectada a cada neurona de la siguiente. El número de neuronas de la capa anterior será igual al número de pixeles, y el número de neuronas de la capa *Fully Conenected* tendrá el valor del número de las clases que se deben predecir.

La salida de las capas convolucionales y de reducción genera un mapa de características de alto nivel de la imagen de entrada, que permite a la capa *Fully Conenected* clasificarla en distintas clases tras entrenar la red asignando a cada clase una probabilidad.

La suma de todas las probabilidades es 1, hecho que se consigue al utilizar la función de activación sofmax.

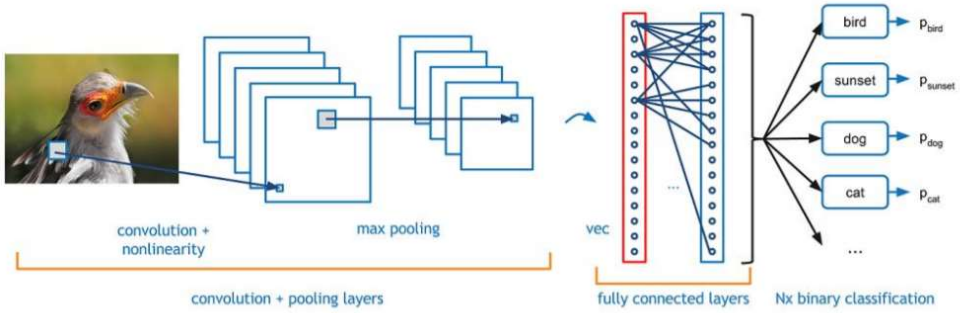


**Ilustración 2.8.2 Esquema general de redes neuronales. (Arias, 2019)**

## 2.10. Consideraciones al entrenamiento de la Red Convolucional.

Además de las distintas capas mencionadas anteriormente, en muchas ocasiones se suele utilizar un *Dropout.* Se trata de una técnica de regularización con la finalidad de evitar el sobreajuste de la red neuronal, que consiste en no considerar todas las conexiones entre neuronas y capas.

Para ello se seleccionan neuronas aleatoriamente en cada iteración, las cuales se ignorarán para el *forward propagation* y el *backward propagation*.

A medida que la red neuronal aprende, los pesos de las neuronas se van ajustando y especializando, por lo tanto, si se dejan de actualizar los pesos de las neuronas desactivadas, deberán intervenir para realizar las predicciones para las que faltan.

El resultado es que la red se vuelve menos sensible a los pesos específicos de las neuronas, se reduce la dependencia de las relaciones con las neuronas vecinas y es capaz de una mayor generalización haciendo menos probable la adaptación a los datos de entrenamiento.

El *Dropout* tiene un parámetro que oscila entra 0 a 1, siendo este la probabilidad de que las neuronas se queden activadas. Usualmente se suele emplear un valor de 0,5 que corresponde a una desactivación de la mitad de las neuronas.

Actualmente, está en auge el uso de las redes neuronales en el campo de la medicina sobre todo en el análisis de datos e imágenes médicas.

# 3. Herramientas

Este capítulo mostrará varias herramientas que existen para el desarrollo del trabajo. Incluyendo sus características, así como también el criterio utilizado para la selección de las herramientas utilizadas.

## 3.1. Lenguajes de Programación

### 3.1.1. R

El lenguaje R es una herramienta de programación, que se ha popularizado a raíz de ser utilizado por empresas del tamaño de Facebook, Google, Pfizer, Bank of America o Shell Oil para obtener el mayor valor de la información almacenada.

Con millones de usuarios, estas compañías podrían tardar semanas en poner todos sus datos en una gráfica para estudiarlos, y al recurrir a un lenguaje de programación de estadística reducen este tiempo a pocos minutos.

Las profesiones relacionadas con el Big Data se han convertido en una de las más prometedoras del mercado actual y la formación en las herramientas de analítica es una buena baza si quieres encontrar las mejores salidas profesionales.

R es un lenguaje de programación gratuito que opera bajo licencia GNU. Se trata de un entorno estadístico que se utiliza, principalmente, para la computación en este ámbito, ofreciendo una variedad de algoritmos que se usan mucho en el aprendizaje automático.

El lenguaje R también es un entorno compuesto por un conjunto de paquetes de *software* que se pueden usar para realizar cálculos numéricos y trazar gráficos para la manipulación de datos. Esto lo hace especialmente útil en la investigación estadística.

Este lenguaje se ejecuta en las plataformas UNIX, Windows, MacOS, FreeBSD y Linux y se caracteriza por tener una estructura compleja de datos, operadores y funcionalidades. De esta manera, te pueden ofrecer desde matrices hasta bucles y recursividad, junto con la integración de otros lenguajes de programación, como C, C ++ y Fortran.

### 3.1.2. Octave

Octave es un lenguaje de alto nivel para realizar cálculos numéricos en el ordenador, y también es un programa capaz de interpretar este lenguaje y realizar los cálculos. Octave ofrece una interfaz de usuario interactiva, orientada a línea de comandos, pero también puede ser utilizado en modo no interactivo, leyendo sus órdenes de fichero.

Octave originalmente fue desarrollado para facilitar la tarea a los estudiantes de Ingeniería Química de la universidad de Texas, sin que estos tuvieran que enfrentarse a las dificultades de la programación. Su flexibilidad en seguida lo hizo popular y su uso se expandió a otros problemas relacionados con el álgebra lineal y las ecuaciones diferenciales y favoreció su desarrollo, agregando las aportaciones de la comunidad de usuarios.

Otros programas de características similares, y hasta cierto punto compatibles, son son el lenguaje R de la FSF, Matlab y Scilab. Estos dos últimos propietarios. Octave es software libre (bajo licencia GNU), lo que significa que se puede usar y redistribuir libremente, y que cualquiera puede ayudar para mejorarlo.

### 3.1.3. Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Java es uno de los mejores lenguajes de programación después de Python para desarrollar soluciones de Inteligencia Artificial. Una cosa de la que deben darse cuenta es que no hay un lenguaje individual sólo para desarrollar soluciones de Inteligencia Artificial. A pesar de eso, las herramientas existentes ayudan a los desarrolladores a obtener grandes resultados al desarrollar soluciones de IA.

Entre sus principales características tenemos:

- Facilidad de depuración
- Facilidad de uso

- Simplificación de proyectos a gran escala
- Representación de datos en gráficos
- Mejor interacción con el usuario

La tecnología de máquina virtual de Java permite a los desarrolladores de Java crear una versión única de la aplicación que se activa al ejecutarla en las plataformas compatibles con Java. De este modo, añadirá valor a su negocio. La incorporación del Standard Widget Toolkit como Swing y SWT hará que las interfaces y los gráficos en Java se vean más atractivos y al mismo tiempo sofisticados.

### 3.1.4. Python

Python es un lenguaje de programación interpretado, es decir que requiere de programas adicionales para ser compilado y dar órdenes al computador; orientado a objetos (POO) es decir que emplea términos como: clases, objetos, propiedades y métodos para hacer más sencilla la escritura del código.

Es gratuito y de código abierto por lo que cualquier persona puede descargarlo sin costo y colaborar con su código.

Posee una sintaxis sencilla, a diferencia de lenguajes de programación como Java o C, escribir código en Python es muy similar a como nos expresamos de manera habitual, facilitando el aprendizaje y la escritura de código.

La gran ventaja de Python en el entorno de inteligencia artificial es que es un lenguaje gratuito por lo que diversas empresas y entornos relacionados con este sector pueden usarlo sin aumentar sus costes. Ha marcado un estándar importante dentro de este sector debido a la gran cantidad de librerías disponibles como TensorFlow y Keras.

La inteligencia artificial con Python es uno de los sectores mejor remunerados, pero de más elevada curva de aprendizaje, debido a las nociones matemáticas de la misma.

La forma correcta de comenzar es aprender las bases de este lenguaje de programación, las buenas prácticas del código y desarrollo de aplicaciones sencillas antes de abordar elementos complejos como formas de aprendizaje automático (supervisado y no supervisado) que este sector demanda.

Tras analizar los lenguajes existentes; para efectos del trabajo saltan a la luz ciertos aspectos que hacen la diferencia sustancial para el trabajo los cuales son:

- Lenguajes de fácil sintaxis con varias funciones en el ámbito de la inteligencia artificial.
- Ser código abierto, lo cual permite encontrar varios ejemplos y desarrollos gratis que pueden ser tomados y modificados.
- Facilidad de aplicación y de aplicación en distintos equipos.

Con estas características, se encuentra el R, pero no fue seleccionado ya que este lenguaje está más enfocado a trabajos de *big data* o desarrollos estadísticos. La otra opción fue Octave, la cual es ampliamente usada para desarrollo de aplicaciones, pero tampoco se eligió porque es necesario descargar la aplicación que, a pesar de ser descarga libre, es un lenguaje con sus diferencias, y con el ánimo de cumplir el último punto de aplicación es distintos equipos con mayor facilidad; se termina escogiendo Python como lenguaje para el trabajo.

Esto se debe a que Python, también es lenguaje abierto, con múltiples desarrollos y ha tenido especial uso en los campos de *machine Learning* y *Deep Learning*. Además de una sintaxis más sencilla y facilidad de comprensión.

## 3.2. Técnicas y Librerías

Para el desarrollo del tema del proyecto. Se trabajó con gran cantidad de librerías, desde librerías de imágenes, *machine Learning* e inclusive librerías en desarrollo para análisis de imágenes médicas. Por la complejidad del trabajo, fue necesario usar muchas de las funciones incluidas en gran cantidad de librerías. Por esta razón se mencionan las librerías utilizadas.

### 3.2.1. Análisis de imágenes y visión artificial

Por la naturaleza misma del trabajo, fue necesario manejar varias librerías que permitieran el manejo y tratamiento de las imágenes. Siendo escogidas aquellas que ofrecieran mayor confiabilidad y funciones que se adaptaran a las necesidades. Estas son:

- Pillow
- OpenCV
- Matplotlib
- PIL
- CV2

### 3.2.2. Transformación de imágenes, operaciones de reducción.

Para los procesos de transformación de las imágenes fue necesario utilizar librerías para la transformación matricial y también las que incluyen las operaciones de reducción dimensional. Las utilizadas son:

- **Numpy:** Tiene funciones que permiten transformar una imagen en matriz y realizar varias operaciones matriciales.
- **Scikit-learn:** es probablemente la librería más útil para *Machine Learning* en Python, es de código abierto y es reutilizable en varios contextos, fomentando el uso académico y comercial. Proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados en Python.

## 3.3. Aprendizaje Profundo

Se utilizaron las librerías:

- **Keras:** Es una biblioteca de código abierto (con licencia MIT) escrita en Python, que se basa principalmente en el trabajo de François Chollet, un desarrollador de Google, en el marco del proyecto ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). La primera versión de este software multiplataforma se lanzó el 28 de marzo de 2015. El objetivo de la biblioteca es

acelerar la creación de redes neuronales: para ello, Keras no funciona como un framework independiente, sino como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos.

- **Tensorflow:** Es una biblioteca de software de código abierto para computación numérica, que utiliza gráficos de flujo de datos. Los nodos en las gráficas representan operaciones matemáticas, mientras que los bordes de las gráficas representan las matrices de datos multidimensionales (tensores) comunicadas entre ellos. Es importante señalar que permite construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

### 3.4. Librerías evaluadas.

A continuación, se mencionan algunas librerías que fueron evaluadas, pero no se pudo lograr que corrieran correctamente en la computadora.

- **Pytorch:** es un framework de *Machine Learning* de código abierto creado para ser flexible y modular para la investigación, con la estabilidad y el soporte necesarios para el despliegue de producción. PyTorch proporciona un paquete de Python para funciones de alto nivel como el cálculo de tensor (como NumPy) con una fuerte aceleración de GPU y TorchScript para una transición fácil entre el modo «eager» y el modo gráfico. Con la última versión de PyTorch, el framework proporciona ejecución basada en gráficos, capacitación distribuida, implementación móvil y cuantización. La sencillez de su interfaz, y su capacidad para ejecutarse en GPUs (lo que acelera el entrenamiento de los modelos), lo convierten en la opción más asequible para crear redes neuronales artificiales.
- **Fastai:** Es una librería basda en PyTorch que permite simplificar operaciones al personal que tiene conocimientos previos en ML. Sin embargo, igual que el caso de Pytorch, al ser aplicaciones en desarrollo su instalación tiene muchos detalles a tener en cuenta lo cual es muy fácil equivocarse y para las aplicaciones desarrolladas en el presente trabajo, daba muchos problemas generando muchas veces problemas con el kernel.

Adicionalmente se evaluaron varias bibliotecas para análisis de imágenes médicas en desarrollo las cuales no fueron implementadas porque generaban muchos problemas en la instalación y el consumo de recursos era muy alto.

### 3.5. Aplicaciones desarrolladas en la red.

Parte del trabajo consistió en revisar varias aplicaciones disponibles existentes sobre varios programas desarrollados para clasificación de LA COVID-19 por imágenes. De estos programas se pudo observar lo siguiente:

- Todos manejaban bases de datos de imágenes, muy poca cantidad de instancias. Varias de ellas no daban acceso a la base de datos que usaban.
- Hay varios algoritmos desarrollados con fines comerciales que se puede observar los resultados mas no revisar su estructura. Solamente usarla y bajo parámetros bien restringidos. Siendo redes pre-entrenadas para clasificación de imágenes varias.
- Los programas basados en librerías médicas de imágenes daban muchos problemas a la hora de correr en la computadora, así como fastai y pytorch. Razón por la cual no se decidió utilizar estas librerías.
- Todas las redes observadas tenían criterios de trabajo diferentes. Por ejemplo, muchas redes hacían clasificación entre pacientes normales, con neumonía y LA COVID-19pero no mostraban matriz de confusión y la mayoría con una cantidad de imágenes para analizar y de prueba muy baja. Lo cual siempre le daban muy buenos valores de clasificación.

En consecuencia, de lo anterior y como el objetivo del trabajo era evaluar el efecto de la reducción de dimensionalidad en la clasificación. Se decidió desarrollar el programa para la detección según algoritmos seleccionados.

### 3.6. Entornos de Trabajo

Parte del trabajo consistió en revisar varias aplicaciones disponibles existentes sobre varios programas desarrollados para clasificación de LA COVID-19por imágenes. De estos programas se pudo observar lo siguiente:

- **Jupyter Notebook:** Es un entorno de trabajo interactivo que permite desarrollar código en Python de manera dinámica, a la vez que integrar en un mismo documento tanto bloques de código como texto, gráficas o imágenes.

- **Anaconda Python:** Anaconda es una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python. En líneas generales Anaconda Distribution es una distribucción de Python que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto. Con Anaconda, se tiene acceso a Python, R y Jupyter Notebook.

- **Google Colaboratory:** Es un entorno de la plataforma Google que permite escribir y ejecutar códigos de Python en cualquier navegador. Se puede crear y desarrollar cualquier script, así como también subir bases de datos y correrlo. Tiene una versión gratuita con una capacidad definida de GPU y memoria disponible.

Google colab tiene la facilidad que está en la nube y se puede tener acceso desde cualquier ordenador. La versión gratis no permite guardar la base de datos utilizada siendo necesaria cargarla cada vez que se realizan operaciones. Para cargar la base de datos de imágenes utilizada se necesitaban más de 4 horas en cada ocasión, lo cual resultó ser totalmente impráctico. Sin embargo, es una excelente opción para tomar en cuenta por el uso de los recursos disponibles en línea y la facilidad de acceso desde cualquier máquina para el usuario con mayor capacidad de opciones.

- **Kaggle:** Es una popular plataforma con excelentes recursos para aquellos que quieran aprender *Machine Learning* e inclusive ciencia de los datos. Cuenta con varias competiciones de *Machine Learning* que tienen más de 1 millón de dólares de premios y cientos de competidores. Los mejores equipos cuentan

con décadas de experiencia combinada, abordando problemas ambiciosos como la mejora de la seguridad aeroportuaria o el análisis de datos satélites.

Kaggle ofrece una gran cantidad de scripts enfocado en la inteligencia artificial, se puede correr desde cualquier navegador, pero al igual que colab funciona con recursos limitados y también su programación es diferente del Python convencional lo cual es necesario hacer varias modificaciones para correr en esta plataforma.

Finalmente se escogió Jupyter ya que permite hacer aplicaciones y guardarlas en cuadernos de aplicación que tiene mayor libertad de adaptación que la plataforma tradicional de Python.

# 4. Conjuntos de imágenes para la identificación de covid-19.

## 4.1. Consideraciones Iniciales.

Uno de los principales aspectos a considerar en este trabajo fue la selección adecuada de las imágenes a ser utilizadas. Principalmente porque en la gran mayoría de trabajos encontrados y de imágenes disponibles casi todos eran de distintos formatos, distintas resoluciones, algunas a colores y otras no. E inclusive se encontraron algunos algoritmos que realizaban clasificación de imágenes de cortes axiales de resonancias y otros con radiografías.

Por la novedad de la enfermedad y la naturaleza de esta, trae como consecuencia dificultad en conseguir una base de datos con una cantidad de instancias considerable. Las que existen de mayor tamaño son base de datos privadas de personas que están desarrollando su algoritmo de forma comercial. Hay iniciativas como el caso de la UBM que está generando una base de datos de imágenes de radiografías para estudios médicos sin embargo es un trabajo en desarrollo y se requiere de varios trámites para conseguir acceso a las imágenes.

El otro punto importante es que la enfermedad está catalogada como una neumonía atípica. Lo que quiere decir es que es una neumonía de un desarrollo acelerado. En consecuencia, si se analiza la imagen se va a diagnosticar una neumonía. Por esta razón, la prueba valida hasta el momento es la de PCR (análisis de la mucosa) para clasificar casos de enfermedad. Por análisis de imágenes, mientras no se tenga un banco de imágenes de mayor tamaño, se realiza la clasificación de pacientes sanos y pacientes con neumonía. Por la gravedad de la situación actual, al detectar un paciente con neumonía, se puede activar otro procedimiento de investigación de cada caso. Con esto se podría igual hacer un mejor uso de los recursos médicos ya que revisarían en detalle un número menor de pacientes.

Es importante señalar que, debido a la información anterior, las bases de datos médicas de mayor confiabilidad solamente hacen clasificación binaria de los pacientes como sanos y enfermos.

Adicionalmente, había radiografías con trazos de colores y otras no, así como imágenes modificadas previamente de ser guardadas con diferencias en resolución (por mencionar una modificación). Y hay casos donde la base de datos utilizada para trabajar

algunos algoritmos no está disponible para su uso sino solamente para ese algoritmo. Siendo en varios casos no posible descargar.

También se encontraron algoritmos que ofrecen su base de datos que van desde 10 imágenes hasta 1000 imágenes. Parte de esto se debe a que varios de estos algoritmos se modificaron para correr en Google colab y es lento la carga de base de datos de gran tamaño.

En vista de todo lo anterior se tomaron las siguientes definiciones para el desarrollo del trabajo:

- Se va a realizar el proceso de clasificación en imágenes de radiografías de tórax: Esto obedece a un mejor uso de los recursos computacionales ya que en este caso solo se analiza una imagen. En el caso de clasificación de imágenes por cortes axiales se requiere procesar un mínimo de 50 imágenes (pueden ser muchas más, según la cantidad de imágenes tomadas por corte).
- Se utilizará una base de datos con la mayor cantidad posible de imágenes pero que tengan coherencia en tipo y forma. Es decir; que todas las imágenes sean iguales en tipo, color, forma.
- Se utiliza base de datos que sea de una fuente que tenga veracidad.
- Si se cuenta con diferentes bases de datos, se tomará aquella que cumpla con las condiciones anteriores y tenga la mayor resolución de imagen.

Existen proyectos de creación de bases de datos de radiografías los cuales están en desarrollos y ofrecen aspectos interesantes como por ejemplo ieee8023/covid-chestxray-dataset o el Banco de imágenes médicas de Valencia, donde se han encontrado aplicaciones para extraer imágenes de manera aleatoria de la base de datos para entrenamiento y prueba.

Esta base de datos ya tiene un más de un terabyte de información, lo cual incluye imágenes y un reporte del diagnóstico médico. Es un poco complicada descargar la información completa por el tamaño de la base de datos y el tipo de archivo que son pesados, además de tener restricciones para su descarga. En GitHub permite descargar un script donde se puede ver la cantidad de imágenes y su distribución.

Para el momento de la investigación fue el repositorio que cumplía con las condiciones previas planteadas anteriormente y ofrecía mayor cantidad de información fue el repositorio encontrado en la ´página de mendeley Data, titulado: “Tomografía de coherencia óptica (OCT) etiquetada e imágenes de rayos X de tórax para clasificación”.

El conjunto de datos utilizado corresponde a imágenes de rayos X de tórax y OCT validadas descritas y analizadas en "Clasificación y derivación de enfermedades humanas tratables basadas en el aprendizaje profundo". Las imágenes OCT se dividen en un conjunto de entrenamiento y un conjunto de pruebas de pacientes independientes. Las imágenes de OCT están etiquetadas como (enfermedad) - (ID de paciente aleatorizado) - (número de imagen de este paciente) y se dividen en 4 directorios: CNV, DME, DRUSEN y NORMAL. Tiene un 1 Gb de imágenes.  Todas en formato jpeg.

Esta información es validad por la Universidad de California. Publicada el 6 de enero del 2018. Contribución de: Daniel Kermany, Kang Zhang,Michael Goldbaum.

Mendeley Data, es un repositorio ideado para que los investigadores, de forma libre, gratuita y en línea, suban los datos de sus investigaciones, estos queden almacenados de forma segura en su servidor y puedan compartirlos con otros investigadores. Los conjuntos de datos, bajo licencias abiertas, se pueden compartir de forma privada con determinados colegas, o bien darlos a conocer al mundo entero, haciendo posible la reutilización de la ciencia y la multiplicación exponencial de estudios e investigaciones.

Las características de este conjunto de datos son:

- Todas las imágenes están en un único formato que es jpeg
- Tiene clasificación binaria (Normal, *Pneumonia*).
- Si dirección es: https://data.mendeley.com/datasets/rscbjbr9sj/2

Es importante señalar que otra de las bases de datos que se tuvo en consideración fue la encontrada en la dirección https://github.com/ari-dasci/OD-covidgr el cual es un repositorio en GitHub del instituto Andaluz de investigación en ciencia de datos e inteligencia computacional. La razón que impulsó a usar la base de datos del instituto Mendeley fue que poseía mayor cantidad de instancias.

## 4.2. Investigación de trabajos previos:

Parte del trabajo previo fue realizar la investigación de trabajos anteriores de clasificación de imágenes en el ámbito médico que permitan ser referencia al trabajo actual. Finalmente, no se utilizó ninguno, sin embargo, se comentan.

De todos los programas encontrados en repositorios de código abierto que se encontraron en la red; a continuación, se muestran aquellos con mejores características en su estudio. Sea por su estructuración, información, etc.

- **Automated Detection of COVID-19 Cases Using Deep Neural Networks with Xray Images**

Estudio donde se desarrolla una red neuronal para tratar de detectar la COVID-19. Se puede encontrar en el repositorio https://github.com/muhammedtalo/COVID-19. A continuación, se muestran los comentarios de este.

- Se usó red variante de DarkNet, se utilizó dos tipos de clasificación:
  - *Pneumonia, Covid y no-findings.*
  - *Covid y no-findings*
- Obviamente el segundo tipo de clasificación tuvo resultados mejores, ya que no hay que olvidar que el covid es una clasificación de la neumonía.
- Uno de los grandes problemas es la poca cantidad de imágenes disponibles para el estudio y la baja resolución de varias de ellas.
- Para un diagnóstico eficaz del Covid es necesario estudios en las etapas anteriores y no hay suficiente documentación al respecto ya que se tiene en la mayoría de los casos son casos ya corroborados y avanzados.
- Ambos scripts están en jupyter y están bien ejemplificados, obviamente dan buenos resultados, pero eso está influido por la poca cantidad de datos que hay

- **COVID-Net *Open Source Initiative***

Este es un algoritmo abierto para análisis de COVID, tiene base de datos que se actualiza con el tiempo. A continuación, se muestran sus valores de rendimiento.

| Params (M) | MACs (G) | Accuracy (%) |
|---|---|---|
| 117.4 | 2.26 | 92.6 |

Table 2. Sensitivity for each infection type.

| Sensitivity (%) | | |
|---|---|---|
| Normal | Non-COVID19 | COVID-19 |
| 97.0 | 90.0 | 87.1 |

Table 3. Positive predictive value (PPV) for each infection type.

| Positive Predictive Value (%) | | |
|---|---|---|
| Normal | Non-COVID19 | COVID-19 |
| 89.8 | 94.7 | 96.4 |

**Ilustración 4.2.1 Resultados de las ejecuciones del algoritmo (Wangg, 2021)**

Este tal vez el algoritmo código abierto que he encontrado más referencia al respecto. Se trata un algoritmo CNN del cual se han derivado diferentes versiones y tiene gran cantidad de referencias. Obtiene buenos resultados, se remite a una base de datos en línea con lo cual debería actualizarse en el tiempo. La BD se llama Covidx y cuenta con 16756 radiografías de 13645 pacientes.

Para la evaluación utilizaron un modelo GSInquiere y la red fue pre entrenada con Image.Net

Los problemas que se encuentran son en la cantidad de datos para entrenamiento y los creadores comentan que la metodología de entrenamiento podría ser mejorada.

Pero al ser un tipo de aprendizaje supervisado, se considera que podría modificarse el porcentaje de entrenamiento y error para una mejor selección.

También al ser una red pre entrenada, ofrece muy pocas posibilidades de poder evaluar su rendimiento con algoritmos de reducción de dimensionalidad. Su enlace es: https://github.com/lindawangg/COVID-Net

- **COVID-19-Detection-Flask-App-based-on-Chest-X-rays-and-CT-Scans**

    - La limitación principal encontrada es en el tema de la cantidad de imágenes de entramiento ya que ellos utilizaron solamente 1000 imágenes de radiografías de tórax y 750 imágenes de scanner.
    - Se trabajó con 4 algoritmos: VGG16, ResNet50, InceptionV3 and Xception fueron entrenados por separado las radiografías y los CT Scans, obteniendo untotal de 8 modelos de aprendizaje profundo. Modelo bien interesante ya que compara 4 tipos de algoritmos diferentes para obtener un total de 8 modelos de aprendizaje. La proporción entrenamiento/prueba es de 80/20.
    - Resultados con altos valores de acierto, pero evidentemente se ve afectado por la baja cantidad de imágenes para la evaluación.

Su enlace es: https://github.com/kaushikjadhav01/COVID-19-Detection-Flask-App-based-on-Chest-X-rays-and-CT-Scans

- **velebit-ai/COVID-Next-Pytorch**

Basado en el modelo COVID-NET pero usando pytorch. Este modelo utiliza la biblioteca Pytorch, y el principal problema detectado es el tiempo de carga de las imágenes en la memoria ya que usualmente las imágenes son en alta resolución. Razón por la cual los creadores del modelo recomiendan reducir el tamaño de la imagen.

También indican que se puede modificar el número de subprocesos pero advierten que esto afectará el uso de la memoria.

Utiliza también la librería pydicom de análisis de imágenes médicas, se recomienda instalar en un entorno virtual y es bastante engorroso de tratar de correr.

| | Accuracy | F1 Macro | Precision Macro | Recall Macro |
|---|---|---|---|---|
| COVID-Net (Large) | 91.90% | 91.39% | 91.4% | 91.33% |
| **COVID-Next** | 94.76% | 92.98% | 96.40% | 90.33% |

Tabla 4.1. Resultados actualizados el 20 de marzo 2020.

Su enlace es: velebit-ai/COVID-Next-Pytorch

- **MuhammadMiqdadKhan/COVID_19_Detection_with_Chest_X_Ray_Images**

Utiliza Keras, es un algoritmo básico pero una base de datos bastante corta que comparte desde Google drive.

Una aplicación bastante sencilla en comparación con las anteriores y el script que colocan tiene algunos detalles ya que básicamente es un recuento de como corrió el programa con lo cual hay algunas diferencias cuando se trata de aplicar.

- Su enlaces es: https://github.com/MuhammadMiqdadKhan/COVID_19_Detection_with_Chest_X_Ray_Images/blob/master/COVID_19_Prediction_with_Chest_X_Ray_Images.ipynb

- **includeamin/COVID-19**

Modelo con Keras que solo utiliza 10 imágenes, la forma como esta hecha el programa es un poco mas rudimentaria. No utiliza Jupyter sino todo directamente en Python.

Al tener un número tan reducido de imágenes es difícil determinar su efectividad, tan solo es el algoritmo, no tiene algún método de comprobación como matriz de confusión sino solamente gráfica precisión y exactitud.

Su enlace es: https://github.com/includeamin/COVID-19/blob/master/README.md

- **rabia174/COVID-19-Deep-Learning-CNN-Model :**

Tiene bastante precisión, pero se indica que puede modificarse la estructura para una mejor detección. También el número de imágenes no ayuda (sólo 130 imágenes).

Usa SimpleItk que es un paquete desarrollado y es medio experimental, es medio complicado para instalar.

Su enlace es: rabia174/COVID-19-Deep-Learning-CNN-Model

- **DengPingFan/Inf-Net**

Hace análisis de imágenes en CT. Dataset se puede descargar en Google drive tiene 48 imágenes para entrenamiento.

Utiliza, varias redes convolucionales que ya están preentrenadas y deben descargarse para su uso, igual que detection Flask.

Tiene un *toolbox* de evaluación en Matlab, con lo cual debe descargarse y evaluarse en este programa adicionalmente. En el vínculo https://arxiv.org/pdf/2004.14133.pdf está el artículo.

# 5. Técnicas de reducción desarrolladas. Experimentación y resultados.

Este capítulo está dedicado a mostrar el desarrollo del trabajo, las técnicas de reducción utilizadas y los parámetros los cuales se configuraron. Así como los cuadernos de clasificación desarrollados.

El objetivo del trabajo es ver cómo afecta la reducción de dimensionalidad en el procesamiento de imágenes para el caso de determinación de la COVID-19 por análisis de imágenes de radiografía de tórax, lo cual implica dividir el desarrollo en dos grandes grupos:

- Procesamiento de las imágenes con técnicas de reducción
- Algoritmo de clasificación.

## 5.1. Técnicas de reducción de dimensionalidad.

A menudo puede parecer que un algoritmo de aprendizaje automático funcionará mejor cuando disponga de mucha información en cada instancia de entrenamiento, sin embargo, esto en la práctica no funciona así. Los algoritmos empeoran su rendimiento a medida que aumentamos la dimensión de los datos.

En la siguiente ilustración se muestra cómo evoluciona de forma general la eficiencia de los algoritmos de aprendizaje automático con el aumento de las dimensiones. Se observa que se alcanza un máximo en la eficiencia con un número concreto de dimensiones y a medida que aumentamos las dimensiones el rendimiento empieza a decaer.

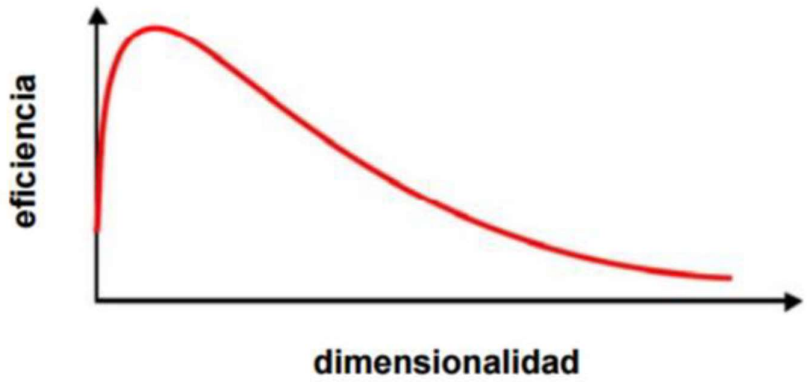


**Ilustración 5.1.1 Evolución de la eficiencia con el aumento de las dimensiones (Maganto, 2017)**

Por tanto, el principal motivo por el que se debe realizar una reducción de la dimensionalidad es para reducir el efecto conocido como maldición de la dimensionalidad (en inglés, *curse of dimensionality).* Este fenómeno fue descubierto por Richard E. Bellman [Bellman, 1961; Bellman y Corporation, 1957] cuando trabajaba con problemas de optimización dinámica. Consiste en que a medida que aumentamos el número de dimensiones, el volumen del espacio crece exponencialmente y provocamos que los datos estén más dispersos, lo que produce que la cantidad de datos de entrenamiento necesarios para extraer resultados fiables crezca también exponencialmente. (Bellman Richard Ernest, CORPORATION, Rand, 1957)

La maldición de la dimensionalidad puede ser interpretada visualmente como un espacio casi vacío donde es complicado encontrar patrones en los datos de ejemplo para tomar decisiones. Esta situación provoca que los modelos obtenidos se ajusten a particularidades de los datos de entrenamiento.

Adicionalmente, existe la limitación de los recursos del equipo usado para el procesamiento de los programas desarrollados. Su capacidad muchas veces limita desarrollar programas de mayor complejidad y a su vez incrementa demasiado el tiempo de ejecución.

En consecuencia, se evaluaron los distintos tipos de algoritmos de reducción de dimensionalidad que existen para determinar cuáles utilizar en función del tiempo y uso de recursos para su aplicación.

De esta evaluación se definieron los algoritmos de Análisis de componentes principales y la técnica clásica de Resize como los más adecuados para este trabajo. A continuación, se explican brevemente cada uno de ellos.

### 5.1.1. Análisis de Componentes Principales (PCA)

*Principal Component Analysis* (PCA) es la técnica estadística más tradicional y conocida que existe, fue propuesta por Karl Pearson [Pearson, 1901] y más tarde desarrollada por Harold Hotelling [Hotelling, 1933]. Esta técnica se trata de un método de reducción de la dimensionalidad mediante extracción de características. (Géron, 2019)

El método consiste en la transformación de las variables de entrada, probablemente relacionadas, a otro conjunto de variables llamadas componentes principales que son

independientes. Estas nuevas componentes, ortogonales y linealmente independientes entre sí, se obtienen mediante la combinación lineal de las variables originales.

$$
\begin{aligned}
Y_1 &= a_{11}X_1 + a_{12}X_2 + \cdots + a_{1d}X_d \\
Y_2 &= a_{21}X_1 + a_{22}X_2 + \cdots + a_{2d}X_d \\
&\vdots \quad \vdots \quad \vdots \quad \vdots \\
Y_d &= a_{d1}X_1 + a_{d2}X_2 + \cdots + a_{dd}X_d
\end{aligned}
$$

**Fórmula 5.1.1 Transformación lineal realizada por PCA**

Donde:

- $Yd$ es el *d-ésimo* componente principal.
- $Xd$ es el *d-ésimo* componente original.

Desde el punto de vista geométrico, PCA puede ser visto como una rotación de los ejes originales sobre sus medias.

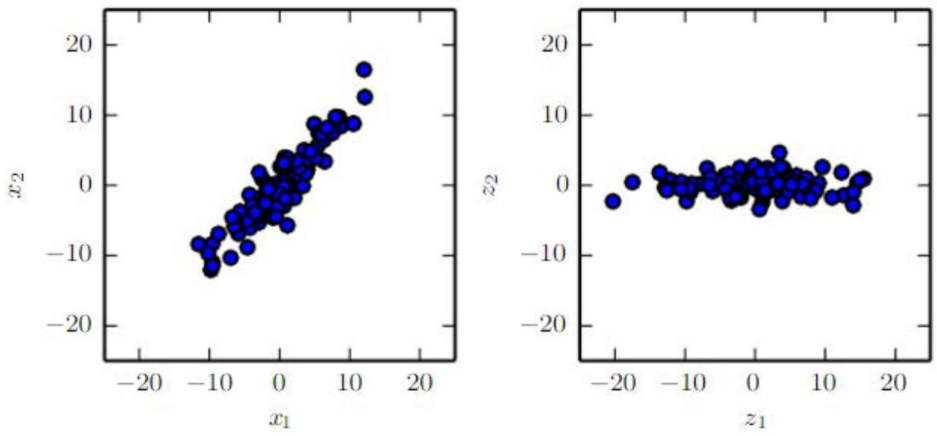


**Ilustración 5.1.2 Rotación de los datos. (Maganto, 2017)**

El primer componente principal es definido por el método donde el conjunto de datos presenta una mayor varianza, el segundo componente principal es definido donde los datos

presentan la segunda mayor varianza, y así sucesivamente. Hay que tener en cuenta que el número máximo de componentes principales que se obtienen mediante PCA siempre será igual al número de variables del conjunto de datos, aunque como el objetivo es reducir la dimensionalidad habrá que reducir el número de componentes seleccionando aquellos que expliquen la mayor cantidad de variabilidad de los datos.

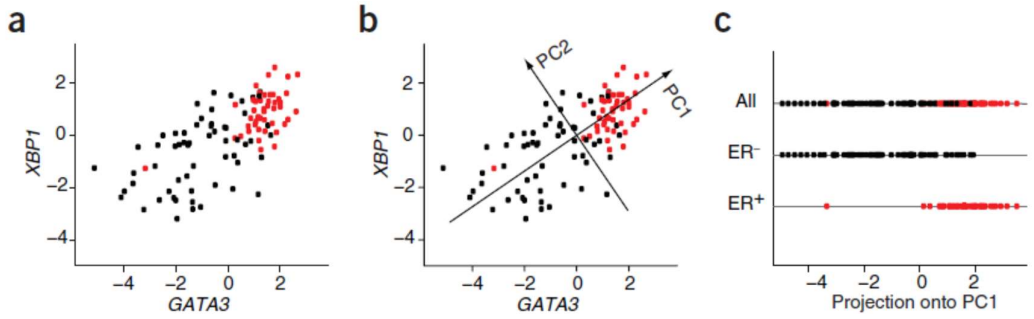


**Ilustración 5.1.3 Ejemplo de funcionamiento de PCA. (Maganto, 2017)**

En la Ilustración anterior podemos visualizar en forma gráfica los datos están etiquetados con dos clases distintas (ER+ y ER-), podemos visualizarlo en la Ilustración (a) como puntos rojos (ER+) y negros (ER-). En la Ilustración (b) vemos los componentes principales (PC1 y PC2) sobre el conjunto de datos. En la Ilustración (c) se muestran los datos sobre una única dimensión, es decir, se ha eliminado la segunda componente (PC2); se puede apreciar que se ha perdido información quitando un componente, pero es posible extraer conclusiones visualizando como los datos de las dos clases están bastante separados, es decir, en extremos opuestos de la gráfica. Nótese que si los datos son proyectados sobre el PC2 las conclusiones que se podrían obtener serían engañosas, ya que veríamos los datos de las clases juntos.

Por tanto, para este método es importante ordenar las componentes principales de acuerdo a la varianza de los datos, siendo el componente principal más importante el que más variabilidad explica de los datos y el componente menos importante el que menos varianza tiene. De acuerdo a esta reflexión, para reducir la dimensionalidad de los datos, eliminaremos las variables que menos cantidad de variabilidad explican.

Esta técnica presenta como principal ventaja que retiene las características del conjunto de datos que contribuyen más a su varianza. Sin embargo, tiene el inconveniente

que no ofrece resultados satisfactorios en muchos problemas supervisados de clasificación o regresión al tratarse de una técnica no supervisada.

Hay que tener en cuenta que existen distintas formas de realizar el cálculo de las componentes principales. En este caso se va a explicar detalladamente el proceso más conocido que utiliza la descomposición ortogonal de la matriz de covarianzas. Sin embargo, es posible utilizar la descomposición de valores singulares (SVD) de la matriz de datos y, en la actualidad, es el método más eficiente para el cálculo de PCA.

#### 5.1.1.1. Cálculo de los componentes principales

La covarianza entre dos variables mide el nivel de relación que existe entre ellas. La fórmula para calcular esta medida es la siguiente:

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y - \overline{Y})}{(n-1)}$$

**Fórmula 5.1.2 Fórmula de Covarianza**

La matriz de covarianzas recoge las covarianzas entre cada par de variables de un determinado conjunto de datos. Esta matriz siempre es simétrica, ya que se cumple que $cov(X,Y)=cov(Y,X)$. Además, la diagonal principal contiene las varianzas de cada variable porque se cumple que $cov(X,X)=var(X)$.

Por tanto, para construir la matriz de covarianzas de un conjunto de datos con *p* variables es necesario calcular la covarianza entre cada par de variables y ordenarlas de la siguiente forma:

$$S = \begin{pmatrix} cov(X,X) & \cdots & cov(X,P) \\ \vdots & \ddots & \vdots \\ cov(P,X) & \cdots & cov(P,P) \end{pmatrix}$$

**Fórmula 5.1.3 Matriz de Covarianzas**

La matriz de covarianzas se puede construir de forma equivalente a la anterior realizando el producto matricial que se puede ver en la ecuación (5.1.4). Es importante

destacar que la matriz *B* es la matriz de datos centrados en media 0, esto se consigue restando la media de cada dimensión a toda la columna de datos.

$$S = \frac{1}{n-1} BB^T$$

**Fórmula 5.1.4 Producto Matricial**

Por tanto, el método PCA se basa en la diagonalización ortogonal de una matriz *S* que corresponde con la matriz de covarianza del conjunto de datos. La diagonalización ortogonal consiste en encontrar una matriz diagonal *L* semejante a *S*, de la siguiente forma:

$$U^T SU = L$$

**Fórmula 5.1.5 Diagonalización ortogonal**

Donde los vectores columna de la matriz *U* son los autovectores ortonormales de la matriz *S*, mientras que los elementos de la diagonal de la matriz *L* son los autovalores de la matriz *S.* Además, como se demuestra en [Hernando, 2013], se cumple que toda matriz simétrica es ortogonalmente diagonalizable y, por tanto, la matriz de covarianzas al ser simétrica siempre será diagonalizable ortogonalmente.

El primer paso del método para calcular los componentes principales consiste en obtener la matriz de covarianzas de los datos, utilizando la ecuación (3.4) o la (3.5).

Una vez calculada la matriz, es necesario su descomposición en autovalores (eigenvalor) y autovectores (eigenvector). Para realizar esto, es necesario resolver la ecuación característica:

$$|S - \lambda I| = 0$$

**Fórmula 5.1.6 Ecuación Característica**

Donde:

- *S* es la matriz de covarianza.

- *I* es la matriz identidad.
- $\lambda$ son los autovalores.

Cuando se desarrolla la ecuación característica se obtiene un polinomio en función de $\lambda$ que es conocido como polinomio característico. Resolviendo esta ecuación obtenemos los distintos autovalores $\lambda$. Estos valores indican la cantidad de información que tiene cada componente, cuanto mayor sea el autovalor $\lambda$ significará que más cantidad de varianza explica su autovector y, por ende, más importante será.

Para calcular los autovectores es necesario resolver, para cada autovalor obtenido, el sistema matricial: $S-\lambda I=0$.

Para toda matriz simétrica se cumple que los autovectores asociados a autovalores distintos son ortogonales entre sí. Teniendo en cuenta esto, como la matriz *S* (matriz de covarianzas) es simétrica, los autovectores asociados a distintos autovalores obtenidos serán ortogonales. Sin embargo, los autovectores asociados al mismo autovalor pueden no ser ortogonales, en este caso será necesario ortogonalizarlos mediante algún método.

Como se comentó anteriormente, la matriz *U*, que contiene los autovectores, debe ser ortogonal, esto quiere decir que sus vectores columna deben ser ortonormales. Por ello, se deben normalizar los autovectores $\vec{v}$. Para realizar esto, se divide cada autovector por su módulo:

$$\vec{u} = \frac{\vec{v}}{|\vec{v}|}$$

**Fórmula 5.1.7 División Autovector**

Es importante destacar que los ejes de coordenadas de los componentes principales están definidos por los autovectores.

#### 5.1.1.2. Transformación de los datos originales

Una vez que tenemos los autovectores, estos deben ser ordenados por sus correspondientes autovalores asociados, de mayor a menor. Este orden da los componentes por orden de significancia y es posible formar una matriz *U*:

$$U=(eig1\ eig2\ldots\ eign)$$

**Fórmula 5.1.8 Matriz U**

Donde $eig1$ tiene el mayor autovalor y corresponde con el primer componente (PC1), $eig_2$ tiene el segundo mayor autovalor y corresponde con el segundo componente (PC2), y así sucesivamente.

Para reducir la dimensionalidad de los datos, es posible construir la matriz *U*, eligiendo los mejores *k* autovectores del conjunto total de *n* autovectores calculados, siendo $n > k$, perdiendo así la mínima cantidad de información posible en el nuevo conjunto de datos.

$$U=(eig1\ eig2\ldots\ eigk)$$

**Fórmula 5.1.9 Nuevo conjunto de datos U**

Existen diferentes criterios para elegir con cuántas dimensiones quedarse en cada caso. Uno de los más conocidos consiste en elegir los atributos que expliquen al menos un 75% de la varianza de los datos. Otro criterio consiste en representar gráficamente los autovalores en un gráfico denominado scree plot y elegir los atributos donde la pendiente cambia más bruscamente.

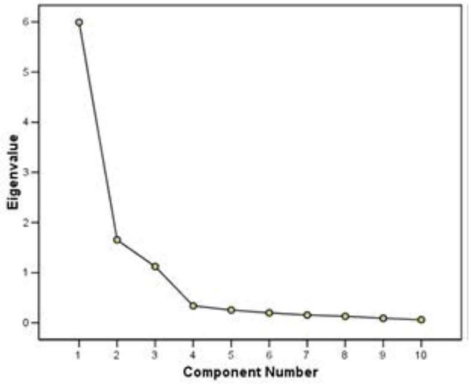


**Ilustración 5.1.4 Scree plot (Maganto, 2017)**

Por último, se aplica la transformación lineal de la ecuación (2.10) a cada uno de los datos.

$$\vec{y}=U[\vec{x}-\overline{x}]$$

**Fórmula 5.1.10 Transformación Lineal**

En este caso el sistema de ecuaciones lineales del anterior apartado (Ilustración 2.5: Transformación lineal realizada por PCA) se ha expresado en forma matricial, pero el resultado es el mismo. Además, es importante destacar que, como se comentó anteriormente, cada dato debe estar centrado con media cero y no sobre los datos originales, por este motivo se realiza la resta $[\vec{x}-\overline{x}]$ antes de realizar la multiplicación con la matriz U.

Cada dato individual transformado es conocido comúnmente con el nombre de *score*, mientras que los coeficientes de la matriz *U* son ampliamente conocidos con el nombre de *loading*.

### *5.1.2.* Técnica de reducción de dimensionalidad *Resize*

Esta función pertenece a la librería Numpy. Esta función como tal permite redimensionar la matriz de valores de la imagen a un tamaño predefinido. De esta manera se puede reducir el tamaño de la matriz de datos de la imagen a valores que decida el usuario.

Una de sus principales características es que, si el nuevo arreglo a generar es mayor que el arreglo original, el nuevo arreglo es llenado con copias repetidas del primer arreglo. Este comportamiento es diferente de la función arreglo.resize(new_shape) que rellena con ceros en vez de los valores del arreglo original. (Numpy.org, s.f.)

Es importante señalar que, esta funcionalidad no es adecuada para cambiar el tamaño de imágenes o datos en los que cada eje representa una entidad separada y distinta.

Para la reducción de tamaño se utilizó la función "*Resize*" de Python, y se incluyó en los scripts directamente.

### 5.1.3. Otras Técnicas de reducción de dimensionalidad.

Anteriormente fueron definidas las técnicas a utilizar en el presente trabajo. Sin embargo, existen otras técnicas las cuales se mencionan a continuación:

- **Factorización no negativa de matrices:** El NMF se diferencia de la mayoría de las técnicas de descomposición porque impone restricciones adicionales a una forma de matriz; Todas las entradas de la matriz deben ser no negativas, lo que conduce a una representación de datos que son más fáciles de inspeccionar.
- **Análisis de componentes independientes:** El objetivo fundamental del Análisis de Componentes Independientes (*ICA* en inglés) es el de proporcionar un método que permita encontrar una representación lineal de los datos no gaussianos de forma que las componentes sean estadísticamente independientes o lo más independiente posible. Una representación de este tipo permite obtener la estructura fundamental de los datos en muchas aplicaciones, incluidas la extracción de características y la separación de señales.

## 5.2. Procesamiento de las imágenes con técnicas de reducción

## 5.3. Scripts Desarrollados

Para los algoritmos se realizaron ciertas comparaciones para un mejor procesamiento, tales como:

- Se realizó clasificación binaria de la enfermedad ya que la base de datos que se dispone tiene radiografías de pacientes sanos y pacientes enfermos. Esto básicamente obedece a la clasificación de La COVID-19 el cual se considera una “neumonía atípica”.
- Todos los procesos de reducción de matrices se consideran de 150 x 150 ya que es el número que se encontró adecuado para realizar los cálculos.

- Se realizaron programas de clasificación de imágenes con algoritmos convolucionales y con *Support Vector Machine*.

Para el caso en especial de las redes convolucionales, se decidió diseñar la misma con una topología sencilla. Ya que el procesamiento de las imágenes requiere un buen uso de recursos de la computadora y también se buscaba evaluar la capacidad de clasificación de imágenes con un arreglo que no fuera tan exigente a nivel de recursos. Los parámetros que funcionaron bien para el trabajo obtenidos se muestran a continuación:

- Epochs=10
- longitud, altura = 150, 150
- batch_size = 32
- pasos = 1000
- validation_steps = 300
- filtrosConv1 = 32
- filtrosConv2 = 64
- tamano_filtro1 = (3, 3)
- tamano_filtro2 = (2, 2)
- tamano_pool = (2, 2)
- clases = 2
- lr = 0.0004

Fue particularmente útil la función ImageDataGenerator para la formación de los conjuntos de datos siendo la técnica usada en su mayoría para clasificación de imágenes con redes convolucionales. Los parámetros usados para el ImageDataGenerator son:

```
test_datagen = ImageDataGenerator(rescale=1. / 255)

entrenamiento_generador = entrenamiento_datagen.flow_from_directory(
    data_entrenamiento,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')

validacion_generador = test_datagen.flow_from_directory(
    data_validacion,
```

```
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')

cnn = Sequential()
cnn.add(Convolution2D(filtrosConv1, tamano_filtro1, padding ="same",
input_shape=(longitud, altura, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Convolution2D(filtrosConv2, tamano_filtro2, padding ="same"))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Flatten())
cnn.add(Dense(256, activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(clases, activation='softmax'))

cnn.compile(loss='categorical_crossentropy',
        optimizer=optimizers.Adam(lr=lr),
        metrics=['accuracy'])
```

Para definir los parámetros de la red, fue a través de ensayo y error con varias configuraciones donde se evaluaba el tiempo de procesamiento vs la precisión de los resultados quedando los valores mostrados anteriormente.

Tanto para el estudio de las técnicas de reducción, como para la clasificación de las imágenes, se desarrollaron varios cuadernos en Jupyter Notebook. Los cuales se explican a continuación sus características:

- **Pca_individual:** Cuaderno que permite visualizar la varianza acumulada según el número de componentes elegidos para hacer la reducción de dimensionalidad con PCA.
- **Transf_PCA:** Desarrollado para realizar la transformación de las imágenes de la base de datos con la técnica PCA y guardarlas con un nuevo nombre en otra carpeta de destino. Esta diseñado para guardar las imágenes colocando un prefijo “NXXX-“antes de cada nombre, las XXX corresponden a los números de componente que van

a conservar para las varianzas. Este script determina el tiempo de ejecución en cada caso. Como algunas imágenes fueron procesadas anteriormente y tenían arreglos internos de diferentes dimensiones, se desarrolló el script “procesamiento carpeta por carpeta” para hacer el filtrado de la carpeta con las imágenes que presentaban problemas en su transformación.

- **Keras_Tf_Unid:** Red convolucional entrenada con la información de la base de datos y permite clasificar una sola imagen escogida por el usuario.
- **Pca_SVM_Svc:** Cuaderno que hace la clasificación de radiografías utilizando la técnica de Support Vector Machine y reducción de imágenes con PCA. Se utiliza todo el dataset y se divide 25% para entrenamiento.
- **Variosreducc**: Crea los datasets de entrenamiento y prueba. Grafica 6 imágenes con las distintas técnicas y en el apartado “Reducción y Clasificación por técnica” es el script para realizar la reducción de todas las imágenes del dataset con cada una de las técnicas, realiza la clasificación con SVM-SVC y posteriormente compara todos los resultados con valores y tiempo de ejecución para comparar efectividad de las mismas. El script carga, pero supera la capacidad de procesamiento de la máquina en tiempo y recursos.
- **Entrenamiento:** Cuaderno para la generación de los archivos de pesos y modelo.
    - La primera parte llamada “Convolucional para Radiografías” es para las imágenes sin procesar. Genera los archivos “modelo.h5” y “pesos.h5”.
    - La segunda parte llamada “Entrenamiento con PCA” fue creado para trabajar con las imágenes anteriormente procesadas con el algoritmo de dimensionalidad. Genera los archivos “modelo_PCA.h5” y el “pesos_PCA.h5”.
- **Predicción:** Este cuaderno tiene Scripts para realizar la predicción de una imagen seleccionada, y comparar también la precisión de la red contra todos los datos de prueba, generando también una matriz de confusión. Cada una de estas predicciones esta disponible para las imágenes sin reducir y para las imágenes reducidas.
- **Autoencoder:** Cuaderno que primero procesa las imágenes con la técnica de Autoencoder y luego realiza la clasificación con red convolucional. El cuaderno compila, pero la ejecución es exageradamente larga teniendo que detener la operación porque los recursos de la computadora no son suficientes para procesarlos en un lapso conveniente.

## 5.4. Procesamiento de las imágenes para clasificación

Para tratar de optimizar las operaciones de la red, y evaluar el efecto de usar reducción de imágenes, se crearon varios datasets de imágenes con distintos valores de componentes, que van de 0 componentes, incrementando en 50 hasta 150. Estos datasets son los que se cargaron y se evaluaron en la red para predicción a través de una matriz de confusión.

## 5.5. Reducción dimensional con reducción de tamaño de imágenes:

Tras ensayo y error, se determinó que le valor más adecuado para trabajar en el ordenador fue redimensionar las matrices de las imágenes a una estructura de 150x150 Como valor de reducción inicial.

Posteriormente se realizaron reducciones de 50 en cada caso hasta el valor de 5 como mínimo. Luego se determinó que el mínimo valor posible para reducción encontrado fue 4. Ya menos de este valor no puede ser procesado por la computadora.

## 5.6. Resultados obtenidos

En esta sección se van a discutir los diferentes resultados obtenidos, incluyendo las consideraciones previas y desarrollos para los procesamientos.

### 5.6.1. Análisis Preliminar con PCA:

Como el PCA es un método que permite reducción a través de mantener las mayores varianzas de la imagen y esto a su vez está relacionado con la cantidad de componentes que se decida mantener para los cálculos. Se desarrolló un script para evaluar cual es el número mínimo de componentes que puedo mantener en los cálculos de reducción de manera de poder tener un análisis satisfactorio con el número óptimo de varianzas a mantener.

Este script se llama “pca_individual” y en el mismo analiza como es el porcentaje de reducción de las imágenes según el número de componentes a mantener para el cálculo de los eigenvectores. Con esta finalidad se hizo 3 variantes donde se analizó la reducción tomando en cuenta desde 0 componentes hasta 450. Tal como se observa en la gráfica; a partir de un número de componentes aproximadamente comienza a estabilizarse la reducción.

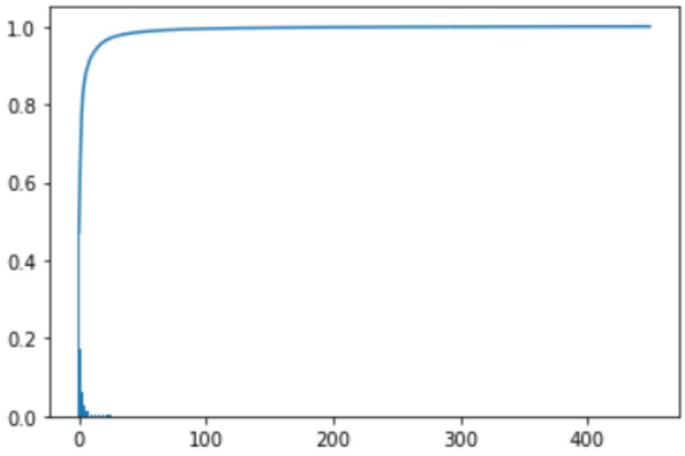


**Ilustración 5.6.1 Comparación de reducción progresiva de una imagen de 0 a 500 con avances de 50 componentes.**

Para tener más precisión sobre el número de componentes que comienza a estabilizar la reducción, se incluyó -en el mismo archivo- una secuencia que evalúa de 0 a 100 componentes con un incremento de 10 componentes por vez. El resultado se muestra a continuación.

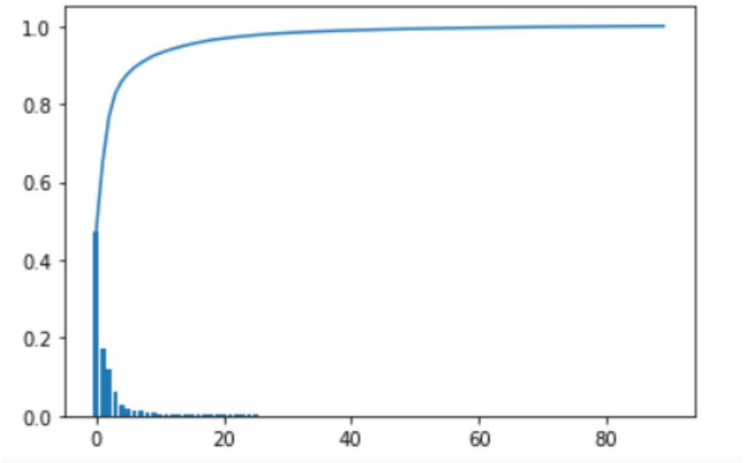


**Ilustración 5.6.2 Gráfico de varianza acumulada y varianza por grupo de componentes de 0 a 100.**

Como se puede observar en el gráfico superior, se observa en los primeros componentes, para entender un poco más el proceso de reducción en las imágenes, se procedió a realizar varias ejecuciones con distintos valores, hasta encontrar que la reducción de 0 a 15 imágenes con incremento de 1 en 1 de los componentes. Muestra la menor cantidad de componentes con mayor impacto en el mismo.

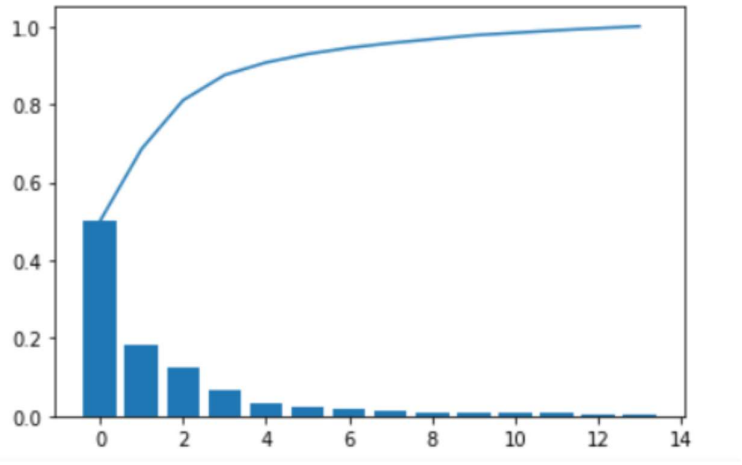


**Ilustración 5.6.3 Gráfico de varianza acumulada de 0 a 15 componentes.**

Pero a pesar de tener el número mínimo de componentes, tras analizas las 3 gráficas, se observa que la varianza acumulada se estabiliza (o tiene un crecimiento mínimo) es a partir

de los 200 componentes más o menos. Razón por la cual se crearon los conjuntos de datos de las imágenes de 0 a 200 imágenes para evaluar la mejor efectividad de los algoritmos.

## 5.7. Análisis de reducción con Script de *Support Vector Machine*:

De forma general se puede observar que el tiempo en generar el clasificador equivale a más del 99% del tiempo de toda la operación y que tanto la precisión mejora considerablemente mientras disminuye el tiempo de operación. A medida que se incrementa componentes aumenta el tiempo de extracción de componentes aumentando en cada uno de los casos.

### 5.7.1. Caso 1, cantidad de componentes = 2

Se buscó realizar la primera clasificación con la menor cantidad posible de componentes seleccionados para la reducción siendo el número inicial de 2 ya que menos de esto daba error el *script*. Los resultados obtenidos muestran que la precisión disminuye bastante al utilizar este

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Sano | 0.40 | 0.76 | 0.52 | 418 |
| Enfermo | 0.85 | 0.54 | 0.66 | 1046 |
| accuracy | | | 0.60 | 1464 |
| macro avg | 0.62 | 0.65 | 0.59 | 1464 |
| weighted avg | 0.72 | 0.60 | 0.62 | 1464 |

**Ilustración 5.7.1 Ejecución sin ningún tipo de reducción.**

Una precisión de 60 % con un tiempo de ejecución de 555.622 segundos siendo el 99,3% del tiempo dedicado al clasificador de la data de entrenamiento.

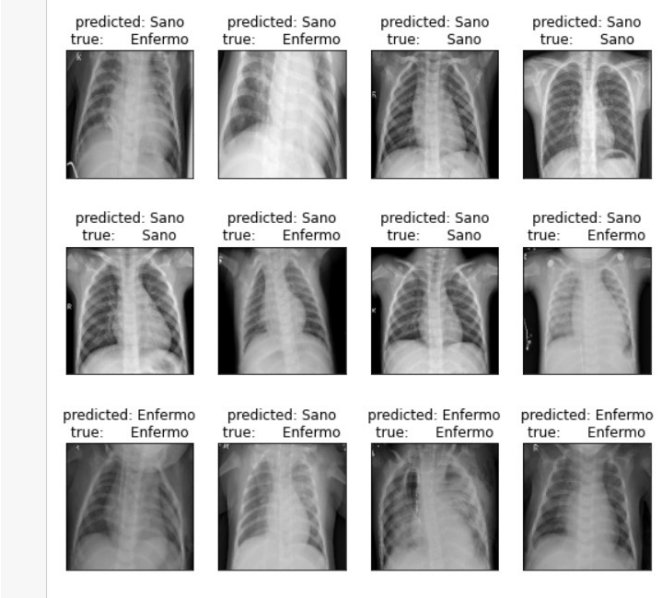


**Ilustración 5.7.2 Imágenes de clasificación inicial.**

### 5.7.2. Cantidad de componentes = 50

En este caso, al incorporar más componentes al cálculo el tiempo de ejecución disminuyó casi a 1/5 del valor del caso anterior y su precisión mejoró notablemente a 95% y el tiempo de ejecución bajó a 116,883 segundos. Lo que significa una reducción de 4,75 veces el tiempo de operación.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Sano | 0.91 | 0.92 | 0.92 | 418 |
| Enfermo | 0.97 | 0.96 | 0.97 | 1046 |
| accuracy | | | 0.95 | 1464 |
| macro avg | 0.94 | 0.94 | 0.94 | 1464 |
| weighted avg | 0.95 | 0.95 | 0.95 | 1464 |

**Ilustración 5.7.3 Resultado de simulación con 50 componentes.**

### 5.7.3. Cantidad de componentes = 100

En este caso, se mantiene la precisión del modelo y el tiempo de ejecución es de 123,661 segundos.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Sano | 0.94 | 0.89 | 0.92 | 418 |
| Enfermo | 0.96 | 0.98 | 0.97 | 1046 |
| accuracy | | | 0.95 | 1464 |
| macro avg | 0.95 | 0.93 | 0.94 | 1464 |
| weighted avg | 0.95 | 0.95 | 0.95 | 1464 |

**Ilustración 5.7.4 Ejecución con 100 componentes.**

### 5.7.4. Cantidad de componentes = 150

La precisión se mantiene en 95% y el tiempo de ejecución total está en 157, 237 segundos.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Sano | 0.94 | 0.89 | 0.91 | 418 |
| Enfermo | 0.96 | 0.98 | 0.97 | 1046 |
| accuracy | | | 0.95 | 1464 |
| macro avg | 0.95 | 0.93 | 0.94 | 1464 |
| weighted avg | 0.95 | 0.95 | 0.95 | 1464 |

**Ilustración 5.7.5 Ejecución con 150 componentes.**

Tal como se observa, para este modelo es altamente efectivo la clasificación con el uso de reducción de imágenes obteniendo un valor de precisión máximo de 95 % utilizando más de 500 componentes para la reducción.

Tal como se observa en los distintos escenarios, no es necesario valores tan grandes de componentes para obtener una buena precisión del modelo con tiempos de ejecución bastante bajos.

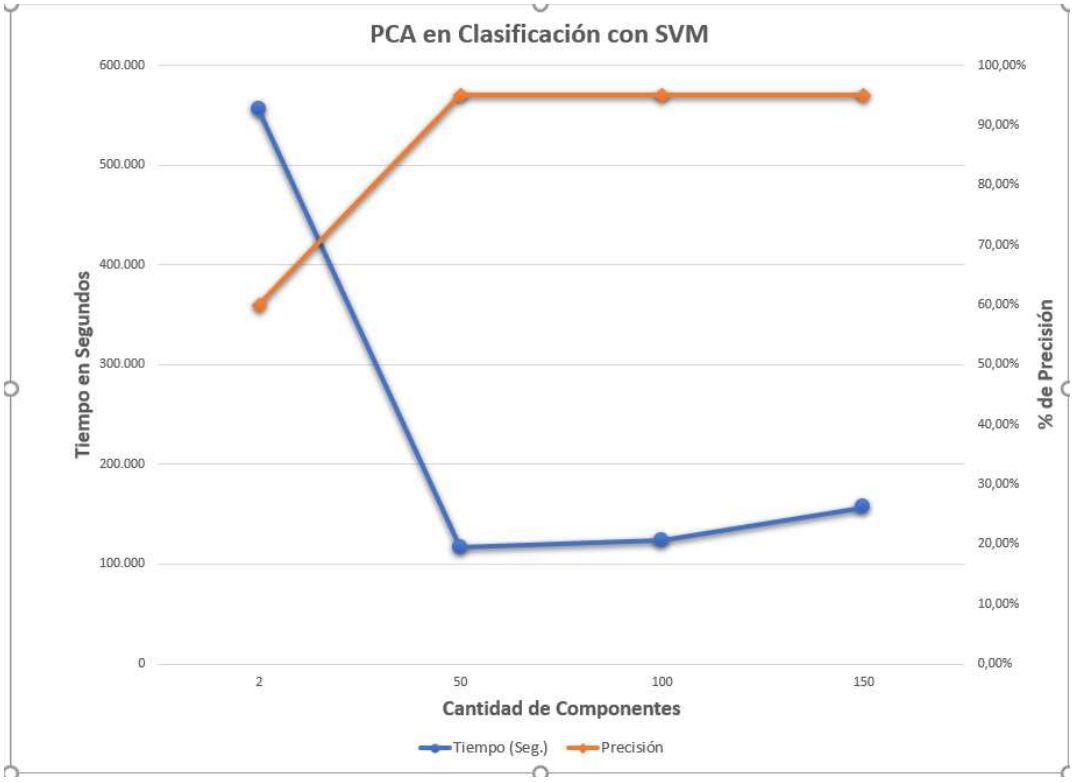


**Ilustración 5.7.6 Tiempo y precisión Vs cantidad de componentes.**

Un hecho importante a resaltar en la imagen anterior es que el comportamiento esperado es el que se obtiene a partir de 50 componentes en adelante y es un incremento de precisión y un leve incremento de tiempo cuando se va aumentando el número de componentes. Sin embargo, cuando se obligó al programa a hacer una reducción de dimensionalidad con el mínimo número de componentes posibles que permitiera ejecutar el algoritmo de reducción. El tiempo de cálculo se incrementó bastante, siendo un comportamiento diferente al esperado. Al analizar los criterios de selección que existen, se mención anteriormente el criterio del 75 % y el uso del *scree plot* (sección 5.1.1.2) y en ambos casos se toman una cantidad de atributos que sean significativos al menos un 75 % de los cambios o donde presente un cambio de pendiente brusco. En este caso se eligió un valor límite en el cual menos de ese valor no se podía aplicar la técnica de reducción. Lo cual aumentó el tiempo de cálculo del programa.

## 5.8. Evaluación de la reducción con técnica Resize

Este caso se evaluó con la red convolucional ya que es la red con mayor complejidad y uso de recursos lo que la hace mucho más crítica para su trabajo.

### 5.8.1. Red Cnn con Resize 150x150

| | | | | |
|---|---|---|---|---|
| 0 | 0.9034 | 0.7991 | 0.8481 | 234 |
| 1 | 0.8873 | 0.9487 | 0.9170 | 390 |
| accuracy | | | 0.8926 | 624 |
| macro avg | 0.8953 | 0.8739 | 0.8825 | 624 |
| weighted avg | 0.8933 | 0.8926 | 0.8911 | 624 |

**Ilustración 5.8.1 Resultados de Entrenamiento Resize 150**

Tiempo ejecución: 8249.644s

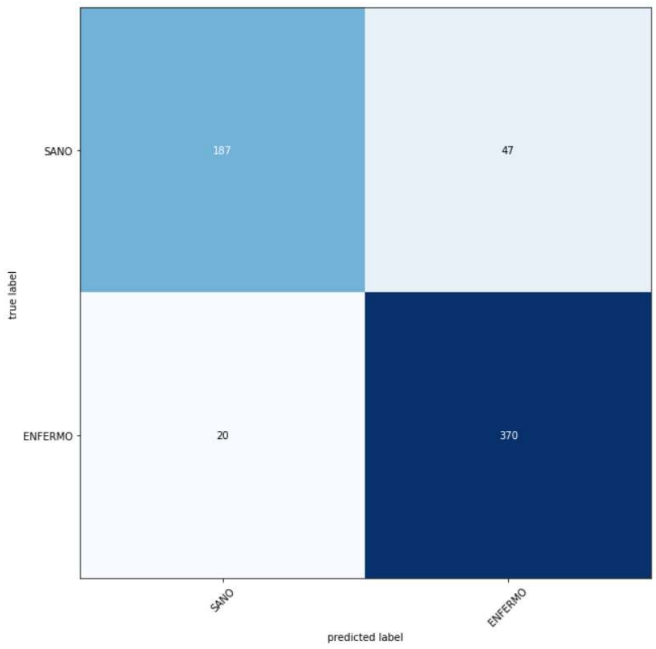


**Ilustración 5.8.2 Matriz de Confusión 150x150**

### 5.8.2. Red Cnn con Resize 100x100

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.9578 | 0.6795 | 0.7950 | 234 |
| 1 | 0.8362 | 0.9821 | 0.9033 | 390 |
| accuracy | | | 0.8686 | 624 |
| macro avg | 0.8970 | 0.8308 | 0.8492 | 624 |
| weighted avg | 0.8818 | 0.8686 | 0.8627 | 624 |

**Ilustración 5.8.3 Resultados de Entrenamiento Resize 100x100**

Tiempo ejecución: 6507.523s

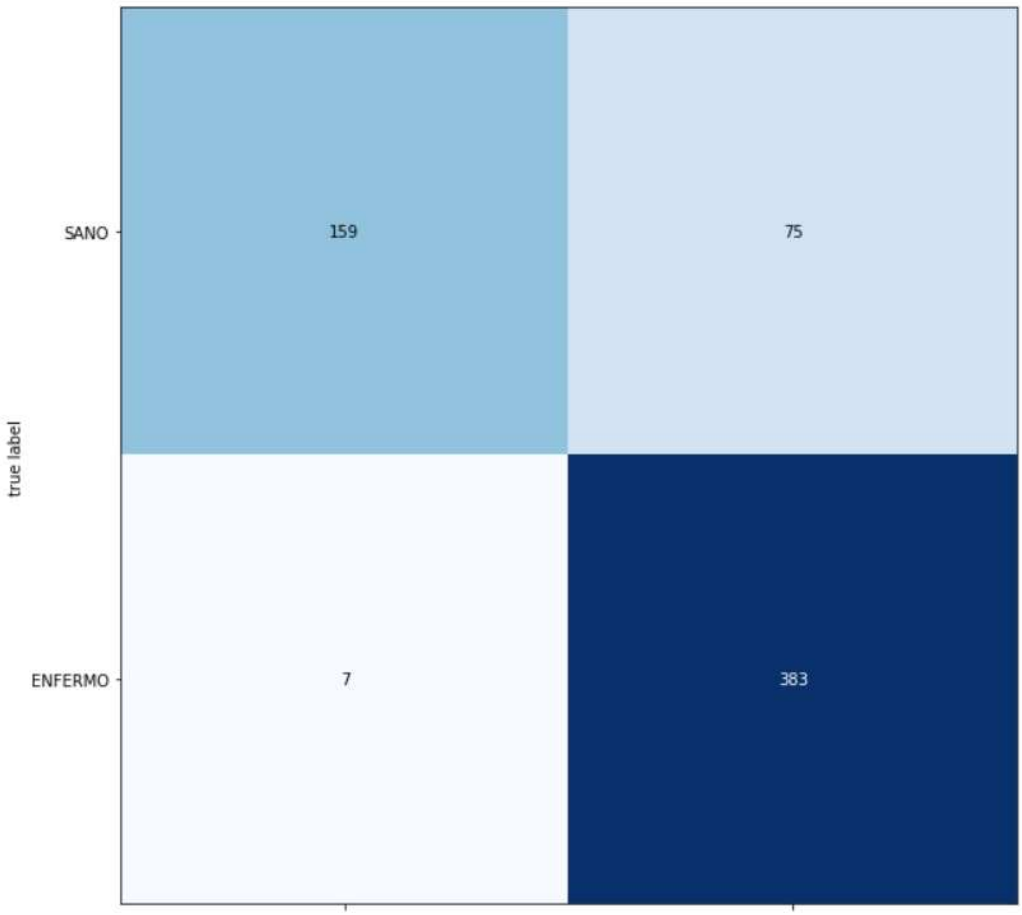


**Ilustración 5.8.4 Matriz de Confusión 100x100**

### 5.8.3. Red Cnn con Resize 50x50

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.9512 | 0.5000 | 0.6555 | 234 |
| 1 | 0.7665 | 0.9846 | 0.8620 | 390 |
| accuracy | | | 0.8029 | 624 |
| macro avg | 0.8588 | 0.7423 | 0.7587 | 624 |
| weighted avg | 0.8357 | 0.8029 | 0.7845 | 624 |

**Ilustración 5.8.5 Resultados de Entrenamiento Resize 50**

Tiempo ejecución: 4811.730s

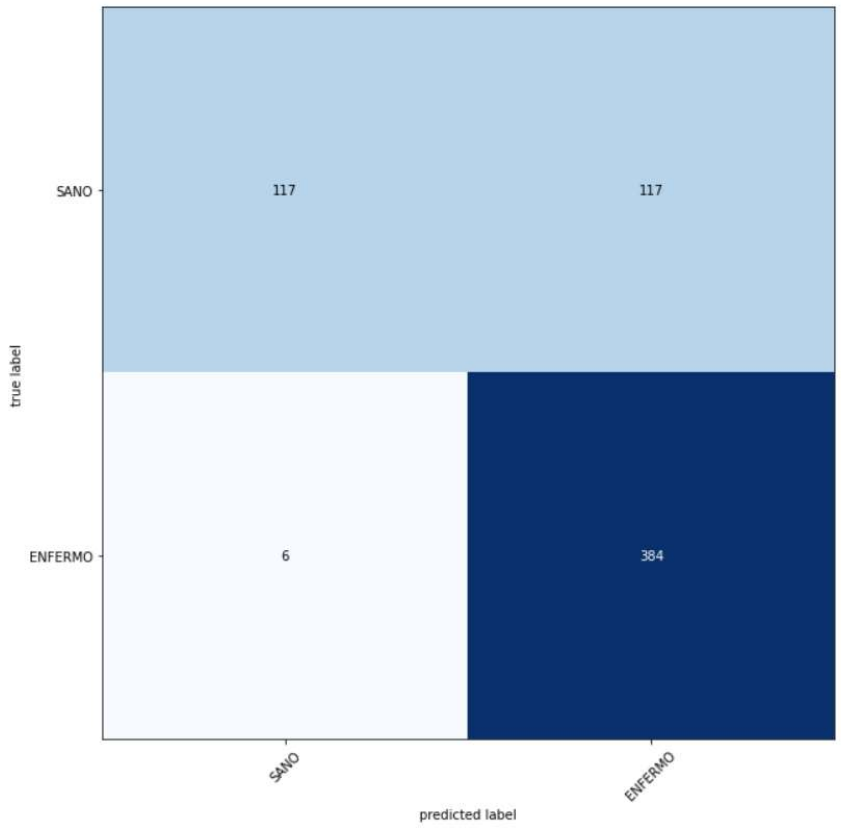


**Ilustración 5.8.6 Matriz de Confusión 50x50**

### 5.8.4. Red Cnn con Resize 5x5:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.8304 | 0.3974 | 0.5376 | 234 |
| 1 | 0.7246 | 0.9513 | 0.8226 | 390 |
| accuracy | | | 0.7436 | 624 |
| macro avg | 0.7775 | 0.6744 | 0.6801 | 624 |
| weighted avg | 0.7643 | 0.7436 | 0.7157 | 624 |

**Ilustración 5.8.7 Resultados de Entrenamiento Resize 5**

Tiempo ejecución: 4300.934 s

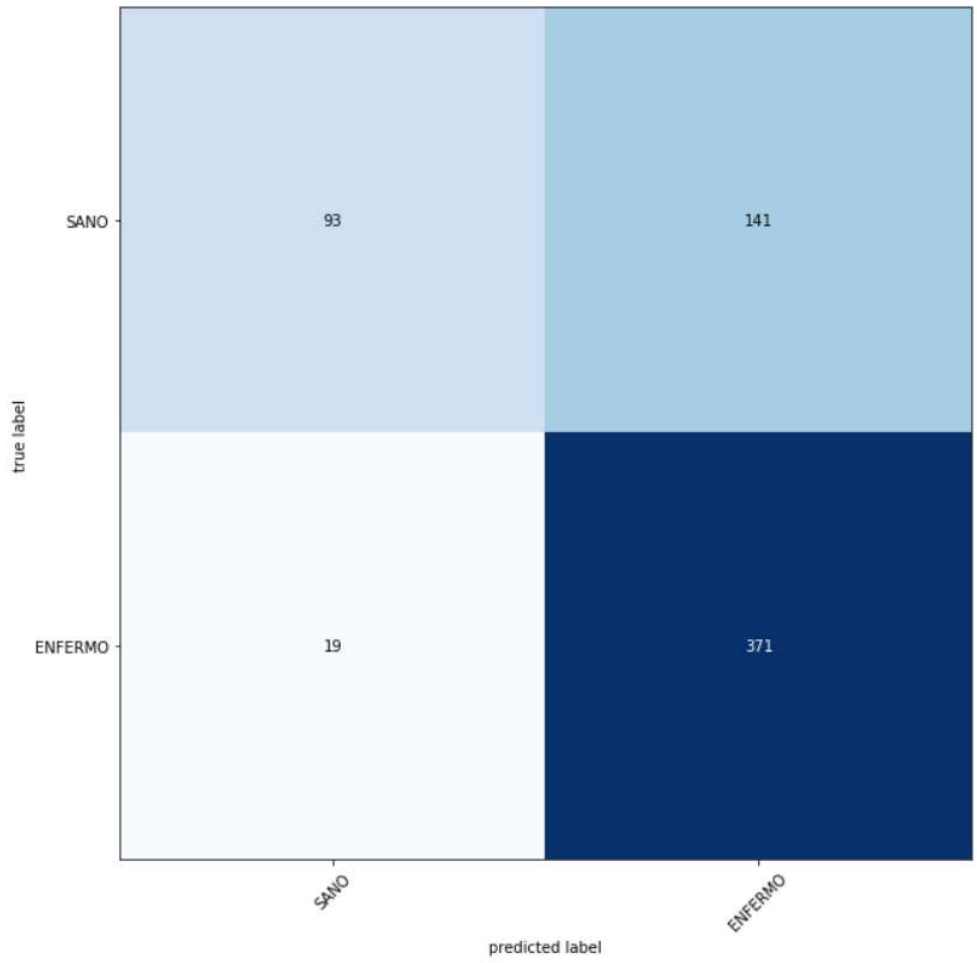

**Ilustración 5.8.8 Matriz de Confusión 5x5**

### 5.8.5. Red Cnn con Resize 4x4:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.5556 | 0.0214 | 0.0412 | 234 |
| 1 | 0.6276 | 0.9897 | 0.7682 | 390 |
| accuracy | | | 0.6266 | 624 |
| macro avg | 0.5916 | 0.5056 | 0.4047 | 624 |
| weighted avg | 0.6006 | 0.6266 | 0.4955 | 624 |

**Ilustración 5.8.9 Resultados de Entrenamiento Resize 4**

Tiempo ejecución: 4376.686s

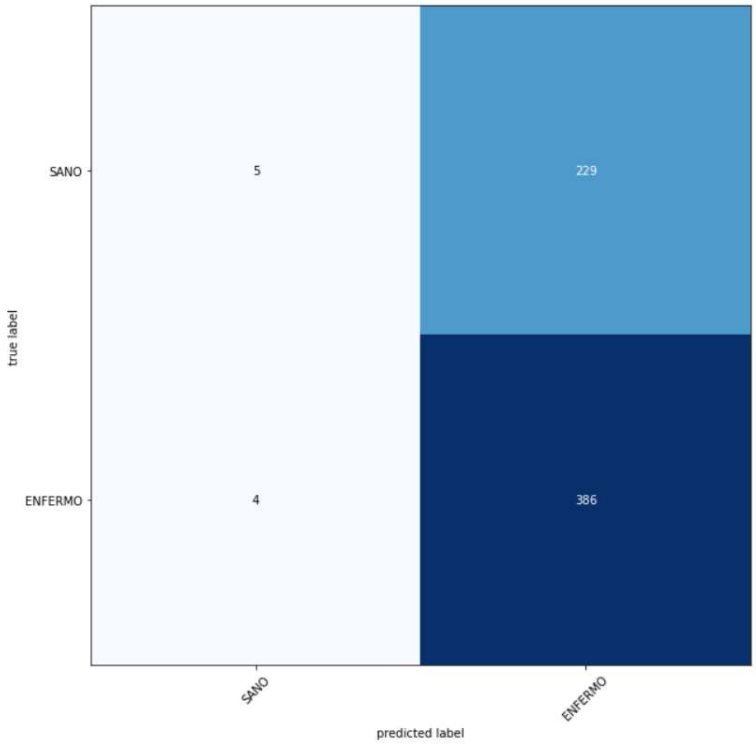


**Ilustración 5.8.10 Matriz de Confusión 4x4**

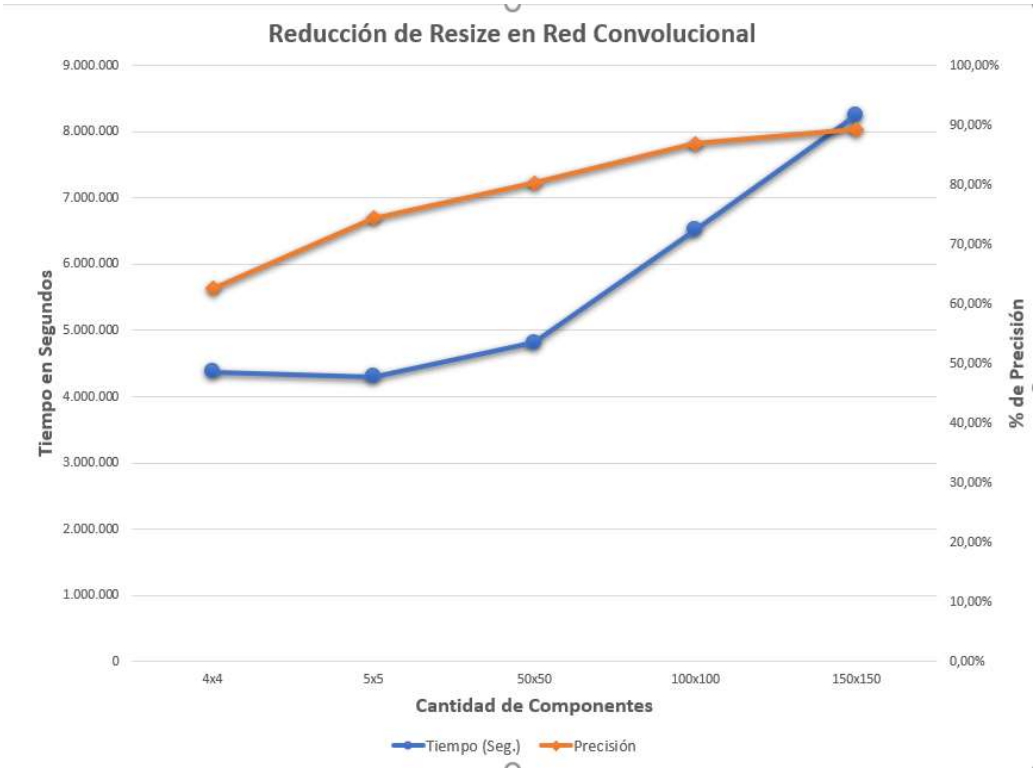


**Ilustración 5.8.11 Tiempo y precisión Vs cantidad de componentes.**

Como se puede ver en la imagen anterior, el uso de esta técnica si tiene un comportamiento donde se incrementa la precisión y el tiempo de ejecución a medida que el tamaño de la matriz es mayor.

## 5.9. Trabajos Futuros

Se trató de realizar evaluaciones de otros procesos de reducción de dimensionalidad que tienen aplicaciones mejores en otros campos. Sin embargo, por falta de tiempo y recursos de la computadora. No fue posible continuar su investigación.

Para tal fin se desarrolló un cuaderno llamado Variosreducc como fue explicado anteriormente. La idea era evaluar estas técnicas con la red convolucional creada. Al ejecutar el programa; se puede observar cómo se realiza el proceso de reducción dimensional de las imágenes con las técnicas y gráfica 6 imágenes aleatorias donde se aplicó las técnicas de reducción a manera referencial para observar cómo quedan las imágenes una vez aplicadas

la reducción. Esto funciona bien para muy poca cantidad de imágenes y solamente reducirlas. Al intentar aplicar a todo el conjunto de imágenes y luego procesar con la red convolucional, el programa no señala error, pero tiene un tiempo de ejecución demasiado largo consecuencia que el programa supera la capacidad de procesamiento del equipo.

Es importante destacar que cada método de reducción tiene aplicaciones diversas, por ejemplo, para señales, otro para imágenes, etc. Tal como se explicó anteriormente en el trabajo. Este desarrollo se puede abordar en otra investigación tomando las consideraciones necesarias.

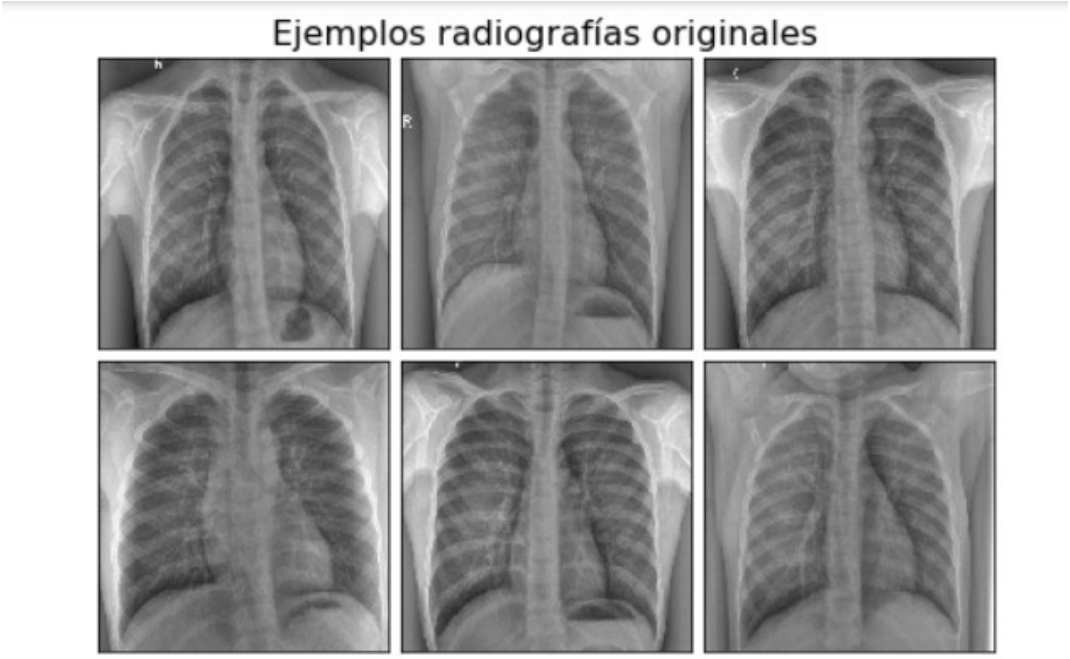


**Ilustración 5.9.1 Imágenes originales usadas para reducción.**

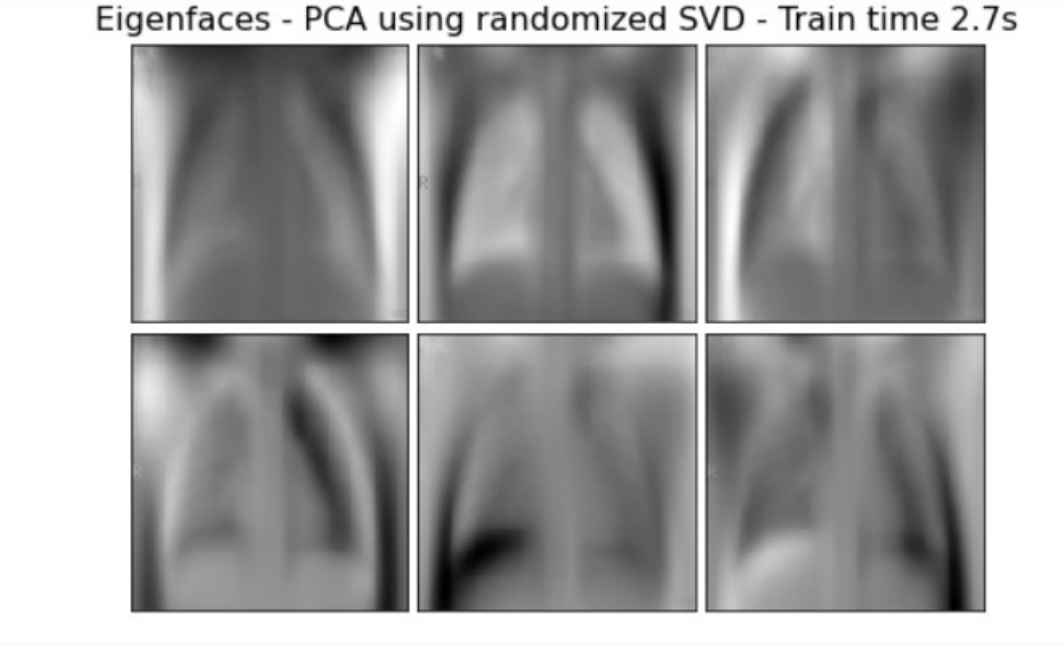


**Ilustración 5.9.2 Imágenes reducidas con PCA.**

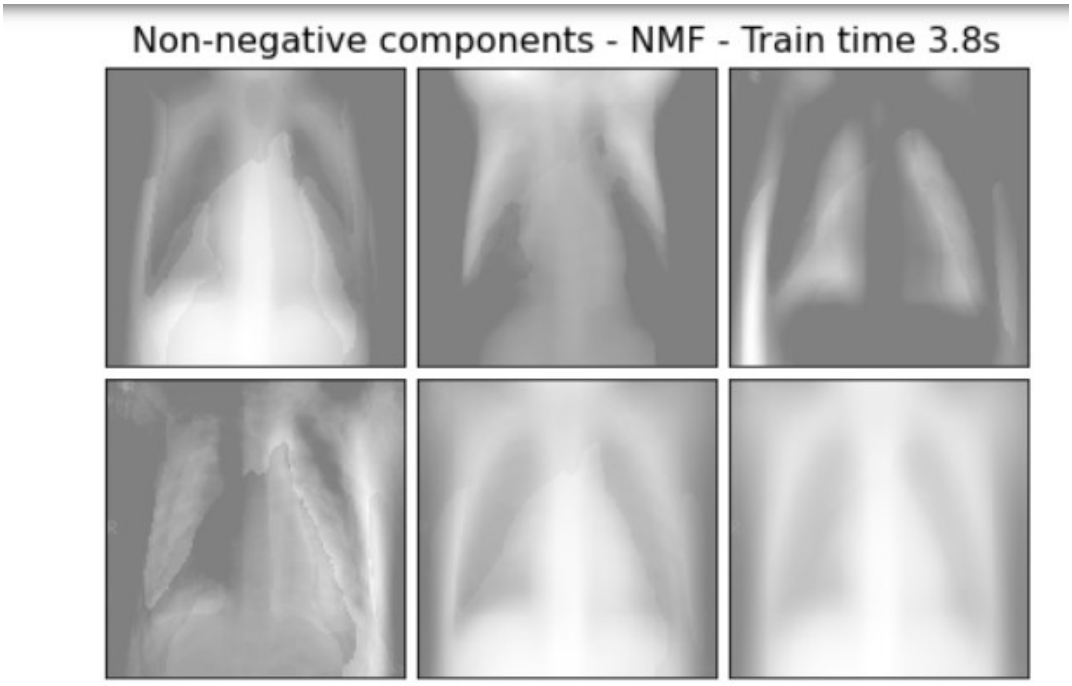


**Ilustración 5.9.3 Imágenes reducidas con técnica NMF.**

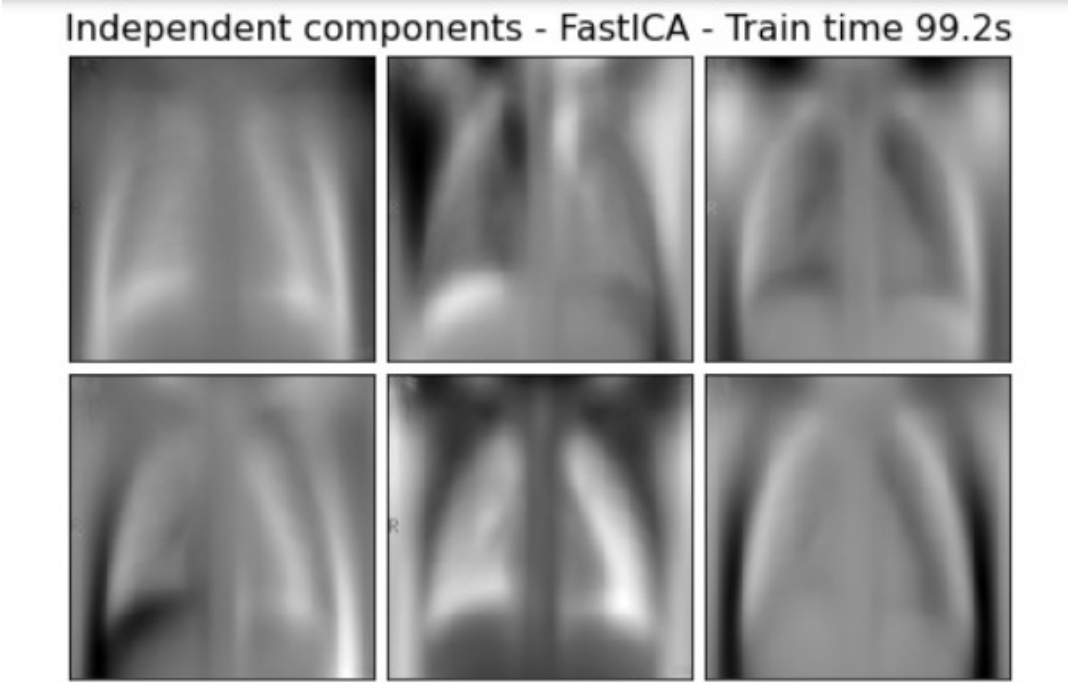


**Ilustración 5.9.4 Imágenes reducidas con técnica FastICA.**

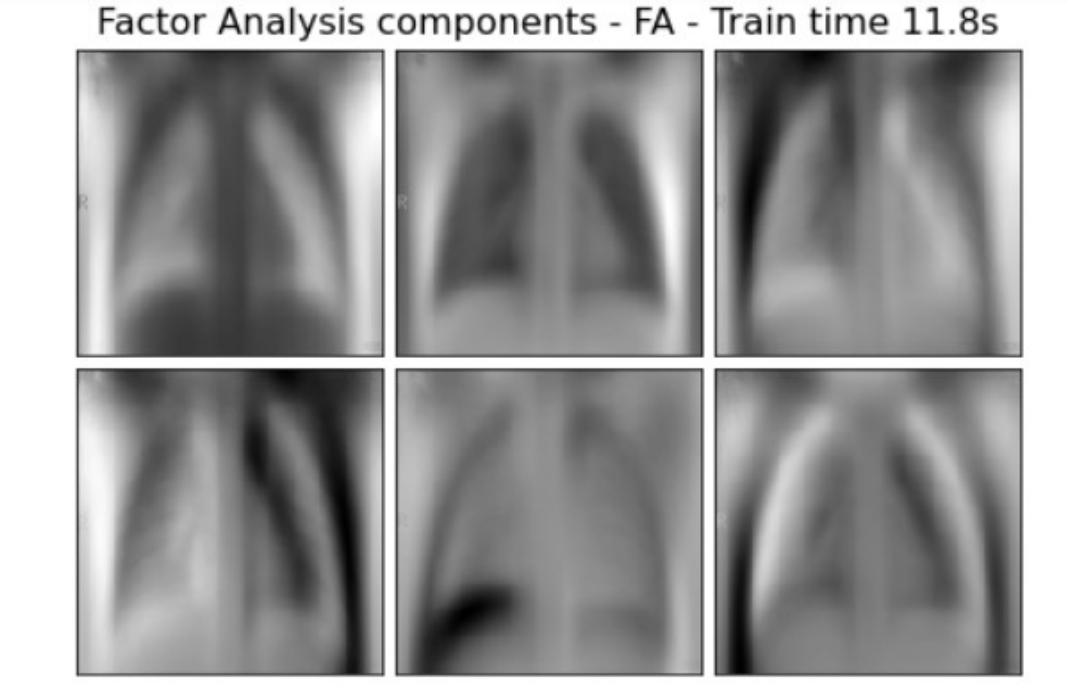


**Ilustración 5.9.5 Imágenes reducidas con técnica FA**

Como se puede observar, el sistema aplica perfectamente los métodos de reducción, pero al aplicarlo de forma masiva a todas las imágenes y realizar la clasificación con la red convolucional requiere una cantidad de recursos mucho mayor de la capacidad del equipo usado en la computadora. Sin embargo, el programa no genera error.

### 5.9.1. Clasificación con Autoencoder.

Autoencoder es un método utilizado para reducción de dimensionalidad en procesos de clasificación no supervisada. El script desarrollado realiza la reducción a través de una red convolucional del autoencoder y posteriormente en el mismo script realiza la clasificación con la red convolucional generada. Como se puede observar en los resultados, se comienza a realizar el proceso de reducción de autoencoder y tanto la reducción como la clasificación se realiza y no genera error, pero igual que el caso anterior, el tiempo de procesamiento es bastante alto, se necesita procesar en un equipo con mayor capacidad y que pueda ser dedicado sin límite de tiempo exclusivamente a este trabajo.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 44, 44, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 32)        18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 16)        4624
_________________________________________________________________
flatten (Flatten)            (None, 1936)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 3874
=================================================================
Total params: 28,754
Trainable params: 28,754
Non-trainable params: 0
_________________________________________________________________
```

**Ilustración 5.9.6 Resultados ejecución con Autoencoder**

# 6. Conclusiones y recomendaciones:

## 6.1. Conclusiones:

En este apartado se muestran las conclusiones obtenidas del análisis de los resultados para todos los casos evaluados anteriormente.

- La COVID-19 se define como una neumonía atípica lo cual, para los efectos de análisis de imágenes por computadora, nos lleva a realizar una clasificación supervisada binaria, ya que no es fiable distinguir un caso de neumonía avanzado de un caso de La COVID-19.
- Las bases de datos que existen de imágenes para esta enfermedad aún están en desarrollo siendo muy pocas confiables y con un número de imágenes muy limitado.
- Todos los programas encontrados donde realizan clasificación de imágenes de radiografías y/o imágenes de cortes axiales de resonancias se realizan con muy pocas imágenes, todas de orígenes diferentes y con funciones de clasificación.
- Tras análisis, la mejor técnica para reducción de dimensionalidad para ser aplicadas en imágenes es la de análisis de componentes principales (PCA en inglés).
- Por la naturaleza de la función PCA, se recomienda hacer un análisis previo para definir el número óptimo de componentes y así optimizar los recursos de la computadora.
- El uso de la técnica de soporte de clasificación de vectores de la librería SVM de Scikit, basado en la técnica de máquina de soporte de vectores. Ofrece mejores resultados en tiempo y precisión que la red neuronal, sin embargo, cabe destacar que la red es una topología sencilla y sin embargo se obtuvieron buenos resultados.
- Con la red convolucional, se pudo demostrar que se tienen buenos resultados de precisión con una estructura sencilla, pero el tiempo de ejecución se incrementaba bastante cuando se realizaba algún pequeño incremento en la estructura de la red.
- La aplicación del redimensionamiento de los valores matriciales de la imagen con el comando resize demostró que al aumentar el tamaño de los datos de la matriz definida aumentaba tanto la precisión como el tiempo de ejecución.

- La aplicación de la técnica de SVM da buenos resultados con la reducción de PCA siendo mayor el tiempo de ejecución con menores componentes de imagen (para este caso se evaluó con el menor número posible de componentes que permitía el programa) dando la más baja precisión y el tiempo mayor de análisis. Siendo el tiempo similar con las imágenes de más componentes.
- Los otros métodos de reducción de dimensionalidad, si bien no son los más adecuados para imágenes, se logró reducir con éxito varias imágenes, pero el equipo utilizado no tenía capacidad de procesamiento para reducir y procesar con toda la base de datos utilizada para el trabajo.

## 6.2. Recomendaciones:

Para el desarrollo de futuros trabajos se recomienda lo siguiente:

- Evaluar la posibilidad de utilizar un equipo de mayor capacidad dedicado especialmente para el procesamiento de los programas.
- Continuar el estudio dado para el desarrollo de los otros algoritmos y variaciones de la topología de la red convolucional.
- Hablar con los institutos que están formando bases de datos de imágenes médicas para tener acceso.
- Extrapolar el presente estudio para evaluaciones médicas de otras enfermedades como por ejemplo enfermedades de piel, etc.

# 7. Anexo

## 7.1. Anexo A. Planificación temporal del proyecto.

| | DISTRIBUCIÓN POR MESES | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **OBJETIVOS** | **1** | **2** | **3** | **4** | **5** | **6** | **Fecha Inicio** | **Fecha Fin** |
| **Revisión de técnicas y métodos de aprendizaje profundo** | | | | | | | 12/10/2020 | 07/12/2020 |
| **Análisis de aplicación de técnicas de reducción de dimensionalidad** | | | | | | | 07/12/2020 | 04/01/2021 |
| **Revisión de Base de Datos Existentes** | | | | | | | 14/12/2020 | 04/01/2021 |
| **Desarrollo de alternativas para la detección del COVID19** | | | | | | | 04/01/2021 | 22/02/2021 |
| **Validación de las propuestas** | | | | | | | 22/02/2021 | 22/03/2021 |
| **Redacción Memoria** | | | | | | | 22/03/2021 | 12/04/2021 |

**Ilustración 7.1.1 Distribución Temporal de las actividades**

El diagrama mostrado corresponde a la distribución inicial de las actividades para la ejecución del trabajo. Sin embargo, en su ejecución hubo desviaciones con respecto al plan incrementando su duración por varios de los puntos explicados en el desarrollo del trabajo.

## 7.2. Ficheros de Código utilizados y desarrollados.

A continuación, se anexan la información utilizada y la generada en el desarrollo del trabajo. Por el tamaño y cantidad de archivos; se coloca los enlaces para su descarga.

- **Cuadernos desarrollados:**
  https://drive.google.com/file/d/10i7aMEn4443uXVb3kcHfISx7TVkoPhJr/view?usp=sharing
- **Base de datos:**
  https://drive.google.com/file/d/1gVaSlltmTYsVPcJXK_kPz9g4ClEm9qwy/view?usp=sharing
- **Resultados obtenidos:**
  https://drive.google.com/file/d/1gGSK9qv76lajkcuRvJ_lzekMUDIjEEQC/view?usp=sharing

## Bibliografía


(s.f.). Obtenido de aloj.us: http://www.aloj.us.es/galba/digital/cuatrimestre_ii/imagen-pagina/

*¿Que es Machine Learning?* (s.f.). Obtenido de https://www.masterdatascienceucm.com/que-es-machine-learning/

Alejo, A. (2014). *Proyecto fin de carrera: Desarrollo de un detector de placas. Universidad de Vallaloid, Tech.*

Arias, J. M. (2019). *Modelo de aprendizaje prfundo/red neuronal convolucional (CNN) para clasificación de calidad de ácidos grasos por imágenes de semillas de Heilianthus annuus.* Cataluña.

Barrios, J. (s.f.). *BIG DATA*. Obtenido de https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Bellman Richard Ernest, CORPORATION, Rand. (1957). *Dynamic programming.* Pinceton University Press.

Buduma, N. (2017). *Fundamentals of Deep Learning.* O`REILLY.

*Canales Sectoriales*. (s.f.). Obtenido de Interempresas: https://www.interempresas.net/MetalMecanica/Articulos/347471-Los-conceptos-de-Machine-Learning-y-Deep-Learning-en-la-industria.html

Chollet, F. (2018). *DEEP LEARNING with Python.* Manning Pulications Co.

*Comparativa de visión humana y visión artificial.* (2014). Obtenido de Sistemas Adaptativos y Bioinspirados en Inteligencia Artificial: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/index.html

*Etiquetado y extracción de características.* (s.f.). Obtenido de https://moodle2012-13.ua.es/moodle/mod/wiki/view.php?

Géron, A. (2019). *Hands-on Machine learnign with Scikit-Learon, Keras & TensorFlow.* O`REAILLY.

Gomez, J. I. (2019). *ResarchGate.* Obtenido de https://www.researchgate.net/figure/Figura-24-Diferencia-entre-el-aprendizaje-profundo-y-el-aprendizaje-maquina-clasico_fig2_334974413

*Imágenes en escala de grises.* (s.f.). Obtenido de http://www.aloj.us.es/galba/digital/cuatrimestre_ii/imagen-pagina/

Maganto, Á. S. (2017). *Estudo de técnicas supervisadas de reducción de dimensionalidad para problemas de clasificación.* Madrid.

Muñoz M, R. (2014). *Sistema de visión artificial para detección y lectura de matrículas.*

*Numpy.org*. (s.f.). Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.resize.html

*Numpy.org*. (s.f.). Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.resize.html

Quereda, I. L. (2018). *Clasificación de imágenes TC de tórax afectados por tuberculosis.* Alicante: Universidad de Alicante.

Rivera, M. (Noviembre de 2018). Obtenido de http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN_LTSM/introduccion_rnn.html

Rodrigo, J. A. (Abril de 2017). *cienciadedatos.* Obtenido de https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

soulmachine. (2017). *Machine Learning Cheat Sheet.* CreativeCommons.

*Tratamiento de imágenes opencv.* (s.f.). Obtenido de https://moodle2012-13.ua.es/moodle/mod/wiki/view.php?

*Tutorial Opencv.* (s.f.). Obtenido de docencia-eupt.unizar.es/ctmedra/tutorial_opencv.pdf

Wangg, L. (21 de 4 de 2021). *github.* Obtenido de https://github.com/lindawangg/COVID-Net

*Wikipedia*. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/S%C3%ADndrome_respiratorio_agudo_grave