Benemérita Universidad
Autónoma de Puebla

Facultad de Ciencias Físico
Matemáticas

Clasificación de MRI hepáticas de infantes utilizando herramientas de Machine Learning.

T E S I S

para obtener el título de:

maestro en ciencias

P R E S E N T A :

Mario Alexis Ramírez Bautista

ASESORES:

Dr. Benito de Celis Alonso
Dr. Javier Miguel Hernández López

junio 2022

Título: Clasificación de MRI hepáticas de infantes utilizando herramientas de Machine Learning.
Estudiante: MARIO ALEXIS RAMÍREZ BAUTISTA.

COMITÉ:

Dr. Eduardo Moreno Barbosa.
Presidente.

Dr. Mario Iván Martínez Hernández.
Secretario.

Dr. Ibrahim Torres Aguilar.
Vocal

Mtra. Margarita Amaro Aranda.
Vocal

Dr. Benito de Celis Alonso Dr. Javier M. Hernández López
Asesores

Agradecimientos

A mi familia por el apoyo que me brinda en cada uno de mis proyectos, por confiar siempre en lo que hago y por ser el pilar más importante en mi vida.

A cada uno de mis amigos con quienes he compartido conocimientos y aventuras, me han regalado momentos inolvidables.

A mis asesores Dr. Benito de Celis y Dr. Javier Hernández por compartirme su sabiduría, por sus consejos y por dirigir este trabajo.

A CONACyT por el apoyo económico otorgado.

Al cuerpo académico de Física Médica por aceptarme dentro del grupo.

A mis profesores de la Licenciatura y Maestría, gracias por sus consejos y enseñanzas.

Abstract

La obesidad infantil es un fenómeno que se ha estudiado desde diferentes perspectivas. En este trabajo se presenta una forma de distinguir entre imágenes de resonancia magnética (MRI) hepáticas de niños con normopeso y con obesidad, utilizando regiones de interés de la zona hepática e imágenes completas. Se emplean herramientas de Machine Learning como clasificadores y modelos de redes neuronales mediante Transfer Learning (TL), finalmente se hace una comparación del desempeño de los modelos empleados para el estudio.

Palabras clave: *Resonancia Magnética*, *Obesidad*, *Hígado*, *Machine Learning*, *Red Neuronal Convolucional*, *Transfer Learning*.

Índice General

Abstract v

1 Introducción **1**
1.1 Justificación 1
1.2 Hipótesis 3
1.3 Objetivo principal 3
1.4 Objetivos secundarios 3
1.5 Resumen de los capítulos 4

2 Marco Teórico **5**
2.1 Obesidad 5
2.1.1 Obesidad infantil 5
2.1.2 Acumulación de grasa en el hígado 6
2.1.3 Padecimientos del hígado como consecuencia de la obesidad 6
2.2 Fundamentos de Resonancia Magnética. 7
2.2.1 Espín y Momento magnético 7
2.2.2 Fenómeno de Resonancia Magnética 9
2.2.3 Tiempos de relajación. 11
2.2.4 Secuencias de saturación de grasa 12
2.2.5 Características generales de las imágenes y su obtención 12
2.3 Machine Learning 15
2.3.1 SVM 16
2.3.2 Redes neuronales 24
2.3.3 Transfer Learning 34

3 Metodología **39**
3.1 Protocolo de obtención de datos 39
3.2 Análisis descriptivo 41
3.3 Clasificadores 41
3.4 SVM 44
3.5 Redes neuronales 44

4 Resultados **49**
4.1 Estadística Descriptiva 49
4.2 Clasificadores 52
4.2.1 Por intensidad promedio de pixeles 52
4.2.2 Por conteo de pixeles 53
4.2.3 Por contraste 55
4.2.4 SVM 56
4.3 Redes Neuronales 58
4.3.1 MODELO: CNN 1. 58
4.3.2 MODELO: CNN 2. 59

5 Discusión y Conclusiones **61**

A Códigos. **65**

Bibliografía **86**

Listado de Figuras

2.1 Configuraciones del agua y grasa para Secuencias de Eco Gradiente (GRE). [35] 13
2.2 Secuencia axial OOP (izquierda) y IP (derecha). [23] . . 14
2.3 Hiperplanos de separación, todos estos planos separan correctamente las clases de datos. [9] 17
2.4 Distancia entre los elementos de ambas clases, el hiperplano óptimo está a la mitad del margen, el cual es la distancia perpendicular entre dos hiperplanos paralelos al central, estos dos planos (líneas grises) son los más próximos a las respectivas clases de datos. [40] 18
2.5 d_- es la proyección del vector x_- sobre w [18] 20
2.6 a) Datos linealmente no separables en $\Re^2$, se muestra el hiperplano de separación que no es capaz de separar las clases. b) se aumenta la dimensión a $\Re^3$, ahora es posible hallar un hiperplano para que las clases se vuelvan linealmente separables. [18] 23
2.7 Neurona biológica [41] 24
2.8 Neurona artificial [41] 25
2.9 Backpropagation, proceso de actualización de los pesos.[48] 28
2.10 Convolución en redes neuronales. [49] 29
2.11 Funcionamiento general de una CNN [27]. 31
2.12 Arquitectura de la red VGG-16 [31] 36
2.13 Arquitectura de la red $VGG-19$.[32] 37

3.1 La región resaltada en color amarillo es la ROI, de tamaño $14x11$ pixeles por cada imagen completa. 40

3.2 Arquitectura del modelo VGG-16 que clasifica MRI hepáticas en normopeso y obesidad. 46
3.3 Arquitectura del modelo VGG-19 que clasifica MRI hepáticas en normopeso y obesidad. 47

4.1 Análisis descriptivo de las clases. 50
4.2 valor de las intensidades promedio de las regiones de interés. 52
4.3 Cantidad de imágenes que tienen un cierto número de pixeles arriba del umbral. 53
4.4 Cantidad de imágenes y contrastes de las ROI´s. 55
4.5 Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a las épocas de entrenamiento. 58
4.6 Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a 500 épocas de entrenamiento. 59
4.7 Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a 700 épocas de entrenamiento. 60

Listado de Tablas

2.1 Constantes giromagnéticas. 8

4.1 Métricas del clasificador basado en la intensidad promedio de pixeles. 53
4.2 Valores de las métricas para el clasificador por conteo de pixeles. 54
4.3 Valores de las métricas para el clasificador por contraste. 56
4.4 Métricas de los clasificadores usando imágenes completas 56

4.5 Tabla de métricas para ROI´s. 56
4.6 Tabla de métricas del SVM para imágenes completas. . . 57

Abreviaturas

ML: Machine Learning.
MR Resonancia magnética.
MRI Imágenes de resonancia magnética.
CNN Red neuronal convolucional.
NAFLD: Enfermedad por hígado graso no alcohólico.
NASH: Esteatohepatitis no alcohólica.
RF Radiofrecuencia.
FID Decaimiento de inducción libre.
SNR: Relación señal-ruido.
CNR: Relación contraste-ruido.
TE: Tiempo de eco.
GRE: Secuencias de eco gradiente.
T1 Tiempo de relajación longitudinal.
T2: Tiempo de relajación transversal.
ROI: Región de interés.
IP: Secuencias en fase.
OOP: Secuencias fuera de fase.
DL: Deep Learning.
TL: Transfer Learning.
VP: Verdadero positivo.
VN: Verdadero negativo.
FP: Falso positivo.
FN: Falso Negativo.
SVM: Máquina de Soporte Vectorial.

CAPÍTULO 1

Introducción

1.1. Justificación

La obesidad es un problema mundial que suele iniciarse en la infancia o en la adolescencia, para estas dos etapas de la vida humana se reporta un incremento a niveles muy altos de obesidad, triplicando las cifras cada diez años [1]. Los estudios sobre este tema se han dado desde diferentes enfoques, los objetivos han sido variados pero el más común es detectar las múltiples patologías que esta condición trae como consecuencia y de las cuales, las enfermedades hepáticas no son la excepción. Las investigaciones han mostrado entre otras cosas, que la acumulación de grasa en la zona del hígado es una de las primeras secuelas de la obesidad [2] además, se ha confirmado que la mayor frecuencia de esteatosis hepática en la población hispana se debe a una mayor prevalencia de obesidad [3]. Los métodos que utilizan para estudios de obesidad son variados, van desde ecografías, bioimpedanciometría, absorción dual de rayos X, plicometría y MRI.

Mediante ecografías se han hecho diagnósticos de hígado graso, sin

embargo, el estudio [4] no respalda el uso de ecografías para el diagnóstico y clasificación de la presencia de grasa en el hígado de niños, proponiendo a la resonancia magnética como un enfoque prometedor en comparación con otras técnicas, lo cual es secundado por [5] donde se describen las ventajas de utilizar imágenes de resonancia magnética para monitorear la presencia de lípidos en el hígado y más aún, se han hecho comparaciones de MRI frente a biopsia, demostrando que la evaluación hepática por resonancia magnética se correlaciona bien con el grado de esteatosis diagnosticado en la biopsia hepática [6], en este sentido, la utilización de MRI de niños con normopeso y obesidad promete ser un buen camino para la realización de un análisis comparativo de ambos grupos.

Hasta hace poco se pensaba que la detección de objetos en imágenes de cualquier tipo era una tarea muy difícil [7]. Actualmente existen técnicas de Machine Learning cuyas aplicaciones en la vida cotidiana son bastas: clasificación, segmentación, reconocimiento facial, funcionamiento de automóviles autónomos y desde luego detección de objetos y muchas aplicaciones que aprovechan los sistemas de visión basados en Redes Neuronales Convolucionales (CNN) por sus siglas en inglés. Dentro del campo de la Física Médica el reconocimiento de objetos en las imágenes y la clasificación son las principales ventajas de los modelos CNN pues han llegado incluso a superar el rendimiento humano [8]. Otros modelos basados en Support Vector Machine (SVM) se enfocan directamente en la clasificación de imágenes, inicialmente se diseñó para clasificación binaria, actualmente se han extendido a clasificaciones multiclase y ha resultado ser de los mejores clasificadores para un amplio abanico de situaciones, por lo que son considerados como referentes dentro del ámbito de aprendizaje estadístico y Machine Learning (ML) [9].

Una de las limitaciones de los modelos de clasificación basados en redes neuronales es la poca disponibilidad de datos, pues es bien conocido dentro del ámbito de Machine Learning que, para hacer un modelo de red neuronal desde cero, se requiere una gran cantidad de datos, existiendo una relación estrecha entre la cantidad de datos del modelo y la calidad del mismo [10]. Para solventar este problema particular, el Transfer Learning permite transferir y aprovechar el conocimiento de

una red profunda que inicialmente se construyó para resolver un problema diferente y aplicarlo en un nuevo modelo haciendo adaptaciones para su correcto funcionamiento [11].
Aplicar modelos de Machine Learning en Física Médica es una de las innovaciones más recientes, se tienen cantidades de datos (principalmente imágenes) de muchos tipos para ser analizados. Recientes investigaciones sobre análisis de imágenes se han hecho notar debido a las ventajas que trae consigo la creación de nuevos modelos, obteniendo cada vez mejores y mejores desempeños de estos.

1.2. Hipótesis

Se pueden construir clasificadores de MRI hepáticas de niños que distingan entre la condición normopeso/obesidad a partir de herramientas de ML.

1.3. Objetivo principal

Hacer un estudio detallado para separar imágenes de resonancia magnética del hígado implementando técnicas de Machine Learning como clasificadores y redes neuronales a partir del Transfer Learning, creando modelos que permitan dar una diferenciación entre las imágenes obtenidas de niños con obesidad de aquellos normopeso.

1.4. Objetivos secundarios

- Analizar las características de imágenes de resonancia magnética hepáticas.
- Realizar un análisis descriptivo de los datos.
- Implementar un modelo de clasificación de los datos utilizando aprendizaje supervisado.

- Estudiar las bases y el desarrollo de las redes neuronales en su aplicación a problemas de valoración de imágenes de resonancia magnética.
- Manejar el framework conjunto de Tensorflow y Keras para la construcción de redes neuronales convolucionales utilizando Transfer Learning.
- Comparar y evaluar los modelos de clasificación creados.

1.5. Resumen de los capítulos

El trabajo de tesis se conforma de la siguiente manera. Se presenta el marco teórico en el capítulo 2, empezando con la Obesidad y su impacto en la comunidad infantil de México, así como los padecimientos que trae como consecuencia la acumulación de grasa en el hígado, se presentan los fundamentos de MRI que abarcan desde entendimiento básico del fenómeno de la Resonancia Magnética (MR) hasta las características generales de las imágenes, su obtención y las secuencias del hígado. Se incluye la parte de Machine Learning, donde aparece la teoría relacionada con el clasificador SMV, las redes neuronales en general, las redes convolucionales y el Transfer Learning. Luego, se describe en el capítulo 3 la metodología seguida para la creación de los clasificadores y los modelos de redes neuronales. Posteriormente, en el capítulo 4, se reportan los resultados de los diferentes clasificadores y de los modelos de redes tras la utilización del Transfer Learning para regiones de interés y para las imágenes completas; además, en el capítulo 5, se abre la discusión sobre los principales resultados y hallazgos del estudio hecho junto con las conclusiones a las que se llegaron. Finalmente, en la última parte se integra el apéndice A con el código implementado en el lenguaje Python sobre los modelos creados.

CAPÍTULO 2

Marco Teórico

2.1. Obesidad

La obesidad es una condición de salud que ha estado presente desde tiempos muy remotos, se define como el resultado de ingerir más calorías de las que se queman durante el día con el ejercicio o las actividades diarias normales; el cuerpo almacena ese exceso de calorías en forma de grasa y por ende, se incrementa el peso corporal, el cual es una combinación de varias componentes como lo son las influencias genéticas, conductuales, metabólicas y hormonales [12].

2.1.1. Obesidad infantil

La obesidad infantil en México es un problema que se presenta desde la infancia, es decir, entre 0 y 5 años. Al menos 1 de cada 20 niños y niñas menores de 5 años padece obesidad [13], lo que favorece el sobrepeso durante el resto de su vida y los pone en riesgo de sufrir enfermedades de muchos tipos como circulatorias, del corazón, riñones, diabetes y particularmente patologías hepáticas, entre otras.

El incremento de casos de obesidad infantil en México se asocia con los malos hábitos alimenticios que tienen su origen desde edades muy cortas y que son el punto de partida de un estilo de vida. Existen muchos estudios que demuestran que los niños y adolescentes con alto consumo de frutas y vegetales tienen el doble de probabilidad de apegarse a recomendaciones dietéticas que son saludables [14], lo cual hace mucho sentido incluso para el caso contrario en cuanto a malos hábitos alimenticios, difícilmente se corrigen a lo largo de los años. Debido a esto, México se encuentra entre los primeros lugares en obesidad infantil en el mundo.

2.1.2. Acumulación de grasa en el hígado

El consumo de bebidas azucaradas como las sodas y las bebidas energéticas son altas en fructosa, que ha mostrado ser la causa de acumulación de grasa en el hígado en niños, misma que se puede ver y diagnosticar mediante MRI, siendo un método no invasivo que permite detectar dicha grasa y cuantificarla; hasta hace poco se creía que el método invasivo más fiable era la biopsia hepática, sin embargo un nuevo estudio afirmó que la resonancia magnética es mejor [15].

2.1.3. Padecimientos del hígado como consecuencia de la obesidad

La acumulación de grasa en el hígado como consecuencia de la obesidad trae como resultado una serie de padecimientos que en primer momento pueden ser asintomáticos y en la edad adulta se llegan a desarrollar llegando a problemas más graves.

La enfermedad del hígado graso (esteatosis) es una afección común causada por la acumulación de grasa en el hígado. Un hígado sano contiene una pequeña cantidad de grasa. Se convierte en un problema cuando la grasa alcanza del 5 % al 10 % del peso de su hígado.

Otro padecimiento común es la Enfermedad del Hígado Graso No

Alcohólico (NAFLD, siglas en inglés) incluye la infiltración grasa simple (una condición benigna llamada hígado graso), consecuencias más graves de obesidad como la esteatohepatitis no alcohólica (NASH) conduce a lipotoxicidad y daño inflamatorio de los hepatocitos [17]. Histológicamente, la NASH es difícil de distinguir de la hepatitis alcohólica. Por lo tanto, para el diagnóstico de NASH, se debe descartar el consumo de alcohol, como en el caso de los niños. La presencia de síndrome metabólico (obesidad, dislipidemia, hipertensión e intolerancia a la glucosa) aumenta la probabilidad de que un paciente tenga NASH en lugar de una simple esteatosis [19].

2.2. Fundamentos de Resonancia Magnética.

2.2.1. Espín y Momento magnético

El espín es una característica intrínseca de la materia, dentro de un átomo se encuentran tres partículas (protón, electrón y neutrón), cuando se hace la medición del estado de espín de una de estas partículas en cualquiera de las tres direcciones espaciales, resulta que el valor es $S_i = \pm\frac{\hbar}{2}$ y por lo tanto la cantidad total de espín de cada partícula es $\frac{\sqrt{3}\hbar}{2}$.

El momento magnético de espín se define de la siguiente manera:

$$\bar{\mathbf{m}} = \gamma\hbar\bar{\mathbf{S}} \tag{2.1}$$

donde γ es la constante giromagnética y tiene un valor específico para cada tipo de partícula o núcleo, en la siguiente tabla se enlistan las constantes giromagnéticas del electrón, protón y neutrón:

Cuando se tiene un sistema de partículas en ausencia de un campo magnético, por ejemplo, protones, cada partícula tendrá una cierta orientación de espín hacia cualquier parte del espacio y en consecuencia un cierto momento magnético asociado; los espines de cada protón se combinan de tal forma que producen un espín total I y a su vez los momentos magnéticos de cada partícula se combinan para formar un

Partícula o núcleo	$\gamma/2\pi \quad (MHz/T)$
electrón	28.025
protón	42.577
neutrón	29.165

Tabla 2.1: Constantes giromagnéticas.

momento magnético resultante.

Por otra parte, cuando se coloca a un conjunto de protones dentro de un campo magnético $\bar{\mathbf{B}}$, los momentos magnéticos $\bar{\mathbf{m}}$ de las partículas tienden a alinearse con $\bar{\mathbf{B}}$ teniendo dos posibilidades de orientación (paralela o antiparalela) al campo magnético $\bar{\mathbf{B}}$; cuando la alineación de los espines ocurre, la energía de los protones cambia, donde inicialmente tenía un valor E_0, convirtiéndose ahora en:

$$E = E_0 - \bar{\mathbf{m}}\bar{\mathbf{B}} = E_0 \pm \frac{1}{2}\hbar\gamma B \tag{2.2}$$

de donde se puede ver que, de la energía original que tenía un protón, tras la presencia de $\bar{\mathbf{B}}$ puede haber dos posibilidades, que tenga una energía mayor (del signo +) y otra de menor energía (del signo -), las cuales corresponden respectivamente a las alineaciones antiparalela y paralela, por este motivo se puede decir que el momento magnético paralelo al campo $\bar{\mathbf{B}}$ es el que requerirá una energía menor para conseguir su alineación y por ende habrá más protones cuyos momentos magnéticos sean paralelos a $\bar{\mathbf{B}}$.

De acuerdo con lo anterior, se puede entender que con la presencia de un campo magnético $\bar{\mathbf{B}}$, la energía de las partículas puede tomar dos valores distintos, la diferencia entre estos dos valores de energía es, precisamente

$$\Delta E = \hbar\gamma B \tag{2.3}$$

Cuando se tiene un protón con orientación paralela a un campo magnético $\bar{\mathbf{B}}$ (estado de menor energía), este protón puede pasar a tener una

orientación antiparalela (energía mayor) y, para que esto ocurra bastará con que el protón absorba un fotón cuya energía sea precisamente la diferencia de energía entre los dos estados (2.3).

La frecuencia a la cual ocurre esta transición se llama frecuencia de Larmor:

$$\nu = \gamma B \tag{2.4}$$

2.2.2. Fenómeno de Resonancia Magnética

El fenómeno de la resonancia magnética nuclear fue descrito experimentalmente por primera vez por Purcell y Bloch en 1946 [20, 21], y seis años más tarde, recibieron el Premio Nobel de Física.
La técnica fue cambiando rápidamente, pues se introdujo una amplia configuración de imanes superconductores, lo que permitió el desarrollo de aplicaciones clínicas y actualmente la resonancia magnética nuclear es una poderosa herramienta clínica disponible.

Los núcleos atómicos de todos los elementos están formados por protones y neutrones, teniendo una carga neta positiva. Existen ciertos núcleos atómicos que tienen espín, lo cual puede interpretarse como el núcleo atómico girando alrededor de su propio eje, sin embargo, el núcleo en sí no gira en el sentido clásico, pero en virtud de sus partes constituyentes induce un momento magnético, generando un campo magnético local con polos norte y sur. La descripción cuántica de este imán es similar a la clásica en cuanto al tratamiento de objetos giratorios, un ejemplo claro es el dipolo, que se entiende como imán común, con polos magnéticos alineados a lo largo de su eje de rotación.

La aplicación de un fuerte campo magnético externo $\bar{\mathbf{B}}_0$ alinea el núcleo en paralelo o antiparalelo al campo externo. Un imán común se orientaría completamente paralelo o antiparalelo al campo, el núcleo tiene un momento angular debido a su rotación, por lo que rotará o tendrá un movimiento de precesión alrededor del eje de $\bar{\mathbf{B}}_0$. Este comportamiento a menudo se compara con el movimiento oscilante de un giroscopio bajo la influencia del campo magnético de la Tierra y explica el uso del "espín" para explicar lo que en realidad es un fenómeno me-

cánico cuántico. La velocidad de rotación alrededor de la dirección del campo es la frecuencia de Larmor. Esto es proporcional a la intensidad del campo y se describe mediante la ecuación de Larmor (2.4)

Los núcleos que poseen espín pueden excitarse dentro del campo magnético estático, $\bar{\mathbf{B}}_0$, mediante la aplicación de un segundo campo magnético de radiofrecuencia (RF) $\bar{\mathbf{B}}_1$, aplicado perpendicularmente a $\bar{\mathbf{B}}_0$. La energía de RF generalmente se aplica en pulsos cortos, cada microsegundo de duración. La absorción de energía por parte del núcleo provoca una transición de niveles de energía más altos a más bajos y viceversa en la relajación. La energía absorbida (y posteriormente emitida) por los núcleos induce un voltaje que puede ser detectado por una bobina de alambre sintonizada adecuadamente, amplificada y mostrada como "decaimiento de inducción libre" (FID) por sus siglas en inglés. En ausencia de una pulsación de RF continua, los procesos de relajación devolverán el sistema al equilibrio térmico. Por lo tanto, cada núcleo resonará a una frecuencia característica cuando se coloque dentro del mismo campo magnético.

Para inducir la transición entre los niveles se requiere cierta energía y es, como ya se mencionó, la diferencia de energía entre los dos estados de espín. Esto depende de la fuerza del campo magnético $\bar{\mathbf{B}}_0$ al que están sujetos los núcleos.
De manera cotidiana se aplican múltiples pulsos de RF para obtener múltiples FID, mismos que posteriormente se promedian y de este modo se mejora la relación señal-ruido (SNR).
La FID promediada por señal es una señal en el dominio del tiempo y se puede resolver mediante un proceso matemático conocido como transformación de Fourier [37], en una imagen de MRI.

El uso de gradientes permite localizar de manera espacial la señal de MR en una región de interés. Usando tres conjuntos de bobinas de gradientes, G_x, G_y, G_z de un sistema RM es posible aplicarlos en cualquier dirección respectiva para llegar a cualquier parte del cuerpo. La construcción de las imágenes en tres dimensiones se da a partir de las mediciones de frecuencia que permiten distinguir las señales de MR en diferentes posiciones en el espacio.

Dependiendo de la región del cuerpo a examinar, las bobinas del

transmisor y del receptor pueden ser independientes. El pulso de RF se aplica mediante una bobina transmisora, que rodea de manera uniforme el área de interés. La bobina receptora puede estar integrada dentro de la bobina transmisora o se puede colocar sobre la región de interés de forma directa. Cuando se tienen diferentes bobinas receptoras, se hace la integración de las señales recibidas mediante algoritmos matemáticos y es así como se genera una imagen reconstruida óptima [22].
Aunque actualmente se utilizan escáneres de resonancia magnética de diagnóstico con imanes superconductores criogénicos en el rango de 0.5 Tesla (T) a 1.5 T, los sistemas de 3T tienen mayor auge en el ámbito de la investigación. Las ventajas de los sistemas de mayor intensidad de campo incluyen una mejor SNR, una mayor resolución espectral, espacial y temporal [22].

2.2.3. Tiempos de relajación.

Relajación es el término utilizado para describir el proceso por el cual un espín nuclear vuelve a su equilibrio térmico tras haber absorbido energía de RF.
Hay dos tipos de relajación, relajación longitudinal y relajación transversal, mismos que se describen mediante los tiempos, T_1 y T_2, respectivamente [38].
T_1 es el tiempo que tarda el sistema en volver un 63 % hacia el equilibrio térmico después de un pulso de RF como una función exponencial del tiempo. El agua tiene valores de T_1 largos que van desde los 3000 a los 5000 milisegundos y en las imágenes ponderadas en T_1 la señal se muestra obscura, por otro lado, la grasa tiene un valor de T_1 corto de aproximadamente 260 milisegundos, por lo cual aparece brillante.

Cuando se aplica un pulso de RF, los núcleos se alinean predominantemente a lo largo del eje de la energía aplicada y en la relajación, a medida que la energía se transfiere entre los núcleos, hay un cambio de fase de las orientaciones de estos, con una disposición más aleatoria de las alineaciones. Esto es T_2, denominado relajación transversal, porque es una medida de la rapidez con que los espines intercambian energía en

el plano "xy". T_2 también se conoce como relajación espín−espín [22].

2.2.4. Secuencias de saturación de grasa

Las imágenes que se obtienen del equipo de MR son llamadas en fase (IP) o fuera de fase (OOP), la importancia de estas dos secuencias radica en que mediante la manipulación de estas, se pueden obtener las secuencias solo de grasa (F) o solo agua (W).
Se parte del hecho de que las imágenes IP, como las OOP, tienen la siguiente información en términos de W y F.

$$IP = W + F$$
$$OOP = W - F$$

Haciendo una transformación se obtiene:

$$W = \frac{1}{2}(IP + OOP) \tag{2.5}$$

$$F = \frac{1}{2}(IP - OOP) \tag{2.6}$$

Donde las ecuaciones (2.5) y (2.6) corresponden a imágenes donde se resalta la señal del agua o grasa, respectivamente. Particularmente esta forma de obtener las MRI con saturación de grasa se aplican a las imágenes axiales de hígado para su posterior análisis.

2.2.5. Características generales de las imágenes y su obtención

Las imágenes que se utilizan en los tratamientos médicos son obtenidas por diferentes métodos que con el paso de los años van siendo mejorados, el objetivo es que finalmente arrojen una MRI que tenga ciertas características que permitan diferenciar la calidad de las mismas. Existen tres parámetros que resultan ser importantes para las imágenes y sobre todo para el diagnóstico:

1. Resolución espacial

2. Relación señal-ruido (SNR)

3. Relación contraste-ruido (CNR)

estos puntos en una imagen conviene tenerlos lo más arriba posibles, pero no siempre se pueden maximizar, pues los mecanismos a veces no funcionan de igual manera y en ocasiones mejorar alguno, significa empeorar los otros, es por esta razón que se debe hacer una elección adecuada dependiendo de cada caso particular para mejorar el diagnóstico.
Debido a que las frecuencias de los protones de agua y grasa varían ligeramente, sus giros entran y salen de fase entre sí en función del Tiempo de Eco (TE), o sea, el tiempo transcurrido entre el primer pulso enviado y la recogida de señal. Las condiciones IP y OOP ocurren dos veces por ciclo, o aproximadamente cada 2.2 milisegundos a $1.5T$ y a $3T$ es dos veces más rápido dado que la intensidad del campo es doblemente mayor, es decir, se produce cada $1.1 milisegundos$ [36], como se muestra a continuación:

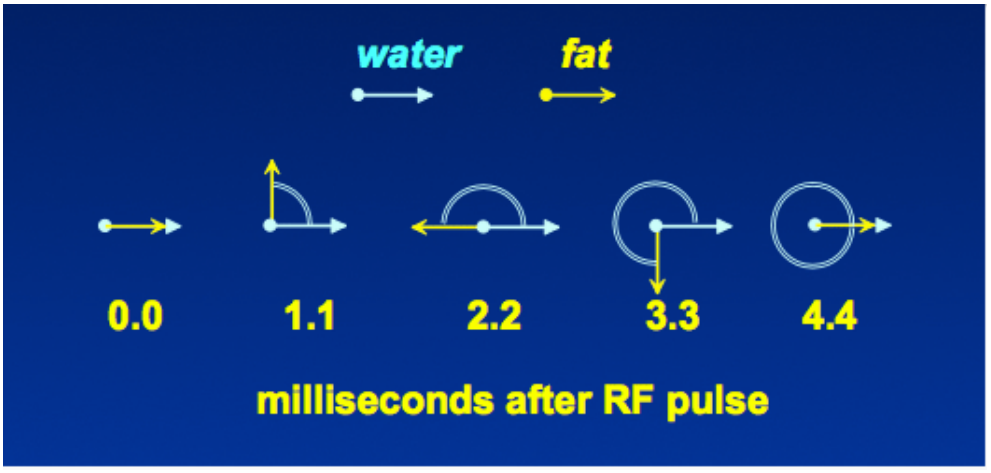

Figura 2.1: Configuraciones del agua y grasa para Secuencias de Eco Gradiente (GRE). [35]

Por lo tanto, siguiendo el patrón de la imagen anterior, podemos notar que las imágenes GRE (pulso de 90^o) obtenidas con un TE de 2.2,

6.6, 11 milisegundos se denominan fuera de fase (OOP); las obtenidas a 4.4, 8.8, etc. se denominan en fase (IP). Ejemplos de estas imágenes son las que se muestran en la siguiente figura:

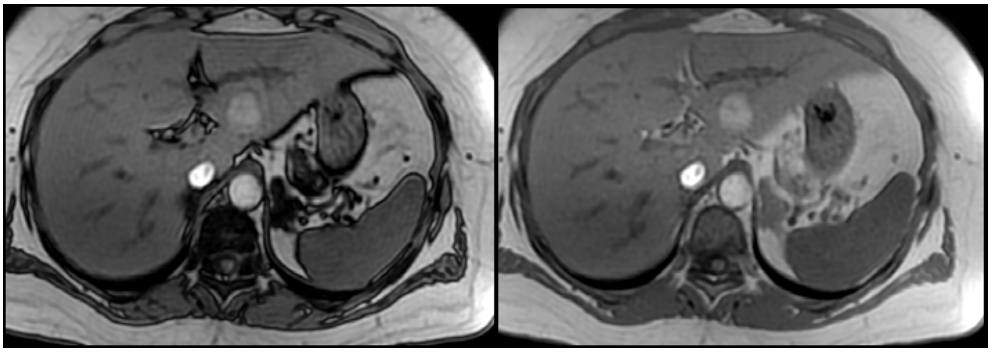

Figura 2.2: Secuencia axial OOP (izquierda) y IP (derecha). [23]

Los protones de la grasa y del agua comienzan en fase cuando se aplica el pulso de RF pero tienen un T_1 diferente, es por eso que se desfasan.

2.3. Machine Learning

La construcción de modelos de Machine Learning requieren un principio de retroalimentación constructiva. En primer lugar, se construye un modelo, se obtienen las métricas y se van realizando mejoras hasta lograr un valor deseable de éstas. Para explicar el rendimiento de un modelo se utilizan las conocidas "métricas de evaluación". Siempre es importante que, al terminar la construcción de un modelo, se compruebe la solidez del mismo.

Para la evaluación de los resultados en el conjunto de datos, normalmente se calculan algunas de las siguientes métricas de rendimiento: Exactitud, Sensibilidad o F1 Score, las cuales se describen a continuación:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

$$Sensibilidad = \frac{TP}{TP + FN} \tag{2.8}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

$$F_1 Score = \frac{2TP}{2TP + FP + FN} \tag{2.10}$$

Se define verdadero positivo (TP) como un indicador de que el modelo predijo correctamente la clase de sujeto al comparar con su etiqueta respectiva "x''. El falso positivo (FP), ocurre cuando el modelo predice incorrectamente que un sujeto de clase "x'' que no es de clase "x'', es de clase "x''. El falso negativo (FN), ocurre cuando el modelo predice incorrectamente que un sujeto de la clase "x'' es de una clase que no es "x''. Finalmente, el verdadero negativo (TN) ocurre cuando el modelo predice correctamente que un sujeto no es de una clase "x''.

La elección de la métrica adecuada depende directamente del tipo de modelo y de cómo se implementa, por ejemplo, en cuanto a problemas

de clasificación y regresión las métricas anteriores son las más utilizadas.

Aquí, la exactitud (2.7) es la proporción de sujetos clasificados correctamente entre la población total, F_1 Score (2.10) es la media armónica de precisión y sensibilidad. El valor ideal de estas métricas es 1 y el objetivo de los modelos de aprendizaje automático es obtener un valor lo más cercano al 1.

2.3.1. SVM

La Máquina de Soporte Vectorial (SVM), por sus siglas en inglés, se desarrolló en la década de los 90 por Vladimir Vapnik [39], originalmente se desarrolló como un método de clasificación binaria, posteriormente, su aplicación se ha generalizado a problemas de clasificación múltiple y ha resultado ser uno de los mejores clasificadores para un conjunto muy grande de situaciones, llegando a ser una de las técnicas más importantes, actualmente es un referente en el ámbito del Machine Learning.
Es muy utilizado modelos de Deep Learning y tiene las siguientes ventajas:

- Efectividad en espacios de dimensiones altas.
- Es efectivo aún en casos en los que la dimensión es más grande que el número de datos.
- Versátil, pues se pueden especificar diferentes funciones del Kernel para la función de decisión.

El SVM consiste en suponer que es posible construir un hiperplano que separa las clases de datos perfectamente con sus etiquetas correspondientes, cuando se tiene un conjunto de datos perfectamente separables linealmente es posible hallar un número infinito de posibles hiperplanos de separación, como se muestra a continuación:

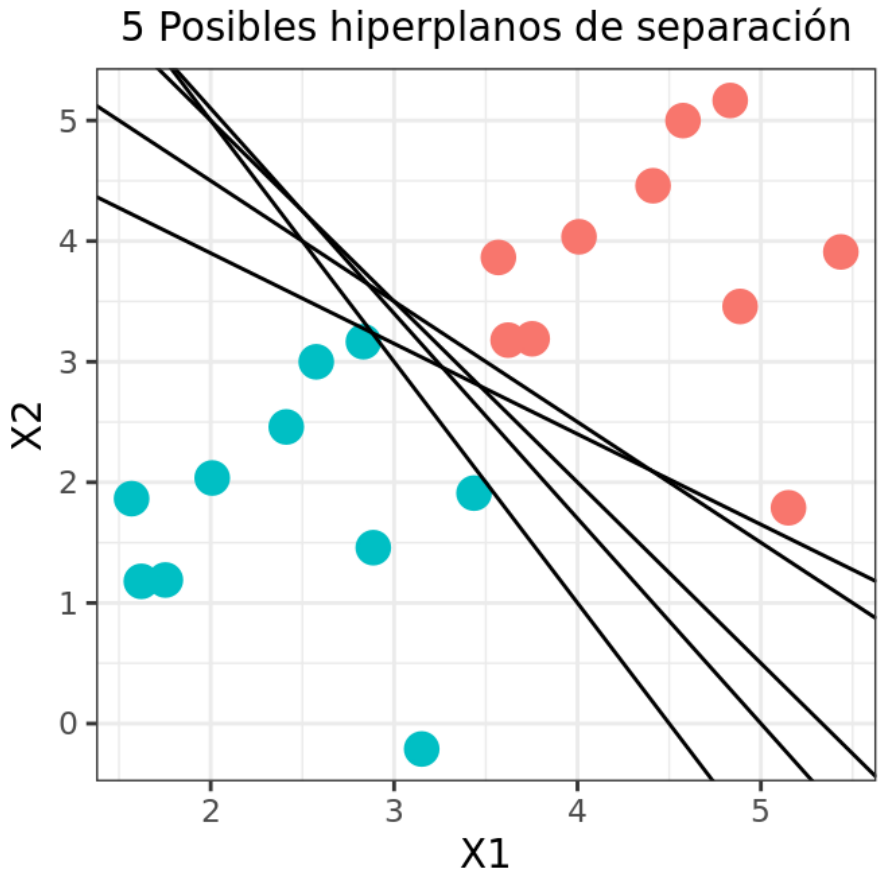

Figura 2.3: Hiperplanos de separación, todos estos planos separan correctamente las clases de datos. [9]

En este sentido, es necesario un método que permita seleccionar uno de ellos como el hiperplano de separación óptimo y la propuesta de Vapnik [39] es hallar aquél que maximice una distancia (margen) entre los elementos más cercanos de amabas clases.

Gráficamente:

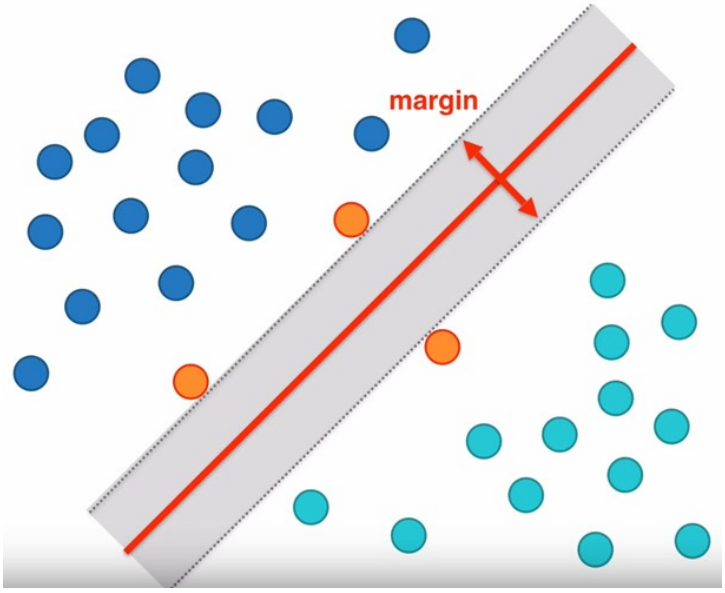

Figura 2.4: Distancia entre los elementos de ambas clases, el hiperplano óptimo está a la mitad del margen, el cual es la distancia perpendicular entre dos hiperplanos paralelos al central, estos dos planos (líneas grises) son los más próximos a las respectivas clases de datos. [40]

El trazo del hiperplano óptimo (línea roja), se debe hacer maximizando el margen, de manera formal se hace de la siguiente manera:

Sea $\mathbf{x} \in \Re^n$ el espacio de características y sea $\mathbf{w} \in \Re^n$ el espacio de parámetros del sistema, es posible determinar un hiperplano de decisión representado por

$$\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = \mathbf{0} \tag{2.11}$$

Para el caso bidimiensional:

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{2.12}$$

Se puede entonces establecer una frontera de decisión de la siguiente

forma:

$$wx + b > 0, \qquad wx + b < 0 \tag{2.13}$$

lo cual significa que si tenemos un par coordenado cualquiera (w_1, w_2) y lo hacemos pasar por la ecuación del plano (2.12) nos dará > 0 para aquellos elementos que están del lado derecho del hiperplano y nos dará < 0 para los que estén a la izquierda del hiperplano. Más aún, se puede reescribir la ecuación (2.13) para cada hiperplano que pasa por los elementos más próximos a las clases (llamados vectores de soporte), quedando de la siguiente forma:

$$wx_+ + b \geqslant 1, \qquad wx_- + b \leqslant -1 \tag{2.14}$$

De esta manera, la ecuación de la izquierda en (2.14) establece el criterio para la clase 1 y la ecuación de la derecha establece el criterio para la clase -1, asegurándonos de que ningún elemento se encuentre dentro del margen. Al maximizar el margen se eliminan los problemas de generalización que son muy comunes y que muchas veces conducen a sobreajustar el modelo.
Equivalentemente a las ecuaciones (2.14), se pueden distinguir las clases -1 y 1 de la siguiente forma:

$$y_i(\mathbf{w} \cdot \mathbf{x}_+ + b) = 1 \qquad y_i(\mathbf{w} \cdot \mathbf{x}_- + b) = -1 \tag{2.15}$$

donde claramente se observa que la ecuación de la izquierda corresponde a la clase 1 cuando $\mathbf{w} \cdot \mathbf{x} + b > 0$ y la ecuación de la derecha corresponde a los datos de la clase -1, es decir, cuando $\mathbf{w} \cdot \mathbf{x} + b < 0$.
Para los vectores de soporte se cumple lo siguiente:

$$y_i(\mathbf{w} \cdot \mathbf{x}_\pm + b) - 1 = 0 \tag{2.16}$$

pues se encuentran sobre los planos auxiliares.
Dado que la idea de Vapnik [39] es conseguir que la longitud del margen se maximice, se pretende primero, encontrar la longitud de dicho margen, para ello, se evalúa la proyección del vector de soporte x_- sobre el vector del plano óptimo w, esto se puede ver en la siguiente gráfica:

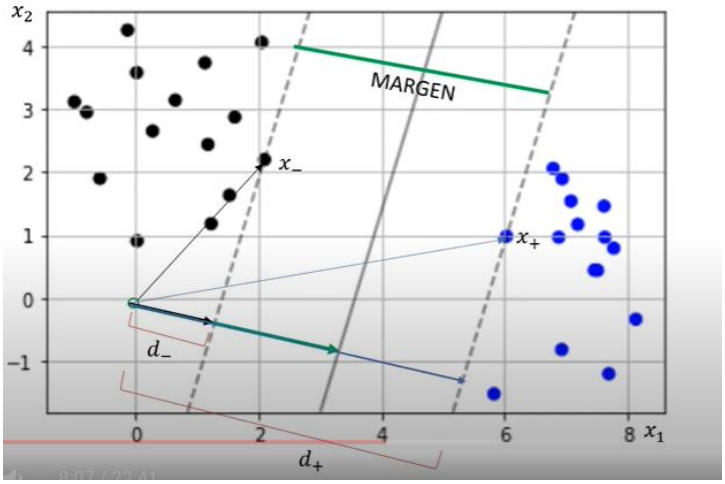

Figura 2.5: d_- es la proyección del vector x_- sobre w [18]

En la figura anterior se ve la distancia d_- que se obtiene con la siguiente expresión:

$$d_- = \frac{\mathbf{x}_- \cdot \mathbf{w}}{||\mathbf{w}||} \tag{2.17}$$

También se puede hacer lo mismo para el vector de soporte que está a la derecha del plano óptimo, es decir el vector x_+:

$$d_+ = \frac{\mathbf{x}_+ \cdot \mathbf{w}}{||\mathbf{w}||} \tag{2.18}$$

En la gráfica: d_- es la línea negra que se calcula con la expresión (2.17); d_+ es la línea azul que se calcula con la expresión (2.18) y con esto es posible calcular fácilmente la distancia del margen, que es precisamente $d_+ - d_-$, es decir, restando únicamente la parte negra de la línea azul en la gráfica.
Así,

$$d = d_+ - d_- = \frac{1}{||\mathbf{w}||}(\mathbf{x}_+ \cdot \mathbf{w} - \mathbf{x}_- \cdot \mathbf{w}) \tag{2.19}$$

Además, para los vectores de soporte:

$$d = d_+ - d_- = \frac{1}{||\mathbf{w}||}(\mathbf{x}_+ \cdot \mathbf{w} - \mathbf{x}_- \cdot \mathbf{w}) \tag{2.20}$$

y por la ecuación (2.16):

$$d = \frac{1}{||\mathbf{w}||}[(1+b) - (-1+b)] = \frac{2}{||\mathbf{w}||} \tag{2.21}$$

Luego, se debe buscar maximizar esta distancia, que se puede escribir como sigue:

$$d = \frac{2}{\mathbf{w^t w}} \tag{2.22}$$

o bien, esta expresión corresponde a minimizar el inverso, es decir:

$$d = \frac{1}{2}\mathbf{w^t w} \tag{2.23}$$

esto significa que minimizando (2.23), se maximiza (2.22).

Para hacer la minimización, se debe considerar que el problema está sujeto a las restricciones (2.16), lo cual significa que los vectores de soporte de ambas clases deben estar forzosamente en los dos planos auxiliares respectivos.
Mediante multiplicadores de lagrange:

$$f(w) = \frac{1}{2}w^t w \qquad (funcion \quad objetivo)$$

$$g(w,b) = y_i(w \cdot x_\pm + b) - 1 = 0 \qquad (restricciones)$$

con esta función objetivo a minimizar y con las restricciones, se puede establecer un nuevo problema de minimización, pero sin restricciones, utilizando el método de los multiplicadores de Lagrange.
El lagrangiano correspondiente a este problema es una función de los parámetros w, b y de los coeficientes de lagrange α_i:

$$L(w,b,\alpha_i) = \frac{1}{2}w^t w - \sum_{i=1}^{l} \alpha_i[y_i(w \cdot x_i + b) - 1] \tag{2.24}$$

donde l es el número de vectores de soporte, que pueden ser obviamente más de dos.

Derivando e igualando a cero:

$$\frac{\partial L}{\partial W} = 0 \quad \Rightarrow w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0$$

Reemplazando estas dos ecuaciones en (2.24), se obtiene:

$$L(\alpha_i) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{2.25}$$

Notemos que aquí el lagrangiano ya no depende de los parámetros w y b, sólo de los coeficientes de Lagrange y del producto escalar de los vectores de soporte. Esto es válido únicamente para cuando se tienen clases linealmente separables, es decir que se encuentra un hiperplano que separa a las clases de datos; por otro lado, cuando los datos no son linealmente separables, se modifica el lagrangiano (2.25) con un kernel K

$$L(\alpha_i) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \tag{2.26}$$

este kernel puede ser personalizado para cada tipo de problema, sin embargo, los más utilizados además del lineal, son el polinomial y el gaussiano [24] , los cuales funcionan cuando es imposible encontrar un hiperplano que separe las clases de datos (que son linealmente no separables). Este procedimiento consiste en aumentar la dimensionalidad y convertir un sistema linealmente no separable en un sistema de clases linealmente separables, como se muestra a continuación:

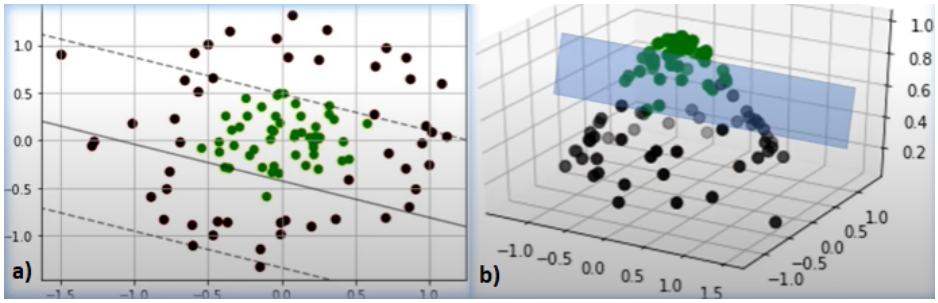

Figura 2.6: a) Datos linealmente no separables en $\Re^2$, se muestra el hiperplano de separación que no es capaz de separar las clases. b) se aumenta la dimensión a $\Re^3$, ahora es posible hallar un hiperplano para que las clases se vuelvan linealmente separables. [18]

Los kernels hacen que los SVM´s sean más flexibles y capaces de gestionar problemas no lineales, como se ve en la figura 2.6, un problema no separable en $\Re^2$ se vuelve linealmente separable en $\Re^3$.

2.3.2. Redes neuronales

Redes Neuronales Biológicas

Las redes neuronales biológicas se caracterizan principalmente por tener una inclinación natural a adquirir el conocimiento a través de la experiencia, se componen por conjuntos muy grandes de células llamadas neuronas cuya función es procesar y recibir información mediante impulsos; tienen la siguiente estructura:

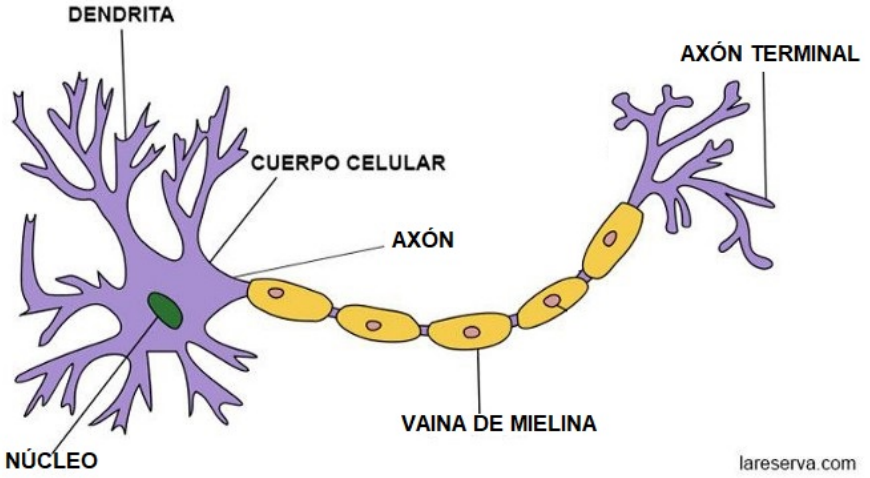

Figura 2.7: Neurona biológica [41]

Como se puede ver en la imagen, las neuronas biológicas constan de varias partes, empezando por las dendritas que son prolongaciones cortas muy ramificadas, funcionan como antenas que reciben información del medio y de otras neuronas para llevarla al núcleo donde se integra, generando un impulso nervioso. El axón es una prolongación larga a través de la cual viaja el impulso generado para llegar a los botones sinápticos, en donde la información llega a otra célula haciendo la "sinapsis".

Las neuronas biológicas se dividen en dos grupos: simples y complejas, las neuronas simples solo ven una pequeña parte del campo visual y

para tener un campo visual amplio es necesario reunir muchas de estas (cada una cubre un pedacito hasta tener el campo visual completo), cada una se activa cuando detecta ejes o líneas orientadas de cierta forma en la pequeña parte que ve, por otro lado, las neuronas complejas agrupan lo que ven varias neuronas simples, es decir, su campo receptivo es más grande, se activan también con los ejes sin importar la posición o dirección que tengan.
Así, las neuronas simples ven ejes en una pequeña región de cada imagen que una persona observa, las neuronas complejas, toman información desordenada de una imagen (como un rompecabezas) y la agrupan en conceptos más concretos como figuras (hasta este punto aún no se distingue nada), el cerebro agrega más capas de neuronas simples y complejas, uniendo figuras en cosas que se complementan hasta finalmente agrupar todo, es un proceso muy eficiente donde no importa la posición inicial de la imagen.

Redes Neuronales Artificiales

En 1943 McCulloch y Pitts inspirados en la neurona biológica desarrollaron un modelo artificial de una neurona [42]. Esta idea fue retomada más tarde por Rosenblatt quien creó una estructura llamada perceptrón [43].

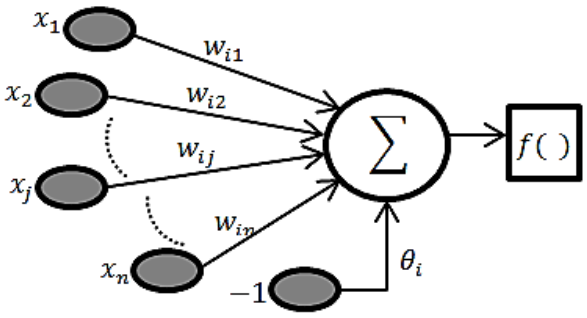

Figura 2.8: Neurona artificial [41]

El perceptrón es una estructura la cual posee una o varias entradas,

una función de activación y una salida, se trata de la aproximación más cercana a una neurona biológica. Las entradas del perceptrón actúan como las dendritas o señales de entrada hacia la neurona. La función de activación realiza el papel del núcleo que decide en qué momento disparar esa señal eléctrica y la salida modela al axón, que al activarse envía información hacia las neuronas con las que se conecta.

A pesar de que no se conoce exactamente cómo funciona el cerebro humano, se han estudiado ciertas características como la estructura que tiene y la forma en la que las neuronas biológicas procesan la información, así es como surgen estos algoritmos; concretamente el algoritmo consiste en un conjunto de unidades denominadas neuronas conectadas entre sí para transmitirse información de unas a otras. Cuando los datos de entrada atraviesan esta red de neuronas, se acaba produciendo un resultado o unos valores de salida que son los que nos interesan. De este modo se puede entender una red neuronal como una gran función que produce una salida dependiente de unos datos de entrada, pero dicha función para ello se apoya en una multitud de subfunciones que la componen. Es decir, se puede ver una red neuronal como una función de funciones que permiten a la función principal calcular y capturar relaciones en los datos.
El objetivo final es que la función de funciones acabe realizando predicciones correctas para los datos de entrada (la salida obtenida por la red sea la esperada). Por tanto, se trata de un algoritmo de aprendizaje supervisado en el que se necesita una función que se adapte a unos datos previamente etiquetados (recibe una serie de salidas esperadas para unas entradas determinadas y necesita ajustarse para producir esas mismas salidas).
Estos algoritmos, no habían sido tan populares hasta ahora debido a la gran cantidad de recursos computacionales que requieren y de la enorme cantidad de datos de la que dependen para su correcto funcionamiento. Hoy en día son consideradas como una de las mejores técnicas para muchas aplicaciones entre las cuales destacan la visión por computadora, el procesamiento del lenguaje natural y sobre todo la clasificación de imágenes.
Una red neuronal simple consta de una sola neurona, se puede entender

simplemente como un modelo de clasificación, sin embargo, cuando la red dispone de más de una neurona el modelo se empieza a complicar. Por otra parte, la salida de la red ya no es la salida de la única neurona que se tenía, pues al haber más de una, necesitamos otra neurona que calcule la salida en base a las salidas de las demás neuronas. Es decir, se ha añadido una neurona que está conectada al resto de neuronas de la red. Se dice que dicha neurona pertenece a otra capa de la misma red, porque necesita tomar como entrada lo que las demás arrojan como salida, siendo un proceso secuencial de capa en capa.
En las Redes Neuronales cada neurona actúa como un clasificador con sus propios parámetros y la decisión de que se arroje, una salida u otra, depende de una función de activación.
Así, cuando se tienen muchas neuronas dispuestas en capas, esta red debe ser entrenada en su conjunto y no de manera independiente.
Una red neuronal puede ser muy compleja, el nivel de complejidad depende directamente de capas y de neuronas por capa que tenga. Se sabe que cuantas más capas disponga la red y más neuronas tenga, mayor será su capacidad para aprender relaciones complejas en datos, pero a cambio mayor será también la dificultad y complejidad de la red. A estas redes neuronales complejas (que tienen varias capas intermedias) se les engloba en un nuevo marco del mundo del aprendizaje autónomo denominado Deep Learning [44].
Para entrenar bien estas redes tan profundas se necesitan muchos datos y técnicas específicas. El entrenamiento de estas redes es un proceso costoso tanto en tiempo como en recursos computacionales, pero se ha comprobado que el uso de éstas tiene múltiples beneficios, sobre todo en aplicaciones complejas como el reconocimiento de voz, reconocimiento facial, en los autos autónomos, detección de fraudes bancarios y muchas otras [8].
Una de las dificultades que se presenta a la hora de entrenar la red es que no se pueden actualizar todos los parámetros del modelo a la vez (habiendo muchos parámetros), se debe ir recorriendo capa a capa actualizando poco a poco los pesos, buscando que el gradiente de la función de error con respecto a cada peso sea cero (mínimo error en ese punto) y para ello se aplica la regla de la cadena, que nos permite

aprovechar cálculos de gradientes anteriores para calcular los actuales; si recorremos la red en sentido contrario (desde las capas más profundas a las primeras capas) a esto se le llama Backpropagation [45].

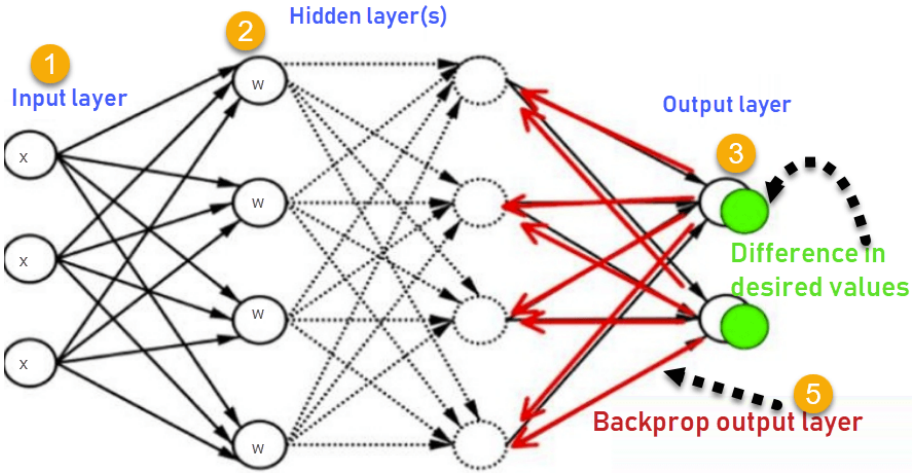

Figura 2.9: Backpropagation, proceso de actualización de los pesos.[48]

La importancia de este proceso radica en que a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas permitiendo que las distintas neuronas aprenden a reconocer las diferentes características de todo el conjunto de entrada y después del entrenamiento, cuando se les presenta un patrón arbitrario de entrada que contenga ruido o que está incompleto, las neuronas de la capa oculta son capaces de responder con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales aprendieron a reconocer durante la etapa de entrenamiento.

Redes Neuronales Convolucionales (CNN).

Las redes neuronales convolucionales (CNN) por sus siglas en inglés, son un tipo de redes que surgen debido a la importancia de la visión por computadora en el mundo del Deep Learning, utilizan una operación llamada Convolución, de ahí el nombre de estas redes.

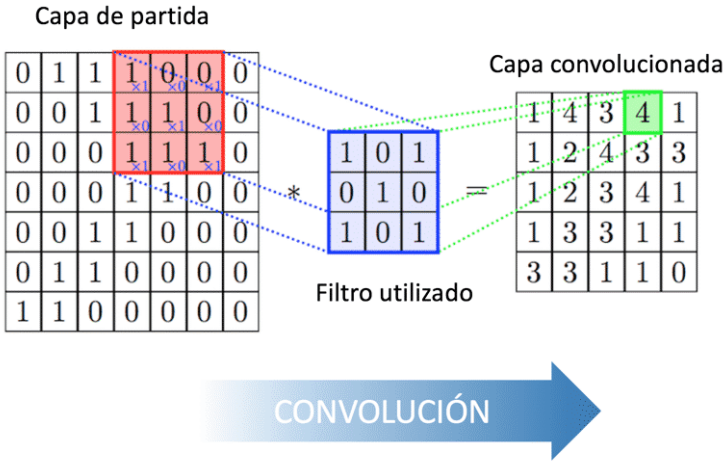

Figura 2.10: Convolución en redes neuronales. [49]

Como se puede ver en la imagen, una convolución es una operación entre matrices (una imagen es computacionalmente una matriz de datos numéricos), consiste en superponer un filtro sobre una imagen e ir deslizándolo sobre la misma hasta haber recorrido todas las regiones de la imagen (todos los píxeles de la imagen deben ser "visitados" por el filtro en algún momento durante el proceso). Por cada desplazamiento del filtro sobre la imagen se calcula el valor resultante de la suma de todas las multiplicaciones de cada pixel por el valor correspondiente a la posición del filtro que superpone a ese pixel (entrada por entrada). De esta forma se construye el mapa de características (capa convolucionada), cada vez que se calcula un nuevo resultado se almacena en la siguiente posición de la capa convolucionada. Así, este tipo de redes se caracterizan porque tienen capas convolucionales y capas Densas comúnmente llamadas Fully Conected, las cuales se definen porque las neuronas de

una capa oculta están conectadas a todas las neuronas de la capa que les precede y sucede. Esto implica que cada una de las características de entrada está conectada a cada neurona de la capa siguiente; para el caso de las imágenes, significa que cada pixel de la imagen de entrada está conectado a cada neurona de la primera capa oculta, en otras palabras, se encargan de realizar un proceso que recibe pixeles de una imagen, extrae características de dicha imagen y clasifica mediante capas posteriores en la red.

Esto no es un trabajo sencillo cuando se trabaja con imágenes grandes de alta resolución (y cada vez mayor). Es por eso que se pretende crear modelos de redes neuronales convolucionales cada vez más complejos que consten de diversas capas convolucionales y de pooling (agrupaciones) aternadas ya que por sí solas estas redes van aprendiendo qué características conviene identificar y trabajar en las imágenes y lo hacen al principio en una fase conocida como fase de entrenamiento, mismo que va a proveer a la red de experiencia con los datos que esté trabajando y de este modo llegar a su objetivo principal que consiste en encontrar los parámetros (pesos) que van asociados siempre a una conexión concreta (cada uno a una neurona) acabando al final con una serie de capas densas.

La entrada de una capa convolucional suele ser una imagen de un cierto tamaño, las capas convolucionales suelen constar de más de un filtro y cada filtro genera mediante la operación de convolución un nuevo mapa de características bidimensional. El total de mapas de características generados a la salida de la capa será igual al número de filtros de la capa, por lo que podemos ver la salida en su conjunto como una nueva imagen cuyo número de canales es igual al número de filtros de la capa. Esta nueva imagen pasará a ser la entrada de la siguiente capa después de que se le haya sumado un término independiente o bias (sesgo) y haber pasado por una función de no linealidad que dará lugar a lo que llamamos mapa de activaciones. Dicha función de activación debe ser elegida según el problema que se tenga, por ejemplo, para el caso de problemas de clasificación binaria, se utiliza la función de activación Sigmoide:

$$f(x) = \frac{1}{1 - e^{-x}} \tag{2.27}$$

esta función transforma los valores introducidos a una escala del 0 al 1, donde los valores más grandes tienen de manera asintótica a 1 y los valores bajos tienden de manera asintótica a 0, de esta forma se puede distinguir entre dos clases.

Adicionalmente se suele introducir una capa de pooling tras la convolución ("max pooling" o "average pooling"), consiste en una operación similar a la convolución donde se hace pasar un filtro (normalmente de tamaño $2x2$) sobre las salidas de la capa convolucionada, extrayendo el pixel más grande de cada región visitada por el filtro para el caso de la capa de max pooling; para una capa de average pooling el proceso es similar con la diferencia de tomar el promedio de todos los pixeles bajo el filtro, tras hacer este proceso iterativo, se guardan los resultados como salidas de esta capa.
Finalmente, como la salida ha de ser una clasificación (probabilidades para cada clase), se incorporan capas fully connected, capaces de realizar dicha labor. En la siguiente imagen se puede ver cómo funciona una red convolucional.

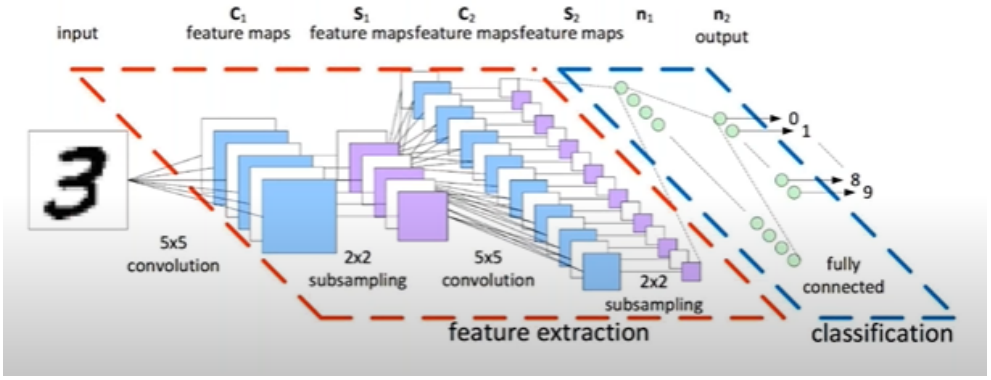

Figura 2.11: Funcionamiento general de una CNN [27].

El gran éxito de estas redes es su gran capacidad a la hora de identificar características en las imágenes de entrada. Se ha demostrado con la experiencia que son muy buenas detectando características cada vez más y más complejas. Esto se debe a que los filtros de las capas menos profundas actúan como detectores de rasgos pequeños como bordes o

texturas y los filtros de las capas más profundas son capaces de detectar estructuras cada vez más y más complejas como formas u objetos a partir de los resultados de las detecciones previas.

Con esto, la estructura de las CNN parece clara, cuando ya se tiene el modelo entrenado, se puede evaluar su rendimiento a partir de la función de pérdida.

Función de pérdida

La función de pérdida es una medida del error de la red neuronal, de modo que entre más grande es el error, mayor es el valor de la función de pérdida. Existen dos tipos principales de funciones de costo para el aprendizaje supervisado de una red neuronal, sin embargo, la función de pérdida que se utiliza en clasificación binaria es la de entropía cruzada binaria cuyo objetivo es devolver valores altos para malas predicciones y devolver valores bajos para buenas predicciones, esta función también es llamada pérdida logarítmica y se define de la siguiente forma:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \qquad (2.28)$$

donde "y'' es la etiqueta de la clase (1 ó 0), $p(y_i)$ es la probabilidad predicha de que el dato $i - esimo$ pertenezca a la clase 1. En otras palabras, lo que la función (2.28) significa es que: cada dato y_i cuya etiqueta es igual a 1, suma la cantidad de $log(p(y_i))$ a la función de pérdida, o sea, la probabilidad logarítmica de que pertenezca a la clase 1, por el contrario, suma $(1 - p(y_i))$, i.e., la probabilidad logarítmica de que sea de clase 0 por cada dato de la clase 0.

Para minimizar el valor de la función de pérdida se han desarrollado diferentes optimizadores, con el objetivo de generar modelos que produzcan mejores y más rápidos resultados a través de la elección de los hiperparámetros. Existen funciones que permiten evitar estancamiento en mínimos locales y funcionan dependiendo del problema que se estudie, entre los más conocidos se encuentran el llamado estimación del

momento adaptativo (Adam) y propagación cuadrática media (RMSprop).

Optimizador ADAM

El algoritmo de optimización Adam se basa en estimaciones adaptativas de los momentos de orden más bajo, de tal forma que solo se requieren los gradientes orden uno. El método consiste en calcular los rangos de aprendizaje adaptativos individuales para diferentes parámetros, desde las estimaciones de los primeros y segundos momentos de los gradientes [30]. El algoritmo conserva un decaimiento exponencial promedio de los gradientes pasados m_t similar al momento:

$$m_t = \beta_1 m_{t-1}(1 - \beta_1)g_t \tag{2.29}$$

$$v_t = \beta_2 v_{t-1}(1 - \beta_2)g_t^2 \tag{2.30}$$

donde m_t y v_t son estimaciones del primer momento, y el segundo momento, es decir el promedio y la varianza de los gradientes, respectivamente. Como m_t y v_t se inicializan en cero y por tanto ambos tienen sesgo hacia el cero, especialmente durante los primeros pasos y cuando los rangos de decaimiento son pequeños (cuando β_1 y β_2 son cercanos a 1).
La corrección de los sesgos se hace calculando las correcciones de las estimaciones del primero y segundo momento como se muestra a continuación:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.31}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.32}$$

y aplicando la regla del método Adam para ajustar:

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.33}$$

Los valores que normalmente se consideran para estos parámetros son:

$$\beta_1 = 0.9, \qquad \beta_2 = 0.999, \qquad \epsilon = 1x10^{-8}$$

El método Adam tiene buenos resultados en aplicaciones prácticas, resulta ser el método más utilizado y se puede obtener un resultado estimado con relativa rapidez [30], es computacionalmente eficiente, tiene pocos requerimientos de memoria y es adecuado para problemas que manejan grandes cantidades parámetros.

Optimizador RMSprop

El algoritmo RMSprop al igual que el Adam, es un método de rangos de aprendizaje adaptativos, surge de la necesidad de resolver el problema de la disminución de la tasa de aprendizaje radical [29]. Este algoritmo utiliza un concepto de ventana para considerar únicamente los gradientes más recientes. El vector de ajuste se calcula de la siguiente forma:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.34}$$

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.35}$$

donde E es el promedio, g la función gradiente y γ representa la influencia del momento cuyo valor normalmente se fija en $\gamma = 0.9$ mientras que el valor de la tasa de aprendizaje α se puede variar, ya sea $\alpha = 0.01$ ó $\alpha = 0.001$, incluso $\alpha = 0.0001$. Podemos ver entonces que el RMSprop divide al rango de aprendizaje por un decaimiento exponencial promedio de los gradientes cuadrados.

2.3.3. Transfer Learning

Entrenar una arquitectura de red neuronal profunda desde cero es difícil en la práctica debido a que contiene una gran cantidad de pesos que se inicializan aleatoriamente antes del entrenamiento y se actualizan iterativamente de acuerdo con los datos etiquetados. La actualización iterativa de los pesos lleva mucho tiempo y, debido a que muchas veces

los datos de entrenamiento son limitados, la arquitectura profunda se puede sobreajustar. La utilización del Transfer Learning resuelve muchos de estos problemas [25].

Se trata de un enfoque conocido de aprendizaje profundo destinado a reutilizar un modelo desarrollado para una tarea como punto de partida para un modelo en otra tarea, es decir, que se vale de la experiencia aprendida en la realización de una tarea para llevar a cabo otras, este método permite abordar problemas con relativamente pocos datos [28] Los métodos de TL se pueden dividir en cuatro categorías según en lo que estén basados: en casos, en características, en parámetros o en relaciones [26] De esta manera se puede hacer el entrenamiento de un modelo transfiriendo a éste los parámetros del modelo preentrenado. Para llevar a cabo el aprendizaje por transferencia de parámetros se toman los datos de entrenamiento como dominios, siendo $\mathbf{D_s}$ el dominio de origen y $\mathbf{D_t}$ el dominio de destino. Teniendo ya un modelo de aprendizaje profundo preentrenado en su dominio de origen, se transfieren los parámetros al dominio de destino, se vuelve a entrenar y se ajusta para lograr el objetivo de identificar muestras de etiquetas desconocidas en el dominio de destino, es como hacer una especie de pegado de modelos.
El uso del Transfer Learning se ha vuelto común, por ejemplo, los resultados experimentales de [28] muestran que los enfoques de TL superan el rendimiento de los enfoques basados en el aprendizaje sin TL, lo que demuestra la eficacia de estos enfoques para la tarea de clasificación binaria. Hay muchos otros algoritmos del mismo tipo que VGG como ResNet, Inception o incluso variantes de estos como Xception.

VGG-16

Los modelos VGG investigan la profundidad de las capas con un tamaño de filtro convolucional muy pequeño (3×3) para manejar imágenes a gran escala. Los autores lanzaron una serie de modelos VGG con diferentes longitudes de capa, de 11 a 19, siendo los modelos VGG−16 y VGG−19 los más conocidos. VGG es un acrónimo del nombre del grupo Visual Geometry Group de la Universidad de Oxford, se trata de mode-

los de redes neuronales convolucional profundas, propuestas por Karen Simonyan y Andrew Zisserman [47].
Con el modelo VGG−16, se obtuvo el segundo lugar en la competencia ILSVRC-2014, donde se logró un 92.7 % de desempeño en la clasificación de las imágenes del conjunto ImageNet, se trata de una base de datos que consta de 14,197,122 imágenes organizadas según la jerarquía de WordNet, siendo una iniciativa para ayudar a los investigadores, estudiantes y otras personas en el campo de la investigación de la imagen y la visión.

El modelo VGG−16 tiene la siguiente estructura:

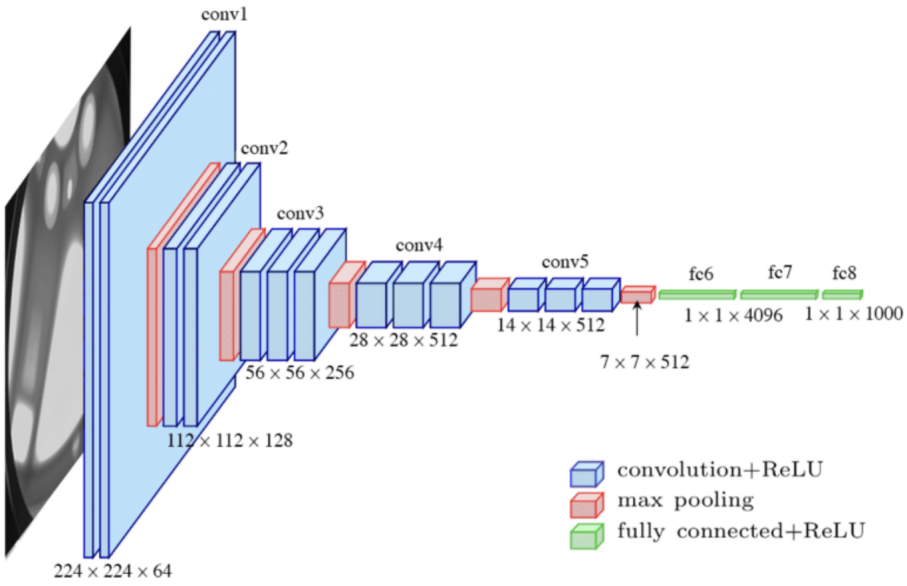

Figura 2.12: Arquitectura de la red VGG-16 [31]

Como se puede observar, las capas se disponen en bloques, el número 16 del modelo VGG-16 se debe a que hay 16 capas que tienen parámetros ajustables (13 capas convolucionales y 3 capas densas), en total tiene 13 capas convolucionales, 5 capas de agrupación máxima y 3 capas completamente conectadas. El número de filtros en el primer bloque es 64, luego este número se va duplicando en los bloques posteriores

hasta llegar a 512. Este modelo se completa con dos capas ocultas completamente conectadas y una capa de salida. Las dos capas totalmente conectadas tienen el mismo número de neuronas (4096). La capa de salida consta de 1000 neuronas correspondientes al número de categorías del conjunto de datos de ImageNet [46].

VGG-19

Al igual que el modelo anterior, la VGG-19 es una arquitectura de red neuronal convolucional similar, se trata de dos arquitecturas muy parecidas que siguen la misma lógica, la diferencia es que VGG19 tiene un mayor número de capas de convolución.

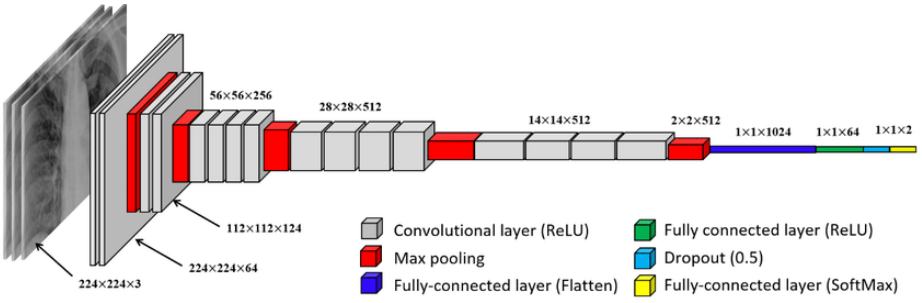

Figura 2.13: Arquitectura de la red $VGG-19$.[32]

Como en el modelo anterior, el nombre de VGG-19 también se debe al número de capas que tienen parámetros ajustables y es precisamente 19 (16 capas convolucionales y 3 capas densas) como se puede ver en la figura 2.13. El número de parámetros de esta red es 143.8 millones, inicialmente se pensaba que, a mayor número de parámetros, el modelo es más eficiente, por ser más complejo, pero no se puede generalizar para todos los problemas.

CAPÍTULO 3

Metodología

3.1. Protocolo de obtención de datos

Los datos con los que se realizó el trabajo son parte de un estudio completo entre el Hospital Infantil de México, Federico Gómez, la Benemérita Universidad Autónoma de Puebla (BUAP), la Universidad Autónoma Metropolitana, King´s College London, Westminster University, Newton Fand, BBSRC y CONACyT que consiste en analizar imágenes de resonancia magnética de cuerpo completo de niños en dos grupos: con obesidad y con normopeso, se trata únicamente de varones no adolescentes con edades de entre 7 y 9 años de todo México y que tienen un nivel socioeconómico promedio, para este trabajo solo se seleccionaron las imágenes de la secuencia de grasa (2.6) del hígado, en cada imagen se puede apreciar la misma zona en corte axial, disponiendo de un total de 167 imágenes.
Las imágenes tienen formato Dicom (.DCM) de tamaño 256x192 pixeles de las cuales se tomaron recortes de regiones de interés (ROI) por sus siglas en inglés, las cuales tienen un tamaño de $14x11$ y que comprenden precisamente la región del hígado para su mejor análisis, a continuación

se muestra la forma en que se recortaron dichas regiones de tal forma que se obtiene una ROI que se guardó en formato .jpg, por cada imagen hepática completa.

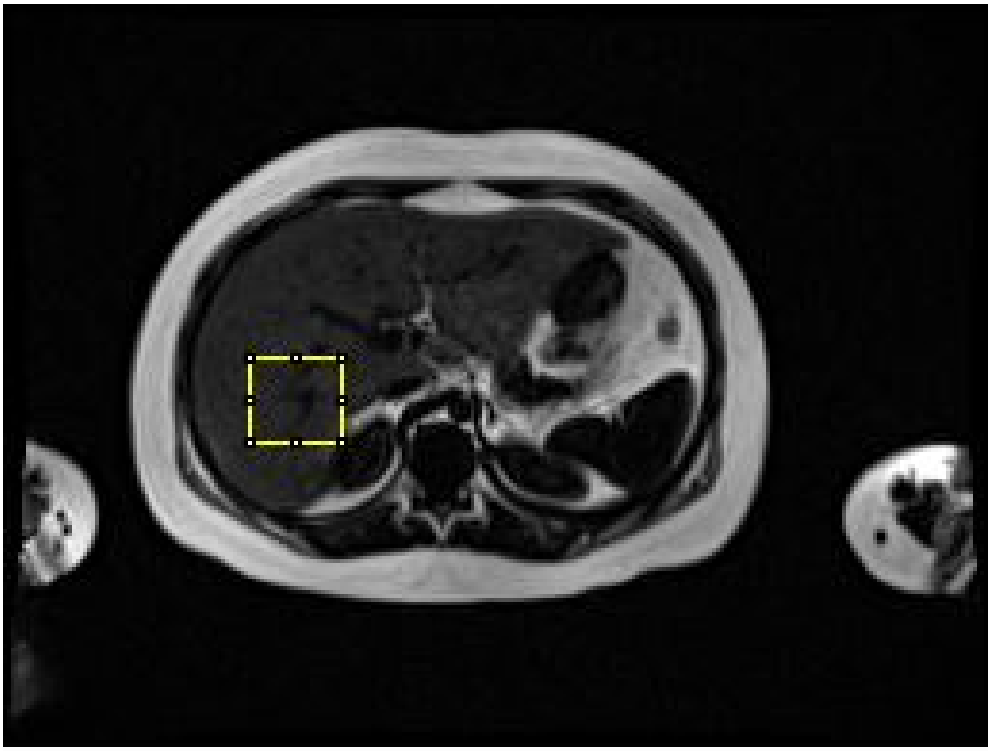

Figura 3.1: La región resaltada en color amarillo es la ROI, de tamaño $14x11$ pixeles por cada imagen completa.

En cuanto a la comparación entre imágenes completas y ROI´s, cabe señalar que las ROI´s corresponden en todo momento a la región del hígado, mientras que las imágenes completas contienen información adicional como la parte del tórax, la grasa corporal alrededor del cuerpo e incluso los brazos.

Las ROI´s se tomaron a partir de las imágenes completas con la ayuda del software "ImageJ" donde se estableció una región de interés de tamaño $14x11$ por cada imagen de resonancia magnética hepática axial

completa, se extrajeron dichas regiones de interés para posteriormente trabajar con ellas en la creación de clasificadores.

Para todos los análisis que se realizan, se utilizó el lenguaje de programación Python en notebooks de Colab, el cual es un producto de Google Research que está enfocado en tareas de aprendizaje automático y análisis de datos. Se utilizó una GPU que Google Colab ofrece. Para leer las imágenes médicas, se utilizó Pydicom, un paquete de Python que permite analizar archivos DICOM, pydicom facilita la lectura de estos archivos complejos en estructuras de fácil manipulación.

3.2. Análisis descriptivo

Las imágenes de resonancia magnética de los hígados de niños con obesidad ponderadas en grasa en principio deberían estar más claras que las imágenes de niños con normopeso, surge así la idea de buscar una forma de clasificar las imágenes a partir de esta característica, empezando por hacer un análisis descriptivo para fortalecer la idea expuesta.
Inicialmente se cargaron los datos y se creó una tabla que contiene los datos de todas las regiones de interés cuyas columnas son: pixeles mínimos, pixeles promedio, pixeles máximos, desviación estándar y varianza, estas medidas corresponden a una medida por cada imagen; se realizó un análisis separando las clases en gráficas de caja y bigotes para cada parámetro; más aún, se realizó la misma comparativa para las imágenes completas en cuanto a cada parámetro y se resumieron todas las gráficas en una sola, se presentan las gráficas de las regiones de interés y sus análogas con imágenes completas para cada parámetro; ahí se puede ver claramente cómo se distribuyen los datos.

3.3. Clasificadores

Por intensidad promedio de pixeles

La implementación de este clasificador consistió inicialmente en crear una gráfica de la cantidad de imágenes contra intensidad promedio de

pixeles de las ROI´s, con el fin de ver qué tan bien se separan los datos con respecto a este parámetro y encontrar un valor dentro de las intensidades promedio de pixeles que permita hacer una separación. Se cargaron las ROI´s y se guardaron en una lista, se hizo el vector de etiquetas correspondiente a las ROI´s y se creó un ciclo para las imágenes, obteniendo el promedio (mean) de cada una de estas y guardándolas en una nueva lista. Todas las listas creadas contienen en la primera parte los datos del grupo normopeso y la parte siguiente corresponde al grupo de obesidad, es así como se hace un vector llamado *condicion*, de este modo se distinguen las clases dentro de las gráficas, poniendo en un color los primeros 85 valores normopeso y los otros 82 en otro color correspondientes a obesidad.
Se crea una gráfica en donde se puede ver cuántas imágenes comparten un valor de intensidad promedio de pixeles.
Se crea posteriormente el clasificador a través de la implementación de una función ($mean_classfier$) cuyos datos de entrada son: una imagen y un valor umbral ($threshold$); dicho umbral se escogió considerando el valor medio de la intensidad promedio de pixeles de todas las imágenes o bien un valor a partir del cual se pueda dar una separación adecuada de los datos con ayuda de la gráfica previamente hecha; el valor resultó ser **8** para ROI´s.
El clasificador distingue entre una ROI correspondiente a un niño con obesidad si el promedio de la intensidad de pixeles de la imagen está por encima del valor umbral. Evidentemente para las ROI´s con un valor de intensidad promedio por debajo del *threshold* corresponderá a un niño normopeso.
Las salidas de este clasificador se guardan en una nueva lista y se implementan las métricas de exactitud (*accuracy*) y $F1$ Score considerando las etiquetas reales y las salidas del clasificador.

Por conteo de pixeles arriba de un umbral de intensidad

Para la creación de este clasificador inicialmente se graficó la cantidad de imágenes contra cantidad de pixeles arriba de un valor umbral (tomado del clasificador anterior), con el fin de analizar si la separación de los datos se puede visualizar así.

Las ROI´s que tienen una cantidad de pixeles por arriba de un *threshold*, corresponden al grupo de obesidad, recordemos que estas regiones de interés tienen un tamaño de $14x11$, en total son 154 pixeles por cada una, la elección del umbral es simplemente $\frac{154}{2} = 77$ pixeles; otra cantidad que se consideró para este clasificador es la intensidad de pixeles **8**, de igual forma que en el clasificador anterior.
Cada imagen pasa por el clasificador, mismo que contará los pixeles dentro de la imagen que tienen una intensidad arriba de **8**, si esta cantidad de pixeles es mayor a **77**, entonces la ROI será clasificada en el grupo de obesidad, notemos que este clasificador requiere tres elementos, los cuales son; la imagen de la ROI, el valor de la intensidad umbral que se fijó como **8** y el valor de la cantidad de pixeles que se fijó como **77**.
Las salidas de este clasificador se guardaron en una lista y para el cálculo de las métricas se utilizaron dichas salidas y las etiquetas reales de las ROI´s.

Por contraste

Este clasificador surge de la idea de que haya una separación según el contraste de las imágenes. Se creó una gráfica al igual que en los clasificadores anteriores con el objetivo de visualizar los datos y ver si hay una separación evidente o no. Esta gráfica consistió en obtener el rango de valores de cada imagen, es decir, ($pixel\ maximo - pixel\ minimo$) y se separaron los datos por colores según la clase a la que pertenecen.
Se graficó cantidad de imágenes contra el contraste de las mismas.
Para este clasificador se requiere definir una cierta diferencia umbral, la cual es un contraste a partir del cual una ROI se considera en un grupo u otro, esta cantidad se fijó a partir de la gráfica con el valor de 9, a pesar de que antes y después de este valor hay datos correspondientes a ambos grupos.
Se hicieron pasar todas las ROI´s por este clasificador y se guardaron los resultados en una lista que posteriormente se utilizó junto con las etiquetas reales para calcular las métricas.

Con la finalidad de mejorar el rendimiento de los clasificadores se utilizaron las imágenes completas, se repitió el procedimiento hecho para

las ROI´s, haciendo las adaptaciones pertinentes, por ejemplo, para el primer clasificador se toma un "threshold" de intensidad promedio de 70,46, tomando en cuenta que este es el promedio de pixeles de todas las imágenes y se registraron todos los resultados de las métricas.

3.4. SVM

La construcción de este modelo SVM consistió en cargar las ROI´s, se hace el vector de etiquetas de las mismas, se divide el conjunto total en un conjunto de entrenamiento y otro de prueba, se carga el modelo cuyo kernel es lineal con un total de 1000 iteraciones, se compila el modelo para posteriormente dar lugar a las salidas llamadas y_pred.
Una vez teniendo el modelo SVM entrenado y habiendo generado las predicciones, se pasa a calcular las métricas de exactitud (2.7) y F1 Score (2.10).
Se realiza el mismo procedimiento para obtener resultados de las mismas métricas utilizando un kernel K gaussiano y uno polinomial de grado 2.

3.5. Redes neuronales

Como se menciona en la sección (2.3), para crear una red neuronal se necesitan muchos más datos de los que disponemos, es por eso que el Transfer Learning [28], permite crear modelos de redes neuronales para conjuntos de datos que son pequeños; se crearon dos modelos de redes neuronales que funcionan con imágenes completas debido a que los modelos de redes neuronales VGG que se usaron funcionan con imágenes de un tamaño específico ($224x224$ pixeles) y resulta que no es posible adaptar dichos modelos para que trabajen con las ROI´s, es decir únicamente es posible trabajar con el tamaño de las imágenes completas.

VGG-16

Se cargaron los datos para iniciar el preprocesamiento de los mismos, el cual consiste en convertir las imágenes en arreglos de numpy y posteriormente guardarlos en una lista. Se creó el vector de etiquetas correspondiente a los dos grupos (obesidad y normopeso). En seguida se separa el conjunto total de imágenes en conjunto de entrenamiento (80 %) y prueba (20 %). Se adaptan las dimensiones necesarias ya que el modelo VGG-16 funciona con imágenes en tres ejes. El conjunto de entrenamiento tiene 133 imágenes y el de prueba tiene 34.

Utilizando el Transfer Learning (sección 2.3.3), se carga el modelo VGG-16 de la biblioteca de Tensorflow, se recorta la última parte del modelo como se ve en la figura 2.12 y se agrega una capa de pooling (en este caso Average Pooling) para finalmente colocar la capa Densa que permita generar las salidas en dos clases; la estructura del modelo queda como se ve en la figura 3.2. Una vez que se definió el modelo, se entrena únicamente la capa final que fue la modificada, se verifica que el modelo a entrenar se queda prácticamente congelado (con excepción de la última capa), se compila el modelo usando el optimizador RMSprop (2.35), la función de pérdida de entropía cruzada (2.28) y la métrica de exactitud (2.7).

El modelo se entrenó en 500 épocas, se obtuvieron las gráficas tanto para la exactitud y pérdida, así como los valores finales de las mismas.

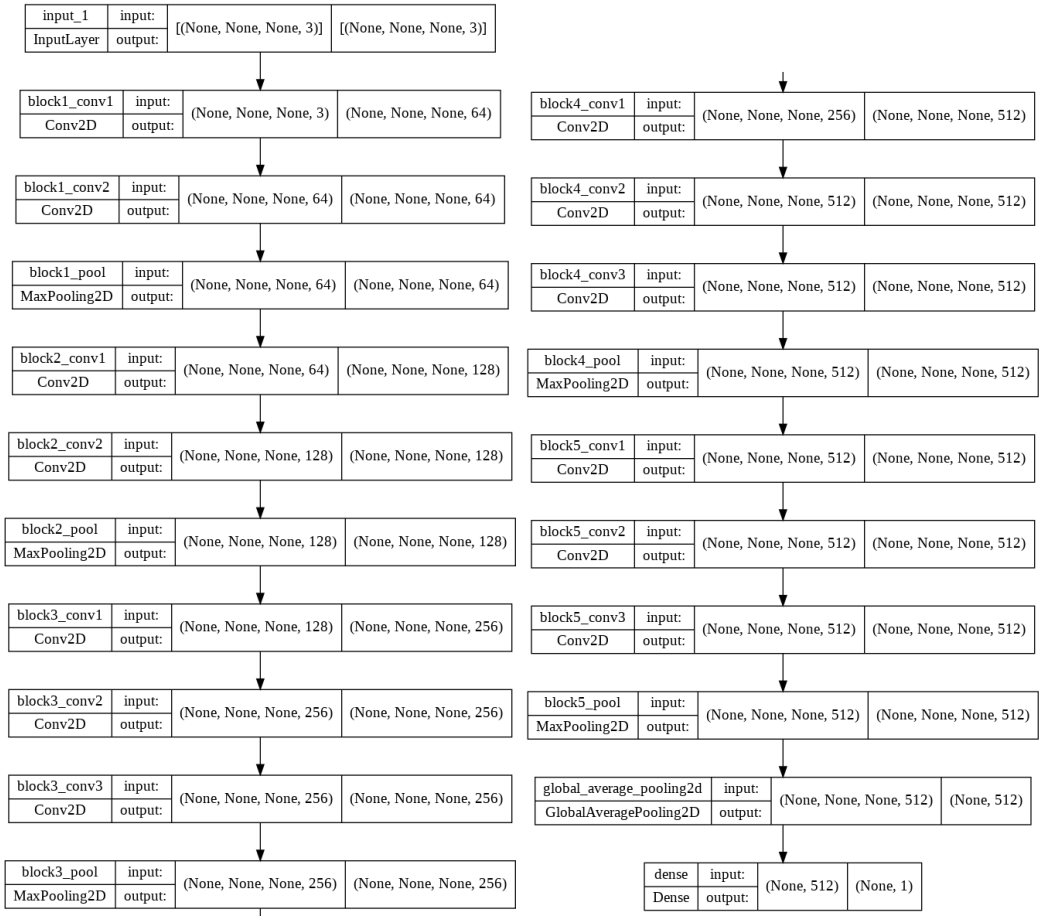

Figura 3.2: Arquitectura del modelo VGG-16 que clasifica MRI hepáticas en normopeso y obesidad.

VGG-19

La siguiente red utilizada es la VGG-19, el procedimiento es análogo al caso anterior (se entrena solo la capa final, se compila el modelo usando el optimizador RMSprop (2.35), la función de pérdida de entropía cruzada (2.28) y la métrica de exactitud (2.7)) y se obtiene la siguiente arquitectura:

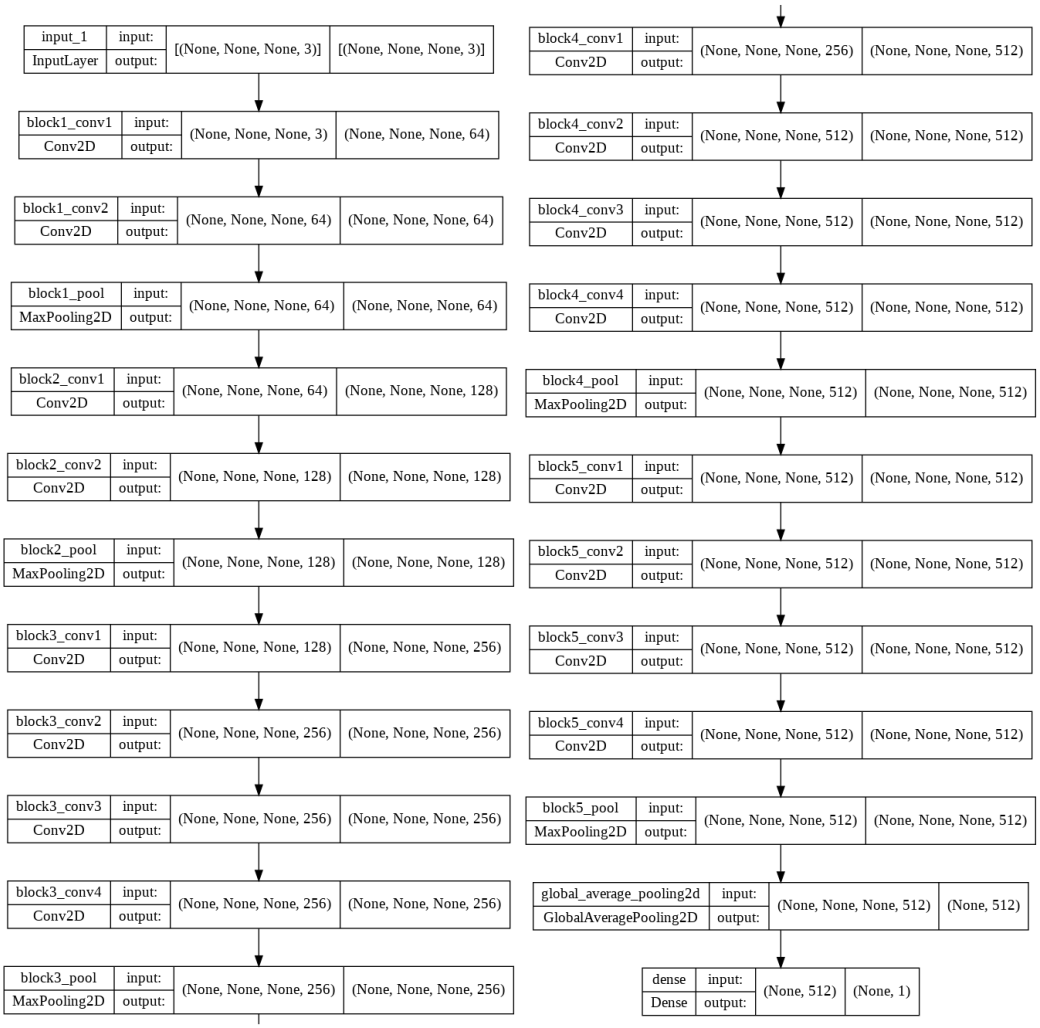

Figura 3.3: Arquitectura del modelo VGG-19 que clasifica MRI hepáticas en normopeso y obesidad.

El modelo se entrenó en 700 épocas, debido a que el entrenamiento con 500 épocas tenía tendencia a seguir disminuyendo, finalmente se obtuvieron las gráficas tanto para la exactitud y pérdida, así como los valores finales de las mismas.

CAPÍTULO 4

Resultados

4.1. Estadística Descriptiva

Se presenta la estadística descriptiva por clases del conjunto de datos en donde se pueden observar las diferencias en cuanto a pixeles máximo, promedio y mínimo de las imágenes correspondientes al grupo normopeso y obesidad, respectivamente; se observa también la diferencia entre la desviación estándar y varianza. Se empezó por analizar las ROI's y se obtuvo la siguiente información:

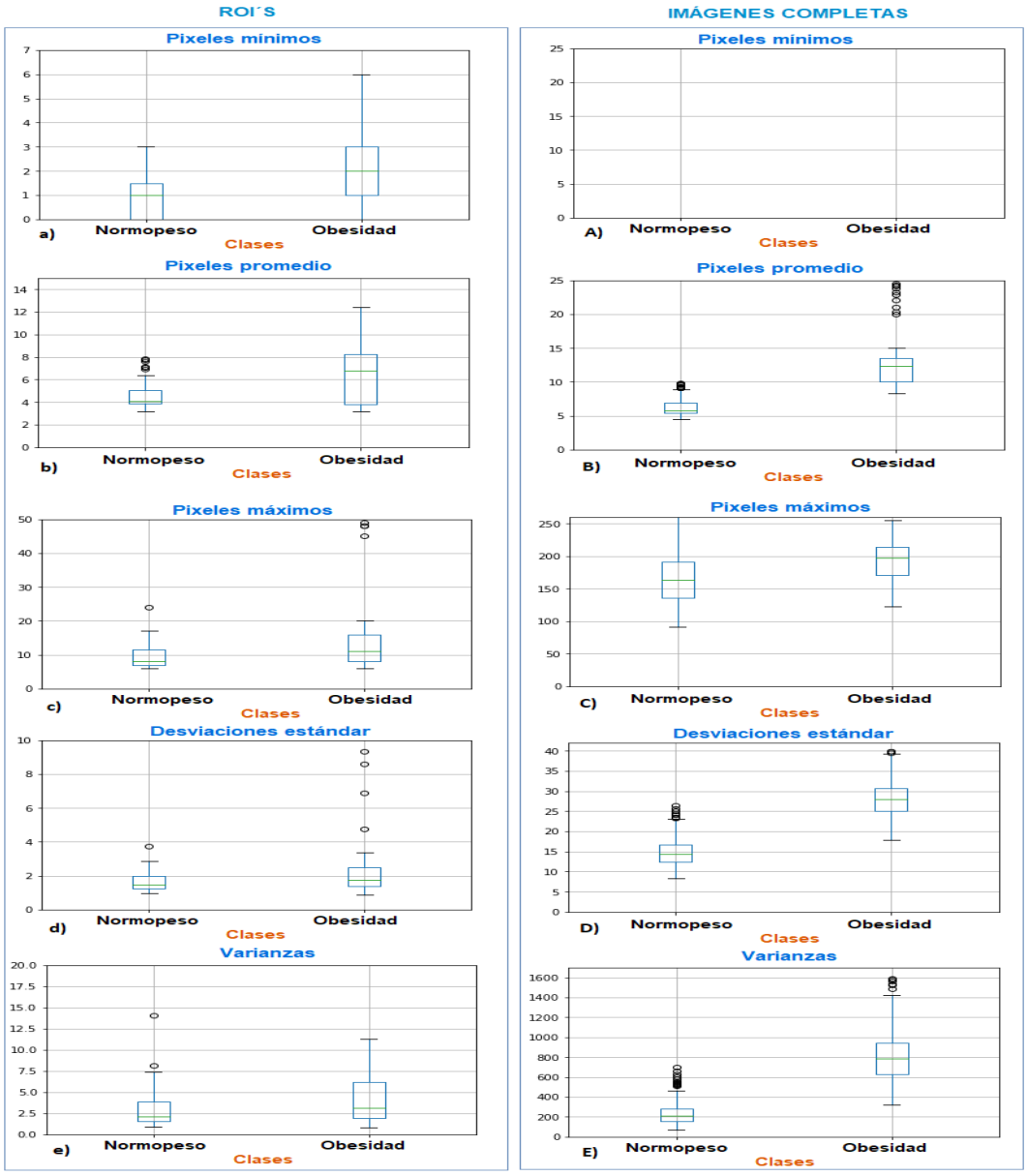

Figura 4.1: Análisis descriptivo de las clases.

Las gráficas de la izquierda (minúsculas) corresponden a las ROI's, mientras que las gráficas de la derecha (mayúsculas) corresponden a las imágenes completas, en ambas columnas se pueden ver diferentes parámetros que son indicadores descriptivos de los dos grupos bajo estudio, a partir de este análisis descriptivo propusieron los primeros tres clasificadores, en primera instancia con ROI´s, posteriormente con las imágenes completas.

4.2. Clasificadores

4.2.1. Por intensidad promedio de pixeles

Para este clasificador se grafican los promedios de los pixeles de cada imagen, en consecuencia, se podrá observar un cierto valor umbral a partir del cual puede darse la separación de las clases.

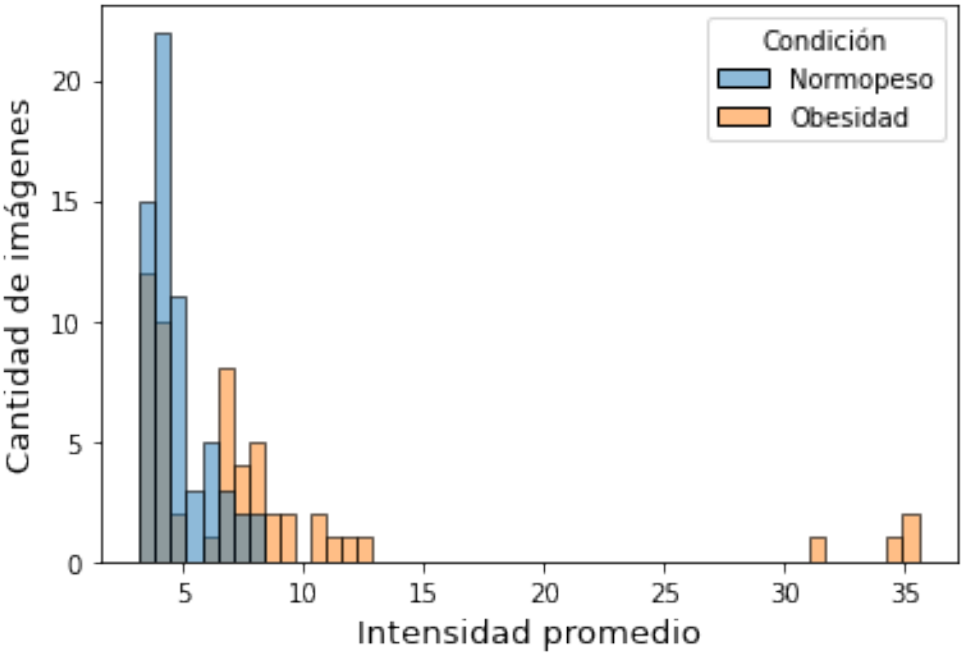

Figura 4.2: valor de las intensidades promedio de las regiones de interés.

Se calcularon las métricas de Exactitud (accuracy) y F1 Score para

este clasificador obteniendo los siguiente resultados:

Métrica	Valor
Exactitud	66.94 %
F1 Score	61.77 %

Tabla 4.1: Métricas del clasificador basado en la intensidad promedio de pixeles.

4.2.2. Por conteo de pixeles

Se muestra la gráfica de cantidad de imágenes (ROI´s), contra cantidad de pixeles (obscuros y claros).

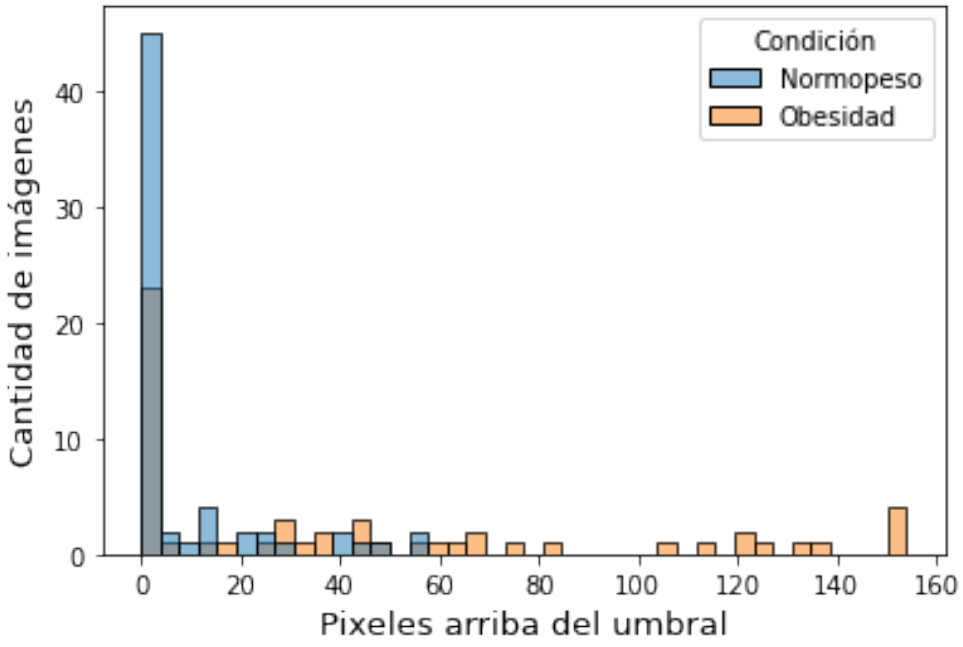

Figura 4.3: Cantidad de imágenes que tienen un cierto número de pixeles arriba del umbral.

Los valores obtenidos para las métricas usadas son los siguientes:

Métrica	**Valor**
Exactitud	59.32 %
F1 Score	49.18 %

Tabla 4.2: Valores de las métricas para el clasificador por conteo de pixeles.

4.2.3. Por contraste

Para aquellas imágenes con mayor rango de valores en cuanto a intensidad de pixeles, se espera que pertenezcan a la clase de obesidad, pues en las gráficas anteriores se notó que hay muchos datos del grupo de obesidad que tienen pixeles bajos, sin embargo, hay pocos datos del grupo de normopeso con pixeles altos.
En la siguiente gráfica se puede ver el número de imágenes que tienen un cierto contraste, esto muestra cómo se podrían clasificar los datos a partir de un cierto valor de contraste.

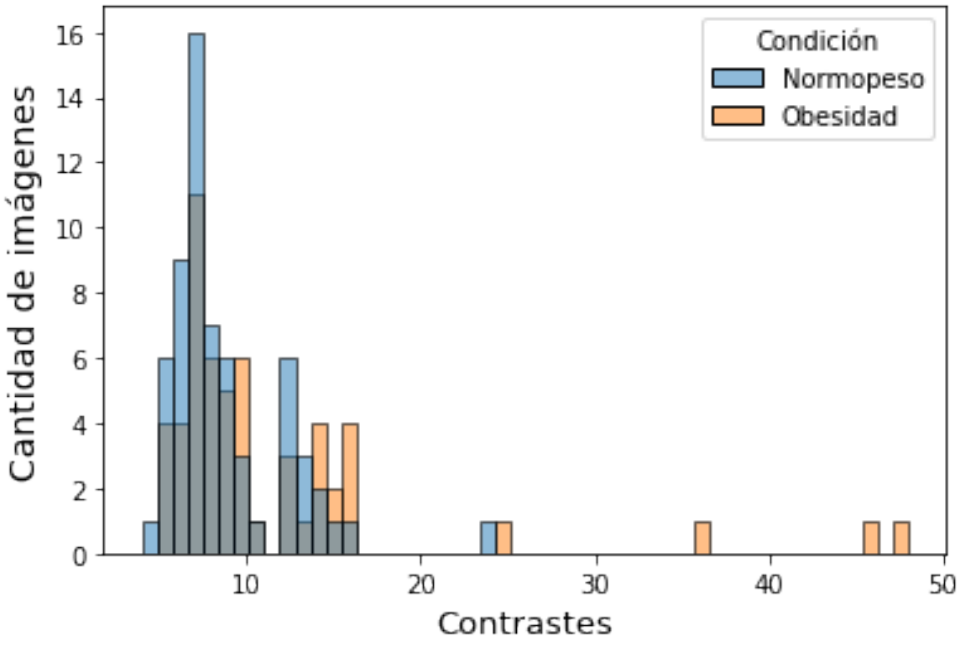

Figura 4.4: Cantidad de imágenes y contrastes de las ROI´s.

Las métricas obtenidas para este clasificador tienen los siguiente valores:

Métrica	Valor
Exactitud	47.45 %
F1 Score	31.47 %

Tabla 4.3: Valores de las métricas para el clasificador por contraste.

Los resultados de las métricas para los clasificadores utilizando las imágenes completas son los siguientes:

Clasificador	Intensidad promedio	Conteo de pixeles	Contraste
Exactitud	68.64 %	63.55 %	59.32 %
F1 Score	68.29 %	63.15 %	60.72 %

Tabla 4.4: Métricas de los clasificadores usando imágenes completas

A continuación, se describen los resultados obtenidos con el SVM como clasificador de los datos.

4.2.4. SVM

Se utilizó el clasificador SVM como un nuevo método de clasificación pues se trata de un clasificador muy recurrido en Machine Learning debido a su efectividad en espacios de dimensiones muy grandes; se crearon nuevamente dos clasificadores y se obtuvieron los siguientes resultados tanto en ROI´s como imágenes completas y para diferentes kernels del SVM.

Kernel	Lineal	Gaussiano	Cuadrático
Exactitud	66.66 %	62.50 %	54.16 %
F1 Score	66.19 %	57.72 %	38.06 %

Tabla 4.5: Tabla de métricas para ROI´s.

Los resultados obtenidos en las métricas para el SVM utilizando las imágenes completas son los siguientes:

Kernel	**Lineal**	**Gaussiano**	**Cuadrático**
Exactitud	95.83 %	87.50 %	83.33 %
F1 Score	95.81 %	87.52 %	82.59 %

Tabla 4.6: Tabla de métricas del SVM para imágenes completas.

A continuación, se presenta un modelo de red neuronal convolucional para poder hacer una comparación de todos los métodos utilizados en este trabajo.

4.3. Redes Neuronales

4.3.1. MODELO: CNN 1.

Este modelo consistió en utilizar un modelo de red neuronal complejo para hacer la clasificación de las imágenes, se trata del modelo preentrenado VGG-16 (sección 2.3.3), mismo que tras un entrenamiento de 500 épocas arrojó los siguientes resultados para la función de pérdida y para la exactitud:

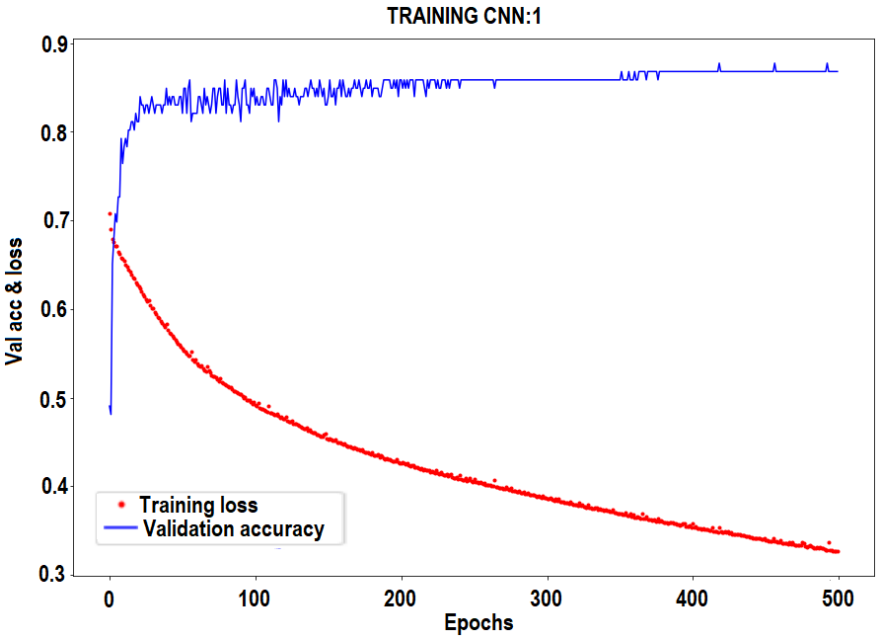

Figura 4.5: Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a las épocas de entrenamiento.

Podemos notar que hay una disminución importante en la función de pérdida durante el entrenamiento. La función de pérdida para el conjunto de prueba llegó al valor de 39.76 %, por otro lado, la exactitud en el conjunto de prueba llegó a un valor de 88.23 %.

4.3.2. MODELO: CNN 2.

Como se explicó en la sección (2.3.3), el modelo VGG19 es un modelo con más parámetros que el VGG16 por lo que es natural pensar que con un modelo más complejo, se podrá obtener una mayor exactitud en cuanto a la clasificación que se pretende hacer; el entrenamiento se hizo también con 500 épocas obteniendo los siguientes resultados:

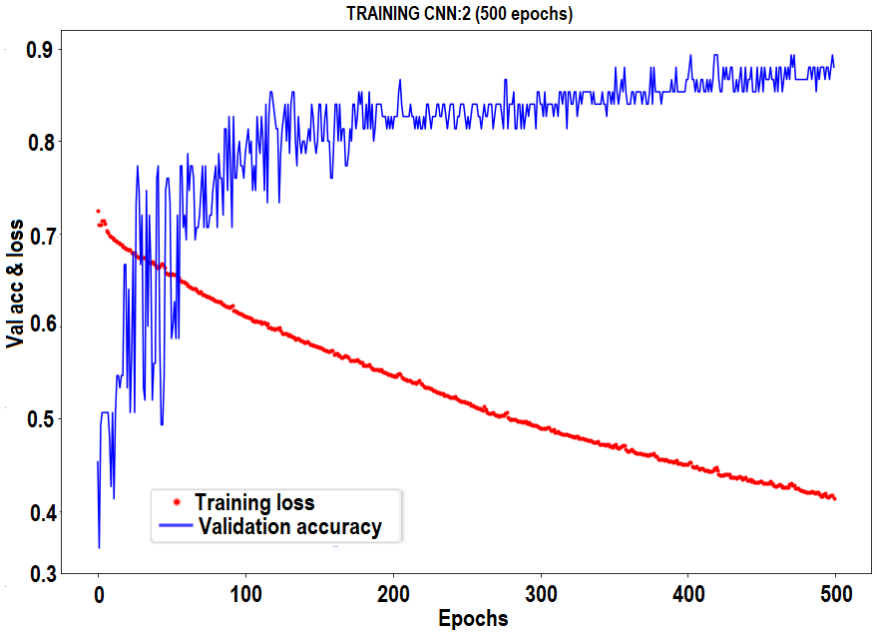

Figura 4.6: Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a 500 épocas de entrenamiento.

La función de pérdida durante el entrenamiento disminuyó y al final de 500 épocas mostraba tendencia a seguir disminuyendo. La exactitud en el conjunto de prueba alcanzó un valor de 83.33 %.

Se aumentan 200 épocas al entrenamiento, obteniendo las siguientes gráficas:

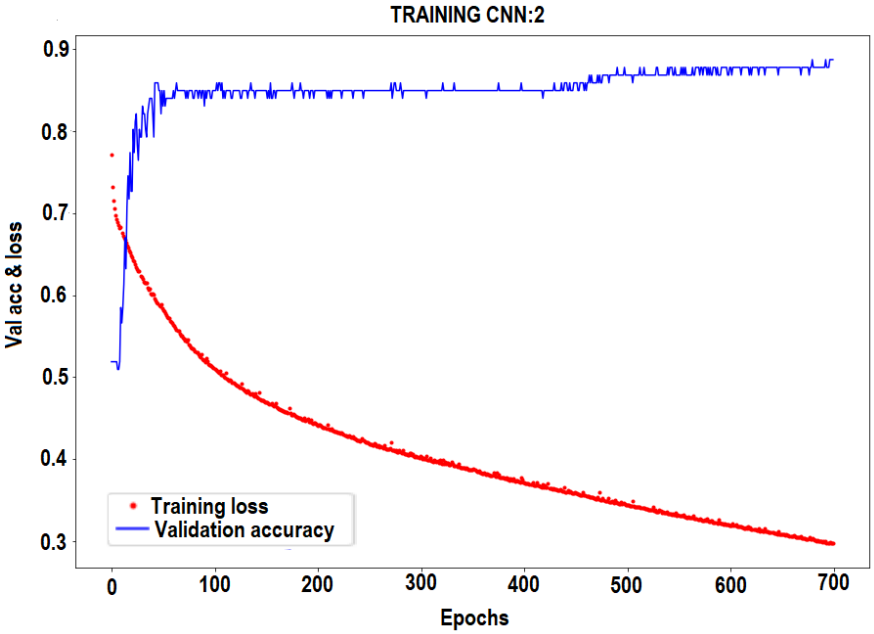

Figura 4.7: Gráfica de los valores de la función de pérdida y la exactitud del conjunto de validación con respecto a 700 épocas de entrenamiento.

La función de pérdida en el conjunto de prueba disminuyó hasta un valor de 38.64 % y la exactitud alcanzó un valor de 79.41 %.

CAPÍTULO 5

Discusión y Conclusiones

En mejor de los tres clasificadores iniciales propuestos es el de intensidad de pixeles promedio, se puede notar de la gráfica 4.2 que hay una separación de los datos; para el clasificador por conteo de pixeles arriba de un cierto threshold, se ve que es complicado hacer una separación por clases de los datos debido a que las imágenes del grupo de obesos contienen cantidades de pixeles intensos que van desde 0 hasta casi 160, significando que los datos de obesidad están repartidos entre todos los datos de normopeso y para el clasificador por contraste, la separación de los grupos es más complicada dado que la mayoría de los datos (de ambos grupos) se encuentran en los mismos rangos de valores de contraste. Al tomar las imágenes completas y clasificarlas de la misma forma, los resultados 4.4 muestran una gran diferencia en comparación con las métricas obtenidas de las ROI´s 4.1,4.2 y 4.3. Por otro lado, el clasificador SVM no resultó ser muy bueno en cuanto a la clasificación de ROI´s, pues las métricas son comparables con las de los clasificadores anteriores, sin embargo, al utilizar imágenes completas para este modelo, se obtienen ganancias muy grandes pues la exactitud mejora cerca del 30 %; además el kernel lineal resulta ser mejor para separar los

datos. Finalmente, los resultados obtenidos en las redes neuronales son similares entre sí, pues las redes en estructura son parecidas y a pesar de que se esperaba tener un mejor desempeño del modelo VGG-19, se tuvo en el VGG-16, lo cual se le atribuye a la forma de los datos, no siempre al elegir un modelo más complejo significa que se obtendrán mejores resultados.

Dentro del campo de la Física Médica destacan trabajos basados en el SVM para la creación de clasificadores binarios similares al de este trabajo de tesis, tal es el caso de [24] en donde se realizó un diagnóstico diferencial entre temblor esencial y enfermedad de Parkinson y al eliminar los casos atípicos llegaron a un 100 % de exactitud, por otro lado, para un análisis similar [33] también utilizaron tres kernels del SVM, la exactitud fue mejorando del kernel lineal, cuadrático y gaussiano de manera respectiva. Para el caso de la clasificación de MRI hepáticas completas de niños con obesidad y normopeso, los resultados fueron disminuyendo en los mismos kernels 4.6, lo cual se atribuye directamente a las características de los datos. Se puede afirmar que el SVM resulta ser más eficiente para la clasificación de MRI respecto a otras técnicas de clasificación estudiadas pues presentan claras ventajas que justifican los buenos resultados obtenidos.

En cuanto a las redes neuronales, se han reportado trabajos como el de Ahsan Bin Tufail et al.[28], en donde se hace un clasificador binario de imágenes de resonancia magnética de cerebro para distinguir los sujetos con enfermedad de Alzheimer de los sujetos sanos, utilizando Transfer Learning con la implementación de los modelos Inception versión 3 y Xception, que se entrenan con el conjunto de datos de ImageNet, igual que para este trabajo. En su caso resultó que el modelo CNN personalizado que construyeron para resolver el problema, no superó al hecho mediante Transfer Learning [28], lo cual muestra que crear una red neuronal adaptándola específicamente al problema, no garantiza que funcione mejor. Otra forma eficiente de crear modelos con transferencia de aprendizaje para clasificación de imágenes es la que se hizo en [34], donde se implementó el modelo VGG-19 para detectar nódulos pulmonares a partir de imágenes de Tomografía Computarizada mediante

segmentación y clasificación, las diferencias son, entre otras cosas, que se recortaron las imágenes al tamaño $224x224$ (que es con el que funcionan los modelos VGG), el lugar de adaptar el modelo al tamaño de las imágenes y obtuvieron mejores resultados en comparación con el modelo VGG-16. Las limitaciones de estudiar la obesidad con MRI hepáticas, empiezan desde la posibilidad de que los resultados se vean afectados al usar las imágenes completas, pues hay información no solo del hígado, sino de otras partes del cuerpo que aparecen a lo largo y ancho de las imágenes, lo cual podría resultar en que los clasificadores se basen en pixeles que están fuera del hígado, las distribuciones de los datos se encuentran sin modificaciones, no se descarta la posibilidad de hacer modificaciones para intentar mejorar la exactitud. Este trabajo se puede extender de varias formas. Como trabajo futuro, se pretende explorar otras modalidades de imágenes, y el problema de clasificación multiclase de la Esteatosis Hepática que logre comprender niños sanos, con Esteatosis leve y niños con EHGNA. Por otro lado, también se pretende explorar otras arquitecturas de redes neuronales y obtener información sobre el aprendizaje de arquitecturas de más redes neuronales a través de diferentes perspectivas.

Se crearon modelos de ML que permitieron hacer la distinción entre los dos grupos estudiados de los cuales el SVM resultó ser mejor incluso que los modelos de TL creados, se analizaron los datos MRI mediante estadística descriptiva por clases que finalmente, al hacer la comparación de los modelos se puede decir que la capacidad de los modelos de TL son bastante eficientes en comparación con los clasificadores realizados, sin embargo, el SVM resultó superior a todos y las métricas lo demuestran.

APÉNDICE A

Códigos.

```
#Codigo del clasificador SVM
"""## Bibliotecas"""
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import seaborn as sns
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, accuracy_score

# Tener acceeso al directorio de los datos
```

```
from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/MyDrive/ROIPRECISAS"

image_dir = sorted(os.listdir(path))
images = []
max = np.zeros(118)
i = 0
for image_name in image_dir:
  image_path = f"{path}/{image_name}"
  img = mpimg.imread(image_path, format='png').reshape(
  (14,11,1))
  max[i] = img.max()
  img_rescale = (img*255)

  img_int = img_rescale.astype(int)
  images.append(img_int)
  i+= 1

image_num = 167
normo_num = 85
#Hacer vector de etiquetas entre obesos y normopesos
Normopeso = np.repeat('Normopeso', normo_num)
Obesidad = np.repeat('Obesidad', image_num-normo_num)
labels = np.hstack( (Normopeso,Obesidad) )

"""### Dividimos dataset"""

x_train,x_test,y_train,y_test=train_test_split(images,
labels,test_size=0.20,random_state=0,stratify=labels)

x_train = np.array(x_train)
x_test = np.array(x_test)
```

```
row_image = 14
col_image = 11

#Definimos modelo LINEAL
model_svm = make_pipeline(StandardScaler(),LinearSVC(
random_state=0, tol=1e-5,max_iter = 1000))

#Entrenamos modelo
model_svm.fit(x_train.reshape((len(x_train),
row_image*col_image)),y_train,)

y_pred = model_svm.predict(x_test.reshape((len(x_test),
row_image*col_image)))

#Definimos modelo gaussiano
model_svmgauss = make_pipeline(StandardScaler(),
SVC(kernel='rbf', random_state=0, tol=1e-5,max_iter = 1000))

#Entrenamos el modelo
model_svmgauss.fit(x_train.reshape((len(x_train),
row_image*col_image)),y_train,)

y_predgauss = model_svmgauss.predict(x_test.reshape(
(len(x_test), row_image*col_image)))

#Definimos modelo cuadratico
model_svmsquare = make_pipeline(StandardScaler(),
SVC(kernel='poly', degree=2, random_state=0, tol=1e-5,
max_iter = 1000))

#Entrenamos el modelo
model_svmsquare.fit(x_train.reshape((len(x_train),
row_image*col_image)),y_train,)
```

```
y_predsqr = model_svmsquare.predict(x_test.reshape(
((len(x_test), row_image*col_image)))

y_predsqr

"""Metricas del modelo lineal"""

print("accuracy", accuracy_score(y_test, y_pred))

f1_score(y_test, y_pred, average='weighted')

"""Metricas del modelo sqr"""

print("accuracy", accuracy_score(y_test, y_predsqr))

f1_score(y_test, y_predsqr, average='weighted')


"""Metricas del modelo gaussiano"""

print("accuracy", accuracy_score(y_test, y_predgauss))

f1_score(y_test, y_predgauss, average='weighted')


#Codigo del modelo VGG-16

### Bibliotecas

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
import os
import seaborn as sns
```

```
import pandas as pd
import matplotlib.image as mpimg
import sys
from os import listdir
from os.path import isfile, join
!pip install pydicom
import pydicom
# Tener acceeso al directorio de los datos
from google.colab import drive
drive.mount('/content/drive')

"""### Code"""

path = "/content/drive/MyDrive/imgsdosgrupos"

image_dir = sorted(os.listdir(path))
images = []
for i in image_dir:
  #image_path = pydicom.read_file(os.path.join(path, i))
  image_path = os.path.join(path, i)
  img = pydicom.dcmread(image_path)
  images.append(img.pixel_array)

images = np.array(images) #Convertimos nuestra lista
de imagenes en un np.array()

images.shape

image_num = 167
normo_num = 85
# Hacer el vector de etiquetas entre obesidad y normopeso
normopeso = np.repeat(0, normo_num)
obesidad = np.repeat(1, image_num-normo_num)
labels = np.hstack( (normopeso,obesidad) )
```

```
x_train,x_test,y_train,y_test=train_test_split(images,
labels,test_size=0.20,random_state=0,stratify=labels)
#separar el conjunto en entrenam, prueba (con split)
x_train = np.expand_dims(x_train, axis=3)

x_train = np.repeat(x_train,repeats=3,axis=3)
x_test = np.expand_dims(x_test, axis=3)
x_test = np.repeat(x_test,repeats=3,axis=3)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(x_train.shape,x_test.shape,y_train.shape,
y_test.shape)

print(x_train.shape,x_test.shape,y_train.shape,
y_test.shape)

"""## Keras transfer learning & fine-tuning


### Databases


import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16
import preprocess_input
#from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D
from tensorflow.keras.layers import Input

from keras.utils import np_utils
```

```
from tensorflow.keras import optimizers
from keras.utils.vis_utils import plot_model
import tensorflow.keras.backend as TF
TF.floatx()

"""### Preprocesamiento de los datasets"""
#transformammos datos
x_train = x_train.astype(TF.floatx()) / 255

#transformammos datos entre 0 y 1
x_test = x_test.astype(TF.floatx()) / 255

"""### Definicion del Modelo"""

base_model = VGG16(weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x) #se aplana aqui
#capa de salida dos clases
predictions = Dense(1, activation='sigmoid')(x)
#siempre se pone una capa dense al final

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)

"""### Visualizacion del Modelo"""

for layer in base_model.layers:
    layer.trainable = False
#Verificamos que solo la ultima capa se va a
entrenar para nuestro modelo
print(model.summary())
# Visualizacion

plot_model(model, to_file='VGG16REDcompl.png',
show_shapes=True, show_layer_names=True)
```

```
# compile the model (should be done *after*
setting layers to non-trainable)
model.compile(optimizer=optimizers.RMSprop(
learning_rate=0.001), loss='binary_crossentropy',
metrics=['accuracy']) #loss es para cuando tenemos
problema binario learning rate es el paso de
descenso que tan rapido va a ir bajando
#con un learning rate de 0.01 esta bien a veces
este parametro significa muy bueno o mal rendimiento
del modelo para que me asegure que voy a llegar mas
abajo en el descenso (pasote o pasitos)
#alpha       = 0.0001    # razon de aprendizaje
#decay       = 0.0001
#model.compile(optimizer =optimizers.Adam(lr=alpha,
decay=decay),loss = 'binary_crossentropy',
metrics    = ['accuracy'])

import time
tic = time.time()
history = model.fit(x_train,y_train ,
epochs=500, batch_size=64,validation_split=0.2, verbose=1)
#x train inclye al validation, lo separo de nuevo.
toc = time.time()
print('seconds=', toc-tic)
#plotear val_loss y val_acc

"""### Visualization of Training"""

# training history
loss = history.history['loss']
acc  = history.history['accuracy']

start = 0
end = len(loss)
```

```
epochs = range(start, end)
plt.figure(2, figsize=(32,10))
plt.subplot(1,2,1)
plt.plot(epochs, loss[start:end], 'r.',
label='Training loss')
#plt.plot(epochs, val_loss[start:end], 'r.',
label='Training val_loss')
#plt.plot(epochs, val_acc[start:end], 'b',
label='Validation val_accuracy')
plt.plot(epochs, acc[start:end], 'b',
label='Validation accuracy')
plt.title('Training and validation BCE loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (BCE)')
plt.legend()
plt.show()

"""### Evaluate on Test"""

test_loss, test_acc = model.evaluate(x_test,
y_test, verbose=1)
print("Test loss:", test_loss)
print("Test accuracy:",test_acc)

model.predict(x_test).shape

"""### Saving Model

"""

model.save("VGG16RED500epocs.h5")
model = keras.models.load_model("myVGG16500epocs.h5")
del model

model.predict(np.expand_dims(x_test[11],axis=0))
```

```
x_test.shape


#Codigo del modelo VGG-19
### Bibliotecas
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
import os
import seaborn as sns
import pandas as pd
import matplotlib.image as mpimg
import sys
from os import listdir
from os.path import isfile, join
!pip install pydicom
import pydicom
# Tener acceeso al directorio de los datos
from google.colab import drive
drive.mount('/content/drive')

"""### Code"""
path = "/content/drive/MyDrive/imgsdosgrupos"

image_dir = sorted(os.listdir(path))
images = []
for i in image_dir:
  #image_path = pydicom.read_file(os.path.join(path,i))
  image_path = os.path.join(path, i)
  img = pydicom.dcmread(image_path)
  images.append(img.pixel_array)

images = np.array(images)
```

```
images.shape

image_num = 167
normo_num = 85
# Hacer el vector de etiquetas entre obesidad y normopeso
normopeso = np.repeat(0, normo_num)
obesidad = np.repeat(1, image_num-normo_num)
labels = np.hstack( (normopeso,obesidad) )

x_train,x_test,y_train,y_test=train_test_split(images,
labels,test_size=0.20,random_state=0,stratify=labels)
x_train = np.expand_dims(x_train, axis=3)
x_train = np.repeat(x_train,repeats=3,axis=3)
x_test = np.expand_dims(x_test, axis=3)
x_test = np.repeat(x_test,repeats=3,axis=3)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)

print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)

"""## Keras transfer learning \& fine-tuning

### Databases
"""

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.vgg19
import preprocess_input
#from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
```

```
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D
from tensorflow.keras.layers import Input

from keras.utils import np_utils
from tensorflow.keras import optimizers
#visualizacion del Modelo
from keras.utils.vis_utils import plot_model
import tensorflow.keras.backend as TF
TF.floatx()

"""### Preprocesamiento de los datasets"""

x_train = x_train.astype(TF.floatx()) / 255
x_test = x_test.astype(TF.floatx()) / 255

"""### Definicion del Modelo"""

base_model = VGG19(weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x) #se aplana aqui
#capa de salida dos clases
#siempre se pone una capa dense al final
predictions = Dense(1, activation='sigmoid')(x)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)

"""### Visualizacion del Modelo"""

for layer in base_model.layers:
    layer.trainable = False
#Verificamos que solo la ultima capa se va entrenar
para nuestro modelo
print(model.summary())
```

```
# Visualizacion

plot_model(model, to_file='VGG19RED.png',show_shapes=True,
show_layer_names=True)

# compile the model (despues de configurar las capas
del no entrenable)
model.compile(optimizer=optimizers.RMSprop(learning_rate=
0.001),loss='binary_crossentropy',metrics=['accuracy'])
#alpha       = 0.0001     # razon de aprendizaje
#decay       = 0.0001
#model.compile(optimizer =optimizers.Adam(lr=alpha,
decay=decay),loss = 'binary_crossentropy',
metrics = ['accuracy'])

import time
tic = time.time()
history = model.fit(x_train,y_train , epochs=700,
#x train inclye al validation
batch_size=64,validation_split=0.2, verbose=1)

toc = time.time()
print('seconds=', toc-tic)
#plotear val_loss y val_acc

"""### Visualization of Training"""

# training history
loss = history.history['loss']
acc  = history.history['accuracy']

start = 0
end = len(loss)
epochs = range(start, end)
plt.figure(2, figsize=(32,10))
```

```
plt.subplot(1,2,1)
plt.plot(epochs, loss[start:end], 'r.',
label='Training loss')
#plt.plot(epochs, val_loss[start:end], 'r.',
label='Training val_loss')
#plt.plot(epochs, val_acc[start:end], 'b',
label='Validation val_accuracy')
plt.plot(epochs, acc[start:end], 'b',
label='Validation accuracy')
plt.title('Training and validation BCE loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (BCE)')
plt.legend()
plt.show()

"""### Evaluate on Test"""

test_loss, test_acc = model.evaluate(x_test,
y_test, verbose=1)
print("Test loss:", test_loss)
print("Test accuracy:",test_acc)

model.predict(x_test).shape

"""### Saving Model

"""

'''model.save("myVGG19RED.h5")
del model
model = keras.models.load_model("myVGG19.h5") '''

'''model.predict(np.expand_dims(x_test[87], axis=0)) '''
```

Bibliografía

[1] H. KAY, M., *Enfermedad por hígado graso*, AMERICAN COLLEGE OF GASTROENTEROLOGY, 2011. Recuperado el 26 de noviembre de 2021, de https://gi.org/patients/recursos-en-espanol/enfermedad-por-higadograso/.

[2] NIH, NHLBI , *Obesity Education Initiative. Clinical Guidelines on the Identification, Evaluation and Treatment of Overweight and Obesity in Adults*, Available online: http://www.nhlbi.nih.gov/guidelines/obesity/ob_gdlns.pdf

[3] J.D. Browning, L.S. Szczepaniak, R. Dobbins, P. Nuremberg, J.D. Horton, J.C. Cohen et al., *Prevalence of hepatic steatosis in an urban population in the United States: Impact of ethnicity.* Hepatology, 40 (2004), pp. 1387-1395 http://dx.doi.org/10.1002/hep.20466

[4] HANNAH I.AWAI KIMBERLY P.NEWTON CLAUDE B.SIRLIN CYNTHIABEHLING, JEFFREY B.SCHWIMMER, *Evidence and Recommendations for Imaging Liver Fat in Children, Based on Systematic Review*, CLINICAL GASTROENTEROLOGY AND HEPATOLOGY VOLU-

ME 12, ISSUE 5, MAY 2014, PAGES 765-773. DOI: HTTPS://DOI.ORG/10.1016/J.CGH.2013.09.050

[5] MIDDLETON M.S. VAN NATTA M.L. HEBA E.R. ALAZRAKI A. TROUT A.T. MASAND P. ET AL., *Diagnostic accuracy of magnetic resonance imaging hepatic proton density fat fraction in pediatric nonalcoholic fatty liver disease*, HEPATOLOGY, 2017; 67 (PUBMED PMID): 858-872

[6] LEVENSON H, GREENSITE F, HOEFS J, ET AL., *Fatty infiltration of the liver: quantification with phase-contrast MR imaging at 1.5 T vs biopsy.*, AJR AM J ROENTGENOL 1991; 156:307–312.

[7] ADYTIA SHARMA, *Redes neuronales convolucionales en Python con Keras*, 2017. RECUPERADO EL 20 DE MAYO 2021, DE DATACAMP SITIO WEB: HTTPS://WWW.DATACAMP.COM/COMMUNITY/TUTORIALS/CONVOLUTIONAL- NEURAL-NETWORKS-PYTHON

[8] ALEXANDER SELVIKVAG LUNDERVOLD, ARVID LUNDERVOLD, *An overview of deep learning in medical imaging focusing on MRI*, Z MED PHYS 29, (2019), 102–127, HTTPS://DOI.ORG/10.1016/J.ZEMEDI.2018.11.002

[9] RODRIGO, J. A., *Máquinas de Vector Soporte (Support Vector Machines), SVMs*, (2017, ABRIL), CIENCIADEDATOS.NET. RECUPERADO 26 DE MARZO DE 2021, DE HTTPS://WWW.CIENCIADEDATOS.NET/DOCUMENTOS/34_MAQUINAS _DE _VECTOR_SOPORTE_SUPPORT_VECTOR_MACHINES

[10] TORRES, J. *Datos y Overfitting Torres AI*, (2019, 1 SEPTIEMBRE). RECUPERADO 24 DE ABRIL DE 2022, DE HTTPS://TORRES.AI/DATOS-Y-OVERFITTING-KERAS-TENSORFLOW/

[11] J., *Transfer Learning Incremental con Keras y Creme. DataSmarts Español*(2020, 2 JULIO). RECUPERADO 1 DE MAYO DE

2022, de https://datasmarts.net/es/transfer-learning-incremental-con-keras-y-creme/

[12] Mayo Clinic, *Obesidad Síntomas y causas*, Mayo Clinic, (2021, 11 noviembre). Recuperado 3 de febrero de 2022, de https://www.mayoclinic.org/es-es/diseases-conditions/obesity/symptomscauses/syc-20375742#:%7E:text=Aunque%20existen%20influencias%20gen%C3%A9ticas%2C%20conductuales,calor%C3%ADas%20en%20forma%20de%20grasa.

[13] UNICEF, Instituto Nacional de Salud Pública (INSP) *(s. f.). Encuesta Nacional de Niños, Niñas y Mujeres 2015*, UNICEF. Recuperado 22 de febrero de 2021, de https://www.unicef.org/mexico/informes/encuesta-nacional-de-ni%C3%B1os-ni%C3%B1as-y-mujeres-2015

[14] Mikkila V, Rasanen L, Raitakari OT, Pietinen P, Viikari J., *Longitudinal changes in diet from childhood into adulthood with respect to risk of cardiovascular diseases: the Cardiovascular Risk in Young Finns Study*, Eur J Clin Nutr 2004;58(7):1038-45

[15] Raúl Jiménez-Agüero, José I Emparanza, Adolfo Beguiristain, Luis Bujanda, José M Alustiza, Elisabeth García, Elizabeth Hijona, Lander Gallego, Javier Sánchez-González, María J Perugorria, José I Asensio, Santiago Larburu, Maddi Garmendia, Mikel Larzabal, María P Portillo, Leixuri Aguirre and Jesús M Bañales*Novel equation to determine the hepatic triglyceride concentration in humans by MRI: diagnosis and monitoring of NAFLD in obese patients before and after bariatric surgery* BMC Medicine 2014, 12:137. Doi: 10.1186/s12916-014-0137

[16] Bolinger L. e Insko E.K., *Spectroscopy: Basic principles and techniques in Clinical Magnetic Resonance Imaging* Edelman R.R., Hesselink J.R. y Zlatkin M.B: eds.,2ª edición. W.B. Saunders Company, Philadelphia, USA (1996) pp. 353-379.

[17] Sissons, C., *What to know about fatty liver disease. MEDICALNEWSTODAY*, (2020, 5 febrero), Recuperado 3 de febrero de 2021, de https://www.medicalnewstoday.com/articles/fatty-liver

[18] Kumar, V., *So why the heck are they called Support Vector Machines? Medium*, (2022, 30 marzo). Recuperado 25 de marzo de 2022, de https://towardsdatascience.com/so-why-the-heck-are-they-called-support-vector-machines-52fc72c990a1

[19] Tholey, D., *Nonalcoholic Fatty Liver Disease (NAFLD)* (2022, 18 abril), MSD Manual Professional Edition. Recuperado 22 de abril de 2022, de https://www.msdmanuals.com/professional/hepatic-and-biliary-disorders/approach-to-the-patient-with-liver-disease/nonalcoholic-fatty-liver-disease-nafld

[20] Bloch F., Hansen W. W., Packard M, *Inducción nuclear*, 1 February 1946, Phys Rev. 1946; 69 :127.

[21] Purcell EM, Torrey HC, Pound RV, *Absorción de resonancia por momentos magnéticos nucleares en un sólido* Phys Rev. 1946; 69 :37–38.

[22] Grover V.P.B., Tognarelli J.M., Crossey M.M.E., Cox I.J., Taylor-Robinson S.D., McPhail M.J.W., *Magnetic resonance imaging: Principles and techniques: Lessons for clinicians*,J. Clin. Exp. Hepatol. 2015;5:246–255. doi: 10.1016/j.jceh.2015.08.001

[23] Alcañas, Ú. *Fase y Fuera de Fase. Resonancia Magnética, los protocolos y su práctica*, (2017, 1 abril). Recuperado 5 de mayo de 2022, de https://protocolosresonanciamagnetica.org/2017/02/20/fase-y-fuera-de-fase/

[24] Roberto González et al., *Aplicación de las Máquinas de Soporte Vectorial (SVM) al diagnóstico clínico de la Enfermedad de Párkinson y el Temblor Esencial*, Revista Iberoamericana de Automática e Informática industrial 14 (2017) 394405 https://digital.csic.es/bitstream/10261/170901/4 /Gonzalez_Aplicacion_Maquinas_RIAI_14_2017.pdf)

[25] Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, *How transferable are features in deep neural networks?* arXiv preprint arXiv:1411.1792v1, 2014

[26] Zhou, J., *Multisignal VGG19 Network with Transposed Convolution for Rotating Machinery Fault Diagnosis Based on Deep Transfer Learning*, (2020, 8 diciembre), Hindawi. Recuperado 3 de abril de 2021, de https://www.hindawi.com/journals/sv/2020/8863388/

[27] I., *Deep Learning CNN: Convolutional Neural Networks with Python*, Views Coupon. (2022, 13 mayo). Recuperado 3 de febrero de 2022, de https://www.viewscoupon.com/2022/05/deep-learning-cnn-convolutional-neural.html

[28] Tufail, A.B., Ma, YK. & Zhang, QN, *Binary Classification of Alzheimer's Disease Using sMRI Imaging Modality and Deep Learning*, J Digit Imaging 33, 1073–1090 (2020). https://doi.org/10.1007/s10278-019-00265-5

[29] Brownlee, J., *Gradient Descent With RMSProp from Scratch. Machine Learning Mastery.* (2021, 11 octubre). Recuperado 29 de abril de 2022, de https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/

[30] Kingma, D. P., Ba, J., *Adam: A method for stochastic optimization.* , arXiv preprint arXiv:1412.6980, (2014).

[31] Le, K., *An overview of VGG16 and NiN models - MLearning* (2021, marzo 25), AI - Medium. MLearning.AI. https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models-96e4bf398484

[32] Kamil, M. y., Saood, A. & Hatem I., *Modified VGG−19 model architecture*, (2021, febrero 1) ResearchGate. https://www.researchgate.net/figure/ModifiedVGG-19modelarchitecture_fig1_344398328

[33] Lingmei, A., Jue, W., Ruoxia, Y., *Classification of parkinsonian and essential tremor using empirical mode decomposition and support vector machine*, Digital Signal Processing. Pag. 543-550. Jan. 2011.

[34] Khan MA, Rajinikanth V, Satapathy SC, Taniar D, Mohanty JR, Tariq U, Damaševičius R., *VGG19 Network Assisted Joint Segmentation and Classification of Lung Nodules in CT Images*, Diagnostics. 2021; 11(12):2208. https://doi.org/10.3390/diagnostics11122208

[35] , *In phase/out of phase. (s/f).* Questions and Answers in MRI. Recuperado el 14 de mayo de 2022, de https://mriquestions.com/in-phaseout-of-phase.html

[36] Outwater EK, Blasbalg R, Siegelman ES, Vala M., *Detection of lipid in abdominal tissues with opposed-phase gradient-echo images at 1.5T: techniques and diagnostic importance*, Radiographics 1998; 18:1465-80.

[37] Lucy Frank; Novelline, Robert A., *Squire's fundamentals of radiology*, 6ª edición, 1997.

[38] Barrie, S. N. & Webb A., *Introduction to medical imaging: Physics, engineering and clinical applications*, Cambridge: Cambridge University Press(2010).

[39] Cortes, C & Vapnik, V., *Support−vector networks.*, Machine Learning (1995), 20(3), 273−297

[40] V. B., A., *Support Vector Machine (SVM) – Machine Learning*, (2019, 22 mayo), Solo Para Entendidos. Recuperado 24 de marzo de 2021, de http://www.soloentendidos.com/support-vector-machine-machine-learning-2099

[41] , *Herramientas con Inteligencia Artificial.*, (2019, 24 octubre). Solutecia. Recuperado 24 de marzo de 2021, de https://www.solutecia.com/es/noticias-y-blog

[42] McCulloch, W.S., Pitts, W., *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 5, 115–133 (1943). https://doi.org/10.1007/BF02478259

[43] Rosenblatt, F., *The perceptron: A probabilistic model for information storage and organization in the brain* (1958), Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519

[44] J. Schmidhuber, *Deep Learning in Neural Networks: An Overview*, (2014), http://arxiv.org/abs/1404.7828,

[45] Goodfellow, Bengio & Courville, *Furthermore, backpropagation is often misunderstood as being specific to multi-layer neural networks, but in principle it can compute derivatives of any function*, 2016, p. 200.

[46] Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L., *ImageNet: A Large-Scale Hierarchical Image Database*, CVPR09 (2009), http://www.image-net.org/papers/imagenet_cvpr09.bib

[47] Simonyan Karen & Zisserman Andrew , *Very Deep Convolutional Networks for Large-Scale Image Recognition*, preimpresión de arXiv arXiv:1409.1556, (2014)

[48] Johnson, D., *Back Propagation Neural Network: What is Backpropagation Algorithm in Machine Learning?* (2022, 8 marzo). Guru99. Recuperado 23 de abril de 2022, de https://www.guru99.com/backpropogation-neural-network.html

[49] Calvo, D., *Red Neuronal Convolucional CNN* (2018, 8 diciembre), Diego Calvo. Recuperado 24 de noviembre de 2021, de https://www.diegocalvo.es/red-neuronal-convolucional/