Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias Físico Matemáticas

# Diseño y aplicación de una Red Neuronal Artificial para categorización e identificación de imágenes histopatológicas

Tesis presentada al

Colegio de Física

como requisito para la obtención del grado de

Licenciado en Física Aplicada

Por

Octavio Mendoza Gómez

Asesorado por

Dra. Beatriz Bonilla Capilla

Puebla, Pue.
Marzo 2021


**Benemérita Universidad Autónoma de Puebla**
Facultad de Ciencias Físico Matemáticas
Licenciatura en Física Aplicada

# Diseño y aplicación de una red neuronal artificial para categorización e identificación de imágenes histopatológicas

**Tesis que presenta el C. Octavio Mendoza Gömez, para obtener el grado de Licenciado en Física Aplicada**

**Autor**

**Octavio Mendoza Gómez**

**Directora de tesis**

**Dra.Beatriz Bonilla Capilla**

**Puebla, Puebla**
**Marzo del 2021**

# Resumen

La inteligencia artificial tiene una gran cantidad de aplicaciones, desde el reconocimiento facial hasta la predicción texto, siendo una de las más importantes el reconocimiento de imágenes mediante patrones de semejanza. Para aplicar la inteligencia artificial al área médica existen un sinfín de ideas, entre ellas destaca el reconocimiento de imágenes histopatológicas para dar un diagnóstico sobre el tipo de un tumor, benigno o maligno, de tal manera que se pueda dar un resultado al paciente con mayor seguridad.

# Dedicatoría

Este trabajo de tesis está dedicado a mi padre y hermana, que han visto mi esfuerzo y dedicación desde el principio hasta el final, a mi abuelo por su amor y cariño incondicional. Y sobre todo dedico mi esfuerzo a mi, que mi futuro lo tenga como base para formarse.
Dedico este trabajo al fallecido Dr. Ivan Hernandez Orzuna, quien me enseñó el desafío que es estudiar esta carrera.

# Agradecimientos

Agradezco a mi asesora la Dra. Beatriz Bonilla Capilla, por sus enseñanzas a lo largo de mi trabajo con ella y su guía al elaborar este trabajo.

Agradezco a Fernando Garza, Pablo, Marlene, Hugo, Carlos, Pamela, Willy, Rodolfo, Benjamin, Aurora, Alexis, Fernando Mendez, Ricardo Antonio y Ricardo Regalado por su apoyo moral y amistad a lo largo de la carrera. También agradezco a mis compañeros que me han acompañado a través de este tiempo como estudiante. Agradezco a Eduardo, Eric Ricardo, Luis y Victor por su amistad, apoyo y confianza en mí antes y durante mis estudios de licenciatura.

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

Desde el nacimiento de la humanidad ha existido la necesidad de crear herramientas para continuar con nuestro desarrollo y mantener un estilo de vida cómodo de acuerdo a las necesidades que aparezcan respecto a la época.
Podemos ver ejemplos de estas herramientas en nuestra historia como especie, al transcurrir el tiempo, se han presentado múltiples situaciones que nos han llevado como especie a evolucionar nuestra manera de pensar y actuar. Desde la creación de la rueda para que tribus antiguas tuvieran una movilidad menos desgastante, la creación de la bombilla para no quedarnos a oscuras en la noche, el desarrollo de las matemáticas necesarias para crear la primera computadora y así lograr cálculos complejos con mayor facilidad y en menor tiempo, hasta la creación de máquinas capaces de simular el comportamiento de los órganos humanos. Todos los ejemplos anteriores, y muchas más herramientas, nacieron gracias a la existencia de cubrir necesidades que se fueron presentando en distintas situaciones pasadas. A la fecha, muchas de esas creaciones humanas forman parte de nuestra cotidianidad, pero en su tiempo, eran un lujo o algo que simplemente no era necesario.

Recordemos un suceso muy importante en la historia humana, en la segunda guerra mundial, las tropas nazis habían creado una codificación única para sus mensajes y era muy difícil descubrir decodificarlos y descubrir que decían. Muchos países crearon equipos especializados en la decodificación de dichos mensajes, conformados por personal con estudios especializados y capacidades enfocadas a ello, aún con varios equipos dedicados, no se obtuvo ningún avance y la guerra estaba casi perdida. No fue hasta que Alan Turing y su equipo se enfocó en la creación de una herramienta capaz de usar algoritmos y procesos matemáticos para la decodificación de dichos mensajes. Esta máquina, para su tiempo, fue clave para salvar a millones de personas y detener la invasión por parte del ejercito nazi.
Aunque sea difícil de creerlo, nosotros llevamos en los bolsillos un aparato mil veces más potente que la máquina de Turing, nuestro celular. Pero ¿cómo es posible llevar tener algo así? La respuesta es sencilla, es gracias a la inteligencia artificial, la cual se ha aplicado a múltiples necesidades para mantener la vida como la conocemos, no solamente para detener guerras.

Una de sus aplicaciones más importante es el área médica, donde se desarrolla desde el diagnóstico de enfermedades comunes como la gripe, hasta la implementación adecuada de un medicamento para un paciente con una enfermedad más peligrosa.

En esta tesis nos enfocaremos en una tarea no tan complicada, implementaremos una red neuronal para que reconozca las diferencias entre imágenes histopatológicas de tumores benignos y malignos y así nos otorgue un diagnóstico concreto de qué tipo de tumor tiene cada paciente. Empezaremos exponiendo un poco de los antecedentes de las redes neuronales, la explicación del concepto "inteligencia artificial", qué es el aprendizaje de máquinas y sus algoritmos más importantes para una red neuronal y explicaremos la rama de aprendizaje profundo con todos los concepto necesarios para entender el funcionamiento de una red neuronal artificial, desde su inspiración biológica hasta su estructura.
Después presentaremos la metodología que se llevó a cabo para diseñar e implementar una red neuronal para la clasificación de imágenes en el lenguaje de programación python, siendo más específicos, el uso del módulo TensorFlow, para programar un algoritmo de red neuronal y se le introducirá una base de datos de imágenes histopatológicas específica para que nuestra red neuronal sea entrenada para comparar de manera gráfica el costo computacional y la precisión de nuestra red neuronal en el entrenamiento con datos y la validación de datos. Procederemos a mostrar las gráficas obtenidas de diez entrenamientos de nuestra red neuronal usando la misma base de datos y compararemos cada una.
Para finalizar presentaremos los resultados obtenidos del análisis de veinte imágenes de nuestra base de datos de prueba, para comprobar el grado de aprendizaje de nuestra red neuronal, y expondremos las conclusiones obtenidas.

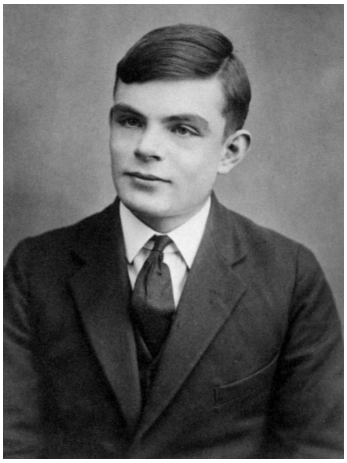
# Capítulo 2

# Inteligencia artificial

En la época actual, la mayoría de nuestras actividades las realizamos en distintos medios digitales, la mayoría de ellos están conectados a internet y aunque no lo notemos pasamos una cantidad significativa de nuestro tiempo interactuando con sistemas inteligentes. Pero ¿qué son los sistemas inteligentes? algunos ejemplos cotidianos de estos sistemas los podemos ver en acciones que para nosotros se han vuelto involuntarias, el uso de un buscador web para encontrar información que nos pidieron en algún trabajo, desbloquear nuestro celular mediante el reconocimiento de los vectores característicos de nuestro rostro o incluso la corrección ortográfica de palabras en un mensaje que estamos redactando. El pilar de los sistemas inteligentes que usamos es la inteligencia artificial, por ello, se está convirtiendo en una parte muy importante de nuestro estilo de vida moderno. Si vemos a los sistemas inteligentes como un organismo vivo, podemos decir que la inteligencia artificial su corazón, sin ella no podrían funcionar. En nuestro mundo, existe una gran gama de fenómenos, naturales y no naturales, y para estudiarlos usamos las distintas ciencias que hemos creado, las ciencias exactas se encargan de estudiar estos fenómenos desde un punto de vista matemático, las ciencias biológicas estudian su funcionamiento e intentan explicar sus características internas y externas y las ciencias sociales estudian cómo interactúan y se ven afectados los humanos por estos fenómenos. Muchos de los fenómenos estudiados son complejos y el poder explicarlos se vuelve difícil, gracias a la inteligencia podemos resolverlos o hacerlos un poco más sencillos, esto lo hace mediante la aplicación de matemáticas y el uso una serie de reglas ordenadas que conocemos como algoritmos [5]. A pesar de tener sus bases en las ciencias exactas, la inteligencia artificial se puede aplicar también a las ciencias biológicas y sociales.

## 2.1. Antecedentes

A principios del siglo XX se crearon los fundamentos para la lógica y teoría de ciencia computacional, los teoremas de Kurt Gödel de lógica de primer orden para predicciones, uno de ellos nos dicen que todo enunciado verdadero puede ser demostrado haciendo uso de las reglas del cálculo formal cuando lo formulamos en en lenguaje de la lógica de predicados, este teorema es conocido como el teorema de completes[3].



(a) Alan Turing.

(b) Kurt Gödel.

Figura 2.1: Fotos de Alan Turing y Kurt Gödel.

Puede que Gödel no sea muy conocido, pero muchos deben conocer la historia de Alan Turing, el matemático que estuvo involucrado en los primeros pasos para la creación de la computadora. Durante la segunda guerra mundial se presentó la necesidad de procesar grandes cantidades de información en un corto periodo de tiempo lo cual, para la época, era algo imposible, nunca pasó por la mente de las personas que una máquina pudiera ser programada para realizar tareas que requerían la intervención de un humano. En un artículo publicado en 1950, Turing dio un enfoque distinto a las máquinas, en lugar de pensar sobre su construcción o preocuparse por cómo se veía, él quería que se concentrará en su comportamiento observable. Su idea era una conversación a ciegas de un interrogador con dos participantes, una persona y una computadora, dicha conversación sería como cualquier otra, distintos temas de conversación que fluyeran. El interrogador no solo debía entablar una conversación escrita, también debía adivinar cuál participante era el humano, no fue sorpresa para Turing que su hipótesis se cumpliera, la computadora siempre iba a ganar.

Fue en 1940 que McCulloch, Pitts y Hebb, gracias a estudios hechos en neurociencia, crearon el primer modelo matemático en las redes neuronales biológicas de un humano, los cuales con el paso del tiempo han evolucionado y tomando una forma distinta a la que sus autores diseñaron desde el principio [3].

Todos y cada uno de los descubrimientos anteriormente mencionados, y muchos otros más, son importante para entender lo que realizará en este trabajo de tesis, gracias a ellos tenemos los fundamentos matemáticos, teórico computacionales y del funcionamiento de las neuronas biológicas para poder entenderlas y así poder simularlas en forma de unidades computacionales conocidas como redes neuronales artificiales.

## 2.2. ¿Qué es la inteligencia artificial

Para nosotros, como humanos, el término "inteligencia artificial" genera ciertas emociones, cuando escuchamos la palabra inteligencia, en ocasiones, aparecen muchas dudas en nuestra mente ¿Qué es la inteligencia?, ¿Cómo podemos medir la inteligencia?, ¿Cómo funciona el cerebro?, etc. Gracias a distintas áreas de estudio hemos podido resolver hasta cierto nivel estas dudas, las cuales juegan un gran papel al momento de entender todo lo que abarca el concepto "inteligencia artificial", o en términos más sencillos, "I.A".
Principalmente, sobre todo en las ciencias computacionales, se busca la respuesta a una sencilla pregunta ¿Cómo podemos hacer que una máquina se comporte como una persona?. Gracias a distintas películas de ciencia ficción existe cierta confusión de lo que es en realidad una IA, muchas personas creen que un día las máquinas tomarán las armas contra la humanidad, así como en Terminator, otras piensas en un robot que nos ayude a nuestras labores del hogar, como Robotina en los Supersonicos y algunas más ni siquiera sabrán de la existencia de ella. Cada quien tiene una interpretación propia de lo que es una inteligencia artificial debido a esto se vuelve complicado poder darle una definición al concepto "inteligencia artificial", sin embargo, gracias a John McCarthy en 1955 podemos encontrar una definición muy concreta, para ser más precisos nos deja en claro cuál es el objetivo real del estudio de la inteligencia artificial [3].

***"El objetivo principal de la inteligencia artificial, como un área de estudio, es desarrollar y programar máquinas de tal manera que tengan un comportamiento que nos haga pensar que son inteligentes."*** **John McCarthy 1955.** [ver 3, p 1].

Al investigar sobre IA hacemos un intento para descubrir los distintos aspectos de la inteligencia humana para así describirlos de tal manera que puedan ser simulados por una máquina. En el presente existen máquinas que aplican IA para distintas tareas, aprender a una estrategia en los juegos de mesa que dará una victoria asegurada, reconocer patrones de audio o en imágenes, procesar información del lenguaje humano, demostrar teoremas matemáticos (lo cual me hubiera servido en mis tareas de cálculo) y poder resolver ciertos problemas con ecuaciones bien formuladas (cosa que muchos de mis compañeros y yo agradecemos profundamente que sea posible). La cantidad de máquinas que pueden hacer estas tareas, sin la necesidad de intervención humana, es limitada, hasta ahora solo computadoras que cumplen con ciertas características cumplen con las tareas más sencillas de las mencionadas. Sin embargo, muchas de las máquinas modernas pueden mostrar un comportamiento similar a la inteligencia humana o superior. Otro enfoque dado a las investigaciones sobre IA, está en obtener teorías matemáticas que nos permitan describir el comportamiento de aquellas cosas, naturales o hechas por humanos, que observamos y muestran "inteligencia" y nos puedan servir para elaborar cálculos e implementarlos al diseño de una máquina inteligente [4].
La inteligencia artificial no solo es el estudio y la aplicación de algoritmos y cálculos matemáticos para que una máquina se comporte de manera inteligente, también la podemos ver como gran árbol con varias ramas de estudios, que a su vez tienen pequeñas ramitas con varias hojas o frutos. En esta tesis nos enfocaremos en la rama

de la inteligencia artificial denominada como aprendizaje de máquinas, para ser más específico en su ramita conocida como aprendizaje profundo y en su hojita, conocida mundialmente como redes neuronales artificiales. Podemos ver una representación en la figura 2.2.

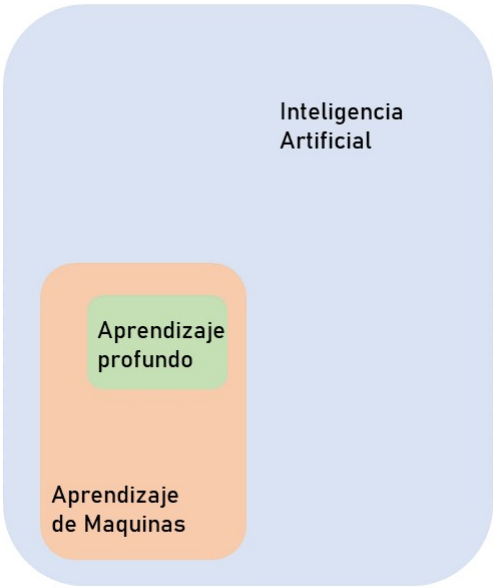



Figura 2.2: Relación de inteligencia artificial, aprendizaje de máquinas y aprendizaje profundo.

# Capítulo 3

# Aprendizaje de máquinas

Para entender por completo el objetivo de este trabajo necesitamos profundizar en lo que es la inteligencia artificial junto con sus ramas, donde daremos mayor enfoque al aprendizaje de máquinas, la cual está relacionada con el proceso de aprendizaje que tenemos los humanos y al cómo darles a las máquinas algo similar. El enfoque principal del aprendizaje de máquinas, o ML por su abreviatura en inglés, hace que las máquinas se comporten de manera inteligente mediante la adquisición de aprendizajes con la aplicación de un programa computacional "inteligente".

En el pasado, cuando hablábamos de programas inteligentes, solo podíamos hablar de un código hecho con una gran cantidad de reglas creadas con ciclos "if" y "else" que permiten el procesamiento o ajuste de los datos según las condiciones necesarias, pero al usar este tipo de configuración para nuestras reglas se nos presentan ciertas desventajas:

1. No podemos dedicar la misma lógica a distintas tareas, tendríamos que crear distintas reglas que se ajusten a la lógica de cada tarea por hacer

2. No todas las decisiones se pueden analizar desde el mismo punto de vista, usualmente es necesario de cierta experiencia para tomar un tipo de decisión

Estas desventajas mencionadas anteriormente se pueden solucionar por el uso de algoritmos de machine learning que hacen estas tareas más sencillas [7].

## 3.1. ¿Cómo aprenden las máquinas?

Para poder responder esta pregunta necesitamos definir la palabra"aprender" primero. Cuando nosotros mencionamos "aprender" en una oración, nos referimos a ganar conocimiento, ya sea por estudiar, alguna experiencia, o porque se nos enseña mediante un profesor o tercer persona. Podemos ver el aprendizaje de máquinas como el uso de algoritmos que nos permitan obtener descripción estructural de ejemplos en forma de datos (los cuales pueden tener distintos formatos). Una computadora puede aprender algo sobre las estructuras que representa la información en un conjunto de datos en crudo. Las descripciones estructurales son los modelos creados para contener la información extraída de datos en crudo y podemos usarlos para predecir datos desconocidos pero que deseemos. Estos modelos toman distintas formas y los conoceremos como algoritmos de aprendizaje de máquinas [8].
Los algoritmos de machine learning han sido un gran apoyo al momento de proveer soluciones a problemas complejos. Algunas de sus aplicaciones las vemos en nuestras actividades cotidianas y que van desde hacer una sencilla búsqueda en internet hasta el reconocimiento de un comando a un asistente inteligente en nuestro teléfono.

El campo de machine learning, o aprendizaje de máquinas, tiene como objetivo principal el estudio, diseño y aplicación de algoritmos que nos permitan automatizar la solución a problemas complejos y que son difíciles de resolver con los algoritmos estudiados por la programación convencional. En la metodología de la programación convencional debemos seguir dos pasos sencillos. El primer paso es que al toparnos con una problemática en específico haremos el diseño de nuestro programa, donde definiremos un conjunto de reglas y pasos que deben seguir cierto orden para poder obtener la mejor solución a nuestro problema, a esto lo conocemos como algoritmos. El segundo paso es la escritura de nuestro algoritmo en algún lenguaje computacional. Sin embargo, este método se vuelve difícil de hacer dependiendo de la problemáticas que se nos presenten, varios de los problemas del mundo real no poseen características ideales para poder diseñar e implementar algún algoritmo de la programación convencional, para este tipo de problemas la aplicación de los algoritmos del aprendizaje de máquinas es la mejor opción [9].

La mayor ventaja que tenemos al aplicar el aprendizaje de máquinas es que podemos obtener información sobre estructura o patrones que existen dentro de un conjunto de datos pero que no podemos ver a simple vista. También nos permite la creación de modelos, esto mediante el aprendizaje que obtiene de conjuntos de datos existentes y de esta manera podemos hacer predicciones de resultados o pronosticar algún comportamiento de nuestro objeto de estudio. Algunos de los problemas a los cuales podemos aplicar el aprendizaje de máquinas son:

1. Clasificación: tomaremos un conjunto de datos y dividirlo en una o más categorías que conoceremos como clases

2. Clustering: a una gran cantidad de datos puntuales contenidos dentro de un conjunto de datos los colocaremos en un cluster o grupo dependiendo si poseen características en común.

3. Predicción: basándonos en datos históricos podemos construir un modelo y usarlo para predecir un valor futuro [9].

## 3.2. Una aplicación del aprendizaje de máquinas

Para nosotros es sencillo reconocer a alguno de nuestros familiares, pero al momento de intentar describirlo a algún amigo, este no podrá crear una imagen exacta del familiar del que le estamos hablando, solo hasta que le mostremos una foto es que podrá reconocerlo entre una multitud.

Esto es lo que buscamos que nuestra tecnología simule. Al no ser capaz de definir ciertos objetos o definiciones, la manera en la cual podemos obtener una mayor exactitud en su aprendizaje es mediante ejemplos [6].

## 3.3. Algoritmos de aprendizaje de máquinas

Como en todas las ciencias, existen letras pequeñas en el uso de un algoritmo de machine learning. Cada uno se caracteriza por su habilidad de aprender de los conjuntos de datos disponibles, no comparten la misma forma de aprendizaje, algunos necesitan un formato único de conjuntos de datos, estos algoritmos generan modelos de aprendizaje y gracias a ellos es que podemos encontrar la solución a las distintas problemáticas. Para poder entender los distintos tipos de aprendizaje primero debemos entender que son los datos etiquetados y no etiquetados. Cuando los datos están descritos y podemos encontrar diferencias entre ellos los conoceremos como datos etiquetados, mientras que a los datos cuya descripción es muy poca o nula y no podemos encontrar diferencias a primera vista los llamaremos datos no etiquetados. Dependiendo del tipo de datos el aprendizaje será distinto [9]. Los algoritmos utilizados para el aprendizaje de máquinas son los siguientes.

- Regresión lineal.
- Regresión logística.
- Árbol de decisiones.
- Máquina de soporte de vectores
- Naive Bayes.
- Clustering.
- Vecinos Cercanos.
- Algoritmos de reducción dimensional.
- Algoritmos de aumento de gradiente.

Para poder entender cómo funciona el algoritmo de las redes neuronales artificiales necesitamos enfocarnos en los algoritmos de regresión lineal y regresión logística, y explicaremos cómo es su comportamiento.

## 3.4. Tipos de aprendizaje

Como en todas las ciencias, existen letras pequeñas en el uso de un algoritmo del aprendizaje de máquinas. Claro, es posible que nuestras máquinas aprenderán de algún ejemplo que nosotros le demos, pero no tendrán el mismo tipo de aprendizaje y esto tiene una razón, no todos los ejemplos que les mostraremos serán iguales, esto lleva a que existan distintos tipos de aprendizaje que nuestras máquinas pueden tener.

### 3.4.1. Aprendizaje supervisado

En este tipo de aprendizaje, a la máquina se le da un conjunto de datos que es una colección de ejemplos etiquetados con uno o varios vectores característicos. Esto hace a nuestro conjunto de datos como uno de datos etiquetados, ya que cada dato viene con una o varias descripciones con lo que podemos diferenciarlos del resto.
Un ejemplo, a un niño de edad temprana le damos un perro de mascota, el niño sabe que su perro tiene orejas, nariz, ojos, cuatro patas, una cola y todos con características propias que le pertenecen a un perro y lo hacen un perro. El niño al salir a la calle sin su mascota encontrará a otros perros muy diferentes del suyo, pero reconocerá las mismas características de las orejas, nariz, ojos, hocico, cola, entre todas las características y reconocerá las similitudes que tiene con su perro. Así el niño aprende que son perros y cuando se les muestre en el futuro algunas fotos de animales o se le presente con otra mascota, sabrá diferenciar que es un perro. Para que una máquina logre tener este aprendizaje no podemos darle una mascota, todavía, por ello le daremos una base de datos, formada por imágenes, que esté etiquetada como "perro" con la cual aprenda las características de un perro. La base de datos estará conformada por al menos 200 imágenes y aún así la precisión con la que aprenda la máquina será baja, por ello se recomienda una base de datos de unas 1000 imágenes, solo si son muy distintas entre sí [9].
En el aprendizaje supervisado encontramos dos problemas muy comunes, la clasificación y la regresión. La clasificación se utiliza para separar los datos en distintos conjuntos de clases o categorías, de tal manera que podamos reconocerlas de forma más sencilla. La regresión se encarga de la predicción de valores de una variable continua . La regresión es la manera en que las máquinas logran aprender de datos históricos, o datos en crudo, para encontrar relación alguna entre los mismos datos que utiliza de tal manera que podamos hacer predicciones de la variable de interés [9].

### 3.4.2. Aprendizaje no supervisado

Para este aprendizaje, a la máquina se le presenta un conjunto de datos que no están bien descritos, es más las descripciones de los datos no nos ayudan a diferenciarlos a primera vista, entonces son datos no etiquetados. En este caso es necesaria una gran cantidad de datos para que la máquina pueda encontrar algunas similitudes, el algoritmo más utilizado es el clustering o agrupamiento de datos por características similares. Todos recordamos tener un juguete que era una caja con tres o más agujeros en forma de triángulo, círculo, cuadrado y otras figuras geométricas, que incluía varios bloques de colores, pero de distinta longitud, en este caso eran pocas las características compartidas por los bloques que nos permiten

saber a qué agujero pertenecía. Nosotros reconocemos los patrones similares entre los bloques, lo mismo hacen las máquinas, encontraron patrones similares entre los datos los agrupan y generan nueva información [9].

### 3.4.3. Aprendizaje semi-supervisado

Este aprendizaje lo podemos encontrar entre los dos tipos de aprendizaje anteriores. Le damos a la máquina un gran conjunto de datos en el cual solo algunos datos están etiquetados, a los datos no etiquetados los categoriza para después recrear lo que hace con los datos etiquetados. Un ejemplo muy común de este tipo de aprendizaje es identificar si una foto tiene un perro o un gato. Debemos decirle a la computadora que representa cada foto, un gato o un perro, esto significa que este tipo de aprendizaje necesita la intervención humana para funcionar [9].

### 3.4.4. Aprendizaje reforzado

Este tipo de aprendizaje se ve involucrado en situaciones más complejas, pueden ser una situación no constante, que tiene cambios constante con respecto al tiempo, o que tienen un gran espacio de estado, tienen una cantidad casi infinita de configuraciones y soluciones [9].


(a) Conjunto de datos etiquetados. (b) Conjunto de datos no etiquetados.

Figura 3.1: Diferencia entre datos etiquetados y datos no etiquetados.

En las figuras 3.1(a) y 3.1(b) notamos la diferencia entre los tipos de datos, mientras que los datos etiquetados nos muestra características principales de cada imagen, los datos no etiquetados no dan información alguna de lo que se encuentra en la imagen, solo tiene un título.

## 3.5. Regresión lineal

El algoritmo de regresión lineal, también conocido como método de mínimos cuadrados, se utiliza para predecir valores para variables continuas, crea una relación entre las variables mediante el uso de una función lineal. Los modelos de regresión utilizan datos como variables independientes $x_1, x_2, ....x_n$ para encontrar una variable de predicción $y$ mediante la suma ponderada del producto de las variables independientes y los coeficientes $w_1, w_2, ..., x_n$ estos coeficientes determinan la contribución de las variables $x_i$ a la variable predictora $y$ [9].

$$y = w_0 + w_1x_1 + w_2x_2 + .... + w_nx_n = \sum_{i=0}^{n}(w_ix_i) \tag{3.1}$$

La ecuación 3.1 nos muestra un caso de modelo lineal con varios datos como entradas, lo cual nos sirve para una regresión lineal múltiple, en el caso de una regresión lineal simple usaremos el valor de $x_0$ igual a 1 y la variable desconocida será $x_1$, de tal manera que la ecuación para obtener una variable predictora en una regresión lineal simple tendrá la siguiente forma [8].

$$y = w_0 + w_1x_1 \tag{3.2}$$

Y su comportamiento se presenta en la gráfica siguiente.

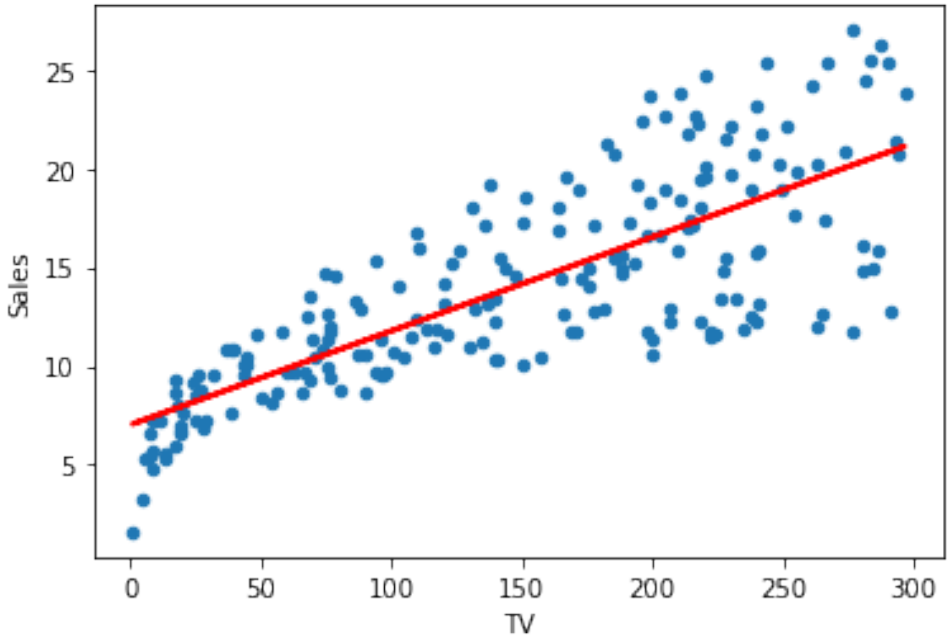



Figura 3.2: Comportamiento de función lineal.

## 3.6. Regresión Logística

El algoritmo de aprendizaje de máquinas que más se usa para las redes neuronales es la regresión logística, que es parte de los algoritmos de aprendizaje supervisado y considerado como una red neuronal sencilla. Cuando hablamos de aprendizaje supervisado podemos dividirla en dos partes, los algoritmos dedicados a la clasificación y otro apartado que conoceremos como regresión, en este último encontraremos algoritmos que nos permitan predecir valores. Puede ser sorprendente, pero la regresión logística es catalogada como un algoritmo supervisado de clasificación, todo porque se basa en un algoritmo estadístico y la comunidad de aprendizaje de máquinas lo ha utilizado como un algoritmo de clasificación [10].

Se usa a la regresión logística como un modelo binario para predecir una respuesta binaria, nos dará como resultado una variable categórica dependiente basada en una o más variables. Su ecuación tiene la forma:

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{3.3}$$

Y su comportamiento se muestra en la figura 3.3.

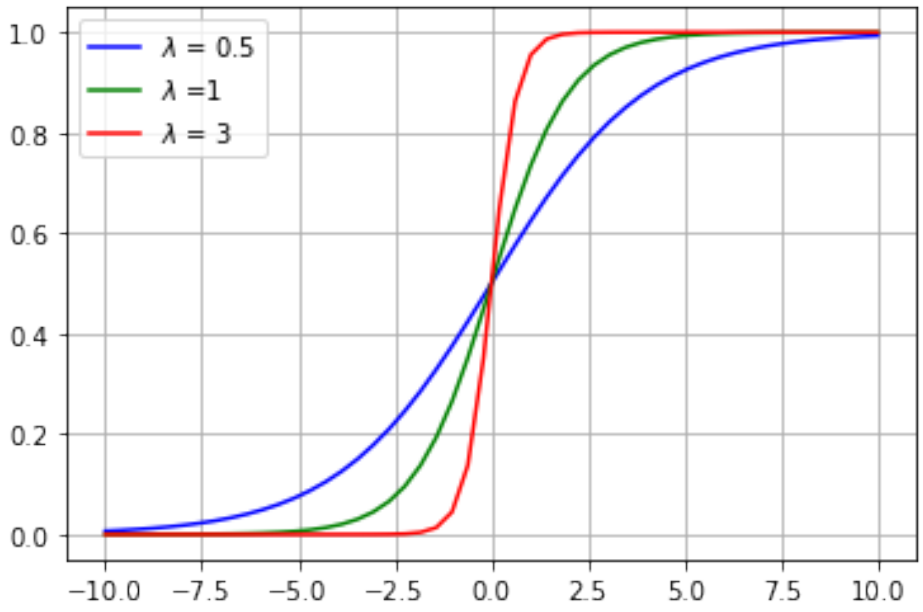



Figura 3.3: Comportamiento de función logística.

La función logística toma cualquier valor x desde menos infinito hasta infinito y nos dará resultados con valores entre 0 o 1, conocidas como variables booleanas que pueden representar “si” y “no” respectivamente, de esta manera pueden ser expresados como probabilidades [9].

# Capítulo 4

# Aprendizaje profundo

Para poder darle una definición concreta al aprendizaje profundo, primero debemos diferenciarlo de los demás enfoques que existen del aprendizaje de máquinas. Los algoritmos de programación convencionales toman un conjunto de datos que se procesan por una serie de reglas para darnos un resultado, los algoritmos del aprendizaje de máquinas toman datos y resultados para hacer reglas [2]. Viéndolo más a fondo el procedimiento de un algoritmo de aprendizaje de máquinas toma un conjunto de datos y les da una representación de acuerdo a como están ordenados, entonces ¿qué hacen los algoritmos considerados en el aprendizaje profundo?.Los algoritmos de aprendizaje profundo consisten en modelos de aprendizaje con capas consecutivas, cada una con distintas capacidades de representar los datos, lo podemos ver como un aprendizaje en jerarquía y la cantidad de capas en nuestro modelo es su profundidad. A estas representaciones de capas para el aprendizaje las conocemos como "redes neuronales artificiales" y hablaremos de ellas más adelante [2].

Dependiendo del orden de su arquitectura, podemos categorizar a una red neuronal artificial entre las siguientes clases

- Redes artificiales pre entrenadas
- Redes neuronales convolucionales
- Redes neuronales recurrentes
- Redes neuronales recursivas

Y existen múltiples variaciones que combinan las cuatro clases para obtener un resultado en específico, depende de las características de los datos y la regla que buscamos [8]. La explicación de que es una red neuronal artificial viene a continuación.

## 4.1. Redes neuronales artificiales

Las redes neuronales artificiales son un algoritmo del aprendizaje de máquinas, para ser más precisos es parte del aprendizaje profundo y está inspirado por el funcionamiento del cerebro biológico. Este algoritmo trata de imitar cómo funcionan las neuronas en un cerebro en lugar de seguir un algoritmo programado, se usa principalmente para buscar la solución de un problema mediante el aprendizaje por ejemplo y están compuestas por una gran cantidad de elementos de procesamiento interconectados que trabajan en paralelo, los cuales conoceremos como nodos [11].

## 4.2. Inspiración biológica

En nuestro cerebro (el cual es una red neuronal biológica) posee alrededor de 86 millones de neuronas conectadas entre sí, pero ¿qué son las neuronas biológicas y cómo funcionan?. Las neuronas biológicas son consideradas unidades excitables capaces de procesar y transmitir información, ya sea por señales eléctricas o químicas [8]. La arquitectura de una neurona biológica es la siguiente:

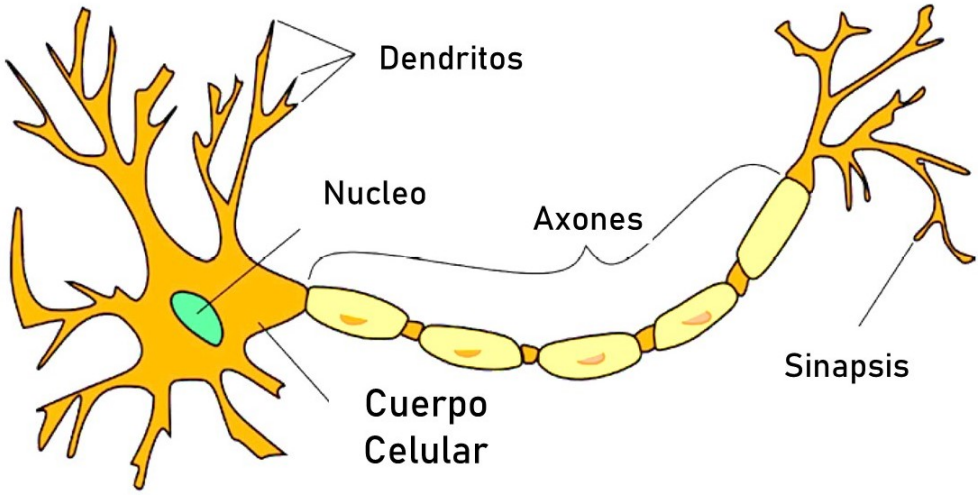



Figura 4.1: Neurona biológica.

- Dendritos
- Cuerpo celular
- Sinapsis
- Núcleo
- Axones

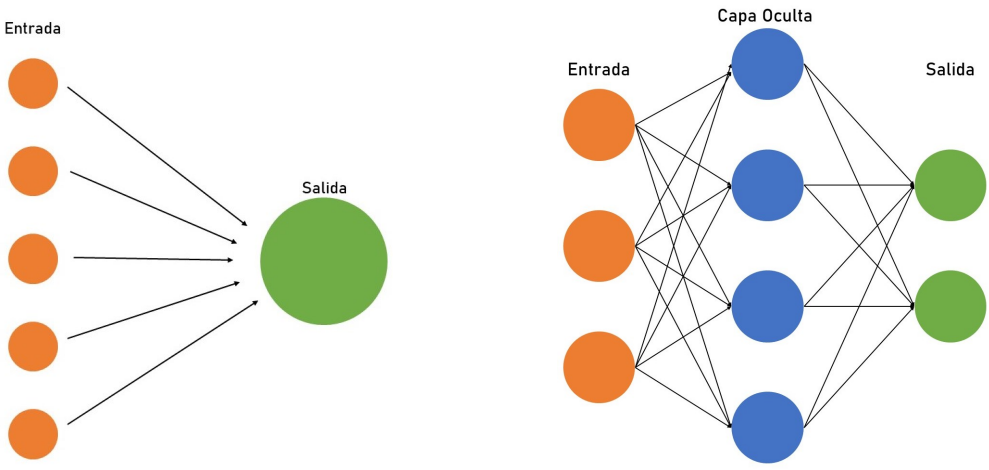
Las neuronas están conectadas una a la otra mediante axones y dendritos de tal manera que las neuronas trabajarán en conjunto, esto es lo que conocemos como una red neuronal biológica, las cuales se conectan una a la otra mediante regiones conocidas como sinapsis. Los dendritos de una neurona le permiten recibir señales de neuronas vecinas conectadas, cada dendrito es capaz de aumentar o disminuir las señales eléctricas o químicas recibidas. Los axones son fibras alargadas, más largas que los detritos, y provienen del cuerpo celular, o soma, y se conectan a los dendritos. Reciben señales en forma de impulsos electroquímicos que activan las conexiones sinápticas entre las neuronas. Las sinapsis son conexiones encargadas de mandar señales desde los axones de una neurona hasta llegar a los dendritos de otra neurona.

Las sinapsis encargadas de aumentar el potencial de un pulso electroquímico se llaman sinapsis excitadoras, mientras que las encargadas de disminuir dicho potencial son llamadas sinapsis inhibidoras. Las neuronas generan nuevas conexiones unas con otras y hasta migran de su posición original, esto permite que el proceso de aprendizaje no sea siempre constante [8].

El aprendizaje en las redes neuronales biológicas ocurre dependiendo de las fuerzas que tengan las múltiples conexiones sinápticas entre varias redes neuronales, las cuales son creadas por señales eléctricas o químicas. El objetivo principal de una red neuronal artificial es generar una función para las distintas entradas de datos, para esta tarea usará unidades computacionales, cuyo comportamiento es similar a las neuronas por lo que las llamaremos igual, que recorrerán un camino con pesos, que se comportan como las conexiones sinápticas, hasta llegar a las neuronas de salida [1].

## 4.3. La base de las redes neuronales artificiales: el perceptrón.

Podemos dividir a las redes neuronales artificiales en dos tipos, las redes neuronales artificiales de una sola capa o perceptrones, que se caracterizan por mandar valores de entrada directamente a una neurona de salida que nos dará un resultado en forma de valores numéricos, para hacer esto posible usa variaciones de alguna función lineal.
También tenemos las redes neuronales artificiales de múltiples capas, las cuales son un arreglo de varias capas ocultas por las cuales pasarán los datos de entrada hasta llegar a la neurona de salida, dependiendo de la cantidad de capas ocultas existentes en este tipo de red neuronal artificial, nuestro resultado de salida puede ser diferente y tener un costo computacional mayor o menor. También se les conoce como redes neuronales artificiales de alimentación hacia adelante [1].




(a) Red neuronal de una capa o perceptrón. (b) Red neuronal artificial de múltiples capas.

Figura 4.2: Tipos de redes neuronales artificiales.

### 4.3.1. El perceptrón simple.

Para un mayor entendimiento de la notación que se utilizará en los siguientes apartados de esta tesis vamos a mostrar una tabla donde se describe cada término: El perceptrón es la red neuronal artificial más sencilla y todas las redes neuronales

| Notación | Descripción |
|---|---|
| $x_i$ | Valores de entrada $x_1, x_2, ..., x_n$ |
| n | Número de valores de entrada |
| y | Resultado de obtenido por la neurona |
| $g(x)$ | Función de activación dentro de la neurona |
| $w_i$ | Pesos en las conexiones $w_1, w_2, ..., w_n$ |
| $t_i$ | Valor deseado de cada neurona |

Cuadro 4.1: Tabla de notación.

artificiales más avanzadas tienen como base de su funcionamiento al perceptrón, así que para entender redes neuronales artificiales más complejas necesitamos definir bien lo que es el perceptrón y cómo se comporta [8]. La arquitectura de un perceptrón está conformada por las siguientes componentes:

- **Capa de entrada**: El inicio de nuestra perceptrón, similar para todas las redes neuronales artificiales. Se encargan de recibir estímulos o información. Esta información se denota mediante valores $x_i$ y cada valor de entrada tiene un peso respectivo que lo conecta a la neurona artificial[11].

- **Pesos**: Denotados por $w_i$, son los pequeños cuadros que encontramos en las flechas que conectan a nuestra capa de entrada con la siguiente capa donde se encuentra la neurona artificial o nodo de salida [11].

- **Nodo de salida o neurona**: Es la encargada de hacer la suma ponderada de nuestra red neuronal artificial y de aplicarle una función denominada como función de activación[11].

En la figura 4.3 podemos ver la relación entre una red neuronal artificial y una red neurona biológica, esta relación se da por las similitudes en el comportamiento de cada una de las componentes de ambas redes neuronales.

Este es un modelo de red neuronal artificial de una neurona, por lo que solo tiene una capa. Similar a una neurona biológica, el perceptrón recibirá información de entrada de la forma $\bar{X}$=[$x_1$,..,$x_n$] para poder emitir un resultado de salida y. A cada valor de entrada en $\bar{X}$ se le asigna un peso de $\bar{W}$=[$w_1$,...,$w_n$], los pesos son los parámetros de nuestro modelo, el valor de salida es una función dependiente del valor de "x,w" y de la función de activación que la neurona aplique a la suma pondera [1].
Ahora, nuestro resultado deseado $\hat{y}$ será dado por las categorías a las que pertenezcan nuestros datos, para obtener un valor cercano al valor $\hat{y}$ y usaremos la función de Heaviside aplicada a la suma ponderada que se calculó en nuestra neurona.

$$y = g\left(\sum_{i=1}^{n} w_i x_i\right) \tag{4.1}$$

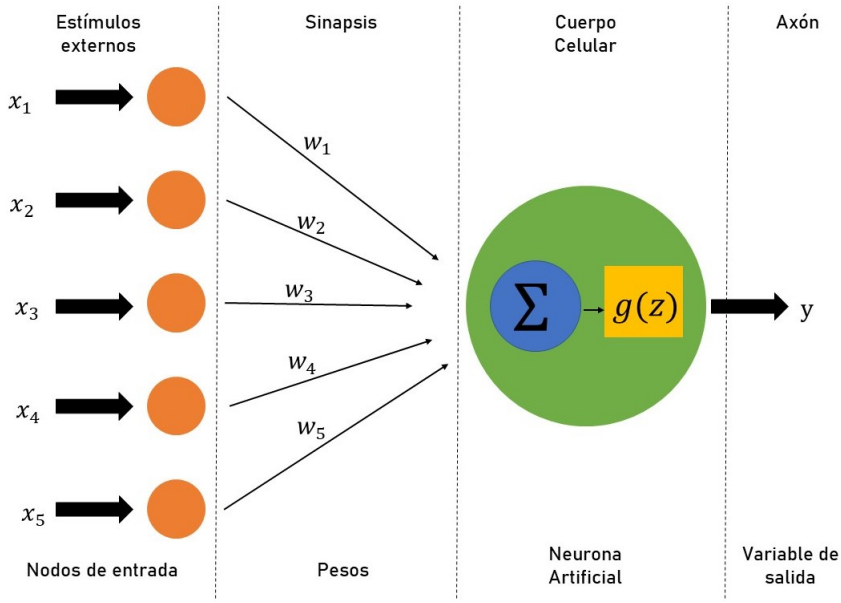



Figura 4.3: Arquitectura del perceptrón y su relación con las partes de una red neuronal biológica.

Considerando la entrada de un nodo de sesgo, obtenemos una predicción invariante conocida como sesgo, esto permitirá a la suma ponderada tomar la siguiente forma:

$$y = g\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4.2}$$

El nodo de sesgo, o neurona de sesgo, transmite un valor constante (siempre será 1) al nodo de salida. Esto hace que su peso siempre tenga el valor $b$ [9].

Debemos considerar la existencia de un error al momento de implementar un algoritmo de aprendizaje con el perceptrón, comparamos el resultado obtenido $y_i$ con el resultado objetivo $\hat{y}$, este error lo obtenemos mediante la función de error cuadrado.

$$E = \frac{1}{2}\sum_{i=1}^{n} (t_i - y_i)^2 \tag{4.3}$$

Esto nos permitirá calcular el error de cada nodo en nuestra red neuronal artificial.

### 4.3.2. El aprendizaje mediante un perceptrón

El algoritmo de aprendizaje contenido en un perceptrón modifica los pesos en el mismo perceptrón hasta que todos los valores están correctamente clasificados o el valor del error obtenido sea mínimo. El algoritmo finaliza únicamente cuando nuestro conjunto de datos esté separado en dos clases, esto es conocido como una separación no lineal. Los pesos dentro de nuestro perceptrón empiezan el entrenamiento con valores aleatorios y calcula un valor de resultado para que sea comparado contra el valor deseado de la categoría [8].

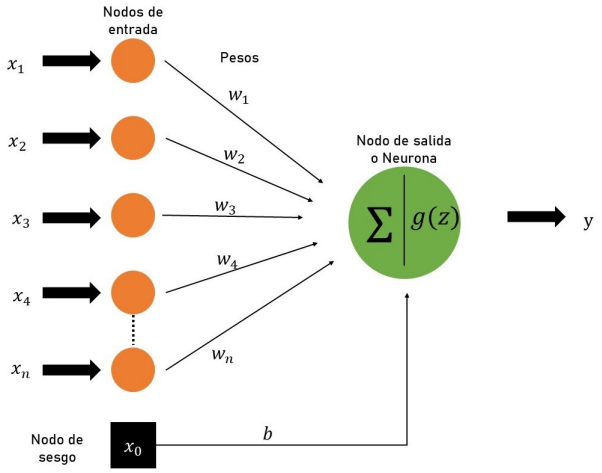



Figura 4.4: Arquitectura básica de un perceptrón simple con nodo de sesgo.

## 4.4. Red neuronal artificial multicapa

Este tipo de redes neuronales artificiales están conformadas por más de una capa computacional, mientras que el perceptrón tiene una capa de entrada y una capa de salida, de los cuales la capa de salida es la única que hace cálculos. Las redes neuronales artificiales de múltiples capas incluyen capas consecutivas entre la capa de entrada y la de salida, a estas capas las conocemos como capas escondidas, esto porque los cálculos no son vistos a simple vista por el usuario. Su arquitectura es considerada de propagación hacia adelante, esto porque los datos se propagan de manera sucesiva hacia adelante de capa en capa y los todos los nodos de una capa deben estar conectados a los nodos de su capa sucesiva.

La arquitectura está casi completa una vez que declaramos; la cantidad de capas y la cantidad de nodos por capa, por ello no todas las redes neuronales artificiales tienen la misma arquitectura. Lo único que falta para terminar nuestra red neuronal artificial de múltiples capas es definir nuestra función de error que se ve optimizada en la capa de salida [1].

De la misma manera que hicimos en el caso de la notación descrita en el perceptrón, para las redes neuronales artificiales de múltiples capas definiremos la notación que se utilizará en este trabajo de tesis:

| Notación | Descripción |
|---|---|
| $z_i^{(P)}$ | Suma ponderada por neurona i-ésima en la capa P |
| $g_i^{(P)}$ | Función de activación en neurona i de la capa P |
| $a_i^{(P)}$ | Valor de activación la neurona i de la capa P |
| $w_{ji}^{(P)}$ | Peso que va de capa de entrada i a la capa oculta j |
| $A^{(P)}$ | Vector de valores de activación por capa P |
| $W^{(P)}$ | Matriz de pesos por cada capa P |
| $t_i$ y $y_i$ | Valor deseado y obtenido por cada neurona de salida |
| $\hat{y}$ | Valor total de salida de la red neuronal artificial |

Cuadro 4.2: Notación para red neuronal artificial de múltiples capas.

Procederemos a dar explicación a cada una de las descripciones de la tabla anterior

acompañadas de las ecuaciones necesarias para cada una con el ejemplo en la figura 4.5 que tiene dos capas, cada una con dos neuronas:

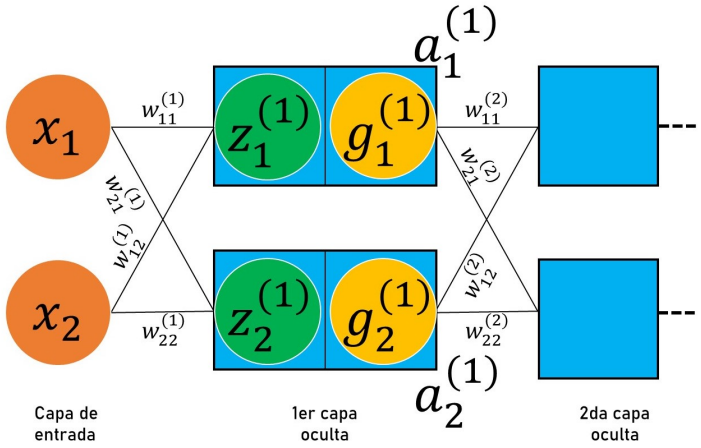



Figura 4.5: Ejemplo de red neuronal artificial de dos capas con dos neuronas.

Primero veremos la forma de la matriz de los pesos que conecta a la capa de entrada con la primer capa oculta[10]:

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \tag{4.4}$$

Las sumas ponderadas ($z_1^{(1)}$ y $z_2^{(1)}$) dentro de las neuronas tienen las siguientes formas:

$$z_1^{(1)} = x_1 w_{11}^{(1)} + x_1 w_{21}^{(1)} \tag{4.5}$$

$$z_2^{(1)} = x_2 w_{12}^{(1)} + x_2 w_{22}^{(1)} \tag{4.6}$$

Al aplicar las funciones de activación de las neuronas de nuestra primer capa oculta ($g_1^{(1)}$ y $g_2^{(1)}$) obtendremos los valores de activación $a_1^{(1)}$ y $a_2^{(1)}$ que tienen la forma

$$a_1^{(1)} = g_1^{(1)}\left(z_1^{(1)}\right) = g_1^{(1)}\left(x_1 w_{11}^{(1)} + x_1 w_{21}^{(1)}\right) \tag{4.7}$$

$$a_2^{(1)} = g_2^{(1)}\left(z_2^{(1)}\right) = g_2^{(1)}\left(x_2 w_{12}^{(1)} + x_2 w_{22}^{(1)}\right) \tag{4.8}$$

Nuestro resultado en la red neuronal artificial se vería de la siguiente manera:
Ahora las sumas ponderadas en la segunda capa oculta sumarán los productos de los valores de activación con los pesos en la matriz que conecta a la 1er capa oculta con la 2da capa oculta, nuestra matriz de pesos que conectan estas capas ocultas tiene la forma[10]:

$$\mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} \tag{4.9}$$

Lo siguiente que mostraremos será los valores de las sumas ponderadas en las neuronas de la segunda capa oculta:

$$z_1^{(2)} = a_1^{(1)} w_{11}^{(2)} + a_1^{(1)} w_{21}^{(2)} \tag{4.10}$$

$$z_2^{(2)} = a_2^{(1)} w_{12}^{(2)} + a_2^{(1)} w_{22}^{(2)} \tag{4.11}$$

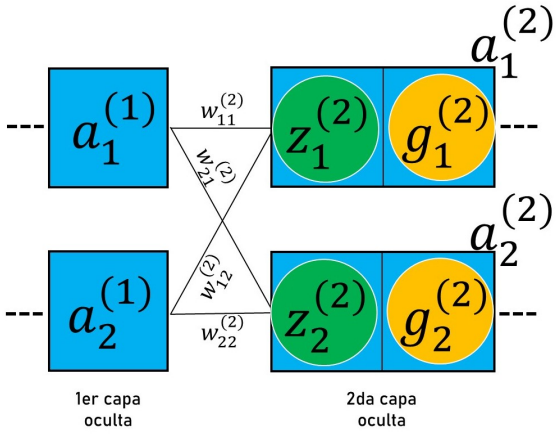



Figura 4.6: Valores de activación de la 1er capa oculta conectados a la 2da capa oculta.

En este caso los valores de activación harán el papel de los valores de entrada. Para obtener los valores de activación de la segunda capa oculta solo aplicaremos las funciones de activación $g_1^{(2)}$ y $g_2^{(2)}$ a las sumas ponderadas $z_1^{(2)}$ y $z_2^{(2)}$ respectivamente.

$$a_1^{(2)} = g_1^{(2)}\left(a_1^{(1)}w_{11}^{(2)} + a_{1^{(1)}}w_{21}^{(2)}\right) \tag{4.12}$$

$$a_2^{(2)} = g_2^{(2)}\left(a_2^{(1)}w_{12}^{(2)} + a_2^{(1)}w_{22}^{(2)}\right) \tag{4.13}$$

Así el resultado final de la red neuronal artificial de la figura 4.5 será [8]:

$$y = \left(a_1^{(2)}, a_2^{(2)}\right) \tag{4.14}$$

## 4.5. Funciones de error en una red neuronal artificial de múltiples capas

Para saber si el entrenamiento de nuestra red neuronal artificial es exitoso necesitamos obtener el valor de la función de error (E)de nuestra red neuronal. Primero debemos considerar los errores locales de cada neurona, a los errores locales los obtendremos de la resta del valor deseado por neurona $t_i$ menos el valor obtenido por cada neurona de cada capa $y_i^{(P)}$ (este valor lo podemos ver como los valores de activación en el caso de tener múltiples capas). Así el error local por neurona es:

$$\left(t_i - y_i^{(P)}\right)^2 \tag{4.15}$$

Existen varias maneras de calcular el error:

- **Error cuadrático medio**(MSE): En este tipo de error tomamos la suma ponderada de los errores locales y obtenemos su promedio [11].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(t_i - y_i^{(P)}\right)^2 \tag{4.16}$$

- **Error cuadrático**(SE):Este caso es similar al error cuadrático medio, pero ahora dividiremos la suma ponderada en dos [11].

$$SE = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - y_i^{(P)}\right)^2 \tag{4.17}$$

- **Raíz media cuadrática**(RMS): Ahora aplicamos una raíz cuadrada al error cuadrático medio [11].

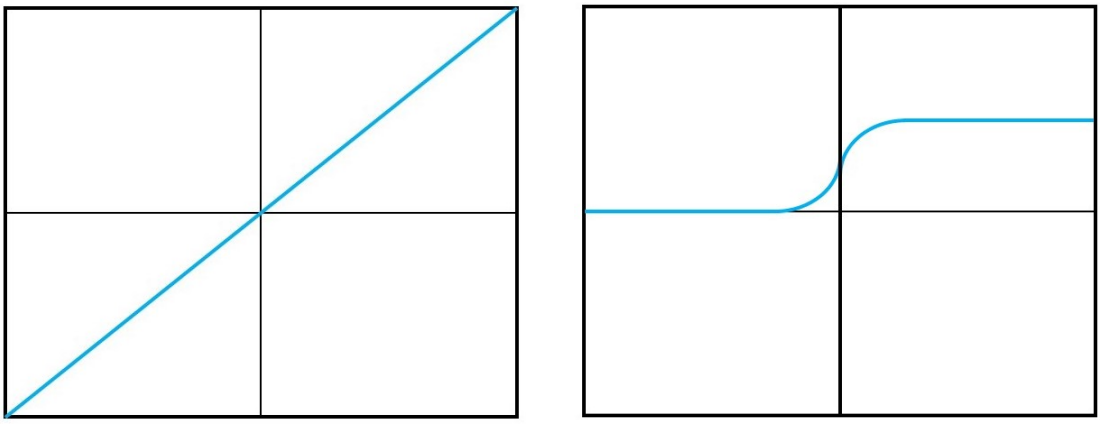
$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(t_i - y_i^{(P)}\right)^2} \tag{4.18}$$

- **Suma de errores cuadrados**(SSE): Solo se hará una suma de los errores locales [11].

$$SSE = \sum_{i=1}^{n}\left(t_i - y_i^{(P)}\right)^2 \tag{4.19}$$

## 4.6. Funciones de activación

Usamos las funciones de activación $g_i^{(P)}$ para poder obtener los valores de activación $a_i^{(P)}$, estos pasarán de las neuronas en una capa $P$ a la siguente capa $P+1$. Existen distintas funciones de activación cada una con una aplicación adecuada.



(a) Transformación lineal.

(b) Transformación sigmoidal.

Figura 4.7: Comportamiento de transformaciones lineal y sigmoidal.

- **Transformación lineal**: Esta función la podemos ver como una función identidad g(x)=Wx [11].

- **Transformación sigmoidal**: Así como la regresión logística la función sigmoidal nos permite reducir valores al intervalo (0,1) sin necesidad de borrar datos [11].

- **Transformación tangente hiperbólica**: Similar a la transformación sigmoidal pero ahora estará en un rango numérico (-1,1) y lo obtenemos con la ecuación [11].

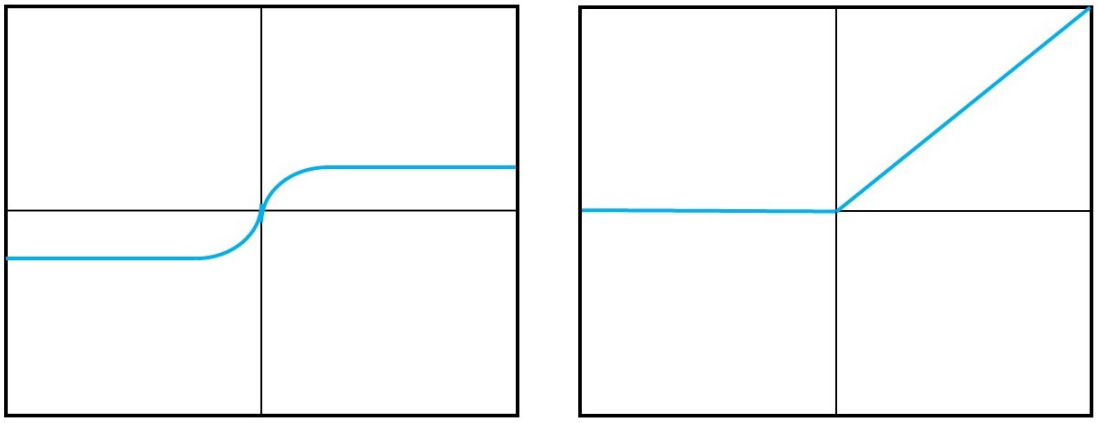
$$tanh(x) = \frac{sinh(x)}{cosh(x)} \tag{4.20}$$

  Su ventaja sobre todas las demás funciones es que puede manejar mejor a los números negativos [11].

- **Softmax**: la activación softmax es similar a la regresión logística pero ahora podemos tener múltiples fronteras en la decisión.

- **Rectificador**: Esta activación activa a los nodos sólo si los valores de entrada son mayores que cierta cantidad. Mientras que el valor sea menor a cero, será mandado a cero, si el valor del resultado es mayor a cierto umbral, su relación lineal está dada por la ecuación.

$$g(x) = max(0, x) \tag{4.21}$$



(a) Transformación tangencial hiperbólica. (b) Transformación lineal rectificada.

Figura 4.8: Comportamiento de transformaciones tangencial hiperbólica y lineal rectificada.

## 4.7. Actualización del valor de los pesos

Ya presentados los conceptos de error y pesos de una red neuronal artificial, podemos llegar al método de aprendizaje de una red neuronal. La propagación de los datos en una red neuronal artificial de múltiples capas tiene dos sentidos:

- **Propagación hacia adelante**: Conocida también como feedforward. En esta propagación es que son introducidos los datos de entrada que pasan de manera ordenada por cada nodo de cada capa oculta hasta llegar a la capa final o capa de salida. Gracias a esta propagación obtenemos el valor neto de nuestra red neuronal y su función de error neto [11]. Pero ¿cómo hacemos que nuestro error sea mínimo?

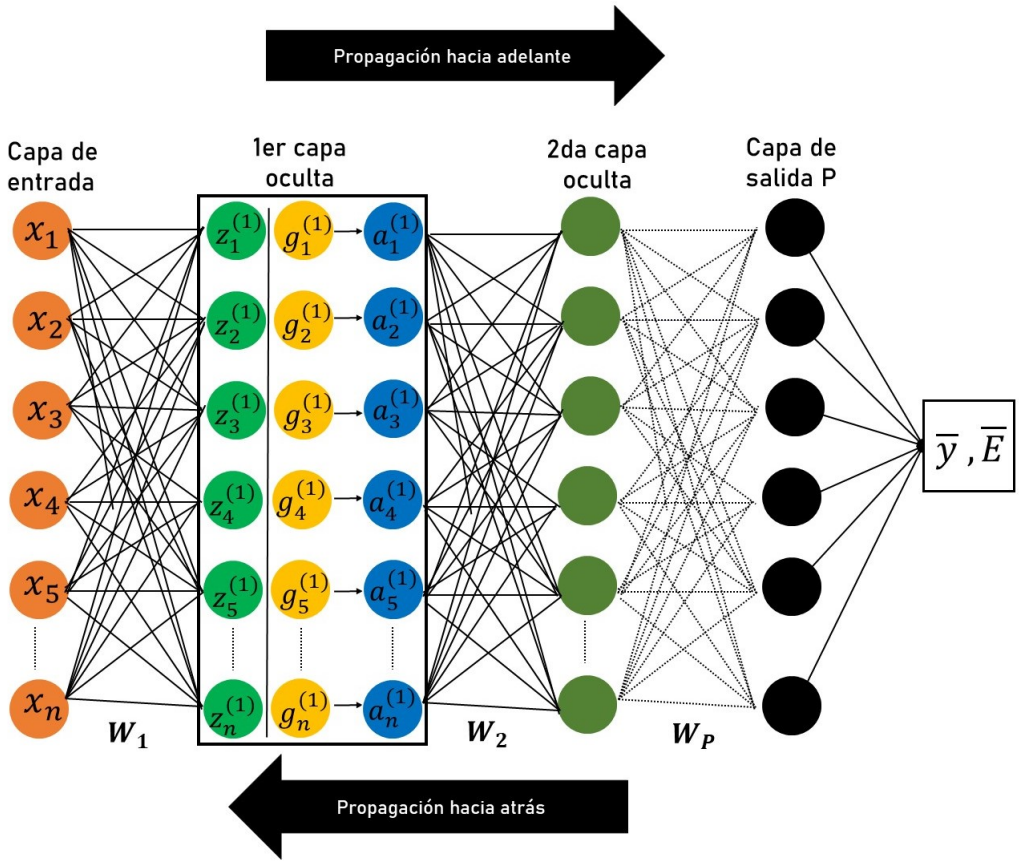



Figura 4.9: Propagación hacia adelante y hacia atrás.

- **Propagación hacia atrás**: Más conocido como algoritmo de backpropagation. Este es el método que utilizan las redes neuronales artificiales para aprender de los datos, en esta caso comparan el valor deseado con el valor obtenido por cada neurona y en lugar de hacer el cálculo de cada camino entre las capas, va a buscar al neurona que provoque un mayor error en cada capa[8].

### 4.7.1. Propagación hacia atrás

Para actualizar los valores de los pesos que se encuentran en las conexiones entre capa y capa

$$w_{nuevo} = w_{anterior} - \eta \nabla E \tag{4.22}$$

Para actualizar los valores de los pesos que se encuentran en las conexiones entre capa y capa se utiliza el algoritmo de propagación hacia atrás. Al obtener un error de tamaño considerable nuestra red neuronal artificial mandará una señal donde nos dirá que el resultado no será muy preciso, esto provocará un paso hacia atrás, capa por capa, de nuestra red neuronal artificial, todo esto para encontrar al peso y a la neurona responsables de dicho error. Pero ¿cómo encontramos a esta neurona?. La respuesta es sencilla, debemos conocer cómo cambia el valor del error con respecto a los pesos, matemáticamente lo vemos como una derivada parcial, similar al comportamiento del descenso del gradiente. Claro que debemos considerar que la derivada

es respecto a algún peso $w_{ji}$ que encontramos en la capa $P$

$$\frac{\partial E}{\partial w_{ji}^{(P)}} \tag{4.23}$$

Pero esta derivada lleva muchos más cálculos, primero debemos entender la composición de la función de error. Primero la neurona aplica la suma ponderada al producto de los valores de entrada y valores de activación con los peso, esto haría una función $z_i^{(P)}(x_i, w_{ji}^{(P)})$. El paso siguiente es la aplicación de la función de activación para obtener los valores de activación, esto es $a_i^{(P)}(z_i^{(P)}(x_i, w_{ji}^{(P)})) = g(z_i^{(P)}(x_i, w_{ji}^{(P)}))$ y para finalizar aplicamos la función error $E(a_i^{(P)})$. Por lo tanto la composición de nuestra función de error es la siguiente.

$$E(a_i^{(P)}(z(a_i^{(P-1)}, w_{ji}^{(P)}))) \tag{4.24}$$

Usando la regla de la cadena para derivadas parciales obtenemos la derivada siguiente:

$$\frac{\partial E}{\partial w_{ji}^{(P)}} = \frac{\partial E}{\partial a_i^{(P)}} \frac{\partial a_i^{(P)}}{\partial z_i^{(P)}} \frac{\partial z_i^{(P)}}{\partial w_{ji}^{(P)}} \tag{4.25}$$

Hagamos derivada por derivada, empezando por $\partial E_{a_i^{(P)}}$, primero debemos definir nuestra función de error, la que se utilizará para el ajuste realizado en este trabajo de tesis es la función de error cuadrático:

$$SE = \frac{1}{2} \sum_{i=1}^{n} \left( t_i - a_i^{(P)} \right)^2 \tag{4.26}$$

En este caso utilizamos los valores de activación $a_i^{(P)}$ que juegan el papel de los valores de resultado $y_i^{(P)}$ de un perceptrón en el caso de una red neuronal artificial multicapa. Con esto queremos decir que en el caso de una red neuronal artificial multicapa que los valores de activación son los valores de resultado de cada neurona.

$$a_i^{(P)} = y_i^{(P)} \tag{4.27}$$

Al hacer la derivada $\partial E_{a_i^{(P)}}$ de nuestra función de error cuadrático obtenemos

$$\frac{\partial E}{\partial a_i^{(P)}} = \frac{\partial}{\partial a_i^{(P)}} \left( \frac{1}{2} \sum_{i=1}^{n} \left( t_i - a_i^{(P)} \right)^2 \right) = \left( a_i^{(P)} - y_i^{(P)} \right) \tag{4.28}$$

El paso siguiente será calcular la derivada parcial de la activación con respecto a la suma ponderada, pero esto depende de la función que utilizaremos, para dejarlo en un caso general definiremos al error de salida por neurona en cada capa, esto es:

$$\delta_i^{(P)} = \frac{\partial E}{\partial a_i^{(P)}} \frac{\partial a_i^{(P)}}{\partial z_i^{(P)}} \tag{4.29}$$

Por lo tanto el cambio del error $E(a_i^{(P)}(z(x_i, w_{ji}^{(P)})))$ con respecto al valor de los pesos $w_{ji}^{(P)}$ será definida en la ecuación 4.28.

$$\frac{\partial E}{\partial w_{ji}^{(P)}} = \delta_i^{(P)} \frac{\partial z_i^{(P)}}{\partial w_{ji}^{(P)}} \tag{4.30}$$

Así que para casos generales solo necesitamos obtener el valor de $\partial z_{a_i^{(P)}}$, esto es:

$$\frac{\partial z_i^{(P)}}{\partial w_{ji}^{(P)}} = \frac{\partial}{\partial w_{ji}^{(P)}} \left( \sum_{i=1}^{n} w_{ji}^{(P)} x_i^{(P)} \right) = a_i^{(P-1)} \tag{4.31}$$

Así, después de tantos cálculos complicados, podemos definir al cambio del error con respecto a los pesos de la siguiente manera.

$$\frac{\partial E}{\partial w_{ji}^{(P)}} = \delta_i^{(P)} a_i^{(P-1)} \tag{4.32}$$

Existe otro "*pero*" en esta situación, el anterior es el caso donde nos encontramos en la última capa de la red neuronal artificial de múltiples capas.

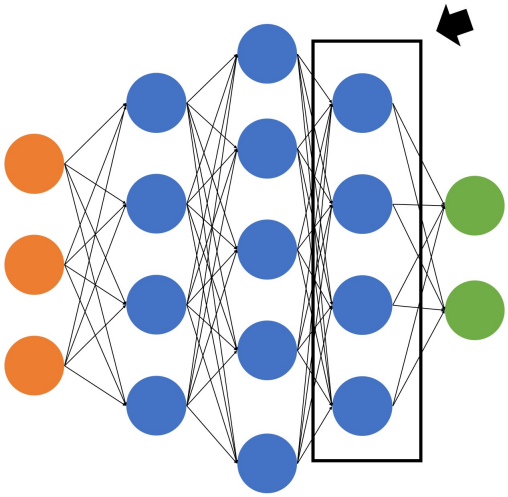


Figura 4.10: Caso de propagación hacia atrás en la última capa oculta.

Pero ¿qué pasa en casos donde deseemos ver las demás capas ocultas?, este caso lo podemos ver en la imagen 4.11 e igual usaremos derivadas parciales y la regla de la cadena, relacionando ahora con la capa oculta $P-1$.

Empecemos mostrando los pasos matemáticos de las derivadas parciales, podemos partir desde la última capa, solo tendremos que agregar más derivadas parciales. Primero debemos encontrar la composición de la función para la anterior capa oculta.

$$E\left(a_i^{(P)}\left(z_i^{(P)}\left(w_{ji}^{(P)}, a_i^{(P-1)}\left(w_{ji}^{(P-1)}, a_i^{(P-2)}\right)\right)\right)\right) \tag{4.33}$$

Si deseamos calcular las derivadas parciales de la propagación hacia atrás y obtener el cambio del error con respecto a los pesos de la capa $P-1$ definimos lo siguiente:

$$\frac{\partial E}{\partial w_{ji}^{(P-1)}} = \frac{\partial E}{\partial a_i^{(P)}} \frac{\partial a_i^{(P)}}{\partial z_i^{(P)}} \frac{\partial z_i^{(P)}}{\partial a_i^{(P-1)}} \frac{\partial a_i^{(P-1)}}{\partial z_i^{(P-1)}} \frac{\partial z_i^{(P-1)}}{\partial w_{ji}^{(P-1)}} \tag{4.34}$$

Con las mismas definiciones utilizadas en el caso anterior tenemos la siguiente ecuación:

$$\frac{\partial E}{\partial w_{ji}^{(P-1)}} = \delta^{(P-1)} a_i^{(P-2)} \tag{4.35}$$

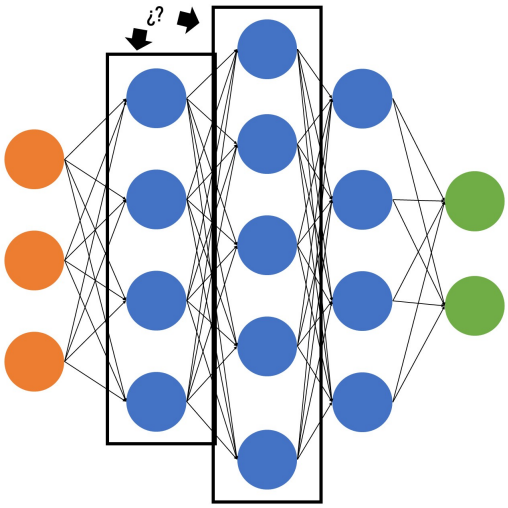



Figura 4.11: Caso de propagación hacia atrás en las demás capas ocultas.

Donde $\delta^{(P-1)}$ es definida de la siguiente manera:

$$\delta^{(P-1)} = \frac{\partial E}{\partial a_i^{(P)}} \frac{\partial a_i^{(P)}}{\partial z_i^{(P)}} \frac{\partial z_i^{(P)}}{\partial a_i^{(P-1)}} \frac{\partial a_i^{(P-1)}}{\partial z_i^{(P-1)}} = \delta^{(P)} \frac{\partial z_i^{(P)}}{\partial a_i^{(P-1)}} \frac{\partial a_i^{(P-1)}}{\partial z_i^{(P-1)}} \tag{4.36}$$

La ecuación 4.33 es un caso general para poder obtener el error respecto a los pesos de cualquier capa deseada, solamente debemos ajustarnos de acuerdo a la composición de la función desde los pesos deseados[10][11].

## 4.8. Red Neuronal Convolucional

Son un tipo de redes neuronales artificiales de múltiples capas que están diseñadas para trabajar con entradas con una estructura de cuadrícula, un ejemplo de este tipo de entradas son imágenes dos dimensionales. Las imagenes presentan dependencias espaciales en regiones locales de la misma cuadrícula, las regiones espaciales adyacentes en una imagen tienen valores de color similares de los píxeles RGB individuales. Una dimensión adicional captura las diferentes tonalidades de color, dandole así un volumen a la misma imagen. Las imágenes tienen una propiedad única entre los datos con estructuras de cuadrícula, los datos no cambiarán si rotamos o trasladamos las imagénes, los datos siempre se verán de la misma manera sin importar la orientación de la imagen. También conocidas como ConvNets, tienen el objetivo de reconocer y aprender de las características de orden superior en los datos a través de convoluciones [8]. La convolución es una operación matemática que aplica el producto punto de un conjunto de pesos con una estructura de cuadrícula con un conjunto de datos con la misma estructura obtenida de las diferentes regiones espaciales que podemos encontrar en los datos de entrada [1].

### 4.8.1. Convolución

Esta operación nos sirve para unir dos conjuntos de datos. Toma los datos, les aplica un filtro de convolución y arroja un mapeado de características como resultado. Existen distintos tipos de filtros que se utilizan para obtener las distintas características de nuestra imagen[8].

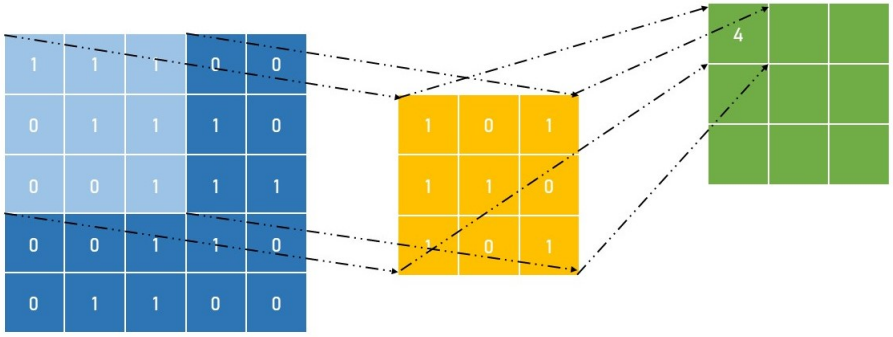



Figura 4.12: Inicio de una convolución de bordes.

### 4.8.2. Arquitectura de una red neuronal convolucional

De forma similar al perceptrón y a las redes neuronales artificiales de múltiples capas, debemos entender la arquitectura de una red neuronal convolucional.

- **Capas de entrada**: En estas capas cargamos los datos en forma de imágenes que tienen las características de ancho y alto especificadas y también incluyen la información de los canales RGB por cada píxel [8].

- **Capas convolucionales**: Son las encargadas de aplicar la convolución a nuestros datos obtenidos de la descomposición de una imagen a datos. Calculará un producto escalar entre las neuronas de la capa de entrada y los pesos que los conectan localmente a las neuronas en la capa de salida [8].

- **ReLU**: En las redes neuronales convolucionales encontramos una capa dedicada a la rectificación lineal unitaria, la cual aplicará la función de activación ReLU por cada elemento que se encuentre por encima del valor del umbral, de tal manera que obtendremos valores de misma dimensión tanto en entrada como en salida [1].

- **Capas de unión**: Es necesario colocar a las capas de unión entre capas convolucionales consecutivas, esto se hace para reducir las dimensiones espaciales ancho y altura de la imágenes que utilizamos como datos. Las capas de unión se encargan de reducir los datos representados de manera progresiva para evitar sobre ajustes, para hacer esto usan la operación max pooling, que es un filtro de 2x2, se aplica en un área de 4 números y se escoge el mayor para representar esa área, dicha operación no afecta a la profundidad de las imágenes [1].

$$Softmax = \frac{e_j^Y}{\sum_{k=1}^{K}(e_k^Y)} \tag{4.37}$$

- **Conexión completa**: Para finalizar el proceso es necesario hacer los cálculos de los valores de clases, estos serán nuestros resultados de la red neuronal artificial. La dimensión del resultado es [1x1xN], N es el número de clases que estamos evaluando [8].

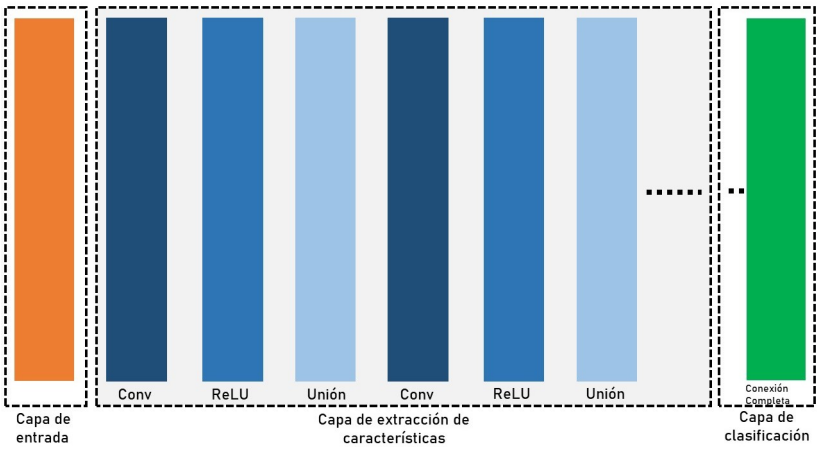



Figura 4.13: Arquitectura de una red neuronal convolucional.

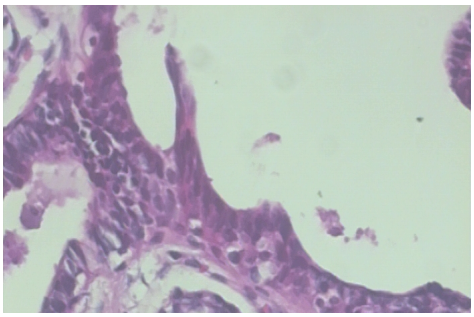
# Capítulo 5

# Procesamiento de imágenes y manejo de datos

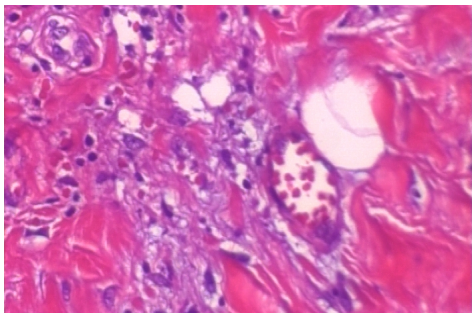
En este apartado explicaremos dos partes importantes para la realización del algoritmo y el código en python. El primero será la manera en que se usó del conjunto de datos, que son obtenidos de un conjunto de imágenes y el segundo será la obtención de datos numéricos mediante el procesamiento de las mismas imágenes, este proceso es hecho de manera automática gracias al comando Imagedatagenerator(), parte del módulo de diferenciación automática TensorFlow, el cual se utilizó para en el código.

Agradecemos al Laboratório Visao Robótica e Imagem de la UFPR, por la base de datos de imágenes que nos proporcionó para entrenar, validar y probar nuestra red neuronal.

## 5.1. Imágenes histopatológicas



(a) Imagen con diagnóstico benigno.



(b) Imagen con diagnóstico maligno.

Figura 5.1: Comportamiento de transformaciones tangencial hiperbólica y lineal rectificada.

Para entender qué son las imágenes histopatológicas primero debemos hablar de la histopatología. Es una rama de la anatomía patológica que busca dar diagnóstico de alguna enfermedad mediante el estudio de tejidos, pueden ser de órganos, tumores, entres otros. A estos tejidos se les aplican ciertos químicos, cómo cloroformo o colorante. Después se observan a distintas escalas en un microscopio, con distintos aumentos, de tal manera que podamos distinguir con precisión todo lo que tiene un tejido y reconocer aquellas partes del tejido están dañadas o no deberían estar en él, así es que se diagnostica una enfermedad. Los tumores tienen muchas categorías en los cuales podemos colocarlos, dependiendo de las observaciones hechas en los tejidos extraídos de los mismos, pero en este trabajo de tesis nos enfocaremos más en entrenar un modelo que de un diagnóstico si un tumor es maligno o benigno. A grandes rasgos podemos destacar tres características que nos permitirán reconocer a qué categoría pertenece cada imagen:

- El tamaño de las células con un tejido de diagnóstico maligno son de mayor tamaño que en un tejido de diagnóstico benigno.
- La coloración de un tejido maligno tiene tonos más oscuros.
- Podemos distinguir el núcleo de la célula cuando el tejido tiene un diagnóstico maligno.

## 5.2. Lectura de las imágenes

Una gran diferencia que tienen los humanos y las computadoras es la perspectiva de las imágenes, nosotros podemos diferenciar entre las distintas figuras, los múltiples colores e identificar distintos objetos, todo esto nos permite darle características propias a una imagen. En el caso de las computadoras su forma de ver una computadora es distinta, es más, no podemos decir que ven una imagen, más bien que procesan la información de la misma. Para entender este proceso mostraremos los pasos mediante imágenes y diagramas y también veremos como es el procesamiento de imágenes a datos con un código en python. Para dicho proceso usaremos los modulos de Computer Vision y Matplotlib.

Primero empiezaremos dividiendo la altura y el ancho de la imagen en pixeles, los cuales contienen información de la intensidad y los tonos exitentes de alguno de los canales de color que conforman a ese pixel, los canales rojo, verde y azul (RGB) los más utilizados. Veremos la parte de código utilizado para ver la altura y el ancho de una imagen al igual que su volumen o cantidad de canales de color, poer ejemplo, a una imagen de nuestra carpeta de pruebas, aplicaremos el siguiente código:

```
image = cv2.imread('C:/Users/octav/OneDrive/Documentos/Tesis red
    neuronal/RN1BC/validation/malign/SOB_M_DC-14-4372-40-010.png')
plt.imshow(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
plt.show()
```

Con a esta parde del código obtuvimos los siguente, la imagen tiene una altura de 460 pixeles, un ancho de 700 pixeles y un volumen de 3 canales. Gracias a esto podemos agregarle a nuestras imagenes la propiedad de volumen, que se refiere a los canales de color en los que se presente cada imagen.

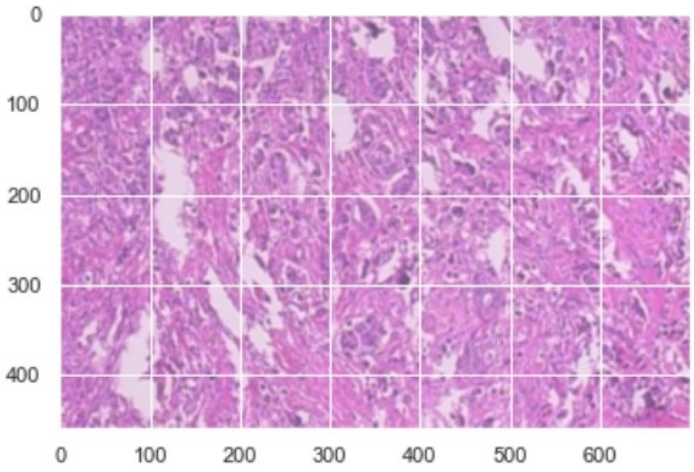



Figura 5.2: Pixeles de altura y ancho de imagen.

Al introducir una imagen a una red neuronal convolucional es imposible que trabaje con un volumen mayor a uno, esto significa que le será imposible trabajar con los canales RGB en los que nuestra imagen se presenta. Por ello se implementa un conversión de RGB a escala de grises y la información de los canales RGB que encontramos en formato de bits se convertirá de la misma manera para tener solamente un canal de grises con una matriz numérica respectiva. La conversión a escalas de grises se hace de la siguiente manera:

```
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray,cv2.COLOR_BGR2RGB))
plt.show()
```

Ahora la conversión a escala de grises obtenemos un canal y se respeta la dimensión (460, 700)pixeles, ahora habrá solo un canal, el gris, en lugar de los canales rojo, verde y azul. Es sencillo solo ver una imagen, pero una red neuronal convolucional

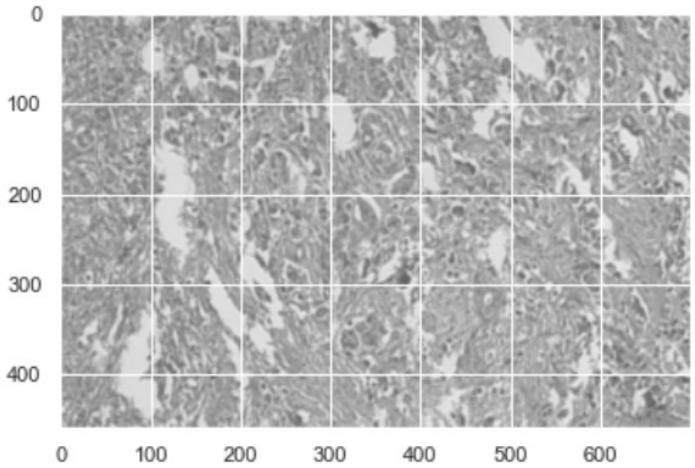



Figura 5.3: Imagen en escalas de grises.

necesita de valores numéricos que se encuentran en los pixeles, estos son dados por los bits de cada pixel. Veremos el código que nos permitirá ver los valores numéricos en un arreglo numpy.

```
data = np.array(gray)
flattened = data.flatten()
```

Es arreglo tiene un total de 322000 valores, el cual es el producto de 460 con 700. Algunos de los datos numéricos se presentarán a continuación.

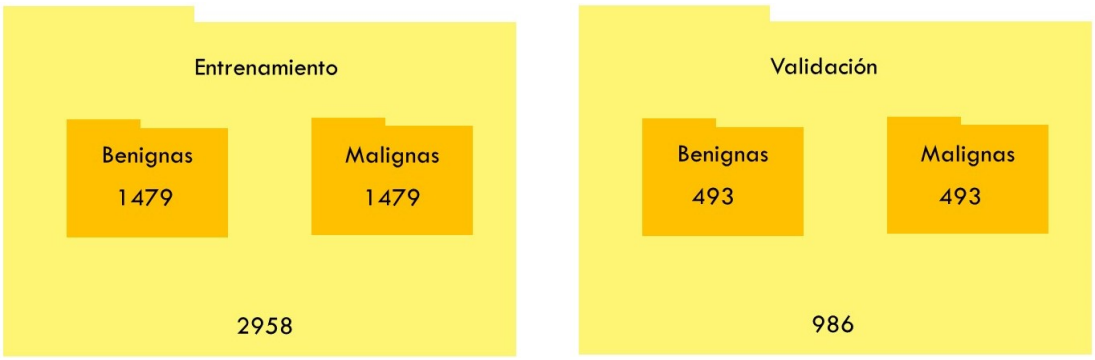
```
array([137, 136, 123, ..., 171, 166, 169], dtype=uint8)
```

Este proceso se repite por cada imagen que entrará a nuestra red neuronal convolucional.

## 5.3. Manejo de datos

Como mencionamos anteriormente, las imágenes utilizadas para los procesos de entrenamiento, validación y prueba de la red neuronal convolucional diseñada para este trabajo de tesis fueron proporcionadas por el Laboratório Visao Robótica e Imagem de la UFPR, y venían acompañadas de un archivo con formato csv que nos indicaba el diagnóstico dado a cada imagen por un patólogo. El conjunto de imágenes incluye un total de 4144 imágenes histopatológicas de tumores de mama, nos enfocaremos en el diagnóstico de cada imagen, puede ser benigno o maligno, por ello dividiremos las imágenes en estas dos categorías para que la red neuronal convolucional logre categorizar las imágenes que introducimos en la prueba del modelo en alguna de estas dos. La separación de nuestro conjunto de imágenes es el siguiente:

- Imágenes de entrenamiento: 2958 imágenes separadas en dos categorias, 1479 benignas y 1479 malignas.
- Imágenes de validación: 986 imágenes separadas en dos categorias, 493 benignas y 493 malignas.




(a) Datos de entrenamiento. (b) Datos de validación.

Figura 5.4: Carpetas de datos de entrenamiento y validación.

En el proceso de prueba utilizaremos un total de 200 imágenes de las cuales obtendremos su diagnóstico dado por el modelo de red neuronal convolucional.

# Capítulo 6

# Algoritmo de red neuronal convolucional

Hemos descrito todos los fundamentos teóricos que se utilizarán en este tesis, principalmente usaremos los conceptos de:

- Función de error cuadrático
- Propagación hacia atrás
- Función de activación rectificadora
- Red neuronal convolucional

Estos fundamentos nos servirán para explicar el algoritmo computacional realizado, también expondremos el diseño de la red neuronal convolucional al igual que presentaremos la base de datos que será utilizada y cómo fue que la dividimos para los distintos procesos de aprendizaje de nuestra red neuronal convolucional aplicada a la categorización de imágenes.

## 6.1. ¿Por qué una red neuronal convolucional?

Existe una amplia gama de problemas a los cuales podemos aplicar los algoritmos del aprendizaje de máquinas, pero muchos de esos problemas son casos muy específicos y especiales en los cuales incluso falta información importante. Al visualizar desde otra perspectiva el problema podemos observar más información del mismo, pero esto aumenta su complejidad y el aplicar los algoritmos de aprendizaje de máquinas más populares no será adecuado, aquí es donde entran la redes neuronales artificiales.

Puede que sean un algoritmo de aprendizaje profundo muy difícil de entender y aplicar, pero permiten una solución precisa a problemas más complejos. Veamos un ejemplo, si deseamos dividir una pequeña cantidad de datos igual a los presentados en la figura 6.1 podemos aplicar un algoritmo de agrupación que es parte del aprendizaje de máquinas. Podemos observar dos tipos de datos, uno perteneciente a la

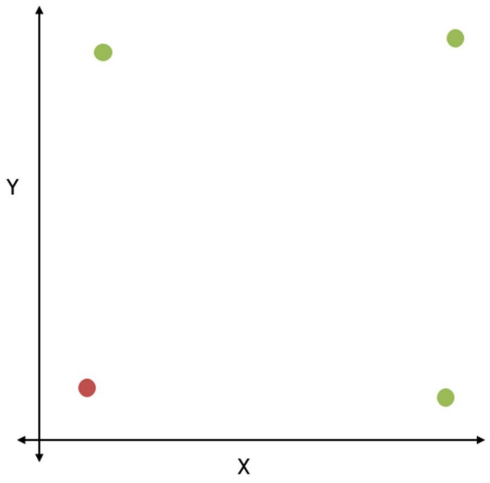



Figura 6.1: Problema de agrupación.

categoría A (el punto de color rojo) y otros pertenecientes a la categoría B (los puntos de color verde). Para separarlos simplemente debemos encontrar características que los relacionen y aplicaremos un algoritmos de agrupación o clustering.

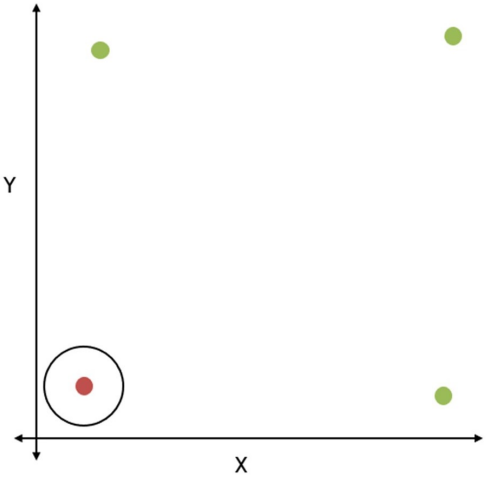



Figura 6.2: Aplicación de un algoritmo de agrupación.

Veamos otro caso, con las mismas categorías, pero ahora tendremos una mayor cantidad de datos y una distribución más realista. Si deseamos implementar un al-

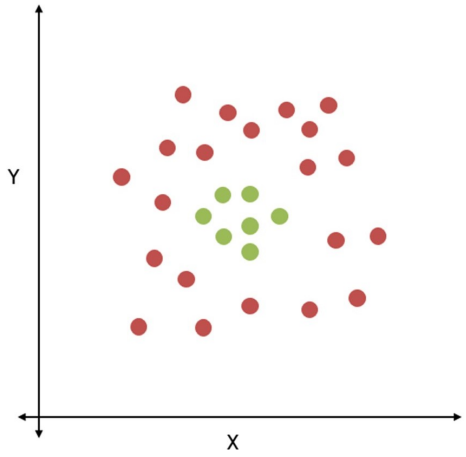



Figura 6.3: Problema con distribución más grande.

goritmo de agrupación, mezclaremos las categorías, esto no funcionará para obtener una solución correcta para este problema. Lo más adecuado sería solo separar los

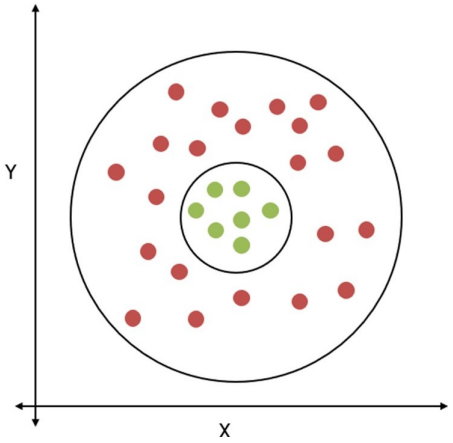



Figura 6.4: Aplicación del algoritmo de agrupación a nueva distrubución.

datos ¿no?, pero verlos desde la perspectiva actual no es de mucha ayuda, en estos momentos estamos viendo los datos distribuidos en el los ejes (x,y). Veámoslo desde el eje (x,z), podemos ver más características de los datos, vemos que tienen alturas distintas, los datos pertenecientes a la categoría A se encuentran en el fondo mientras que los datos de la categoría B los encontramos en puntos más altos en el eje (x,z). Incluso desde esta perspectiva no podemos aplicar un algoritmo de agrupación con más confianza, esto porque debemos considerar que al cambiar de perspectiva los datos siguen teniendo todas sus características en los ejes (x,y,z). Por eso se mencionó anteriormente que al obtener más información de los datos el problema se puede volver más complejo, esto fue lo que pasó. Si ignoramos las características de los datos que se encuentran en el eje y podríamos hacer el problema más sencillo y le aplicamos un algoritmo de aprendizaje de máquinas incluso más práctico, pero no obtendremos un resultado correcto o sería muy impreciso, esto debido a que debemos respetar la información que ahora tenemos de los datos, la

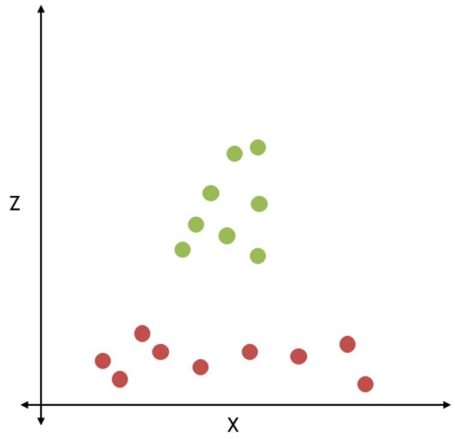



Figura 6.5: Una perspectiva distinta del problema.

que se verá reflejada en su distribución en el eje (x,y,z). Si aplicamos un algoritmo de red neuronal convolucional con una activación relu lograremos una división de los datos de mayor exactitud.

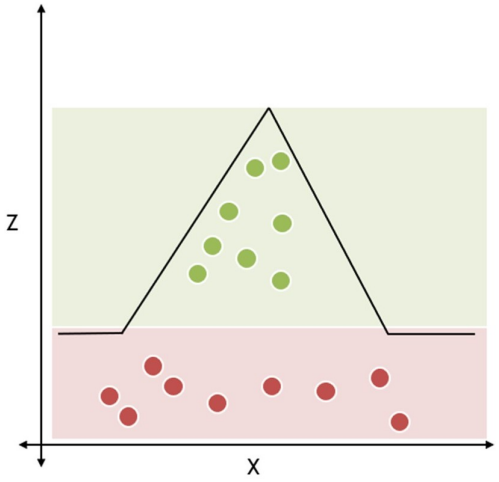



Figura 6.6: Algoritmo de red neuronal convolucional aplicado al problema.

Se hará una explicación más a fondo porque se utilizó la función de activación relu en el siguiente capítulo, en este capítulo se buscaba exponer a grandes rasgos la razón principal de porqué se aplicó una red neuronal convolucional para la clasificación de imágenes.

## 6.2. Diseño de la red neuronal convolucional

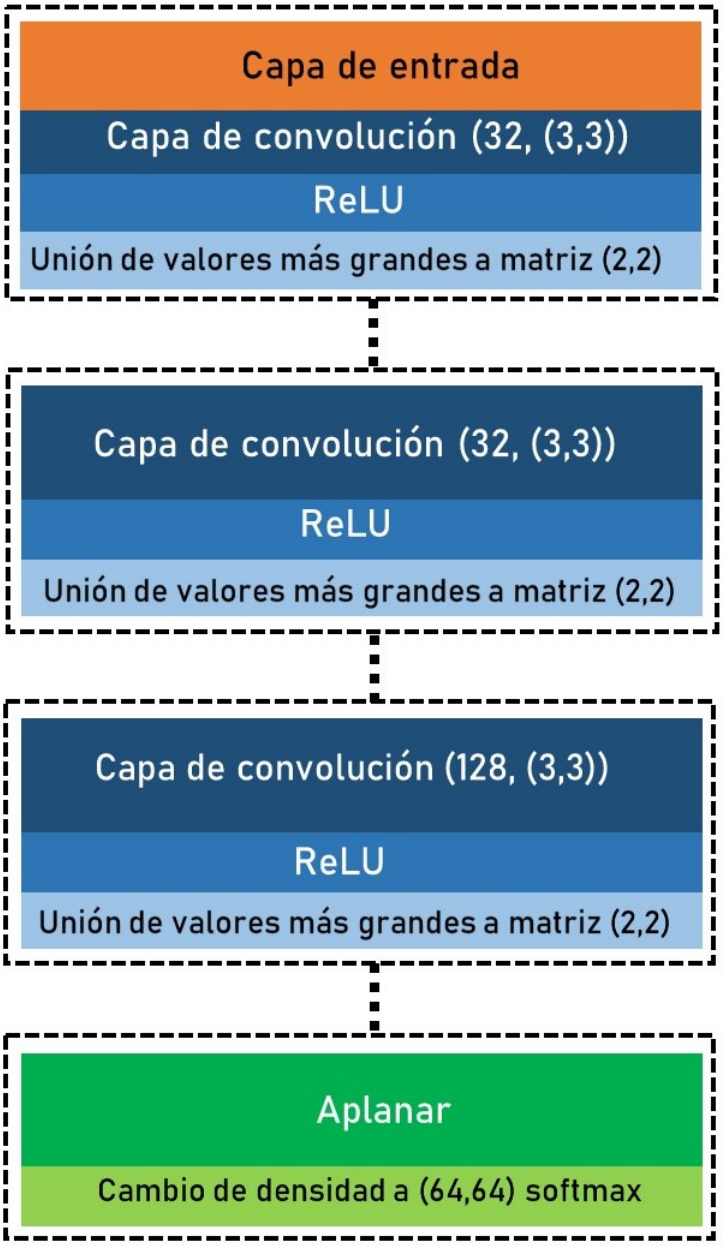



Figura 6.7: Diseño de las capas convolucionales.

Explicaremos cada cuadro del diseño de la red neuronal convolucional, la primera capa convolucional incluye la capa de entrada donde todas la imágenes de las fases de entrenamiento y validación serán procesadas para obtener datos de las mismas. Después pasarán a una capa de convolución donde se aplicarán 32 filtros de convolución o kernels de dimensión (3,3), estos filtros se aplicarán de manera convolutoria a la matriz de datos numéricos de cada imagen, y obtendremos una matriz filtrada de datos, donde veremos cierta información destacada, esto depende del tipo de filtro aplicado, pueden aumentar bordes, aumentar la intensidad de ciertos colores grises, disminuir grises más claros, entre otro.
El siguiente paso es la aplicación de la función de activación, en nuestro caso la

función de rectificación lineal unitaria, esto nos permitirá obtener valores en un rango mayor a cero, porque algunos valores dentro de nuestra matriz son negativos. Para finalizar la primera capa convolucional nuestras matrices deben pasar por un proceso de aplanado, el cual tomará los valores más grandes de una submatriz de dimensiones (2,2) y solo permitirá que los valores más grandes pasen a la siguiente capa convolucional.
Los mismos procesos de las primeras capas de convolución, activación y aplanado se repiten en la segunda capa convolucional. Es hasta la tercer capa convolucional donde aplicaremos 128 filtros de convolución con la misma dimensión que en las capas anteriores, los proceso de activación y aplanado son iguales, solamente cambia la cantidad de filtros aplicados a la matriz de valores numéricos.

Este diseño es muy agradable para obtener precisiones grandes, la mayor hasta el momento ha sido de 92.67 % en pruebas del algoritmo de red neuronal convolucional. A continuación presentaremos los pasos que se utilizaron para implementar este diseño en un algoritmo computacional en el lenguaje Python.

### 6.2.1. ¿Por qué usar la función de activación Relu?

Al aplicar la convolución obtenemos una matriz de características cuyos valores estarán en el rango de 0 a 255, que es el rango de la escala de grises. Se aplicará como función de activación la función relu porque nos permitirá conservar todos los valores positivos obtenidos de la convolución, mientras que a todos los valores negativos los mandará a cero, de esta manera no perderemos información alguna sobre las características filtradas al momento de aplicar la convolución.

## 6.3. Algoritmo computacional de la red

Para implementar el diseño de la red neuronal convolución en un algoritmo computacional utilizaremos el lenguaje de programación Python, para ser más específicos sus módulos de TensorFlow y Keras los cuales son módulos de diferenciación automática, esto para obtener el algoritmo de propagación hacia atrás más rápido y de una manera no tan complicada.

Primero mostraremos los módulos importados para crear el algoritmo computacional de nuestra red neuronal convolucional.

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import os
import logging
logger = tf.get_logger()
logger.setLevel(logging.ERROR)
```

En este caso se utilizaron:

- **Tensorflow y Keras**: Para la creación del modelo en lenguaje Python.
- **Numpy**: Cálculos necesarios en arreglos multidimensionales.

- **ImageDataGenerator**: Convierte las imágenes a datos por las matrices de colores RGB, intensidad de cada una.
- **image**: Nos permite manipular la imagen en tiempo real.

Ahora, debemos importar los archivos de entrenamiento y validación. En total tenemos 4930 imágenes, de las cuales usaremos el 60 % para el entrenamiento (en total son 2958 imágenes), 20 % para la validación y el 20 % sobrante serán utilizadas para la prueba. La base de datos fue dada por el Laboratório Visao Robótica e Imagem de la UFPR que incluye imágenes histopatológicas de cancer de mama. Está dividido en las categorías de los distintos tipos de tumor, para nuestro interés solo nos enfocaremos en si son benignas y malignas.

Para importar nuestras imágenes usamos las siguientes:

```
val_dir = 'C:/Users/octav/OneDrive/Documentos/Tesis red neuronal/
    RN1BC/validation'
train_dir = 'C:/Users/octav/OneDrive/Documentos/Tesis red neuronal/
    RN1BC/training'
test_dir = 'C:/Users/octav/OneDrive/Documentos/Tesis red neuronal/
    RN1BC/test'


train_dir_b = train_dir + "/benign"
train_dir_m = train_dir + "/malign"

val_dir_b = val_dir + "/benign"
val_dir_m = val_dir + "/malign"
```

Y es necesario que nuestras imágenes generen datos, esto será dado mediante la siguiente línea de código:

```
data_generator = ImageDataGenerator(rescale=1./250,zoom_range=0.2)
```

ImageDataGenerator() crea datos numéricos de cada imagen de entrada, esto se explico en el apartado 5.1 del capítulo 5. A continuación mostraremos el código escrito basado en el diseño de la red neuronal convolucional de la figura 6.7.

```
 Model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32,(3,3),activation='relu',input_shape
   =(150,150,3)),
    tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),
tf.keras.layers.Dense(64,activation='softmax')

])
```

Usamos el comando **tf.keras.Sequential([ ])** para permitir una secuencia lineal de las capas convolucionales de tal manera que la información siempre pase a la siguiente ConvLayer de manera directa. Gracias al comando **tf.keras.layers.Conv2d** indicamos el comienzo de una capa de convolución, viene acompañada de una activación ReLU y para finalizar le sigue una capa de unión o pooling con el comando

**tf.keras.layers.MaxPooling**. Es importante agregar **.layers.** para indicar que se está agregando una capa y de qué tipo será esa capa.

Para pasar a la conexión completa necesitamos compactar los datos, para ello usaremos el comando **Flatten()** y tendrá valores de activación mediante la función de activación Softmax.

```
batch= 100
img_shape = 150
Model.compile(optimizer = 'adam',loss='
    sparse_categorical_crossentropy',metrics=['accuracy'])
```

El siguiente paso es preparar los datos y separarlos por categorías, primero debemos indicar el tamaño de los lotes de información, en este caso **batch=100**, la dimensión de la imagen será (150,150).

El optimizador que utilizamos es Adam que servirá para la aplicación de un gradiente del descenso estocástico para actualizar el valores de los pesos y para obtener el valor de pérdida de nuestra información de los datos de salida usaremos **.sparse_categorical_crossentropy.** para calcular la diferencia entre las etiquetas y las predicciones.

```
EPOCHS = 25
val_data_gen = data_generator.flow_from_directory(batch_size =batch
    ,

    directory=val_dir,
                                                              shuffle
     = False,

    target_size = (img_shape,img_shape),

    class_mode = 'binary')

train_data_gen = data_generator.flow_from_directory(batch_size=
    batch,

    directory=train_dir,

    shuffle = True,

    target_size = (img_shape,img_shape),

    class_mode = 'binary')
```

Después dividiremos en categorías los datos obtenidos de nuestra red neuronal convolucional, esto se hace aplicando una class_mode binaria, que usa una regresión logística, y así obtener las dos categorías reconocidas en nuestros datos.

Para finalizar debemos ajustar el modelo a nuestros datos de entrenamiento y de validación, que los analice por épocas y cada época tendrá sus propios pasos de análisis.

```
history = Model.fit_generator(
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train / float(batch))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val / float(batch))))
```

El comando creado para los pasos por época (**int(np.ceil(total_val / float(batch)**) lo usamos para adecuar los pasos a cualquier cantidad de imágenes que sean introducidas en nuestra red neuronal convolucional. Utilizaremos una división de datos totales para el entrenamiento sobre el tamaño de lotes ya definido anteriormente. El comando **np.ceil** mando los valores irracionales a enteros. Un ejemplo a continuación.

```
a = np.array([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0])
np.ceil(a)
array([-1., -1., -0.,  1.,  2.,  2.,  2.])
```

# Capítulo 7

# Resultados

En este apartado mostraremos los resultados obtenidos de nuestro modelo, al correr el código en python obtuvimos parámetros que nos indicaron el costo y la precisión del entrenamiento y la validación, estos valores nos dicen que tan preciso es el modelo de red neuronal convolucional, y cuál es el porcentaje de error final que debemos considerar al momento de calcular la precisión real de nuestro modelo, la cual obtendremos de la fase de prueba de nuestro modelo. Los parámetros obtenidos se presentarán en una gráfica y haremos énfasis en los picos de la precisión de entrenamiento y validación. La prueba del modelo será comparada con los datos del archivo csv que nos indica el diagnóstico de todas las imágenes de prueba. Presentaremos una matriz de confusión, la cual será explicada, y al final calcularemos la precisión real del modelo.

## 7.1. Presición y costo

En las fases de entrenamiento y validación, la red neuronal convolucional aprenderá todos las características únicas de cada categoría, benignas y malignas, de tal manera que permitirá al modelo, obtenido después de las fases anteriormente mencionadas, dar a las imágenes de prueba una predicción o un diagnóstico de la categoría a la que pertenezcan. Primero presentaremos la gráfica de la precisión de ambas fases, entrenamiento y validación, de los datos obtenidos por el ajuste del modelo de red neuronal convolucional.

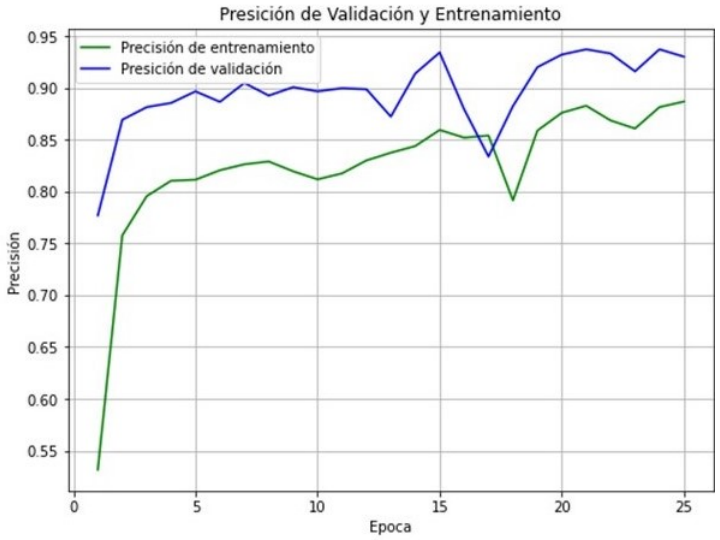



Figura 7.1: Gráficas de precisión.

Observamos que el valor más bajo para ambas fases se encuentra en la primera época, para la fase de entrenamiento el valor más bajo se encuentra en el rango de 0 a 0.55, mientras que en el caso de la fase de validación el valor más bajo pertenece al rango de 0.75 a 0.80. Ahora el valor más alto de cada fase se encuentran en distintas épocas, el entrenamiento tiene su pico más alto en la época 25 y está en el rango de 0.85 a 0.90. La validación tiene dos picos, el primero en la época 15, perteneciente al rango de 0.90 a 0.95, y el siguiente en la época 24, este se encuentra en el mismo rango que el pico anterior. En la siguiente gráfica mostraremos los resultados del ajuste para el costo del modelo. Primero veremos los picos del costo de entrenamiento, el pico más alto se encuentra en la época 1 y lo podemos encontrar en un rango 0.8 a 0.9, mientras que el punto más bajo está en la época 25 entre los valores 0.2 a 0.3, con tendencia a valores cercanos a 0.3. El costo de validación muestra un pico en la primera época, lo podemos encontrar entre los valores 0.8 a 0.9, mientras que el punto más bajo lo encontramos en un rango de 0.1 a 0.2.
Con esto podemos decir que la precisión neta del modelo estará en el rango de 80 % a 90 % mientras que el error neto tendrá un valor del 20 %. Debemos considerar estos valores y comprobarlos con la matriz de confusión.

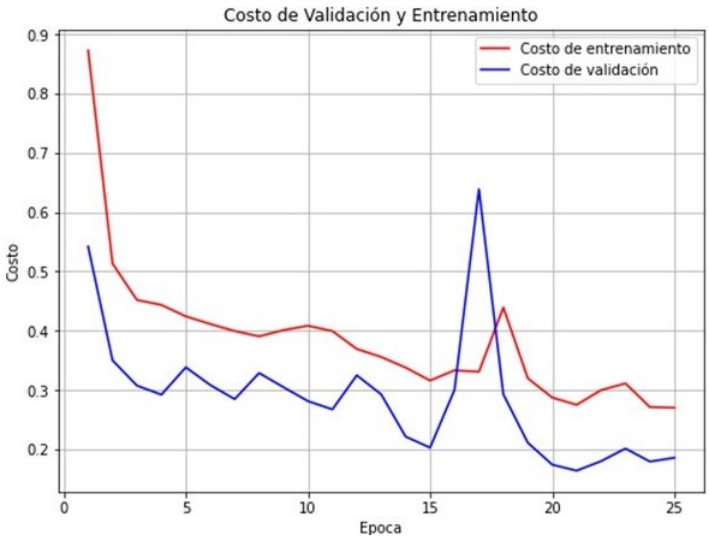



Figura 7.2: Gráficas de costo.

## 7.2. Matriz de confusión

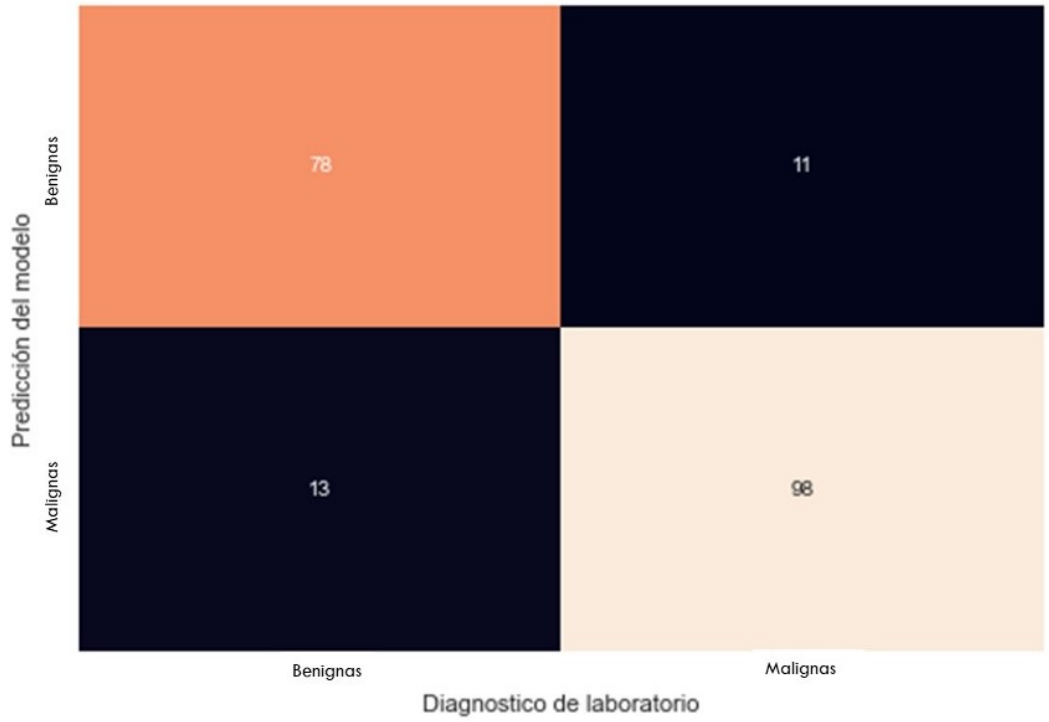



Figura 7.3: Matriz de confusión de los resultados.

Para leer la matriz de confusión debemos enfocarnos en los ejes y los valores que se encuentran en ellos. El eje horizontal representa el diagnóstico hecho por el laboratorio de la UFPR y el vertical representa las predicciones hechas por el modelo. Observemos que cada eje tiene etiquetas de benigno y maligno. Si ambos parámetros, la predicción del modelo y el diagnóstico del laboratorio coinciden entonces se contarán los valores dentro del recuadro naranja, si contamos coincidencias con etiqueta benigna, o dentro del cuadro color durazno, si coinciden en etiqueta ma-

ligna. Entonces tenemos 78 imágenes con coincidencias benignas y 98 imágenes con coincidencias malignas, esto nos da un total de 176 predicciones correctas hechas por el modelo. Los recuadros azul oscuro son los valores cuyas etiquetas no tienen coincidencia alguna, observemos que tenemos 13 imágenes con predicciones malignas que en realidad tienen un diagnóstico benigno, esto lo podemos ver en el cuadro azul oscuro que se encuentra abajo del cuadro naranja. En el cuadro azul oscuro que está arriba del cuadro durazno observamos un total de 11 imagenes cuyas predicciones fueron benignas mientras que sus diagnósticos dicen que pertenecen a la categoría de malignas. La prueba se hizo con un total de 200 imágenes, para obtener la precisión real del modelo debemos dividir el total de coincidencias sobre el total de pruebas.

$$PR = \frac{Totalcoincidencias}{Totalpruebas} = \frac{176}{200} = 88\,\% \tag{7.1}$$

# Capítulo 8

# Conclusiones

En conclusión, los modelos de redes neuronales convolucionales en el lenguaje de Python, con los módulos de TensorFlow y Keras, principalmente aplicados al análisis de imágenes histopatológicas, son exactos, pero su precisión depende mucho de la cantidad y calidad de imágenes que se introduzcan en la capa de entrada.

El modelo diseñado en este trabajo de tesis es de una red neuronal convolucional sencilla. Su diseño base fue para categorizar imágenes de perros y gatos y tenía una precisión similar aunque los rasgos de los perros y gatos son más reconocibles, al igual que la base de datos tenía una mayor cantidad de imágenes. Al diseño original se le agregó una nueva capa convolucional y se agregó el proceso de compactar datos con flatten(). Esto permitió que la precisión real del modelo fuera constante a pesar de que los patrones de similitud fueran más complicados de diferenciar.

El siguiente paso de este trabajo es la obtención de una base de datos más grandes y de mejor calidad, así como la creación de un nuevo diseño y hacer la prueba del módulo Pytorch, en una computadora más potente.

# Bibliografía


[1] Charu C. Aggarwal. *Neural Networks and Deep learning*. Shelter Island, Nueva York: Springer, 2018.

[2] Francois Chollet. *Deep Learning with Python*. Yorktown Heights, Nueva York: Manning Publications, 2018.

[3] Wolfgang Ertel. *Introduction to Artificial Intelligence*. Cham, Suiza: Springer, 2017.

[4] Philip C. Jackson. *Introduction to Artificial Intelligence*. Mineola, Nueva York: Dover Publicactions, 2019.

[5] Prateek Joshi. *Artificial Intelligence with Python*. UK: Packt Publishing Ltd, (2017).

[6] Miroslav Kubat. *An Introduction to Machine Learning*. Coral Gables, Florida: Springer, 2017.

[7] Andreas C. Müller y Sarah Guido. *Introduction to Machine Learning with Python*. Sebasstopol, Rusia: O´Reilly, 2017.

[8] Josh Patterson y Adam Gibson. *Deep Learning*. Sebasstopol, Rusia: O´Reilly, 2017.

[9] Gopinath Rebala, Ajay Ravi y Sanjay Churiwala. *An Introduction to Machine Learning with Python*. Cham, Suiza: Springer, 2019.

[10] Sandro Skansi. *Introduction to Deep Learning*. Zagreb, Croatia: Springer, 2018.

[11] Michael Taylor. *Neural Networks: A visual introduction for Beginners*. Blue Windmill Media, 2017.