UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS QUÍMICAS

IDENTIFICACIÓN DE CONTACTOS RELEVANTES EN ESTADOS FUNCIONALES DE RECEPTORES ACOPLADOS A PROTEÍNAS G MEDIANTE REDES NEURONALES CONVOLUCIONALES

TESIS

PARA OPTAR POR EL GRADO DE

MAESTRO EN CIENCIAS

PRESENTA

QUÍMICO ARSENIO NATAHEL CRUZ CARDOSO

DRA. LAURA DOMÏNGUEZ DUEÑAS
FACULTAD DE QUÍMICA, EDIFICIO F, LABORATORIO 226 SIM-Q

CIUDAD DE MÉXICO, JUNIO, 2023

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS QUÍMICAS

IDENTIFICACIÓN DE CONTACTOS RELEVANTES EN ESTADOS FUNCIONALES DE RECEPTORES ACOPLADOS A PROTEÍNAS G MEDIANTE REDES NEURONALES CONVOLUCIONALES

T E S I S
PARA OPTAR POR EL GRADO DE

MAESTRO EN CIENCIAS

P R E S E N T A

QUÍMICO ARSENIO NATAHEL CRUZ CARDOSO

Ciudad de México, 2023

II

Dedicado a mi familia

y en especial a mis padres.

Agradecimientos

Este trabajo es el resultado de la colaboración y el esfuerzo de muchas personas, y estoy agradecido por su ayuda y apoyo.

Comienzo agradeciendo a mi familia, quienes siempre han estado a mi lado y han creído en mí incondicionalmente. Su amor y apoyo incansable es lo que me ha permitido llegar hasta aquí.

Agradezco también a mis amigos, quienes han hecho que este camino sea mucho más llevadero.

También quiero expresar mi profundo agradecimiento a mis tutores y profesores, en especial a la doctora Laura Domínguez Dueñas, César Iván León Pimentel y Guillermo David Goode, quienes me han guiado y brindado su invaluable sabiduría y conocimiento. Gracias por ser una fuente constante de motivación y por ayudarme a crecer como científico y ser humano. Adicionalmente, agradezco a mis sinodales, ya que sus valiosas sugerencias mejoraron la calidad y claridad del presente documento.

Este trabajo no hubiera sido posible sin el apoyo del Conacyt mediante la beca de apoyo a estudios de posgrado con el CVU:1140891 y el número de registro de becario: 798667.

Finalmente, agradezco a todos aquellos que de alguna manera han contribuido a la realización de esta tesis. Su apoyo y dedicación son inmensamente valorados.

Jurado asignado

Presidente Dr. Alberto Marcial Vela Amieva

Vocal Dr. Enrique García Hernández

Vocal Dr. José Antonio Neme Castillo

Vocal Dr. Jorge Martín del Campo Ramírez

Secretario Dr. José Marco Antonio Franco Pérez

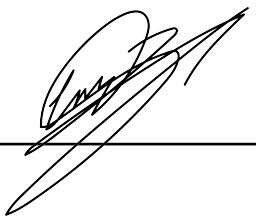
Facultad de Química, edificio F, laboratorio 226 Sim-Q

Dra. Laura Domínguez Dueñas

Asesora

Arsenio Natahel Cruz Cardoso

Sustentante



Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Preámbulo y resumen del proyecto.

Los receptores acoplados a proteínas G, también conocidos como GPCR (*G Protein-Coupled Receptors*), son una familia de proteínas que han resaltado por su relevancia como blancos terapéuticos para fármacos.

El mecanismo general mediante el cuál los GPCR son regulados involucra una señal extracelular que se denomina ligando. La interacción del ligando con el GPCR desencadena sutiles cambios conformacionales en la estructura de la proteína. En caso de que el GPCR se active por la interacción con el ligando, se libera la proteína G acoplada al GPCR hacia el interior de la célula, lo que desencadena una cascada de señales metabólicas. Disponer de un mejor entendimiento de aquellos cambios conformacionales involucrados en la regulación de estas proteínas es de gran relevancia para plantear nuevas moléculas que puedan modular los cambios funcionales de los GPCR y por ende tener un uso potencial como fármacos. Aunque existen diversos estados funcionales para los GPCR, en este trabajo sólo trabajamos con aquellos generados por los

ligandos agonistas y antagonistas, debido a la disponibilidad de datos correspondientes a estos casos y la relevancia de los estados funcionales inducidos por dichos ligandos.

Previamente, durante la licenciatura se desarrolló un algoritmo capaz de clasificar estados funcionales de GPCR usando la información estructural de las proteínas correspondientes, dicho modelo tiene una tasa de acierto del 92,85 %. El objetivo fundamental de este proyecto fue mejorar la capacidad de clasificación en dicho algoritmo previamente construido, basado en redes neuronales, y que denominaremos modelo 0. El modelo 0 puede clasificar el estado funcional de un GPCR entre los dos estados más comunes para los GPCR, aquellos favorecidos por ligandos agonistas o antagonistas, tomando solamente la información estructural del complejo generado entre la proteína y el ligando. Adicionalmente el modelo 0 se diseño usando una sola red neuronal convolucional, ya que la información estructural de las proteínas se codificaba en matrices de contactos.

En este trabajo se buscó determinar las variaciones estructurales más relevantes que generaban un cambio en el estado funcional del GPCR. Debido a esto, analizamos los resultados obtenidos de los cambios estructurales detectados por el algoritmo y se comparó dicha información con aquella reportada previamente en estudios experimentales. Realizar este análisis nos permitió detectar las similitudes entre aquellos contactos relevantes para el modelo y aquellos reportados en la literatura.

Podemos considerar que los modelos desarrollados en este trabajo, basados en el modelo 0, lograron aprender de forma autónoma patrones de contactos en los GPCR y usar dicha información para generar sus predicciones. Con base en los resultados obtenidos se estudiaron aquellas secciones cruciales para las clasificaciones del algoritmo, lo que nos permite sugerir que cambios pueden ser relevantes estudiar de forma experimental para ampliar el entendimiento de los mecanismos

de regulación en los GPCR durante el cambio de estados funcionales.

1.2. Objetivos

- Mejorar el acierto del modelo 0, que distingue entre los estados funcionales de un GPCR inducidos por ligandos agonistas o antagonistas. Dicho algoritmo tiene una tasa de acierto del 92,85 % en su grupo de prueba.
- Determinar si los resultados del nuevo algoritmo construido, basado en el modelo 0, pueden extrapolarse mediante un algoritmo Grad-CAM a los conocimientos experimentales actuales que se tienen acerca de los mecanismos de regulación de un GPCR.

Capítulo 2

Antecedentes teóricos

2.1. Receptores acoplados a proteínas (GPCR)

Los GPCR conforman la superfamilia más grande, diversa y común de proteínas en mamíferos presentes en las membranas celulares.[42] Los miembros de esta superfamilia se encuentran relacionados evolutivamente y debido a esto comparten características estructurales y funcionales, lo que permite estudiar los elementos comunes de estas proteínas. La función principal de los GPCR es ser transductores de señales, en otras palabras, los GPCR ayudan a modular rutas metabólicas mediante indicadores extracelulares

Las señales que pueden recibir los GPCR comprenden un rango amplio de estímulos, desde pequeñas moléculas orgánicas, péptidos, otras proteínas e incluso radiación electromagnética. Típicamente, los GPCR se activan mediante la interacción o unión a un ligando específico, que corresponden a la señal. Adicionalmente el proceso de activación involucra la interacción del GPCR con una proteína G intracelular, una proteína transductora con actividad de GPTasa, que les ayuda a transmitir dicha señal al interior de la célula.

Algunos de los miembros más conocidos de los GPCR son los receptores de dopamina, serotonina y noradrenalina, que son neurotransmisores y por ende están, principal pero no exclusivamente, relacionados a la actividad neuronal.[35] Se pueden encontrar ejemplos de GPCR que cumplen una gran diversidad de funciones fisiológicas en los mamíferos. Algunos de los sistemas en los que los GPCR se encargan de mediar señales son el endocrino, inmunológico, olfatorio, de visión, de gusto, etc.[15,19,44,46]

La importancia de estos receptores en el metabolismo se extiende a mediar alrededor de dos tercios de las hormonas y neurotransmisores en los seres humanos.[25] En consecuencia, no es de sorprender que la relevancia de estas proteínas se refleja en la cantidad de fármacos que actúan regulando de forma directa a algún GPCR. En el 2017, alrededor del 34 % de los fármacos aprobados por la FDA tenían como blanco a un GPCR.[14] Otro factor determinante para el porcentaje de GPCR que son diana de algún fármaco es la farmacobilidad (*druggability* en inglés) de estas proteínas, que permite modular la respuesta de los GPCR de forma relativamente fácil. Se entiende por farmacobilidad como la propiedad de una molécula biológica para ser accesible y poder interactuar con algún ligando con propiedades farmacológicas de interés.[49] Una forma de determinar la farmacobilidad es mediante propiedades fisicoquímicas de la interacción ligando-proteína.[20]

2.1.1. Características estructurales de los GPCR

Al plegarse en su estructura terciaria, los GPCR adquieren una conformación de siete hélices-α que atraviesan de extremo a extremo la membrana celular, como se puede ver en la figura 2.1. Estas hélices transmembranales se denotan como TMx de *TransMembrane helix*, donde la x indica el número de alguna de las siete hélices. Las hélices se encuentran unidas mediante tres bucles intracelulares y otros tres ex-

tracelulares donde el prefijo intra o extra hace referencia a la sección en la que se encuentre el bucle con respecto a la membrana celular. Estos bucles se denotan como ICLx para los intracelulares (de *IntraCellular Loop*) o ECLx para los extracelulares (de *ExtraCellular Loop*); donde la x indica el número del bucle.[15,19,35] Al considerar las asas terminales tenemos un extremo amino, que se encuentra en el exterior de la célula y un extremo carboxilo terminal que se encuentra en el citosol. Una última parte de los GPCR es la hélice yuxtamembranal, conocida como H8 y que es una continuación de TM7 que entra en la membrana celular pero no la cruza por completo.

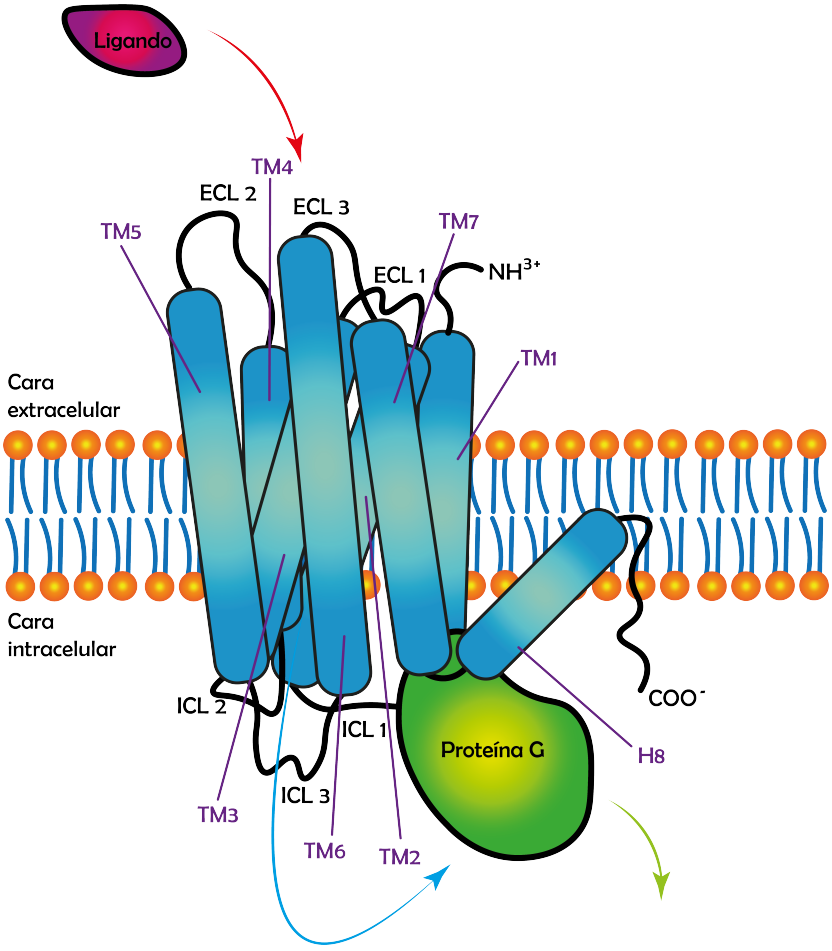


Figura 2.1: Estructura general de un receptor acoplado a proteína G (GPCR).

Dentro de la estructura general de un GPCR, el sitio de unión al ligando se conoce como sitio ortostérico. Se entiende por sitio ortostérico como la sección donde normalmente se une el ligando con el que la proteína suele interactuar. En caso de unirse en un sitio diferente al ortostérico, se denomina al nuevo sitio como alostérico y sus ligandos se conocen como exógenos.[45] De esta forma podemos distinguir tres zonas en la proteína; una zona intracelular, otra extracelular y por último un núcleo transmembranal. Esta descripción general es válida para cualquier GPCR, aunque hay ligeras variaciones conformaciona-

les para cada familia. La conformación de las siete hélices provee a los GPCR de una plasticidad estructural y funcional, que resulta fundamental para las funciones fisiológicas de estas proteínas. En este caso, se entiende por plasticidad a la flexibilidad que presentan los GPCR al variar ligeramente su conformación estructural.

Basados en el análisis filogenético de sus secuencias, los GPCR se agrupan en diferentes clases o familias. Las principales familias de receptores son las de: Rodopsina (clase A), secretina y adhesión (clase B), glutamato (clase C) y *frizzled* (clase F). De estas familias, la clase A es la más común. Debido a esto también es la clase más estudiada y la que cuenta con la mayor cantidad de información estructural en bases de datos como RCSB PDB.[48] A pesar de sus similitudes, los GPCR poseen una gran diversidad entre sus miembros y diversos cambios estructurales en zonas transmembranales, lo que hace del estudio de interacciones, estructura y actividad un reto complejo.

Las principales características estructurales para cada región en los GPCR[21] son:

1. **Región extracelular.** Las principales diferencias estructurales entre los GPCR se encuentran en el dominio extracelular, particularmente en los sitios de unión al ligando y los bucles adyacentes. Se ha encontrado que el bucle extracelular 2 (ECL2) es el más relevante en el cambio de actividad en los GPCR. A su vez ECL2 es más grande que ECL1 y ECL3 y debido a esto puede adquirir diferentes conformaciones.[4]

2. **La región transmembranal y la zona de unión al ligando.** En comparación con los dominios intra y extracelulares, esta es la región más conservada en su ambiente químico entre los GPCR, en especial para aquellos de una misma familia. Lo que se conserva es la función del sitio ortostérico y no necesariamente la secuencia de aminoácidos involucrados. Es de relevancia destacar

el ambiente químico sumamente conservado en la hélice TM7. En la región transmembranal, el ligando suele interactuar con la proteína en lo que se conoce como sitio ortostérico. Está región se encuentra involucrada en los cambios estructurales asociados con la activación de estas proteínas, debido a las ligeras variaciones en la conformación de las hélices de la proteína.

3. **La región intracelular.** Está región se encuentra relativamente conservada, esto se debe principalmente a la pequeña diversidad de proteínas con las que un GPCR se puede enlazar en el interior de la célula y que sirven de mensajeros entre el GPCR y el citosol (Proteínas G, arrestinas, cinasas de receptores acoplados a proteínas G, etc.).

Una de las técnicas para la obtención de estructura de estas proteínas es mediante cristales con alta resolución para difracción de rayos X, siendo la obtención de estos cristales un proceso complejo.[39] Otras técnicas usadas para determinar la estructura de los GPCR son la criomicroscopía electrónica y la resonancia nuclear magnética de estado sólido. [9, 34] La dificultad para obtener información estructural de los GPCR se debe a que estas proteínas suelen requerir de una serie de lípidos, que normalmente se encuentran en la membrana, para estabilizarse y de esa forma obtener cristales estables y que tengan correspondencia con las estructuras *in vivo*. Afortunadamente, con lo años se han desarrollado novedosas técnicas y modificaciones estructurales que han ayudado a obtener cristales con una difracción adecuada y de esa forma obtener la información estructural de la respectiva proteína. Algunas de estas metodologías consisten en la cristalización del GPCR con sus respectivos ligandos, truncar segmentos de la proteína para aumentar la estabilidad del complejo proteína-ligando, mutagénesis, uso de detergentes que simulen la membrana celular, etc.[8,28,48] A pesar de estas técnicas, que han logrado aumentar significativamen-

te la disponibilidad de estructuras de GPCR, la cantidad de archivos estructurales disponibles de GPCR en las bases de datos *online* es relativamente pequeña para las metodologías tradicionales de aprendizaje profundo. Donde usualmente los algoritmos de aprendizaje profundo requieren grupos de datos con miles o decenas de miles de ejemplos para entrenar los modelos. A pesar de dicho inconveniente, en el presente trabajo se implementaron técnicas de extracción de características (*feature engineering*) para usar modelos de aprendizaje profundo cuando se cuenta con pocos datos.

2.1.2. Familia de GPCR tipo A, familia de la Rodopsina

Esta familia es una de las cuatro principales de los GPCR y para febrero del 2023 contaba con 612 estructuras de GPCR interactuando con diversos ligandos en GPCRdb. La familia A se subdivide en 4 subgrupos principales que comparten una gran similitud en sus secuencias ($\alpha - \delta$). Dentro de cada uno de los subgrupos de la familia de la rodopsina, la similitud entre las secuencias es mayor al 25 %. Debido a su similitud, estas proteínas frecuentemente tienen ligandos en común que generan la misma respuesta fisiológica.[19] La familia A de GPCR incluye receptores que interactúan con diversos ligandos como neurotransmisores, hormonas y péptidos.

2.1.3. Esquema de numeración de Ballesteros-Weinstein para los GPCR

Como se menciono previamente, el análisis filogenético de la secuencia de los GPCR lleva a clasificarlos en familias. En dichas familias, las zonas que están mejor conservadas son aquellas que se encuentran en los segmentos transmembranales, principalmente debido a su relevancia para el correcto funcionamiento del GPCR. Debido a esto, Ballesteros y Weinstein diseñaron una notación a la que denominaron números de

residuos genéricos, también conocida como esquema de numeración de Ballesteros-Weinstein.[3] Dicha notación consiste de dos números, donde el primero de ellos se refiere a la hélice y el segundo hace referencia a la posición relativa del residuo. Para el segundo número, el residuo más conservado en cada hélice se indica con el índice 50. El número 50 es bastante conveniente debido a que la mayoría de las hélices alfa en los GPCR tienen menos de cien residuos. Este número disminuye hacia la dirección del amino terminal e incrementa hacia el carboxilo terminal. Dicha notación se acompaña al principio por el aminoácido correspondiente en su notación de una letra.[3,17] Por ejemplo, W6x48 indica la presencia de un triptófano (W) en la hélice TM6, dos residuos abajo del más conservado en dicha hélice. Al utilizar esta notación, podemos establecer un marco común para los GPCR.

De acuerdo con Isberg[17] la **tabla 2.1** condensa los residuos más conservados por hélice para la familia de GPCR A.

Tabla 2.1: Residuo más conservado en los GPCR de la familia A.

Hélice	**Residuo más conservado**	**Numeración de Ballesteros-Weisntein**
TM1	N (Asparagina)	N1x50
TM2	D (Ácido aspártico)	D2x50
TM3	R (Arginina)	R3x50
TM4	W (Triptófano)	W4x50
TM5	P (Prolina)	P5x50
TM6	P (Prolina)	P6x50
TM7	P (Prolina)	P7x50

2.2. Efecto de los ligandos en la geometría de los GPCR

La capacidad de señalización del GPCR desde el sitio extracelular, a través de la membrana celular, hacia el sitio intracelular se debe a la plasticidad inherente de la proteína. Lo que le permite mandar información rápidamente desde el ambiente extracelular hacia los segmentos transmembranales. Poder identificar los cambios estructurales que intervienen en dicho proceso solamente mediante las técnicas experimentales es una tarea desafiante.[6,33,35]

Para los GPCR, existe un nivel de actividad base que se conoce como **actividad constitutiva** o **actividad basal**. La actividad basal, es el nivel de señalización de una proteína mediada por ligandos, pero en ausencia de algún ligando. En el caso de los GPCR, es el nivel de actividad del receptor cuando no interactúa con ningún ligando. La mayoría de los GPCR se encuentran en un equilibrio dinámico entre su estado inactivo y activo, por lo que cuentan con un estado de actividad basal.[19] Este equilibrio se desplaza hacia un lado u otro, entre los estados inactivo y activo, en presencia de los diferentes tipos de ligandos con los que la proteína puede interactuar.[19]

Las conformaciones estructurales activas e inactivas de los GPCR comparten varias características en común, a pesar las variaciones filogenéticas de sus secuencias. Un caso claro de esto es el movimiento relativo entre la hélice 6 y 5, que resulta fundamental en el proceso de unión entre los ligandos y la proteína. Otro ejemplo, son los movimientos en las hélices 7 y 3, que dependen más del tipo de receptor y el ligando con el que la proteína se encuentre interactuando.[19]

Una gran cantidad de conocimiento del proceso de activación en los GPCR se debe a los estudios de los receptores de la clase A.[6,16,19,21,33,43] El trabajo de Hauser y colaboradores,[13] fue fundamental para nuestro estudio, ya que lo usamos como contraparte experimental para comparar nuestros resultados. En dicho trabajo se describe para los GPCR de la familia A el movimiento exterior y la rotación de TM6 en el lado citosólico es una característica universal del proceso de activación en

toda la familia A. Ellos también notaron que el movimiento de la hélice TM6 produce el segundo mayor número de contactos encargados de estabilizar el estado activo o inactivo de los GPCR clase A. Otros contactos relevantes son aquellos de las hélices TM3, TM5 y TM7, donde existen interacciones de prolina, que permiten a dichas hélices acercarse mutuamente para estabilizarse. TM6 y TM3 poseen la mayor cantidad de contactos con otros segmentos de los GPCR, lo que lleva a la hélice TM3 a convertirse en un importante centro de estabilización para los diferentes estados de los GPCR.[13] El análisis estructural de Hauser y colaboradores se basa en el análisis de la frecuencia relativa para ciertos contactos críticos entre diferentes segmentos de estructuras en GPCRs para los estados activo e inactivo. Para determinar los contactos se utilizó una métrica basada en el puntaje RRCS (*Residue-Residue Contact Score*), que desarrollaron Zhou *et al.*[51] para analizar a los GPCR. La métrica RRCS se obtiene mediante un cálculo que cuantifica la preponderancia de un contacto entre diferentes pares de residuos y toma en cuenta todas las interacciones entre átomos para cada residuo en la proteína.[51]

El movimiento de las hélices transmembranales se encuentra acompañado de pequeños cambios en la geometría de los aminoácidos presentes en aquellas hélices que varían su conformación. Estos cambios favorecen la interacción entre residuos específicos, lo que ayuda a determinar el estado activo o inactivo de la proteína. Por ejemplo, en los GPCR tipo A al usar la numeración relativa, la arginina 50 de la hélice 3 (R3x50), que tiene un porcentaje de conservación del 96 %, forma un puente iónico con un residuo vecino aspartato o glutamato (D/E3x49), que tienen un porcentaje de conservación del 68 % y 20 %, respectivamente. Dicho enlace es característico de los estados inactivos de GPCR tipo A.[19]

2.2.1. Estados funcionales de los GPCR

Los estados funcionales de los GPCR se ven afectados por su ambiente en la membrana celular, así como por las diferentes interacciones ligando-proteína. Estas interacciones resultan en conformaciones específicas para los GPCR, que pueden activar o desactivar cascadas de señales metabólicas.[15,16] Experimentalmente se pueden estudiar dichas interacciones mediante técnicas como la cristalografía de rayos X y la criomicroscopía electrónica.[4,11,33,43] El proceso de activación de un GPCR, inducido por las interacciones ligando-proteína, se debe a cambios sutiles, de alrededor de un nanómetro.[13,15,16,19,33,35] A pesar del conocimiento actual, todavía quedan algunas dudas para entender por completo detalles conformacionales claves en el proceso de modulación de los GPCR mediados por ligandos, así como algunas de las interacciones ligando-proteína y proteína-proteína que afectan los estados funcionales de los GPCR.[13,23,43]

2.2.2. Ligandos agonistas y antagonistas de los GPCR

Los ligandos agonistas se unen al receptor para promover la respuesta celular más allá del nivel basal de actividad, mientras que los ligandos antagonistas se oponen a la acción de un ligando agonista y previenen una respuesta celular, llegando incluso a disminuir la actividad basal del receptor.[15,33] Una descripción más amplia de las diferentes actividades que los ligandos de GPCR pueden presentar se encuentra en la sección 7.1. Estas respuestas se pueden visualizar con curvas de dosis respuesta. Un ejemplo de dichas curvas se muestra en la figura 2.2.

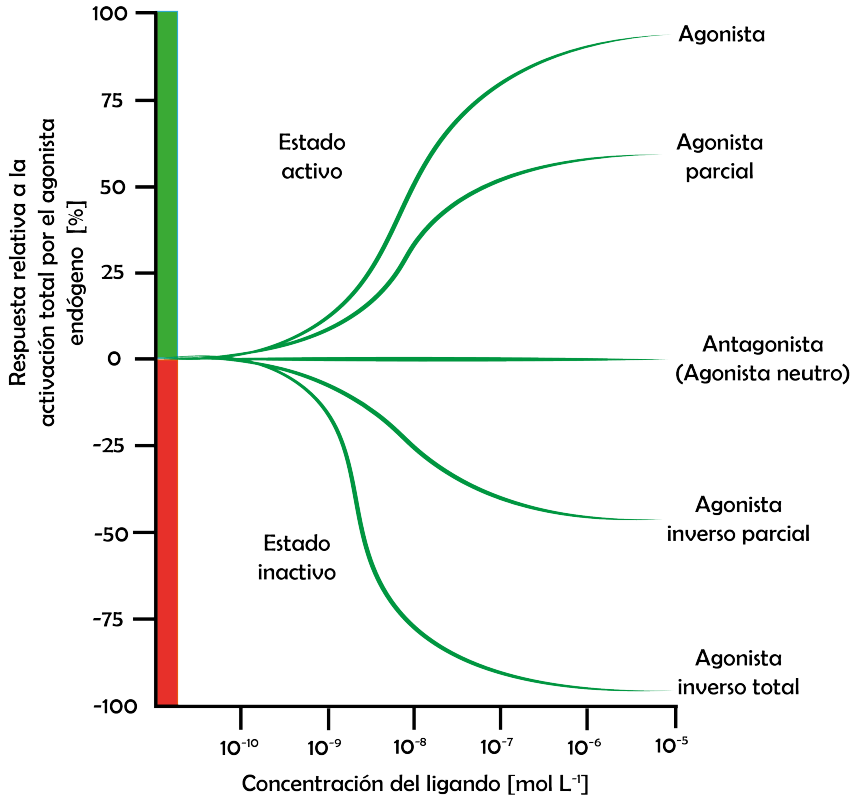


Figura 2.2: Curvas de respuesta ante diferentes actividades de ligandos

2.3. Redes neuronales

Una red neuronal artificial es un sistema computacional inspirado en sus equivalentes biológicos, las células neuronales, y está compuesta por una gran cantidad de nodos o "neuronas"que se encuentran interconectadas.[1,12] Las redes neuronales artificiales se utilizan para resolver una gran diversidad de problemas que involucran el procesamiento de información, tales como el reconocimiento de patrones, la toma de decisiones y el aprendizaje automático. Se pueden clasificar en varios tipos,

como las redes neuronales *feed-forward*, redes neuronales recurrentes, redes neuronales convolucionales, etc. Las neuronas artificiales se activan o se desactivan en función de los valores de entrada que reciben, y transmiten información a través de las conexiones o "sinapsis"que las unen. En otras palabras, una red neuronal computacional es un sistema computacional que busca simular, mediante una simplificación, la forma en que el cerebro humano procesa la información, permitiendo aprender y adaptarse a patrones complejos en los datos usados para entrenar estos sistemas.

Varios algoritmos de redes neuronales se han aplicado para resolver diversos problemas quimioinformáticos y relacionados a la química, como lo pueden ser el desarrollo de fármacos, diseño de materiales, predicción de reacciones o rutas sintéticas, etc.[24,27,41] Actualmente, las soluciones que dependen de la inteligencia artificial, y más recientemente del aprendizaje profundo, se han convertido en parte de algunas metodologías novedosas[1,18,37,52].

2.3.1. La neurona

En una red neuronal artificial, una neurona, como se muestra en la figura 2.3, es un nodo computacional que recibe entradas, las procesa y emite una salida. Cada neurona está conectada a otras neuronas mediante conexiones, llamadas sinapsis, que tienen un peso asociado. La entrada (x) que recibe una neurona es una combinación lineal de las salidas de las neuronas anteriores, y esta combinación se multiplica por los pesos correspondientes (w). Después de dicha multiplicación se suman los valores obtenidos y se les agrega una constante llamada sesgo (b). Luego, la neurona aplica una función de activación (z) para determinar si la neurona se activa o no. Una función de activación es una función matemática que toma como entrada la combinación lineal y produce una salida (y). Una neurona se activa si el resultado

de la función de activación es mayor a cierto umbral y no se activa en caso contrario. La salida de una neurona se utiliza como entrada para las neuronas a las que está conectada. La única excepción es cuando la neurona se encuentra en la última capa de la red neuronal, en cuyo caso la salida de dicha neurona se usa como salida de la red neuronal.[1,12] Una red neuronal es el resultado de conectar varias capas de neuronas, donde una capa es un grupo de neuronas que reciben las mismas entradas y pasan sus resultados a otro grupo de neuronas. De esta forma las redes neuronales pueden aprender patrones complejos subyacentes en los datos.[1]

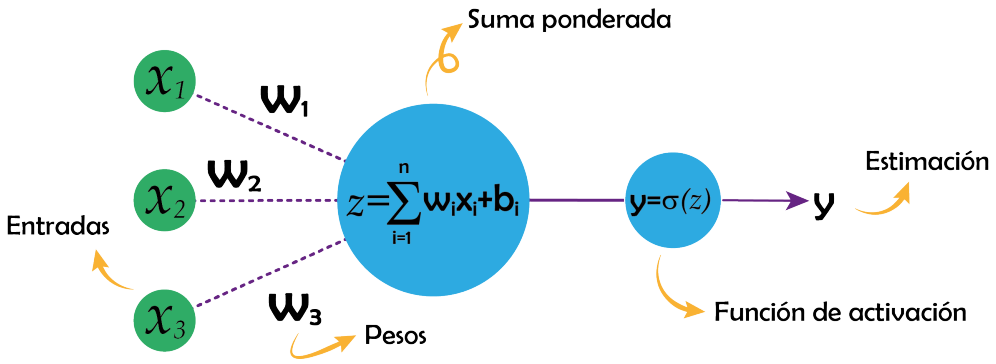


Figura 2.3: Esquema general de una neurona artificial. En esta imagen se muestran las entradas x, que se multiplican por sus correspondientes pesos w y se suman con un sesgo b para obtener el respectivo valor z. Este valor se pasará a una función de activación $\sigma()$ para obtener el valor final de la neurona y.

Una neurona en una red neuronal artificial se puede modelar mediante dos funciones. Primero tenemos una función lineal que consiste en una suma ponderada de todos los n valores de entrada x y sus respectivos n pesos asociados w. Después de dicha suma, se adiciona el sesgo b **ecuación 2.1**. La salida de esta función anterior se pasa a la segunda función matemática, que consiste en una transformación no lineal aplicada a la función lineal z producto de la suma ponderada **ecuación 2.2**. Esta transformación no lineal se conoce como como

función de activación (**sección 2.3.3**).

$$z = \sum_{i=1}^{n} w_i x_i + b_j = W^T X + B \tag{2.1}$$

$$y = \sigma(z) \tag{2.2}$$

Donde:
w_i = Peso i
x_i = Dato de entrada i
b_j = Sesgo j
W^T = Vector transpuesto de pesos
B = Vector de sesgos

Los valores en x corresponden con una representación numérica de los datos de entrada de la red neuronal y se pueden agrupar como un vector de características con una dimensionalidad l ($x \in \mathbb{R}^l$), donde l corresponde al tamaño del vector de datos de entrada de la neurona. Por otro lado, los valores de y se pueden agrupar en un vector diferente de dimensionalidad m ($y \in \mathbb{R}^m$), donde m corresponde al número de clasificaciones posibles para los datos usados. Computacionalmente conviene representar a la **ecuación 2.1** como el producto de matrices en lugar de operaciones entre vectores, ya que de esta forma se puede acelerar las operaciones de la red neuronal. Un ejemplo general de cómo se procesaría información en una red neuronal se puede encontrar en la sección 7.2.

2.3.2. Parámetros e hiperparámetros en una red neuronal

Las redes neuronales contienen una serie de valores conocidos como parámetros, que corresponden con los valores de los pesos en las

funciones matemáticas en la red neuronal que permiten generar clasificaciones adecuadas de los datos, como lo son w y b en la función lineal. Dichos valores se aprenden de forma autónoma a partir de los datos usados para entrenar al modelo y durante el entrenamiento de la red. Durante este proceso de entrenamiento, el algoritmo de aprendizaje automático ajusta estos parámetros, mediante funciones de optimización, para minimizar el error entre las predicciones del modelo y los datos de entrenamiento. Una vez entrenado, el modelo utiliza estos parámetros para hacer predicciones sobre nuevos datos.

Por otro lado, los hiperparámetros son valores numéricos de un modelo de aprendizaje automático que no son aprendidos directamente a partir de los datos, sino que se establecen previamente antes de comenzar el entrenamiento del modelo. Estos hiperparámetros son utilizados para controlar el comportamiento del modelo y su rendimiento durante el tiempo de ejecución del entrenamiento del modelo. Los hiperparámetros se modifican manualmente con la intención de mejorar el desempeño del modelo entrenado.

Algunos ejemplos de hiperparámetros en un modelo de aprendizaje profundo son

- Función de activación. (**sección 2.3.3**)
- Valores asociados a los algoritmos de optimización (α, β, γ). (**sección 2.3.5**)
- Algoritmo de optimización. (**sección 2.3.5**)
- Arquitectura de la red neuronal. (**sección 2.3.6**)
- Número de unidades ocultas en el modelo. (**sección 2.3.6**)
- Número de capas. (**sección 2.3.6**)
- Épocas de entrenamiento. (**sección 2.3.7**)

El proceso de seleccionar los hiperparámetros adecuados se conoce como ajuste de hiperparámetros. Es una tarea importante ya que un conjunto inadecuado de hiperparámetros puede causar un mal rendimiento del modelo. En este trabajo se usó el muestreo aleatorio de una distribución uniforme para determinar los valores iniciales de los hiperparámetros usados, aunque posteriormente se modificaron para optimizar al modelo usando conocimiento empírico.

2.3.3. Funciones de activación

Una función de activación, **ecuación 2.2**, es una función matemática que se utiliza en las redes neuronales artificiales para determinar si una neurona pasa su información a la siguiente neurona o no, aunque también puede determinar la relevancia de una característica para una clasificación. Esta función se aplica previa a la salida de la neurona, al resultado de la combinación lineal de las entradas y los pesos, antes de ser utilizada como entrada para las siguientes capas de la red neuronal o como salida final del modelo.

La función de activación introduce no linealidad en la red neuronal, lo que permite al modelo aprender a reconocer diversos patrones en los datos. Sin ella, una red neuronal, independientemente de la cantidad de neuronas concatenadas, sería equivalente a una sola combinación lineal de transformaciones, cumpliendo con la **ecuación 2.1**, y no sería capaz de aprender patrones complejos. En otras palabras, sin la función de activación, una red neuronal tendría el mismo efecto que una sola neurona, con lo cual se perderían las ventajas computacionales característica de estos modelos.[31]

El principal requerimiento de las funciones de activación, aparte de ser transformaciones no lineales, es que deben polarizar a los datos procesados, por lo que al pasar por la función de activación estos valores se van a aproximar hacia diferentes valores, dependiendo de la clase a la

que pertenezcan. Mediante este mecanismo podemos prender o apagar neuronas. Una neurona se apaga cuando la matriz X de entrada tiene valores muy cercanos a un valor definido por la función de activación, por lo general cero. Por lo qué al operar la suma ponderada por la función de activación, la neurona va a regresar valores cercanos a cero. El proceso contrario corresponde con una neurona prendida, donde los valores de salida de la neurona tendrán valores distintos a cero o el valor definido por la función de activación. Por ejemplo, en la tangente hiperbólica ese valor corresponde con -1 para la neurona apagada.

Hay varias funciones de activación comunes que se utilizan en las redes neuronales, como se muestran en la figura 2.4. Algunos ejemplos de estas son la función sigmoide, la función ReLU y la función tangente hiperbólica. Cada una de estas funciones tiene sus propiedades y se utilizan en diferentes situaciones. Por ejemplo, la función sigmoide se utiliza a menudo en las capas de salida de una red neuronal para generar probabilidades, mientras que la función $ReLU$, de las siglas en inglés para *Rectified Linear Unit*, se suele utilizar en las capas ocultas para evitar el problema del gradiente desvaneciente. Este es un problema que puede ocurrir en las redes neuronales profundas, donde debido a la profundidad del modelo se dificulta el proceso de aprendizaje ya que los valores de actualización de los parámetros se hacen casi nulos.

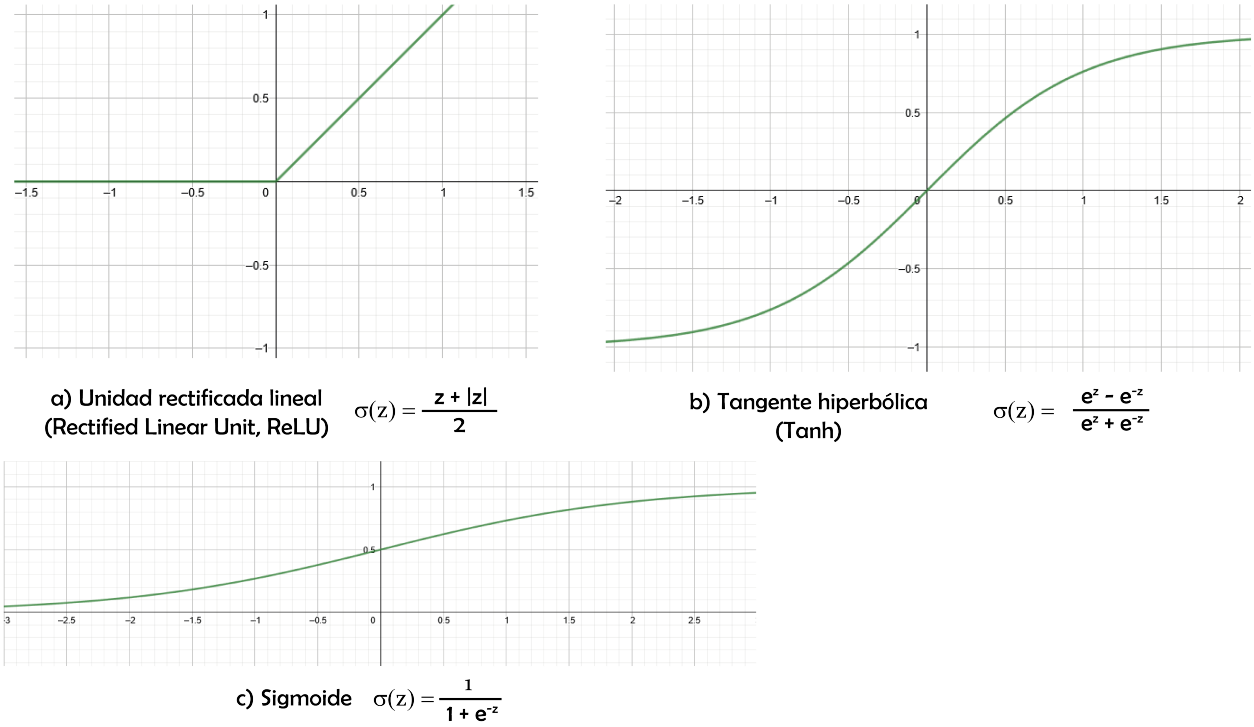


Figura 2.4: Algunos ejemplos de funciones de activación: (a) tanh, (b) ReLU y (c) Sigmoidea

Aunque la función ReLU puede aparentar ser una función lineal, se trata de una función no lineal como se demuestra en la sección 7.3.

2.3.4. Función de pérdida y de costo

Una vez que se han descrito los elementos básicos que conforman una neurona es posible asignar las salidas de algunas neuronas como las entradas de otras neuronas, lo que da paso a una red neuronal. El siguiente paso consiste en realizar el entrenamiento de dicha red. El proceso de entrenamiento consiste en encontrar los valores óptimos de los parámetros w y b de cada neurona. Dichos valores óptimos permiten que el modelo pueda realizar la clasificación de los valores de entrada con la mayor tasa de acierto posible.

Las funciones de pérdida y de costo son herramientas utilizadas en el entrenamiento de un modelo de aprendizaje automático, que miden el error entre las predicciones del modelo y los datos de entrenamiento. Estas funciones se usan para guiar el proceso de ajuste de los parámetros del modelo. En un principio los valores de los parámetros se asignan aleatoriamente, y mediante la función de pérdida, podemos estimar una medida de error para el modelo. Esta función de pérdida nos permite obtener la bondad de ajuste del modelo generado por los parámetros obtenidos para cada paso del entrenamiento de la red neuronal y modificar los parámetros del modelo para mejorar ese ajuste en el siguiente paso de entrenamiento del modelo.

Dado que la etiqueta de los datos usados para entrenar la red neuronal se conocía previamente, el modelo de entrenamiento es supervisado. Mediante la función de pérdida,[24,26] podemos medir qué tan diferentes son las distribuciones del conjunto de datos de predicciones $\hat{y}$ para la red neuronal con respecto a la distribución del conjunto de datos de los valores verdaderos y, que corresponde con las etiquetas de los datos. Una función de pérdida comúnmente usada en el entrenamiento de redes neuronales, y que se usó en este trabajo, es la función de entropía cruzada, que se deine en la **ecuación 2.3**

$$L(y, \hat{y}) = y \log\left(\frac{y}{\hat{y}}\right) + (1 - y) \log\left(\frac{1 - y}{1 - \hat{y}}\right). \tag{2.3}$$

Con base en esta ecuación, es posible formular la función de costo que se muestra en la **ecuación 2.4**. La función de costo es el promedio de las funciones de pérdida del sistema y con la cual podemos evaluar el desempeño global de la red neuronal, donde m es el número total de muestras usadas para entrenar al modelo.

$$J(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, \hat{y}_i). \tag{2.4}$$

El modelo anterior para calcular la pérdida también se conoce como pérdida por entropía cruzada binaria. Esto se debe a que se aplica el concepto de entropía proveniente de la teoría de la información, también conocida como entropía de Shannon.[10] (Apéndice 7.4) El modelo de entropía binaria cruzada es el estándar en casos donde se considera una clasificación binaria. Es decir, solo se tienen dos resultados posibles.

El valor de la función de costo $J(y_i, \hat{y}_i)$ se aproximará a cero conforme la red neuronal aumente la exactitud de sus predicciones, dado que $\hat{y}$ y y tendrán el mismo valor.

2.3.5. Algoritmos de optimización

Los algoritmos de optimización son técnicas utilizadas para ajustar los parámetros de un modelo de aprendizaje automático de manera que se minimice el error entre las predicciones del modelo y los datos de entrenamiento.[26] Estos algoritmos buscan encontrar los valores óptimos de los parámetros del modelo para que el error sea lo más pequeño posible.

Existen varios algoritmos de optimización que se utilizan en el aprendizaje automático, incluyendo:

- Gradiente descendente, **ecuaciones 2.5**: Es el algoritmo más utilizado en el aprendizaje automático. Este algoritmo utiliza el gradiente de la función de pérdida para ajustar los parámetros del modelo, w y b. El gradiente se calcula para cada parámetro y se utiliza para actualizar el valor de ese parámetro en la dirección que reduce la función de costo $J(w, b)$. Esto se logra al usar un algoritmo recursivo, lo que significa que se repetirá un determinado número de veces hasta cumplir alguna de las siguientes condiciones: La función de costo $J(w, b)$ llega a un valor umbral muy cercano a cero, o bien si cumple un cierto número de pasos.

- Gradiente descendente estocástico: Es una variante del gradiente descendente que actualiza los parámetros del modelo utilizando solo un subconjunto de los datos de entrenamiento en lugar de todos los datos de entrenamiento.

- Algoritmo de gradiente con memoria a corto plazo (*RMSprop*): Se basa en el algoritmo de gradiente descendente estocástico, pero utiliza una técnica llamada "memoria a corto plazo"para ajustar el tamaño del paso de actualización de pesos en cada iteración. Esto ayuda a evitar que el algoritmo se quede atascado en mínimos locales y mejora la velocidad de convergencia. RMSprop es especialmente útil para entrenar redes neuronales profundas, ya que ayuda a mantener un buen equilibrio entre la velocidad de entrenamiento y la precisión de los resultados.

- Algoritmo de optimización Adam: Es un algoritmo de optimización que combina las ventajas de los algoritmos de gradiente descendente estocástico y de memoria a corto plazo. La combinación de estas técnicas hace que Adam sea una excelente opción para entrenar redes neuronales, ya que tiene una buena capacidad de evitar los mínimos locales y tiene una buena capacidad de convergencia.[27] En general se considera un algoritmo robusto y es ampliamente utilizado en la práctica.

Las ecuaciones asociadas al algoritmo de gradiente descendentes, ecuaciones 2.5, consisten en un proceso de actualización, donde se toma el parámetro a actualizar y se le resta un término que se encuentra asociado con el gradiente de la función de costo con respecto al parámetro correspondiente, lo cual permite modificar a dicho parámetro.

$$w_i := w_{i-1} - \alpha \frac{\partial J(w,b)}{\partial w}$$
$$b_j := b_{j-1} - \alpha \frac{\partial J(w,b)}{\partial b}, \tag{2.5}$$

Aquí el parámetro α se conoce como la tasa de aprendizaje y nos indica la magnitud de los cambios en cada paso del algoritmo. Dicho en otras palabras 'qué tan grande es el paso'. Este parámetro se debe seleccionar con cuidado ya que de ser muy pequeño hará que sea más costoso el cálculo de este algoritmo, aparte de disminuir la rapidez de convergencia y en caso de que α sea muy grande podemos dificultar la convergencia a valores mínimos de la función de costo. Otro problema que puede surgir al buscar un mínimo en la función de costo es encontrar un mínimo local en vez de alcanzar al mínimo global, por lo que la solución del algoritmo no será la óptima. Esto se debe a la complejidad del espacio de búsqueda, la inicialización de los pesos en la red neuronal o la elección de un algoritmo de búsqueda inadecuado. Para evitar estos problemas se suelen aplicar técnicas adaptativas, donde durante el entrenamiento se busca variar al parámetro de la tasa de aprendizaje en función del gradiente u otros hiperparámetros, para modificar el valor de la tasa de aprendizaje.

2.3.6. Topología de una red neuronal

Una vez que tenemos todos los elementos para construir la red neuronal y optimizarla para mejorar su capacidad de predicción, podemos cambiar su topología. La topología de una red neuronal artificial se refiere a la estructura y disposición de las capas y las conexiones entre las neuronas. En otras palabras, es la arquitectura de la red. Existen diversas arquitecturas, cada una diseñada para resolver un problema diferente.

La elección de la arquitectura varía en función del tipo de problema que se quiere solucionar. Por ejemplo el modelo *AlexNet CNN*[44], que es una red neuronal convolucional, ha mostrado ser muy útil para clasificar imágenes; también se pueden usar arquitecturas para resolver problemas que involucran secuencias de datos, como son el caso de las GRU (*Gated Recurrent Unit*), LSTM (*Long Short Term Memory*) o más recientemente los *transformers*, que están basados en algoritmos conocidos como mecanismos de atención. En estos casos, aparte de los cambios entre las conexiones de las neuronas, se considera el uso de neuronas modificadas que son funciones matemáticas diferentes aplicadas a procesar de forma más eficiente los datos con los que se quiere entrenar al modelo.

2.3.7. Entrenamiento de una red neuronal y tipos de entrenamiento

El entrenamiento de una red neuronal se refiere al proceso de ajustar los pesos w_i y sesgos b_j de las conexiones entre las neuronas en una red neuronal con el objetivo de que la red pueda realizar una tarea específica de manera precisa. Dependiendo del tipo de datos y arquitectura usada existen diversos tipos de entrenamiento, que se describen en mayor detalle en la sección 7.6. Durante el entrenamiento, se presenta un conjunto de datos a la red y se utilizan algoritmos de optimización para ajustar los pesos y sesgos de manera que los errores entre las salidas deseadas y las salidas actuales de la red sean minimizados.

El proceso de entrenamiento se divide en dos fases:

- La primera fase es la propagación hacia adelante (*feedforward*), donde se proporciona un conjunto de entradas a la red y se calculan las salidas correspondientes a través de las capas de la red. Después, se evalúan los errores de cada predicción mediante la función de costo respectiva (**sección 2.3.4**).

- La segunda fase es la propagación hacia atrás (*backpropagation*), donde se utiliza un algoritmo para calcular el gradiente del error con respecto a los pesos y sesgos de la red, para actualizarlos en la dirección que reduce el valor del error. La actualización de los parámetros se realiza mediante el algoritmo de optimización seleccionado para la red neuronal (**sección 2.3.5**).

Este proceso, conocido como época de entrenamiento, se repite varias veces hasta que el error sea más bajo que un valor predefinido o se hayan alcanzado el número máximo de épocas previamente seleccionadas para entrenar al algoritmo.

El proceso de entrenamiento suele implicar validación cruzada, donde se divide el conjunto de datos en subconjuntos y se utilizan diferentes subconjuntos para validar el rendimiento de la red neuronal en cada iteración del entrenamiento. Mediante esta metodología se pueden construir modelos robustos que puedan realizar de forma adecuada su tarea.

Existen varios métodos de validación cruzada, pero los más comunes son:

- Validación cruzada k-*fold*: se divide el conjunto de datos en k subconjuntos, llamados *fold*. Después, se entrena al modelo en k-1 de los *fold* y se utiliza el *fold* restante para evaluar el rendimiento final del modelo. Este proceso se repite k veces, utilizando un *fold* diferente, en cada caso, como conjunto de prueba en cada iteración. El rendimiento del modelo se promedia a través de las k iteraciones.

- Validación cruzada dejando uno afuera: se divide el conjunto de datos en dos subconjuntos, uno para entrenamiento y otro para prueba. El proceso se repite n veces, donde n es el número de ejemplos en el conjunto de datos, cada vez utilizando un ejemplo diferente como conjunto de prueba.

La validación cruzada es útil para evaluar el rendimiento del modelo en un conjunto de datos desconocido, ya que permite obtener una estimación más precisa del rendimiento real del modelo. También es útil para detectar si el modelo está sobreajustado, ya que un modelo sobreajustado tendrá un rendimiento bajo en los conjuntos de prueba.

El sobreajuste, también conocido por su nombre en inglés *overfitting*, en una red neuronal ocurre cuando un modelo ha aprendido de manera muy específica los patrones presentes en los datos de entrenamiento, pero no tiene la capacidad de generalizar esos patrones a nuevos conjuntos de datos. Es decir, el modelo se ajusta demasiado bien a los datos de entrenamiento, pero tiene un rendimiento bajo en los datos de prueba o de validación.

En general, la validación cruzada, es una técnica importante para evitar problemas de sobreajuste y poder establecer una medida de la capacidad de generalización de un modelo.

Una vez que se completa el proceso de entrenamiento, la red estará preparada para realizar la tarea para la que se ha entrenado en conjuntos de datos nuevos, como la clasificación de imágenes o la generación de texto. Sin embargo, es importante mencionar que el entrenamiento de una red neuronal no es un proceso estático, y en ocasiones es necesario volver a entrenar o actualizar una red neuronal ya entrenada para adaptarse a nuevos datos o mejorar su rendimiento.

2.4. Redes Neuronales Convolucionales

Una red neuronal convolucional (CNN, de sus siglas en inglés *Convolutional Neural Networks*) es un tipo de arquitectura de red neuronal artificial o aprendizaje profundo especialmente diseñada para trabajar

con datos matriciales. Estas redes son capaces de aprender características espaciales en los datos a través de la aplicación de filtros y máscaras sobre las imágenes de entrada.[26] Esto permite a las CNN detectar patrones complejos y características en las imágenes, como rostros, objetos y escenas. Un ejemplo general de la arquitectura de una CNN se muestra en la figura 2.5.

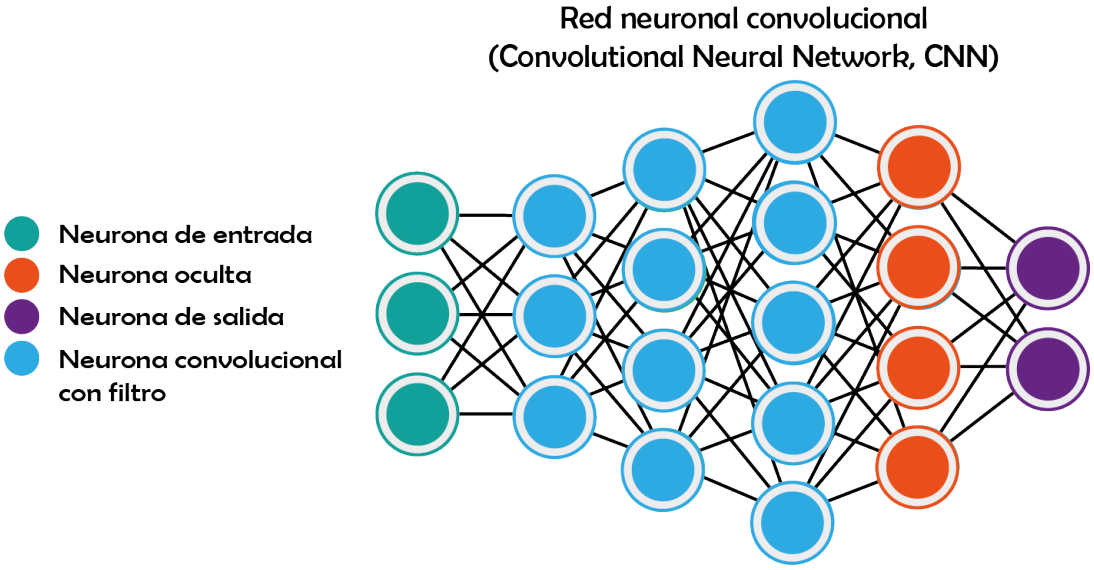


Figura 2.5: Estructura general de una red neuronal convolucional. Al inicio están las neuronas de entrada, seguidas por las neuronas convolucionales que incluyen la convolución y su filtro. Por último, tenemos las neuronas ocultas, que son neuronas normales seguidas por las neuronas de salida.

Los modelos de CNN entran en lo que se considera aprendizaje profundo. El punto decisivo en las arquitecturas de aprendizaje profundo es que estas constan de un gran número de capas de neuronas ocultas que permiten al modelo aprender representaciones de datos a niveles cada vez más abstractos. Aunque el término aprendizaje profundo, en inglés *deep learning*, es ampliamente usado no hay una definición clara de en qué punto una red neuronal artificial se hace profunda. Jürgen Schmidhuber realizó un estudio en el cual propone una medida deno-

minada CAP, siglas de *Credit Assignment Paths*, que es una medida de las posibles operaciones desde la capa de entrada hasta la capa de salida en una red neuronal, y con la cual se puede determinar si un modelo es profundo. En su artículo se proponen algunas arquitecturas con un CAP>4 que son ampliamente reconocidas como redes neuronales profundas.[38]

El núcleo de estas redes es un proceso llamado convolución, que consiste en aplicar el producto interno de Frobenius (Sección 7.5) entre una matriz llamada filtro o núcleo a otra matriz que representa la imagen a procesar, moviendo este filtro con su respectiva operación a través de la imagen. Esto se hace para extraer las características más relevantes de una imagen como bordes, formas y texturas. En este proceso, se puede disminuir el tamaño de la matriz, lo cual resulta bastante útil computacionalmente cuando se manejan grupos de cientos de miles de imágenes, ya que disminuye la dimensión de las matrices que se tendrán que calcular y por ende los cálculos requeridos se aceleran.

En las redes neuronales convolucionales se conserva la estructura jerárquica mientras la red aprende las representaciones de las características internas y las generaliza. Esto quiere decir que la red neuronal aprende a reconocer patrones en las imágenes. Por ejemplo, las llantas en un coche o las orejas puntiagudas de un gato, y generaliza los patrones detectados para realizar sus clasificaciones. Con esto se han podido resolver problemas de reconocimiento de objetos y visión de computadora[26]. Otra ventaja, relacionada a las convoluciones, es que los valores de entrada se hacen invariantes ante rotación, inversión y otras operaciones de simetría. Lo cual permite una mayor flexibilidad al momento de procesar imágenes para la red neuronal ya que la red ahora puede detectar patrones dentro de las categorías a clasificar. En otras palabras, si nuestra red clasificara imágenes de perros, sería irrelevante la posición u orientación relativa del perro en la imagen. Situación que no necesariamente se cumple con otras arquitecturas de

redes neuronales.

2.4.1. Elementos de las Redes Neuronales Convolucionales

Una red neuronal convolucional (CNN) consta de varios elementos clave, algunos de los cuales son:

- Capa de entrada: esta capa recibe los datos de entrada, que suelen ser imágenes en formato matricial.
- Capas de convolución: estas capas aplican un filtro a los datos de entrada, que se mueve por la imagen aplicando una operación matemática, el producto interno de Frobenius, a los píxeles cercanos. El objetivo de estas capas es detectar patrones específicos en los datos de entrada, como bordes o formas.
- Capas de *pooling*: estas capas reducen la dimensionalidad de los datos, resumiendo las características detectadas en las capas de convolución.
- Capas totalmente conectadas: estas capas combinan las características detectadas en las capas anteriores para producir una salida.
- Capa de salida: esta capa produce la salida final del modelo, que suele ser una clasificación o una predicción.
- Filtros o *kernels*: son matrices de pesos que se aplican en las capas de convolución para detectar patrones específicos en los datos de entrada.
- Funciones de activación: son funciones para introducir no linealidad en la red y permitir que aprenda patrones complejos.

- Función de pérdida: es una función matemática que se utiliza para medir el error en el modelo y guiar el proceso de entrenamiento.
- Algoritmo de optimización: es el algoritmo utilizado para actualizar los pesos de la red neuronal durante el proceso de entrenamiento.

Las capas en una red neuronal convolucional se construyen a partir de mapas de imágenes, las matrices que representan a una imagen, y los filtros, lo cual se presenta en la **figura 2.6**, donde la imagen es la primera matriz, a la que se le aplicará una convolución con el filtro presentado en la imagen, después se pasará por otro filtro, en este caso un filtro *MaxPooling*, y se obtendrá una matriz final a la que denominaremos mapa de características. La forma en que las computadoras visualizan cualquier imagen es en realidad mediante una serie de matrices, que con sus valores numéricos le indican a una computadora los valores que debe de adquirir cada píxel para en conjunto representar la imagen en un monitor.

Es de destacar que en una red neuronal convolucional los filtros se inicializan de forma aleatoria y de las diferentes iteraciones aleatorias, la red neural aprende cuales son los filtros más eficientes para detectar patrones para resolver el problema dado.

Para entender cómo interactúan los mapas de imágenes con los filtros supongamos que partimos del siguiente mapa de imagen:

$$\begin{pmatrix} 1{,}00 & 3{,}00 & 2{,}00 & 1{,}00 \\ 2{,}00 & 9{,}00 & 1{,}00 & 1{,}00 \\ 1{,}00 & 3{,}00 & 2{,}00 & 3{,}00 \\ 5{,}00 & 6{,}00 & 1{,}00 & 2{,}00 \end{pmatrix}, \tag{2.6}$$

y definimos dos filtros: Un filtro *Max Pooling* que regresa el valor máximo, fijamos su tamaño en $2x2$ con un salto de 2 y otro filtro *Average Pooling* que regresa el valor promedio, el cual tendrá el mismo tamaño

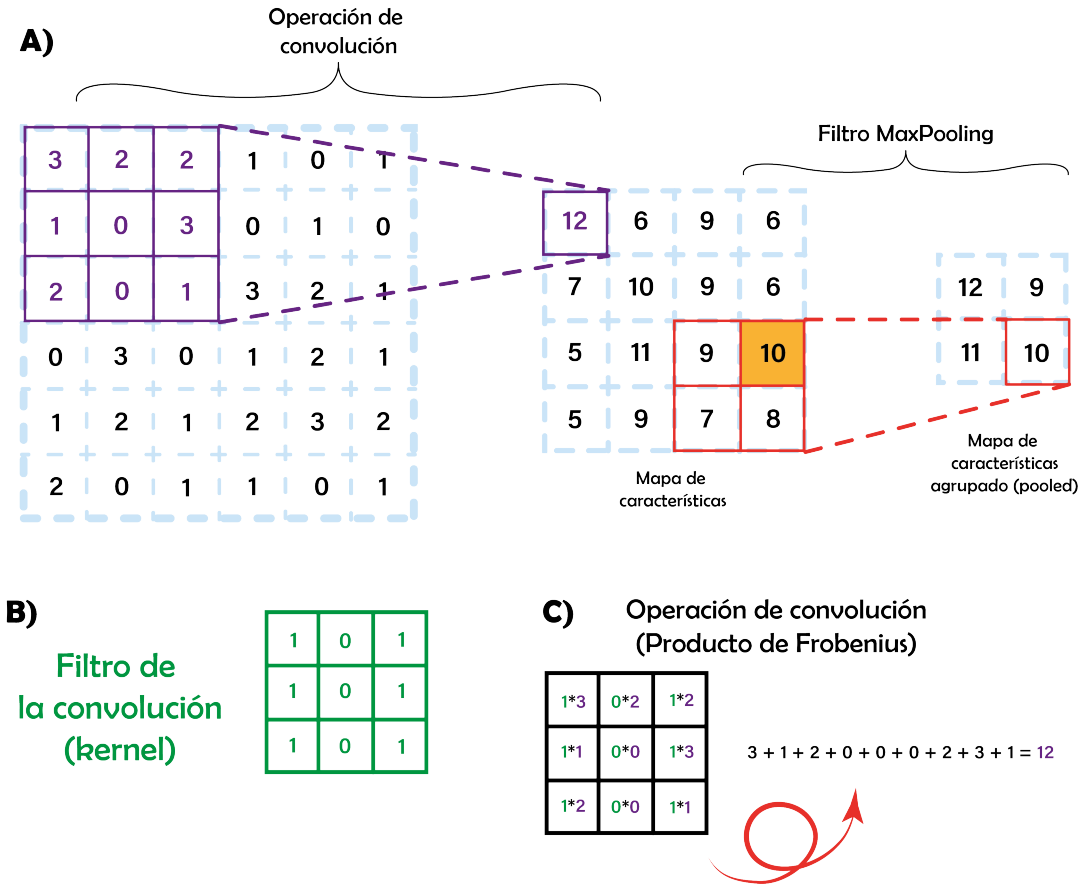


Figura 2.6: Ejemplo de una convolución con un filtro *MaxPooling*.

y salto que el filtro *Max Pooling* previamente definido. Al aplicar el filtro *Max Pooling* al mapa de imagen obtenemos la siguiente matriz

$$\begin{pmatrix} 9{,}00 & 2{,}00 \\ 6{,}00 & 3{,}00 \end{pmatrix}, \tag{2.7}$$

y al aplicar el filtro *Average Pooling* al mapa de imagen obtendremos la siguiente matriz

$$\begin{pmatrix} 3{,}75 & 1{,}25 \\ 3{,}75 & 2{,}00 \end{pmatrix}. \tag{2.8}$$

La imagen que se obtiene después de aplicar un filtro se llama mapa de características. Existen diferentes tipos de filtros y se pueden construir filtros que aprendan a reconocer patrones como bordes, siluetas o formas. Aunque en realidad la red neuronal es la que en sus diferentes iteraciones va construyendo los filtros que necesite para detectar patrones. Después de las operaciones de convolución se debe aplicar una capa *flatten*, que transforma las imágenes obtenidas de los filtros en vectores, o matrices de $1 \times n$, para pasarlos a una red neuronal tradicional.

Cada vez que se realiza una convolución la dimensión de la matriz disminuye, pero aumenta el número de matrices de acuerdo con los filtros aplicados. Por ejemplo, en la **imagen 2.7** partimos de 4 matrices de entrada y se aplican 2 filtros por convolución, luego se aplica un filtro que reduce la dimensionalidad de las imágenes con lo que se obtienen 8 matrices más pequeñas (4 matrices por 2 filtros), este proceso se suele realizar varias veces antes de convertir esas matrices en vectores y pasarlas a una red neuronal.

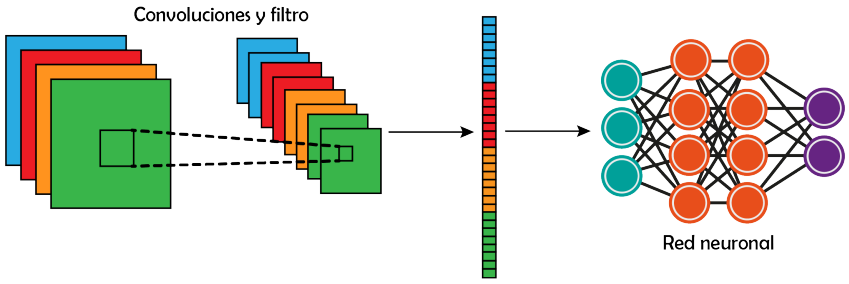


Figura 2.7: Proceso de convoluciones en una red neuronal convolucional.

Cada filtro puede detectar diferentes características de las matrices originales y entre más convoluciones se apliquen, más abstractas serán las características abstraídas de las matrices originales. Un ejemplo del proceso de convolución aplicado a la imagen de un gato se muestra en la seción 7.7.

2.4.2. Entendiendo lo que una CNN puede ver

Debido a que diversos modelos basados en CNN han logrado obtener excelentes resultados en problemas de visión de computadora y han superado a diversos modelos tradicionales de redes neuronales, se ha generado un gran interés por entender los mecanismos subyacentes dentro de los modelos CNN. Para mejorar el rendimiento de los modelos, identificar las razones que generan que un modelo se llegue a equivocar e incluso para generar un modelo que nos ayude a resolver un problema complejo mediante técnicas diferentes a las usadas por redes neuronales. En un principio, se veían las redes neuronales como cajas negras que tomaban datos de entrada y podían clasificarlos o identificar patrones subyacentes en los datos. Para evitar esto se han realizado importantes esfuerzos para construir modelos que complementen a diferentes arquitecturas de redes neuronales y nos ayuden a identificar los complejos mecanismos que usa un modelo de redes neuronales para generar una clasificación.[40]

En el caso de las redes CNN hay diferentes propuestas exitosas que nos permiten visualizar los elementos de las imágenes a los cuales la red les da un mayor peso para generar una clasificación.[40,50] En este trabajo, decidimos usar el algoritmo Grad-CAM (*Gradient-weighted Class Activation Mapping*), que se explica en la sección 7.8. Dicho modelo es una mejora de un grupo de algoritmos llamados CAM (*Class Activation Mapping*) y que ha tenido buenos resultados identificando las secciones más relevantes en las matrices de entrada, para una amplia variedad de algoritmos basados en CNN.[40] La aproximación de este algoritmo consiste en aplicar un gradiente a los vectores previos a la capa de clasificación en la red neuronal, dicho gradiente se calcula con respecto a la última capa convolucional para producir un mapa donde se remarcan las secciones que más impacto tienen en la clasificación final.

2.5. Ingeniería de características

La calidad de los datos y su representación tienen una gran influencia en la eficacia de los algoritmos de aprendizaje profundo. En este contexto, la ingeniería de características, también conocida como injerencia de características o por su nombre en inglés *feature engineering*, se refiere al proceso de transformar los datos de entrada en bruto en una forma que sea más adecuada para el modelo. Mediante este proceso, se busca crear, transformar y seleccionar características en los datos que se van a utilizar para entrenar al modelo de forma óptima. Estas técnicas son altamente específicas para cada problema y usualmente requieren del conocimiento de expertos en el área para identificar los patrones más significativos.[2]

Este es un paso importante en el proceso de aprendizaje profundo, ya que puede mejorar drásticamente el rendimiento del modelo. Aunque puede ser un proceso que requiera mucho tiempo es crucial para que el modelo pueda aprender las características más importantes de los datos. Otra ventaja de este paso es que una buena ingeniería de características puede disminuir la cantidad de datos requeridos para entrenar un modelo de aprendizaje profundo sin afectar el rendimiento final del modelo.

Para lograr un mejor rendimiento en nuestro modelo CNN, diseñamos un enfoque llamado RRCS-Ballesteros-Matrices (RBM), Dicho enfoque, basado en la ingeniería de características, sirve para extraer la información relacionada con los contactos entre los segmentos de la GPCR. La aproximación RBM consiste en obtener dos matrices de contacto que codifican la información relacionada con qué segmentos de los GPCR tienen contactos; la primera es la matriz RRCS, en

la que sus contactos se miden utilizando la puntuación RRCS y la segunda, llamada matriz Ballesteros, usa la distancia euclidiana entre C-β en los aminoácidos de la cadena principal en la proteína. El uso de la representación RBM y la red neuronal usada en este trabajo produjeron buenos resultados con una cantidad relativamente pequeña de datos.

Capítulo 3

Hipotesis

3.1. Hipótesis

La capacidad de clasificación del modelo 0 (92,85 %) se puede mejorar aumentando la cantidad de ejemplos usados para el entrenamiento del modelo, usando una representación de los datos enriquecida mediante la injerencia de características y modificando la arquitectura de la red neuronal del algoritmo. Adicionalmente, se pueden vincular los patrones detectados por la red neuronal con contactos relevantes para el cambio en el estado funcional de GPCRs de la clase A y se pude vincular lo detectado por la red neuronal con la conformación estructural del GPCR analizado.

Capítulo 4

Metodología

4.1. Construcción del conjunto de datos de entrenamiento

4.1.1. Obtención y filtrado de los archivos estructurales

Debido a que los GPCR de la clase A son los más comunes y estudiados,[7] estos cuentan con la mayor cantidad de información estructural disponible en bases de datos. Por lo que el conjunto de datos usados para entrenar el algoritmo de redes neuronales de este trabajo está conformado por estructuras de esta familia. La cantidad de archivos encontrados en GPCRdb [32] para cada familia están descritos en la tabla 4.1. En esta tabla podemos observar que los archivos de la clase A son por mucho los más abundantes y los ligandos con mayor abundancia en las familias más comunes son los agonistas y antagonistas, lo que se puede observar en la tabla 4.2. También podemos ver que, en general, las demás clasificaciones de ligandos contienen muy pocos archivos estructurales. De los 612 archivos estructurales solo se con-

sideraron aquellos que estaba adecuadamente etiquetados, lo que nos dejó con 563 archivos para la clase A, como se puede observar en la tabla 4.2, de los cuales solo se consideraron 486 para entrenar al modelo. Aquellos con ligandos agonista y antagonista, que eran los disponibles al momento de construir el modelo.

Tabla 4.1: Número de archivos estructurales disponibles por clase en GPCRdb. [32]

Clase	**Cantidad de archivos estructurales**
A	612
B1	90
B2	9
C	61
D1	1
F	20

Al desarrollar las primeras pruebas de la red neuronal se consideraron todos los archivos estructurales de los GPCR con los que contábamos, independientemente de la clase a la que pertenecen. De ahí se obtuvo que, debido a la cantidad reducida de datos, las clases menos comunes eran difíciles de clasificar. Cuando se descartaron a los archivos de las clases menos comunes en el grupo de datos de entrenamiento, los resultados de las clasificaciones mejoraron significativamente en las pruebas preliminares de los modelos. Debido a esto se descartó usar estos archivos del conjunto de datos final que se usó para el entrenamiento del modelo que se presenta en este trabajo.

Para construir el conjunto de datos, se siguió el diagrama de flujo de la figura 4.1. Primero identificamos las estructuras a usar mediante GPCRdb [32] y obtuvimos sus respectivos códigos de RCSB PDB. Una vez que se contaba con los códigos de las estructuras, el siguiente paso fue obtener los archivos con coordenadas tridimensionales de RCSB PDB. Posteriormente separamos al conjunto de datos en tres subgrupos; entrenamiento, validación y prueba.

Tabla 4.2: Incidencia de distintos tipos de ligandos de los GPCR de las clases A, B1 y C, disponibles en GPCRdb. [32]

Función del ligando	**Clase A**	**Clase B1**	**Clase C**
Agonista	300	67	11
Antagonista	194	2	6
Agonista inverso	41	0	0
Agonista (parcial)	13	2	0
Desconocido	9	0	0
Antagonista alostérico	4	0	0
Agonista alostérico	1	0	0
NAM (Negative Allosteric Modulator)	1	8	11

4.1.2. Construcción de las representaciones Matrices de RRCS-Ballesteros (MRB)

Debido a que no es conveniente pasar la información de los archivos PDB sin procesar a un modelo de redes neuronales, se realizó un preprocesamiento de estos archivos. En este preprocesamiento se aplicó un enfoque de ingeniería de características para crear una representación del conjunto de datos adecuada para el modelo y de esta forma lidiar con la carencia de suficientes ejemplos para entrenar al modelo de redes neuronales.

Para conseguir un rendimiento óptimo del modelo CNN, se diseñó una representación a la que llamamos Matrices de RRCS-Ballesteros (MRB) o en inglés *RRCS-Ballesteros-Matrices (RBM)*. La representación RBM parte de generar dos matrices de contactos personalizadas que codifican la información relacionada con los segmentos de los

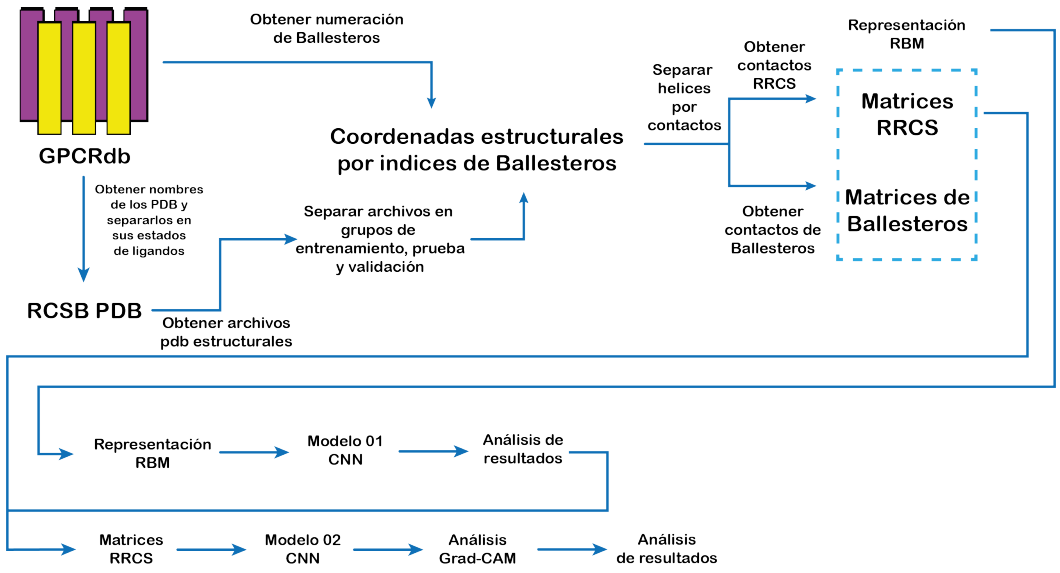


Figura 4.1: Diagrama de flujo del presente proyecto, desde la construcción de las matrices RBM, hasta el desarrollo y análisis de los modelos generados.

GPCR que entran en contacto. La primera matriz se denominó RRCS, debido a la métrica con la que se miden los contactos, que en este caso es la puntuación RRCS. La segunda matriz, llamada matriz de Ballesteros, utiliza la distancia euclidiana entre todos los C-β en los aminoácidos de la cadena principal en la proteína, como métrica para determinar un contacto.

El código para obtener las matrices RBM se encuentra disponible en https://github.com/As337N/GPCR_fs_classifier. Para obtener ambas matrices primero se tuvo que obtener la información del esquema de numeración de Ballesteros-Weinstein. Una vez que se obtuvieron los números alternativos para cada residuo en los GPCR se realizó un análisis de la longitud de cada sección en los GPCR usando sus números alternativos. Con este análisis separamos las hélices de los GPCR en secciones que abarcaban un aproximado de 10 aminoácidos por sección. Las secciones de los extremos de las hélices son las que se aproximan a los 10 aminoácidos ya que dependiendo de la hélice pue-

den contener más o menos aminoácidos. Por otro lado, las secciones intermedias de las hélices son las que si cumplen con 10 residuos por sección.

Este preprocesamiento se realizó solamente con la intención de generar matrices de contactos más pequeñas, dado que se cuantificarían únicamente los contactos establecidos entre secciones en lugar de los contactos entre todos los pares de residuos de las proteínas. En caso contrario, las matrices generadas tendrían un tamaño que impediría su procesamiento computacional debido a las limitaciones de memoria RAM de la tarjeta gráfica (GPU) con la que se trabajó (Una GPU RTX 2070 con 8 GB de RAM). Otra ventaja de este preprocesamiento es que no teníamos que trabajar con longitudes variables debidas a los contactos entre todos los pares de aminoácidos. En cambio, se normalizó la longitud de las estructuras al considerar los contactos entre secciones. Ya que todos los GPCR cuentan con las secciones seleccionadas para representarlos, las longitudes de las representaciones finales de los GPCR eran las mismas independientemente de su cantidad de aminoácidos en sus secuencias.

Una vez que se identificaron las secciones a trabajar con los GPCR se mapearon todas las estructuras del conjunto de datos para ser representadas por la nueva notación de secciones. Se guardó la información de qué residuos pertenecían a cada sección para posteriormente poder regresar a la notación normal de Ballesteros-Weinstein.

El siguiente paso fue aplicar las métricas de contactos, dependiendo del caso podía ser RRCS o la distancia Euclidiana entre los C-β. Posteriormente se identificó la cantidad de contactos que se establecían entre secciones de acuerdo con el esquema planteado previamente. Con los contactos establecidos entre secciones se construyeron matrices de contactos que codifican dicha información. Estas son las matrices que conforman la representación RBM, con una dimensión de 67×67. Por estructura se podían generar sus dos matrices donde la única diferen-

cia se debe a la métrica usada para determinar los contactos entre secciones.

4.1.3. Construcción y tratamiento de los grupos de datos de entrenamiento, validación y prueba

El grupo de datos con el que se entrenó al modelo de redes neuronales se trató de acuerdo con la figura 4.2. El grupo de datos completo, que consistía con 486 archivos, se separó de forma aleatoria en tres subgrupos; El grupo de entrenamiento con alrededor del 80 % de los archivos (388), el grupo de validación con alrededor del 10 % (50), y por último, el grupo de prueba con los archivos restantes (48) que corresponden aproximadamente al 10 % de los archivos restantes. Cabe destacar que no hay intersección entre los grupos de datos generados, por lo que no hay ningún archivo que pertenezca a más de un subgrupo.

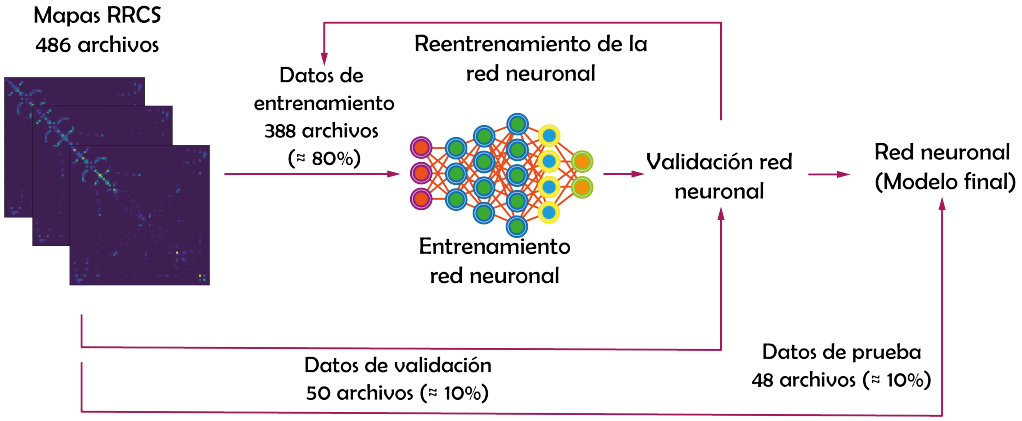


Figura 4.2: Manejo de las matrices RBM en sus diferentes grupos; El de entrenamiento, validación y prueba.

4.2. Construcción de la red neuronal convolucional

La red neuronal construida para este proyecto utilizo la API de TensorFlow para Python y la librería Keras. Un esquema general de las arquitecturas construidas se encuentra en la figura 4.3.

Para este proyecto se construyeron dos modelos, que denominaremos modelo 01 y modelo 02. Empezaremos con la descripción del modelo 01 y continuaremos con la del modelo 02 que está basado en el modelo 01. Se presenta un diagrama de la arquitectura del modelo 01 en la figura 4.3 (a). Como se puede observar, el modelo 01 consiste en dos redes neuronales convolucionales. Una de ellas toma las matrices de Ballesteros y la otra toma las matrices RRCS. Después del proceso de convoluciones con sus respectivos filtros se construyen vectores a partir de las matrices generadas, se concatenan los dos vectores generados y el nuevo vector se pasa a una tercera red neuronal que procesa la información extraída por las CNN, de esta forma se conjuntan los resultados de ambas redes convolucionales. La tercera red se encarga de generar las predicciones finales.

Podemos ver que para la entrada encargada de procesar las matrices RRCS se tienen dos capas convolucionales con sus respectivos filtros *MaxPooling*. Después de las convoluciones, se aplica una capa *Flatten* unida a dos capas de neuronas oculta. Por otro lado, la entrada de Ballesteros esta seguida por prácticamente la misma arquitectura que la presente en la entrada RRCS, a excepción de que solamente tiene una capa de neuronas ocultas al final, en lugar de dos. Después, se concatena la salida de ambas entradas y se pasan los vectores concatenados a dos nuevas capas densas de neuronas donde se realiza la clasificación

final del modelo.

Posteriormente vimos que los resultados de este modelo presentaban dificultades al ser analizados por el algoritmo Grad-CAM, lo que se verá a detalle en la siguiente sección. Debido a esto, decidimos construir un segundo modelo, el modelo 02, diseñado específicamente para ser compatible con el algoritmo Grad-CAM.

El modelo 02 parte del modelo 01, donde solamente usamos la CNN que procesa las matrices RRCS. Debido a que prácticamente quitamos la mitad de la información que procesaba el modelo 01, tuvimos que reducir la cantidad de capas ocultas en el modelo 02 para evitar el sobreaprendizaje de la red neuronal sobre el conjunto de datos.

Podemos ver que el modelo 02, figura 4.3 (b), consiste en una capa de entrada unida a una capa convolucional con su filtro *MaxPooling*, después se aplica una capa *Flatten* que construye un vector a partir de los mapas de características generados por la convolución. Ese vector se pasa a capas de neuronas ocultas, denotadas en la figura por las neuronas *Dense* y se aplica una función de *Dropout* que ayuda a evitar el sobreajuste. Por último, se pasan los resultados de la red a una última capa de neuronas que genera la clasificación.

4.2.1. Entrenamiento de la red neuronal y evaluación

Las redes neuronales descritas previamente se entrenaron usando las funciones prediseñadas de Keras. Se uso una función de entropía cruzada binaria como función de costo y el algoritmo Adam como optimizador.[36] Los modelos usaron el conjunto de validación para mejorar los parámetros de la red mediante retropropagación y se entrenaron por 20 épocas. Adicional a esto, se usó una función *callback* de TensorFlow para guardar los parámetros del modelo si se satisfacía una condición de acierto superior al 92 % y un valor de la función de perdida menor a 0,5.

4.3. Análisis del algoritmo *Grad-CAM* sobre modelos seleccionados

---

Se implemento el algoritmo *Grad-CAM* en los modelos 01 y 02 para detectar los contactos más relevantes para cada estado funcional. Para aplicar dicho algoritmo, se siguió lo establecido en la metodología disponible en (https://github.com/keras-team/keras-io/blob/master/examples/vision/grad_cam.py). La información que se obtuvo de aplicar *Grad-CAM* sobre ambos modelos nos indicó que secciones en la matriz de contactos que tienen una mayor influencia para cada estado funcional. En otras palabras, se puede decir que *Grad-CAM* nos indica en qué partes de las matrices se fija la red neuronal convolucional para generar una predicción.

En este paso, surgió un problema relevante. Debido a que en este trabajo sólo se consideraron dos clases; agonista y antagonista, se construyeron modelos de clasificación binarios. Por otro lado, *Grad-CAM* se diseñó para clasificaciones no binarias, aquellas con más de dos posibles clases. Esto resultó en que, al aplicar *Grad-CAM* fácilmente obteníamos valores de gradientes con valores muy cercanos a cero. Estos valores eran entendidos como ceros en los mapas *Grad-CAM* e impedían un análisis posterior de dichos mapas. Para solventar este inconveniente se aplicó el algoritmo *Grad-CAM* sobre la capa anterior a las neuronas con la función sigmoide que es la que genera el problema de clasificación binaria. Está solución nos convenía, ya que evitaba variar la cantidad de neuronas de la capa final o la función de activación de dicha capa.

Al evaluar los resultados de *Grad-CAM* sobre el modelo 01, nos percatamos de que los mapas *Grad-CAM* de las matrices RRCS eran

más significativos que aquellos obtenidos de las matrices de Ballesteros. Con base en esto se construyó el modelo 02 que solamente tomaba como entradas a las matrices RRCS.

Para finalizar, se obtuvieron los mapas *Grad-CAM* para cada caso en el grupo de datos de prueba, como se describe en la figura 4.4. Para analizar los resultados se juntaron todos los mapas generados por clase. El proceso consistía en primero dividir los valores de intensidad de cada valor en los mapas *Grad-CAM* entre el número total de casos por clase. Luego, para cada clase se sumaron los valores de intensidad de todos los nuevos mapas. Las matrices generadas corresponden con un promedio de los mapas, que nos permiten detectar los patrones de los valores de contactos entre secciones más relevantes para cada clase. Un ejemplo visual de este proceso se presenta en la figura 4.4.

Las secciones resaltadas por *Grad-CAM* se pueden vincular a contactos en las proteínas. Dichos contactos nos ayudan a determinar cuáles son las interacciones cruciales para los estados funcionales considerados en este trabajo. Después del análisis, es fundamental comparar los resultados obtenidos con la información experimental disponible. Este paso nos puede ayudar a entender la validez de la metodología planteada en este trabajo, ya que si encontramos una clara relación entre la información experimental y los mecanismos de atención de los modelos de redes neuronales desarrollados podemos tener confianza en que la capacidad de predicción de los algoritmos está fundamentada en características relevantes de los contactos entre secciones de los GPCR. Por ende, las predicciones de las redes neuronales construidas no son producto del azar.

Para conjuntar los resultados experimentales y los generados por el modelo 02, se tuvo que preparar a los datos experimentales de tal forma que se pudieran comparar directamente ambas fuentes de información. El proceso para los datos experimentales fue el siguiente; Primero se obtuvieron los datos experimentales de contactos fundamentales para

los GPCR de la clase A con ligandos agonistas y antagonistas. Para este paso se usó el trabajo descrito en la sección 2.2 de Hauser y colaboradores,[13], debido a que fue la fuente más completa y actualizada que encontramos de un análisis de los contactos experimentales para este grupo de proteínas. Una vez que se contaba con la información experimental, el segundo paso consistía en presentar la información de estos contactos de acuerdo con la matriz de secciones generada para este trabajo. Por lo que se mapearon los contactos experimentales con cada una de las secciones consideradas para las matrices de contactos por secciones. Por último, se pasó la información de los contactos experimentales por secciones a su representación matricial. A estas matrices generadas las denominamos matrices de Hauser y contienen la información de los contactos experimentales más relevantes por sección de acuerdo con el formato usado para este trabajo.

Con los mapas *Grad-CAM* promediados y las matrices Hauser ya listos, el siguiente paso consistía en simplemente sobreponer ambas matrices y ver las secciones en las que se podían encontrar similitudes, identificando aquellas secciones donde se dé un traslape entre ambas matrices comparadas.

Una vez que se identificaron los contactos fundamentales para predecir el estado funcional de cada proteína de acuerdo con el algoritmo, el paso final en el análisis de los contactos consistía en visualizar dichos contactos en las proteínas. Debido a esto, se construyó un algoritmo que partía de los mapas *Grad-CAM* y regresaba los aminoácidos más relevantes de cada proteína para la clasificación de su estado fundamental, de acuerdo con el modelo 02. Los resultados de este análisis se visualizaron en Pymol.

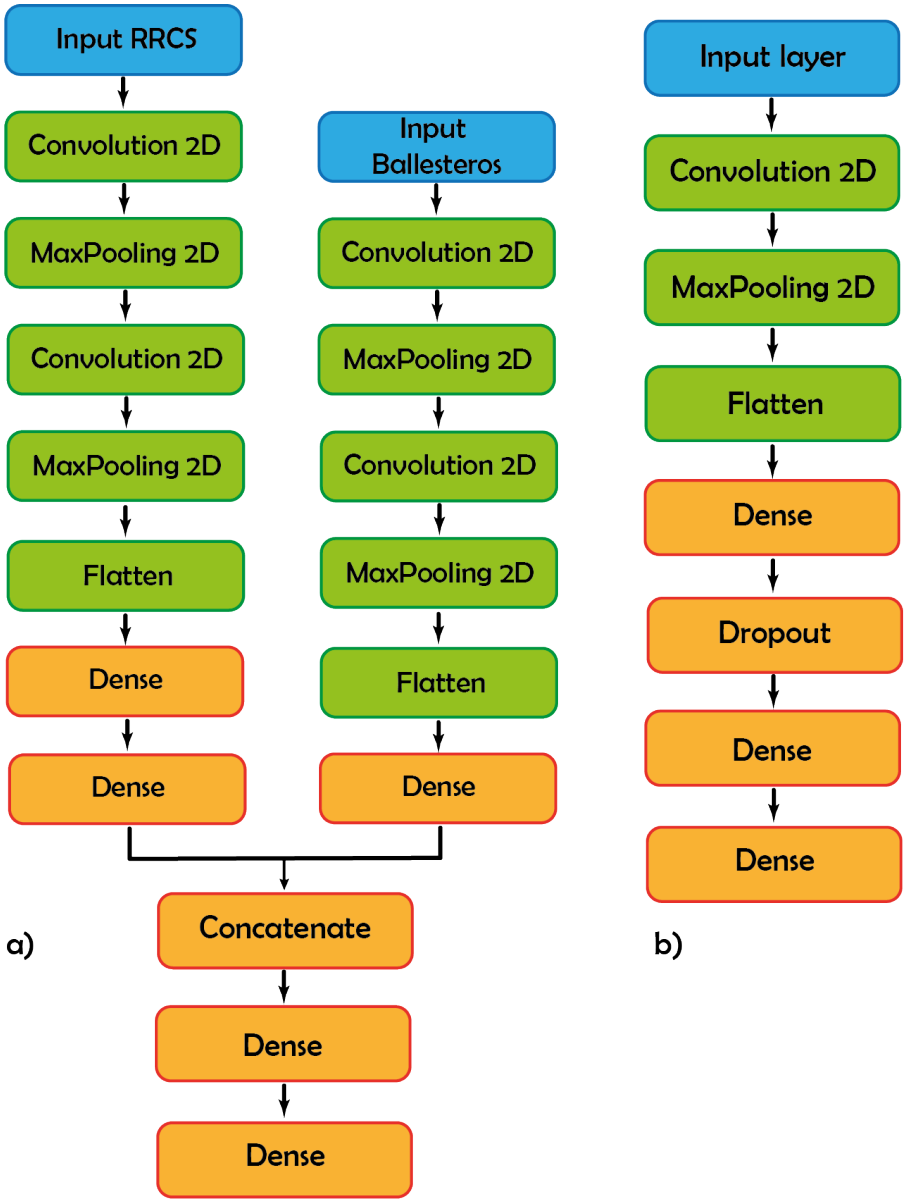


Figura 4.3: a) Estructura del modelo 01, este modelo consta de dos entradas, una para las matrices RRCS y otra para las matrices de Ballesteros. Cada entrada está unida a una CNN adaptada para procesar de forma eficiente sus matrices. La salida de ambas CNN se concatena en una nueva red neuronal que es la encargada de generar las predicciones. b) estructura del modelo 02. Consta de una entrada para las matrices RRCS, que está unida a una CNN adaptada para procesar de forma eficiente la matriz RRCS. Después se encuentra el proceso de convolución, donde se pasan los resultados de los mapas de características a tres capas densas con una función de *dropout* intermedia para evitar el sobreaprendizaje.

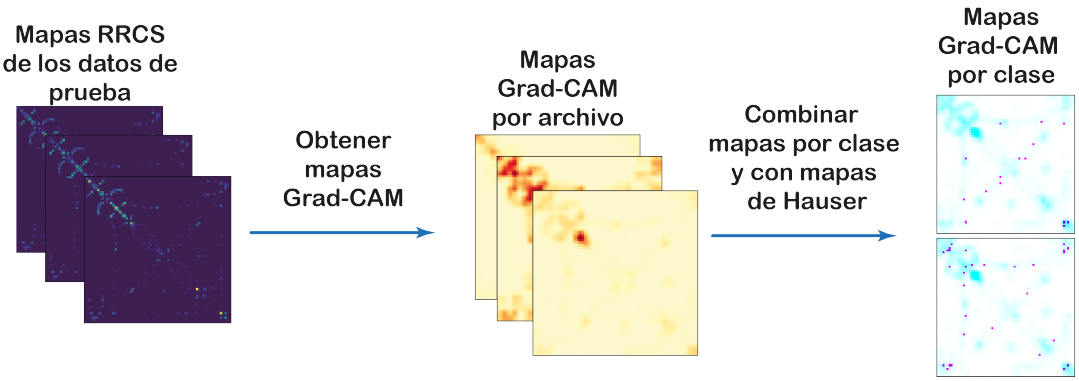


Figura 4.4: Pasos para procesar los mapas Grad-CAM. Primero se obtuvieron los correspondientes mapas para cada archivo predicho del grupo de datos de prueba. Después de eso se combinaron los resultados de los mapas por clase.

Capítulo 5

Resultados y análisis de resultados

5.1. Modelos CNN

Después de procesar el conjunto de datos de prueba en el modelo 01, se obtuvo una tasa de acierto del 95,83 %, una puntuación f1 del 95,86 % y un valor de pérdida de 0,33. Es relevante recordar que el conjunto de datos de prueba fue un conjunto de datos que no se usó para el entrenamiento del modelo 01.

Los resultados completos del modelo 01, se resumen en la tabla 5.1. En esta tabla, las clasificaciones erróneas del algoritmo se denotan con rojo, mientras que los aciertos se denotan con verde. Como el modelo construido fue de aprendizaje supervisado nosotros conocíamos previamente las etiquetas reales de cada caso del conjunto de datos de prueba, por lo que pudimos comparar las predicciones del modelo 01 con los valores reales. Adicionalmente se pudo analizar la calidad de las predicciones mediante los valores obtenidos con la función de perdida. Como se describió en los antecedentes (sección 2.3.4), la función de pérdida mide qué tanto se alejan los valores predichos de los valores verdaderos. El valor idealmente esperado es de cero y el resultado es

más cercano a su valor verdadero entre más cercano a cero sea el valor de la función de pérdida. Es relevante notar que para obtener este valor se requiere conocer previamente el valor verdadero del ejemplo analizado por la red neuronal. Por lo que en un sistema de predicción real donde no se conoce dicho valor verdadero, no se puede usar la función de pérdida como métrica para analizar la predicción.

Tabla 5.1: Predicciones del modelo 01, usando el grupo de datos de prueba. Se compara los resultados del modelo (actividad predicha) contra los valores verdaderos (actividad descrita), adicionalmente se anexa el valor de la pérdida para cada predicción. Los aciertos del modelo se remarcan en verde y los errores en rojo.

PDB ID	Actividad predicha del ligando	Actividad descrita del ligando	Pérdida	PDB ID	Actividad predicha del ligando	Actividad descrita del ligando	Pérdida
5VRA	Antagonista	Antagonista	7.1526E-7	6MXT	Agonista	Agonista	8.2913E-5
3NYA	Antagonista	Antagonista	0.1633	6ME8	Agonista	Agonista	0.0087
3V2W	Antagonista	Antagonista	0.0040	7NA7	Agonista	Agonista	0.0082
5JTB	Antagonista	Antagonista	0.0005	4PXZ	Agonista	Antagonista	8.2831
6GT3	Antagonista	Antagonista	0.0002	7L0R	Agonista	Agonista	0.0001
5OM1	Antagonista	Antagonista	0.0004	7BB7	Agonista	Agonista	0.0092
5YWY	Antagonista	Antagonista	0.0188	6UP7	Agonista	Agonista	0.0013
4S0V	Antagonista	Antagonista	0.0019	6LI0	Agonista	Agonista	0.6043
4EIY	Antagonista	Antagonista	0.0002	7VUI	Agonista	Agonista	0.0014
5O9H	Antagonista	Antagonista	0.0002	7WQ3	Agonista	Agonista	0.0006
5IUB	Antagonista	Antagonista	0.0007	6ME6	Agonista	Agonista	0.0104
5CXV	Antagonista	Antagonista	0.2031	7U2L	Agonista	Agonista	0.0056
5OLG	Antagonista	Antagonista	0.0005	7EJ0	Agonista	Agonista	0.0026
7JNI	Antagonista	Antagonista	0.4581	7WC8	Agonista	Agonista	0.1429
6IIV	Antagonista	Antagonista	0.0006	7EIB	Agonista	Agonista	0.0203
5MZP	Antagonista	Antagonista	0.0003	6DRY	Agonista	Agonista	0.0277
5NLX	Antagonista	Antagonista	1.5497E-6	6G79	Agonista	Agonista	0.0017
5UIW	Antagonista	Antagonista	0.0027	6FUF	Agonista	Agonista	0
6A94	Antagonista	Antagonista	0.4399	6OS2	Agonista	Agonista	0.0001
7T10	Agonista	Agonista	0.0012	6FK6	Agonista	Agonista	0
5VBL	Agonista	Antagonista	5.4059	7TD3	Agonista	Agonista	1.5438E-5
6WHA	Agonista	Agonista	0.0004	7VUJ	Agonista	Agonista	0.0014
7CRH	Agonista	Agonista	0.0067	7VDH	Agonista	Agonista	0.0021
7EW7	Agonista	Agonista	6.6757E-6	3ZEV	Agonista	Agonista	0.0006

Podemos destacar que el valor de pérdida se aproxima a cero, a excepción de los casos donde el modelo clasifico erróneamente la actividad del ligando. Otra cosa relevante es que en dos predicciones se obtuvo un valor de cero (6FUF y 6FK6). Al examinar estos valores nos percatamos que los valores reales en realidad son muy cercanos a cero, por lo que TensorFlow, el programa usado para analizar los resultados los almacenó como ceros. También podemos destacar que ambas estructuras (6FUF y 6FK6) corresponden a una rodopsina y una opsina,

respectivamente. La rodopsina es la estructura prototipo de la cual se empezaron a estudiar las estructuras de GPCR, lo que nos hace suponer que al ser los arquetipos de agonistas para los GPCR de la clase A el modelo 01 predice con una certeza casi absoluta las estructuras de estas proteínas. Esta proteína tiene la peculiaridad de estar unida covalentemente al retinal, un cromóforo. Por otro lado, la opsina es la misma estructura sólo que esta únicamente contiene a la proteína de la rodopsina y no incluye al retinal unido covalentemente.[5]

Para el modelo 02, que se diseñó para implementar el algoritmo *Grad-CAM*, los resultados finales del conjunto de datos de prueba son una tasa de acierto del 91,67 %, una puntuación f1 de 91,72 % y una perdida global del 0,29. Las predicciones de este modelo se encuentran descritas en la tabla 5.2.

Tabla 5.2: Predicciones del modelo 02, usando el grupo de datos de prueba. Se compara los resultados del modelo (actividad predicha) contra los valores verdaderos (actividad descrita), adicionalmente se anexa el valor de la pérdida para cada predicción. Los aciertos del modelo se remarcan en verde y los errores en rojo.

PDB ID	Actividad predicha del ligando	Actividad descrita del ligando	Pérdida	PDB ID	Actividad predicha del ligando	Actividad descrita del ligando	Pérdida
5VRA	Antagonista	Antagonista	1.1087E-5	6MXT	Agonista	Agonista	0.0022
3NYA	Antagonista	Antagonista	0.2087	6ME8	Agonista	Agonista	0.0637
3V2W	Antagonista	Antagonista	0.0113	7NA7	Agonista	Agonista	0.0036
5JTB	Antagonista	Antagonista	0.0085	4PXZ	Agonista	Antagonista	2.7926
6GT3	Antagonista	Antagonista	0.0093	7L0R	Agonista	Agonista	0.0018
5OM1	Antagonista	Antagonista	0.0118	7BB7	Agonista	Agonista	0.0032
5YWY	Antagonista	Antagonista	0.0335	6UP7	Agonista	Agonista	0.0804
4S0V	Antagonista	Antagonista	0.0182	6LI0	Agonista	Antagonista	2.4932
4EIY	Antagonista	Antagonista	0.0063	7VUI	Agonista	Agonista	0.0081
5O9H	Antagonista	Antagonista	0.0073	7WQ3	Agonista	Agonista	0.0073
5IUB	Antagonista	Antagonista	0.0118	6ME6	Agonista	Agonista	0.0787
5CXV	Antagonista	Antagonista	0.1192	7U2L	Agonista	Agonista	0.0034
5OLG	Antagonista	Antagonista	0.0126	7EJ0	Agonista	Agonista	0.0255
7JNI	Antagonista	Agonista	2.9330	7WC8	Agonista	Agonista	0.5432
6IIV	Antagonista	Antagonista	0.1219	7EIB	Agonista	Agonista	0.0197
5MZP	Antagonista	Antagonista	0.0095	6DRY	Agonista	Agonista	0.3747
5NLX	Antagonista	Antagonista	4.6254E-5	6G79	Agonista	Agonista	0.0034
5UIW	Antagonista	Antagonista	0.0408	6FUF	Agonista	Agonista	3.6956E-5
6A94	Antagonista	Antagonista	0.5165	6OS2	Agonista	Agonista	7.0221E-4
7T10	Agonista	Agonista	0.0123	6FK6	Agonista	Agonista	8.7563E-5
5VBL	Agonista	Antagonista	3.1027	7TD3	Agonista	Agonista	0.0006
6WHA	Agonista	Agonista	0.0380	7VUJ	Agonista	Agonista	0.0054
7CRH	Agonista	Agonista	0.0048	7VDH	Agonista	Agonista	0.0006
7EW7	Agonista	Agonista	0.0009	3ZEV	Agonista	Agonista	0.1444

Finalmente se construyó una matriz de confusión para ambos mode-

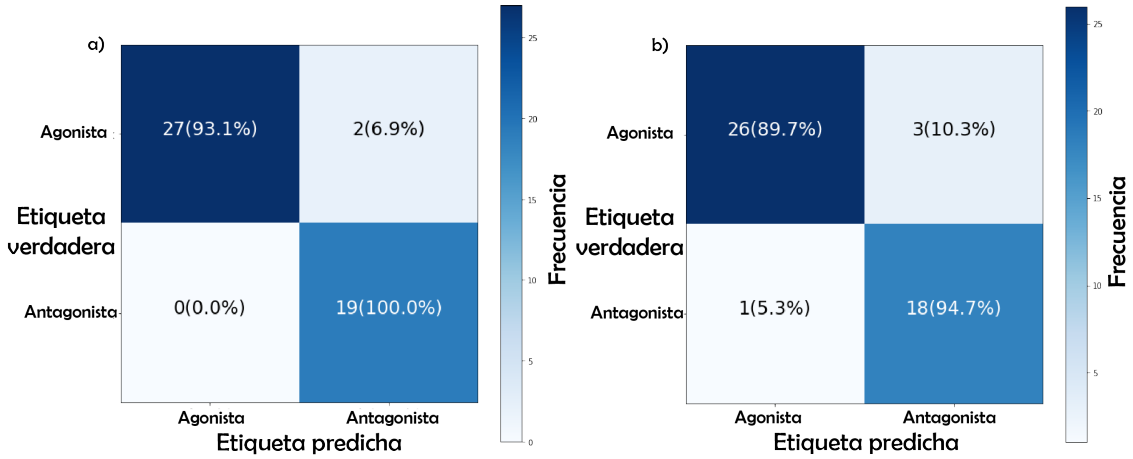


Figura 5.1: Matrices de confusión para las predicciones de ambos modelos construidos. En a) se encuentran las predicciones del modelo 01 y en b) las del modelo 02.

los con la intención de visualizar los resultados generados, las matrices se muestran en la figura 5.1. Una matriz de confusión es una herramienta utilizada en el campo de la estadística y el aprendizaje automático para evaluar el rendimiento de un modelo de clasificación. Se utiliza para visualizar el desempeño del modelo al clasificar correcta o incorrectamente las clases de un conjunto de datos. En dicha matriz, cada fila i representa la clase real de los datos y cada columna j representa la clase predicha por el modelo. Los elementos en $i = j$ son aquellos cuyo valor real fue correctamente predicho por el modelo. Todos los demás valores son errores en la predicción del modelo.

Las curvas ROC (*Receiver Operating Characteristic*) y las puntuaciones AUC (*Area Under the Curve*) para ambos modelos se presentan en la figura 5.2. Ambas son herramientas utilizadas para evaluar y comparar el rendimiento de modelos de clasificación, especialmente en problemas de clasificación binaria.

La curva ROC es un gráfico que representa la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos (1 - especificidad) a medida que se varía el umbral de clasificación del

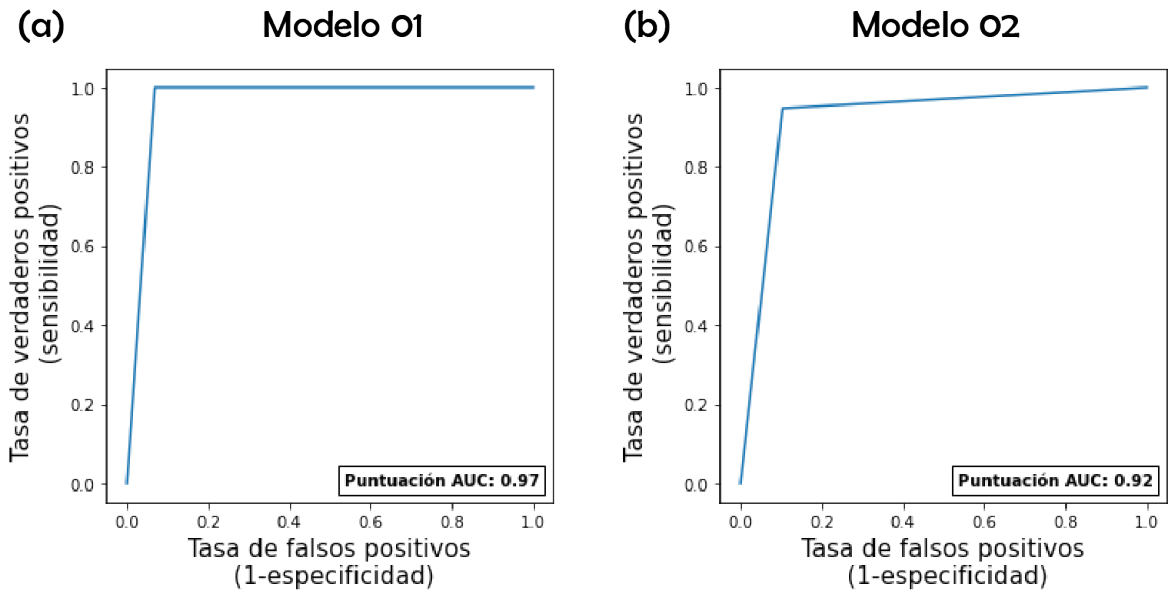


Figura 5.2: Curvas ROC para a) el modelo 01 y b) el modelo 02. Ambas curvas tienen indicado el valor AUC correspondiente.

modelo. Cada punto en la curva ROC representa un umbral diferente, y el gráfico muestra cómo cambia el equilibrio entre la sensibilidad y la especificidad a medida que se ajusta el umbral.

La puntuación AUC es el área bajo la curva ROC. Indica la capacidad de un modelo para distinguir entre las clases positiva y negativa. Un valor de AUC cercano a 1 significa que el modelo tiene un rendimiento bueno, ya que tiene una alta sensibilidad y una baja tasa de falsos positivos. Un valor de AUC cercano a 0.5 indica que el modelo tiene un rendimiento pobre, ya que es similar a una clasificación aleatoria.

Como se puede ver tanto en las curvas ROC como en los respectivos valores AUC ambos modelos son validados como buenos para la tarea de clasificación.

5.2. Algoritmo Grad-CAM

Al aplicar el algoritmo *Grad-CAM*, se obtuvieron matrices que indicaban las secciones de la entrada original que eran relevantes para generar una clasificación en la respectiva CNN. Se nombraron a las matrices generadas como mapas *Grad-CAM*.

Los resultados de adicionar los mapas *Grad-CAM*, azul, a los mapas de Hauser, rosa, se pueden visualizar en la figura 5.3. Como se puede observar, hay bastantes zonas relevantes donde se presenta un traslape, lo cual es un buen indicativo de que el modelo pudo aprender de forma automatizada a distinguir los contactos relevantes para clasificar una proteína en sus estados funcionales debido a la presencia de ligandos agonistas o antagonistas. También es relevante notar las secciones donde el modelo presta atención, pero no están descritas como relevantes de acuerdo con el trabajo de Hauser, o aquellas secciones relevantes de acuerdo con el trabajo de Hauser pero que tienen un peso casi nulo para el algoritmo al momento de discernir entre ambos estados funcionales.

Ambos mapas se ven muy similares a simple vista, lo cual es concordante con el hecho de que los cambios estructurales entre los estados agonista y antagonista son sutiles. A pesar de la aparente similitud, las diferencias son lo bastante grandes como para que el algoritmo CNN usado pueda diferenciar adecuadamente entre ambos estados funcionales. Los valores en los mapas *Grad-CAM* tienen valores entre cero y uno. Un valor de cero nos indica que la sección correspondiente no tiene una gran relevancia al momento de que la CNN genera una clasificación. Por el otro lado los valores cercanos a uno tienen una mayor relevancia para la clasificación, conforme se alejan del cero. Con base en esto se identificaron a los contactos entre secciones que son críticos

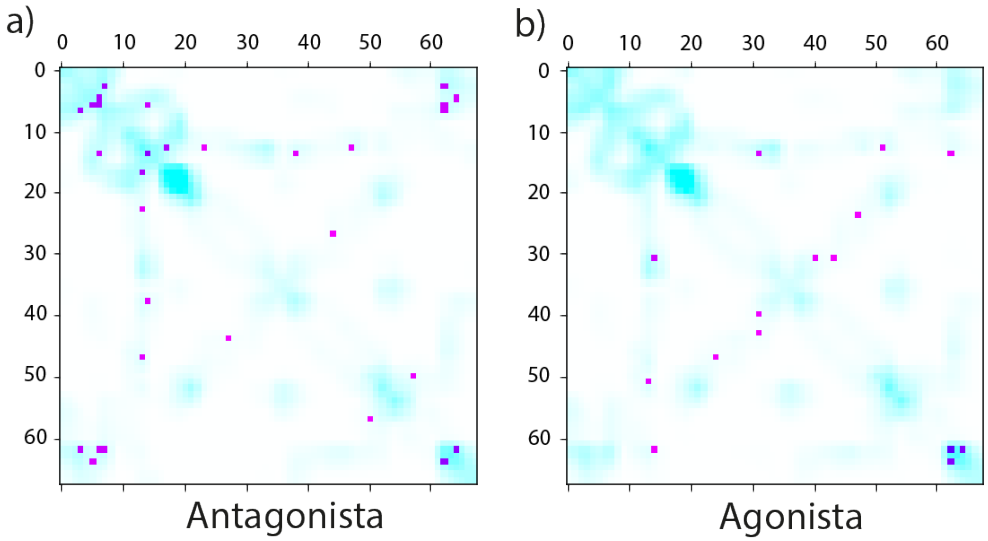


Figura 5.3: Mapas Grad-CAM de antagonistas (a) y agonistas (b) en azul, junto a los contactos experimentalmente relevantes reportados por Hauser en rosa. Cada mapa es el promedio de todos los mapas Grad-CAM por clase para todas las predicciones de los archivos en el grupo de prueba.

para cada estado funcional.

Adicionalmente se analizaron la frecuencia de valores en los mapas Grad-CAM, como se puede ver en la figura 5.4. El histograma indica la distribución de puntuaciones Grad-CAM para los valores en la matriz correspondiente al mapa Grad-CAM. Recordemos que cada coordenada en dicho mapa corresponde con un contacto entre secciones de los GPCR y la puntuación obtenida para dicha coordenada tiene que ver con la relevancia de dicho contacto para la clasificación del modelo CNN.

Como era de esperar la mayor parte de contactos no son relevantes para el modelo CNN, tienen una puntuación Grad-CAM relativamente cercana al cero. Adicionalmente se marcan en lineas punteadas donde se ubican los contactos relevantes reportados por Hauser[13], de acuerdo a nuestro modelo. Es interesante notar que dichos valores terminan

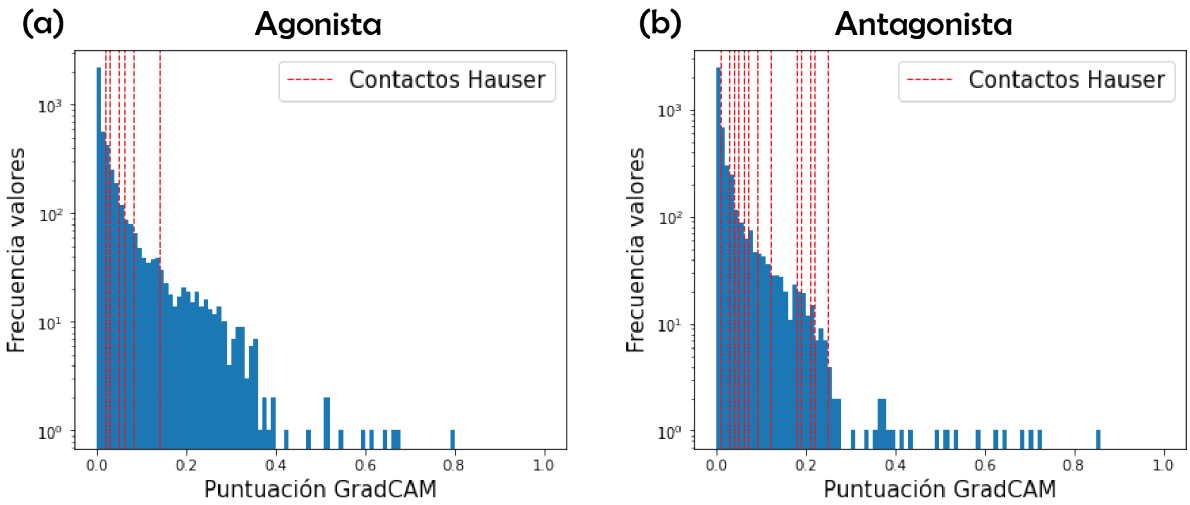


Figura 5.4: Histogramas con la frecuencia de puntuaciones Grad-CAM en las mapas Grad-CAM para a) agonista y b) antagonista. Las lineas rojas punteadas indican la puntuación obtenida para los contactos experimentales descritos por Hauser.

obteniendo valores bajos en la puntuación Grad-CAM, lo que resulta contraintuitivo. Aunque se puede notar que ambos patrones son diferentes, ya que para los ligandos antagonistas se obtienen puntuaciones más altas para algunos contactos.

Los valores para los contactos descritos por Hauser y la puntuación Grad-CAM que obtuvieron de acuerdo a nuestro modelo se presenta en las tablas 5.3 para agonista y 5.4 para ligandos antagonistas.

Para realizar los siguientes análisis sólo se consideraron los valores mayores a 0.5 en los mapas *Grad-CAM*. Con lo que se obtuvo la tabla 5.6, que usa la notación de Ballesteros-Weinstein. Los aminoácidos por hélice cubiertos en las secciones relevantes se describen en la tabla 5.5. Es relevante destacar que ECL2 abarca todos los aminoácidos que pertenecen a este bucle extracelular.

Con base en el análisis de resultados del modelo 02 con *Grad-CAM*, los contactos entre secciones cruciales para su clasificación son aquellos que se establecen entre ECL2 y las secciones extracelulares de TM4 y

Tabla 5.3: Contactos de Hauser y su puntuación Grad-CAM para GPCR interactuando con ligandos agonistas.

Contacto entre secciones	**Puntuación Grad-CAM**
TM34-TM512	0.08
TM34-TM79	0.03
TM512-TM67	0.02
TM512-TM64	0.03
TM79-H81	0.14
TM79-TM79	0.06
TM55-TM611	0.02
TM33-TM615	0.05

TM5. Estos resultados concuerdan con lo descrito en el trabajo de Wingert,[47] donde ECL2 cumple un rol fundamental en el mecanismo de activación de los GPCR que pertenecen a la clase A. En el trabajo de Nicoli[30], se agrupan las conformaciones del ECL2 en GPCR de la familia A por sus secuencias, formas y contactos intramoleculares. ECL2 es el dominio más desafiante de modelar en los GPCR debido a su longitud y diversidad, aunque dichas características contribuyen a la selectividad y afinidad de los ligandos agonistas, antagonistas y ligandos intermediarios con el GPCR.[30]

Por último, se lograron identificar los contactos clave en las secciones descritas como relevantes y visualizarlos mediante Pymol. Este análisis se realizó solamente para los GPCR con el menor valor en su función de pérdida. Estos resultados se muestran en la figura 5.2.

Debido a como se generaron las representaciones de los datos estructurales, no se puede realizar un análisis estructural más detallado con esta metodología. Por lo que queda para posteriores estudios estudiar las zonas indicadas por nuestros resultados para determinar su relevancia en los mecanismos de activación en los GPCR.

Tabla 5.4: Contactos de Hauser y su puntuación Grad-CAM para GPCR interactuando con ligandos antagonistas.

Contacto entre secciones	**Puntuación Grad-CAM**
TM33-TM54	0.05
TM33-TM42	0.21
TM614-TM74	0.03
TM33-TM611	0.04
TM22-TM79	0.06
TM13-TM79	0.06
TM13-TM22	0.22
TM21-TM79	0.05
TM21-TM34	0.12
TM58-TM68	0.01
TM34-TM34	0.25
TM34-TM62	0.04
TM79-H81	0.09
H81-ICL1	0.07
TM21-TM21	0.19
TM21-ICL1	0.18

Tabla 5.5: Aminoácidos considerados por sección con su respectiva notación Ballesteros-Weinstein.

Sección	**Hélice transmembranal**	**Aminoácidos correspondientes, de acuerdo con la notación de Ballesteros**
TM43	TM4	55-75
TM51	TM5	15-35

Tabla 5.6: Contactos entre secciones más relevantes de acuerdo con los valores de los mapas Grad-CAM. Sólo se consideran aquellos contactos con un valor mayor a 0.5 en los mapas Grad-CAM.

Valor Grad-CAM	**Secciones agonistas**	**Secciones antagonistas**
0.5-0.6	TM43-TM43, TM43-TM51	TM43-TM43, TM43-TM51, TM51-TM51
0.6-0.7	TM43-ECL2, TM51-TM51	TM43-ECL2, TM51-ECL2
0.7-0.8	TM51-ECL2	ECL2-ECL2
0.8-0.9	ECL2-ECL2	—

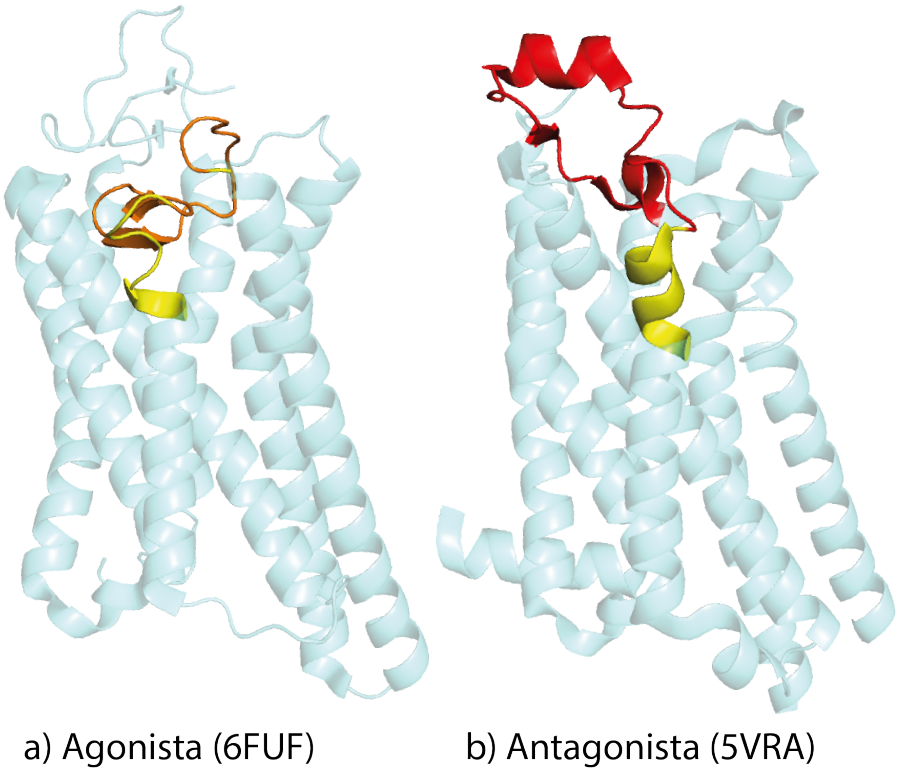


Figura 5.5: Visualización de las secciones más relevantes por color de acuerdo con la tabla 5.6. Rojo corresponde con las secciones con mayor relevancia para el algoritmo, le sigue el naranja y por último el amarillo. Las secciones en azul corresponden con las que tuvieron una puntuación menor a 0.5 en sus matrices *Grad-CAM*.

Capítulo 6

Conclusiones y perspectivas

6.1. Conclusiones

En este trabajo, se logró superar la capacidad predictiva de un algoritmo para clasificar los estados funcionales en estructuras de GPCR. Con el modelo 01 se obtuvo una tasa de acierto del 95,83 %, una puntuación f1 del 95,86 % y un valor de pérdida de 0,33. Para el modelo 02 se logró una tasa de acierto del 91,67 %, una puntuación f1 de 91,72 % y una pérdida del 0,29. Sólo los resultados del modelo 01 superan a los del modelo previo que se buscaba mejorar, el modelo 0, dicho modelo tenía una tasa de acierto del 92,85 %. A pesar del desempeño del modelo 02 con este modelo fuimos capaces de capturar la información estructural de los GPCR de la clase A mediante el análisis de sus correspondientes mapas de contactos con el algoritmo Grad-CAM.

Los mapas de contactos construidos para los modelos 01 y 02 se generaron mediante un enfoque de ingeniería de características, donde se buscó obtener representaciones óptimas de los datos, para un algoritmo de redes neuronales. Usando dichos mapas de contactos logramos clasificar a las estructuras de GPCR representadas de acuerdo

con dos clasificaciones. Las clases consideradas son aquellas que corresponden con las actividades más comunes para los ligandos de GPCR, que son agonista y antagonista. Dicha clasificación de estructuras se realizó usando un algoritmo de redes neuronales convolucionales que extraía las características más relevantes de los mapas de contactos para predecir la actividad del ligando asociado en la estructura del GPCR.

El análisis de los mapas Grad-CAM y los contactos de Hauser nos indican que el modelo 01 usa contacto diferentes a los usualmente reportados en la literatura para generar sus predicciones. Aunque debido a los buenos resultados del modelo para diferenciar entre las clases de ligandos, valdría la pena estudiar a detalle dichos contactos ya que podrían aportar nueva información relevante respecto a los mecanismos de activación dentro de los GPCR.

Posteriormente, se analizaron a las redes neuronales construidas, con la idea de identificar los patrones más significativos usados por los modelos para clasificar los datos. En este análisis, se logró identificar una relación clara entre los contactos relevantes para los modelos y aquellos previamente reconocidos de forma experimental como cruciales en el cambio de estado funcional. El análisis de los modelos CNN sugiere que los contactos que establece ECL2 cumplen un rol crucial en la determinación del estado funcional de un GPCR cuando se une a ligandos agonistas o antagonistas. La relevancia de este bucle ha sido previamente descrita en la literatura, adicionalmente el modelo logró detectar de forma autónoma la relevancia de estos contactos. Aunque se sabe que ECL2 es relevante en el mecanismo de activación de los GPCR de la clase A, no se suelen usar sus contactos como criterio fundamental para determinar el estado funcional de estas proteínas, principalmente debido a la dificultad de modelar este dominio extracelular.

Dado que la información que el modelo usa para sus predicciones

tiene una relación clara con lo previamente descrito de forma experimental, podemos atribuir el buen rendimiento del modelo a que este logró identificar automáticamente los contactos relevantes en los GPCR que favorecen el estado funcional debido a ligandos agonistas o antagonistas.

6.2. Perspectivas

Los algoritmos presentados en este trabajo se pueden continuar perfeccionando para mejorar su capacidad predictiva. Conforme pasan los meses nuevas estructuras de GPCR que interactúan con ligandos son publicadas en RCSB PDB. Debido a esto se cuentan con un conjunto de datos más amplio, lo cual debe de ayudar a mejorar el desempeño del modelo. Adicionalmente, el campo de las redes neuronales constantemente propone nuevos enfoques para procesar los datos y nuevas metodologías, que se pueden emplear para mejorar el rendimiento de los modelos presentados en este trabajo.

Otra área de oportunidad es extender lo aprendido por los modelos a otras clases de ligandos u otras familias de los GPCR mediante transferencia de aprendizaje. También se pueden explorar nuevos enfoques de ingeniería de características, buscando mejorar el rendimiento y la capacidad de explicar lo aprendido por el modelo.

En un futuro, este trabajo también nos puede ayudar, junto a otras herramientas, a proponer nuevas moléculas que logren generar un cambio en el estado funcional del GPCR al que se unan. Mediante dinámica molecular se pueden estudiar los efectos de nuevos ligandos y con los algoritmos presentados en este trabajo se podría clasificar el estado funcional final de la estructura del GPCR obtenida mediante dinámica

molecular.

Bibliografía

[1] Oludare Isaac Abiodun et al. «Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition». En: *IEEE Access* 7 (2019), págs. 158820-158846. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2945545.

[2] Laith Alzubaidi et al. *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. Vol. 8. 1. Springer International Publishing, 2021. ISBN: 4053702100444. DOI: 10.1186/s40537-021-00444-8. URL: https://doi.org/10.1186/s40537-021-00444-8.

[3] Weinstein H. Ballesteros JA. «Integrated methods for the construction of three-dimensional models and computational probing of structure-function relations in G protein-coupled receptors. Methods». En: *Methods in neurosciences* 25 (1995), págs. 366-428. ISSN: 00316997.

[4] Stefano Costanzi. «Modeling G protein-coupled receptors and their interactions with ligands». En: *Current Opinion in Structural Biology* 23.2 (abr. de 2013), págs. 185-190. ISSN: 0959440X. DOI: 10.1016/j.sbi.2013.01.008. URL: http://dx.doi.org/10.1016/j.sbi.2013.01.008%20https://linkinghub.elsevier.com/retrieve/pii/S0959440X13000249.

[5] Stefano Costanzi et al. «Rhodopsin and the Others: A Historical Perspective on Structural Studies of G Protein-Coupled Receptors». En: *Current Pharmaceutical Design* 15.35 (2009), págs. 3994-4002. ISSN: 13816128. DOI: 10.2174/138161209789824795.

[6] Sijia S. Dong, William A. Goddard y Ravinder Abrol. *Identifying multiple active conformations in the G protein-coupled receptor activation landscape using computational methods*. 2.ª ed. Vol. 142. Elsevier Inc., 2017, págs. 173-186. DOI: 10.1016/bs.mcb.2017.07.009. URL: http://dx.doi.org/10.1016/bs.mcb.2017.07.009.

[7] Correspondence D E Gloriam et al. «GPCRdb: the G protein-coupled receptor database-an introduction». En: *British Journal of Pharmacology* 173 (2016), pág. 2195. DOI: 10.1111/bph.13509.

[8] Eshan Ghosh et al. «Methodological advances: The unsung heroes of the GPCR structural revolution». En: *Nature Reviews Molecular Cell Biology* 16.2 (2015), págs. 69-81. ISSN: 14710080. DOI: 10.1038/nrm3933.

[9] Joseph Goncalves et al. «Magic Angle Spinning Nuclear Magnetic Resonance Spectroscopy of G Protein-Coupled Receptors». En: *Methods in Enzymology* 522 (ene. de 2013), págs. 365-389. ISSN: 0076-6879. DOI: 10.1016/B978-0-12-407865-9.00017-0.

[10] Robert M Gray. *Entropy and information theory*. Second edi. Sprringer, 2011. ISBN: 9781441979704.

[11] Reinhard Grisshammer. «New approaches towards the understanding of integral membrane proteins: A structural perspective on G protein-coupled receptors». En: *Protein Science* 26.8 (2017), págs. 1493-1504. ISSN: 1469896X. DOI: 10.1002/pro.3200.

[12] Enzo Grossi y Massimo Buscema. «Introduction to artificial neural networks». En: *European Journal of Gastroenterology and Hepatology* 19.12 (2007), págs. 1046-1054. ISSN: 0954691X. DOI: 10.1097/MEG.0b013e3282f198a0.

[13] Alexander S. Hauser et al. «GPCR activation mechanisms across classes and macro/microscales». En: *Nature Structural and Molecular Biology* 28.11 (2021), págs. 879-888. ISSN: 15459985. DOI: 10.1038/s41594-021-00674-7. URL: https://doi.org/10.1038/s41594-021-00674-7.

[14] Alexander S. Hauser et al. «Trends in GPCR drug discovery: New agents, targets and indications». En: *Nature Reviews Drug Discovery* 16.12 (2017), págs. 829-842. ISSN: 14741784. DOI: 10.1038/nrd.2017.178. URL: http://dx.doi.org/10.1038/nrd.2017.178.

[15] Daniel Hilger, Matthieu Masureel y Brian K. Kobilka. «Structure and dynamics of GPCR signaling complexes». En: *Nature Structural and Molecular Biology* 25.1 (2018), págs. 4-12. ISSN: 15459985. DOI: 10.1038/s41594-017-0011-7. URL: http://dx.doi.org/10.1038/s41594-017-0011-7.

[16] C. Hoffmann et al. «Conformational changes in G-protein-coupled receptors - The quest for functionally selective conformations is open». En: *British Journal of Pharmacology* 153.SUPPL. 1 (2008), págs. 358-367. ISSN: 00071188. DOI: 10.1038/sj.bjp.0707615.

[17] Vignir Isberg et al. *Generic GPCR residue numbers - Aligning topology maps while minding the gaps*. 2015. DOI: 10.1016/j.tips.2014.11.001.

[18] Biswajit Jena et al. «Artificial intelligence-based hybrid deep learning models for image classification: The first narrative review». En: *Computers in Biology and Medicine* 137.August (oct. de 2021), pág. 104803. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2021.104803. URL: https://doi.org/10.1016/j.compbiomed.2021.104803%20https://linkinghub.elsevier.com/retrieve/pii/S0010482521005977.

[19] Vsevolod Katritch, Vadim Cherezov y Raymond C Stevens. «Structure-Function of the G Protein-Coupled Receptor Superfamily». En: (2012). DOI: 10.1146/annurev-pharmtox-032112-135923. URL: www.annualreviews.org.

[20] Thomas H Keller, Arkadius Pichota y Zheng Yin. «A practical view of 'druggability'». En: *Current Opinion in Chemical Biology* 10 (4 ago. de 2006), págs. 357-361. ISSN: 13675931. DOI: 10.1016/j.cbpa.2006.06.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S1367593106000834.

[21] Amit Kessel y Nir Ben-Tal. *Introduction to Proteins*. second edi. New York: Chapman y Hall/CRC, mar. de 2018. ISBN: 9781315113876. DOI: 10.1201/9781315113876. URL: https://www.taylorfrancis.com/books/9781498747189.

[22] Nikhil Ketkar y Jojo Moolayil. *Deep Learning with Python*. 2021. ISBN: 9781617294433. DOI: 10.1007/978-1-4842-5364-9.

[23] Brian K. Kobilka y Xavier Deupi. «Conformational complexity of G-protein-coupled receptors». En: *Trends in Pharmacological Sciences* 28.8 (2007), págs. 397-406. ISSN: 01656147. DOI: 10.1016/j.tips.2007.06.003.

[24] Florian Koensgen et al. «Unsupervised Classification of G-Protein Coupled Receptors and Their Conformational States Using IChem Intramolecular Interaction Patterns». En: *Journal of Chemical Information and Modeling* 59.9 (2019), págs. 3611-3618. ISSN: 15205142. DOI: 10.1021/acs.jcim.9b00054.

[25] Albert J. Kooistra et al. «GPCRdb in 2021: Integrating GPCR sequence, structure and function». En: *Nucleic Acids Research* 49.D1 (2021), págs. D335-D343. ISSN: 13624962. DOI: 10.1093/nar/gkaa1080.

[26] Navin Kumar Manaswi. *Deep Learning with Applications Using Python*. 2018. ISBN: 978-1-4842-3515-7. DOI: 10.1007/978-1-4842-3516-4.

[27] Adam C. Mater y Michelle L. Coote. «Deep Learning in Chemistry». En: *Journal of Chemical Information and Modeling* 59 (2019), págs. 2545-2559. ISSN: 15205142. DOI: 10.1021/acs.jcim.9b00266.

[28] Alexander McPherson y Jose A. Gavira. *Introduction to protein crystallization*. 2014. DOI: 10.1107/S2053230X13033141.

[29] Fabio Nelli y Fabio Nelli. *Deep Learning with TensorFlow*. 2018, págs. 349-407. ISBN: 9781484230954. DOI: 10.1007/978-1-4842-3913-1_9.

[30] Alessandro Nicoli et al. «Classification Model for the Second Extracellular Loop of Class A GPCRs». En: *Journal of Chemical Information and Modeling* 62.3 (2022), págs. 511-522. ISSN: 1549960X. DOI: 10.1021/acs.jcim.1c01056.

[31] Chigozie Nwankpa et al. «Activation Functions: Comparison of trends in Practice and Research for Deep Learning». En: (2018), págs. 1-20. arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378.

[32] Gáspár Pándy-Szekeres et al. «GPCRdb in 2018: adding GPCR structure models and ligands». En: *Nucleic Acids Research* 46 (D1 ene. de 2018), págs. D440-D446. ISSN: 0305-1048. DOI: 10.1093/NAR/GKX1109. URL: https://academic.oup.com/nar/article/46/D1/D440/4634004.

[33] Paul S.H. Park, David T. Lodowski y Krzysztof Palczewski. «Activation of G Protein–Coupled Receptors: Beyond Two-State Models and Tertiary Conformational Changes». En: *Annual Review of Pharmacology and Toxicology* 48.1 (feb. de 2008), págs. 107-141. ISSN: 0362-1642. DOI: 10.1146/annurev.pharmtox.48.113006.094630. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3624763/pdf/nihms412728.pdf%20https://www.annualreviews.org/doi/10.1146/annurev.pharmtox.48.113006.094630.

[34] Michael J. Robertson, Justin G. Meyerowitz y Georgios Skiniotis. «Drug discovery in the era of cryo-electron microscopy». En: *Trends in Biochemical Sciences* 47.2 (feb. de 2022), págs. 124-135. ISSN: 0968-0004. DOI: 10.1016/J.TIBS.2021.06.008.

[35] Daniel M. Rosenbaum, Søren G. F. Rasmussen y Brian K. Kobilka. «The structure and function of G-protein-coupled receptors». En: *Nature* 459.7245 (mayo de 2009), págs. 356-363. ISSN: 0028-0836. DOI: 10.1038/nature08144. URL: http://www.nature.com/articles/nature08144.

[36] Sebastian Ruder. «An overview of gradient descent optimization algorithms». En: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[37] Jack Scantlebury et al. «Data Set Augmentation Allows Deep Learning-Based Virtual Screening to Better Generalize to Unseen Target Classes and Highlight Important Binding Interactions». En: *Journal of Chemical Information and Modeling* 60.8 (ago. de 2020), págs. 3722-3730. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00263. URL: https://dx.doi.org/10.1021/acs.jcim.0c00263%20https://pubs.acs.org/doi/10.1021/acs.jcim.0c00263.

[38] Jürgen Schmidhuber. «Deep Learning in neural networks: An overview». En: *Neural Networks* 61 (2015), págs. 85-117. ISSN: 18792782. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[39] Jendrik Schöppe et al. «Universal platform for the generation of thermostabilized GPCRs that crystallize in LCP». En: *Nature Protocols* 17.3 (2022), págs. 698-726. ISSN: 17502799. DOI: 10.1038/s41596-021-00660-9.

[40] Ramprasaath R. Selvaraju et al. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization». En: *International Journal of Computer Vision* 128.2 (2020), págs. 336-359. ISSN: 15731405. DOI: 10.1007/s11263-019-01228-7. arXiv: 1610.02391.

[41] Andrew W. Senior et al. «Improved protein structure prediction using potentials from deep learning». En: *Nature* 577.7792 (2020), págs. 706-710. ISSN: 14764687. DOI: 10.1038/s41586-019-1923-7. URL: http://dx.doi.org/10.1038/s41586-019-1923-7.

[42] J Craig Venter et al. *The Sequence of the Human Genome Downloaded from.* Inf. téc. 2001. URL: www.sciencemag.org/cgi/content/full/291/.

[43] Daniel Wacker, Raymond C. Stevens y Bryan L. Roth. «How Ligands Illuminate GPCR Molecular Pharmacology». En: *Cell* 170.3 (2017), págs. 414-427. ISSN: 10974172. DOI: 10.1016/j.cell.2017.07.009. URL: http://dx.doi.org/10.1016/j.cell.2017.07.009.

[44] Xinming Wang et al. «Emerging roles for G-protein coupled receptors in development and activation of macrophages». En: *Frontiers in Immunology* 10.AUG (2019), págs. 1-14. ISSN: 16643224. DOI: 10.3389/fimmu.2019.02031.

[45] Gerard J.P. van Westen, Anna Gaulton y John P. Overington. «Chemical, Target, and Bioactive Properties of Allosteric Modulation». En: *PLoS Computational Biology* 10.4 (2014). ISSN: 15537358. DOI: 10.1371/journal.pcbi.1003559.

[46] Phillip Wibisono et al. «Neuronal GPCR NMUR-1 regulates distinct immune responses to different pathogens». En: *Cell Reports* 38.6 (2022), pág. 110321. ISSN: 22111247. DOI: 10.1016/j.celrep.2022.110321. URL: https://doi.org/10.1016/j.celrep.2022.110321.

[47] Bentley Wingert, Pemra Doruker e Ivet Bahar. «Activation and Speciation Mechanisms in Class A GPCRs». En: *Journal of Molecular Biology* 434.17 (2022), pág. 167690. ISSN: 10898638. DOI: 10.1016/j.jmb.2022.167690. URL: https://doi.org/10.1016/j.jmb.2022.167690.

[48] Jin Xiang et al. «Successful Strategies to Determine High-Resolution Structures of GPCRs». En: *Trends in Pharmacological Sciences* 37.12 (2016), págs. 1055-1069. ISSN: 18733735. DOI: 10.1016/j.tips.2016.09.009. URL: http://dx.doi.org/10.1016/j.tips.2016.09.009.

[49] Wenzhan Yang et al. «The Evolving Druggability and Developability Space: Chemically Modified New Modalities and Emerging Small Molecules». En: *The AAPS Journal* 22 (2 mar. de 2020), pág. 21. ISSN: 1550-7416. DOI: 10.1208/s12248-019-0402-2. URL: https://link.springer.com/10.1208/s12248-019-0402-2.

[50] Bolei Zhou et al. «Learning Deep Features for Discriminative Localization». En: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (2016), págs. 2921-2929. ISSN: 10636919. DOI: 10.1109/CVPR.2016.319. arXiv: 1512.04150.

[51] Qingtong Zhou et al. «Common activation mechanism of class a GPCRs». En: *eLife* 8 (2019), págs. 1-31. ISSN: 2050084X. DOI: 10.7554/eLife.50279.

[52] S Kevin Zhou et al. «A Review of Deep Learning in Medical Imaging: Imaging Traits, Technology Trends, Case Studies With Progress Highlights, and Future Promises». En: *Proceedings of the IEEE* 109.5 (mayo de 2021), págs. 820-838. ISSN: 0018-9219. DOI: 10.1109/JPROC.2021.3054390. URL: https://www.ieee.org/publications/rights/index.html%20https://ieeexplore.ieee.org/document/9363915/.

Capítulo 7

Apéndice

7.1. Actividades adicionales de los ligandos para GPCR

Anteriormente se creía que sólo existían las respuestas generadas por los ligandos agonistas y antagonistas, pero posteriormente, al momento de estudiar a las proteínas *in vivo*, se descubrió la existencia de ligandos que generan respuestas intermedias.[43] La actividad que un ligando tiene al interactuar con un GPCR se puede agrupar en diferentes clases, tales como agonista completo, agonista parcial, agonista inverso, antagonista, moduladores alostéricos positivos y negativos, etc.[15,33] Dependiendo del tipo de ligando, la respuesta del GPCR puede ser modulada para ser completamente o parcialmente activada o inactivada.

Las principales clasificaciones que actualmente se usan para los ligandos son:

1. **Agonista.** Estos ligandos inducen un estado activo sobre la proteína a la que se acoplan, con lo que aumentan al máximo la actividad de la proteína y por ende la intensidad de la señale asociadas al GPCR.

2. **Agonista parcial.** Son ligandos que por lo general inducen un estado que no es 100 % activo. En estos casos la actividad de las señales debidas a la proteína, aunque no llegan a un máximo, adquiere valores elevados y en algunas ocasiones particulares estos ligandos pueden inducir un estado completamente activo.

3. **Antagonistas.** En esta clasificación se encuentran los ligandos que bloquean la actividad de ligandos agonistas y desfavorecen la activación del receptor. Esto lo hacen al evitar que los ligandos agonistas se puedan unir al GPCR, al bloquear el sitio de unión ortoestérico (que significa sitio correcto en griego).

4. **Agonistas inversos.** Son ligandos que disminuyen la actividad constitutiva del GPCR, al estabilizar una conformación inactiva de la proteína.

Adicionalmente los GPCR pueden regularse alostéricamente por moléculas que se unen a un sitio distinto al sitio ortoestérico, el sitio ortoestérico corresponde a la región de unión entre ligandos y proteínas. Estos reguladores alostéricos modulan la actividad de los ligandos ortoestéricos y por ende afectan el estado funcional de la proteína[43], y se pueden clasificar en:

1. **Moduladores alostéricos negativos (MAN, o por sus siglas en inglés NAM).**

   Disminuyen la actividad de los ligandos ortoestéricos y también disminuyen la estabilidad relativa del estado activo basal.

2. **Moduladores alostéricos positivos (MAP, o por sus siglas en inglés PAM).**

   Aumentan la actividad de los ligandos ortoestéricos, además pueden activar al receptor en ausencia de agonistas ortoestéricos.

Estos ligandos se conocen como ligandos alostéricos, se unen en sitios diferentes al ortoestérico, y pueden ser tanto endógenos (dentro de la estructura formada por las siete hélices del GPCR), como exógenos (Fuera de la estructura de siete hélices del GPCR).

7.2. Ejemplo general de las operaciones en una red neuronal

A continuación, vamos a describir el proceso de una neurona, donde queremos clasificar imágenes de dos tipos de diagonales, diagonales con pendiente positiva y aquellas con pendiente negativa. Un ejemplo visual de las diagonales a clasificar corresponde con la **imagen 7.1**.

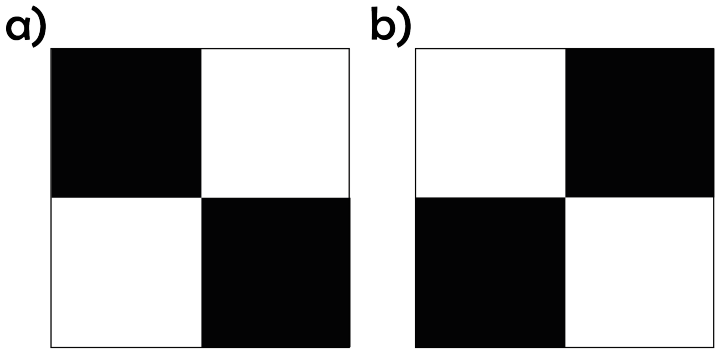


Figura 7.1: Imágenes de diagonales a clasificar.

Estás imágenes pueden representarse como matrices, donde un valor de 1 indica que el píxel está encendido, representado por el color blanco, mientras que un valor de 0 indica que está apagado, representado por el color negro. La representación en matriz y vector para la primera diagonal sería:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix},$$

mientras que para la segunda diagonal nos quedaría lo siguiente

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

En ambas representaciones se generaron sus correspondientes vectores al juntar los valores de cada renglón y generar la respectiva matriz transpuesta. Si se tuvieran solo tres datos a procesar por una neurona, tendríamos la siguiente ecuación:

$$W^T \times X = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0w_1 + 1w_2 + 1w_3 + 0w_4 & 1w_1 + 0w_2 + 0w_3 + 1w_4 & 1w_1 + 0w_2 + 0w_3 + 1w_4 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix},$$

donde el vector W^T representa los pesos para cada una de las entradas en la neurona y la matriz X representa los valores de entrada de la neurona. Si después le sumamos el vector de sesgos B, que corresponde con el vector que tiene almacenados los valores de b para cada neurona, llegaríamos a

$$z = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} z_\alpha & z_\beta & z_\gamma \end{bmatrix}.$$

Recordemos que cada columna representa un caso en el entrenamiento y por ende cada z representa un vector con dimensión de $l \times 1$. Después de aplicar la suma ponderada toca aplicar la función de activación respectiva a los valores obtenidos

$$\mathbf{y} = \sigma(z) = \begin{bmatrix} \sigma(z_\alpha) & \sigma(z_\beta) & \sigma(z_\gamma) \end{bmatrix} = \begin{bmatrix} y_\alpha & y_\beta & y_\gamma \end{bmatrix},$$

donde cada y es un vector con una dimensión $m \times 1$ y que representa los valores de predicción para cada entrada generados por la neurona.

7.3. No linealidad de la función ReLU

Partimos de los requisitos para las funciones lineales. Para una función $f : D \to S$, es lineal si se cumple

$$\forall_{x,y \in D}, f(x) + f(y) = f(x + y) \tag{7.1}$$

$$\forall_{k,x \in D}, f(kx) = kf(x) \tag{7.2}$$

Para la función ReLU,

$$f(x) = \frac{x + |x|}{2},$$

se cumple la ecuación 7.2

$$f(kx) = kf(x)$$

$$\frac{kx + |kx|}{2} = k\frac{x + |x|}{2}$$

pero no la ecuación 7.1,

$$\begin{aligned} f(x) + f(y) &= f(x+y) \\ f(1) + f(-1) &= f(1-1) \\ f(1) + f(-1) = \frac{1+|1|}{2} + \frac{-1+|-1|}{2} &= 1+0=1 \\ f(1-1) = \frac{(1-1)+|(1-1)|}{2} &= 0 \\ 1 &\neq 0 \end{aligned}$$

Por lo que ReLU no es lineal.

7.4. Entropía de Shannon

La entropía de Shannon se obtiene mediante la **ecuación 7.3**, donde la entropía para un mensaje $H(x)$, es el valor medio de la cantidad de información de los diversos estados del mensaje, el valor de $p(x_i)$ corresponde con la probabilidad de ocurrencia para cada evento.

$$H(X) = \sum_{i}^{n} p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) \qquad \text{si } x \text{ es una variable discreta} \tag{7.3}$$

en esta ecuación, X es el conjunto de datos y $p(x)$ es la probabilidad de cada elemento x_i en el conjunto de datos.

7.5. Producto interno de Frobenius

El producto interno de Frobenius es una operación entre dos matrices del mismo tamaño y que genera un escalar. La operación se define por

$$\langle A, B\rangle_F = \sum_{i,j} \overline{A_{i,j}} B_{i,j} \tag{7.4}$$

Donde $\overline{A_{i,j}}$ denota el complejo conjugado. Explícitamente, para una matriz de dimensión $n \times m$, la operación queda de la siguiente forma

$$\begin{aligned}\langle A, B\rangle_F = &\overline{A_{1,1}} B_{1,1} + \overline{A_{1,2}} B_{1,2} + \cdots + \overline{A_{1,m}} B_{1,m} \\ &+ \overline{A_{2,1}} B_{2,1} + \overline{A_{2,2}} B_{2,2} + \cdots + \overline{A_{2,m}} B_{2,m} \\ &\vdots \\ &+ \overline{A_{n,1}} B_{n,1} + \overline{A_{n,2}} B_{n,2} + \cdots + \overline{A_{n,m}} B_{n,m}\end{aligned}$$

7.6. Tipos de entrenamientos

Dependiendo de la cantidad y tipo de datos de los que disponemos podemos escoger entre tres opciones principales para los modelos de aprendizaje de máquina[29]:

1. **Entrenamiento supervisado**, es el tipo más común en redes neuronales. Este entrenamiento se realiza cuando contamos con un grupo de datos de entrenamiento previamente etiquetados y clasificados en las clases que nos interesa predecir. En este caso, el modelo aprende a predecir los valores de los parámetros (w, b, etc.) al tratar de disminuir su error al comparar sus predicciones con las clasificaciones de los datos de entrenamiento y así obtener los valores óptimos de los parámetros, con los cuales se logra la mejor tasa de acierto posible. El objetivo es que la red neuronal aprenda a producir las salidas deseadas para cada entrada.

2. **Entrenamiento no supervisado**, es el que se realiza cuando solamente tenemos los datos de entrenamiento sin clasificar, por

lo que no contamos con ningún tipo de etiqueta con la cual diferenciar a los datos. Este tipo de algoritmos buscara cómo separar a los datos provistos en un número de diferentes categorías, mediante la detección de patrones comunes en los ejemplos con los que el modelo trabaje. Cabe destacar que en estos casos se requieren grupos de datos muy grandes, del orden de decenas o cientos de miles de datos de entrenamiento.

3. **Entrenamiento por refuerzo**, en este caso se plantea un sistema acción-recompensa, en el que cada vez que el algoritmo acierte en su clasificación recibirá una señal que le indicará su acierto y optimizará su algoritmo para favorecer dichas señales de recompensa. En caso de que la predicción sea errónea, el algoritmo recibirá una señal que indique su error para que actualice sus parámetros al buscar disminuir las señales de error y aumentar el número de señales asociadas con recompensas.

En este trabajo, dado que se cuenta con una cantidad reducida de estructuras de GPCR previamente clasificadas, se optó por utilizar un modelo de entrenamiento supervisado.

7.7. Ejemplo del proceso de convolución

Para entender de una forma más clara el proceso de convolución podemos usar las imágenes del ejemplo de François Chollet presentado en el cuadernillo de Google Collaboratory https://colab.research.google.com/github/fchollet/deep-learning-with-python-notebooks/[22] (consultar cuadernillo 5.4 de la primera edición) y el cual se describe a continuación:

Partimos de una imagen de un gato con una resolución de 150×150,

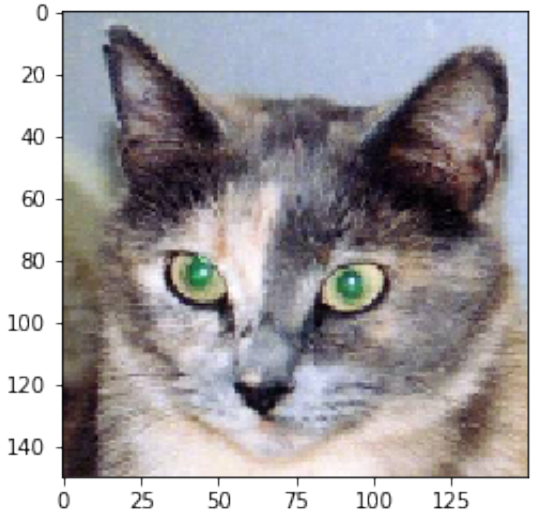


Figura 7.2: Imagen a procesar por una red neuronal convolucional.
Imagen de François Chollete

Aplicamos un filtro con 32 convoluciones en dos dimensiones, cuyos filtros son inicializados de forma aleatoria. Estas convoluciones aleatorias empezarán a detectar patrones y con el transcurso de las épocas de entrenamiento se optimizarán los filtros que generen las convoluciones que puedan destacar los patrones más significativos para lograr las clasificaciones deseadas. Al aplicar las convoluciones para una sola imagen se obtienen 32 nuevas imágenes, cada una de ellas es el resultado de una convolución (filtro) diferente, como se muestra en la **figura 7.3**.

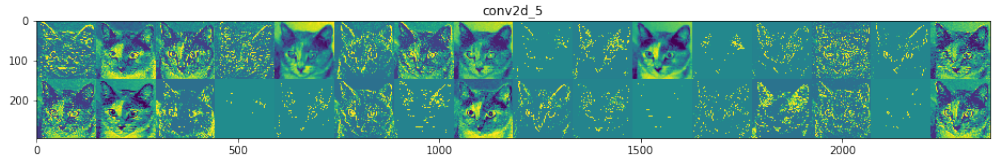


Figura 7.3: Imágenes obtenidas tras 32 convoluciones.Imagen de François Chollete

Después de cada convolución se suele aplicar un filtro que permite

abstraer más la información y reducir la dimensionalidad de las imágenes. En este ejemplo se aplica un filtro *max pooling* a las convoluciones obtenidas y obtendríamos las imágenes de la **figura 7.4**.

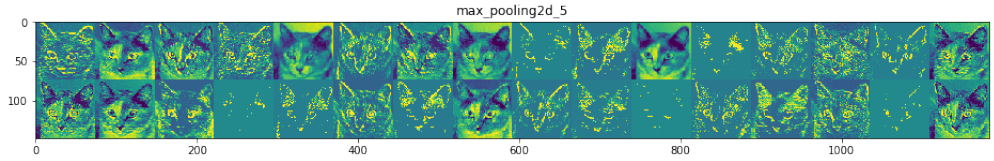


Figura 7.4: Imágenes obtenidas al aplicar un filtro 'max pooling'.
Imagen de François Chollete

Por lo general se aplican varias capas de convoluciones y subsecuentes filtros que disminuyan la dimensionalidad de las imágenes para detectar patrones abstractos. En este ejemplo se aplican 3 capas de convoluciones y filtros *max pooling*. En la segunda capa se aplican 2 convoluciones nuevas, con lo que se dobla el número de imágenes, que se presentan en la **figura 7.5**,

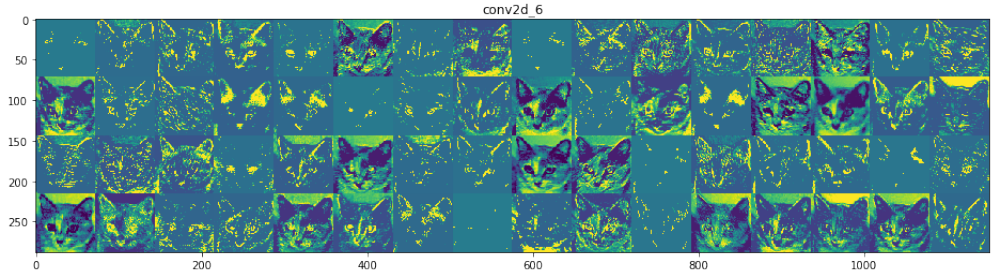


Figura 7.5: Imágenes obtenidas tras 2 nuevas convoluciones.
Imagen de François Chollete

Nuevamente se aplica un nuevo filtro *max pooling* a los datos, como se muestra en la **figura 7.6**,

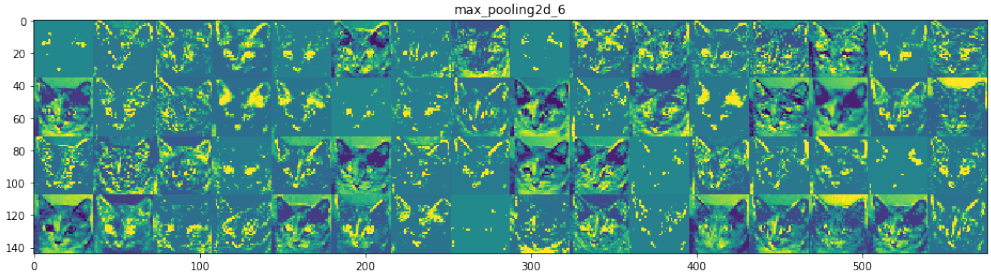


Figura 7.6: Imágenes obtenidas al aplicar un filtro 'max pooling'.
Imagen de François Chollete

Posteriormente se aplican dos nuevas convoluciones y en este punto podemos ver cómo los datos de las imágenes se empiezan a abstraer y cada vez se alejan más de la imagen de la que partimos, los resultados de la convolución se presentan en la **figura 7.7**,

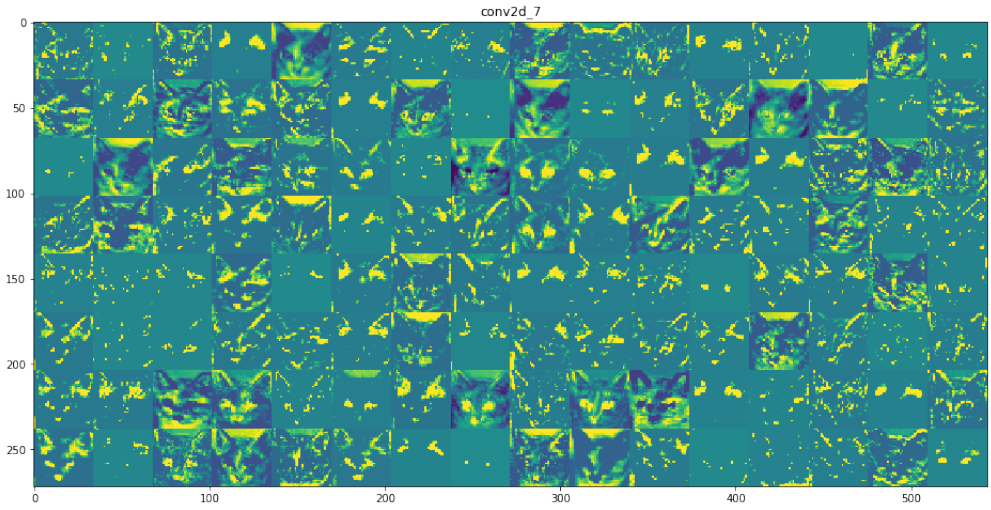


Figura 7.7: Imágenes obtenidas tras 2 nuevas convoluciones.
Imagen de François Chollete

Por último, aplicamos otro filtro *max pooling* y obtenemos las imágenes de la **figura 7.8**,

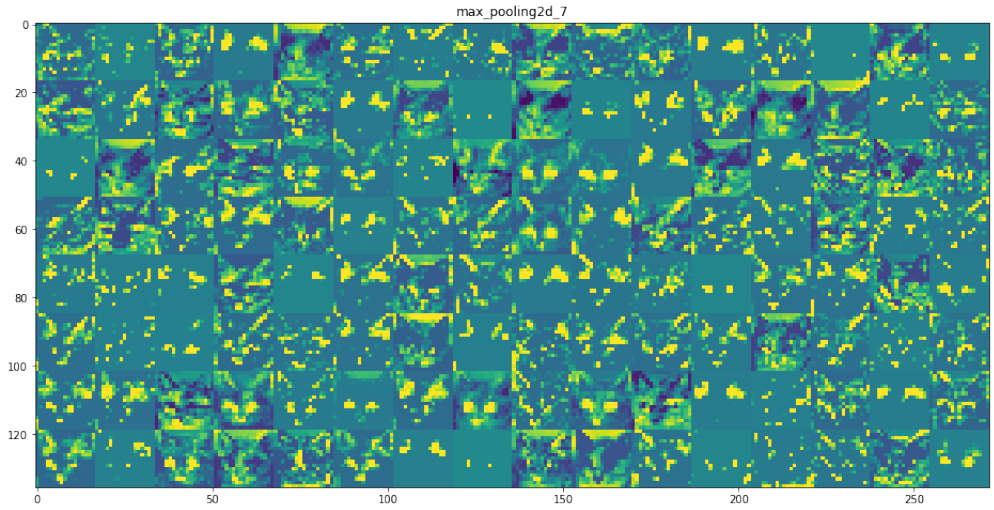


Figura 7.8: Imágenes obtenidas al aplicar un filtro 'max pooling'.
Imagen de François Chollete

Con este ejemplo podemos entender mejor cómo es que las convoluciones actúan para abstraer información de una matriz, que en este caso representa la imagen de un gato.

7.8. Algoritmo *CAM* y *Grad-CAM*

Los modelos CNN carecen de una explicación intuitiva del papel que juegan sus elementos individuales para en conjunto generar una predicción, lo que resulta en modelos difíciles de entender. Debido a esto no siempre es claro que elementos en los datos de entrada son relevantes para generar buenas o malas predicciones. Identificar dichos elementos es crucial para construir mejores y más confiables modelos CNN, que en caso de superar ampliamente al desempeño humano en alguna tarea nos pueden ayudar a identificar los patrones relevantes en los datos a los que debemos prestar atención.

Buscando resolver este problema Zhou *et al.*[50] construyeron un algoritmo llamado *CAM* (*Class Activation Mapping*) para identificar

aquellas secciones cruciales para la clasificación en modelos CNN. Una desventaja del algoritmo *CAM* es que sólo funciona en redes que no tienen capas completamente conectadas (*fully-conected layers*), con lo que se termina sacrificando poder predictivo por mejorar la interpretabilidad del modelo. Lo que resulta ser un problema común para los algoritmos que buscan mejorar la interpretabilidad de los modelos CNN. El algoritmo *CAM* se obtiene al aplicar una capa *global average pooling* en la salida del modelo CNN y multiplicar la matriz resultante por un mapa de activación de clases. Aplicar el *global average pooling* genera una matriz con el promedio espacial del mapa de características de la última capa de convolución en la red. Por último, el mapa de activación de clases es un vector con los pesos aprendidos por el modelo para clase considerada en la red. Esencialmente el peso de cada clase indica la importancia del mapa de activación para cada una de las clases consideradas. La función con la que se obtienen los mapas *CAM* es

$$M_c(x, y) = \sum_k w_k^c f_k(x, y) \tag{7.5}$$

Donde:
$M_c(x, y)$ es el mapa de activación para la clase c.
w_k^c es el peso correspondiente a la clase c para el elemento k.
$f_k(x, y)$ corresponde con el mapa de activación del elemento k en la locación espacial (x, y).

Por otro lado, Selvaraju *et al.* diseñaron el algoritmo *Grad-CAM* (*Gradient-weighted Class Activation Mapping*).[40] Que es una mejora de *CAM*, una de las principales implementarse en cualquier modelo basado en CNN. *Grad-CAM* aplica una operación de gradiente de los valores de las neuronas finales (y^c) para cada clase c antes de pasar por su respectiva función de activación con respecto al mapa de características (A^k) de la última capa de convolución para asignar la

relevancia de dichos mapas en cada clasificación. La matriz resultante de aplicar el gradiente posteriormente se pasa por una capa *global average pooling*, con lo que se obtienen los pesos de relevancia para cada neurona, de acuerdo con la siguiente expresión (α_k^c).

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{global\ average\ pooling} \frac{\partial y^c}{\partial A_{ij}^k} \tag{7.6}$$

Por último, se multiplica los valores α_k^c obtenidos por los respectivos mapas de características A^k en una combinación lineal a la que se le aplica la función $ReLU(x)$ para obtener los mapas *Grad-CAM* para cada clase c.

$$M_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right) \tag{7.7}$$