MODELO DE DEEP LEARNING PARA LA PREDICCIÓN DEL MAPA DE CONTACTO DE PROTEÍNAS

Darwin Yamid Ledesma Samboni
Adolfo Alexander Díaz Dávila

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR: ING. EMBER UBEIMAR MARTINEZ FLOR
CO-DIRECTOR: ING. NESTOR DIAZ MARIÑO

UNIVERSIDAD DEL CAUCA FACULTAD DE INGENIERÍA
FACULTAD INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS

GRUPO DE INVESTIGACIÓN EN INTELIGENCIA COMPUTACIONAL
LÍNEA DE INVESTIGACIÓN: INTELIGENCIA COMPUTACIONAL
POPAYÁN
SEPTIEMBRE DE 2022

Modelo de Deep learning para la predicción del mapa de contacto de proteínas

Darwin Yamid Ledesma Samboni
Adolfo Alexander Díaz Dávila

DIRECTOR: ING. EMBER UBEIMAR MARTINEZ FLOR
CO-DIRECTOR: ING. NESTOR DIAZ MARIÑO

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE SISTEMAS
GRUPO DE INVESTIGACIÓN EN INTELIGENCIA COMPUTACIONAL
LÍNEA DE INVESTIGACIÓN: INTELIGENCIA COMPUTACIONAL
POPAYÁN
Septiembre de 2022

Agradecimientos

El resultado de este proyecto está dedicado para todas aquellas personas que hicieron parte de nuestro proceso de aprendizaje, a través del cual obtuvimos todos los conocimientos necesarios para terminar esta importante etapa de nuestra vida, de igual manera a las personas que nos apoyaron en este camino, empezando por nuestros padres que siempre estuvieron ahí incondicionalmente, a nuestra familia que siempre nos apoyó y que sin ellos no habría sido posible haber llegado hasta este punto.

De igual forma agradecer al ingeniero Ember Ubeimar Martínez Flor quien nos supo guiar y brindar sus conocimientos y aportes para la conclusión de este proyecto.

TABLA DE CONTENIDO

INDICE DE FIGURAS

LISTA DE TABLAS

Presentación

El problema de predicción de contactos a través de secuencias de aminoácidos es un tema de investigación muy amplio y para el cual durante el transcurso de muchos años se han venido desarrollando distintos tipos de algoritmos con diferentes aportes desde puntos de vista y enfoques diferentes para darle solución o generar un aporte a este problema.

En este documento se muestra el proceso que se lleva a cabo para construir un modelo de Deep learning con el fin de aportar conocimiento de valor en la predicción del mapa de contacto de proteínas, mostrando el paso a paso desde la creación del dataset y del modelo de Deep learning, hasta su posterior implementación y entrenamiento generando predicciones y resultados que serán evaluados y comparados frente a otros algoritmos y trabajos existentes.

Este documento está organizado de la siguiente forma:

El capítulo 1 muestra la introducción del documento en ella se encuentra planteamiento del problema, los aportes, objetivos generales y específicos y los resultados obtenidos.

El capítulo 2 muestra las metodologías utilizadas a través del proyecto, que fases se definieron y cuáles son los resultados asociados.

En el capítulo 3 se muestra el proceso de investigación para la construcción del mapeo sistemático el cual es la base para el desarrollo del proyecto.

En el capítulo 4 se muestra cual fue todo el proceso para generar el dataset con el cual se realiza el entrenamiento del modelo propuesto, también podemos observar cual es el origen de los datos usados, como se hizo la limpieza de los mismos, que características fueron seleccionadas para realizar el entrenamiento del modelo, como y de qué forma se hace el procesamiento de los datos seleccionados y finalmente que estructura va a tener el dataset.

En el capítulo 5 nos enfocamos en el modelo implementado a lo largo de este proyecto, mostrando que estructura se definió y el porqué de esa estructura, mostrar cuales fueron las capas seleccionadas para componer este modelo, hacer una descripción del mismo, que cantidad de capas se usaron y que tipo de capas, y cuáles fueron los parámetros con los que se dieron mejores resultados.

El capítulo 6 describe la implementación del código con el que se realizó la generación del dataset, así como también la construcción del modelo de Deep learning, cual fue el lenguaje y tecnologías usadas en el desarrollo del proyecto, que librerías, frameworks fueron utilizados, como fue el proceso de aprendizaje, y la descripción por medio de diagramas de clases.

El capítulo 7 muestra cómo se realizó la evaluación del modelo propuesto, cuáles fueron los datos que se utilizaron para realizar dicha validación, haciendo su respectiva descripción, también mostrar que métricas fueron utilizadas y como fueron calculadas, y con esta información mostrar los resultados en comparativa con otros modelos propuestos en la literatura.

En el capítulo 8 finalmente se muestran cuáles fueron las conclusiones que dejó el trabajo, y cuales son todos esos pendientes que quedan para trabajar a futuro, en un tema investigativo que tiene aún mucho por explorar y mejorar.

CAPITULO 1

1. INTRODUCCION

1.1. PLANTEAMIENTO DEL PROBLEMA

La célula es la unidad fundamental de los organismos vivos, forma todos los organismos y los tejidos del cuerpo [1]. Las tres partes principales de la célula son la membrana celular, el núcleo y el citoplasma. La membrana celular rodea la célula y controla las sustancias que entran y salen, el citoplasma es la porción fluida del interior de la célula y el núcleo contiene el ADN celular [2]. El ADN es la molécula portadora de la información genética, contiene las instrucciones para todos los rasgos y funciones de un organismo y tiene la capacidad de copiar fielmente a sí mismo. A diferencia de las células eucariotas, las células procariotas no tienen núcleo y albergan su información genética dentro del citoplasma [3].

El ADN o material genético "debe poder replicarse para que las copias puedan transmitirse de célula a célula y de padres a hijos; debe contener información para dirigir las actividades celulares y guiar el desarrollo, funcionamiento y comportamiento de los organismos; además debe poder cambiar para que con el tiempo los grupos de organismos puedan adaptarse a diferentes circunstancias"[4]. El ADN se divide en unidades funcionales llamadas genes los cuales codifican las proteínas o ARN [1].

Los genes que codifican polipéptidos, que son las moléculas que forman las proteínas, se expresan en tres pasos: inicialmente se duplica la molécula de ADN (Replicación), posteriormente el ADN se transcribe en forma de ARNm (Transcripción) y este a su vez se traduce en lo que conocemos como proteína, por medio de la síntesis generada en el ribosoma (Traducción), este proceso se conoce como el dogma central de la biología molecular [5] [6].

A partir de la información o código genético se determina qué proteínas tiene una célula, un tejido y un organismo. Por consiguiente y de forma técnica "las proteínas son macromoléculas formadas por cadenas lineales de aminoácidos los cuales están organizados secuencialmente y esto está determinado por la secuencia de nucleótidos de su gen correspondiente llamados genes estructurales [7]. La mayoría de los organismos estudiados tiene proteínas formadas por combinaciones de 20 aminoácidos los cuales pueden modificarse en su proceso de síntesis [4].

Entre las moléculas orgánicas más abundantes en los seres vivos se encuentran las proteínas. Esas moléculas revisten particular interés porque, de su presencia y de la compleja interacción que se establece entre ellas y con otras sustancias a través del tiempo dependen muchas, sino todas, las características fenotípicas de los seres vivos, es decir su morfología y funcionamiento [7].

Las proteínas se presentan en gran variedad y se pueden encontrar miles de tipos diferentes que varían en tamaño desde proteínas relativamente pequeñas hasta proteínas enormes con pesos moleculares en millones [8]. Están compuestas básicamente por átomos de carbono, hidrógeno, oxígeno y nitrógeno que pueden además contener azufre y algunos otros tipos contienen también fósforo, hierro, magnesio y cobre entre otros elementos [9].

La función de las proteínas está ligada a la secuencia de aminoácidos, cada proteína tiene un número de residuos y secuencia característicos, entre algunas de las funciones

conocidas se encuentra la enzimática, hormonal (regula el metabolismo de la glucosa), reconocimiento de señales, transportadora (hemoglobina), defensiva (anticuerpos), estructural (colágeno), contráctil (Fibras musculares) [10] [11].

Por su complejidad, una proteína se puede describir de acuerdo con cuatro jerarquías conformacionales llamadas estructuras de las proteínas. La estructura primaria es la secuencia en la cual los aminoácidos están conectados mediante enlaces peptídicos, los cuales no son más que moléculas formadas por la unión de aminoácidos, dando como resultado así a una cadena polipeptídica [12]. La estructura primaria de una proteína dicta la forma en que se pliega en su estructura terciaria, es una conformación estable que es idéntica a la forma de otras moléculas de la misma proteína, es decir, su conformación nativa [9].

La siguiente jerarquía que describe las proteínas es la estructura secundaria, que se refiere al plegamiento que forman las cadenas peptídicas a partir de los enlaces de las moléculas de carbono y puentes débiles de Hidrógeno, lo que le permite plegarse de diferentes formas. Sin embargo, se distinguen esencialmente estas estructuras: Hélice α, hoja β y hélices triples [13]. Están formadas y estabilizadas principalmente por enlaces de hidrógeno [14].

La estructura terciaria es propiamente la estructura tridimensional y la biológicamente más activa, esta contiene varios péptidos, conocida como polipéptido complejo. En algunas proteínas, existe la estructura cuaternaria, que es la organización de dos o más cadenas de polipéptidos que en conjunto forman una proteína con muchas subunidades. “Al procedimiento mediante el cual una proteína alcanza su estructura tridimensional se le llama plegamiento” [12].

La función de una proteína se determina según su estructura tridimensional. Tener una estructura proteica proporciona un mayor nivel de comprensión de cómo funciona una proteína, lo que puede permitir, crear hipótesis sobre cómo afectarla, controlarla o modificarla. La determinación experimental de la estructura de la proteína requiere mucho tiempo y es cara; existe mucho interés en la predicción de estructuras como un proceso de selección de proteínas que no son sostenibles para la determinación experimental [15]. Si la estructura de una proteína es homóloga a una estructura ya conocida, entonces es posible usar esa estructura de la proteína como plantilla para modelarla. En muchas ocasiones se da el caso en que no existen plantillas adecuadas para muchas proteínas, por lo tanto, es necesario emplear métodos que usen secuencias de aminoácidos para predecir la estructura de la proteína [16].

En diversos trabajos de predicción de estructura de proteínas se utiliza el mapa de contacto como entrada del predictor. Un mapa de contacto es una representación bidimensional gráfica y simplificada de las interacciones de los residuos en una matriz, lo que permite realizar estudios de la estructura de las proteínas [17]. En los casos en los que no se cuenta con las coordenadas entre residuos de la proteína, se utiliza un predictor de mapa de contacto (técnica de aprendizaje de máquina) para predecir el mapa de contacto correspondiente a la proteína [18].

Los métodos de predicción actuales utilizados para construir mapas de contacto de proteínas se dividen en métodos de análisis de acoplamiento directo (DCA) y métodos de aprendizaje automático. Los métodos DCA utilizan alineamientos de secuencia múltiple (MSA) para determinar la correlación entre pares de residuos. Entre los métodos DCA más utilizados se encuentran CCMPred, PSICOV. Estos métodos son útiles para construir mapas de contacto de proteínas cuando hay disponible un gran número de homólogos de secuencia, su precisión es pobre cuando el número de homólogos es bajo [18].

Los métodos de aprendizaje automático pueden funcionar mejor que los métodos DCA cuando hay menos secuencias homólogas disponibles, ya que pueden aprender las relaciones de las características de la secuencia cuando se les da un conjunto de datos ordenado, cabe señalar que la mayoría de estos métodos utilizan uno o varios métodos DCA como entradas [18].

Estas se representan en mapas de distancia, mapas de contacto binarios y mapas de contacto difusos, donde cada uno de estos representan características diferentes. Esta investigación se centra en mapas de contacto de n clases, los cuales son una representación de un mapa de contacto difuso llevado a un nivel mucho más general, el cual contiene no sólo un umbral de distancia sino n umbrales y hasta m interpretaciones semánticas de contacto [19].

Teniendo en cuenta los elementos descritos anteriormente, se plantea la siguiente pregunta de investigación:

¿Qué tan eficientes son los modelos Deep Learning para la predicción del mapa de contacto de n clases de una proteína a partir de su secuencia de aminoácidos?

1.2 APORTES

Con los resultados de esta investigación se contribuyó al problema de predicción de la estructura de una proteína mediante un nuevo método para la predicción del mapa de contacto de n-clases de una proteína.

Se abrió una línea de investigación referente al impacto que un mapa de contacto de n-clases pueda tener en la mejora de la predicción de la estructura terciaria de la proteína.

Se exploró y evaluó la eficiencia de un modelo de Deep Learning en predicción de mapas de contacto de n clases.

1.3 OBJETIVOS

1.3.1. Objetivo General:

- Proponer un modelo de deep learning para la predicción del mapa de contacto de n clases de una proteína a partir de la secuencia de aminoácidos, estructura secundaria y características 2D propias de la proteína.

1.3.2. Objetivos específicos:

- Caracterizar los modelos de deep learning propuestos en la literatura para la predicción de mapas de contacto binarios con el objetivo de identificar posibles modelos que se puedan extender a n clases.
- Proponer un modelo de deep learning para la predicción del mapa de contacto de n clases.
- Validar y evaluar el modelo de deep learning propuesto a través de métricas (desempeño calidad) y comparar los resultados con el mejor modelo identificado en la revisión de la literatura.

1.4. RESULTADOS OBTENIDOS

Concluida la investigación se obtienen los siguientes entregables:

Monografía del trabajo de grado: Este documento proporciona en detalle, un resumen de los resultados más relevantes de esta investigación referentes al estado del arte y a la comparación de desempeños. Además, está constituido por las siguientes secciones:

- Introducción: que contiene el planteamiento del problema, aportes, objetivo general y específicos, así como resultados obtenidos.
- Metodología: Contiene las metodologías utilizadas para el desarrollo del proyecto, para la construcción de la revisión bibliográfica y para la construcción del modelo.
- Revisión bibliográfica: que contiene el estado del arte.

En las siguientes secciones se describen

- Construcción dataset de entrenamiento: que contiene el origen y limpieza de datos, además de la selección de características.
- Modelo de deep learning: contiene redes neuronales artificiales, deep learning, arquitectura resnet y la propuesta del modelo de deep learning.
- Implementación que contiene: Lenguaje de programación, librerías, entorno de desarrollo y sus limitaciones, productos generados, repositorio y características generales de la implementación.
- Evaluación que contiene: conjunto de datos, implementación, métricas, ajuste de parámetros, resultados y comparación de métodos casp.
- Conclusiones.
- Trabajo futuro.
- Finalmente se concluye el documento con las respectivas referencias.

Artículo: Con el objetivo de divulgar los resultados obtenidos en la investigación se hace entrega de un documento en formato pdf, con normas IEEE, en el que están plasmadas las técnicas y el proceso utilizado en esta investigación.

Implementaciones: Se comparten los repositorios que contienen el código fuente realizado en Python de la implementación y del modelo de deep learning utilizado para la predicción del mapa de contacto de proteínas. De esta manera, se tiene la posibilidad de reproducir las diferentes pruebas y resultados obtenidos.

CAPITULO 2

2. METODOLOGIA

2.1 Metodología para el desarrollo del proyecto

Para el desarrollo del proyecto se adaptó la metodología Design Patterns for Research Methods: Iterative Field Research [20], donde se proponen 4 etapas: observación, identificación, desarrollo y evaluación;

- El primer paso del patrón de la investigación iterativa son las observaciones de campo, estas se utilizan para identificar las necesidades dentro de la tarea elegida. A medida que un proyecto de investigación avanza por su ciclo de vida, las observaciones de campo cambiarán;
- El segundo paso en el patrón es usar las observaciones de campo para identificar el problema de investigación. Con el problema o pregunta de investigación específica identificada (ver Capitulo 3).
- El siguiente paso es hacer el trabajo de desarrollo. La naturaleza de este paso, por supuesto, depende de la aplicación y el tipo de investigación (Ver capítulo 4-5);
- El paso final en el patrón es la evaluación donde se evalúan cada uno de los productos o resultados obtenidos en cada una de las iteraciones (ver Capitulo 7).

2.2 Metodología para la construcción de la revisión bibliográfica

Para desarrollar el estado del arte se realizó una adaptación de la metodología Guidelines for conducting systematic mapping studies in software engineering: An update [21] donde se consideraron las siguientes fases:

- Preguntas de investigación
- Cadena de búsqueda
- Selección de bases de datos de consulta
- Selección y filtrado de artículos relevantes

De las cuales se obtuvieron los siguientes resultados asociados:

Se realizó un análisis de modelos para la predicción del mapa de contacto en proteínas a partir de su secuencia de aminoácidos y determinar qué trabajos existen en el estado del arte e identificar los avances y desarrollos tecnológicos que hasta la fecha han sido implementados, permitiendo identificar los vacíos y las necesidades en esta área que se puedan cubrir con trabajos futuros.

El análisis de los resultados se realiza categorizando los hallazgos y contando la frecuencia de publicaciones dentro de cada categoría para determinar la cobertura de las distintas áreas de un tema de investigación específico.

A partir de dicha clasificación se plantearon las siguientes preguntas para el desarrollo del estado del arte.

En la tabla 1 se establecen preguntas de investigación que se abordaron en el desarrollo del documento.

Tabla 1 Preguntas de Investigación

Número	**Pregunta de Investigación**
RQ1	¿Qué estudios primarios existen?
RQ2	¿Qué tipo de modelos de deep learning han sido utilizados para la predicción del mapa de contacto en proteínas?
RQ3	¿Cuál es la precisión o exactitud de los predictores?
RQ4	¿Qué Datasets o bases de datos han sido utilizados por los modelos de deep learning?

Fuente: Los autores.

En la Tabla 2 se muestra la cadena de búsqueda formada a partir de un conjunto de palabras clave utilizadas para encontrar respuesta a las preguntas de investigación.

Tabla 2 Cadena de búsqueda

Cadena de búsqueda (palabras clave)
(algorithm Or method Or Model) And (identification Or detection Or classification Or prediction) And (protein) And (deep learning)

Fuente: Los autores

Las bases de datos de consulta que se utilizaron para el mapeo sistemático fueron: SCIENCE DIRECT, SCOPUS, OXFORD ACADEMIC, SPRINGER LINK. El rango de tiempo establecido para este mapeo fue desde el año 2015 hasta el año 2020.

Posteriormente se hizo la aplicación de la cadena de búsqueda en cada una de las bases de datos de consulta con el fin de encontrar los principales y más relevantes artículos. Para determinar si un artículo era relevante se hizo un filtro, en el cual se tuvo en cuenta el rango de años de publicación (2015 - 2020), si el tipo de acceso del artículo era libre o abierto,

que fueran artículos de investigación que hayan sido publicados en revistas indexadas, publicaciones en inglés y español.

A continuación, se hizo un análisis y clasificación de los artículos más relevantes teniendo en cuenta el algoritmo o técnica usado, los resultados obtenidos, resumen, conjunto de datos, tipo de validación, limitaciones y medidas.

Por medio de esta metodología se llegó a tener los artículos más relevantes según nuestra consideración, obteniendo así el informe final del mapeo sistemático (ver Capitulo 3).

2.3 Metodología para la construcción del modelo

Para la fase de desarrollo / construcción del modelo de deep learning se utiliza la adaptación de la metodología CRISP-ML [22]. A continuación, observamos las fases utilizadas en la construcción del modelo propuesto

- Comprensión del negocio y datos:
  Inicialmente se identifican los datos que serán procesados para el entrenamiento del modelo (ver Capitulo 4.1).
- Ingeniería de datos (preparación de datos):
  Realizar la respectiva limpieza de los datos, hacer la selección de características, estandarización de datos (ver Capitulo 4.2 - 4.3).
- Modelado:
  Seleccionar la arquitectura que tendrá el modelo, definir número de capas, que métricas de evaluación serán usadas en el entrenamiento y en la validación, realizar entrenamiento y validación (ver Capitulo 5.4)
- Evaluación del modelo:
  Verificar rendimiento del modelo con un conjunto de datos de prueba, documentar y mostrar resultados de validación (ver Capitulo 7)
- Despliegue solución:
  Compartir el entorno de ejecución y sus especificaciones (ver Capitulo 6.3). Generar repositorio donde se albergue la implementación del proyecto (ver Capitulo 6.5).

CAPITULO 3

3. REVISION BIBLIOGRAFICA

3.1. ESTADO DEL ARTE

Luego de revisar la literatura relevante acerca del tema de investigación se han encontrado algunos estudios entre el año 2015 y el año 2020 relacionados con métodos, técnicas o algoritmos de aprendizaje profundo que han sido utilizados para la predicción de mapas de contacto, los cuales son importantes para el análisis de los resultados, ya que, aportan una base de conocimientos para profundizar en el tema de desarrollo.

El problema de predicción de contactos a través de secuencias de aminoácidos es un tema de investigación muy amplio y para el cual durante el transcurso de muchos años se han venido desarrollando muchos algoritmos con diferentes aportes desde puntos de vista distintos para darle solución a este problema.

RaptorX-Contact es uno de los predictores de contacto de última generación. Su alta precisión, confirmada por CASP, demuestra el beneficio de adquirir la proteína completa para usarla como contexto para construir un mapa de contacto. Aplica la estructura ResNet para la predicción que puede resolver los problemas de desaparición y explosión del gradiente debido a su identidad y características de mapeo residual, pero el número de parámetros es proporcional a su profundidad [23].

En [24] se propone un algoritmo basado en redes neuronales convolucionales profundas de dos niveles. En un primer nivel cuenta con 5 redes neuronales convolucionales que se encargan de predecir contactos con umbrales de distancia de 6, 7.5, 8, 8.5 y 10 Å (Angstroms) los cuales son utilizados para determinar la distancia mínima en la que se considera que dos residuos están o no en contacto. La última red neuronal convolucional toma el resultado de las anteriores como características para predecir el mapa de contacto final. Según los resultados del estudio obtenidos para un conjunto de longitud de contacto L/5 y con un conjunto de datos de modelo libre en CASP10, 11 y 12, este método obtuvo un 35%, 50% y 53,4% respectivamente, superando a métodos tales como MetaPSICOV con un 30.6% en el conjunto de datos CASP10, MetaPSICOV con un 34% en el conjunto de datos CASP11 y Raptor -X con un 46,3% en el conjunto de datos CASP12. Este método es una mejora de DNCON.

En [25] se analiza una metodología basada en diferentes enfoques mapReduce con el objetivo de equilibrar la distribución de clases mediante un sobre muestreo aleatorio, detectar las características más relevantes mediante un proceso evolutivo de ponderación de características y un umbral para elegirlas, construye un modelo random forest apropiado a partir de los datos preprocesados y finalmente clasificar los datos de prueba. Esta metodología ganó el desafío de big data ECBDL'14 para un problema de big data de bioinformática.

En [26], se hace uso de redes Generativas Antagónicas o Generative Adversarial Networks (GAN) para predecir mapas de contacto precisos de proteínas. GANCon presenta dos redes competitivas de aprendizaje contradictorio. La red del generador tiene como función la captura de información de contacto subyacente de las proteínas para generar mapas de contacto. La segunda es la red del discriminador la cual, aprende las diferencias de los mapas de contacto generados y los reales, además, esta alimenta la red del generador para

que este produzca mapas de contacto precisos. Por otra parte, se aborda el problema del desequilibrio y mantener la simetría de los mapas de contacto. Este se comparó con métodos de aprendizaje profundo de última generación, incluidos DNCON2, DeepContact, PconsC4 y DeepCov, y dos métodos ECA bien conocidos, incluidos CCMpred y FreeContac. Según este estudio GANCon fue quien obtuvo una mejor precisión para los contactos superiores de largo alcance L / 5, L / 2 y L de 89,93%, 80,84% y 65,87% respectivamente.

En [27] se hace la predicción de mapas de contacto de proteínas utilizando entrenamiento de redes neuronales junto con clasificadores de Bayes ingenuos. NeBcon, utiliza el teorema ingenuo del clasificador de Bayes (NBC) para combinar ocho métodos de contacto de última generación que se construyen a partir de enfoques de coevolución y aprendizaje automático. Las probabilidades posteriores del modelo NBC se combina con características estructurales intrínsecas de la secuencia de consulta y haciendo uso de una red neuronal se realiza la predicción del mapa de contacto final. NeBcon se probó en 98 proteínas no redundantes, lo que mejora la precisión del mejor predictor de metaservidor basado en la coevolución en un 22%; la magnitud de la mejora aumenta al 45% para los objetivos duros que carecen de secuencia y homólogos estructurales en las bases de datos. La precisión promedio de las predicciones de contacto de L / 5 fueron 0.651, 0.574 y 0.628.

En [28] realiza la predicción de contacto basada en covariaciones en un medio viable para el modelado 3D preciso de proteínas, sin más información que la secuencia requerida. Se combinan tres enfoques distintos para inferir señales de covariación de múltiples alineamientos de secuencia, se considera una amplia gama de otras características derivadas de la secuencia y, únicamente, una gama de métricas que describen tanto la calidad local como global de la alineación de secuencia múltiple de entrada. Se usa un predictor de dos etapas, donde la segunda etapa filtra la salida de la primera etapa. PSICOV muestra una precisión media sustancialmente mejor que el enfoque de información mutua normalizada de mejor rendimiento y las redes bayesianas. Para 118 de 150 objetivos, la precisión de L / 5 (es decir, predicciones de L / 5 superiores para una proteína de longitud L) para contactos de largo alcance (separación de secuencia> 23) fue ≥0,5, lo que representa una mejora suficiente para ser de beneficio significativo en la predicción de la estructura de la proteína o la evaluación de la calidad del modelo.

En [23] se presenta un nuevo método de aprendizaje profundo que predice contactos mediante la integración de información de conservación de secuencia y acoplamiento evolutivo (EC) a través de una red neuronal ultra profunda formada por dos redes neuronales residuales profundas. La primera red residual lleva a cabo una serie de transformaciones convolucionales unidimensionales de características secuenciales; la segunda red residual lleva a cabo una serie de transformaciones convolucionales bidimensionales de información por pares que incluye la salida de la primera red residual, información de EC y potencial por pares. Mediante el uso de redes residuales muy profundas, se puede modelar con precisión los patrones de ocurrencia de contacto y la relación compleja secuencia-estructura y, por lo tanto, obtener una predicción de contacto de mayor calidad independientemente de cuántos homólogos de secuencia estén disponibles para las proteínas en cuestión. Probado en 105 objetivos CASP11, 76 objetivos duros CAMEO anteriores y 398 proteínas de membrana, la precisión media de predicción de largo alcance L superior obtenida por este método, un método EC representativo CCMpred y el ganador de CASP11 MetaPSICOV es 0,47, 0,21 y 0,30, respectivamente; la precisión media de largo alcance L / 10 superior de este método, CCMpred y MetaPSICOV es 0,77, 0,47 y 0,59, respectivamente.

En [29] se utiliza un método inteligente para predecir los contactos residuo-residuo a nivel intra proteico. La columna vertebral del método de aprendizaje profundo es una red neuronal recurrente (RNN) con células de memoria a corto plazo (LSTM) de 5 capas. Se describe este modelo computacional para predecir los contactos residuo-residuo, se evalúa el método en tres conjuntos de datos de la cadena de proteínas y se informa el rendimiento predictivo en la obtención de precisiones de predicción. Además, también muestra los efectos de las características de los aminoácidos involucrados en la predicción de contactos residuo-residuo mediante el uso de tres métodos de aprendizaje automático sin supervisión. El rendimiento de este método entrenado en un pequeño conjunto de datos de secuencias de proteínas arroja luz sobre la utilidad potencial de aplicar una red neuronal recurrente en la predicción de contacto de residuos de residuos. Evaluando el método en tres conjuntos de datos de la cadena de proteínas e informando el rendimiento predictivo, se obtuvieron precisiones de predicción del 45,72%, 40,35% y 39,06% en un rango largo en el valor de corte L, respectivamente, lo que muestra una pequeña mejora.

En [30] se propone RawMSA para la predicción de novo basado en una técnica de lenguaje natural llamada incrustación, que consiste en convertir cada carácter de residuo en el MSA en un vector de punto flotante de tamaño variable. Se diseñaron varias redes neuronales profundas basadas en esta técnica para predecir SS, RSA y mapas de contacto de residuos-residuos (CMAP). Se muestran tres problemas de predicción diferentes: estructura secundaria, accesibilidad relativa al solvente y mapas de contacto entre residuos. Este método fue probado con un gran conjunto de proteínas y se determina que supera a métodos basados en matrices de puntuación específicas de posición (PSSM) ya que predice la estructura secundaria y la accesibilidad del solvente, mientras se desempeña a la par con métodos que utilizan más precalculados características en la categoría de predicción de mapas de contacto entre residuos en CASP12 y CASP13. Las predicciones de CMAP para el conjunto de referencia CASP12 RR se evaluaron de acuerdo con los criterios de CASP mediante el cálculo de la precisión de los contactos de largo alcance predichos L / 5 superiores, donde L es la longitud de la proteína, y los contactos de largo alcance son contactos entre residuos con una distancia de separación de secuencia superior a 23. La red CMAP final es un conjunto de 10 redes entrenadas en 10 a 1000 secuencias de entrada y un número variable de capas (10 a 24 capas convolucionales). Las predicciones CMAP para cada objetivo se han ordenado según la salida de la medida de probabilidad de contacto del conjunto, luego se han evaluado los contactos superiores L / 5 de largo alcance contra el mapa de contacto nativo. La precisión final se ha calculado como el promedio de las precisiones para todos los objetivos. Para hacer una comparación justa con los otros predictores, hemos descargado todas las predicciones hechas en CASP12 y las evaluamos con el mismo sistema.

En [31] se da a conocer la actualización de MemBrain que es un predictor de contacto entre hélices de dos etapas. La primera etapa toma características basadas en secuencia como entradas y salidas de probabilidades de contacto, que se introducirán en la red neuronal convolucional junto con las predicciones de tres enfoques de análisis de acoplamiento directo en la segunda etapa. Para la validación del método se utilizó una estricta Validación Cruzada Jackknife basada en secuencia, con la que para un conjunto de datos de entrenamiento se obtuvo una precisión media de 81,6% para las predicciones L / 5 principales, sin embargo, para el conjunto de datos de prueba se obtuvo una precisión media de 79,4%. Además, para los contactos de bucle L/5 se obtuvo una precisión del 56,4%.

En [32] se muestra un nuevo método de predicción de contacto basado en deep learning. MapPred, como fue nombrado, consta de dos métodos de componentes, DeepMSA y

DeepMeta, ambos entrenados con las redes neuronales residuales. DeepMeta funciona combinando contactos predichos y otras características de perfil de secuencia. Los experimentos en tres conjuntos de datos de referencia sugieren que la contribución de los datos de la secuencia del metagenoma es significativa. Aquí el objetivo es resolver el problema de realizar predicciones basadas en la disponibilidad de alineamientos profundos de secuencias múltiples (MSA). Sin embargo, muchas proteínas de las familias poco pobladas no tienen un número suficiente de homólogos en la base de datos UniProt convencional, por lo tanto, se busca esta alternativa mediante la cual se realiza la exploración de la rica secuencia de datos de los proyectos de secuenciación del metagenoma.

En [33] se usa un servidor web que predice el mapa de contacto de proteínas y la estructura 3D usando un nuevo método. CoinFold predice contactos mediante la integración de análisis de acoplamiento evolutivo (EC) multifamiliares conjuntos y aprendizaje automático supervisado. Este análisis de EC conjunto es único en el sentido de que no solo utiliza información de coevolución de residuos en la familia de proteínas objetivo, sino también en las familias relacionadas que pueden tener secuencias divergentes, pero pliegues similares. El aprendizaje supervisado mejora aún más la precisión de la predicción de contacto al hacer uso del perfil de secuencia, el potencial de contacto (distancia) y otra información. Finalmente, este servidor predice la estructura terciaria de una secuencia alimentando sus contactos predichos y la estructura secundaria a la suite CNS. Probado en los objetivos CASP y CAMEO, este servidor muestra ventajas significativas sobre los existentes de categoría similar en predicción de estructura terciaria y de contacto.

CAPITULO 4

4. CREACION DATASET DE ENTRENAMIENTO

4.1. Origen de los datos

Para la creación del dataset utilizado en el entrenamiento del modelo de deep learning, se realizó un proceso de búsqueda en el que se visitaron plataformas virtuales que ofrecieran el servicio y acceso a información sobre proteínas de las cuales se pudieran obtener los datos necesarios para realizar el proyecto, la plataforma seleccionada fue https://www.rcsb.org/ en la cual se encuentra el Protein Data Bank.

Protein Data Bank (PDB), es una base de datos de la estructura tridimensional de las proteínas y ácidos nucleicos. Estos datos, generalmente obtenidos mediante cristalografía de rayos X o resonancia magnética nuclear, son enviados por biólogos y bioquímicos de todo el mundo. Están bajo el dominio público y pueden ser usados libremente. El PDB está supervisado por una organización llamada Worldwide Protein Data Bank (wwPDB) [34].

Una vez seleccionada nuestra fuente de información, con todos los datos de proteínas sin procesar, se continuo con la búsqueda de una librería que permitiera acceder y manipular esta información, esto se logró en primera instancia por medio de Biopython, que es un conjunto de herramientas disponibles gratuitamente para el cálculo biológico, escrito en Python por un equipo internacional de desarrolladores [35], y se especializa en la manipulación y obtención de datos de proteínas directamente descargadas desde el PDB (Protein Data Bank) anteriormente nombrado.

Luego de ser instalada la librería y de conocer sus métodos se realiza la descarga de las proteínas con las que se van a trabajar, por medio de una lista de id's de las proteínas, que son un código de 4 dígitos alfanuméricos que identifican a cada proteína y se especifica la ruta en donde se desea realizar el guardado, en la página se encontró un archivo en formato txt que contiene todos los id's de las proteínas existentes en el PDB (Protein Data Bank) la cual llega de momento a una cantidad de 192489 proteínas hasta el momento en que se realiza este proyecto, del total de proteínas se seleccionaron como muestra 1000 id's correspondientes a 1000 proteínas para generar el dataset.

4.2. Limpieza de los datos

En el transcurso de la descarga de las proteínas desde el PDB (Protein Data Bank), se realiza una depuración, de aquellas proteínas que presentan problemas generando: primero que no se pueda realizar directamente la descarga y segundo que ocasionen la detención de la ejecución del programa, por lo cual, se utilizaron los códigos identificadores de las proteínas más actualizados y que presentaran la menor cantidad de errores.

Una vez descargados los archivos de las proteínas, para generar el dataset se tuvo en cuenta únicamente proteínas cuya secuencia de aminoácidos no superara en tamaño los 1000 nucleótidos, esto con el fin de mejorar el proceso de creación en cuanto a tiempos y rendimiento de espacio en disco y uso de memoria RAM.

4.3 Selección de características

Después de realizar la descarga, se empezaron a seleccionar aquellas características de las proteínas que se tendrían en cuenta para armar el dataset de entrenamiento del modelo de deep learning, en total se seleccionaron 7 características, se debe tener en cuenta que todas las características se calculan a partir de la secuencia de aminoácidos perteneciente a cada proteína, la secuencia de aminoácidos está representada por una cadena de caracteres unidos secuencialmente en la que cada carácter representa un aminoácido en específico y que esta secuencia a su vez se divide en ventanas por medio de un proceso de ventaneo y combinación, del cual resultan porciones o subdivisiones de la secuencia de aminoácidos original a la cual llamamos ventanas deslizantes (Figura 1) y poseen un tamaño fijo de 19 aminoácidos representados en caracteres, estas ventanas surgen como todas las posibles combinaciones ordenadas de tamaño 19 de cada secuencia.

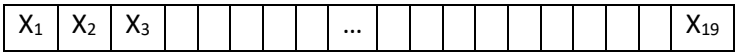

Figura 1. Ventana deslizante

Donde X_i es un aminoácido representado por un carácter alfabético.

A continuación, mostramos los 21 caracteres con sus respetivas equivalencias:

- D: Acido aspártico
- E: Glutámico
- R: Arginina
- K: Lisina
- N: Asparagina
- H: Histidina
- Q: Glutamina
- S: Serina
- T: Treonina
- A: Alanina
- G: Glicina
- V: Valina
- P: Prolina
- L: Leucina
- F: Fenilalanina
- Y: Tirosina
- I: Isoleucina
- M: Metionina
- W: Triptófano
- C: Cisteína
- -: Desconocido

Teniendo en cuenta lo anterior describimos a continuación las características seleccionadas:

- **Matrices de distancia de ventana deslizante**: Se genera una lista de 7 matrices de distancia por cada ventana deslizante, de 21 filas por 21 columnas cada matriz, donde cada fila y columna representan un aminoácido en particular, y el cruce entre ellos representan la relación de distancia entre si. Siendo así, para la primera matriz de distancia uno, se pone un uno entre el cruce de dos aminoácidos en el que su distancia dentro de esa ventana equivalga a uno, si ese cruce se repite de nuevo dentro de otro punto de la ventana el valor se irá acumulando, para la matriz dos se realiza el mismo análisis, solo que en este caso la distancia que se tendrá en cuenta entre uno y otro aminoácido será de dos, y así sucesivamente con cada matriz hasta llegar a la distancia 7.

  Ejemplo:

  Sea la ventana deslizante V1:

D	C	D	W	Y	A	S	R	T	T	T	L	F	Y	I	E	H	Q	A

  Calculando la primera matriz de distancia 1, para ello se codifican los caracteres asociando un valor numérico a cada uno de los aminoácidos y así poder trabajar más fácilmente las matrices para su posterior llenado.

  Codificando:

  D:0, E:1, R:2, K:3, N:4, H:5, Q:6, S:7, T:8, A:9, G:10, V:11, P:12, L:13, F:14, Y:15

  I:16, M:17, W:18, C:19, -:20

  Reemplazando en V1:

0	19	0	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

  Y se realiza el proceso de llenado de la matriz para distancias de tamaño 1 de la siguiente manera:

0	19	0	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

  Comenzando con el primer elemento de la ventana, este será seleccionado como el elemento a evaluar en la primera iteración, se observa cual es el aminoácido que se encuentra en este caso a distancia 1 de él, es cual es elemento 19 de la segunda casilla

0	**19**	0	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

  Teniendo ya la pareja de aminoácidos con distancia 1, se llena la matriz (Figura 2) usando los valores numéricos de cada aminoácido codificado como coordenadas

en la matriz, el primer elemento como numero de fila y el segundo elemento indicando la columna (0, 19), sumando 1 al valor que se encuentra previamente en esa casilla, por defecto empiezan en 0.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2. Matriz de distancia de ventana deslizante

Una vez puesto el valor, se continua el llenado de la matriz, el aminoácido codificado a evaluar sigue siendo 0 (cero) durante toda la primera iteración, por tanto, se busca en la ventana el siguiente 0, una vez identificado, si es que se encuentra, se repite el mismo proceso, seleccionando el valor que está a su derecha con una distancia equivalente a 1, para este caso es el valor 18 y se suma en la matriz en sus respectivas coordenadas (0,18) un 1 al valor que se encuentra previamente.

0	19	**0**	**18**	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

La matriz de distancia 1 (Figura 3) quedaría de la siguiente manera:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3. Matriz de distancia de ventana deslizante

Al finalizar la primera iteración se observa que no hay más ceros, por lo tanto, continua con el siguiente aminoácido codificado a evaluar, repitiendo los mismos pasos que con el anterior.

0	**19**	0	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

Una vez seleccionado el nuevo aminoácido a evaluar se obtiene el que se encuentra a su derecha a distancia 1.

0	**19**	**0**	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

Graficando (Figura 4):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 4. Matriz de distancia de ventana deslizante

Para el aminoácido codificado 19 no hay más parejas con distancia 1 en lo que resta de la ventana deslizante, por tanto, termina la iteración.

El siguiente aminoácido a evaluar es de nuevo el cero:

0	19	**0**	18	15	9	7	2	8	8	8	13	14	15	16	1	5	6	9

Como este aminoácido ya se evaluó una vez, simplemente se salta y se continua con el siguiente. Para el resto de aminoácidos se repite el proceso anteriormente mostrado, hasta llegar al último aminoácido, que por ser el último no posee un vecino a su derecha, por lo tanto, termina la iteración de toda la ventana y el llenado de la matriz distancia 1.

Una vez entendido el proceso de llenado de matriz de distancia 1, se generan las otras 6 matrices, para un total de 7, siguiendo los mismos principios y pasos, la única diferencia es que, para sumar un valor en la matriz correspondiente, las distancias que se tendrán en cuenta serán respectivamente de 2, 3…hasta 7, generando de esta manera las 7 matrices con sus distancias asociadas.

- **Matrices de distancia ventana deslizante posterior**: Esta es la segunda característica seleccionada para la creación del dataset de entrenamiento del modelo de deep learning para la predicción del mapa de contacto de proteínas, el proceso y la creación de estas 7 matrices es exactamente igual al expuesto anteriormente en la primera característica seleccionada, se generan 7 matrices de distancia por ventana deslizante, la diferencia es que estas son generadas para ventanas posteriores a las generadas en la primera característica.

- **Distancia:** Esta característica es un numero entero calculado a partir de la resta entre las posiciones de p y q, donde p y q representan los aminoácidos a evaluar entre dos ventanas deslizantes, cada uno de estos aminoácidos se encuentra en la parte central de cada ventana.

  Ejemplo:

  Dada la siguiente secuencia de aminoácidos S

A	D	E	E	G	M	M	K	R	C	W	I	T	S	H	E	R	F	Y	I	E	A	A	A	R	W	A	M	K	K	T	T
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

  Obtenemos dos ventanas deslizantes, V1 y V2 de S y ubicamos el aminoácido a evaluar en el centro de cada ventana respectivamente:

  V1

									p								
A	D	E	E	G	M	M	K	R	**C**	W	I	T	S	H	E	R	F	Y
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

  V2

									q								
S	H	E	R	F	Y	I	E	A	**A**	A	R	W	A	M	K	K	T	T
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

  Una vez obtenidas las posiciones de cada aminoácido en su respectiva ventana, procedemos a hacer la operación de resta entre los índices en (1), primero colocando la posición del aminoácido a evaluar de la ventana V2, y le restamos el índice que indica la posición del aminoácido a evaluar de la ventana V1 así:

$$Distancia = p - q \quad (1)$$

$$Distancia = 23 - 10$$

$$Distancia = 13$$

Por medio de este proceso se guarda como una característica más dentro de cada registro, generado a partir de las ventanas evaluadas para la creación del dataset.

- **Matriz de puntuación específica de la posición (PSSM):** Es una matriz que contiene la información posicional de aminoácidos o nucleótidos que representa la relación funcional existente de un conjunto de aminoácidos alineados previamente. La matriz contiene una probabilidad para cada aminoácido en su posición de la alineación [36] [37].

  Para generar una matriz de distancia es necesario tener en cuenta la matriz de conteo que consiste en contar las ocurrencias de cada aminoácido en su posición específica y una matriz de probabilidad que representa la frecuencia de existencia de un aminoácido en una posición especifica. Por ejemplo:

  Sean X_i las secuencias a alinear.

$$x_1 = TTATA$$
$$x_2 = TCATT$$
$$x_3 = ATAGC$$

  Donde,

$$\delta_{a,b} = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases}$$

  Entonces definimos C como la matriz de conteo (2), donde las columnas representan la posición de cada uno de los aminoácido o nucleótidos de las secuencias alineadas y las filas indican la cantidad de aminoácidos existentes.

$$C_{b,i} = \sum_{j=1}^{N} \delta_{b,x_j[i]} \quad (2)$$

$$C_{b,x[i]} = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 1 & 0 & 3 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 2 & 2 & 0 & 2 & 1 \end{bmatrix}$$

  Ahora se calcula la matriz de probabilidad (3), que consiste en calcular la probabilidad de que un aminoácido existe en una posición, para esto se divide la numero de elementos existentes en una posición sobre el número de secuencias alineadas.

$$f_{b,i} = \frac{C_{b,xj[i]}}{\sum_{b'=A}^{T} C_{b',i}} \quad (3)$$

$$P_{b,x[i]} = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0.5 & 0.0 & 1{,}5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\ 1.0 & 1.0 & 0.0 & 1.0 & 0.5 \end{bmatrix}$$

Finalmente se calcula la matriz la matriz PSSM (4)

$$S(X) = \sum_{i=1}^{k} \log\left(\frac{f_{x[i],i}}{P_{x[i]}}\right) \quad (4)$$

$$S(X) = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0.42 & 0.00 & 2.00 & 0.00 & 0.42 \\ 0.00 & 0.42 & 0.00 & 0.00 & 0.42 \\ 0.00 & 0.00 & 0.00 & 0.42 & 0.00 \\ 1.42 & 1.42 & 0.00 & 1.42 & 0.42 \end{bmatrix}$$

La matriz Pssm se utilizó, como una de las entradas principales para entrenamiento del modelo, y donde se alinearon las ventanas deslizantes de las cuales siguiendo el mismo proceso se obtuvo la matriz correspondiente a dicha alineación. Este proceso se realizó gracias a los métodos que proporciona BioPython.

- **Escalas VHSE** (Vectores de puntuación de componentes principales de propiedades hidrofóbicas, estéricas y electrónicas): se derivan del análisis de componentes principales (PCA) en familias independientes de 18 propiedades hidrofóbicas, 17 propiedades estéricas y 15 propiedades electrónicas, respectivamente, que se incluyen en total 50 variables fisicoquímicas de 20 aminoácidos codificados.

  El promedio calculado de las escalas VHSE de todos los aminoácidos en la secuencia peptídica correspondiente. Cada escala VSHE representa una propiedad de aminoácido de la siguiente manera:

  VHSE1 y VHSE2: propiedades hidrofóbicas
  VHSE3 y VHSE4: propiedades estéricas
  VHSE5 a VHSE8: propiedades electrónicas [38].

  Para el cálculo de las propiedades VHSE se hizo uso del paquete de R llamado peptides y se uso el método vhseScales que recibe una secuencia de aminoácidos como entrada y retorna un vector de 8 posiciones como salida con la información físico química asociada a esa secuencia en particular.

  Ejemplo:

  Sea seq la secuencia de aminoácidos:

  seq= **'QWGRRCCGWGPGRRYCVRW'**

  Calculamos las propiedades VHSE por medio del método de R vhseScales

Así,

vhseScales (seq = "QWGRRCCGWGPGRRYCVRWC")

el cual nos genera el siguiente vector de características

VHSE1 VHSE2 VHSE3 VHSE4 VHSE5 VHSE6 VHSE7 VHSE8
-0.1150 0.0630 -0.0055 0.7955 0.4355 0.2485 0.1740 -0.0960

- **Contacto:** Se define como el espacio entre dos residuos o aminoácidos que se encuentran a una distancia cercana medida en angstroms, un par de aminoácidos están en contacto si la distancia entre sus átomos específicos (principalmente carbono-alfa o carbono-beta) es menor que un umbral de distancia [39], para este caso se subdividieron las distancias en 4 rangos generando 4 clases, 3 de contactos y una de no contacto, estas 4 características funcionan como salidas deseadas del modelo propuesto, cada contacto está asociado a una única clase según su distancia, al estar asociado a esa clase el valor de la misma será de 1, y por ende las otras clases tendrán valor de 0, solo podrá pertenecer a una clase cada contacto o no contacto.

- **Clase 1:** Esta clase se define como todo aquel contacto que se encuentra en un umbral de distancia entre los valores mayores a 0 y los valores menores o iguales a 8 angstrom (0,8].

- **Clase 2:** Esta clase se define como todo aquel contacto que se encuentra en un umbral de distancia entre los valores mayores a 8 y los valores menores o iguales a 10 angstrom (8,10].

- **Clase 3:** Esta clase se define como todo aquel contacto que se encuentra en un umbral de distancia entre los valores mayores a 10 y los valores menores o iguales a 15 angstrom (10,15].

- **Clase 4:** Esta clase se define como NO contacto para todos aquellos valores mayores a 15 angstrom (15, ∞).

Formalmente, la matriz de distancias (5) para una proteína p de n residuos, se calcula como:

$$Dp = \{dp(i,j) : i,j = 1, \dots, np\} \quad (5)$$

donde cada posición de la matriz dp(i,j) es la distancia euclidiana entre los carbonos alfa (Cα-Cα) entre cada par de residuos i y j con base en sus coordenadas atómicas [40].

Finalmente se empiezan a generar registros los cuales se van guardando y acumulando hasta completar el dataset, cada registro está formado por las 7 características seleccionadas: matrices de distancia de ventana deslizante, matrices de distancia de ventana deslizante posterior, distancia, vector de propiedades VHSE de ventana deslizante, vector de propiedades VHSE de ventana deslizante posterior, matriz PSSM, clase.

Para guardar todos estos registros se hizo uso de tensor flow, tensor flow es una librería de código abierto utilizada para proyectos de aprendizaje automático, cuenta con herramientas y recursos que permite desarrollar e implementar aplicaciones con esta tecnología de manera fácil y eficiente [41].

Para la cantidad de proteínas procesadas, 1000 en total, se generaron 3'.571.378 registros que en conjunto forman el dataset utilizado para entrenar y validar el modelo propuesto.

CAPITULO 5

5. MODELO DE DEEP LEARNING

Los algoritmos de aprendizaje profundo han tenido una participación muy importante en la investigación y estudio de las proteínas gracias a la capacidad de identificar patrones y características.

5.1. REDES NEURONALES ARTIFICIALES

Las redes neuronales son un conjunto de neuronas conectadas entre sí, donde cada una de estas recibe una entrada, procesa la información y proporciona una salida que a su vez será la entrada de otra neurona generando así una red de miles o millones de neuronas que se caracteriza por tener robustez y alta capacidad de aprendizaje. Cada neurona se conecta con las demás a través de enlaces. El objetivo principal de las redes neuronales artificiales consiste en la resolución de problemas a través del aprendizaje, la generalización y procesamiento automático de datos [42].

5.2. DEEP LEARNING

Deep Learning es un tipo de aprendizaje automático que permiten realizar análisis a grandes volúmenes de datos y proporcionar una precisión alta para las predicciones. Estas redes neuronales intentar parecerse a el cerebro humano con la diferencia de que se enfocan en aprender una tarea específica [43].

5.3. ARQUITECTURA RESNET

Este tipo de arquitectura surge a partir de la necesidad de que las redes neuronales primarias se volvieron cada vez más profundas ya que pretendían resolver problemas de mayor complejidad, provocando que cada vez fuese más difícil realizar el entrenamiento, esto se conoce como problema del gradiente de desaparición, donde la información de valor al pasar por cada capa tiende a reducirse lo que implica una reducción en la precisión.

En 2015 los investigadores propusieron el modelo ResNet (Figura 5), una arquitectura que utiliza bloques residuales y que tuvo un avance significativo para el problema del gradiente.

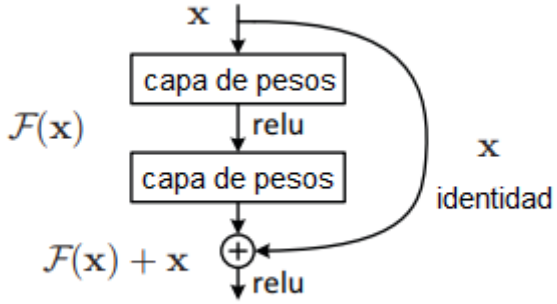

Figura 5. Bloque Resnet

Generalmente los bloques residuales están compuestos por una conexión que hace un salto de varias capas del modelo, esto es la parte más importante del bloque, sin este salto, la entrada se multiplicaría por los pesos y tendería a caer en error. El salto permite concatenar a X la entrada original y mantener una entrada constante. Posteriormente se agregar una función de activación f () y el bloque genera la salida H(x) (6).

$$H(x) = f(\,wx\, +\, b\,) \quad (6)$$

Con la nueva adición de salto, a la salida se le adiciona X (7).

$$H(x) = f(x) + x \quad (7)$$

Existen dos tipos de bloques residuales que son: bloque residual identidad, el cual aplica un relleno de ceros para mantener la dimensión original de la entrada y bloque residual convolucional que consiste en aplicar capas de convolución para igualar la dimensión de la entrada [44].

5.4 PROPUESTA DEL MODELO DE DEEP LEARNING

5.4.1 CARGA DE DATOS:

Para realizar la carga del dataset generado se utiliza un método de lectura proveniente de la librería tensor flow, una vez obtenida la información del dataset, se pasa por un proceso inicial de decodificación, el cual permite utilizar la información en las dimensiones, cantidades y formatos correspondientes para ser leída y procesada.

5.4.2 PRE PROCESAMIENTO

Para el pre procesamiento del dataset, se realizaron una serie de pasos, primero se obtiene el bloque completo de información proveniente de la carga datos, segundo se separan una a una las características del dataset y se individualizan para posteriormente ser procesadas haciendo la separación de cada una de ellas en dos segmentos de información, uno para entrenamiento y otro para la evaluación del modelo de deep learning, el porcentaje que se tuvo en cuenta para cada segmento fue de 70% y de 30% respectivamente. Se evalúa el modelo para estimar la calidad de la generación de patrones para los datos en los que el modelo no ha sido entrenado. Sin embargo, dado que las instancias futuras tienen valores de destino desconocidos y se puede comprobar ahora mismo la precisión de las predicciones para las instancias del futuro, se tienen que utilizar algunos de los datos para los que ya se conoce la respuesta como proxy para los datos futuros. Evaluar el modelo con los mismos datos que se han utilizado para el entrenamiento no es útil, ya que recompensa a los modelos que pueden "recordar" los datos de entrenamiento en lugar de generalizar [45].

5.4.3. CONSTRUCCION DEL MODELO

5.4.3.1. Arquitectura

El modelo (Figura 6) se compone inicialmente de dos estructuras paralelas de 7 bloques RESNET que actúan sobre las matrices de distancia de ventana deslizante a la cual llamaremos ‘matriz a’ y sobre las matrices de distancia ventana deslizante posterior a la cual llamaremos ‘matriz b’ esto con el fin de ir ajustando los pesos de las matrices y de ir bajando de a poco sus dimensiones, de igual manera a la altura del cuarto bloque se empieza a procesar la matriz de puntuación específica de la posición (PSSM) con 4 bloques RESNET, la PSSM necesita menor cantidad de bloques ya que sus dimensiones iniciales son más pequeñas que las de las matrices a y b, una vez terminado este proceso, se realiza la concatenación de las salidas resultantes de los bloques de la ‘matriz a’ y de la ‘matriz b’, este resultado es pasado de nuevo por un bloque RESNET y una capa de Maxpooling2d con el fin de reducir sus dimensiones y poder concatenarlas al resto de características. Como paso final se aplica la capa de flatten a las tres matrices procesadas para dejarlas unidimensionales y se concatenan junto a la distancia, las escalas VSHE de ventana deslizante y las escalas VSHE de ventana deslizante posterior, una vez concatenadas se pasan como entrada a 6 capas Dense conectadas entre sí, 5 con función de activación ReLU y la última con función de activación Softmax, ya que es un problema de clasificación.

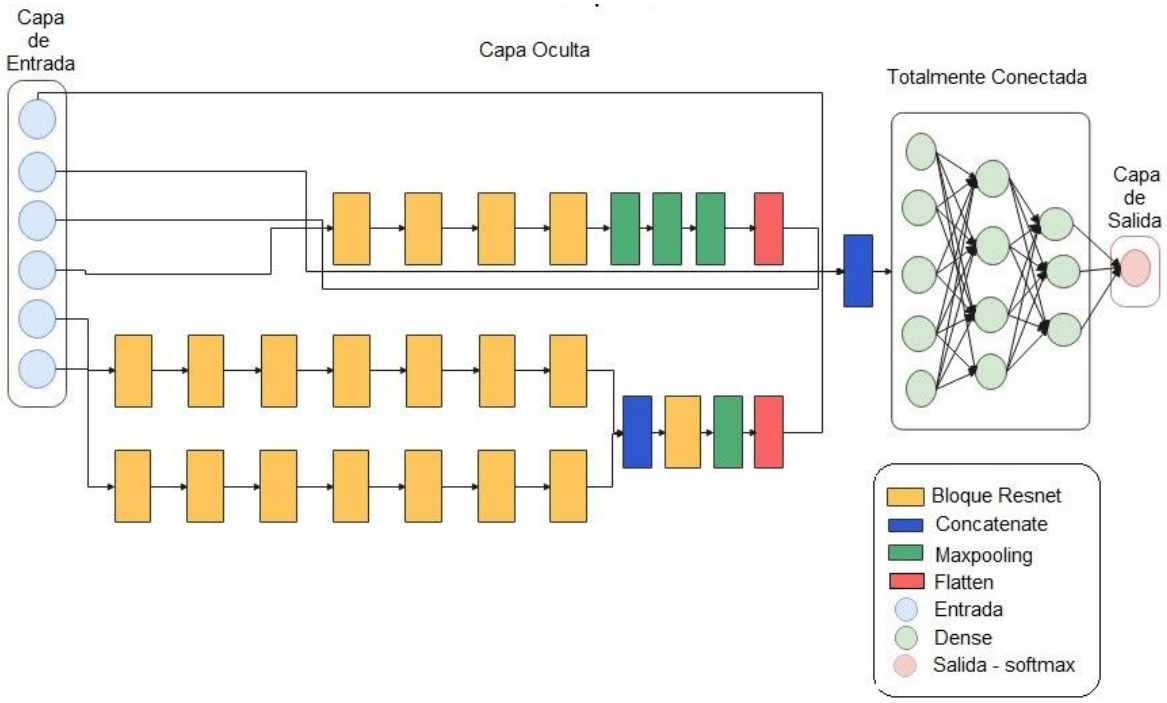

Figura 6. Modelo Red Neuronal Profunda

5.4.3.2 Descripción

Entradas:

Para la construcción del modelo se generaron 6 entradas respectivamente:

- Matrices de distancia de ventanas deslizantes (Figura 7): Son dos tensores independientes, los cuales representan cada uno una matriz multidimensional de elementos, todos los elementos son de un único tipo de datos conocido, sus dimensiones son de 21x21x7, donde 21x21, representa el tamaño de filas y columnas por matriz y el 7 la cantidad total de matrices asociadas a cada ventana deslizante.

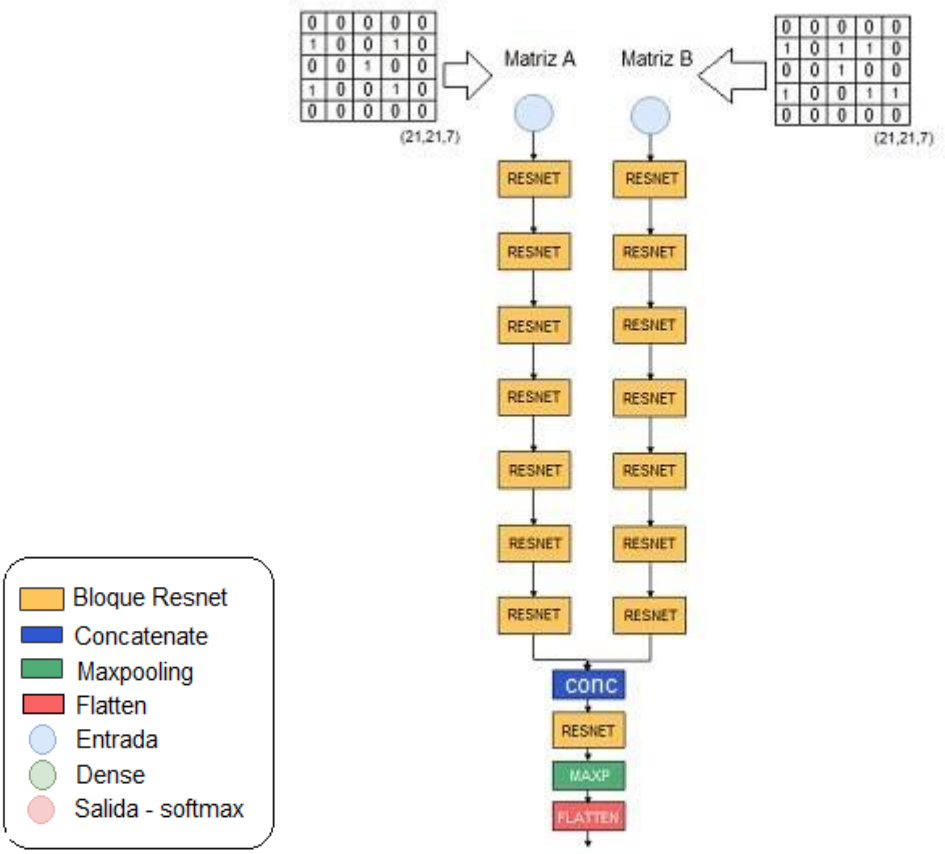

Figura 7. Entradas Matriz A y Matriz B

Las matrices de entrada, denominadas Matriz A y Matriz B pasan por 7 bloques resnet convolucionales cada una de manera paralela, a través de estos bloques se procesan y redimensionan las matrices, disminuyendo en 2 unidades sus dimensiones de filas y columnas por cada bloque resnet, esto sucede ya que en cada bloque resnet (Figura 8) internamente se le aplican dos capas convolucionales, una capa convolucional que reduce su dimensión, y la otra que mantiene las dimensiones originales, pero realiza su respectivo proceso de entrenamiento.

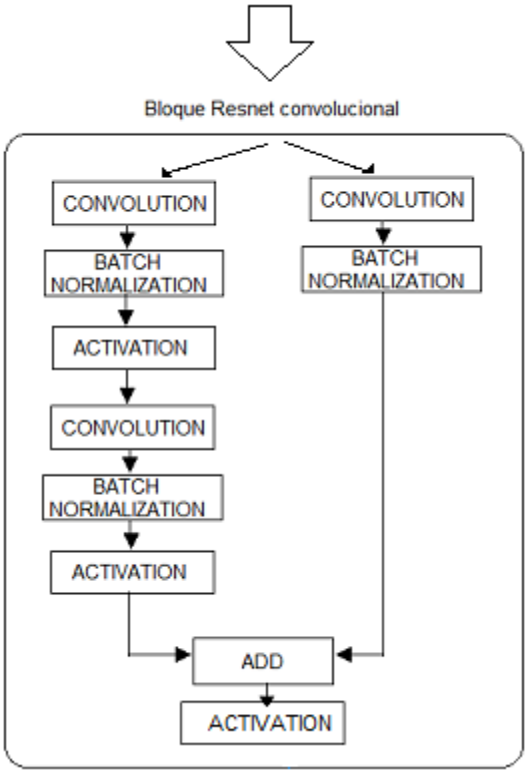

Figura 8. Resnet Convolucional

Tanto a la entrada como al residuo se le aplica una capa convolucional que disminuye sus dimensiones en dos unidades en filas y columnas, mientras que la última dimensión de los tensores toman el valor aplicado a los filtros de la capa convolucional, luego la entrada continua el proceso pasando de nuevo por tres capas (convolucional, batchnormalization, activation), para luego ser sumada con el residuo , ya que poseen las mismas dimensiones, esto sucede, gracias a que la segunda capa de convolución tiene el parámetro padding en 'same' por ello no modifica la dimensión, así las dos son simétricas y se pueden sumar.

Pasados los 7 bloques resnet para cada matriz de distancia de ventana deslizantes, se concatenan sus salidas y ahora de manera conjunta se pasan estas características por un nuevo bloque resnet que disminuye sus dimensiones en dos unidades y una capa de maxpooling que selecciona las características más relevantes de sus entradas disminuyendo aún más el valor de las dimensiones hasta en un 50%, luego al producto de este proceso se le aplica la capa de Flatten para generar un vector que finalmente será concatenado con el resto de características.

- Distancia: Es el valor numérico resultante entre la resta de las posiciones de los elementos centrales de cada par de ventanas deslizantes.
- Matriz de puntuación específica de la posición (PSSM) (Figura9): Es un tensor de dimensiones 22x19x1, que es una matriz de 22 filas y 19 columnas, las filas representan los 22 aminoácidos elementales que se encuentran en la secuencia de una proteína, y 19 simboliza la cantidad de elementos o aminoácidos que posee cada ventana, al ser una matriz, se hace el procedimiento similar a las matrices de distancia de ventana

deslizantes, pasando la matriz por 4 bloques resnet, esta vez menor cantidad de bloques ya que las dimensiones de la matriz también son menores, una vez se pasan por los cuatro bloques, el producto generado es pasado finalmente por 3 capas de maxpooling que ayudan a reducir su dimensionalidad a un punto en el cual, se puede realizar fácilmente su aplanamiento por medio de la capa flatten para logar crear un vector unidimensional que puede ser concatenado con el resto de características tratadas.

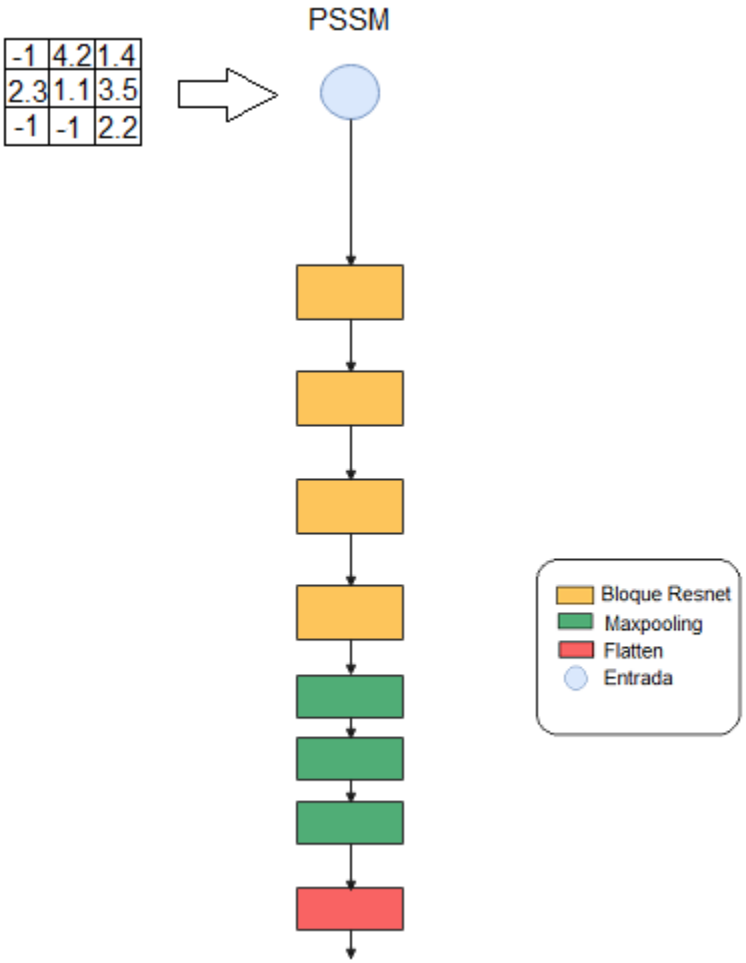

Figura 9. Matriz de puntuación específica de la posición

- Escalas VSHE de ventana deslizante y de ventana deslizante posterior: Cada una de estas entradas está representada por un vector de 8 elementos, cada elemento representa una característica o propiedad de los aminoácidos que componen cada ventana deslizante propia de la secuencia de aminoácidos de una proteína en particular.

Capas:

En el modelo de deep learning propuesto se hace uso de varias capas que interconectadas entre si componen la arquitectura del modelo, las capas son los componentes básicos de

las redes neuronales, una capa consta de una función de cálculo de tensor de entrada y tensor de salida y algún estado. Se puede llamar a una instancia de capa, como una función, sin embargo, a diferencia de una función, las capas mantienen un estado, se actualizan cuando la capa recibe datos durante el entrenamiento y se guardan en una variable de almacenamiento [46].

En el modelo se utilizaron bloques RESNET cuya arquitectura contiene las siguientes capas:

- Capa convolucional: La convolución es una operación matemática sencilla que se suele utilizar para el tratamiento y el reconocimiento de imágenes, su efecto se asimila a un filtrado[47]. En este caso fue implementada en las matrices usadas para mostrar la relación de distancias entre los aminoácidos de cada ventana deslizante, así como también en la matriz de puntuación específica de la posición (PSSM).

La capa convolucional hace uso de los siguientes parámetros: tamaño de kernel, filtros, y padding.

Kernel (Figura 10): se utiliza para extraer ciertas 'características' de una imagen o matriz de entrada. Un kernel es una matriz de pesos, que se desliza a lo largo de la imagen o matriz y se multiplica con la entrada, de modo que la salida obtiene el valor representativo o característico de esa fracción de la matriz [48]. A la capa convolucional se le pasa el tamaño del kernel que será aplicado a la matriz, en este caso se usó un kernel de tamaño tres, lo cual quiere decir que tendrá tres filas y tres columnas como se ve en el siguiente ejemplo.

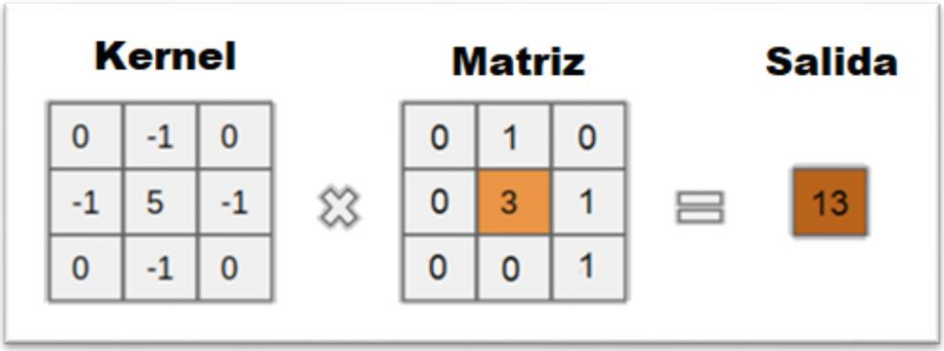

Figura 10. Aplicación Kernel (3,3)

Filtros: Un filtro es una concatenación de múltiples kernel, cada kernel asignado a un canal particular de la entrada [49].

Padding (Figura 11): Es una ‘capa’ extra de filas y columnas de ceros que se le adicionan alrededor de la matriz de entrada para que al aplicar la convolución está mantenga su dimensión. Para ser aplicada se usa el parámetro ‘same’, si no se quiere usar se utilizar el parámetro ‘valid’, de modo que la dimensión original disminuye en proporción al tamaño del kernel [49].

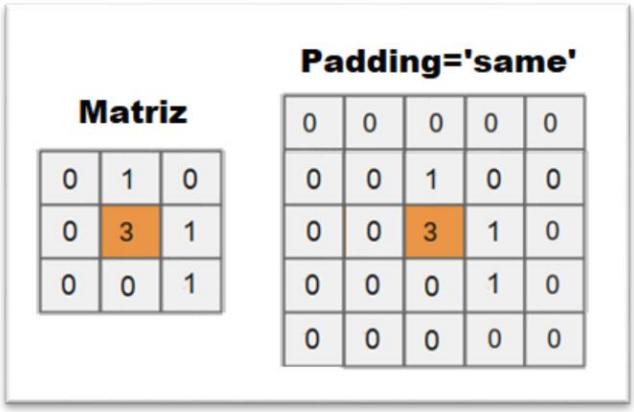

Figura 11. Aplicación Padding 'same'

- BatchNormalization: La normalización por lotes aplica una transformación que mantiene la salida media cercana a 0 y la desviación estándar de salida cercana a 1 [50].
- Activation(ReLU): Esta capa aplica una función de activación a una salida, en los bloques RESNET se usa la función ReLU, la cual es una función que toma valor cero si el valor a evaluar es negativo, o el mismo valor si es mayor o igual a cero max(x, 0) [51].
- Add: Toma como entrada una lista de tensores, todos de la misma dimensión, y devuelve un solo tensor (también de la misma dimensión) [52]
- Concatenate: Toma como entrada una lista de tensores, todos de la misma forma excepto el eje de concatenación, y devuelve un único tensor que es la concatenación de todas las entradas [53].
- Capa Maxpooling (Figura 12): Reduce la muestra de la entrada a lo largo de sus dimensiones espaciales (alto y ancho) tomando el valor máximo sobre una ventana de entrada (de tamaño definido por pool_size) para cada canal de la entrada. La ventana se desplaza a strides a lo largo de cada dimensión.
  Pool_size: entero o tupla de 2 enteros, tamaño de la ventana sobre la que tomar el máximo. tomará el valor máximo en una ventana de agrupación de 2x2. Si solo se especifica un número entero, se utilizará la misma longitud de ventana para ambas dimensiones. (2, 2).
  Strides: Entero, tupla de 2 enteros o Ninguno. Valores de zancadas. Especifica cuánto se mueve la ventana de agrupación para cada paso de agrupación [54].

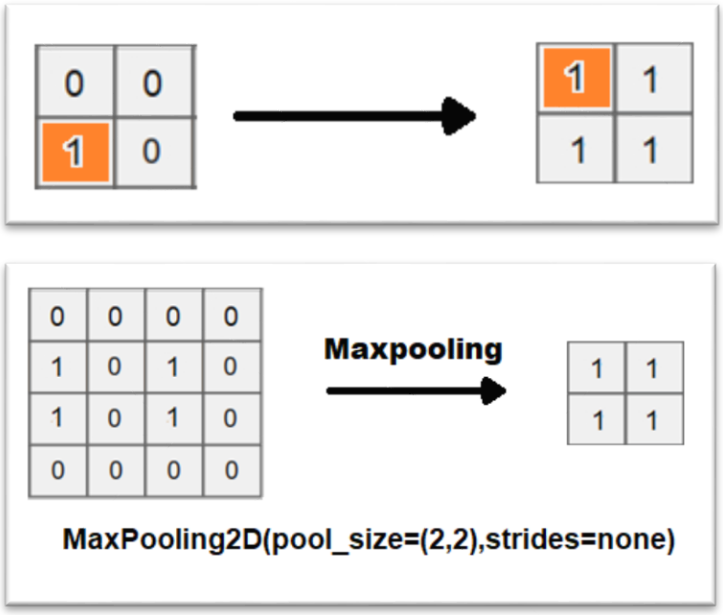

Figura 12. Aplicando Maxpooling (2,2)

- Flatten: Aplana la entrada. No afecta el tamaño del lote [55].
- Dense: es la capa regular de red neuronal profundamente conectada. Es la capa más común y de uso frecuente. La capa densa realiza la siguiente operación en la entrada y devuelve la salida.

  output = activation(dot(input, kernel) + bias), donde:

  Input: Representa los datos de entrada
  Kernel: Representa los datos de peso
  Dot: Representa el producto escalar de todas las entradas y sus pesos correspondientes
  Bias: Representa un valor sesgado utilizado en el aprendizaje automático para optimizar el modelo
  Activation: Representa la función de activación [56].

Salidas:

- Activation(Softmax): Softmax convierte un vector de valores en una distribución de probabilidad. Los elementos del vector de salida están en el rango (0, 1) y suman 1. Softmax se usa a menudo como la activación de la última capa de una red de clasificación porque el resultado podría interpretarse como una distribución de probabilidad [57].

A continuación, en el siguiente link, se encuentra la estructura del modelo de deep learning propuesto a un nivel más detallado de capas y dimensiones:

https://drive.google.com/file/d/13TLgt1enFr7uzSLnKEmTyZOBxMBBUgqO/view?usp=sharing

5.4.4 ENTRENAMIENTO

Una vez creado el modelo se procede a realizar el entrenamiento, pero antes de comenzar a entrenar el modelo es necesario determinar la función de perdida, optimizador y las métricas de evaluación. Tensor Flow define la función **compile**, que permite realizar la respectiva configuración.

El optimizador: es una función que permite realizar el ajuste de pesos. Según la documentación de Tensor Flow se han desarrollado diferentes tipos de algoritmos que se usan para realizar esta tarea dependiendo del objetivo de cada modelo, entre estos se encuentra Adam, un método que usa un enfoque de descenso de gradiente estocástico (SGD) que se basa en la estimación adaptativa del error de gradiente para el estado actual del modelo, permitiendo así actualizar la tasa de aprendizaje o learning_rate.

La tasa de aprendizaje es un hiperparametro configurable que por defecto contiene un valor 0,001, generalmente el valor se comprende entre rangos de 0,0 a 1,0 [58].

La función de perdida: evalúa el desempeño de la red neuronal con base en las predicciones realizadas para un conjunto de datos, si su resultado es alto, indica que el aprendizaje es deficiente, mientras que si es bajo indica que la red tiene un buen aprendizaje.

Existen diferentes funciones de perdida en la documentación de TensoreFlow, sin embargo, durante la investigación, CategoricalCrossentropy es la función que se usa para enfoques que contemplan más de dos clases de etiquetas [59].

Finalmente, para terminar la configuración se establece el conjunto de métricas que permitirán medir la capacidad de aprendizaje del modelo, entre ellas se establecen.

Accuracy (6): que evalúa el desempeño del modelo y se calcula con el número de aciertos sobre el total de predicciones.

$$accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}} \quad (6)$$

Recall (7): Evalúa la capacidad del modelo para predecir muestras positivas y se calcula como la relación entre el número de muestras positivas clasificadas correctamente como positivas y el número total de muestras positivas.

$$recall = \frac{True_{positive}}{True_{positive} + False_{negative}} \quad (7)$$

Precisión (8): evalúa la exactitud del modelo para hacer clasificaciones de muestras positivas y se calcula con la relación entre el número de muestras positivas clasificadas correctamente y el número total de muestras clasificadas como positivas (correctas o incorrectas)

$$precision = \frac{True_{positive}}{True_{positive} + False_{positive}} \quad (8)$$

Finalmente, la métrica AUC o Área Bajo la Curva mide la exactitud de las predicciones del modelo, define cuatro variables para realizar los cálculos, true_positives, true_negatives, false_positives false_negatives [60].

5.4.5 VALIDACION

La validación del modelo se ejecuta después de haber finalizado el entrenamiento por cada época, Keras permite separar una porción del dataset tanto para realizar el entrenamiento, como para validar el desempeño del modelo. El método Test_on_batch recibe como parámetros las x que son las estradas e y la salida, la información proporcionada es un valor escalar o una lista de escalares dependiendo de la configuración establecida en la compilación del modelo y estas medidas permiten tener una visión parcial del entrenamiento[61] .

CAPITULO 6

6. IMPLEMENTACION

Durante el proceso de creación y desarrollo del proyecto se hizo uso de una serie de herramientas las cuales fueron fundamentales para la realización e implementación del mismo, a continuación, explicamos y exponemos cada una de las herramientas utilizadas y los productos generados:

6.1. LENGUAJE DE PROGRAMACION

Para realizar la implementación del modelo de deep learning para la predicción del mapa de contacto de proteínas se hizo en primera instancia la selección del lenguaje de programación, el cual finalmente fue Python. Python es un lenguaje de programación que admite la creación de una amplia gama de aplicaciones. Los desarrolladores lo consideran una excelente opción para proyectos de inteligencia artificial (IA), aprendizaje automático y aprendizaje profundo como es el caso.

En el aprendizaje automático hace uso de algoritmos para analizar datos, aprender y tomar mejores decisiones, con el aprendizaje profundo ocurre de manera similar, pero tiene capacidades diferentes, como sacar conclusiones que se asemejan a la toma de decisiones humana. Es posible gracias al uso de capas bien estructuradas de algoritmos inspirados en la red neuronal del cerebro humano [62].

Las razones más importantes para usar Python en este proyecto es que provee librerías y frameworks que facilitan la programación para esta rama del machine learning y en especial para ayudar a resolver o generar un aporte al problema de la predicción del mapa de contacto de proteínas como lo son TensorFlow, biopython, itertools, las cuales en su gran mayoría han sido utilizadas para proyectos similares, generando buenos resultados y un ahorro significativo de tiempo.

6.2. LIBRERIAS

- **TensorFlow:** Es un framework creado por Google para crear modelos de aprendizaje profundo que utilizan redes neuronales multicapa, se utiliza en aplicaciones complejas con gran precisión para detección de imágenes, videos, texto, incluso audio [63].
- **Biopython:** Un conjunto de herramientas de acceso libre para estudios en biología computacional y bioinformática, desarrollada en python. Dentro de sus funciones se puede encontrar el manejo de secuencias y anotaciones de secuencias, además tiene la capacidad de manejar diferentes formatos de archivo.
- **Itertools:** Este módulo implementa varios bloques de construcción de iteradores inspirados en construcciones de APL, Haskell y SML. Cada uno ha sido refundido en una forma adecuada para Python. El módulo estandariza un conjunto básico de herramientas rápidas y eficientes en memoria que son útiles por sí mismas o en

combinación. Juntos, forman un "álgebra iteradora" que hace posible construir herramientas especializadas de manera eficiente en Python puro [64].

- **Peptides:** Paquete R para calcular varias propiedades fisicoquímicas e índices para secuencias de aminoácidos, así como para leer y trazar archivos de salida 'XVG' del paquete de dinámica molecular 'GROMACS' [65].

6.3. ENTORNO DE DESARROLLO

6.3.1. COLAB:

Para la implementación se hizo uso de la plataforma Google Colab que permite a los usuarios escribir y ejecutar código python en el navegador. Se enfoca principalmente en tareas de aprendizaje automático, análisis de datos y educación.

6.3.2. LIMITACIONES:

Esta plataforma permite hacer uso de servidores con excelentes características, sin embargo, su uso no es ilimitado y esto varía dependiendo de la subscripción adquirida, entre las limitaciones se encuentra el tiempo de ejecución, cantidad de memoria Ram y espacio en disco, así como el uso de GPU, TPU [66].

Para la implementación se adquirió una suscripción paga a Colab Pro +, que a diferencia de las dos opciones disponibles (Gratuita, Colab Pro) tiene acceso a mayores privilegios, entre ellos acceso a mejores y más rápidas maquinas remotas para realizar la ejecución de nuestros códigos, mayor tiempo de ejecución, ejecución en segundo plano y mayor espacio en disco, características que fueron determinantes y necesarias para poder procesar la gran cantidad de información que manejamos y generar los resultados en tiempos relativamente cortos.

6.4. PRODUCTOS GENERADOS

6.4.1 DATASET

La implementación para la captura y obtención de características utilizadas en el entrenamiento de la Red Neuronal propuesta, se genera a partir de la siguiente lógica. Antes de iniciar con la creación del algoritmo principal, se debe mencionar que previamente se vio la necesidad de descargar un conjunto de identificadores de proteínas y sus respectivos archivos, que fueron la fuente principal de información para alimentar el dataset. Ahora bien, lo primero es configurar variables globales que tienen como finalidad establecer reglas de control. En la figura 13, para las (líneas 2 y 3) se establecen probabilidades iniciales para aceptar o no, un registro que sea no contacto.
Se establece en la (línea 4) el tamaño de las ventanas deslizantes, que debe ser un número impar, y finalmente se calcula el número de caracteres que se agregan por defecto tanto al inicio como al final de cada secuencia.

```
1  INICIO
2     rechazo=0.97
3     ingreso=0.03
4     n=19
5     numero_caracteres=(n // 2)
6  FIN
```

Figura 13. Configuración inicial variables para balance dataset

Ya establecidos estos valores, se inicia la lógica principal para la generación de la data. Inicialmente definimos dos vectores y dos variables de conteo para controlar la cantidad de registros almacenados durante el proceso (Líneas 2-5).

Se obtiene un conjunto de Id's alojados en una ruta especifica (ver Figura 14 línea 7). Posteriormente se crea la ruta donde se va a almacenar el dataset. Se establece una cantidad finita de proteínas donde se procesará una por cada iteración. Para cada proteína se obtendrá su correspondiente archivo con formato .ent haciendo uso de BioPython, librería que permite manipular este tipo de archivos para obtener la secuencia de aminoácidos a la cual, en la línea siguiente se adicionará un número específico de caracteres especiales tanto al inicio como al final de la secuencia (ver Figura 13 línea 5). La idea surge a partir de la necesidad de relacionar los centros de las ventanas deslizantes. Ahora, con la secuencia ya lista se genera un conjunto de ventanas diferentes, combinadas entre sí, sin repetición y donde la cantidad varía dependiendo de la dimensión de la secuencia. La finalidad de las ventanas es obtener la relación que existe entre aminoácidos.

Ahora se genera la matriz de distancia, a partir del archivo .ent perteneciente a la proteína en proceso, esto se logra usando métodos existentes de BioPython para realizar este tipo de cálculos.

Con la información obtenida hasta este punto se comienza a iterar la matriz de distancia, es preciso comprender que las dimensiones de la matriz de distancia, la cantidad de combinaciones de ventanas y el tamaño de la secuencia al cuadrado, deben ser iguales, ya que todo parte desde la secuencia original, por lo tanto, cada iteración procesa la relación directa entre dos aminoácidos, pero representados de forma distinta. Por lo que en cada iteración se obtienen la dupla de ventanas relacionadas y la distancia entre los dos aminoácidos en estudio.

Retomando, se obtiene una combinación de ventanas que se compone de una tupla con dos vectores de 19 aminoácidos, donde el aminoácido estudiado en cada ventana está ubicado en el centro, específicamente en la posición 9 de cada ventana. Es preciso comprender que las posiciones en vectores se inician en 0.

Las ventanas deslizantes serán procesadas más adelante para obtener una serie de matrices por ventana que contendrá información relacionada con los aminoácidos vecinos.

A continuación, se toma la distancia que existe entre los aminoácidos analizados en la iteración, que es proporcionada por la matriz de distancia, la cual fue calculada en la (ver Figura 14 línea 14). Esta medida se codifica para 4 clases (ver Figura 14 línea 19), donde la clase correspondiente depende de la distancia que separa los aminoácidos:

$$(0 , 8] = \text{Clase } 1$$

$$(8 , 10] = \text{Clase } 2$$

$$(10 , 15] = \text{Clase } 3$$

$$(15 , \infty) = \text{Clase } 4$$

Una vez definida la clase a la que corresponde la distancia obtenida anteriormente es preciso definir si esta relación de aminoácidos se acepta o no como un registro valido en el dataset. El siguiente pseudocódigo (líneas 20-23) controla que el número de registros no contacto entre aminoácidos sea mayor que los contactos, ajustando la probabilidad por cada iteración. Este control se tiene en cuante ya que para todas las proteínas el número de no contacto es muy alto, generando un desbalance en los datos de entrenamiento. Si la clase es aceptada, será el objetivo que el modelo debe predecir y así se ha generado la primera característica.

Posteriormente (línea 25), se calcula la distancia posicional entre las ventanas deslizantes obtenidas en la (línea 17), que es la diferencia de las posiciones i y j de la iteración en curso. Esta será la segunda característica encontrada.

La tercera y cuarta característica es el vector de puntuación de componentes principales de propiedades hidrofóbicas, estéricas y electrónicas (VHSE) que indica la composición química de los aminoácidos que componen a una ventana (líneas 29-30).

Por otro lado, se calcula la matriz PSSM (position specific score matrix) que es la representación de un conjunto de secuencias alineadas y en este caso representaría los aminoácidos de las ventanas deslizantes. Generando así la quinta característica (Línea 31).

Se codifican las ventanas, asignando un identificador numérico a cada aminoácido (línea 33-34), para posteriormente (líneas 35-36) calcular un conjunto de matrices por cada ventana que representan la relación de distancia de cada aminoácido frente a sus vecinos y con esto obtener la sexta y séptima característica.

Finalmente se agrupa la información capturada y se escribe como un solo registro en el archivo que se utilizará para entrenar el modelo (Línea 38). Este proceso se repetirá para cada aminoácido de una secuencia que representa una proteína.

```
INICIO
    datos = [0,1]
    peso = [rechazo,ingreso]
    conteoCeros=0
    conteoUnos=0
    Pdblista = ObtenerListadoIds(UrlFile)
    Path = RutaDeAlmacenamiento
    ESCRIBIR_EN_ARCHIVO(Path)
       PARA proteinaID EN Pdblista
           PDBFile = RutaDeAlmacenamientoBD + proteinaID + formato
           SequenciaOriginal = ObtenerSecuencia(PDBFile)
           SequenciaPreparada = prepararSecuencia(SequenciaOriginal)
           IteradorCombinaciones =ObtenerIteradorCombinaciones(SequenciaPreparada)
           MatrizDeDistancia = ObtenerMatrizDeDistancia(SequenciaOriginal)
           PARA i HASTA tamaño_SequenciaOriginal-1
               PARA j+1 HASTA tamaño_SequenciaOriginal
                   VentanaA,VentanaB = siguiente(IteradorCombinaciones)
                   distanciaEntreAminoacidos = MatrizDeDistancia(i,j)
                   clase = ClasificarDistancia(distanciaEntreAminoacidos)
                   peso = controlRegistros(CantidadCeros,CantidadUnos,peso)
                   aleatorio = Aleatorio_Entre_0_y_1()
                   aleatorioConPesos = elegir_Aleaotrio(datos,peso
                   SI (clase[3]==0) O (clase[3]!=0  Y aleatorioConPesos == 1)
                   ENTONCES
                      SI clase[3]!=0 O aleatorio<0.3 ENTONCES

                          distancia = Calculo_Distancia_Para_Ventanas(i-j)

                          properties_VentanaA =
                          ObtenerComposicionesQuimicas(VentanaA)
                          properties_VentanaB =
                          ObtenerComposicionesQuimicas(VentanaB)

                          pssm=Obtener_Matriz_Pssm(VentanaA, VentanaB)

                          ventanaACodificada = CodificarVentana(ventanaA)
                          ventanaBCodificada = CodificarVentana(ventanaB)
                          matricesA = crear_matrices(ventanaACodificada)
                          matricesB = crear_matrices(ventanaBCodificada)

                          Escribir_en_archivo(PropVA,PropVB,pssm,matricesA,matrices
                          B,distancia,clase)
                      FIN SIN
                   FIN SI

               FIN PARA
           FIN PARA
       FIN PARA
FIN
```

Figura 14. Obtención características dataset

6.4.2. DIAGRAMA DE CLASES

Durante el proceso de implementación del dataset se llegó a un consenso para diseñar el diagrama de clase (Figura 15), el cual pretende mostrar la interacción entre los objetos. Se creó la clase principal Program que es quien contiene el método de inicialización y el cual se encarga de instanciar la clase Protein para construir un conjunto de proteínas que son la base para la construcción de las características. Protein contiene un listado de pares de ventanas por lo que instancia la clase Window que brinda la información. Por otro lado, la clase Program instancia la clase ControlData para excluir registros que se consideran inapropiados y que entorpecen el balance del dataset. Finalmente se consideró crear la clase FeaturesFunctions con el objetivo de contener las funciones principales que permiten la obtención de la mayoría de características.

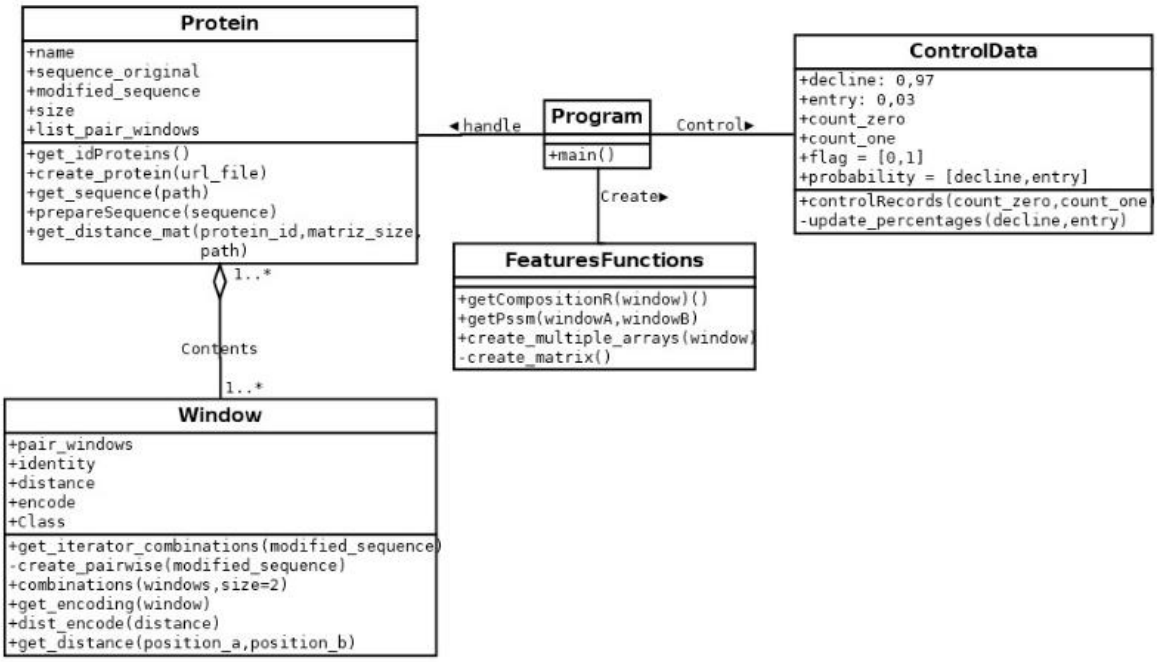

Figura 15. Diagrama de clases dataset

6.4.3. MODELO

Preprocesamiento:

Para la implementación del método de preprocesamiento se toma el dataset completo (ver Figura 16 línea 2) y se empieza a separar por características, esto con el fin de pasarlos de manera individual al método de Split en donde se define el porcentaje en el que cada característica será dividida dejando en este caso el 30% de sus datos individuales para hacer pruebas y el resto (70%) para hacer entrenamiento (línea 13), una vez se ha separado la información con sus respectivos porcentajes se retornan para ser utilizados (línea 15), unos para realizar el entrenamiento del modelo y los otros para validarlo.

```
1   INICIO
2          preprocesamiento(dataset)
3
4            //Separando las características del dataset
5            A=dataset[‘matriz_a’]
6            B=dataset[‘matriz_b’]
7            D=dataset[‘distancia’]
8            P1=dataset[‘propiedadesA’]
9            P2=dataset[‘propiedadesB’]
10           PSSM=dataset[‘pssm’]
11           Clase=dataset[‘clase’]
12
13           Entrenamiento,test=Split(A,B,D,P1,P2,PSSM,Clase,0.30)//dividir
14
15          RETORNAR Entrenamiento, test
16
17  FIN
18
19
```

Figura 16. Preprocesamiento dataset

Creación del modelo

Para la implementación de la creación del modelo primeramente se definen cuáles van a ser sus entradas (ver Figura 17 líneas 5 a 10), posteriormente se pasa cada una de las matrices por bloques resnet consecutivos, 7 para la matriz_a (línea 13), 7 para la matriz_b (línea 14) y 4 para matriz pssm (línea 15), luego se concatenan las salidas de los bloques resnet de la matriz_a y de la matriz_b, se pasan de nuevo por bloque resnet y después de ser procesados se concatenan y se almacenan en una sola variable (línea 23 - 24), estas características finalmente son pasadas por una red de neuronas totalmente conectadas que generan una salida (línea 27). Se crea el modelo pasándole las entradas inicializadas al principio del código y la salida que es el resultado del procesamiento de todas las capas sobre las características concatenadas (línea 30), se le asigna un nombre al modelo y se retorna (línea 32)

```
1   INICIO
2          cnn()
3
4            //Definir entradas
5            matriz_a=Input()
6            matriz_b=Input()
7            distancia=Input()
8            propiedadesA=Input()
9            propiedadesB=Input()
10           pssm=Input()
11
12           //bloques resnet para matrices
13           bloque_resnet(matriz_a)*7
14           bloque_resnet(matriz_b)*7
15           bloque_resnet(pssm)*4
16           //concantenar salidas de los bloques resnet matriz_a,matriz_b
17           mix=concatenar(matriz_a,matriz_b)
```

```
18
19          //bloque resnet para concatenación de matrices
20          bloque_resnet(mix)
21
22          //concatenar todas las características
23          características=concatenar(mix, pssm,
24          distancia,propiedadesA,propiedaesB)
25
26          //red neuronal totalmente conectada
27          características=neuronas(características)
28
29          //creación del modelo
30          modelo=Model(entradas,salidas=caracteristicas,nombre)
31
32        RETORNAR modelo
33  FIN
```

Figura 17. Creación del modelo deep learning

Entrenamiento y evaluación del modelo:

La implementación del modelo (ver Figura 18) inicialmente crea el modelo llamando a su respectivo método (línea 3) el cual genera un modelo de deep learning, una vez lo crea es necesario compilarlo (línea 4), en la compilación se establece cual será la función de perdida a utilizar, el learning rate que genera ajuste en los pesos a medida que el modelo se va entrenando y va aprendiendo, y las métricas de evaluación de ese entrenamiento que se tuvieron en cuenta, el accuracy, el recall, la precisión y el AUC, posteriormente se cargan los datos con los que se va a entrenar el modelo y se decodifican ya que vienen en un formato específico y se deben amoldar a los formatos requeridos (línea 5 y 6), después de cargado el dataset, este se divide en 'batch' o lotes de información con el fin de no saturar la memoria disponible (línea 7), este trozo de información uno a uno es enviado al método de preprocesamiento que separa internamente los datos y los divide en porciones de entrenamiento y pruebas (línea 10), después de tener separada la información se procede a alimentar el modelo con la misma, y realizar el entrenamiento pasándole tanto las entradas como las salidas deseadas (línea 13), terminado el entrenamiento y validación para todos los lotes de información se guarda el modelo generado y entrenado (línea 17)

```
1   INICIO
2
3         modelo = cnn()//crear modelo
4         modelo.compile(loss, Adam, metrics)//compilar modelo
5         path=dataset//cargar dataset
6         dataset=decodificar(dataset)//decodificar dataset
7         lector=dataset.batch(4096)//se divide el dataset en batch
8
9         PARA record EN lector
10              Entradas,salidas,testx,testy=preprocesamiento(record)
11              //divide dataset entrenamiento,test
12              PARA EPOCAS EN RANGO(0,25)
13                    Train=modelo.train(entradas,salidas)//entrena el modelo
14              FIN PARA
15              Test=modelo.test(testx,testy)//prueba el modelo
16        FIN PARA
17        Modelo.guardar()//guarda el modelo entrenado
```

```
18  FIN
```

Figura 18. Guardar modelo entrenado

6.4.4. Diagrama de clases:

Para la implementación del modelo se utilizaron dos clases, Program crea una instancia de la clase Dataset para obtener la información necesaria que permite realizar el entrenamiento y validación del modelo, por otra parte, se crea una instancia de la clase model, la cual nos permite agregar las entradas, capas y salidas del modelo, una vez generado, se procede a compilarlo y realizar el entrenamiento y validación, finalmente se guarda el modelo generado y entrenado (Figura 19).

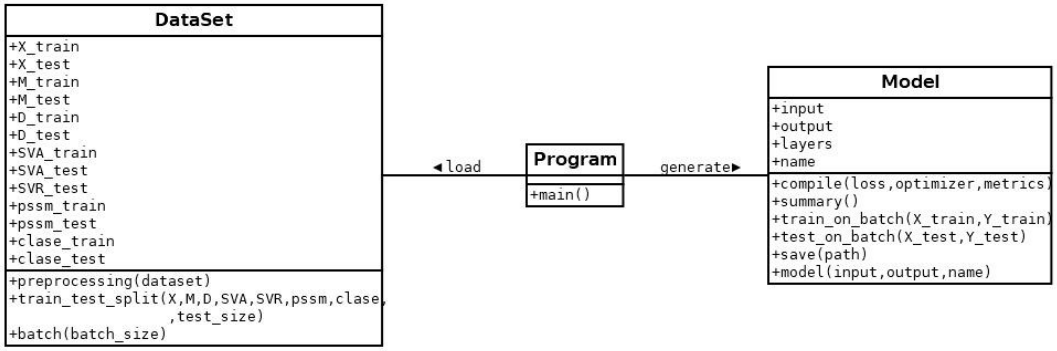

Figura 19. Diagrama de clases modelo

6.5. REPOSITORIO:

Para guardar y compartir el código desarrollado en este proyecto, usamos Github. GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones llamado Git [67].

En el siguiente link se encuentra el repositorio generado:

https://github.com/alexanderdavila/Tesis

6.6. CARACTERISTICAS GENERALES DE LA IMPLEMENTACION

Para realizar la implementación del proyecto se optó por utilizar una arquitectura distribuida (Figura 20), esto con el fin de evitar al máximo los retrasos en tiempo por fallas del sistema o por falta de recursos de hardware y software, se utiliza google colab para tener acceso a maquinas remotas aleatorias por cada ejecución que satisfacen las necesidades de requerimientos que este tipo de problemas conllevan como lo son el espacio en disco, memoria RAM, capacidad de procesamiento y tiempos de ejecución, si bien en cada ejecución se usa una maquina diferente, estas máquinas tienen siempre características muy similares, por medio de colab se hace conexión a la base de datos, y así se obtiene la información necesaria para ser procesada y guardada en la nube, posteriormente desde la

parte del cliente se accede a la nube desde cualquier maquina con conexión a internet y se hace la descarga de todos los productos generados.

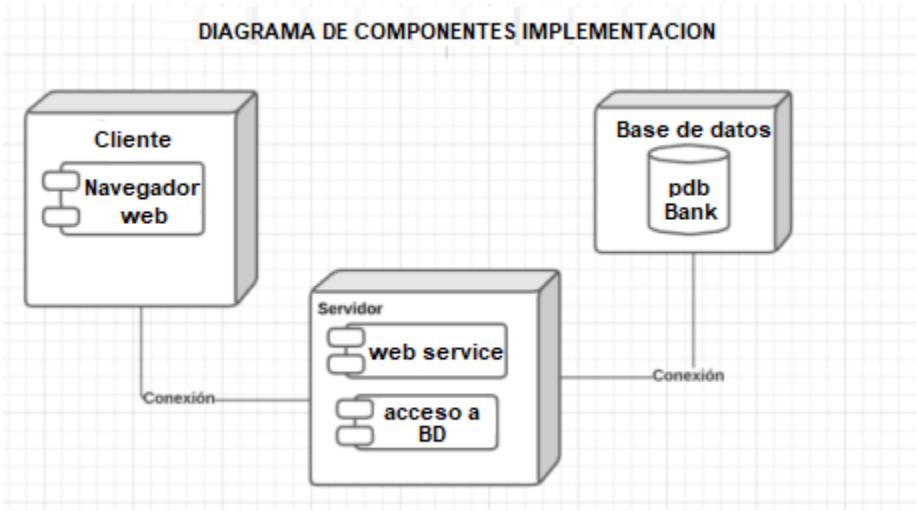

Figura 20. Diagrama de componentes

.

CAPITULO 7

7. EVALUACION

En este capítulo se muestra cómo se realizó la evaluación del modelo de deep learning propuesto para la predicción del mapa de contacto de proteínas, cuáles fueron los datos que se utilizaron para realizar esta evaluación, su descripción y procesamiento, también mostrar las métricas que fueron utilizadas y como fueron calculadas, y mostrar los resultados en comparativa con otros modelos propuestos en la literatura.

7.1 CONJUNTO DE DATOS

Para la evaluación del modelo se utilizó el paquete de pruebas pdb25-test-500.contactFeatures.pkl, de las cuales se obtuvo un conjunto de 250 proteínas que vienen en formato pkl. Los archivos pkl son archivos creados por pickle, un módulo Python que permite serializar objetos a archivos en el disco y de-serializarlos de nuevo en el programa en tiempo de ejecución. Contiene un flujo de bytes que representa los objetos [68]. Es de considerar que estas 250 proteínas de prueba son diferentes a las proteínas que se usaron para la creación del dataset, con las que luego se entrenó y valido el modelo, por lo tanto, esto permite verificar de una manera más certera que tan bueno es el modelo para la predicción de contactos de n clases.

7.2. IMPLEMENTACION

Respecto a la evaluación, inicialmente se realiza la carga del modelo generado y entrenado en el proceso anterior, de igual manera se realiza la carga del dataset de prueba. Luego los datos del dataset de prueba son procesados proteína a proteína por medio de los mismos métodos que fueron utilizados en la creación del dataset, con el fin de generar las características que el modelo entrenado toma como entradas para producir una salida que es la predicción de un contacto o de un no contacto para cada par de ventanas deslizantes provenientes de la secuencia de aminoácidos de la proteína en evaluación, simultáneamente se crea un archivo que va almacenando toda la información de la proteína que se está evaluando en formato .cm el cual es el formato nativo que utiliza CMView la cual es una herramienta interactiva de visualización y análisis de mapas de contacto [69], además esta herramienta permite también cargar directamente desde el PDB (Protein Data Bank) el mapa de contacto original de la proteína deseada por medio de su código identificador. Entonces, cuando se terminan de generar y almacenar todas las predicciones que en conjunto forman el mapa de contacto predicho para esa proteína, se procede a utilizar la herramienta CMView para dibujar y visualizar el mapa de contacto predicho al igual que el mapa de contacto original y así poder evaluar que tan buena es la predicción del modelo vs la original.

7.3. METRICAS

Las métricas que se usaron para la evaluación del proyecto miden que tan buena es la predicción que genera el modelo respecto a la proteína original y su mapa de contacto, también se generan métricas de tipo estadístico para obtener resultados del modelo y poder comparar con los otros modelos propuestos en la literatura, así como con los métodos experimentales de CASP. Los experimentos de Evaluación crítica de la predicción de la estructura de proteínas (CASP) tienen como objetivo establecer el estado actual del arte en la predicción de la estructura de proteínas, identificar qué progreso se ha logrado y resaltar dónde se puede enfocar el esfuerzo futuro de manera más productiva [70]. De este modo se presentan a continuación las métricas y resultados obtenidos:

- **MODA:** La moda es el dato que más se repite o el dato que ocurre con mayor frecuencia. Un grupo de datos puede no tener moda, tener una moda (unimodal), dos modas (bimodal) o más de dos modas (multimodal) [71].
- **MEDIA:** La media o media aritmética (9), usualmente llamada promedio, se obtiene sumando todos los valores de los datos y divide el resultado entre la cantidad de datos. Si los datos proceden de una muestra la media se representa con una x testada (x) como es el caso [72].

$$\bar{X} = \frac{\sum_1^n x_i}{n} \text{ (9)}$$

- **MEDIANA:** En ocasiones se le llama media posicional (10)(11), porque queda exactamente en la mitad de un grupo de datos, luego de que los datos se han colocado de forma ordenada [73].

$$\tilde{X} = X\ \frac{n+1}{2} \text{ (10)}$$

Para número impar de valores

$$\tilde{X} = \frac{1}{2}\ (X\ \frac{n}{2} + X\ \frac{n}{2} + 1) \text{ (11)}$$

Para número par de valores

7.4. AJUSTE DE PARAMETROS

Después de crear la arquitectura y generar el modelo, se aplicaron diferentes combinaciones de parámetros en el entrenamiento con el fin de buscar mejorar las métricas que representan la calidad de las predicciones.

En la tabla número 3 se puede observar cuales fueros los parámetros usados en cada entrenamiento.

Modelo	Tamaño dataset	#capas (Dense)	#épocas	Shuffle(size)	Tiempo entrenamiento
Entrenamiento 1	3’571.378 registros	7	60	250.000 registros	21.01 Horas
Entrenamiento 2	3’571.378 registros	7	25	400.000 registros	8.71 Horas
Entrenamiento 3	3’571.378 registros	5	25	400.000 registros	8.44 Horas
Entrenamiento 4	3’571.378 registros	5	25	No	8.21 Horas
Entrenamiento 5	3’571.378 registros	6	25	No	8.60 Horas

Tabla 3 Ajuste Parámetros

A continuación, en las Figuras 21-25 se muestran los gráficos de los resultados obtenidos en los entrenamientos generados por TensorFlow para la predicción de clases de contactos de n clases a partir de las ventanas deslizantes.

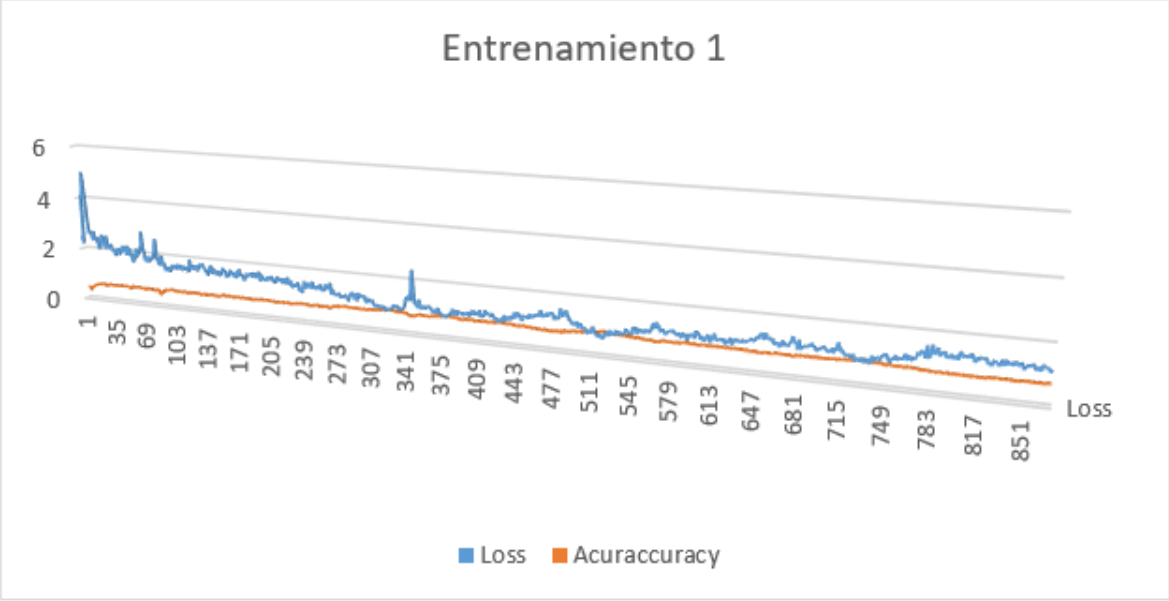

Figura 21. Primer entrenamiento loss vs accuracy

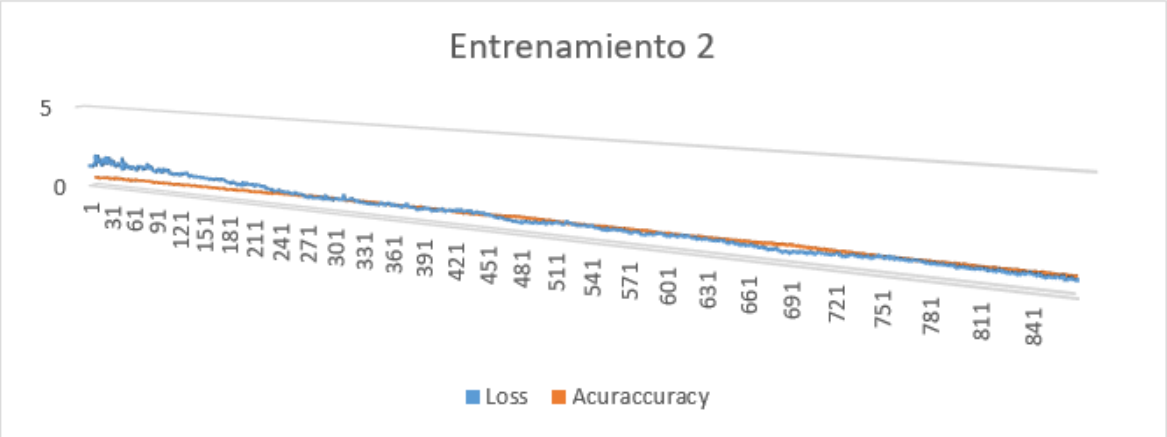

Figura 22. Segundo entrenamiento loss vs accuracy

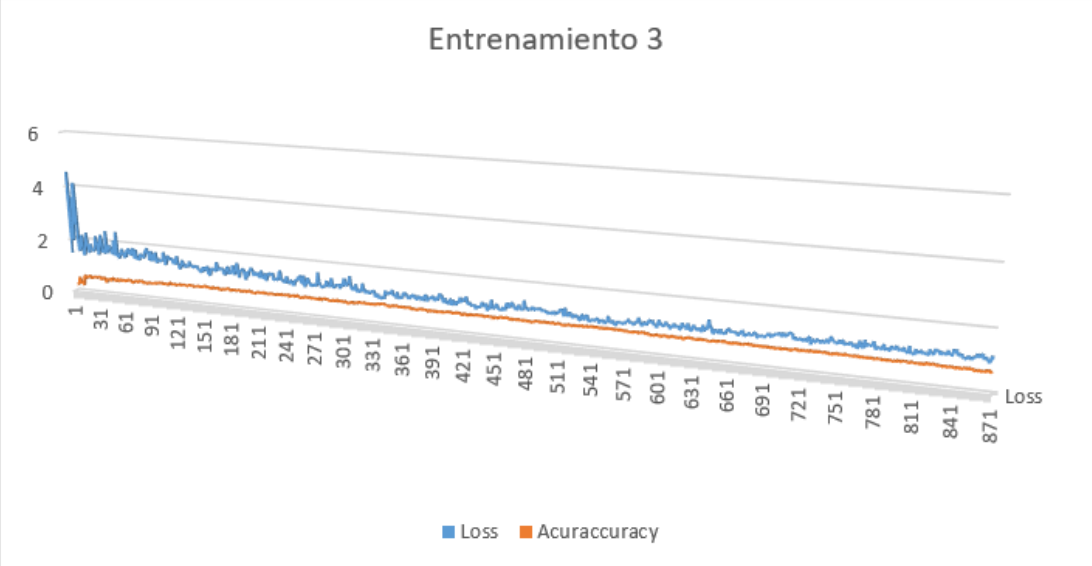

Figura 23. Tercer entrenamiento loss vs accuracy

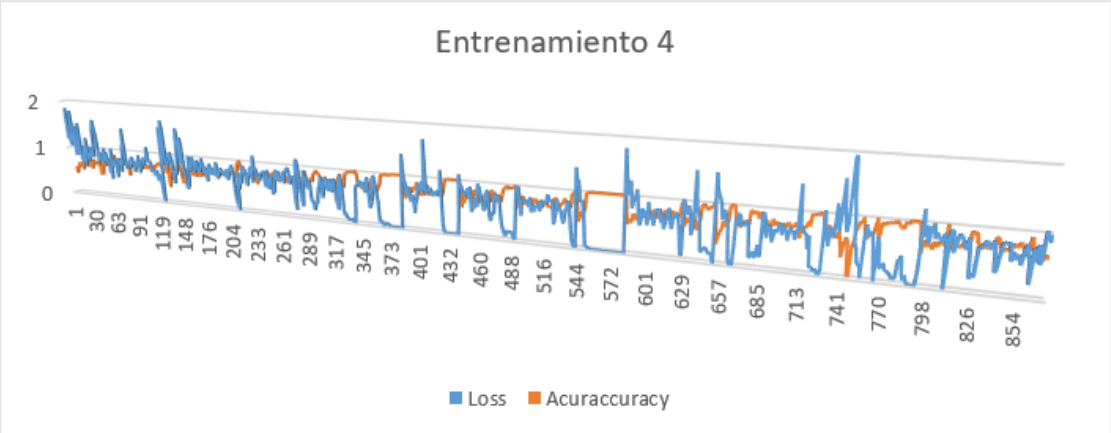

Figura 24. Cuarto entrenamiento loss vs accuracy

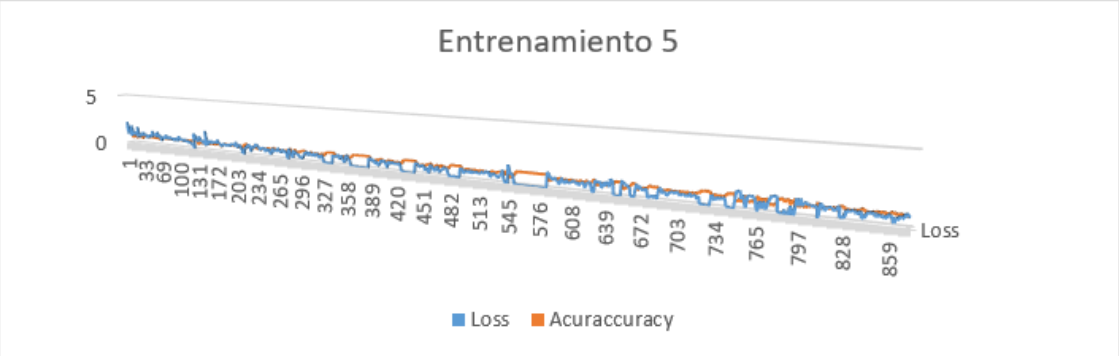

Figura 25. Quinto entrenamiento loss vs accuracy

7.5 RESULTADOS

En las Figuras 26-28 se pueden observar los resultados de las predicciones por proteína generadas por el modelo de deep learning para objetivos de prueba diferentes a los utilizados en el entrenamiento con sus respectivas métricas de evaluación, se realizó la prueba en 250 proteínas para las cuales se hicieron 3 predicciones con diferentes valores de corte (8A, 10A, 15A) de las cuales se muestra el top 20 con los mejores resultados para cada una de ellas.

Para obtener los resultados de todas las predicciones se implementó un método que compara la proteína predicha vs la proteína original obteniendo los contactos positivos, falsos positivos, falsos negativos y verdaderos negativos.

- Corte 8A: Para esta clase se obtuvieron los mejores resultados en comparación con las otras dos clases, se evidencia en las predicciones una similitud importante con respecto a la original, especialmente en la parte central del mapa de contacto, sin embargo, se denota la presencia de ruido, y poca precisión para los contactos que están alejados de la diagonal principal.

	Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
2	3wmi	179	39	0	1378	0,97556391	1	0,821100917	0,901763224	0,821100917
3	4tsh	349	89	39	6783	0,982369146	0,899484536	0,796803653	0,84503632	0,731656184
4	2wie	266	92	19	2944	0,966576332	0,933333333	0,74301676	0,827371695	0,705570292
5	4lku	85	9	32	280	0,899014778	0,726495726	0,904255319	0,805687204	0,674603175
6	2h88	387	164	27	9152	0,98036999	0,934782609	0,702359347	0,802072539	0,669550173
7	3ze3	372	117	88	7808	0,97555158	0,808695652	0,760736196	0,78398314	0,644714038
8	1tqg	315	167	32	4946	0,963553114	0,90778098	0,653526971	0,75995749	0,612840467
9	2jps	297	165	24	4974	0,965384615	0,925233645	0,642857143	0,75862069	0,611111111
10	1xg1	188	71	55	1897	0,943012212	0,773662551	0,725868726	0,74900384	0,598726115
11	4lpi	446	252	47	11036	0,974620151	0,904665314	0,638968481	0,74895062	0,598657718
12	3pmc	390	176	86	8939	0,972682723	0,819327731	0,689045936	0,74856061	0,598159509
13	1zv1	158	83	24	1815	0,948557692	0,868131868	0,65560166	0,74704917	0,596226415
14	4b4y	422	221	65	11073	0,975723623	0,866529774	0,6562986	0,74690265	0,596045198
15	3hm5	220	70	80	3908	0,964936886	0,733333333	0,75862069	0,74576271	0,594594595
16	3d3b	408	220	70	9172	0,970618034	0,853556485	0,649681529	0,73779852	0,584527221
17	1p2x	449	243	79	11790	0,974365098	0,850378788	0,648843931	0,73606574	0,582360571
18	3dww	407	230	66	11700	0,976134806	0,860465116	0,638932496	0,73333333	0,578947368
19	1v54	793	397	180	32560	0,9829944	0,815005139	0,666386555	0,73324069	0,578832117
20	2bhw	586	366	63	25781	0,983990148	0,902927581	0,615546218	0,73204273	0,577339901

Figura 26. Resultados corte 8

- Corte 10A: Para esta clase a nivel general se ve cierta similitud con la clase anteriormente expuesta, al tener mayor valor de corte este genera mayor cantidad de predicciones de contactos no necesariamente correctos lo que implica el incremento de ruido, se nota un intento de predecir los contactos que se encuentran más alejados de la diagonal principal con baja precisión.

	Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
2	4pna	130	22	18	265	0,908045977	0,878378378	0,855263158	0,866666667	0,764705882
3	3wmi	227	90	1	1278	0,942982456	0,995614035	0,716088328	0,83302752	0,713836478
4	4lku	98	34	25	249	0,854679803	0,796747967	0,742424242	0,76862745	0,624203822
5	4tsh	467	224	62	6507	0,960606061	0,882797732	0,675832127	0,7655737	0,620185923
6	2wie	366	208	20	2727	0,93134598	0,948186528	0,637630662	0,7625	0,616161616
7	3ci9	181	50	72	643	0,871035941	0,71541502	0,783549784	0,74793388	0,597359736
8	4jpr	314	149	117	2270	0,906666667	0,728538283	0,678185745	0,70246085	0,54137931
9	3ze3	513	298	144	5715	0,933733133	0,780821918	0,632552404	0,69891008	0,537172775
10	4b4y	622	505	55	9844	0,949210956	0,918759232	0,55190772	0,68957871	0,526226734
11	1xg1	254	189	40	1728	0,896426956	0,863945578	0,573363431	0,68928087	0,525879917
12	1zv1	217	180	22	1292	0,881940386	0,907949791	0,546599496	0,68238994	0,517899761
13	4lpi	660	599	45	10324	0,944616443	0,936170213	0,524225576	0,67209776	0,506134969
14	2jps	423	393	35	4609	0,921611722	0,923580786	0,518382353	0,66405023	0,49706228
15	2bhw	817	764	69	23103	0,966347513	0,922121896	0,516761543	0,66234293	0,495151515
16	3d3b	608	539	82	8362	0,935251799	0,88115942	0,530078466	0,66194883	0,494711147
17	1tqg	425	419	18	4598	0,91996337	0,959367946	0,503554502	0,66045066	0,493039443
18	3tx3	967	865	192	27137	0,963752958	0,834339948	0,527838428	0,64660648	0,477766798
19	3pmc	495	490	70	7591	0,935230164	0,876106195	0,502538071	0,63870957	0,469194313
20	3r2d	582	491	182	7790	0,925594251	0,761780105	0,542404473	0,63364181	0,46374502

Figura 27. Resultados corte 10

- Corte 15A: Finalmente para esta última clase con corte 15A las métricas calculadas muestran un bajo rendimiento en la calidad de las predicciones, ya que se evidencia

un incremento importante respecto a las otras clases, generando mayor cantidad de falsos positivos, es decir, predicción de contactos errados.

	Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
2	4pna	141	75	7	212	0,81149425	0,9527027	0,652777778	0,774725275	0,632286996
3	4lku	111	88	12	195	0,75369458	0,90243902	0,557788945	0,689440994	0,526066351
4	3wmi	227	242	1	1126	0,84774436	0,99561404	0,484008529	0,651362984	0,482978723
5	3ci9	197	156	56	537	0,77589852	0,77865613	0,558073654	0,650165017	0,481662592
6	4tsh	527	919	2	5812	0,8731405	0,99621928	0,364453665	0,533670886	0,363950276
7	2wie	386	809	0	2126	0,75639868	1	0,323012552	0,488298545	0,323012552
8	4jpr	344	654	87	1765	0,74	0,79814385	0,344689379	0,481455563	0,317050691
9	3ze3	578	1316	79	4697	0,79085457	0,87975647	0,305174234	0,453155625	0,292954891
10	1e8j	237	600	3	486	0,54524887	0,9875	0,283154122	0,440111421	0,282142857
11	3la9	774	1834	210	12060	0,86261594	0,78658537	0,296779141	0,430957684	0,274662881
12	2cc6	293	766	30	927	0,60515873	0,90712074	0,27667611	0,424023155	0,269054178
13	4hrs	254	677	22	1192	0,67412587	0,92028986	0,272824919	0,42087821	0,266526758
14	2rh0	517	1373	148	6740	0,82672591	0,77744361	0,273544974	0,404696673	0,253680079
15	1xg1	285	831	9	1086	0,62008141	0,96938776	0,255376344	0,404255319	0,253333333
16	1zv1	221	637	18	835	0,61718293	0,92468619	0,257575758	0,402917046	0,252283105
17	1ctf	336	997	28	917	0,5500439	0,92307692	0,252063016	0,395992929	0,246877296
18	3bdu	209	632	57	480	0,5	0,78571429	0,248513674	0,377597109	0,232739421
19	1msc	453	1490	11	6302	0,81819283	0,9762931	0,233144622	0,37640216	0,231832139
20	1tif	314	1055	21	1460	0,62245614	0,93731343	0,2293645	0,368544601	0,225899281

Figura 28. Resultados corte 15

7.5.1. ANALISIS POR CLASE

En las siguientes matrices de confusión (Figura 29) pertenecientes a cada clase, se pueden visualizar los resultados entre la proteína original y la predicción generada, a continuación, se hace la descripción de los valores que aparecen en cada casilla:

		prediccion	
		contacto	no contacto
real	contacto	Verdadero positivo	Falso Positivo
	no contacto	Falso Negativo	Verdadero Negativo

Verdadero Positivo	**(TP)**	Es un contacto y fue predicho como contacto
Falso positivo	**(FP)**	Es un contacto y fue predicho como no contacto
Falso Negativo	**(FN)**	No es contacto y fue predicho como contacto
Verdadero Negativo	**(TN)**	No es contacto y fue predicho como no contacto

Figura 29. Matriz de confusión

Para realizar la comparación total entre la predicción y el mapa de contacto original, se utilizó el coeficiente de Tanimoto, el cual es una métrica (o puntuación) para medir la similitud de dos conjuntos de elementos.

El coeficiente de Tanimoto (12) se puede definir simplemente como la relación de la intersección de los dos conjuntos sobre la unión de los dos conjuntos.

Más precisamente, el coeficiente de Tanimoto del conjunto A y el conjunto B se puede definir como:

$$T = \frac{Nc}{(Na+Nb-Nc)} \quad (12)$$

dónde:

Na es el número de elementos en el conjunto A

Nb es el número de elementos en el conjunto B

Nc es el número de elementos que se comparten en A y B [74]

Clase 1:

		PREDICCIÓN	
		Contacto	No Contacto
REAL	Contacto	52,14%	47,86%
	No Contacto	1,36%	98,64%

Figura 30. Matriz de confusión para clase 1

Según la matriz de confusión (Figura 30) para la primera clase en el dataset de 183916 contactos se predijeron correctamente 95894 contactos lo cual equivale a un 52.14% del total de contactos correctos, y para los no contactos de 3982282 no contactos se predijeron correctamente 3928215 no contactos lo que equivale a un 98.64% del total de no contactos predichos correctamente. Para esta clase se alcanza el 40.29% de superposición al comparar con el mapa de contacto original.

Métricas promedio (Figura 31) para todo el conjunto de proteínas de evaluación de esta clase:

Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
TOTAL	95894	88022	54067	3928215	0,95542389	0,716970774	0,538230223	0,607191692	0,441485968

Figura 31. Métricas Clase 1

Clase 2:

		PREDICCIÓN	
		Contacto	No Contacto
REAL	Contacto	40,32%	59,68%
	No Contacto	1,61%	98,39%

Figura 32. Matriz de confusión clase 2

Según la matriz de confusión (Figura 32) para la segunda clase en el dataset de 326254 contactos, se predijeron con éxito 131550 contactos lo cual equivale a un 40.32% de contactos predichos correctamente, y para los no contactos de 3839944, se hizo una

predicción correcta de 3778247 no contactos equivalentes al 98.39%. Para esta clase se alcanza el 33,91% de superposición al comparar con el mapa de contacto original.

Métricas promedio (Figura 33) para todo el conjunto de proteínas de evaluación de esta clase:

Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
TOTAL	131550	194704	61697	3778247	0,918783305	0,741831273	0,42310605	0,532376063	0,367098646

Figura 33. Métricas Clase 2

Clase 3:

		PREDICCIÓN	
		Contacto	No Contacto
REAL	Contacto	17,38%	82,62%
	No Contacto	1,18%	98,82%

Figura 34. Matriz de confusión para clase 3

Según la matriz de confusión (Figura 34) para la tercera clase de 893781 contactos, se predijeron con éxito 155323 contactos lo cual equivale a un 17.38% de contactos correctamente predichos y para los no contactos de 3241750, se hizo una predicción correcta 3203549 no contactos equivalente al 98.82%de no contactos predichos correctamente. Para esta clase se alcanza el 16,66% de superposición al comparar con el mapa de contacto original.

Métricas promedio (Figura 35) para todo el conjunto de proteínas de evaluación de esta clase:

Name	TP	FP	FN	TN	Accuracy	Recall	Precision	F1	Overlap
TOTAL	155323	738458	38201	3203549	0,75597512	0,87095354	0,19187348	0,309369374	0,185367373

Figura 35. Métricas Clase 3

A continuación, se muestran los valores de las métricas estadísticas calculadas para los resultados de las predicciones de cada clase (Figuras 36-38):

Clase 1(8A):

	MEDIANA	MODA	MAXIMO	MINIMO	PROMEDIO
TP	381	340	27578	66	517,4349593
FP	339,5	286	25951	9	531,2235772
FN	137,5	94	10260	0	282,3943089
TN	13517	26181	1210090	271	25838,28455
Accuracy	0,965900383	#N/A	0,988005691	0,662878788	0,957787579
Recall	0,727727273	0,933333333	1	0,146505608	0,700016362
Precision	0,526060707	0,510752688	0,904255319	0,087319244	0,525203685

Figura 36. Resultados métricas clase 1

Clase 2 (10A):

	MEDIANA	MODA	MAXIMO	MINIMO	PROMEDIO
TP	534,5	409	1605	98	582,3114035
FP	733,5	618	3243	22	860,6885965
FN	163	70	4909	1	274,1359649
TN	10820,5	#N/A	130218	249	16742,00439
Accuracy	0,92802	#N/A	0,985178425	0,75616836	0,918898056
Recall	0,75737	0,86394558	0,995614035	0,196431495	0,741216143
Precision	0,40594	#N/A	0,855263158	0,278247502	0,42319131

Figura 37. Resultados métricas clase 2

Clase 3 (15A):

	MEDIANA	MODA	MAXIMO	MINIMO	PROMEDIO
TP	627	453	2119	111	684,2422907
FP	2743	1775	12161	75	3253,118943
FN	63	0	4677	0	168,2863436
TN	8725	11644	121300	195	14112,55066
Accuracy	0,76908	#N/A	0,955484574	0,5	0,755975123
Recall	0,90476	1	1	0,23440825	0,87095354
Precision	0,17944	#N/A	0,652777778	0,133120424	0,19187348

Figura 38. Resultados métricas clase 3

7.5.2. VISUALIZACION DE PREDICCIONES

Las siguientes imágenes (Figuras 39-47) muestran una serie de predicciones generadas por el modelo entrenado de deep learning para cada una de las clases, así como también se puede observar el mapa de contacto original para un determinado corte y una tercera imagen de comparación donde se superponen la imagen original con la generada por el modelo y así validar la calidad de la predicción.

En la tercera imagen se observan 4 colores que representan lo siguiente:

Blanco: Verdadero negativo

Verde: Falso Negativo

Negro: Verdadero Positivo

Fucsia: Falso Positivo

Proteína: 2wie

Corte 8A:

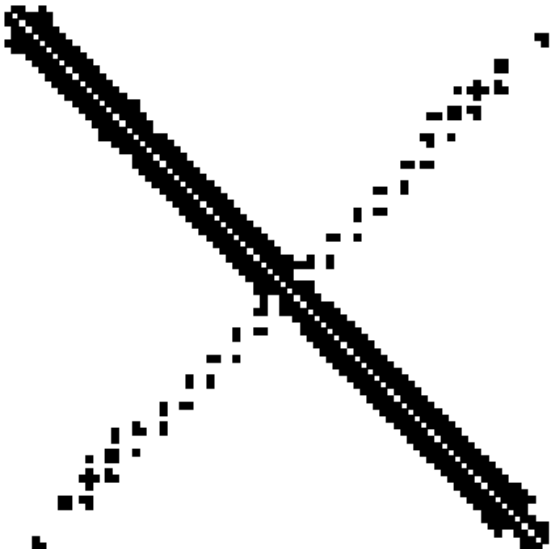

Figura 39. Mapa de contacto original

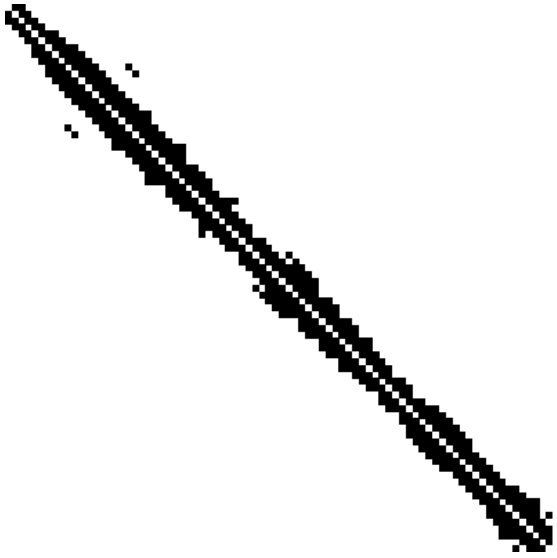

Figura 40. Predicción mapa de contacto

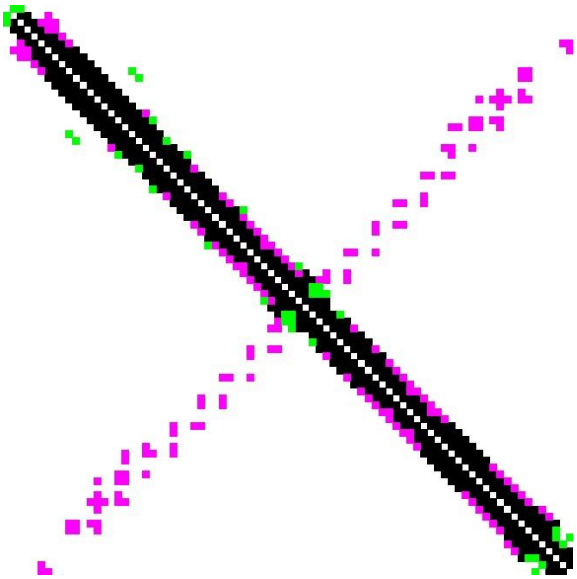

Figura 41. Comparación original vs predicción

Para realizar el análisis de esta predicción se superponen los mapas de contacto original y predicho, de ello se concluye lo siguiente:

- La zona más cercana a la diagonal principal es aquella que mayor aglomeración de contactos predichos correctamente contiene y están representados en color negro, de los 360 contactos originales se logró predecir 268 de manera correcta alcanzando una superposición del 71,08%.
- En fucsia se observa la cantidad de contactos que pertenecen al mapa de contacto original y que no fueron predichos por el modelo, alcanzando la suma de 92 contactos.
- En verde se evidencian los contactos que el modelo predijo, pero no son contactos en el mapa de contacto original, alcanzando 17 contactos.

Proteína: 3wmi

Corte: 10A

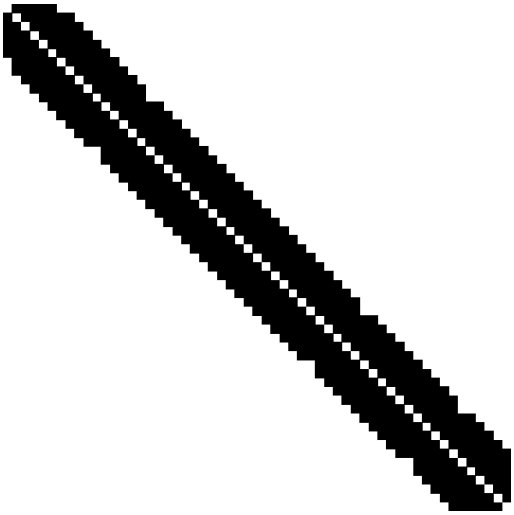

Figura 42. Mapa de contacto original

Figura 43. Predicción mapa de contacto

Figura 44. Comparación original vs predicción

- Para esta proteína con corte 10A se puede evidenciar la gran cantidad de contactos correctamente predichos en la zona central cercana a la diagonal principal, como resultado de un total de 317 contactos originales, se predijeron 227 de manera correcta, alcanzando una superposición de 71,38%.
- En fucsia se observa la cantidad de contactos que pertenecen al mapa de contacto original y que no fueron predichos por el modelo, alcanzando la suma de 90 contactos.
- En verde se evidencian los contactos que el modelo predijo, pero no son contactos en el mapa de contacto original, alcanzando 1 contacto.

Proteína: 4tsh

Corte: 15ª

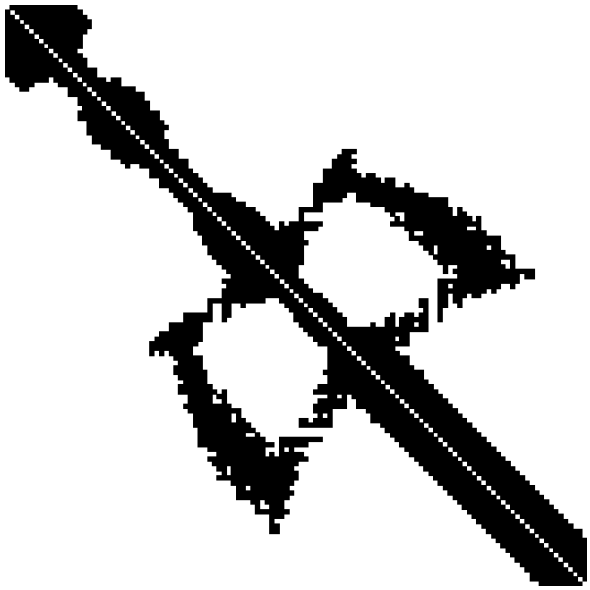

Figura 45. Mapa contacto original

Figura 46. Predicción mapa de contacto

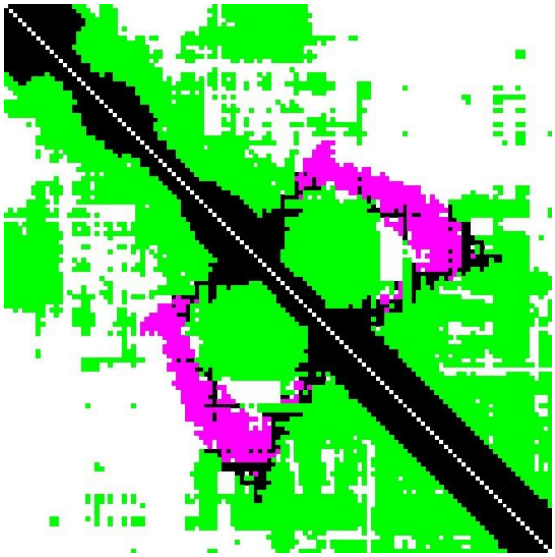

Figura 47. Comparación original vs predicción

- Para esta proteína con corte 15A se puede evidenciar la gran cantidad de ruido en toda la gráfica, como resultado de un total de 438 contactos originales, se predijeron 349 de manera correcta, alcanzando una superposición de 73,16%.
- En fucsia se observa la cantidad de contactos que pertenecen al mapa de contacto original y que no fueron predichos por el modelo, alcanzando la suma de contactos.
- En verde se evidencian los contactos que el modelo predijo, pero no son contactos en el mapa de contacto original, alcanzando 1 contacto.

7.6 COMPARACION METODOS CASP

Antes de comparar los resultados con otros modelos se resalta que las predicciones obtenidas en nuestro modelo se obtuvieron con objetivos diferentes a los utilizados en los predictores tops en el ranking de CASP14, además nuestro modelo fue entrenado para aprender cuatro tipos de clases según el corte de distancia para contactos el cual se conforma por los siguientes rangos: clase 1 = (0,8], clase 2 = (8,10], clase 3 = (10,15] y clase 4 =(15,+∞) mientras que los métodos de CASP14 utilizan rangos de corte más pequeños [75].

Basados en la puntuación F1, que relaciona las métricas de recall y precisión, el modelo implementado presento una puntuación de 47.75% en promedio, superando a los mejores métodos de CASP14, donde (tFold-CaT_human) mejor método tiene una puntuación F1 de 41.15%, seguido por EMAP_CHAE y tFold-IDT_human que se encuentra en un 39.398% y 39.374% respectivamente [76].

CAPITULO 8

8. CONCLUSIONES Y TRABAJO FUTURO

8.1. CONCLUSIONES

Este documento propone un modelo Deep Learning basado en bloques Resnet para la predicción del mapa de contacto de n clases, un tema que sigue siendo complejo de estudiar y al que las redes neuronales como alternativa intentan dar solución.

- Como conclusión se evidencia que, a partir de la investigación y los resultados generados en este proyecto, los modelos basados en Deep Learning son en efecto una alternativa prometedora en la predicción y clasificación de mapas de contacto de proteínas de n clases.
- Las características seleccionadas para el entrenamiento y validación del modelo propuesto son efectivas en la predicción de contactos entre aminoácidos posicionalmente cercanos dentro de la secuencia de cada proteína.
- Este tipo de implementaciones requieren de gran cantidad de recursos computacionales tanto de software como hardware, entre los más importantes se encuentran GPU, espacio en disco del cual se requirió un aproximado de 100 GB, procesador, memoria Ram especialmente para realizar el entrenamiento se necesitó de mínimo 25 GB, que influyen en tiempo y capacidad de almacenamiento, esto debido a la gran cantidad de información que debe ser procesada y almacenada.
- La selección de características es fundamental para el tipo de contacto que se quiere predecir, se debe tener un amplio conocimiento de las relaciones físico-químicas que existen entre los aminoácidos para poder determinar cuáles influyen más en un determinado contacto.
- Durante la evaluación del modelo de deep learning propuesto se obtuvo un promedio para 3 clases de 48.29% de efectividad en la predicción de nuestro modelo para el conjunto de datos de evaluación, es de enfatizar que este conjunto de datos es diferente a los utilizados en las competencias CASP13 y CASP14 por lo que puede verse contrarrestado el resultado teniendo en cuenta que nuestros objetivos tienen menor dificultad.

8.2. TRABAJO FUTURO

- Realizar una investigación profunda para determinar las características que influyen en los contactos generados entre aminoácidos que se encuentran a mayor distancia posicional dentro de la secuencia.
- Utilizar de manera experimental la matriz de puntuación específica de la posición para determinar el valor que representan las ventanas respecto a toda la proteína y estudiar su impacto.
- Generar un dataset con distribuciones de clases más homogéneo y balanceado.
- Una vez aplicadas estas modificaciones y mejoras en el modelo, se pretende participar en la competencia CASP, utilizando los mismos objetivos y métricas de validación y ver una comparación más exacta frente a otros predictores actuales.

REFERENCIAS BIBLIOGRAFICAS

9. REFERENCIAS BIBLIOGRAFICAS

[1] Bruce Alberts, «Celulas y Genomas», en *Biología molecular de la célula*, 5.ª ed., Casa del libro, 2010, p. 1. Accedido: 5 de marzo de 2021. [En línea]. Disponible en: https://www.casadellibro.com/libro-biologia-molecular-de-la-celula-5-ed/9788428215077/1774956

[2] «Célula», *Diccionario de cáncer del NCI*, 2011. https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/celula (accedido 29 de marzo de 2021).

[3] V. T y V. G, «The origin of eukaryotes: the difference between prokaryotic and eukaryotic cells», *Proc. Biol. Sci.*, vol. 266, n.º 1428, jul. 1999, doi: 10.1098/rspb.1999.0817.

[4] D. Peter Snustad y Simmons, «DNA as genetic material», en *Principles of Genetics*, 6.ª ed., 2012, p. 6. Accedido: 5 de marzo de 2021. [En línea]. Disponible en: https://books.google.com/books/about/Principles_of_Genetics.html?hl=es&id=27-dPh4V0ccC

[5] Mercedes Vázquez y Pedro P. García Luna, «Proteínas en nutrición artificial», p. 18, 2005.

[6] Dolores Corella y Jose M. Ordovas, «Conceptos básicos en biología molecular relacionados con la genética y la epigenética», *Rev. Esp. Cardiol.*, vol. 70, n.º 9, pp. 744-753, sep. 2017, doi: 10.1016/j.recesp.2017.02.034.

[7] M. Mardarás, V. V. Corbacho, L. D. Galotti, y A. G. Maggi, *Del gen a la proteína*. Buenos Aires: Ministerio de Educación de la Nación, 2012. Accedido: 5 de marzo de 2021. [En línea]. Disponible en: http://repositorio.educacion.gov.ar:8080/dspace/handle/123456789/110366

[8] Trudy McKee y James R. McKee, «Aminoacidos, peptidos y proteinas», en *Bioquímica las bases moleculares de la vida*, 5.ª ed., 2003, p. 110. Accedido: 5 de marzo de 2021. [En línea]. Disponible en: https://www.mheducation.es/bioquimica-las-bases-moleculares-de-la-vida-9786071511270-spain

[9] M. V. Luque Guillen, «Estructura y propiedades de las proteínas», Universidad de Valencia, España, 2009. [En línea]. Disponible en: http://www.uv.es/tunon/pdf_doc/proteinas_09.pdf

[10] Julio César Quintana Zaez, Reinaldo Molina Ruiz, Jesús S. Aguilar Ruiz, y Cosme E. Santiesteban Toca, «Evaluación de esquemas de predicción de plegamiento de proteínas», en *ResearchGate*, 2004. [En línea]. Disponible en: https://www.researchgate.net/publication/317913425_Evaluacion_de_esquemas_de_prediccion_de_plegamiento_de_proteinas

[11] Colleen Smith, Michael Lieberman, y Allan Marks, «General characteristics of three-dimensional structure», en *Marks Basic Medical Biochemistry: A Clinical Approach*, 2nd edition., 2003, p. 60. [En línea]. Disponible en: https://www.amazon.com/Marks-Medical-Biochemistry-Clinical-Approach/dp/B0042NJYKI#detailBullets_feature_div

[12] Jeremy M. Berg, John L. Tymoczko, y Lubert Stryer, «Composicion y estructura de las proteinas», en *Biochemistry*, 6ta Edición., New York, USA: W H Freeman, 2007, pp. 34, 46-. Accedido: 5 de marzo de 2021. [En línea]. Disponible en: https://www.ncbi.nlm.nih.gov/books/NBK21154/

[13] W. Kabsch y C. Sander, «Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features», *Biopolymers*, vol. 22, n.º 12, pp. 2577-2637, dic. 1983, doi: 10.1002/bip.360221211.
[14] L. Skipper, «*PROTEINS | Overview*», Second Edition. Oxford: Elsevier, 2005.
[15] B. Kuhlman y P. Bradley, «Advances in protein structure prediction and design», *Nat. Rev. Mol. Cell Biol.*, vol. 20, n.º 11, Art. n.º 11, nov. 2019, doi: 10.1038/s41580-019-0163-x.
[16] H. Deng, Y. Jia, y Y. Zhang, «Protein structure prediction», *Int. J. Mod. Phys. B*, vol. 32, n.º 18, jul. 2018, doi: 10.1142/S021797921840009X.
[17] J. M. Duarte, R. Sathyapriya, H. Stehr, I. Filippis, y M. Lappe, «Optimal contact definition for reconstruction of Contact Maps», *BMC Bioinformatics*, vol. 11, n.º 1, Art. n.º 1, may 2010, doi: 10.1186/1471-2105-11-283.
[18] Z. Li, Y. Lin, A. Elofsson, y Y. Yao, «Protein Contact Map Prediction Based on ResNet and DenseNet», *BioMed Res. Int.*, vol. 2020, p. 12, abr. 2020, doi: https://doi.org/10.1155/2020/7584968.
[19] Juan González y David Pelta, «On Using Fuzzy Contact Maps for Protein Structure Comparison», jun. 2007. doi: 10.1109/FUZZY.2007.4295614.
[20] K. S. Pratt, «Design Patterns for Research Methods: Iterative Field Research», p. 7.
[21] K. Petersen, R. Feldt, S. Mujtaba, y M. Mattsson, «Systematic Mapping Studies in Software Engineering», presentado en 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), jun. 2008. doi: 10.14236/ewic/EASE2008.8.
[22] S. Studer *et al.*, «Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology», mar. 2020, doi: 10.48550/arXiv.2003.05155.
[23] S. Wang, S. Sun, Z. Li, R. Zhang, y J. Xu, «Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model», *PLOS Comput. Biol.*, vol. 13, n.º 1, Art. n.º 1, ene. 2017, doi: 10.1371/journal.pcbi.1005324.
[24] B. Adhikari, J. Hou, y J. Cheng, «DNCON2: improved protein contact prediction using two-level deep convolutional neural networks», *Bioinformatics*, vol. 34, n.º 9, Art. n.º 9, may 2018, doi: 10.1093/bioinformatics/btx781.
[25] I. Triguero, S. del Río, V. López, J. Bacardit, J. M. Benítez, y F. Herrera, «ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem», *Knowl.-Based Syst.*, vol. 87, pp. 69-79, oct. 2015, doi: 10.1016/j.knosys.2015.05.027.
[26] H. Yang, M. Wang, Z. Yu, X. Zhao, y A. Li, «GANcon: Protein Contact Map Prediction With Deep Generative Adversarial Network», *IEEE Access*, vol. 8, pp. 80899-80907, 2020, doi: 10.1109/ACCESS.2020.2991605.
[27] H. B, M. Sm, W. Y, S. Hb, y Z. Y, «NeBcon: protein contact map prediction using neural network training coupled with naïve Bayes classifiers», *Bioinformatics (Oxford, England)*, 8 de enero de 2017. https://pubmed.ncbi.nlm.nih.gov/28369334/
[28] D. T. Jones, T. Singh, T. Kosciolek, y S. Tetchner, «MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins», *Bioinformatics*, vol. 31, n.º 7, pp. 999-1006, abr. 2015, doi: 10.1093/bioinformatics/btu791.
[29] W. Chen, J. Sun, y C. Gao, «Improving Residue-Residue Contacts Prediction from Protein Sequences Using RNN-Based LSTM Network», en *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, jul. 2019, pp. 1-7. doi: 10.1109/ICMLC48188.2019.8949207.
[30] C. Mirabello y B. Wallner, «rawMSA: End-to-end Deep Learning using raw Multiple Sequence Alignments», *PLOS ONE*, vol. 14, n.º 8, Art. n.º 8, ago. 2019, doi: 10.1371/journal.pone.0220182.

[31] J. Yang y H.-B. Shen, «MemBrain-contact 2.0: a new two-stage machine learning model for the prediction enhancement of transmembrane protein residue contacts in the full chain», *Bioinformatics*, vol. 34, n.º 2, Art. n.º 2, ene. 2018, doi: 10.1093/bioinformatics/btx593.
[32] Q. Wu, Z. Peng, I. Anishchenko, Q. Cong, D. Baker, y J. Yang, «Protein contact prediction using metagenome sequence data and residual neural networks», *Bioinformatics*, vol. 36, n.º 1, pp. 41-48, ene. 2020, doi: 10.1093/bioinformatics/btz477.
[33] S. Wang, W. Li, R. Zhang, S. Liu, y J. Xu, «CoinFold: a web server for protein contact prediction and contact-assisted protein folding», *Nucleic Acids Res.*, vol. 44, n.º W1, pp. W361-W366, jul. 2016, doi: 10.1093/nar/gkw307.
[34] «Protein Data Bank - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Protein_Data_Bank (accedido 6 de julio de 2022).
[35] «Biopython · Biopython». https://biopython.org/ (accedido 6 de julio de 2022).
[36] J. Wang *et al.*, «POSSUM: a bioinformatics toolkit for generating numerical sequence feature descriptors based on PSSM profiles», *Bioinformatics*, vol. 33, n.º 17, pp. 2756-2758, sep. 2017, doi: 10.1093/bioinformatics/btx302.
[37] *Protein Sequence Analysis*. Academic Press, 2010, pp. 29-62. doi: 10.1016/B978-8-1312-2297-3.50002-3.
[38] M. H, L. Zh, Z. Y, y L. Sz, «A new set of amino acid descriptors and its application in peptide QSARs», *Biopolymers*, vol. 80, n.º 6, 2005, doi: 10.1002/bip.20296.
[39] B. Adhikari y J. Cheng, «Protein Residue Contacts and Prediction Methods», *Data Min. Tech. Life Sci.*, pp. 463-476, 2016, doi: 10.1007/978-1-4939-3572-7_24.
[40] C. Charry-Ceballos y Ó. Bedoya-Leiva, «Predicción estructural de proteínas usando técnicas de clasificación», *Rev. UIS Ing.*, vol. 17, n.º 2, pp. 75-85, 2018.
[41] M. Abadi *et al.*, «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems», mar. 2016, doi: 10.48550/arXiv.1603.04467.
[42] J. L. Sarmiento-Ramos, «Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica», *Rev. UIS Ing.*, vol. 19, n.º 4, pp. 1-18, jun. 2020, doi: 10.18273/revuin.v19n4-2020001.
[43] «What is Deep Learning? | IBM». https://www.ibm.com/cloud/learn/deep-learning (accedido 17 de agosto de 2022).
[44] «Libro APRENDIZAJE PROFUNDO de Gonzalo Pajares y Otros - RC Libros». https://rclibros.es/producto/aprendizaje-profundo/ (accedido 25 de julio de 2022).
[45] «Dividir los datos en datos de formación y evaluación - Amazon Machine Learning». https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html (accedido 4 de agosto de 2022).
[46] «tf.keras.layers.Layer | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer (accedido 8 de agosto de 2022).
[47] T. rédac, «Convolutional Neural Network - Deep Learning - DataScientest», *Formation Data Science | DataScientest.com*, 16 de diciembre de 2021. https://datascientest.com/es/convolutional-neural-network-es (accedido 9 de agosto de 2022).
[48] P. Ganesh, «Types of Convolution Kernels : Simplified», *Medium*, 18 de octubre de 2019. https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37 (accedido 11 de agosto de 2022).
[49] «tf.keras.layers.Conv2D | TensorFlow v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D (accedido 7 de septiembre de 2022).

[50] «tf.keras.layers.BatchNormalization | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization (accedido 14 de agosto de 2022).
[51] «tf.keras.layers.ReLU | TensorFlow Core v2.9.1». https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU (accedido 14 de agosto de 2022).
[52] «tf.keras.layers.Add | TensorFlow Core v2.9.1». https://www.tensorflow.org/api_docs/python/tf/keras/layers/Add (accedido 14 de agosto de 2022).
[53] «tf.keras.layers.Concatenate | TensorFlow Core v2.9.1». https://www.tensorflow.org/api_docs/python/tf/keras/layers/Concatenate (accedido 14 de agosto de 2022).
[54] «tf.keras.layers.MaxPool2D | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D (accedido 15 de agosto de 2022).
[55] «tf.keras.layers.Flatten | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten (accedido 15 de agosto de 2022).
[56] «tf.keras.layers.Dense | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense (accedido 15 de agosto de 2022).
[57] «tf.keras.activations.softmax | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/activations/softmax (accedido 16 de agosto de 2022).
[58] «tf.keras.optimizers.Adam | TensorFlow Core v2.9.1», *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (accedido 17 de agosto de 2022).
[59] K. Team, «Keras documentation: Probabilistic losses». https://keras.io/api/losses/probabilistic_losses/ (accedido 17 de agosto de 2022).
[60] K. Team, «Keras documentation: Classification metrics based on True/False positives & negatives». https://keras.io/api/metrics/classification_metrics/ (accedido 17 de agosto de 2022).
[61] K. Team, «Keras documentation: Model training APIs». https://keras.io/api/models/model_training_apis/#testonbatch-method (accedido 17 de agosto de 2022).
[62] «Why Python is Good for Machine Learning», *Engineering Education (EngEd) Program | Section*. https://www.section.io/engineering-education/why-python-is-good-for-machine-learning/ (accedido 18 de agosto de 2022).
[63] M. Kofler, «Deep Learning with Tensorflow: Part 1 — theory and setup», *Medium*, 17 de agosto de 2017. https://towardsdatascience.com/deep-learning-with-tensorflow-part-1-b19ce7803428 (accedido 18 de agosto de 2022).
[64] «itertools — Functions creating iterators for efficient looping — Python 3.10.6 documentation». https://docs.python.org/3/library/itertools.html (accedido 18 de agosto de 2022).
[65] «Paquete de péptidos - RDocumentación». https://www.rdocumentation.org/packages/Peptides/versions/2.4.4 (accedido 18 de agosto de 2022).
[66] «Google Colab». https://research.google.com/colaboratory/intl/es/faq.html (accedido 21 de agosto de 2022).

[67] «Qué es GitHub y cómo usarlo para aprovechar sus beneficios», *Platzi*. https://platzi.com/blog/que-es-github-como-funciona/ (accedido 21 de agosto de 2022).
[68] «Los pickles de Python. Programación en Castellano.» https://programacion.net/articulo/los_pickles_de_python_1860 (accedido 23 de agosto de 2022).
[69] «CMView - Visualización y análisis de mapas de contacto de proteínas». http://www.bioinformatics.org/cmview/index.html (accedido 23 de agosto de 2022).
[70] «Home - Prediction Center». https://predictioncenter.org/index.cgi (accedido 25 de agosto de 2022).
[71] E. Grudemi, «Moda estadística - ¿Qué es?, tipos, ejemplos y más», *Enciclopedia Económica*, 22 de junio de 2021. https://enciclopediaeconomica.com/moda-estadistica/ (accedido 26 de agosto de 2022).
[72] E. Grudemi, «Media aritmética - ¿Qué es?, usos, ¿cómo calcularla? y más», *Enciclopedia Económica*, 24 de febrero de 2021. https://enciclopediaeconomica.com/media-aritmetica/ (accedido 26 de agosto de 2022).
[73] «¿Que es la mediana y como calcularla? | Superprof», *Material Didáctico - Superprof*. https://www.superprof.es/apuntes/escolar/matematicas/estadistica/descriptiva/mediana.html/ (accedido 26 de agosto de 2022).
[74] «What Is Tanimoto coefficient». http://biotech.fyicenter.com/1000134_What_Is_Tanimoto_coefficient.html (accedido 7 de septiembre de 2022).
[75] V. Ruiz-Serra, C. Pontes, E. Milanetti, A. Kryshtafovych, R. Lepore, y A. Valencia, «Assessing the accuracy of contact and distance predictions in CASP14», *Proteins Struct. Funct. Bioinforma.*, vol. 89, n.º 12, pp. 1888-1900, dic. 2021, doi: 10.1002/prot.26248.
[76] «RR Results - CASP14». https://www.predictioncenter.org/casp14/rrc_avrg_results.cgi (accedido 29 de agosto de 2022).