

UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

División de Tecnologías para la Integración Ciber-Humana

Maestría en Ciencias en Bioingeniería y Cómputo Inteligente



“Asociación por aprendizaje profundo de k-meros genómicos con etiquetas ontológicas”

Tesis que para obtener el grado de
MAESTRA EN CIENCIAS EN BIOINGENIERÍA Y CÓMPUTO
INTELIGENTE

Presenta:
Ing. Diana Laura González Ríos

Director de Tesis
Dr. J. Alejandro Morales Valencia

Co-Directora de Tesis
Dra. Adriana Patricia Mendizabal Ruiz

Guadalajara, Jalisco, Febrero de 2023



C. Diana Laura González Ríos
P r e s e n t e

Por medio de la presente me permito comunicarle que fue aceptado por la Junta Académica correspondiente, el tema de tesis solicitado a esta Coordinación el día 8 de abril de 2022, bajo el título:

“Asociación por aprendizaje profundo de K-meros genómicos con etiquetas ontológicas”

mismo que usted desarrollará, con objeto de dar lugar a los trámites conducentes a la obtención de grado de:

Maestra en Ciencias en Bioingeniería y Cómputo Inteligente

Así mismo le comunico que para el desarrollo de la citada tesis le ha sido designado como **Director al Dr. José Alejandro Morales Valencia** y como **Co-Directora a la Dra. Adriana Patricia Mendizabal Ruiz**.

Sin otro particular de momento, aprovecho la ocasión para enviarle un cordial saludo.

A t e n t a m e n t e
“Piensa y Trabaja”

**“2022, Guadalajara, hogar de la Feria Internacional del Libro y
Capital Mundial del Libro”**

Guadalajara, Jal., 25 de abril de 2022

COLEGIO UNIVERSITARIO DE
EXACTAS E INGENIERÍAS
COORDINACIÓN
DE PROGRAMAS
DOCENTES



**Dr. Luis Guillermo Guerrero Ramírez
Coordinador de Programas Docentes**

Registro 045/2022

LGGR/sijo

DRA. MARÍA TERESA ROMERO GUTIÉRREZ
JEFA DE LA UNIDAD DE POSGRADOS
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS
P R E S E N T E

Por este conducto, los abajo firmantes declaramos que hemos revisado el documento de tesis titulado:

Asociación por aprendizaje profundo de K-meros genómicos con etiquetas ontológicas

Y desarrollado por:

Diana Laura González Ríos

Consideramos que se han atendido las observaciones a satisfacción de todos y que reúne los elementos de formato, calidad, rigor y originalidad necesarios para su impresión.

ATENTAMENTE
“PIENSA Y TRABAJA”

2022, GUADALAJARA, HOGAR DE LA FERIA INTERNACIONAL DEL LIBRO Y CAPITAL MUNDIAL DEL LIBRO
Guadalajara, Jalisco, a domingo, 04 de diciembre de 2022


DR. JOSE ALEJANDRO MORALES VALENCIA
Director


DRA. ADRIANA PATRICIA MENDIZABAL RUIZ
Lector Revisor


DR. OMAR PAREDES
Lector Revisor


DR. ERNESTO TORRADO CARBAJAL
Lector Revisor



CUCEI/MCBCI/273/2022

DRA. MARÍA TERESA ROMERO GUTIÉRREZ

JEFA DE LA UNIDAD DE POSGRADOS

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

PRESENTE

Por este medio, los miembros de la Junta Académica de la Maestría en Ciencias en Bioingeniería y Cómputo Inteligente declaramos que, después de haber revisado la tesis que lleva por título:

Asociación por aprendizaje profundo de K-meros genómicos con etiquetas ontológicas

Que presenta la alumna DIANA LAURA GONZÁLEZ Ríos, bajo la dirección del DR. JOSÉ ALEJANDRO MORALES VALENCIA, consideramos que **la disertación es original, rigurosa y de calidad.**

Después de revisar que el alumno ha reunido los demás requisitos de egreso marcados en el Dictamen de Creación de este programa de maestría, avalamos la declaración de **impresión del documento de Tesis** emitida por los Lectores Revisores, para que el alumno proceda después a defenderla.

ATENTAMENTE

“PIENSA Y TRABAJA”

2022, GUADALAJARA, HOGAR DE LA FERIA INTERNACIONAL DEL LIBRO Y CAPITAL MUNDIAL DEL LIBRO

Guadalajara, Jalisco, a lunes, 05 de diciembre de 2022

Junta Académica de la
Maestría en Ciencias en Bioingeniería y Cómputo Inteligente

Dra. Adriana Patricia Mendizabal Ruiz

Dra. Flor del Carmen Rodríguez
Gómez

Ricardo A. Salido L.

Dr. Ricardo Antonio Salido Ruiz

Clayton

Dr. Hugo Abraham Vélez Pérez

Adolfo Flores S. Farías

Dr. Adolfo Flores-Saiffe Farías

Francisco Javier Álvarez Padilla

(Coordinador)



¿Sabes de qué están hechos los sueños?

Hechos? Sólo son sueños.

*No. No lo son. La gente cree que no son reales porque no son materia, partículas.
Son reales. Están hechos de puntos de vista, imágenes, recuerdos, juegos de palabras
y esperanzas perdidas...*

"THE SANDMAN", NEIL GAIMAN

Agradecimientos

Agradezco a mi familia, quienes se esforzaron toda mi vida día con día para darme educación y los recursos que necesitaba para formarme profesionalmente, así como los esfuerzos que hacen hoy en día para cuidarme y demostrarme su cariño, y que hoy en día continúan enseñándome y expresándome su cariño.

Agradezco a mis amistades viejas y nuevas, por acompañarme y levantarme el ánimo, cuidarme y apoyarme como una segunda familia, y estar cerca cada que necesité el abrazo y las risas.

Agradezco a mi psicóloga, que con su trabajo me ha ayudado a mantener claras mis prioridades y me ha enseñado que en mi vida tengo opciones, y que gracias a eso he podido direccionar mi vida por el camino que he escogido para mí.

Agradezco a mis profesores por dedicar atención a mis preguntas y fomentar mi desarrollo en las áreas de mi interés, así como las que en su momento no lo fueron tanto pero que aún así me permitieron abrir mi panorama de lo que era interesante y divertido en el mundo.

Agradezco a mis asesores por dedicar atención a mi proyecto y a mí misma, quienes en su momento se encargaron de romper mi forma de pensamiento y me enseñaron a entender las cosas de manera distinta, mostrándome no sólo lo que puedo ser sino lo que ya soy.

Agradezco el apoyo de mis compañeros de laboratorio, mentores e investigadores con los que conviví, con los que pude dialogar mis ideas y aprender de puntos de vista distintos, abriendo mi panorama de conocimientos y ampliando mi perspectiva de investigación.

Agradezco a mi pareja, quien por un tiempo fue amigo y que ahora además de

mostrarme su cariño y atención, me motiva y apoya para continuar mi carrera profesional, acompañándome en momentos complicados y situaciones divertidas.

Agradezco al universo, por permitirme vivir y experimentar más cosas antes de irme, por enseñarme cada día que mi camino está hecho por mí pero también por otros, y que mis decisiones tienen impacto y significado en el mundo.

Me agradezco a mí misma por no rendirme, por cuidarme y enfocarme no solo en la meta sino en el camino, por saber cosas y aprender nuevas y permitirme aceptar mis errores.

Finalmente, reconozco y agradezco al programa de Maestría en Ciencias en Bioingeniería y Cómputo Inteligente de la Universidad de Guadalajara por permitirme realizar mis estudios de posgrado; así como al CONACyT por el apoyo recibido con la No. CVU: 1103063.

Índice general

Agradecimientos	II
Índice de figuras	VI
Resumen	X
Abstract	XI
1. Introducción	1
1.1. Hipótesis	3
1.2. Objetivos	3
1.2.1. Objetivo general	3
1.2.2. Objetivos particulares	3
1.3. Metodología	3
1.3.1. Descripción general	3
1.3.2. Base de datos	4
1.3.3. Modelo de Deep Learning	8
1.3.4. Clasificación de kmeros	10
2. Antecedentes	11
2.1. Alineamiento de secuencias	12
2.2. Herramientas de procesamiento de secuencias de DNA	17
3. Resultados y Discusión	19
3.1. Base de datos	19

3.1.1. Retrotraducción y Generación de vectores de kmeros	21
3.1.2. Estadística de etiquetas ontológicas	23
3.1.3. Vectores de Ontologías	24
3.2. Modelos	24
3.2.1. CNN	26
3.2.2. RNN	39
3.2.3. Testing	40
3.3. Clasificación de kmeros	51
4. Conclusión	52
5. Perspectiva a futuro	53
Referencias	55

Índice de figuras

1.1.	Diagrama de Metodología de proyecto.	4
2.1.	Diagrama de ancestros de la ontología GO:0060491. Recuperada de [24] el 16 de Noviembre del 2022.	17
3.1.	Diagrama de Generación de kmeros a partir de secuencia de proteína.	22
3.2.	Histograma de frecuencia de ontologías de <i>E. coli</i>	23
3.3.	Arquitectura inicial del modelo CNN A como modelo simple para procesar la información.	28
3.4.	Historial de red neuronal convolucional A para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.	30
3.5.	Arquitectura del modelo CNN B de dos bloques de convolución. . . .	31
3.6.	Historial de red neuronal convolucional B para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.	32
3.7.	Arquitectura CNN C basada en modelo DanQ y NCNet. Combinación de arquitectura de una capa convolucional y una capa Bidirectional LSTM RNN.	33

3.8. Historial de red neuronal convolucional C para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.	34
3.9. Arquitectura de modelo CNN D. Se utilizan capas de convolución 1D con memoria junto una capa GRU como un segundo ejercicio de combinación con arquitectura de tipo recurrente, enfocando la memoria a la capa convolucional.	35
3.10. Historial de red neuronal convolucional D para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.	36
3.11. Historial de medias y medianas de modelos convolucionales en sus tres versiones.	37
3.12. Resultados de entrenamiento largo de los cuatro modelos convolucionales en sus terceras versiones.	38
3.13. Arquitectura de RNN A con dos capas LSTM.	39
3.14. Historial de rendimiento de RNN A. 1) Historial de modelo con valores de media y mediana de valores de loss y AUC ROC en entrenamiento y validación. 2) Historial de entrenamiento prolongado de este modelo.	40
3.15. Comparativa de desempeño en testing de modelos CNN A, CNN B, CNN C, CNN D y RNN A con misma muestra de datos. Se muestran valores de Loss y de AUC ROC en la tabla del lado derecho.	41
3.16. Matriz de confusión de modelo CNN A. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.	42

3.17. Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN A.	42
3.18. Matriz de confusión de modelo CNN B. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.	43
3.19. Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN B.	43
3.20. Matriz de confusión de modelo CNN C. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.	44
3.21. Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN C.	44
3.22. Matriz de confusión de modelo CNN D. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.	45
3.23. Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN D.	45
3.24. Matriz de confusión de modelo RNN A. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.	46
3.25. Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo RNN A.	46

3.26. Diagrama de ancestros de la ontología GO:0016020. Recuperada de QuickGO[74] el 15 de Noviembre del 2022.	48
3.27. Diagrama de ancestros de la ontología GO:005886. Recuperada de QuickGO[73] el 16 de Noviembre del 2022.	49

Resumen

La información que describe a un ser vivo puede ser estudiada desde distintos ángulos. Se ha analizado la información en torno a las proteínas durante mucho tiempo, y esto ha permitido un entendimiento parcial en la forma en la que un organismo vive. El DNA ha demostrado ser más que secuencias que codifican a proteínas, y el entendimiento a profundidad de esta macromolécula es fundamental para el entendimiento de la vida y la evolución. En el presente trabajo se lleva a cabo un esfuerzo por avanzar hacia este entendimiento, aprovechando la información generada en bases de datos actuales, utilizando herramientas de aprendizaje profundo para analizar y encontrar las palabras de DNA que describen función y actividad biológica. En total se desarrollaron 5 modelos de redes neuronales distintos para analizar secuencias de DNA y el desempeño de éstas permite afirmar que hay kmeros estadísticamente representativos de funciones, lo que permitiría desarrollar experimentos donde se pueden configurar secuencias de DNA para tener ciertas funcionalidades, además de el entendimiento de la distribución de estas funciones y la *gramática* del DNA para distintos seres vivos.

Abstract

The information that describes a living being can be studied from different angles. Information about proteins has been analyzed for a long time, allowing a partial understanding of how an organism lives. DNA has been shown to be more than protein-coding sequences, and in-depth knowledge of this macromolecule is fundamental to understanding life and evolution. In the present work, an effort is carried out to advance this understanding, taking advantage of the information generated in current databases, and using deep learning tools to analyze and find the DNA words that describe biological function and activity. In total, 5 different neural network models were developed to analyze DNA sequences. Their performance allows us to affirm that there are statistically representative kmers of functions, which would allow the development of experiments where DNA sequences can be configured to have certain functionalities, in addition to the understanding of the distribution of these functions and the *grammar* of DNA for different living beings.

Capítulo 1

Introducción

Las etiquetas ontológicas de Gene Ontology son una representación de las características funcionales de los productos de los genes de un organismo. Estas etiquetas se dividen en 3 categorías, describiendo los procesos biológicos en los que participa el producto, las funciones moleculares que es capaz de ejecutar el producto y la locación de la célula en la que estos fenómenos ocurren[24]. En conjunto las etiquetas nos permiten entender la funcionalidad del producto génico en cuestión, y esta información es obtenida a través del análisis y observación de la actividad de los productos génicos de diferentes organismos y la asociación de estas etiquetas a otros productos basados en su similitud, siendo un ejemplo muy popular BLAST que basa la asociación de información entre biomoléculas a partir de la similitud de sus secuencias[4], limitando este enriquecimiento informático solo a secuencias de organismos cuya distancia genética es corta.

Sin embargo, las etiquetas no son exclusivas de una secuencia de producto específica, las mismas etiquetas que describen la actividad de un producto pueden aparecer en un producto distante tanto en secuencia como en distancia genética, además de que se ha encontrado que secciones de DNA ajenas a los genes son funcionales[20], y estas funcionalidades no son asociables con la información obtenida de los productos génicos. Esto nos abre la entrada a trabajos como DanQ[20] donde se analiza la secuencia de secciones de DNA no codificante por medio de algoritmos de deep learning para encontrar patrones a los que se asocian funcionalidades que influyen en la apa-

rición de enfermedades en humanos procesando la información a manera de imagen, o trabajos como TANGO[57] donde utilizan estas etiquetas ontológicas para asociar la descripción funcional de una proteína a otra a través de la similitud semántica de sus secuencias, o ONTOPROTEIN[58] donde utilizan la secuencia de la proteína para predecir la funcionalidad asociada a secciones de proteína del mismo modo en el que se predice la funcionalidad de una palabra en un texto a través de su relación con otras palabras, en este caso secciones de proteínas, y la frecuencia de uso de tal fragmento asociado a las etiquetas ontológicas.

Aunque las funciones de un producto génico pueden obtenerse del análisis de las secciones funcionales del producto que éste codifique, las etiquetas ontológicas finalmente son asociadas a la secuencia completa del gen, dejando de lado que las funcionalidades presentes en el gen también pueden analizarse por medio de secciones de su secuencia de DNA, modificando el enfoque de asociación de funcionalidad por alineamiento de secuencias para la identificación de dominios[32], siendo un método que ha mostrado ser muy variable, y con frecuencia no es posible identificar patrones estadísticos de cambio entre dichas secuencias. Por esto se ha concluido que el alineamiento no es la herramienta para la identificación de dominios o para la atribución de funciones de los genes más adecuada.

Hoy en día existen técnicas que permiten encontrar más y mejores asociaciones a funciones dentro de secuencias de genes que no se basan en el alineamiento y búsqueda de dominios como las técnicas de Machine Learning[41, 34]. En este trabajo propongo desarrollar una herramienta de deep learning que permita atribuir etiquetas ontológicas a segmentos de secuencias génicas. Este proyecto es elaborado en el Laboratorio de Bioinformática de CUCEI, utilizando una computadora apta para el trabajo de un modelo de deep learning capaz de procesar secuencias génicas y asociarlas a etiquetas ontológicas. Se utilizó una base de datos de proteínas del organismo *Escherichia coli K 12* cuyas anotaciones funcionales fueron obtenidas por métodos experimentales, además de un modelo de deep learning desarrollado específicamente para este problema en el lenguaje de programación Python.

1.1. Hipótesis

Existen fragmentos de genes significativamente asociados a sus etiquetas ontológicas.

1.2. Objetivos

1.2.1. Objetivo general

Relacionar las etiquetas ontológicas con subsecuencias de genes.

1.2.2. Objetivos particulares

1. Obtener un modelo que asocie etiquetas ontológicas con k-meros.
2. Crear un conjunto de datos utilizable en la agrupación de kmeros por funciones.

1.3. Metodología

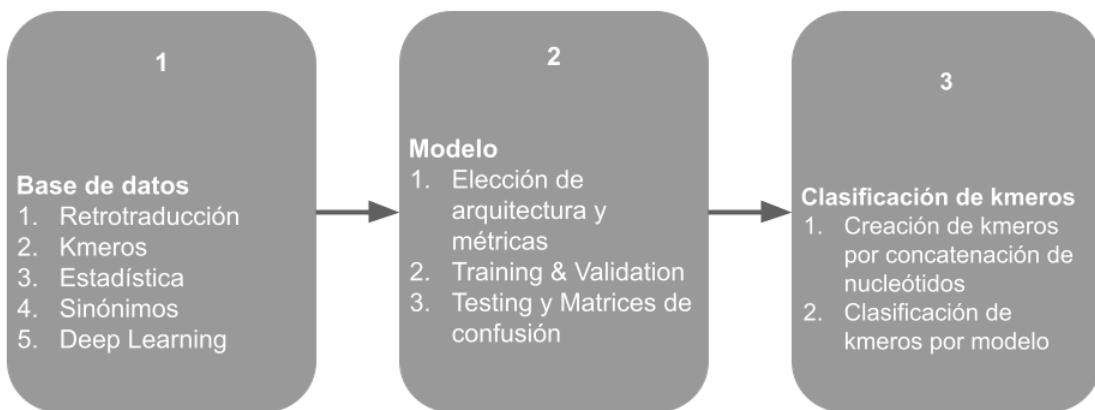
1.3.1. Descripción general

Para este proyecto se dividió la parte experimental en tres etapas descritas a continuación. La metodología puede observarse en la Figura 1.1:

1. Obtención y procesado de base de datos. Donde la base de datos seleccionada será analizada en su contenido para desarrollar algoritmos de procesamiento específico en Python, generando datos analizables por las redes neuronales y analizando la estadística de su contenido para la posterior discusión de sus datos.
2. Arquitecturas y diseño de modelos de redes neuronales. En esta etapa se utilizará la información del Capítulo 2 y las estadísticas de la primera etapa experimental para elegir las arquitecturas adecuadas para el procesado de la información.

3. Clasificación de kmeros. Aquí se clasifican los kmeros posibles existentes por combinación en cada modelo para generar bases de datos que listan las probabilidades de pertenencia de cada kmero en cada clase.

Figura 1.1: Diagrama de Metodología de proyecto.



1.3.2. Base de datos

Se ha seleccionado para trabajar una base de datos de libre acceso del consorcio de Gene Ontology[62], específicamente la perteneciente a la Enciclopedia de metabolismo de E. coli[64] debido a que la información contenida en este archivo fue obtenida por medios experimentales[70] con 55232 registros hasta la fecha (23 de Septiembre del 2022), de cuyo contenido se tomó la base de datos de origen de cada registro, el ID de registro de Uniprot[37] de donde se obtiene la secuencia proteica, y el ID de la etiqueta ontológica registrada en la base de datos.

Se procesó la información de este archivo para unir todas las anotaciones ontológicas por gen obteniendo un total de 3920 proteínas. Utilizando el ID correspondiente se accedió a las secuencias proteicas de cada registro por medio de un algoritmo en

Python. Las secuencias obtenidas eran producto de los genes de *E. coli*, y se procesó cada secuencia por medio de un proceso de retrotraducción.

Retrotraducción

Existen dos herramientas dentro de los servicios del EMBL[69, 52] relacionados a proteínas que devuelven la traducción inversa de las proteínas. La primera herramienta es **EMBOSS backtranambig** traduce la secuencia de aminoácidos a una única secuencia ambigua de DNA[65], utilizando el FASTA de la proteína y la selección de una tabla de codones de acuerdo a los códigos genéticos existentes[33], con una salida de una secuencia de DNA basada en la traducción inversa más favorable de acuerdo a sus registros[68]. La segunda herramienta es **EMBOSS Backtranseq**[66] que se diferencia de la anterior mostrando como resultado al menos un resultado posible acorde a los registros del EMBL, mostrando secuencias de DNA ya registradas y asociadas a la secuencia de proteína ingresada. Para este trabajo se optó por generar un código que generara todas las secuencias de DNA posibles de acuerdo al código genético estándar que pudieran generar la proteína de la que se extrajo la información funcional. Con esto nos aseguramos de que aún en secuencias de DNA con menor similitud entre ellas es posible observar las mismas funciones dentro de un organismo dado.

Para este proceso se ha analizado la información disponible entre los codones de RNA y su equivalencia en aminoácidos en el proceso de traducción y se han observado hasta 33 códigos genéticos distintos que contemplan tanto código genético nuclear como mitocondrial[33], donde se muestran diferencias en la equivalencia de codón, cambiando codones de paro por codones codificantes o cambiando la equivalencia codificante de un aminoácido por otro. Estos códigos varían por el organismo en el que se encontró y bajo las condiciones en las que se presenta el reemplazo de equivalencia entre los que se encuentran las bacterias[1]. Como ejemplo en el caso de *E. coli* se presentan casos donde el codón de Lisina es interpretado como Metionina en la posición de inicio de la lectura de la secuencia de RNA en el proceso de traducción [2], lo que abre la posibilidad a cambios en las secuencias posibles de DNA para

Aminoácido	Abreviación	Codones
Alanina	A	GCT, GCC, GCA, GCG
Cisteína	C	TGT, TGC
Ácido Aspártico	D	GAC, GAT
Ácido Glutámico	E	GAA, GAG
Fenilalanina	F	TTT, TTC
Glicina	G	GGT, GGC, GGA, GGG
Histidina	H	CAT, CAC
Isoleucina	I	ATT, ATC, ATA
Lisina	K	AAA, AAG
Leucina	L	TTA, TTG, CTT, CTC, CTA, CTG
Metionina	M	ATG
Asparagina	N	AAC, AAT
Prolina	P	CCT, CCC, CCA, CCG
Glutamina	Q	CAA, CAG
Arginina	R	AGA, AGG, CGA, CGC, CGG, CGT
Serina	S	TCT, TCC, TCA, TCG, AGT, AGC
Treonina	T	ACT, ACC, ACA, ACC
Selenocisteína	U	TGA
Valina	V	GTT, GTC, GTA, GTG
Triptófano	W	TGG
Tirosina	Y	TAT, TAC
Otro glutámico	Z	GAA, GAG

Tabla 1.1: Código genético estándar de DNA.

Organismo	Codon	Traducción estándar	Traducción alternativa
Bacterias, Arqueas y Plástidos de plantas	UUG CUG	L	M
	AUU AUC AUA	I	M
	GUG	V	M
	UGA	STOP	G
División candidata SR1 y Gracilibacterias			

Tabla 1.2: Códigos genéticos alternativos para bacterias. Extraído de Elzanowski y Ostell

la generación de la proteína correspondiente al gen al que se asocian las etiquetas ontológicas.

En el caso de bacterias se han observado cambios en la interpretación del aminoácido que mantiene la naturaleza del mismo y no afecta la estructura de la proteína, como se muestra en la Tabla 1.2, siendo casos muy poco frecuentes[2], por ello se mantiene el uso del código genético estándar que se mostró en la Tabla 1.1 con la excepción de el añadido de un codón para el aminoácido Selenocisteína que se encuentra en la base de datos utilizada y que normalmente es interpretado como un codón de paro.

Kmeros

Al trabajar con secuencias de DNA es importante establecer la longitud de las secuencias que se van a trabajar en el proyecto. En trabajos como NCNet[35] donde se ingresan fragmentos de DNA para determinar funcionalidades en secciones no codificantes, se buscan todos los fragmentos posibles donde puede existir una función previamente determinada, mientras que en trabajos como el de Avsec et al.[43] se ingresa una secuencia buscando dentro de ésta las secciones que corresponden a una funcionalidad específica. Para este efecto se consideró el trabajo de Borrero et al. donde se concluye que en un genoma bacteriano existe una cantidad de información determinada que puede ser distribuida en kmeros de tamaño promedio 11 a 15, siendo el ideal 13 para trabajar DNA de bacterias[36], razón por la que se considera ideal

el uso de kmeros de este tamaño para encontrar los fragmentos génicos que puedan representar la información funcional de cualquier secuencia de DNA bacteriano posible.

Se fragmentaron las posibles secuencias de DNA de cada proteína de la base de datos con una superposición de 1 nucleótido para obtener los kmeros. En total se encontraron 66129041 kmeros únicos, representando un 98.53 % de los kmeros totales posibles por combinación para el tamaño del kmero 13.

La base de datos se preparó para ser utilizada, de manera que tenemos un conjunto de kmeros asociados a conjuntos de ontologías acordes a las proteínas donde aparecieron dichos kmeros, mezclando aleatoriamente el orden en el que aparecen dichos datos pero manteniendo la asociación para alimentar el modelo. Los datos son utilizados en la proporción de 89 % entrenamiento, 9 % testing y 2 % validación.

1.3.3. Modelo de Deep Learning

Analizando los trabajos relacionados y la naturaleza de la información se ha optado por tres variantes distintas de redes neuronales, una CNN, una RNN (red neuronal recurrente por sus siglas en inglés) y una mezcla de ambas como se observó en trabajos como DanQ[20] y NCNet[35] para procesar la información, cada una siendo entrenada usando la base de datos ya mencionada como se explica a continuación.

1. Cada kmero es procesado para generar un vector One Hot Encode (OHE) de entrada, generando un vector de $4*13*1$ para la CNN, uno de $1*52$ para la RNN y para el modelo que usa ambas capas una de $4*13*1$.
2. Para el vector de salida en el entrenamiento, testing y validación se crearon vectores de ontologías donde cada uno es una etiqueta ontológica presente, y cada cero es una etiqueta ontológica ausente, esto para las tres arquitecturas, generando primero un vector de $1*43558$ y más adelante un vector de $1*4206$.
Este cambio se explica más adelante.
3. Para el proceso de testing se generan matrices de confusión a partir de las clases obtenidas en la clasificación del conjunto de kmeros para esta etapa.

Software

Para el desarrollo de los algoritmos de Python se utilizaron los paquetes y librerías listados a continuación con las versiones en las que fueron utilizados:

- Python 3.9.11
- Pip 22.3
- Obonet 0.3.0
- Tensorflow 2.9.1
- Matplotlib 3.5.3
- Scikit-learn 1.1.2
- pandas 1.4.3
- numpy 1.23.2
- mlxtend 0.21.0
- urllib3 1.26.12
- tqdm 4.64.0

Hardware

La computadora donde se procesó este proyecto tiene la siguientes características de Hardware:

- Procesador Ryzen 7 5800X.
- Tarjeta gráfica NVIDIA GeForce RTX 3070.
- 96 GB Memoria RAM.

1.3.4. Clasificación de kmeros

En esta etapa se generó un algoritmo para producir todas los kmeros posibles que podían existir concatenando los 4 nucleótidos en 13 posiciones. Esta lista de kmeros se procesó en cada modelo y se almacenaron los vectores de probabilidad den un archivo y los vectores con la clase de mayor probabilidad en otro.

Capítulo 2

Antecedentes

El DNA es una macromolécula que contiene secciones que pueden codificar un producto. Dentro de los posibles productos de un gen se encuentran las proteínas, que juegan un papel vital en la existencia de todo ser vivo. Las proteínas de acuerdo a sus propiedades bioquímicas generan estructuras a las que se les pueden asociar funciones acorde a su secuencia específica, y es esta secuencia específica la que generalmente ha sido asociada con funcionalidades que podrían tener similitud con las funcionalidades de secuencias similares, y nos da pistas sobre los orígenes ancestrales/generacionales de una proteína en cuestión[25].

Históricamente el estudio de la biología ha ido evolucionando, pasando por la idea de la herencia, el descubrimiento de la estructura del DNA y la creación del dogma central de la biología molecular, con el enfoque a las secuencias de genes y sus productos codificados se llegó a la comprensión de que una cadena peptídica generada a partir de una secuencia de DNA tenía actividad funcional dentro del metabolismo de una célula[5].

Con el análisis de la nomenclatura se llegó a la hipótesis de que a cada gen le correspondía una función, describiendo los genes con nombres y definiciones desde lo simple como "hidrólisis" hasta una definición más particular como ".amilasa", y para finales de los años 50 se desarrolló la Comisión Internacional de Clasificación de Enzimas y el Centro de Información de Secuencias Proteicas de Múnich, donde se leían reportes y artículos sobre el análisis de la funcionalidad de una enzima y se

resumía para englobar de manera general las funciones de la proteína. Este tipo de descripciones se utilizaron durante mucho tiempo, asociando cada gen a una función de manera lineal, y esta idea evolucionó a la concepción de que proteínas similares rastreadas a través de distintas especies conservarían funciones específicas que después se denominaron dominios[32, 5, 23].

2.1. Alineamiento de secuencias

Las proteínas se pueden dividir por secciones de acuerdo a las estructuras resultantes de su plegamiento, secciones que de acuerdo a estos dominios describen actividades específicas que en conjunto se interpretan en al menos una función específica de esta proteína en el organismo en el que se codificó[25].

Los dominios han sido utilizados para asociar funciones de una secuencia de proteínas a otra por medio de métodos como el alineamiento de secuencias. Este método analiza la similitud de dos o más secuencias, comparándolas de manera global o en secciones, puntuando el porcentaje de similitud de ambas y con esto asociando una probabilidad de que las funciones codificadas por una proteína también se presenten en la otra.

Sabiendo que las secuencias pueden o no tener un porcentaje de similitud por si mismas, se han desarrollado distintas herramientas para analizar si las secuencias pueden ser equivalentes, aún con secuencias no tan similares como en análisis de secuencias de organismos que no son subespecies de otras, y con ello mantener la asociación de las funciones de una proteína a otra, utilizando diferentes tácticas para evaluar la similitud de una secuencia a otra y ajustando la puntuación de similitud y las penalizaciones de *no similitud* en las secuencias.

Una estrategia es el uso de gaps, que son espacios insertados en la secuencia de la proteína para mejorar la similitud de las secuencias analizadas. Además de los gaps también están las sustituciones de aminoácidos, donde en vez de calificar si el aminoácido de una secuencia es igual al otro, analiza si la sustitución es equivalente o no a partir de su naturaleza bioquímica. También existen análisis donde se evalúa

la cercanía evolutiva entre una secuencia y otra por medio de la puntuación de la sustitución de un aminoácido dado según la frecuencia registrada de esa sustitución[8].

Estas estrategias son utilizadas en herramientas como BLAST[9] (Basic Local Alignment Search Tool), que dentro de sus parámetros de alineamiento está la disponibilidad de distintos alineamientos, alineando secuencias de proteína a proteína, secuencias de DNA traducidas a proteína, secuencias de proteína a secuencias de DNA traducidas y de secuencias de DNA a secuencias de DNA, así como las herramientas del EMBL[52, 67] que dispone de varias herramientas similares a las de BLAST pero con distintos algoritmos explicados dentro de los repositorios del EMBL. En contraste la herramienta 3D BLAST[6] analiza la sustitución de un aminoácido por otro a partir de la homología de las estructuras resultantes del plegamiento de la proteína ingresada, asociando las funciones a la estructura de las proteínas.

Es importante recalcar que las funciones de los productos de genes no son exclusivas de secuencias específicas, puesto que además del hecho de la disminución de similitud de secuencias por distancia filogenética, también las vías evolutivas de distintos organismos permitieron la generación de genes que cumplían las funciones necesarias para su supervivencia y que compartían funciones con genes de orígenes distintos y secuencias distantes, genes que pueden ser ortólogos o parálogos.

Los genes ortólogos surgen a través de un proceso de especiación, siendo genes que evolucionaron en distintas especies a partir de un ancestro en común y cuya secuencia puede resultar altamente similar, especialmente en alineamientos de secuencias cortas[12]. Los genes parálogos en cambio son genes duplicados dentro de un mismo genoma, permitiendo la mutación de uno de los genes duplicados sin que el organismo pierda la funcionalidad original, facilitando la generación de proteínas nuevas con secuencias distintas pero que mantienen las funcionalidades del gen a partir del que fue generado[11].

Con estos conceptos en mente es fácil notar que una secuencia altamente similar no es lo que conserva la funcionalidad, sino ciertos fragmentos que después de la diferenciación con respecto al gen de origen o ancestro se mantuvieron y conservaron las funcionalidades que permiten al organismo seguir vivo, y a la vez diferenciarse de

otros[42].

Además de estos genes también existe el fenómeno del empalme alternativo, presente en células eucariotas y en virus, en el que los exones transcritos de un mismo gen se ensamblan en un orden distinto al que fueron transcritos, creando variaciones en las proteínas que, además de modificar su estructura, generan cambios en la funcionalidad de la misma, modificando la regulación en la unión entre proteínas, entre proteínas y ácidos nucleicos, y entre proteínas y membranas[10, 5]. Esto sugiere que las funcionalidades del producto están condicionadas a partir de la secuencia nucleica a partir de la que se tradujo, y que las funciones están descritas por medio de fragmentos de DNA dentro del mismo gen.

En épocas más recientes se ha descubierto que las secciones no codificantes de DNA también describen funcionalidades en el organismo en el que se encuentren, excluyendo la idea de que las funcionalidades de los genes son únicamente visibles en los productos codificados. Para el año 2007 se desarrolló el proyecto de la Enciclopedia de los Elementos del ADN con el fin de encontrar todos los elementos funcionales del genoma humano, encontrando funcionalidades en pseudogenes y secciones no codificantes[43, 5]. En estos esfuerzos se ha tenido claro que hay enfermedades humanas que están dadas a partir de un problema en la regulación de expresión de ciertos genes, siendo que esa regulación se encuentra en secciones no codificantes y cuyas alteraciones pueden presentarse por polimorfismos de un solo nucleótido y que precisamente se han ligado con fenotipos y enfermedades[27, 20].

Por lo anterior mencionado es claro que el alineamiento de secuencias no es la herramienta ideal para la asociación de funcionalidades de proteína a proteína, y las herramientas disponibles para el alineamiento de secuencias de DNA tampoco son útiles para esto, ya que la asociación de la información con nucleótidos es menor y por ende mas complicado de diferenciar[71]. Esto remarca la necesidad de otro tipo de análisis den secuencias en los que sea más sólida la asociación de funciones y secuencias, y las etiquetas ontológicas de Gene Ontology han probado ser útiles en distintos análisis.

Gene Ontology

En la investigación y desarrollo del campo de la biología surgió la necesidad de la homogeneización de la información que se generaba con cada publicación, de manera que las descripciones de los genes y las funcionalidades de sus productos fueran fáciles y rápidas de entender. Esta idea llevó a la formación de distintas organizaciones con este fin, y en 1988 se fundó el consorcio de Gene Ontology, que usaba la información de tres organismos modelo: *Drosophila melanogaster*, *Mus musculus*, y *Saccharomyces cerevisiae*[3, 5]. Con el tiempo más organismos se incluyeron en el catálogo de información de Gene Ontology, teniendo hasta el 15 de noviembre del 2022 más de 7.5 millones de anotaciones de genes con los que describen las funcionalidades de los productos codificados de éstos.

Durante el desarrollo de las ontologías de genes surgieron ajustes a la idea de que cada gen codificaba una función, y es que cada producto tiene distintas funcionalidades, además de que la descripción de cada ontología podía ser distinta a partir del tipo de información que se buscaba, por lo que algo que originalmente se buscaba que fuera lineal se convirtió en un grafo acíclico dirigido con distintas relaciones entre sus componentes que permitían enriquecer cada ontología, entendiendo que cada ontología no tenía una sola dirección descriptiva, sino que podía aparecer en más de una descripción funcional dependiendo del tipo de relación con la que se filtrara la información, y siendo no jerárquico ya que cada ontología podía tener más de un origen[29, 3].

Las ontologías de Gene Ontology son etiquetas descriptivas con relación entre ellas a partir de 3 grandes categorías: Proceso biológico (BP por sus siglas en inglés), Función molecular (MF por sus siglas en inglés) y Componente celular (CC por sus siglas en inglés). Cada etiqueta ontológica es descrita en alguna de las categorías mencionadas por medio de relaciones directas y cuya información es actualizada frecuentemente[24]. Algunas relaciones se observan en la Tabla 2.1.

En la Figura 2.1, se muestra el árbol descriptivo de la etiqueta **GO:0060491 regulation of cell projection assembly**, donde podemos observar la relación de esta

Relación	Significado	Ejemplo
is_a	Relación directa de descripción.	La Fusión mitocondrial <i>is_a</i> Fusión de membrana.
part_of	Relación estandar de fraternidad.	El cerebro es <i>part_of</i> de un cuerpo.
derives_from	Un elemento sucede a otro, tal que una porción biológicamente significativa de la entidad anterior es heredada por la segunda.	Un cigoto <i>derives_from</i> un esperma y un óvulo.
has_participant	Enlaza procesos con entidades que participan en ellos.	El proceso de apoptosis <i>has_participant</i> en una célula.
has_function	Enlaza entidades materiales con sus funciones.	Una enzima <i>has_function</i> de catálisis de un tipo de reacción específica.
regulates	Describe el caso en el que un proceso afecta directamente la manifestación de otro proceso o cualidad	La regulación de proceso celular <i>regulates</i> los procesos celulares

Tabla 2.1: Algunos tipos de relaciones lógicas de etiquetas de Gene Ontology[24, 72].

etiqueta con otras mediante el uso de conectores *is_a*, *part_of* y *regulates*. Es notable que a pesar de se está mostrando el diagrama de ancestros de una etiqueta ya es visible que el significado o descripción de una etiqueta no es ni lineal ni único. Existen proteínas cuyas etiquetas ontológicas pertenecen a las tres categorías, ya que una proteína puede ser descrita por medio de su función molecular, el proceso biológico en el que participa y ser un componente celular[57, 58], por lo que la idea de que cada gen podía codificar una función es anticuada y poco útil.

La información del Consorcio de Gene Ontology ha sido publicada y de acceso libre en forma de herramientas web para analizar y asociar información, además de archivos en formato OWL y OBO para visualizar las ontologías, y formatos GAF y GPAD+GPI para analizar las anotaciones registradas en su página. Estos archivos muestran etiquetas ontológicas y la relación que tienen con otras ontologías usando, entre otros, los calificativos lógicos descritos en la Tabla 2.1. Estas mismas etiquetas han sido utilizadas ampliamente por medio de inferencia informática, creando bases de datos que se encuentran disponibles en sitios específicos y en oficiales, llegando incluso

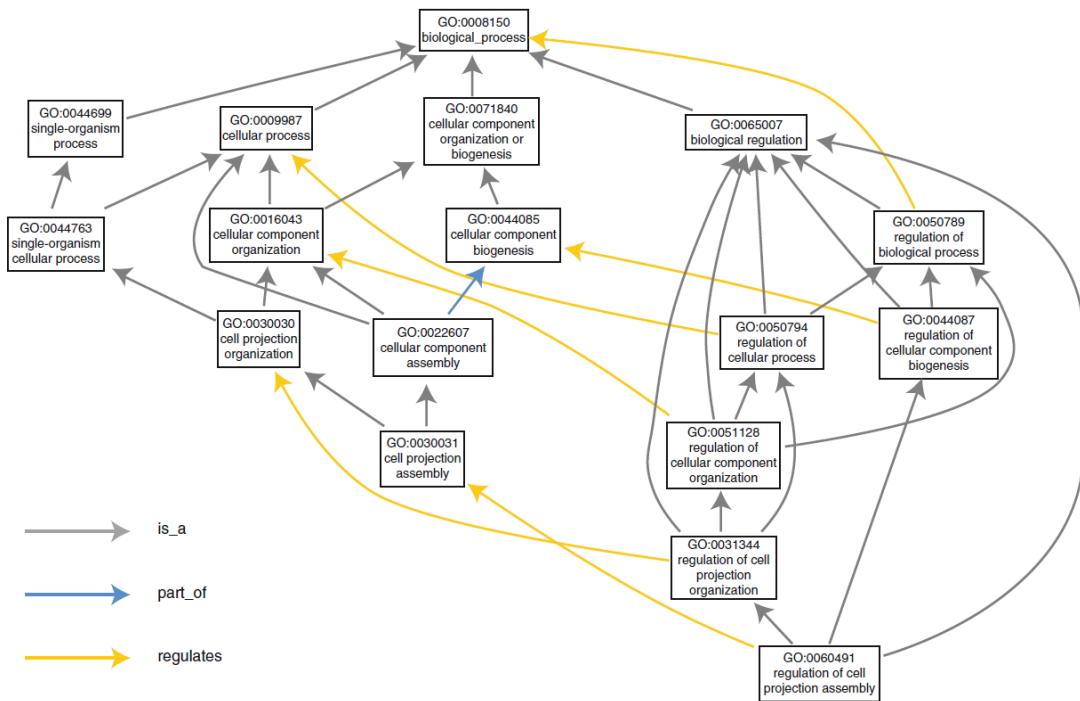


Figura 2.1: Diagrama de ancestros de la ontología GO:0060491. Recuperada de [24] el 16 de Noviembre del 2022.

a mostrar estas relaciones en Uniprot y en las bases de datos de NCBI, probando su utilidad para el análisis de la información de genes y proteínas.

2.2. Herramientas de procesamiento de secuencias de DNA

Ya se mencionó la existencia de herramientas de alineamiento de secuencias de DNA, presente entre el catálogo de herramientas de BLAST[9] y de herramientas del EMBL[52], pero las secuencias de DNA pueden analizarse por otros métodos, como las técnicas de *Machine learning* que han sido ampliamente utilizadas en análisis de secuenciación de conjuntos de datos genómicos, anotaciones de elementos de secuencia, datos epigenéticos, datos proteómicos o metabolómicos[15]. Derivado del Machine Learning, el uso de Deep Learning ha surgido como una técnica muy útil para procesar grandes cantidades de información y ha sido utilizada en el área de

biología de distintas formas[34].

En el trabajo de Morales et al.[41] se procesan secuencias de DNA como señales usando su secuencia específica para obtener atributos únicos que permitan la diferenciar y clasificar secuencias entre secciones codificantes, secciones no codificantes y pseudogenes.

Trabajos como DeepBind[14], DeepSEA[16] y Basset[17], y otras publicaciones como la de Yin et al.[30] y AMBER[48], analizan secuencias de DNA utilizando fragmentos o extractos de las secuencias, utilizando redes neuronales convolucionales para encontrar patrones en estas secuencias y asociarlos a funcionalidades biológicas. Además existen abordajes al análisis de secuencias de DNA por medio de redes neuronales recurrentes (RNN por sus siglas en inglés) leyendo las secuencias de DNA fragmentadas ingresándose al modelo del mismo modo en que se hace análisis de texto con este tipo de arquitectura, tales como deepTarget[18], KEGRU[28], deepMiRGene[19] o el trabajo de Deif et al.

Además han habido modelos que combinan estas arquitecturas para obtener mejores resultados acorde al tipo de características deseadas en sus respectivos proyectos de investigación, usando como ejemplo DanQ[20] y NCNet[35] que combinan capas convolucionales con capas de recurrencia de tipo memoria de largo plazo (LSTM por sus siglas en inglés).

Capítulo 3

Resultados y Discusión

3.1. Base de datos

En el trabajo de Borrero et al.[36] se utilizaron genomas bacterianos de las bases de datos del NCBI, por lo que inicialmente se descargaron genes del NCBI por medio de la librería Entrez de Biopython[7], aunque los tiempos de descarga limitaban mucho el análisis y procesado de la base de datos, y se procedió a utilizar los enlaces FTP del NCBI[59], sin embargo se cambió la base de datos debido a que posterior a conseguir las secuencias de genes había que asociar esas secuencias a ontologías de genes, y no había una asociación directa entre las bases de datos ya que había que utilizar una base de datos de Uniprot para identificar los registros que tenían el enlace a los posibles ID del NCBI y sus respectivos ID de Gene Ontology, añadiendo el hecho de que las etiquetas ontológicas podían o no estar verificadas por Swiss-Prot[56].

Posterior a esto se utilizaron librerías de R para enriquecer los genes con etiquetas ontológicas. Se utilizó GOFuncR[50], EGSEA[21], ClusterProfiler[46], y una herramienta llamada DAVID[55]. En el caso de las librerías de R variaba el algoritmo con el que asociaban la información, pero finalmente se requería un "Gen Name" con el qué asociar las etiquetas ontológicas, utilizando bases de datos de organismos específicos. Para ello era necesario importar los diccionarios de registros de genes y ontologías por organismos, siendo registros hechos en determinada fecha y que en el mejor de los casos se actualizaban anualmente- Además de esto los genes que habían sido des-

cargados del NCBI tenían nombres que no coincidían con los nombres registrados en estas librerías, o que si coincidían era en organismos no bacterianos, por lo que esta asociación funcionaba más por nombre en común que por secuencia. Dentro de los organismos registrados en Bioconductor[60], se encontraba *E. coli* publicado por el NCBI[13] cuya última actualización había sido en 2014 y contenía la información de genomas de E.coli, información que había sido obtenida por los enlaces FTP y que retorna al punto en el que la información estaba desactualizada y probablemente no verificada.

Con la herramienta de DAVID hubo un problema similar, ya que su herramienta utilizaba el nombre de gen para asociar información de distintos sitios. El inconveniente fue el difícil acceso a sus bases de datos para hacer análisis múltiple, además de que la respuesta de la búsqueda mezclaba la información de los registros ingresados a análisis, lo que dificultaba analizar las ontologías de cada secuencia. Aunque DAVID disponía de la descarga de sus bases de datos para uso local, se requerían permisos de acceso que tardaban mucho tiempo en procesarse ya que al final no garantizaba que la información asociada estuviera verificada y no asociada solo por nombre de gen, aún cuando es posible seleccionar el taxón de organismo del que se obtuviera información, ignorando las diferencias que pudiera haber entre genes de mismo nombre pero con secuencias distintas.

Finalmente se utilizó la base de datos mencionada en la Sección de 1.3.2, debido a que era una base de datos que recopila registros de experimentos de laboratorio húmedo que validaban la funcionalidad de proteínas de *E. coli* (cepa k12). De la información extraída de esta base de datos se mantuvo el ID de cada proteína registrada y las etiquetas ontológicas asociadas a cada ID. Se sumaron todos los registros de etiquetas ontológicas a cada ID y después se descargó la secuencia de cada proteína acorde a su registro de Uniprot, obteniendo en total 3920 registros de secuencias de proteínas y sus etiquetas ontológicas asociadas para procesar.

3.1.1. Retrotraducción y Generación de vectores de kmeros

En la sección de 2.2 se observaron distintos ejemplos de modelos que analizaban secuencias de DNA, y en cada uno de estos la longitud de la secuencia era un parámetro determinante en sus proyectos. Por esto se utiliza como referencia el trabajo de Borrero et al. [36]. es posible analizar secuencias de DNA como fragmentos estadísticamente representativos que en el caso de las bacterias tienen tamaño de 11 a 15 nucleótidos, pudiendo contener dentro de su longitud funciones asociadas a los productos de estos genes. Tales funciones pueden ser analizadas por medio de etiquetas ontológicas de genes.

Se realizó una retrotraducción de cada proteína dividiendo cada secuencia en palabras de 5 aminoácidos formando palabras con superposición de 1 aminoácido para generar retrotraducciones de tamaño 15 y después 3 kmeros por cada retrotraducción, manteniendo la continuidad de superposición de un nucleótido para los kmeros eliminando los elementos repetidos de cada sección de kmeros ya que al final son kmeros posibles que en conjunto pueden codificar esa proteína, como se muestra en la Figura 3.1. Estos kmeros se etiquetaron como "NNNNNNNNNNNNN.#", donde cada N representa un nucleótido y el símbolo # representa la proteína de origen del kmero, de manera que sea posible asociar el kmero a su conjunto de ontologías correspondiente. Cada conjunto de kmeros generados por proteína se mezcló con un procedimiento de *Shuffle* en el que por una cantidad de elementos M en un vector dado, en este caso la lista de kmeros para una proteína, y se genera una lista de números desde 1 hasta M de manera aleatoria para reordenar los datos en el orden de la lista generada con el objetivo de que el modelo asocie las etiquetas ontológicas al kmero dado y no al orden en el que los kmeros se ingresan a la red para su entrenamiento.

Cada conjunto de kmeros se dividió en 11 archivos, siendo 10 designados para su uso en entrenamiento de los modelos y de donde se extrajo el conjunto de datos de validación, el archivo restante fue designado para su uso en las pruebas de Testing.

Tales kmeros se repiten en diferentes secciones de la proteína, así como en distintas proteínas, por lo que se contemplan dos posibilidades:

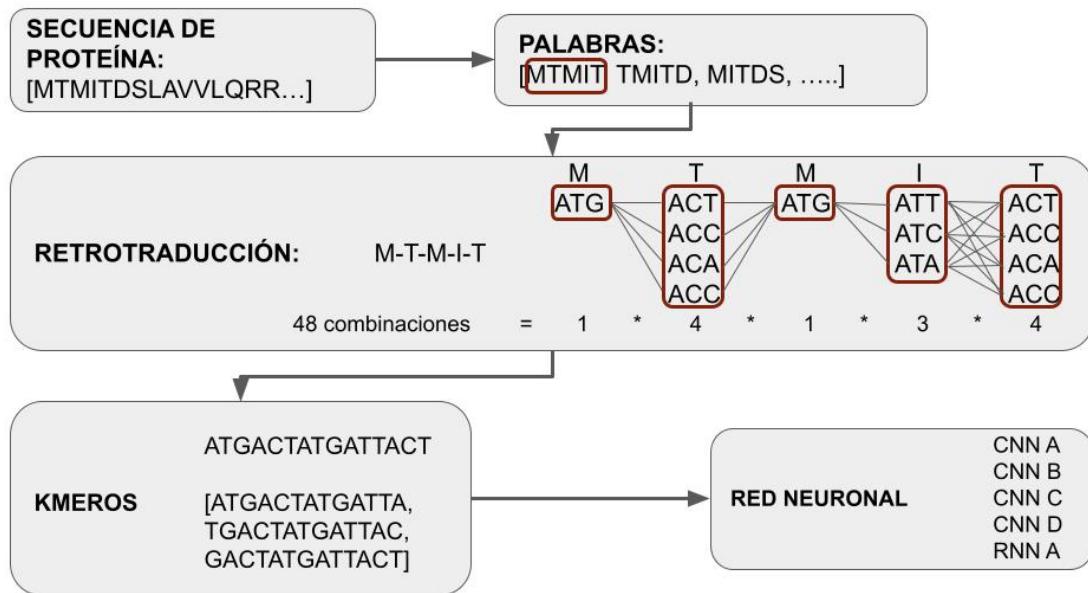


Figura 3.1: Diagrama de Generación de kmeros a partir de secuencia de proteína.

1. Los kmeros que se repiten con mayor frecuencia en distintas proteínas sin ontologías en común pueden ser kmeros con menor relación a etiquetas ontológicas, y que podrían resultar necesarias para la estructura del gen uniendo "los kmeros que sean representativos de funciones ontológicas.
2. Los kmeros que se repiten en proteínas con ontologías en común pueden ser kmeros que representen tal función, mientras que los kmeros que aparecen menos pueden asociarse a las ontologías con menor aparición siempre que estos kmeros se encuentren en las proteínas que contengan estas etiquetas.

Después de las versiones 1 de las redes neuronales se ajustó el formato de los kmeros, tal que en vez de separar cada kmero de las retrotraducciones de cada proteína, se realizara un conjunto de kmeros a los que llamaremos "sinónimos" siempre que un grupo de kmeros apareciera como opción de retrotraducción posible de la sección de proteína de donde se realizó la retrotraducción, separando cada kmero sinónimo como muestra para el modelo, manteniendo la asociación de cada kmero sinónimo al conjunto de ontologías que le corresponde. Cada conjunto de sinónimos se mezcló con el mismo método *Shuffle*, para mezclar el orden de cada conjunto de sinónimos y

distribuyéndose del mismo modo que los archivos anteriores. Este cambio tuvo como objetivo facilitar al modelo el aprendizaje mostrando que más de un kmero puede representar las mismas funciones ontológicas.

3.1.2. Estadística de etiquetas ontológicas

Analizando la distribución de frecuencias de las etiquetas ontológicas obtenemos un total de 4206 etiquetas en esta base de datos de las 43588 etiquetas actualizadas y activas existentes en el catálogo de Gene Ontology[61] hasta el día 23 de Septiembre del 2022, obteniendo el histograma de la Figura 3.2.



Figura 3.2: Histograma de frecuencia de ontologías de *E. coli*.

En los conteos de ontologías por registro el mínimo es 1, dado en 199 casos de los 3920 registros de la base de datos, mientras que el máximo de etiquetas por gen es 34, con un valor promedio de 8.5 etiquetas por gen. Además Analizando las ontologías observamos que el mínimo de aparición por etiqueta es 1 en 1996 casos, casi el 50 % de las ontologías totales de este grupo, y el máximo de aparición de una ontología es 1320.

3.1.3. Vectores de Ontologías

Para la generación de vectores de ontologías se procesó el archivo OBO de Gene Ontology[63] del día 23 de Septiembre del 2022 con el fin de generar un diccionario que contiene las 43558 etiquetas ontológicas disponibles y activas hasta ese día. Con esto se generó un vector de tamaño 1×43558 para cada proteína donde cada 1 es una ontología presente en ésta, y cada 0 una ontología ausente.

A partir de los resultados de la versión 1 de los modelos convolucionales mostrados en las Figuras 3.4, 3.6, 3.8 y 3.10 se modificó el vector de salidas, utilizando 4206 etiquetas ontológicas que son las registradas en la base de datos, eliminando el ruido generado por las 36352 etiquetas ausentes en la base de datos. Originalmente se utilizaron las etiquetas ontológicas totales a manera de control negativo para asegurar que el modelo asociara el kmero con la información que podría contener según su tamaño y el organismo del que era obtenido, sin embargo la capacidad computacional resultó una limitante al momento de generar un modelo lo suficientemente complejo que distinguiera entre las etiquetas relevantes para el kmero de entrada y las no relevantes que serían completamente ausentes en la base de datos. Los resultados de esta modificación se muestran en las Figuras ?? y ?? para la versión 3 de los modelos mostrados.

3.2. Modelos

Hay distintas características a considerar en el desarrollo de modelos, como la complejidad de la arquitectura, la cantidad de neuronas o los hiperparámetros. Para este proyecto se utiliza un modelo multiclasificación, ya que se ingresarán secuencias de kmberos asociadas a un vector de ontologías que puede tener más de una etiqueta positiva, además de que se desea obtener un vector que liste la probabilidad de pertenencia de la muestra para cada clase[51].

De manera inicial se trabajaron dos arquitecturas convolucionales descritas en la Sección 3.2.1, y una arquitectura recurrente descrita en la Sección 3.2.2 para hacer pruebas iniciales, acorde a la información de trabajos relacionados ya citados en la

sección 2.2. Cada una de las arquitecturas iniciales fueron entrenadas y los resultados de estos experimentos son mostrados en la sección correspondiente a cada modelo. Posterior a esto se desarrollaron dos arquitecturas que fueron resultado de la unión de ambos tipos de redes neuronales ya mencionadas, las cuales se encuentran en la Sección 3.2.1 como arquitecturas CNN C y CNN D y a las cuales se les hicieron las mismas pruebas que a las CNN A y CNN B.

Debido a que cada kmero está asociado al vector de ontologías asociado según la proteína del que haya sido obtenido, variando desde 1 etiqueta hasta 34, es necesario utilizar una métrica que nos permita analizar realmente la capacidad del modelo de distinguir entre etiquetas presentes y ausentes, por lo que se recurrió a utilizar área bajo la curva ROC, que a diferencia de la métrica *accuracy* que es bastante utilizada en modelos de redes neuronales tanto para proyectos simples de aprendizaje profundo como proyectos aplicados a investigación, asigna el mismo peso de la métrica entre los verdaderos positivos y los verdaderos negativos(TP y TN respectivamente)[38, 39] por lo que un valor mas cercano a 1 indica mayor capacidad del modelo para distinguir las clases presentes y clases ausentes para un modelo multiclasa, facilitando el análisis del rendimiento del modelo.

Para todos los modelos la función de activación utilizada en la capa de salida es Softmax. Esto es debido a que es una función que en su salida muestra un vector de probabilidad de pertenencia a cada caso, donde la suma del vector total es 1[49]. Con esto se pretende encontrar la clase mayormente representativa de cada kmero, siendo un problema multiclasa. Esta función es compatible con las funciones de pérdida de Cross Entropy, ya que penaliza únicamente los errores de las clases positivas originales, midiendo no solo la presencia del error sino su magnitud[53].

Para cada modelo se siguió el mismo procedimiento para ingresar la información, mismos ciclos de entrenamiento y mismas modificaciones ya mencionadas. Para cada caso se detallarán las modificaciones que se hayan hecho de manera particular al modelo en su respectiva sección según la versión del modelo.

3.2.1. CNN

Se generaron 4 arquitecturas para trabajar las cuales se describen más adelante, así como los cambios hechos a cada una. De manera inicial se buscaron arquitecturas ampliamente utilizadas, como ejemplos prácticos de CNN aplicadas al análisis de números escritos a mano de la base de datos MNIST. Cada red neuronal utilizó datos con el siguiente formato:

1. Vector X: Cada conjunto de kmeros sinónimos fue separado y convertido a un vector OHE (One Hot Encode) de $4*13*1$ para cada kmero del conjunto.
2. Vector Y: Cada conjunto de kmeros viene asociado a su proteína de origen, asociando cada kmero al mismo conjunto de ontologías que le corresponde en un vector OHE de $1*43558$.

Se probaron las redes con los parámetros iniciales de formato de datos, y luego de observar los resultados de las Figuras 3.4, 3.6, 3.8 y 3.10 donde podemos observar las gráficas de rendimiento de cada modelo en cada una de sus versiones. Del lado derecho de cada gráfica se observan los valores de Training Loss (TL), Validation Loss (VL), Training AUC ROC (TAUC) y Validation AUC ROC (VAUC) con valores media y mediana de cada uno obtenidos de los historiales de entrenamiento del modelo. Cada modelo tenía sus particularidades descritas en su sección correspondiente, pero se decidió comparar el desempeño de los modelos haciendo los mismos cambios para cada una bajo dos ideas principales:

1. Los modelos en su versión 1 no tienen la complejidad suficiente para procesar la información ya que los kmeros de entrada tenían una variación muy grande en presencia y frecuencia por secuencia de DNA generada a partir de la proteína de donde se obtuvo el vector de ontologías.
2. Los kmeros producidos de esta base de datos provienen de bacterias y la elección del tamaño de los kmeros para este trabajo fue a partir de la información que puede contener un genoma bacteriano.

Por esto a partir del punto 1 se generó la versión 2 de los modelos donde las entradas se ordenan a partir de kmeros sinónimos como se describe en la Sección 3.1.1. La versión 3 de los modelos se hizo a partir del punto 2, convirtiendo las salidas a vectores exclusivos de ontologías presentes en la base de datos, asociando los kmeros con únicamente las ontologías registradas en la base de datos utilizada. Con esto se desarrolló la fase de testing donde se compararon desempeños de los cuatro modelos convolucionales y el modelo recurrente.

Cada modelo se entrenó con aproximadamente 74300 lotes de datos, con 3000 datos por lote, 5 épocas de entrenamiento y 10 ciclos de entrenamiento y 5 épocas de entrenamiento por ciclo, dando un total aproximado de 223 millones de datos de entrenamiento.

CNN A

Inicialmente se buscaron ejemplos sencillos para el procesado de imágenes y un ejemplo popular en prácticas de CNN es la clasificación de números escritos a mano de la base de datos MNIST, particularmente observando el ejemplo de Brownlee[31] que utiliza una arquitectura muy sencilla para ese problema. A la arquitectura seleccionada se le hicieron modificaciones acorde al tipo de datos obtenidos y la que la entrada es un vector en forma de matriz de tamaño 4*13*1.

Se irán describiendo los cambios correspondientes a cada versión de la red neuronal empezando por la descripción del modelo inicial mostrado en la Figura 3.3. La compilación de este modelo en su **versión 1** se ejecutó con los siguientes parámetros:

- Optimizador: Se utiliza el optimizador ADAM que ha demostrado ser uno de los mejores optimizadores para este tipo de análisis por su buen rendimiento[22, 45].
- Loss: Se utilizó *Categorical Crossentropy* debido a que esta función es útil en problemas de clasificación multiclase, calculando el error del modelo a partir de su clasificación en vectores One Hot como el que se está utilizando para la

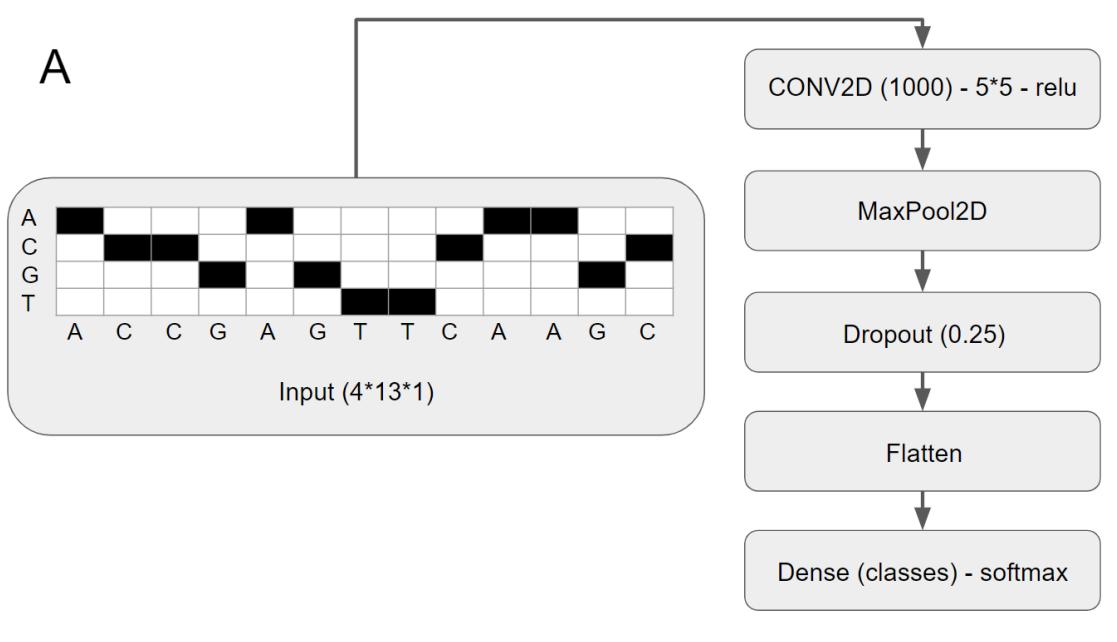


Figura 3.3: Arquitectura inicial del modelo CNN A como modelo simple para procesar la información.

salida de las etiquetas ontológicas, analizando la pérdida de las clases positivas de la muestra[26].

- Métricas: Área bajo la curva ROC (ROC AUC por sus siglas en inglés).

La función de Categorical Cross entropy es comúnmente usada en problemas multiclase, sin embargo para este problema esta función no era adecuada, ya que la evaluación se hacía con respecto al vector original, tomando la muestra no solo como multi etiqueta, sumando los errores de cada clase positiva. Sin embargo este dicho se contrapone con el hecho de usar Softmax como la función de activación final, ya que se pretende obtener una única clase como estadísticamente predominante, por lo que para utilizar la función de Categorical Cross entropy habría que usar una función final de activación como Sigmoid, que retorna un vector de 0 y 1 para problemas multi etiqueta[49, 26]. Por esto se volvió necesario cambiar al menos uno de los dos hiperparámetros del modelo, la función de Loss o la función de activación final.

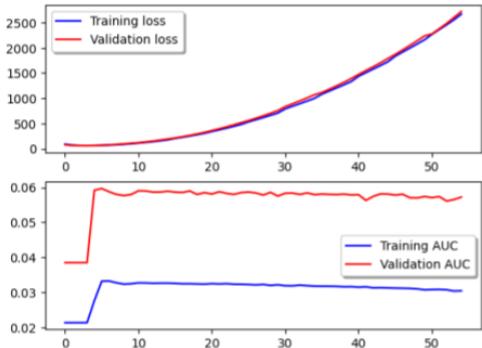
Versión 2: Habiendo comparado los resultados con las primeras pruebas de la arquitectura CNN B, y las que se desarrollaron posteriormente CNN C y CNN D se modificó el modelo de la siguiente forma:

- Loss: Se modificó para tener *Binary Cross-entropy*, ya que se llegó a la conclusión de que aunque la secuencia completa del gen que codifica la proteína de donde se obtuvo cada kmero, los kmberos no necesariamente codificarán todas las etiquetas del conjunto de origen, por lo que un cálculo individual por clase es más conveniente, permitiendo encontrar las funciones asociadas según la frecuencia de aparición y el conjunto de etiquetas de origen, permitiendo evaluar el error de cada clase positiva individualmente sin que el conjunto completo de ontologías positivas afecte al cálculo de pérdida del modelo[26].
- Kmberos: Se agruparon los kmberos por conjuntos, permitiendo al modelo aprender que hay más de una posible combinación de nucleótidos (visto por el modelo como el vector OHE de entrada). El procedimiento de este paso es explicado en la Sección 3.1.1.

El cambio de la función de Loss mostró un buen avance en la evolución del modelo, lo que se puede observar en la Figura 3.6 en su versión 2, donde el valor de Loss disminuyó considerablemente. Esto es ventajoso para este experimento y útil para obtener la clase de mayor probabilidad del modelo, aún si cada muestra inicialmente fue ingresada como multi etiqueta para obtener un vector de probabilidades que se pueda analizar más adelante en la etapa de Testing. Este cambio se utilizó igualmente en el modelo CNN B, descritos en su respectiva sección.

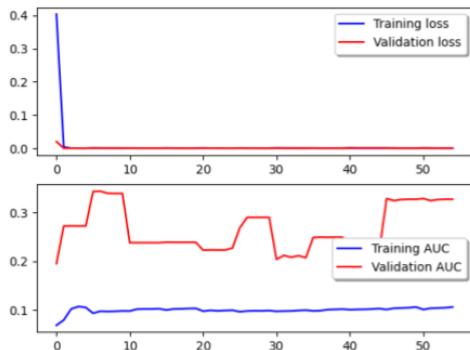
Versión 3: Además de la modificación del vector de salida no hubieron otros cambios para esta modelo, manteniendo la arquitectura y cambiando únicamente el vector de ontologías.

CNN A Versión 1



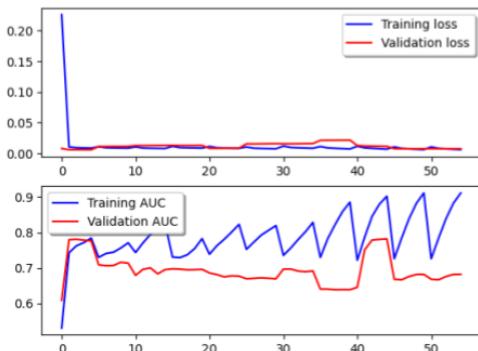
	MEDIA	MEDIANA
TL:	5.7E+03	3.1E+03
VL:	5.8E+03	3.1E+03
TAUC:	2.8E-02	2.96E-02
VAUC:	5.08E-02	5.6E-02

CNN A Versión 2



	MEDIA	MEDIANA
TL:	7.50E-03	8.9E-04
VL:	1.1E-03	1.1E-03
TAUC:	1.1E-01	1.1E-01
VAUC:	1.1E-01	1.1E-01

CNN A Versión 3



	MEDIA	MEDIANA
TL:	1,24E-02	8.4E-03
VL:	1.2E-02	1.1E-03
TAUC:	7.9E-01	7.8E-01
VAUC:	7E-01	6.8E-01

Figura 3.4: Historial de red neuronal convolucional A para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.

CNN B

Para la arquitectura de esta red neuronal se ha usado un modelo para clasificar números escritos a mano[40] con mayor nivel de complejidad que el modelo CNN A, modificando los hiper parámetros para ajustar el número de neuronas según el número de ontologías posibles donde pudiera relacionarse cada kmero ingresado. La arquitectura utilizada se muestra en la Figura 3.5.

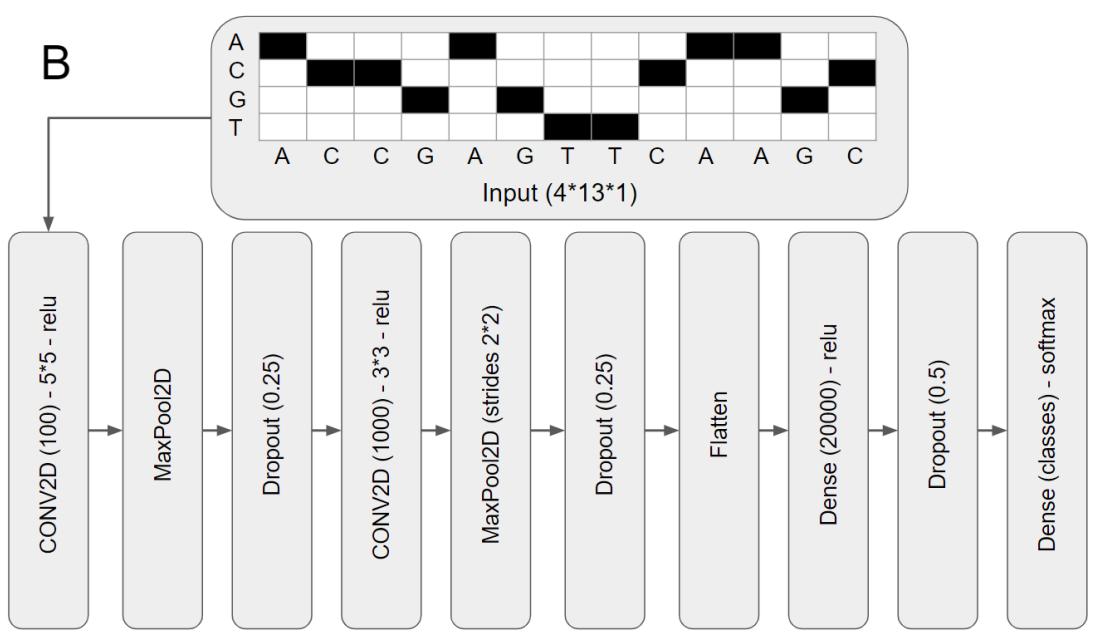


Figura 3.5: Arquitectura del modelo CNN B de dos bloques de convolución.

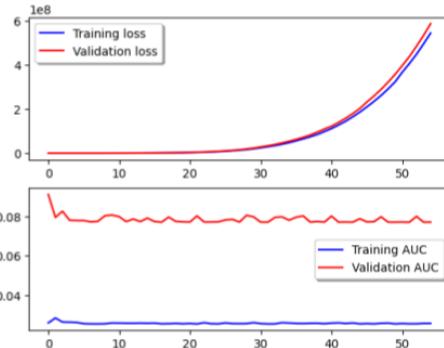
La compilación de este modelo se ejecutó con los siguientes hiper parámetros:

- Optimizador: Al igual que el modelo CNN A se utilizó la función ADAM.
- Loss: Se utilizó *Categorical Cross-Entropy* igual que el modelo CNN A en su estado inicial.
- Métricas: Área bajo la curva ROC (ROC AUC por sus siglas en inglés).

Versión 2: Se modificaron los hiper parámetros descritos inicialmente, cambiando categorical Cross-entropy por Binary Cross-entropy por la misma razón explicada previamente, además de el cambio de agrupación de los kmberos sinónimos.

Para la **versión 3** no hubo más modificaciones que las mencionadas previamente.

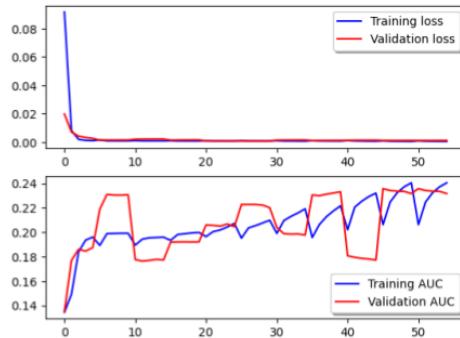
CNN B Versión 1



	MEDIA	MEDIANA
TL:	1.7E+11	5.1E+10
VL:	1.7E+11	5.2E+10

TAUC:	2.6E-02	2.6E-02
VAUC:	7.8E-02	7.7E-02

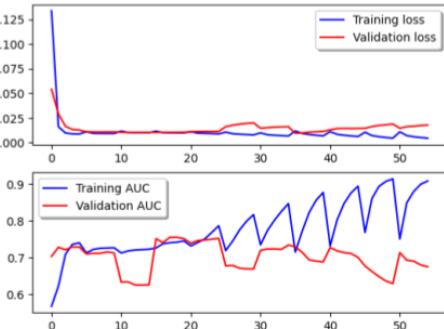
CNN B Versión 2



	MEDIA	MEDIANA
TL:	2.55E-03	7.5E-04
VL:	1.8E-03	1.3E-03

TAUC:	2.06E-01	2.03E-01
VAUC:	2.06E-01	2.05E-01

CNN B Versión 3



	MEDIA	MEDIANA
TL:	5.66E-03	3.9E-03
VL:	1.67E-02	1.2E-02

TAUC:	8.5E-01	8.9E-01
VAUC:	6.71E-01	6.7E-01

Figura 3.6: Historial de red neuronal convolucional B para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.

CNN C

Este modelo fue diseñado a partir de las arquitectura de los modelos DanQ[20] y NCNet[35], mezclando la arquitectura convolucional y la recurrente bidireccional, con el objetivo de aprovechar la capacidad de la capa convolucional de extraer patrones y de la recurrente de aprender de esos patrones y asignarles un valor según el contexto dado, aprovechando la memoria de estas capas. Este modelo está descrito en la Figura 3.7 como la **versión 1**.

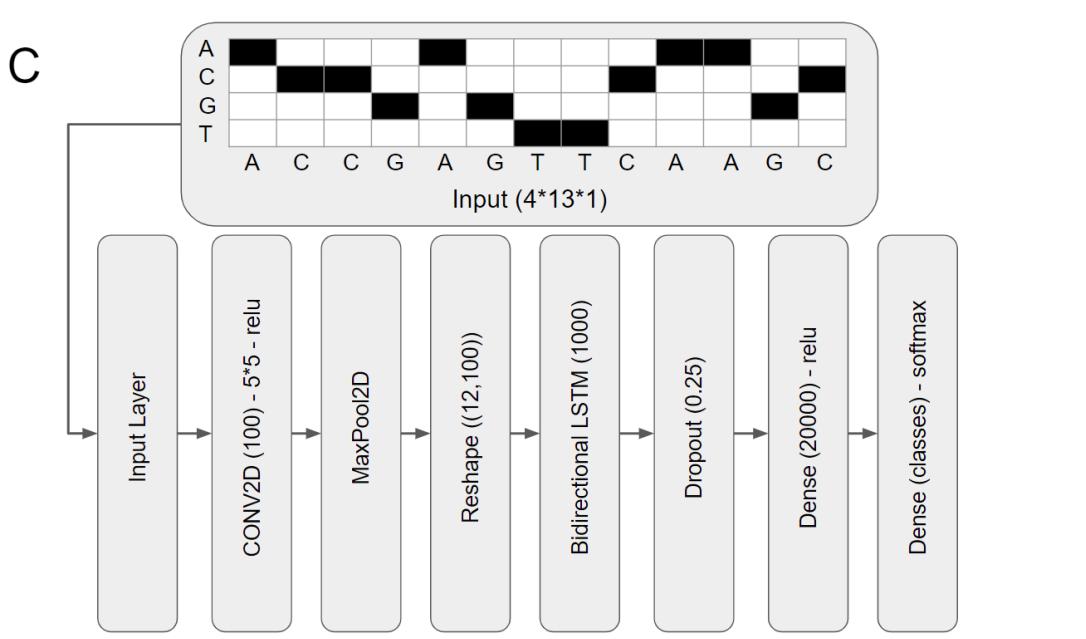
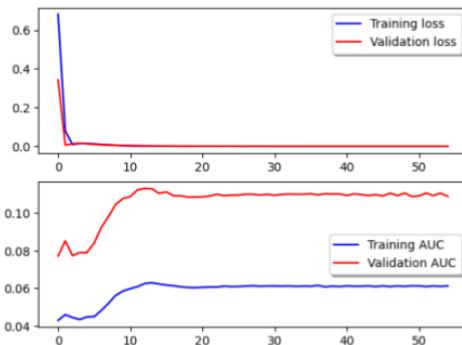


Figura 3.7: Arquitectura CNN C basada en modelo DanQ y NCNet. Combinación de arquitectura de una capa convolucional y una capa Bidirectional LSTM RNN.

El desempeño de este modelo mostró incluso mejor desempeño que la CNN A y CNN B en mismas versiones, visible en la Figura ??pero el tiempo de ejecución era de aproximadamente 8 horas por entrenamiento, más que las convolucionales que terminaban su entrenamiento en 2-3 horas, por lo que se desarrolló una variante de esta mezcla de arquitecturas, obteniendo la arquitectura CNN D que será explicada más adelante.

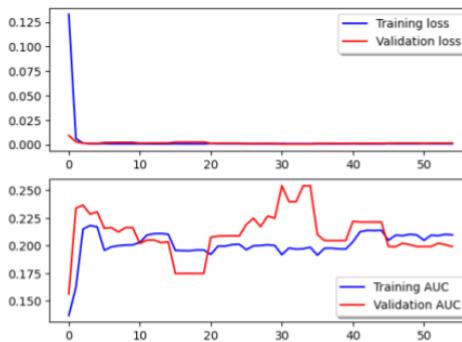
Para la **versión 2** y **versión 3** solo se siguieron los cambios descritos previamente sin modificar ningún otro parámetro de la red.

CNN C Versión 1



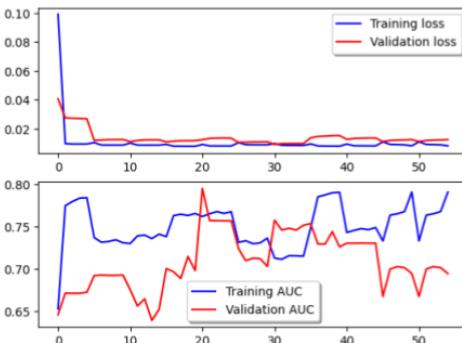
	MEDIA	MEDIANA
TL:	1.63E-02	1.1E-03
VL:	8.7E-03	1.04E-03
TAUC:	5.8E-02	6.1E-02
VAUC:	1.06E-01	1.2E-01

CNN C Versión 2



	MEDIA	MEDIANA
TL:	3.32E-03	8.2E-04
VL:	1.6E-03	1.4E-03
TAUC:	2.01E-01	2E-01
VAUC:	2.1E-01	2.1E-01

CNN C Versión 3



	MEDIA	MEDIANA
TL:	1.06E-02	8.8E-03
VL:	1.4E-02	1.2E-02
TAUC:	7.5E-01	7.4E-01
VAUC:	7.1E-01	7.E-01

Figura 3.8: Historial de red neuronal convolucional C para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.

CNN D

Bajo la misma dirección de un modelo híbrido se desarrolló la CNN D, sin embargo por los tiempos de ejecución se buscó modificar la dirección de la carga de memoria, usando una capa convolucional de una dirección con memoria en vez de la capa convolucional de dos dimensiones, para luego usar una capa bidireccional GRU que usa mecanismos de recurrencia, elegida para este modelo a partir del modelo de Zhang et al.[47]. Esta arquitectura se muestra en la Figura 3.9.

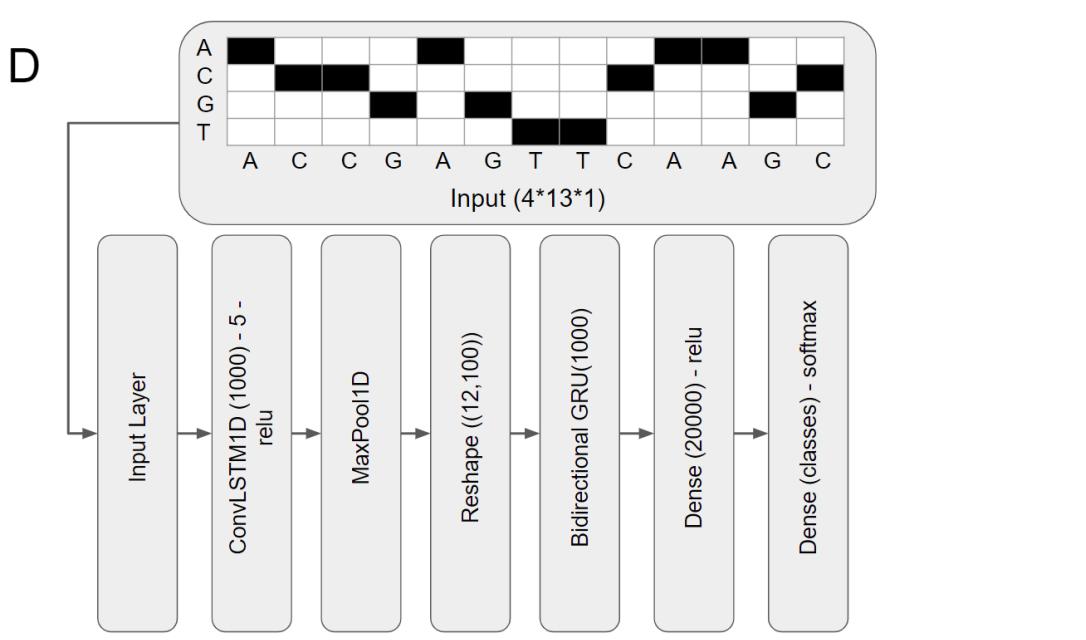
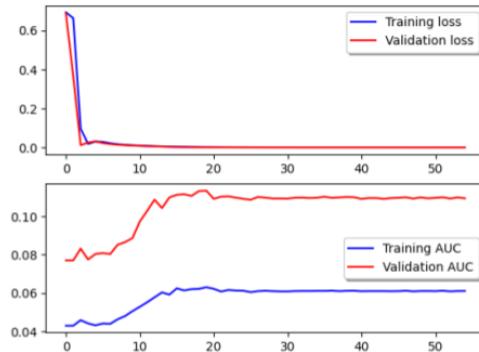


Figura 3.9: Arquitectura de modelo CNN D. Se utilizan capas de convolución 1D con memoria junto una capa GRU como un segundo ejercicio de combinación con arquitectura de tipo recurrente, enfocando la memoria a la capa convolucional.

La **versión 1** de este modelo tuvo desempeño similar a la CNN C en su misma versión, y el tiempo de ejecución se disminuyó a 6 horas. Estos modelos comenzaron a diferenciarse a partir de las siguientes versiones.

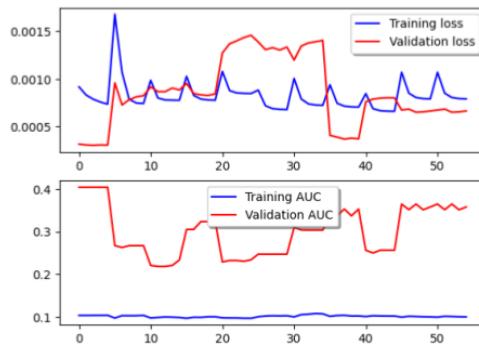
Para la **versión 2** y **versión 3** la CNN D siguió manteniéndose cerca en rendimiento con respecto a las mismas versiones de CNN C.

CNN D Versión 1



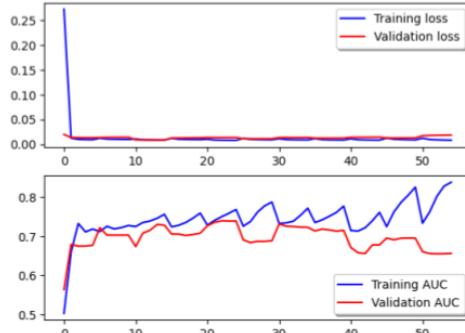
	MEDIA	MEDIANA
TL:	3.06E-02	1.1E-03
VL:	2.3E-02	1.1E-03
TAUC:	5.8E-02	6.1E-02
VAUC:	1.01E-01	1.1E-01

CNN D Versión 2



	MEDIA	MEDIANA
TL:	8.26E-04	7.9E-04
VL:	8.6E-04	1.4E-03
TAUC:	1E-01	1E-01
VAUC:	3E-01	3.03E-01

CNN D Versión 3



	MEDIA	MEDIANA
TL:	8.56E-03	8.3E-03
VL:	1.3E-02	1.3E-02
TAUC:	8.1E-01	7.8E-01
VAUC:	7.1E-01	6.9E-01

Figura 3.10: Historial de red neuronal convolucional D para sus tres versiones. Para cada versión del modelo se muestra la gráfica de resultados de curva de Loss y curva de valores AUC ROC comparativa entre entrenamiento y validación en la posición respectiva para cada modelo, además de sus valores media y mediana al lado derecho de cada gráfica.

Entrenamiento extendido

Ya procesados los modelos es posible generar un historial de desempeño medio de cada modelo para analizar de una manera más directa la evolución de cada modelo y elegir la versión que se utilizaría para la prueba de entrenamiento extendido. Estos resultados se muestran en la Figura 3.11, donde se muestran del lado izquierdo los valores promedio de los históricos de cada versión de cada modelo, y del lado derecho se muestran las medianas para comparar el desempeño a partir de la dispersión de los datos, mostrando que a pesar de la variación vista en los modelos en las Figuras 3.4, 3.6, 3.8 y 3.10, los valores intermedios resultan estar cerca de los promedios.

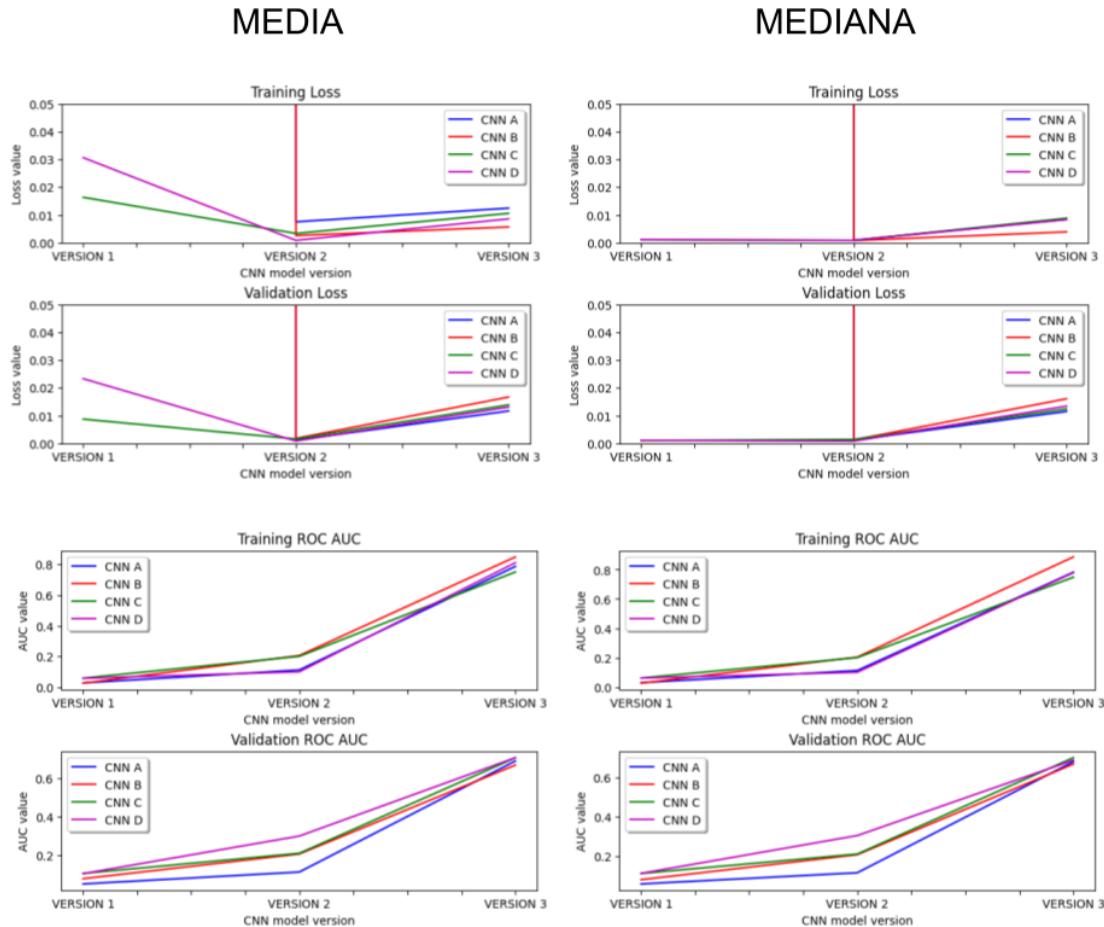


Figura 3.11: Historial de medias y medianas de modelos convolucionales en sus tres versiones.

A partir de esta información se seleccionó la versión 3 de cada modelo para trabajar

en la prueba de entrenamiento extendido, entrenando los modelos con 100 ciclos en vez de 10, usando aproximadamente 22300 millones de kmeros de las bases de datos de entrenamiento para entrenar al modelo, esperando observar una disminución en la variación del historial de curva ROC de cada modelo. Estos resultados se observan en la Figura 3.12 en el que se observa que los modelos no se estabilizan al pasar los ciclos.

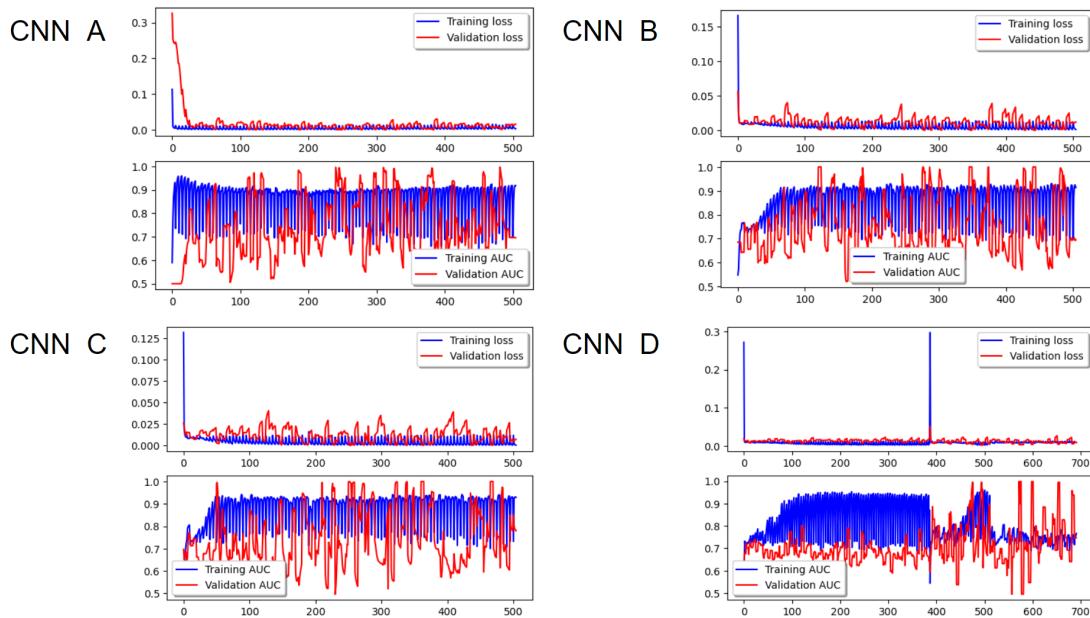


Figura 3.12: Resultados de entrenamiento largo de los cuatro modelos convolucionales en sus terceras versiones.

Es observable que al pasar los ciclos los modelos no se estabilizan, pudiendo deberse a que los registros de proteínas de la base de datos utilizada tuvieran un número específico de ontologías asociadas, no significando que sean éstas las únicas ontologías que podrían representar las funciones de la proteína analizada, pues recordemos que esta base de datos es una lista de registros verificados en laboratorio húmedo de las actividades evaluadas en cada proteína listada, y un falso positivo en realidad podría

representar un enriquecimiento de etiquetas a partir de las ontologías presentes en otras proteínas que comparten mismos kmeros representativos.

Se contempló la posibilidad de overfitting de los modelos debido a los valores media y mediana de validación inferiores a los valores de entrenamiento, sin embargo según Sharma esto podría solucionarse utilizando capas de Dropout para obligar al modelo a olvidar los datos memorizados y procesar únicamente los patrones analizados, sin embargo los modelos CNN A, CNN B y CNN C ya tienen capas Dropout en su arquitectura, por lo que esta idea se descartó.

3.2.2. RNN

Así como se han utilizado redes neuronales convolucionales en algoritmos de aprendizaje profundo para obtener patrones de datos y asociarlos a información, las redes recurrentes muestran otro tipo de cualidades. Se han utilizado en el área de la genómica para asociar funciones a secuencias y tal como se utilizaron en los modelos CNN C y CNN D combinadas con capas convolucionales, se desarrolló un modelo recurrente para evaluar su desempeño final en comparación a los modelos convolucionales.

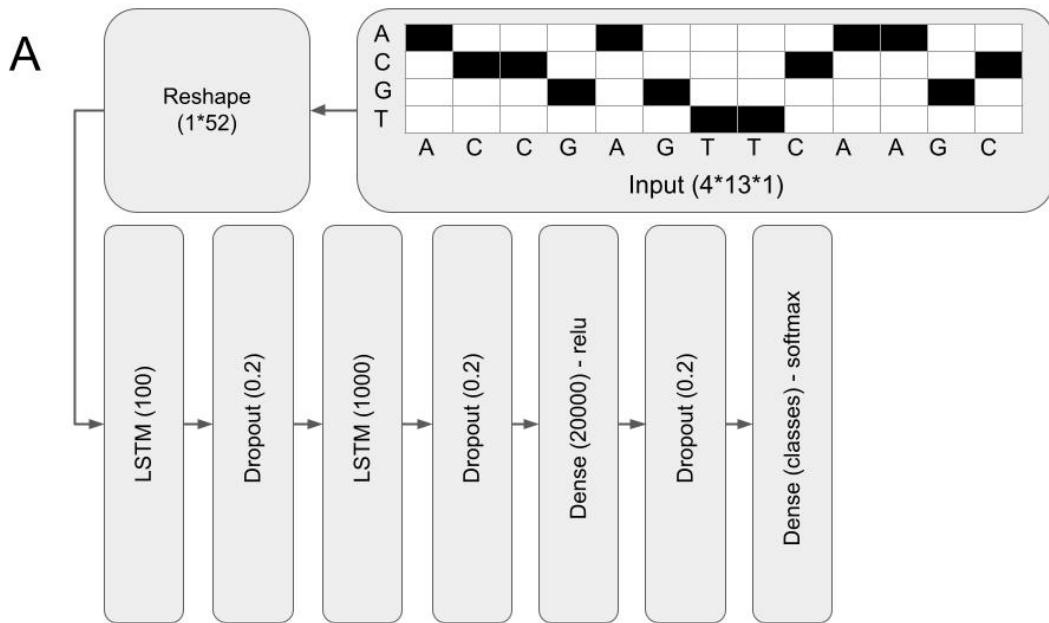


Figura 3.13: Arquitectura de RNN A con dos capas LSTM.

Este modelo fue desarrollado después de los cuatro modelos convolucionales, por lo que se utilizó la arquitectura de la Figura 3.13 modificando el vector de entrada a un vector de forma 1*52 utilizando los formatos de kmeros sinónimos de entrada y el vector disminuido de etiquetas ontológicas para entrenar el modelo. El rendimiento de esta red se muestra en la Figura 3.14 con valores similares a los de la CNN A, CNN C, y CNN D en sus terceras versiones, además de que se realizó la misma prueba de entrenamiento extendido a esta arquitectura, observando resultados similares a los anteriormente observados en la Figura 3.12. Este modelo fue utilizado para el testing y comparación de los modelos.

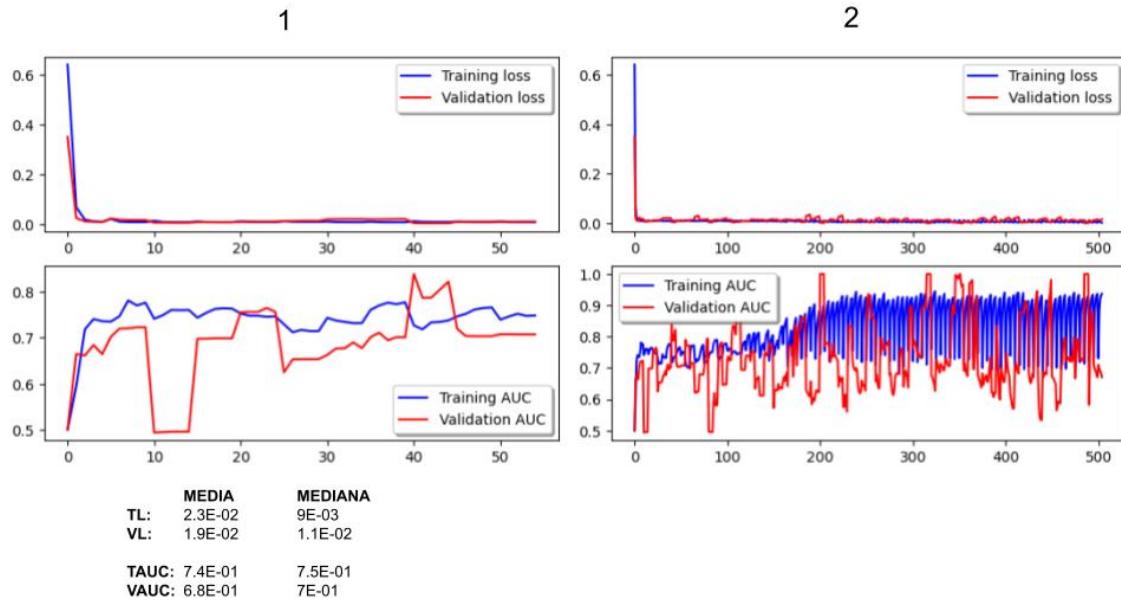


Figura 3.14: Historial de rendimiento de RNN A. 1) Historial de modelo con valores de media y mediana de valores de loss y AUC ROC en entrenamiento y validación. 2) Historial de entrenamiento prolongado de este modelo.

3.2.3. Testing

Se evaluó una muestra de 10308 kmeros para clasificar en cada modelo entrenado, obteniendo los resultados de la Figura 3.15. El modelo con mejor desempeño fue el CNN A, obteniendo un valor de AUC ROC de 0.6454, seguido por la CNN D con 0.6454, aunque la desviación de error de cada modelo tuvo un comportamiento

muestra como mejor modelo el CNN D con un valor de 1.43E-02 seguido por la CNN A con un valor de 1.91E-02.

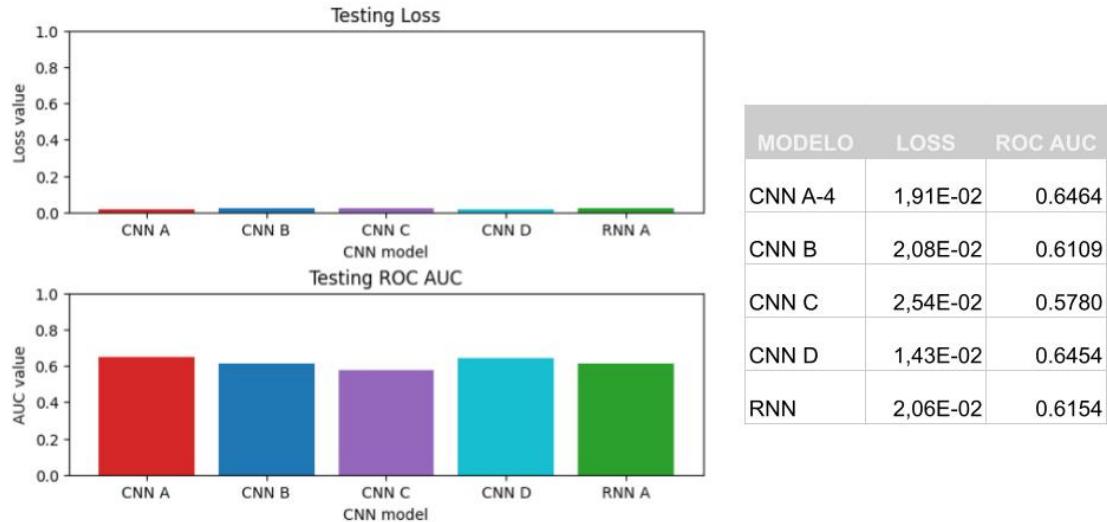


Figura 3.15: Comparativa de desempeño en testing de modelos CNN A, CNN B, CNN C, CNN D y RNN A con misma muestra de datos. Se muestran valores de Loss y de AUC ROC en la tabla del lado derecho.

Ambos modelos se muestran bastante cercanos, y en su arquitectura observamos que la CNN A solo tiene una capa de Dropout, y la CNN D no tiene ninguna, sin embargo el similar comportamiento en comparación con los otros modelos no indica una diferencia significativa que indique overfitting.

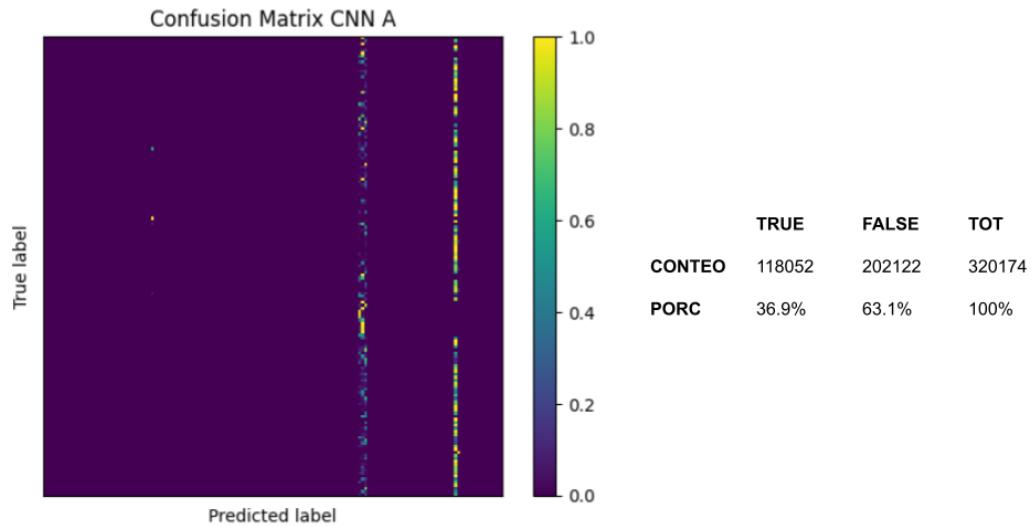


Figura 3.16: Matriz de confusión de modelo CNN A. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.

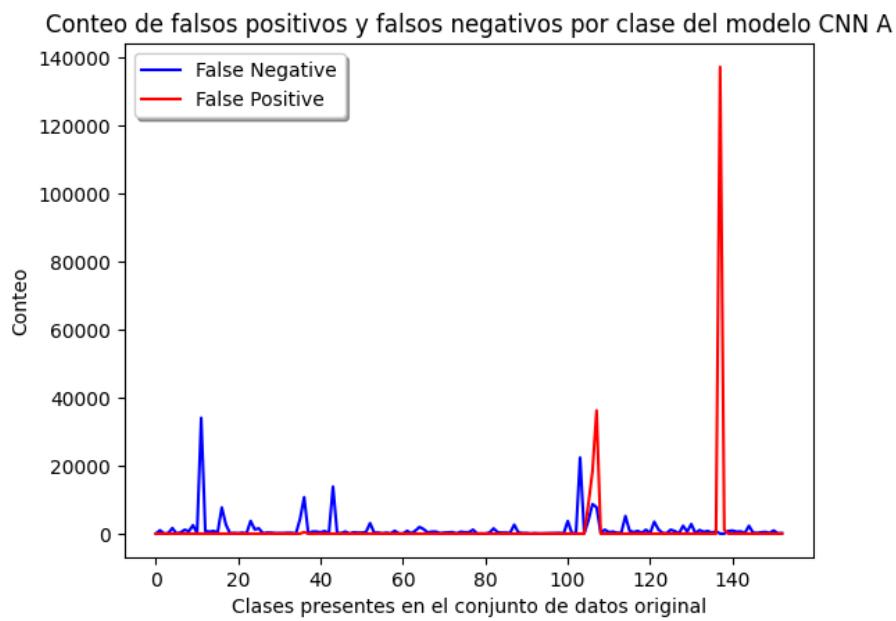


Figura 3.17: Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN A.

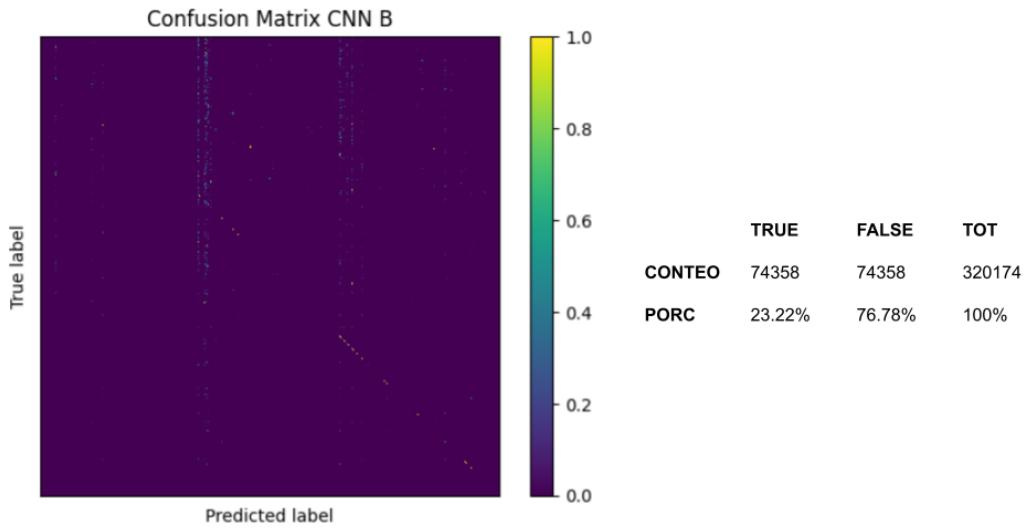


Figura 3.18: Matriz de confusión de modelo CNN B. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.

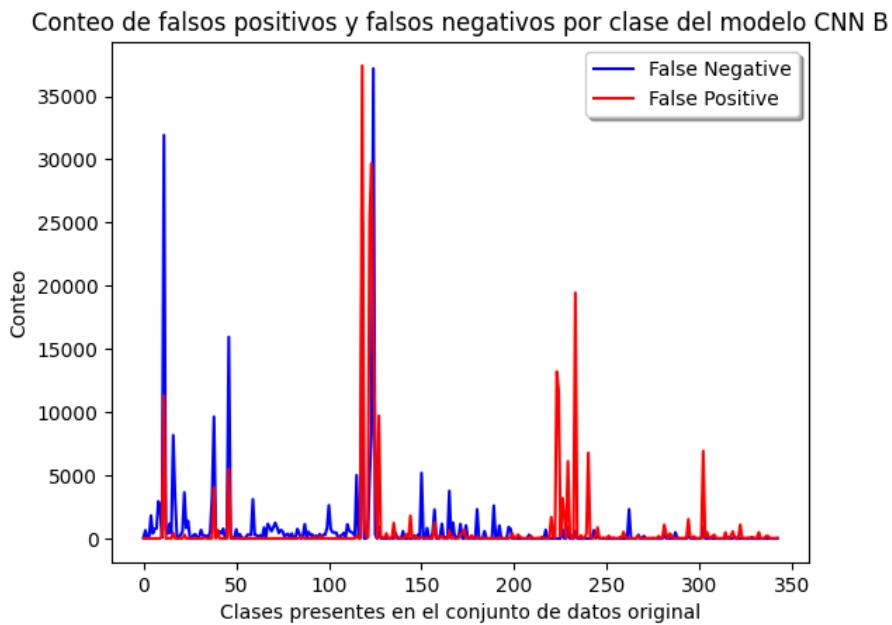


Figura 3.19: Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN B.

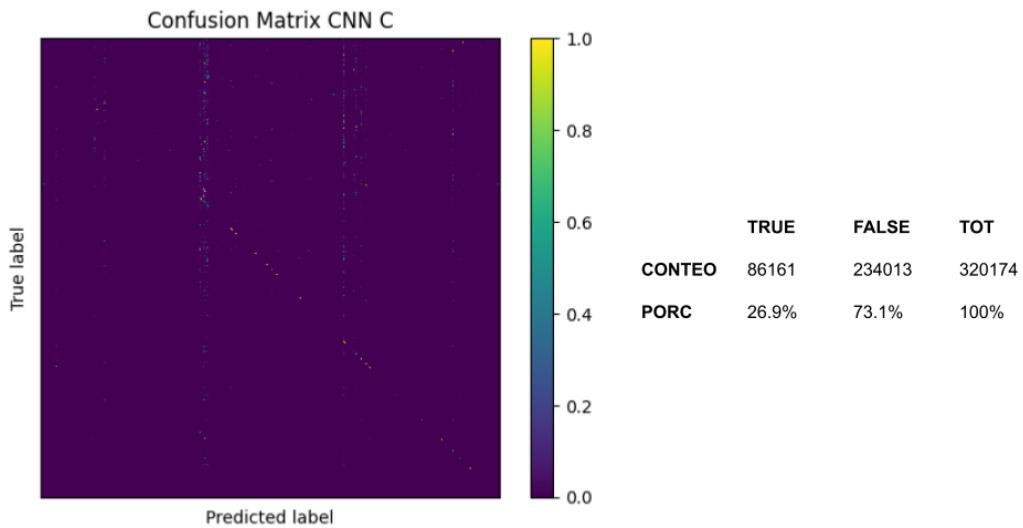


Figura 3.20: Matriz de confusión de modelo CNN C. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.

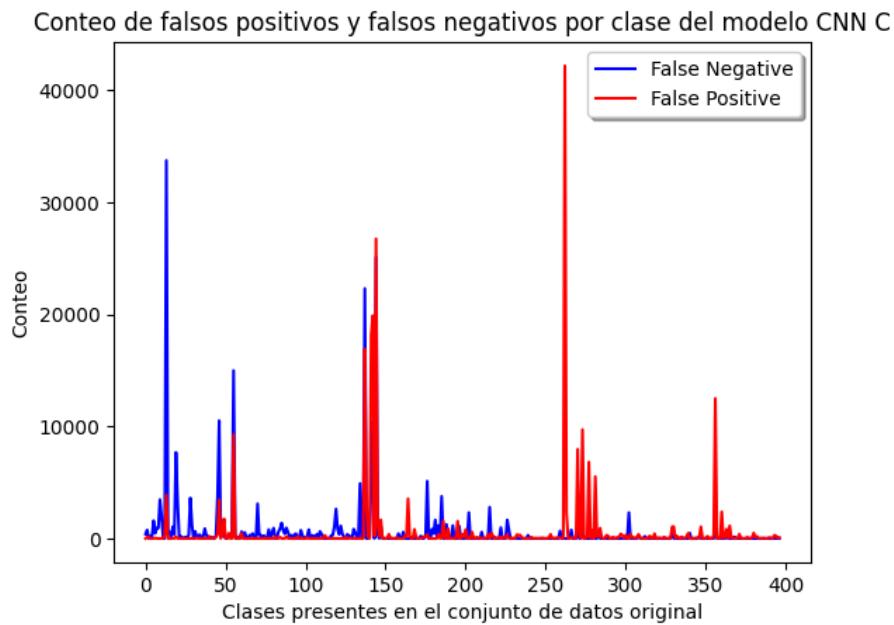


Figura 3.21: Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN C.

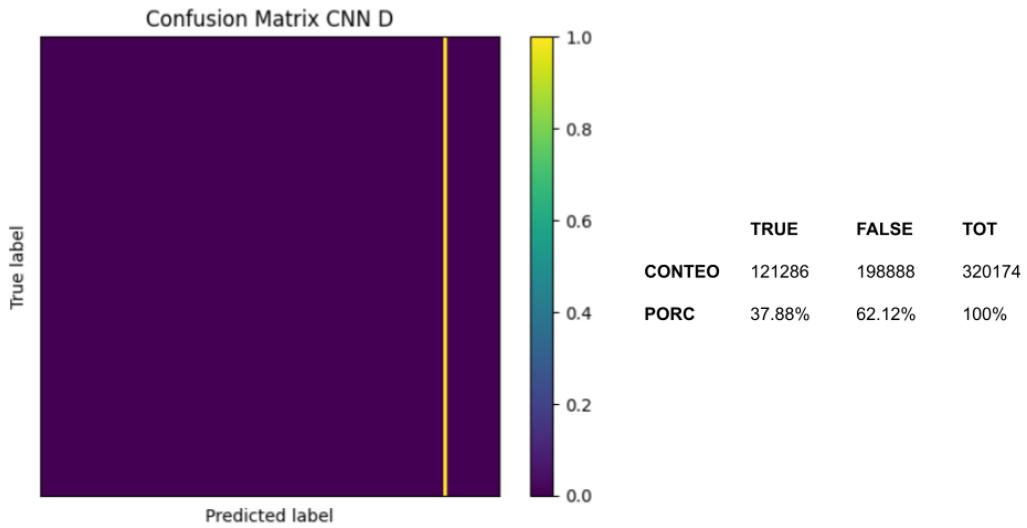


Figura 3.22: Matriz de confusión de modelo CNN D. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.

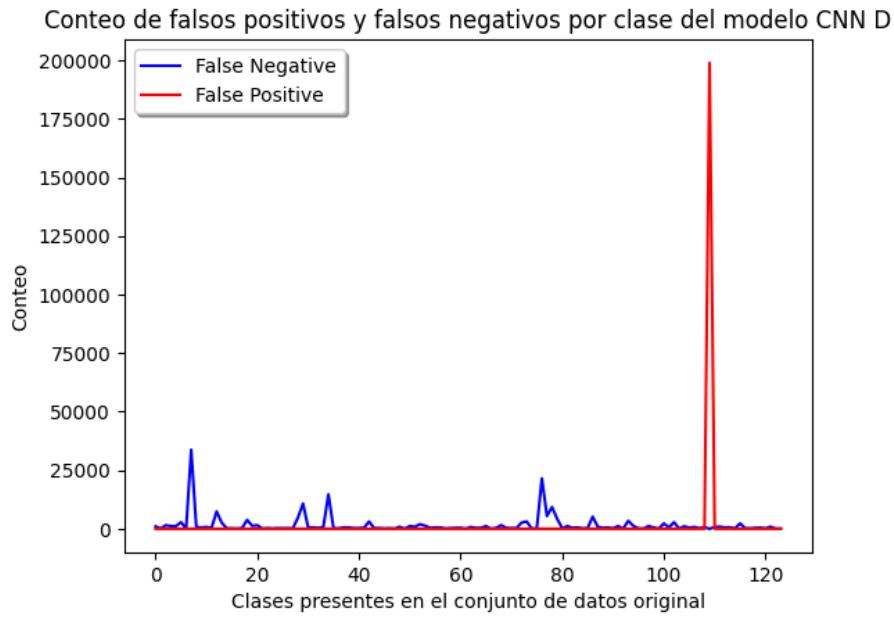


Figura 3.23: Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo CNN D.

Se realizó la evaluación de un conjunto de 320174 datos para cada una de las redes neuronales, evaluando el mismo conjunto en cada red. El vector resultante de

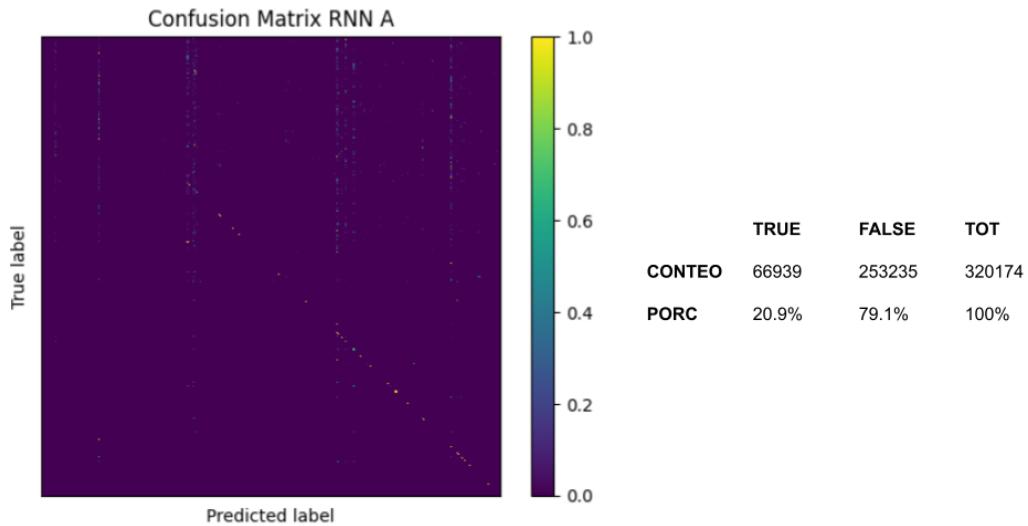


Figura 3.24: Matriz de confusión de modelo RNN A. Incluye los valores de la clasificación, categorizados como TRUE aquellos datos cuya clasificación estaba incluida en el conjunto original de ontologías de la proteína a partir de la cual fue originado, y FALSE en los casos donde la etiqueta resultante no figuraba dentro del conjunto original.

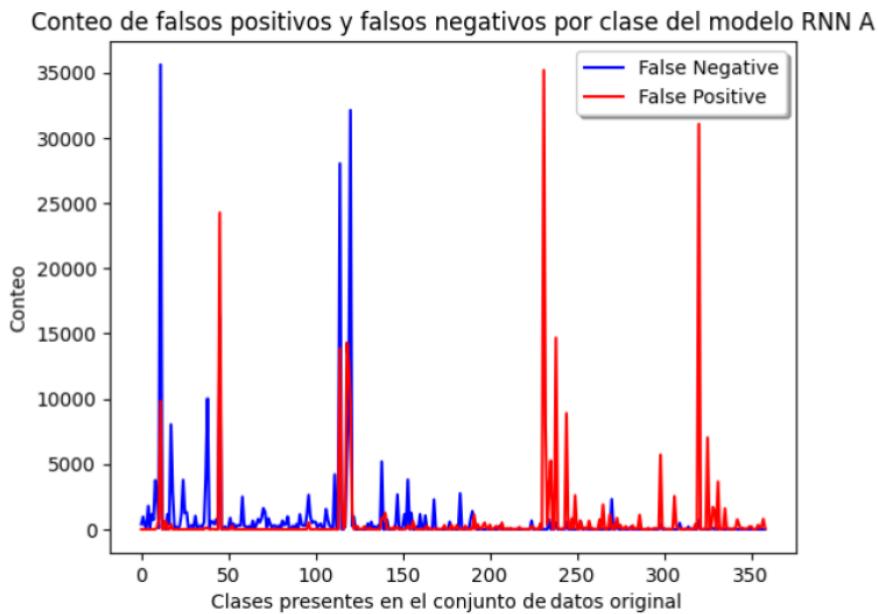


Figura 3.25: Conteo de datos de falsos positivos y falsos negativos para las clases presentes en el conjunto analizado por el modelo RNN A.

esta evaluación es un vector de 1×4206 donde cada uno de los 4206 valores es una probabilidad de pertenencia a cada clase. Para generar las matrices de confusión de

las Figuras 3.16, 3.18, 3.20, 3.22 y 3.24 cada vector de probabilidad se convirtió a un valor único de la probabilidad máxima del vector, obteniendo el numero de clase positiva, mientras que para el eje vertical de las clases originales se evalúa si la clase resultante pertenece al conjunto de ontologías originales de la proteína a partir de la cual se generó el kmero analizado. En caso de que no se encuentre la clase predicha en las clases originales se toma el primer elemento del vector original como la clase verdadera, generando un registro negativo. Del lado derecho de cada matriz hay una barra de escala de colores, donde los colores el púrpura es el valor cero, y el amarillo es el uno, mostrando por escala de colores una referencia a los valores normalizados de los objetos clasificados. Aparentemente las matrices ilustran una clasificación similar para cada modelo dirigiendo todos los datos a unas cuantas clases, comparando la presencia y ausencia de las clases presentes en la base de datos. Esto tiene dos motivos:

1. El conjunto de datos tomado del archivo de testing no es lo suficientemente representativo, por lo que las muestras analizadas tienen una tendencia a un grupo reducido de ontologías.
2. Las ontologías en las que cada modelo clasifica los kmberos no están dentro del conjunto original pero pueden estarlo, ya que el que las proteínas solo tuvieran una ontología registrada no implica que sea la única que pudiera describir la actividad del gen al que corresponde.

Adicional a las matrices de confusión mostradas en las imágenes, también se realizaron conteos de falsos positivos y falsos negativos para cada una de las clases presentes en el conjunto de ontologías original. Con esto es fácilmente distinguible la cantidad de datos dispersos en cada matriz. Estas gráficas de conteo se muestran en las Figuras 3.17, 3.19, 3.21, 3.23 y 3.25, con un total de 343 clases presentes en el conjunto de datos original y a partir del cual se hizo este conteo.

También es notorio en estas matrices que solo la CNN A y la CNN D no muestran la diagonal esperada en estas matrices, sin embargo son las que mayor porcentaje de clasificación de datos TRUE muestran, y es notorio también que son estos modelos los que muestran una línea vertical que es más notoria en la CNN D que en la A, que

pueden indicar una clara tendencia a una clase que tiene una gran aparición, como se mencionaba en la Sección 3.1.2, la etiqueta con más apariciones es la **GO:0016020**, siendo su descripción "membrana como componente celular mostrado en la Figura 3.26, seguida por la etiqueta **GO:0005686**, descrita como "membrana plasmática", derivada de la etiqueta anterior como componente celular, y cuyo diagrama de ancestros se muestra en la Figura 3.27.

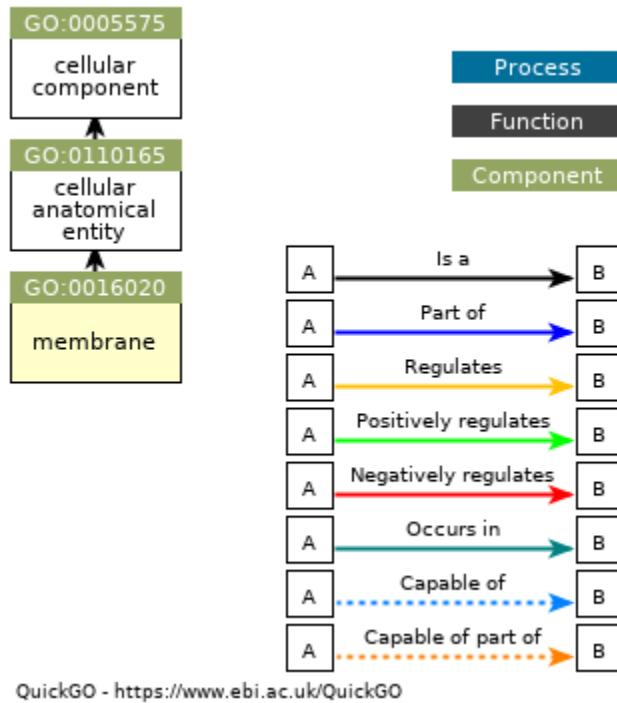


Figura 3.26: Diagrama de ancestros de la ontología GO:0016020. Recuperada de QuickGO[74] el 15 de Noviembre del 2022.

Se realizó un análisis de una muestra de kmeros del archivo de testing. Estos kmeros fueron extraídos del mismo archivo después de un proceso de shuffle, con el objetivo de analizar la etiqueta con mayor número de kmeros clasificados en ella acorde a la probabilidad mayor de su vector de salidas. La muestra utilizada fue de 600 conjuntos de datos, dando un total de 320174 kmeros a procesar, que después de haber sido separados fueron nuevamente mezclados con el procedimiento shuffle. Cada kmero fue clasificado en cada uno de los modelos, y posterior a esto se hizo un conteo de kmeros clasificados por clase presente en el conjunto analizado, generando después diccionarios de frecuencias para cada modelo.

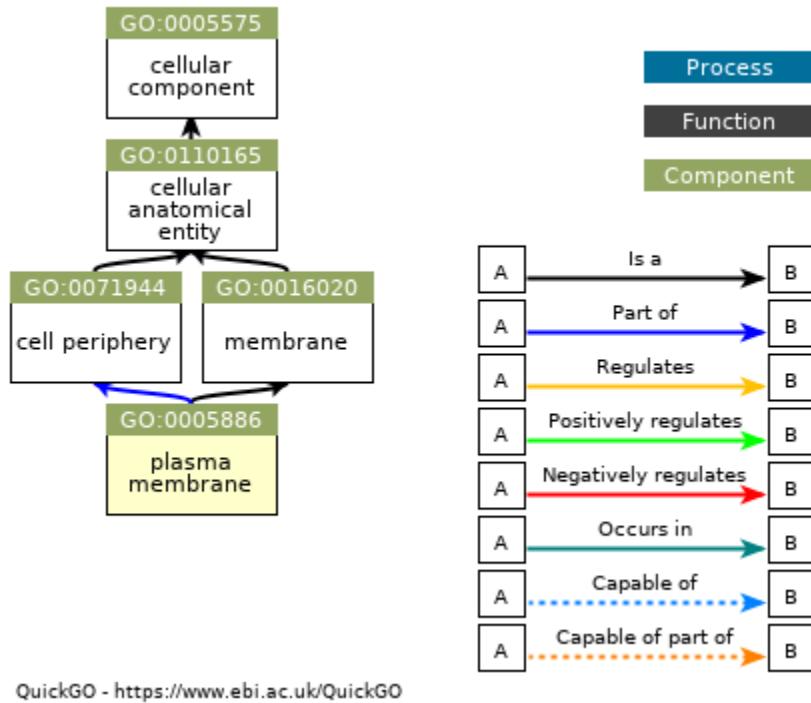


Figura 3.27: Diagrama de ancestros de la ontología GO:005886. Recuperada de QuickGO[73] el 16 de Noviembre del 2022.

La etiqueta con mayor cantidad de kmeros clasificados para cada modelo se muestra en la Tabla 3.1, mostrando por cada modelo la etiqueta con más cantidad de kmeros, el conteo de kmeros clasificados y el porcentaje de kmeros con respecto a los datos ingresados. La última columna es la cantidad de veces que la etiqueta aparecía en la base de datos, siendo un valor relativo ya que cada kmero seguía estando asociado al conjunto de ontologías original.

Los modelos CNN A y CNN D fueron los que clasificaron con una mayor frecuencia a los kmeros en la etiqueta GO:0016020, mientras que la CNN B, CNN C y RNN A clasificaron más la etiqueta GO:0005886 que es la segunda más presente en la base de datos de proteínas utilizadas para generar los kmeros.

El que estas ontologías estén tan presentes en la base de datos y en los kmeros clasificados habla de que las proteínas incluidas en la base de datos en su mayoría eran proteínas de membrana, por lo que realmente estos resultados no son sorprendentes.

Aunque los datos de testing fueron generados a partir de una fracción de los kmeros desordenados generados de cada proteína, y después se mezclaron para tomar una

Modelo	Ontología predicha más veces	Número de predicciones	Porcentaje de kmeros	Aparición en base de datos
CNN A	GO:0016020	98545	30.01	133022
CNN B	GO:0005886	64612	19.68	117762
CNN C	GO:0005886	61858	18.84	117762
CNN D	GO:0016020	133022	40.52	133022
RNN A	GO:0005886	117762	35.87	117762

Tabla 3.1: Etiqueta ontológica con más apariciones en predicciones de cada modelo. Los porcentajes se determinaron con respecto a la cantidad de datos ingresados a cada modelo.

muestra de conjuntos sinónimos, no se esperaba que el modelo CNN D y el RNN A hubieran clasificado la misma cantidad de kmeros en sus respectivas clases más pobladas que la cantidad de kmeros ingresados con esa etiqueta. Podría deberse a que las arquitecturas utilizadas usaban módulos de memoria, sin embargo la CNN C también usaba módulos de memoria y no presentó ese comportamiento que para las CNN D y RNN A solo mostró en la clase de mayor predominancia.

Esto también representa que ambos modelos no tienen una capacidad de distinguir entre clases positivas, clasificando una gran cantidad de kmeros a las clases de mayor conteo de aparición del conjunto original. Esto se muestra en las Figuras 3.17 Y 3.23 con una alta cantidad de falsos positivos solo en las clases mencionadas, con lo que entendemos que el modelo no está clasificando los kmeros por patrón sino por memoria. Estos modelos resultan no ser confiables para analizar información de una base de datos y mostrar resultados enriquecedores, a diferencia de los modelos CNN B, CNN C y RNN A que muestran mayor distribución en sus cantidades de falsos positivos en las Figuras 3.19, 3.21 y 3.25 para cada modelo respectivamente. Esto de igual manera debería revisarse con una base de datos ya etiquetada para comprobar que realmente los modelos muestran información acertada.

Los falsos positivos también son importantes en este análisis, ya que como se mencionó previamente, cada kmero con el que se entrenó el modelo fue ingresado como multi etiqueta, con el objetivo de obtener la etiqueta más representativa para cada kmero por lo que el conteo de falsos negativos debería reflejar las clases excluidas

por kmero, fenómeno visible en los modelos CNN B, CNN C y RNN. Esto concluye que estos modelos pueden ser más capaces de extraer información de un kmero ingresado y mostrar la etiqueta de mayor probable clase para su patrón.

Otra posible razón por la que las matrices de confusión muestran altos valores en falsos positivos es debido a que el vector de clases predichas toma como argumento el valor más alto del vector de confusión, pero esto no significa que ese valor sea suficientemente alto para determinar que el kmero puede representar esa función, aunque esto no significa que el vector no pueda estar asociado a que se presente la función en la que se clasificó, ya que puede ser un vector de conexión entre vectores representativos.

3.3. Clasificación de kmberos

Se generó un algoritmo para concatenar letras de un glosario de las 4 letras representantes de los nucleótidos, en 13 posiciones para generar todas las combinaciones posibles. La base de datos de kmberos con la que se hizo el entrenamiento y testing tenía 66'129,041 kmberos de los 67'108,864 de kmberos posibles por combinación, representando un 98.54 % de los kmberos que pueden existir por concatenación de nucleótidos para este tamaño.

Este conjunto de kmberos se convirtió a vectores de OHE acorde a la entrada de cada modelo, y posterior a esto se evaluaron en cada modelo para generar dos archivos, uno en el que se preserva el vector de probabilidades de pertenencia de cada clase para cada kmero, y otro en el que solo se mantiene el argumento máximo del vector, guardando el índice de la clase como resultado de la clasificación. Con esto es posible analizar la probabilidad de pertenencia y con ello poder hacer ensamblado de secuencias que incluyan estos kmberos.

Capítulo 4

Conclusión

Cualquier modelo de deep learning es capaz de asociar información a través de los algoritmos de su arquitectura por medio de descriptores, y en este caso los descriptores seleccionados fueron las ontologías génicas que, como se mencionó en el capítulo de Antecedentes han sido utilizadas a través de distintos algoritmos como homología de secuencias o la verificación en laboratorio húmedo para la descripción de proteínas por medio de estas etiquetas. También han sido usadas para describir productos y asociar funciones por medio de algoritmos de deep learning y este proyecto representa una posibilidad de uso de las etiquetas ontológicas, analizando patrones intrínsecos a secuencias de DNA codificante de *E. coli* en este caso, pero aplicable a bacterias por tamaño de kmero y por variedad de secuencias analizadas. Con esta investigación se abre la posibilidad del uso de modelos de deep learning para analizar secuencias de DNA de distintos organismos, asociando funcionalidades a patrones de secuencias de DNA que no exclusivas de secciones codificantes, utilizando los mismos descriptores ontológicos y con la posibilidad de disminuir la limitante de descripción de secuencias de DNA a través de sus productos por medio algoritmos que son ajustables a la necesidad de cada experimento.

Capítulo 5

Perspectiva a futuro

Todos los productos de esta tesis, incluidos los códigos, los modelos y las bases de datos utilizadas se disponen a uso del Laboratorio de Biología teórica y Sistemas de la Universidad de Guadalajara, siendo aplicables a otros experimentos para su uso directo o como referencia para otros proyectos.

Este modelo puede ser utilizado de manera directa para comparar las funcionalidades descritas en dominios proteicos de bacterias con las ontologías resultantes del análisis de las secuencias de DNA de dichos modelos, con el fin de comparar la descripción de funcionalidades de cada modelo con los dominios analizados, aunque los modelos pueden estar limitados por la base de datos usada, pudiendo no ser capaz de describir ciertas funciones por las ontologías ausentes en la base de datos con la que se entrenaron los modelos.

Además de esto los algoritmos utilizados en los códigos de este proyecto pueden ser útiles para generar un algoritmo de retrotraducción que por sí mismo ya se diferencia de algoritmos existentes de manera pública, descritos anteriormente como parte de las funcionalidades del EMBL, con sus respectivos ajustes al código genético utilizado según el organismo de donde se obtengan las secuencias de proteína y las particularidades del usuario, permitiendo la generación de bases de datos de secuencias de DNA asociadas a la secuencia original de proteína sed donde se obtuvo.

Incluyo la opción del uso de estos modelos para categorizar kmeros por sus funcionalidades y el uso de los mismos para ensamblar secuencias de DNA que tengan las

características necesarias para codificar proteínas con las funcionalidades deseadas, posibilitando no solo el diseño de genes sino el diseño de genomas bacterianos a partir del ensamblado de su secuencia.

Otra opción de uso es el análisis de distribución de fragmentos genómicos a partir de las funcionalidades del genoma analizado, que permitiría el análisis de la distribución de las funciones y su relación con otras por medio de conectores lógicos como los que describen el grafo de etiquetas ontológicas de Gene Ontology.

Referencias

- [1] M Kozak. “Comparison of initiation of protein synthesis in prokaryotes, eukaryotes, and organelles”. En: *Microbiological Reviews* 47.1 (mar. de 1983), págs. 1-45. DOI: [10.1128/mr.47.1.1-45.1983](https://doi.org/10.1128/mr.47.1.1-45.1983). URL: <http://dx.doi.org/10.1128/mr.47.1.1-45.1983>.
- [2] Frederick R. Blattner et al. “The Complete Genome Sequence of *Escherichia coli* K-12”. En: *Science* 277.5331 (sep. de 1997), págs. 1453-1462. DOI: [10.1126/science.277.5331.1453](https://doi.org/10.1126/science.277.5331.1453). URL: <http://dx.doi.org/10.1126/science.277.5331.1453>.
- [3] Michael Ashburner et al. “Gene Ontology: tool for the unification of biology”. En: *Nature Genetics* 25.1 (mayo de 2000), págs. 25-29. DOI: [10.1038/75556](https://doi.org/10.1038/75556). URL: <http://dx.doi.org/10.1038/75556>.
- [4] S. McGinnis y T. L. Madden. “BLAST: at the core of a powerful and diverse set of sequence analysis tools”. En: *Nucleic Acids Research* 32.Web Server (jul. de 2004), W20-W25. DOI: [10.1093/nar/gkh435](https://doi.org/10.1093/nar/gkh435). URL: <http://dx.doi.org/10.1093/nar/gkh435>.
- [5] Seringhaus y Gerstein. “Qué es la ontología génica”. En: *Investigación y ciencia* 390 (mar. de 2009), págs. 73-81. URL: <https://dialnet.unirioja.es/servlet/articulo?codigo=2931111>.
- [6] Mavridis y W Ritchie. “3D-blast: 3D protein structure alignment, comparison, and classification using spherical polar Fourier correlations”. En: *Pacific Symposium on Biocomputing* 19908380 (2010), págs. 281-292. URL: <https://pubmed.ncbi.nlm.nih.gov/19908380/>.

- [7] NIH - NCBI. *Bio.Entrez package*. 2010. URL: <https://biopython.org/docs/1.75/api/Bio.Entrez.html>.
- [8] Fassler y Cooper. *BLAST Glossary*. Jul. de 2011. URL: <https://www.ncbi.nlm.nih.gov/books/NBK62051/>.
- [9] Grzegorz M. Boratyn et al. “BLAST: a more efficient report with usability improvements”. En: *Nucleic Acids Research* 41.W1 (abr. de 2013), W29-W33. DOI: [10.1093/nar/gkt282](https://doi.org/10.1093/nar/gkt282). URL: <http://dx.doi.org/10.1093/nar/gkt282>.
- [10] Olga Kelemen et al. “Function of alternative splicing”. En: *Gene* 514.1 (feb. de 2013), págs. 1-30. DOI: [10.1016/j.gene.2012.07.083](https://doi.org/10.1016/j.gene.2012.07.083). URL: <http://dx.doi.org/10.1016/j.gene.2012.07.083>.
- [11] Contreras. *Genes parálogos y ortólogos*. Feb. de 2014. URL: <https://biologia.laguia2000.com/genetica/genes-paralogos-y-ortologos>.
- [12] Galpert Cañizares et al. “Aggregation of Similarity Measures for Ortholog Detection: Validation with Measures Based on Rough Set Theory”. En: *Computación y Sistemas* 18.1 (mar. de 2014). DOI: [10.13053/cys-18-1-2014-016](https://doi.org/10.13053/cys-18-1-2014-016). URL: <http://dx.doi.org/10.13053/cys-18-1-2014-016>.
- [13] The Bioconductor Dev Team. *Escherichia coli full genomes*. 2014. URL: <https://bioconductor.org/packages/release/data/annotation/html/BSgenome.Ecoli.NCBI.20080805.html>.
- [14] Babak Alipanahi et al. “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning”. En: *Nature Biotechnology* 33.8 (jul. de 2015), págs. 831-838. DOI: [10.1038/nbt.3300](https://doi.org/10.1038/nbt.3300). URL: <http://dx.doi.org/10.1038/nbt.3300>.
- [15] Maxwell W. Libbrecht y William Stafford Noble. “Machine learning applications in genetics and genomics”. En: *Nature Reviews Genetics* 16.6 (mayo de 2015), págs. 321-332. DOI: [10.1038/nrg3920](https://doi.org/10.1038/nrg3920). URL: <http://dx.doi.org/10.1038/nrg3920>.

- [16] Jian Zhou y Olga G Troyanskaya. “Predicting effects of noncoding variants with deep learning–based sequence model”. En: *Nature Methods* 12.10 (ago. de 2015), págs. 931-934. DOI: [10.1038/nmeth.3547](https://doi.org/10.1038/nmeth.3547). URL: <http://dx.doi.org/10.1038/nmeth.3547>.
- [17] David R. Kelley, Jasper Snoek y John L. Rinn. “Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks”. En: *Genome Research* 26.7 (mayo de 2016), págs. 990-999. DOI: [10.1101/gr.200535.115](https://doi.org/10.1101/gr.200535.115). URL: <http://dx.doi.org/10.1101/gr.200535.115>.
- [18] Byunghan Lee. *deepTarget: End-to-end Learning Framework for microRNA Target...* Mar. de 2016. URL: <https://arxiv.org/abs/1603.09123>.
- [19] Seunghyun Park. *deepMiRGene: Deep Neural Network based Precursor microRNA Prediction*. Abr. de 2016. URL: <https://arxiv.org/abs/1605.00017>.
- [20] Daniel Quang y Xiaohui Xie. “DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences”. En: *Nucleic Acids Research* 44.11 (abr. de 2016), e107-e107. DOI: [10.1093/nar/gkw226](https://doi.org/10.1093/nar/gkw226). URL: <http://dx.doi.org/10.1093/nar/gkw226>.
- [21] Alhamdoosh et al. *EGSEA - Ensemble of Gene Set Enrichment Analyses*. 2017. URL: <https://doi.org/10.1093/bioinformatics/btw623>.
- [22] Kingma y Ba. “Adam: A Method for Stochastic Optimization”. En: *arxiv* (2017). URL: <https://arxiv.org/abs/1412.6980v9>.
- [23] Karla Alejandra Madrigal-Valverde. “Uso de herramientas para alineación de secuencias y creación de árboles filogenéticos para la determinación de especies”. En: *Revista Tecnología en Marcha* 30.5 (dic. de 2017), pág. 30. DOI: [10.18845/tm.v30i5.3218](https://doi.org/10.18845/tm.v30i5.3218). URL: <http://dx.doi.org/10.18845/tm.v30i5.3218>.
- [24] “The Gene Ontology Handbook”. En: *Methods in Molecular Biology* (2017). DOI: [10.1007/978-1-4939-3743-1](https://doi.org/10.1007/978-1-4939-3743-1). URL: <http://dx.doi.org/10.1007/978-1-4939-3743-1>.

- [25] Contreras-Moreira e Yruela Guerrero. *Alineamiento de secuencias de proteína y filogenias / DIGITAL.CSIC*. Jul. de 2018. URL: <https://digital.csic.es/handle/10261/117608>.
- [26] Gómez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. Mayo de 2018. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/.
- [27] Francis Robert y Jerry Pelletier. “Exploring the Impact of Single-Nucleotide Polymorphisms on Translation”. En: *Frontiers in Genetics* 9 (oct. de 2018). DOI: [10.3389/fgene.2018.00507](https://doi.org/10.3389/fgene.2018.00507). URL: <http://dx.doi.org/10.3389/fgene.2018.00507>.
- [28] Zhen Shen, Wenzheng Bao y De-Shuang Huang. “Recurrent Neural Network for Predicting Transcription Factor Binding Sites”. En: *Scientific Reports* 8.1 (oct. de 2018). DOI: [10.1038/s41598-018-33321-1](https://doi.org/10.1038/s41598-018-33321-1). URL: <http://dx.doi.org/10.1038/s41598-018-33321-1>.
- [29] The Gene Ontology Consortium. “The Gene Ontology Resource: 20 years and still GOing strong”. En: *Nucleic Acids Research* 47.D1 (nov. de 2018), págs. D330-D338. DOI: [10.1093/nar/gky1055](https://doi.org/10.1093/nar/gky1055). URL: <http://dx.doi.org/10.1093/nar/gky1055>.
- [30] Yin et al. “An image representation based convolutional network for DNA classification”. En: *arxiv* (2018). DOI: [10.48550/arXiv.1806.04931](https://doi.org/10.48550/arXiv.1806.04931).
- [31] Brownlee. *How to Develop a CNN for MNIST Handwritten Digit Classification*. Mayo de 2019. URL: <https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist手写数字分类/>.
- [32] Cedeño Muñoz, Zambrano Vega y Oviedo Vayas. “Revisión sistemática de literatura: alineamiento secuencial múltiple aplicado a las proteínas transmembrana”. En: *Revista Ibérica de Sistemas e Tecnologias de Informação* E18 (feb. de 2019), págs. 138-155. URL: <https://www.researchgate.net/profile/>

- [Maria - De - Los - Guaman / publication / 337047963 _ RISTI _ 2019 / links / 5dc2352d92851c8180304694/RISTI-2019.pdf#page=154.](https://www.ncbi.nlm.nih.gov/publication/337047963_RISTI_2019_links/5dc2352d92851c8180304694/RISTI-2019.pdf#page=154)
- [33] Elzanowski y Ostell. *The Genetic Codes*. Ene. de 2019. URL: <https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi>.
- [34] Gökcen Eraslan et al. “Deep learning: new computational modelling techniques for genomics”. En: *Nature Reviews Genetics* 20.7 (abr. de 2019), págs. 389-403. DOI: [10.1038/s41576-019-0122-6](https://doi.org/10.1038/s41576-019-0122-6). URL: <http://dx.doi.org/10.1038/s41576-019-0122-6>.
- [35] Hanyu Zhang et al. “NCNet: Deep Learning Network Models for Predicting Function of Non-coding DNA”. En: *Frontiers in Genetics* 10 (mayo de 2019). DOI: [10.3389/fgene.2019.00432](https://doi.org/10.3389/fgene.2019.00432). URL: <http://dx.doi.org/10.3389/fgene.2019.00432>.
- [36] Ernesto Borrero et al. “Whole-Genome k-mer Topic Modeling Associates Bacterial Families”. En: *Genes* 11.2 (feb. de 2020), pág. 197. DOI: [10.3390/genes11020197](https://doi.org/10.3390/genes11020197). URL: <http://dx.doi.org/10.3390/genes11020197>.
- [37] The UniProt Consortium. “UniProt: the universal protein knowledgebase in 2021”. En: *Nucleic Acids Research* 49.D1 (nov. de 2020), págs. D480-D489. ISSN: 0305-1048. DOI: [10.1093/nar/gkaa1100](https://doi.org/10.1093/nar/gkaa1100). eprint: <https://academic.oup.com/nar/article-pdf/49/D1/D480/35364103/gkaa1100.pdf>. URL: <https://doi.org/10.1093/nar/gkaa1100>.
- [38] Ligdi Gonzalez. *Curvas ROC y Área bajo la curva (AUC)*. Ago. de 2020. URL: <https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/>.
- [39] Ligdi Gonzalez. *Matriz de Confusión*. Ago. de 2020. URL: <https://aprendeia.com/matriz-de-confusion-machine-learning/>.
- [40] Jang. *TensorFlow: MNIST CNN Tutorial*. Jun. de 2020. URL: <https://www.kaggle.com/code/amyjang/tensorflow-mnist-cnn-tutorial/notebook>.

- [41] J. Alejandro Morales et al. “Deep Learning for the Classification of Genomic Signals”. En: *Mathematical Problems in Engineering* 2020 (mayo de 2020), págs. 1-9. DOI: [10.1155/2020/7698590](https://doi.org/10.1155/2020/7698590). URL: <http://dx.doi.org/10.1155/2020/7698590>.
- [42] Moses Stamboulian et al. “The ortholog conjecture revisited: the value of orthologs and paralogs in function prediction”. En: *Bioinformatics* 36.Supplement₁ (jul. de 2020), págs. i219-i226. DOI: [10.1093/bioinformatics/btaa468](https://doi.org/10.1093/bioinformatics/btaa468). URL: <http://dx.doi.org/10.1093/bioinformatics/btaa468>.
- [43] Žiga Avsec et al. “Effective gene expression prediction from sequence by integrating long-range interactions”. En: *Nature Methods* 18.10 (oct. de 2021), págs. 1196-1203. DOI: [10.1038/s41592-021-01252-x](https://doi.org/10.1038/s41592-021-01252-x). URL: <http://dx.doi.org/10.1038/s41592-021-01252-x>.
- [44] Mohanad A. Deif et al. “A deep bidirectional recurrent neural network for identification of SARS-CoV-2 from viral genome sequences”. En: *Mathematical Biosciences and Engineering* 18.6 (2021), págs. 8933-8950. DOI: [10.3934/mbe.2021440](https://doi.org/10.3934/mbe.2021440). URL: <http://dx.doi.org/10.3934/mbe.2021440>.
- [45] Luis Velasco. *Optimizadores en redes neuronales profundas: un enfoque práctico*. Dic. de 2021. URL: <https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5>.
- [46] Tianzhi Wu et al. “clusterProfiler 4.0: A universal enrichment tool for interpreting omics data”. En: *The Innovation* 2.3 (ago. de 2021), pág. 100141. DOI: [10.1016/j.xinn.2021.100141](https://doi.org/10.1016/j.xinn.2021.100141). URL: <http://dx.doi.org/10.1016/j.xinn.2021.100141>.
- [47] Jinny X. Zhang et al. “A deep learning model for predicting next-generation sequencing depth from DNA sequence”. En: *Nature Communications* 12.1 (jul. de 2021). DOI: [10.1038/s41467-021-24497-8](https://doi.org/10.1038/s41467-021-24497-8). URL: <http://dx.doi.org/10.1038/s41467-021-24497-8>.

- [48] Zijun Zhang et al. “An automated framework for efficiently designing deep convolutional neural networks in genomics”. En: *Nature Machine Intelligence* 3.5 (mar. de 2021), págs. 392-400. DOI: [10.1038/s42256-021-00316-z](https://doi.org/10.1038/s42256-021-00316-z). URL: <http://dx.doi.org/10.1038/s42256-021-00316-z>.
- [49] Gabriel Furnieles. *Sigmoid and SoftMax Functions in 5 minutes*. Sep. de 2022. URL: <https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9>.
- [50] Grote. *GOfuncR*. 2022. URL: <https://bioconductor.org/packages/release/bioc/html/GOfuncR.html>.
- [51] Li y Lu. *Red neuronal multiclasa: referencia de componente - Azure Machine Learning*. Sep. de 2022. URL: <https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/multiclass-neural-network>.
- [52] Fábio Madeira et al. “Search and sequence analysis tools services from EMBL-EBI in 2022”. En: *Nucleic Acids Research* 50.W1 (abr. de 2022), W276-W279. DOI: [10.1093/nar/gkac240](https://doi.org/10.1093/nar/gkac240). URL: <http://dx.doi.org/10.1093/nar/gkac240>.
- [53] Rubiales. *Funciones de error con Entropía: Cross Entropy y Binary Cross Entropy*. Ene. de 2022. URL: <https://rubialesalberto.medium.com/funciones-de-error-con-entropia-cross-entropy-y-binary-cross-entropy-8df8442cdf35>.
- [54] Sharma. *4 Proven Tricks to Improve your Deep Learning Model’s Performance*. Jul. de 2022. URL: <https://www.analyticsvidhya.com/blog/2019/11/4-tricks-improve-deep-learning-model-performance/>.
- [55] Brad T Sherman et al. “DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update)”. En: *Nucleic Acids Research* 50.W1 (mar. de 2022), W216-W221. DOI: [10.1093/nar/gkac194](https://doi.org/10.1093/nar/gkac194). URL: <http://dx.doi.org/10.1093/nar/gkac194>.
- [56] SIB. *Swiss-Prot*. Oct. de 2022. URL: <https://www.sib.swiss/swiss-prot>.

- [57] Hongxiao Wang, Hao Zheng y Danny Z Chen. “TANGO: A GO-term Embedding Based Method for Protein Semantic Similarity Prediction”. En: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2022), págs. 1-1. DOI: [10.1109/tcbb.2022.3143480](https://doi.org/10.1109/tcbb.2022.3143480). URL: <http://dx.doi.org/10.1109/tcbb.2022.3143480>.
- [58] Ningyu Zhang et al. “OntoProtein: Protein Pretraining With Gene Ontology Embedding”. En: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=yfe1VMYAXa4>.
- [59] Bethesda (MD): National Library of Medicine. *NCBI FTP site*. URL: <https://ftp.ncbi.nlm.nih.gov/>.
- [60] Bioconductor. *3.16 AnnotationData Packages*. URL: <https://bioconductor.org/packages/3.16/data/annotation/>.
- [61] Gene Ontology Consortium. *AmiGO 2: Search*. URL: <http://amigo.geneontology.org/amigo/search/ontology>.
- [62] *Download Annotations / Gene Ontology Consortium*. URL: <http://current.geneontology.org/products/pages/downloads.html>.
- [63] *Download ontology*. URL: <http://geneontology.org/docs/download-ontology/>.
- [64] *EcoliWiki*. URL: https://ecoliwiki.org/colipedia/index.php/Welcome_to_EcoliWiki.
- [65] EMBL-EBI. *EMBOSS Backtranambig*. URL: https://www.ebi.ac.uk/Tools/st/emboss_backtranambig/.
- [66] EMBL-EBI. *EMBOSS Backtranseq*. URL: https://www.ebi.ac.uk/Tools/st/emboss_backtranseq/.
- [67] EMBL-EBI. *Pairwise Sequence Alignment Tools*. URL: <https://www.ebi.ac.uk/Tools/psa/>.
- [68] *EMBOSS: backtranambig*. URL: <http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/backtranambig.html>.

- [69] European Bioinformatics Institute. *Data resources and tools*. URL: <https://www.ebi.ac.uk/services/data-resources-and-tools?category=proteins>.
- [70] *Guide to GO evidence codes*. URL: <http://geneontology.org/docs/guide-go-evidence-codes/>.
- [71] NIH - NCBI. *DNA substitution matrices*. URL: <https://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html>.
- [72] *Relations in the Gene Ontology*. URL: <http://geneontology.org/docs/ontology-relations/>.
- [73] The Gene Ontology Consortium. *GO:0005886*. URL: <https://www.ebi.ac.uk/QuickGO/term/GO:0005886>.
- [74] The Gene Ontology Consortium. *GO:0016020*. URL: <https://www.ebi.ac.uk/QuickGO/term/GO:0016020>.