Universidad Europea MADRID

TRABAJO FIN DE MÁSTER

Desarrollo del software DeepCloneFinder. Una herramienta para la identificación clonal en muestras de metagenómica mediante Deep Neural Networks

Autor: Sergio Olmos Piñero
Tutor del trabajo: Val Fernández Lanza
Tutor académico: María del Rocío González Soltero

Facultad de Ciencias Biomédicas y de la Salud
Titulación: Máster Universitario en Bioinformática
Curso 2021-2022

Índice

Índice de figuras:

Índice de tablas.

1. Resumen.

La metagenómica aborda la problemática de la caracterización de las comunidades microbianas, para ello, se extrae el material genético de una comunidad de organismos y posteriormente se realiza el análisis de la misma. En la actualidad, son numerosos los métodos existentes para el alineamiento de las muestras con genomas de referencia empleando bibliotecas de genomas bacterianos, pero la cuantificación y estimación de la abundancia relativa de las especies presentes en la muestra es más complejo. Existen herramientas que emplean diferentes estrategias, pero no siempre resultan precisas y eficientes.

El presente trabajo aborda el análisis de comunidades bacterianas sintéticas para obtener un algoritmo capaz de estimar la abundancia relativa y el número de clones presentes para un gen específico empleando *Deep Learning*. Para ello se desarrollará un simulador de comunidades bacterianas y se emplearán para obtener algoritmos predictivos de diferente complejidad. La determinación del número de mutaciones y de alelos presentes en una muestra se consigue con una precisión del >97%, mientras que la estimación de la abundancia relativa y la cuantificación del número de clones presenta limitaciones y la precisión alcanzada es del ~12%. La precisión se está viendo afectada por la similitud entre la composición en proporción de las comunidades sintéticas en las que se presenta codominancia entre dos alelos. Además, se emplean comunidades cuya variabilidad genética es inferior al 10%, lo cual dificulta el proceso de aprendizaje de la red neuronal afectando a su capacidad predictiva.

2. Abstract.

Metagenomics addresses the problem of characterizing microbial communities. To do this, the genetic material of a community of organisms is extracted and subsequently analyzed. Currently, there are many existing methods for aligning samples with reference genomes using bacterial genome libraries, but the quantification and estimation of the relative abundance of the species present in the sample is more complex. There are tools that employ different strategies, but they are not always accurate and efficient.

The present work deals with the analysis of synthetic bacterial communities to obtain an algorithm capable of estimating the relative abundance and the number of clones present for a specific gene using Deep Learning. For this, a simulator of bacterial communities will be developed and used to obtain predictive algorithms of different complexity. The determination of the number of mutations and alleles present in a sample is achieved with a precision of >97%, while the estimation of the relative abundance and the quantification of the number of clones has limitations and the precision reached is ~12%. The precision is being affected by the similarity between the composition in proportion of the synthetic communities in which there is codominance between two alleles. In addition, communities whose genetic variability is less than 10% are used, which hinders the learning process of the neural network, affecting its predictive capacity.

3. Palabras clave.

Metagenómica, Abundancia relativa, Métodos computacionales, Genómica, *Deep Learning*.

Metagenomics, Relative Abundance, Computational methods, Genomics, Deel Learning.

4. Introducción.

4.1 La metagenómica.

La metagenómica es el estudio de una mezcla de material genético extraída de una comunidad de organismos, este tipo de estudios sigue dos tipos de aproximaciones, la realización de un estudio genético completo de todos los individuos presentes en la muestra (*shotgun*) o el más habitual que se focaliza en el gen 16s rRNA (Peterson et al., 2009), el cual contiene 9 regiones variables (V1-V9) que son empleadas para la clasificación filogenética debido a sus bajas tasas de evolución. Esto es así debido a la ambigüedad entre especies y cepas microbianas. A medida que ha crecido la base de datos de genomas bacterianos, cada vez, se descubren nuevas coincidencias entre el material genético de las diferentes especies de microorganismos, en muchos casos, estos genomas son casi idénticos. Ejemplo de ello es la semejanza entre *Mycobacterium Bovis y Mycobacterium tuberculosis,* las cuales comparten hasta un 99.95% del material

genético (Garnier et al., 2003). Su alta similitud, indica que deben considerarse como dos cepas de una sola especie, pero conservan diferentes nombres. Los taxónomos han creado la categoría de *Mycobacterium tuberculosis complex* (Brosch et al., 2002) para presentar una colección de taxones que ahora incluye más de 100 cepas de 5 especies diferentes.

La comunidad de microorganismos a menudo abarca una gama desconcertante de diversidad fisiológica. El análisis basado en funciones y secuencias de fragmentos de ADN metagenómico ha dado como resultado la identificación de nuevos genes y productos génicos (Daniel 2004; Streit et al., 2004). A su vez, la secuenciación parcial de metagenomas, como los del biofilm de la mina ácida (Tyson et al., 2004) y el Mar de Sargazos (Tringe et al., 2005) han proporcionado una mejor comprensión de la estructura de las comunidades microbianas.

Uno de los campos donde la metagenómica ha alcanzado su máximo desarrollo es en el estudio de la microbiota del ser humano, un conjunto de microorganismos que viven en simbiosis con el individuo al que hospedan. La naturaleza de estos individuos es diversa, desde bacterias hasta levaduras y virus. En este tipo de estudios tan sólo se tienen en cuenta las comunidades bacterianas.

El éxito de este tipo de tecnologías de secuenciación masiva para la caracterización de los microorganismos presentes en la muestra radica en la especificidad en la detección la cual se consigue gracias a la presencia de regiones hipervariables muy bien conservadas dentro de cada especie y que permiten realizar la clasificación de los diferentes organismos a un nivel taxonómico.

El proyecto del microbioma humano (HMP), dirigido por los Institutos Nacionales de la Salud (NIH), fue uno de los principales proyectos que evaluó la microbiota del cuerpo humano y produjo alrededor de 35 mil millones de lecturas utilizando datos de 16S rRNA MG de 690 muestras en más de 10 sitios (Peterson et al., 2009). El *American Gut Project* (McDonald et al., 2018) y *Human Intestinal Tract* (Qin et al., 2010) han aumentado significativamente los datos del microbioma humano con composición y función.

Dada la naturaleza heterogénea de las muestras obtenidas tras el empleo de estas tecnologías, se necesitan herramientas analíticas avanzadas para explorar y caracterizar conjuntos de datos biológicos para estudiar la naturaleza peculiar del microbioma, la composición, la función y la heterogeneidad. Esto es crucial para predecir la asociación huésped-microbioma útil en el diagnóstico de enfermedades y en la construcción de estrategias de respuesta para mejorar la salud humana (Hassan., *et* al 2022).

La metagenómica ha abierto una puerta al estudio de los sistemas microbianos directamente de la muestra permitiendo comprenderlos de forma más profunda ya que supone el acceso a información sobre la diversidad metabólica funcional que, sin esta técnica, no sería posible conocer (Hassan*., et* al 2022). Actualmente la tecnología con mayor peso en la aplicación clínica es la PCR rápida, que permite el diagnostico en un tiempo de unas 2H y su manejo es sencillo, la parte negativa es que se emplea para la detección de patógenos que se encuentren frecuentemente. El éxito de la metagenómica sobre estos métodos dependerá del éxito en la combinación de la IA ya que pueden ser efectivos en el diagnóstico temprano de infecciones y la identificación de amenazas patógenas emergentes.

4.2 Inteligencia Artificial y *Deep Learning*.

El aprendizaje automático permite predecir el comportamiento de eventos que no hemos visto, mediante el aprendizaje previo de una máquina a partir de muchos eventos similares, esta disciplina es relativamente reciente y su crecimiento se ve favorecido por el auge de las tecnologías que permiten gran capacidad de computación y la digitalización global que permite el acceso a grandes volúmenes de datos.

En esencia, el funcionamiento de estos sistemas es una emulación del procesamiento y razonamiento humano, se asume que, si se entregan cantidades de datos suficientes, junto con unas reglas definidas, se podrá reconocer patrones y características de los datos, de manera que sea capaz de generalizar y de dar respuesta a problemas similares a los encontrados en los datos aprendidos. Debido a los posibles sesgos que pueden generarse al entrenar el modelo, se debe realizar un trabajo concienzudo sobre los datos que

serán empleados para la etapa de entrenamiento para que el resultado sea efectivo. Para que estos datos tengan sentido, se requiere de las herramientas adecuadas para evaluar los resultados del modelo y su adecuación al problema concreto que se trata de resolver. Es decir, se debe realizar un análisis del comportamiento y rendimiento del modelo, y evaluarlo para decidir si es o no apropiado para el problema. Para determinar el rendimiento de un modelo, se emplean una serie de métricas y estadísticos que ayudan a la toma de decisión y la elección del modelo óptimo.

El *Deep Learning* permite predecir el comportamiento de eventos desconocidos a partir del entrenamiento con casos similares, pero, en concreto, el *Deep Learning* permite obtener modelos capaces de capturar relaciones tanto lineales como no lineales de modos más complejos y sofisticados que el *machine learning* tradicional. Es por ello, que este tipo de IA es la elegida para el desarrollo del software *DeepCloneFinder*, ya que las relaciones no son lineales y dada la heterogeneidad de las muestras a tratas, requiere de un modelo sofisticado.

Para problemas sencillos, se pueden emplear modelos de regresión lineal o logística, naive Bayes, modelos SVM con diferentes *kernels*, de clasificación, como son los diferentes algoritmos de *clustering*, o *Random forest*, estos modelos son conocidos como machine *learning* tradicional o clásico.

Desarrollar un algoritmo que pretenda abordar problemas más complejos requiere del empleo de otro tipo de IA, el *Deep Learning*, en concreto redes neuronales que prometen tener un comportamiento especialmente bueno en datos complejos, que incluyen datos de imagen, audio o video, pero también problemas clínicos o biológicos complejos como los abordados en el presente proyecto.

Para obtener relaciones no lineales, partiendo del modelo clásico en el que se parte de una serie de inputs, que se combinan para producir un output, se debe introducir una o más capas ocultas, también conocidas como *Hidden Layers*, sin embargo, esto no es suficiente, ya que sigue siendo una combinación de funciones lineales y, para alcanzar la solución, en la mayoría de los casos, se requiere de obtener relaciones complejas que no se describen con una función

lineal. Es por ello que a los outputs de una o de varias de las capas ocultas, se les debe aplicar una trasformación no lineal.

Para evaluar grandes conjuntos de datos del mundo microbiano, la IA garantiza un aprendizaje profundo y una comprensión en términos jerárquicos (Agrebi *&* Larbi, 2020). En este ámbito la IA propone una alternativa de menor coste y mayor eficiencia en microbiología clínica y otras áreas de la microbiología para la detección de patógenos empleando tecnologías de metagenómica (Hassan., et al 2022).

A pesar de tratarse de una tecnología relativamente reciente, existen casos de éxito como el empleo de herramientas de IA para diferenciar la diversidad microbiana en personas con enfermedad inflamatoria intestinal (EII) y personas si esta patología, en este estudio realizado en 2012, se emplearon modelos de clasificación supervisada y de *Random Forest*, concluyeron que podría utilizarse como una herramienta de diagnóstico eficaz con un poder predictivo adecuado (Li *&* Qian, 2020).

4.2.1 Funciones de activación de redes neuronales

Al añadir funciones de activación supone un mayor impacto al agregar capas de neuronas ocultas, se puede decir que estas funciones “activan” las neuronas para aprendizaje no lineal. Este apilamiento de relaciones lineales sobre no lineales permite moldear relaciones muy complicadas entre entradas y salidas previstas. En resumen, cada capa está aprendiendo de manera efectiva una función más compleja y de mayor nivel sobre las entradas sin procesar.

Entre las funciones de activación, destacan las siguientes:

- Función sigmoidea. Convierte la suma ponderada obtenida a un valor entre 0 y 1.
- Unidad Lineal Rectificada (ReLU). Una de las funciones no lineales más usadas en los outputs de las capas ocultas, el éxito de esta función radica en que tiene un rango de capacidad de respuesta más útil. La capacidad de respuesta de una función sigmoidea disminuye relativamente rápido en ambos lados, perdiendo valor en los extremos. Esta función toma una función lineal y todos los valores por debajo de cero lo transforma a cero.

- Leaky ReLU. Similar a ReLU, pero los valores por debajo de cero sufren una menor modificación, al aplicarles un factor de corrección, lo que previene el descarte masivo de outputs característico de la función ReLU, manteniendo los valores por debajo de cero, pero otorgándoles un menor peso.
- Unidad Lineal Exponencial (ELU). Función de unidades lineales exponenciales, es similar a ReLU, pero los valores por debajo de cero siguen una funcione exponencial típica mediante transformación exponencial.
- Otras funciones. Además de las desarrolladas anteriormente, existen otras como son la RReLU, ReLU paramétrica, tangencial, hiperbólica, binaria, etc. Esta gran variedad permite generalizar y adaptarse a una gran variedad de datos.

4.2.2 Inteligencia Artificial aplicada a la metagenómica.

La combinación de estas dos tecnologías está mostrando un profundo impacto en el sistema de salud y se espera que su aplicabilidad gane impulso dado que, se prevé que la microbiota intestinal desempeñe un papel clave en la medicina de precisión (Harper & Topol, 2012; Ma et al., 2022). Las principales ventajas están en la caracterización de los microorganismos que no pueden ser cultivados y se consideran parte oculta del microbioma, esto permite conocer la diversidad funcional, esto puede conseguirse gracias a la interacción de estas tecnologías.

La predicción computacional *ab initio* de genes a partir de ADN microbiano tiene una larga historia y se ha desarrollado empleando varias herramientas para la predicción de genes y la anotación de secuencias genómicas de especies procariotas de forma individual (Hoff et al., 2008). En el campo de la metagenómica, existen algunos métodos principales, por ejemplo, el basado en búsqueda BLAST (Altschul et al., 1990), este enfoque es computacionalmente costoso y es capaz de encontrar nuevos genes, empleando la homología. Otro método desarrollado para este mismo fin es un enfoque heurístico de GeneMark.hmm que deriva del empleo de monocodón adaptando el contenido de GC de una secuencia de entrada (Besemer et al., 2006).

A medida que el coste de la secuenciación ha disminuido, los experimentos de metagenómica del tipo *shotgun* se han empleado cada vez más. En este ámbito se requiere de un método para estimar la abundancia directamente a partir de los datos de lectura sin un procesamiento previo (*Lu* et al., 2017). Se han desarrollado métodos como el elaborado por Lu J et al., *Bracken* (*Bayesian Reestimation of Abundance after Classification with KrakEN*), este método emplea las asignaciones taxonómicas realizadas por *KrakEN*, un clasificador de lectura rápido que puede producir estimaciones precisas de abundancia a nivel de especie y género incluso con múltiples especies casi idénticas.

Han aparecido varios métodos eficientes de alineación de secuencias leídas con una base de datos de genomas microbianos de forma rápida y precisa. Ejemplo de este tipo de softwares, se puede destacar *MegaBlast* (Morgulis et al., 2008) el cual ha liderado este ámbito hasta el desarrollo de *Kraken*, el cual ha aumentado la velocidad de *MegaBlast* 900 veces (Wood & Salzberg, 2014). La secuenciación con una base de datos de referencia de proteínas es un cuello de botella computacional importante en metagenómica, en este sentido, DIAMOND es un algoritmo de código abierto basado en doble indexación 20000 veces más rápido que el estándar de oro BLASTX en lecturas cortas (Buchfink et al., 2015). En cuanto al rendimiento obtenido, las dos mejores herramientas con *Kraken* cuya última versión es *Kraken2* y *MetaPhlAn*, última versión *MetaPhlAn* 3, se emplean para la clasificación de lecturas dentro de metagenomas de conjuntos de datos ambientales o sanitarios (Wright et al., 2022).

El análisis metagenómico a menudo implica estimar la abundancia de la especie en una muestra particular. Aunque no podemos asignar sin tener en cuenta la ambigüedad, es interesante conocer la abundancia relativa de cada especie en la muestra. En este sentido se han desarrollado varias herramientas de software para estimar la abundancia de especies en muestras de metagenómica cómo *GAAS* (Angly et al., 2009), *ConStrains* (Luo et al., 2015), *GASiC* (Lindner & Renard, 2013) o *KrakEN* (Lu et al., 2017). Estas herramientas emplean diferentes estrategias que no siempre resultan precisas y eficientes.

5. Planteamiento de hipótesis y objetivos del proyecto.

Existen numerosas herramientas útiles para identificar las especies presentes en una comunidad alineándolo con un genoma de referencia empleando bibliotecas de genomas bacterianos. Sin embargo, no existe ninguna herramienta capaz de predecir la numero y/o abundancia de distintos clones de una sola especie.

Empleando una matriz de probabilidad que describa la comunidad clonal de una especie a partir del alineamiento de un gen marcador, un algoritmo de Inteligencia Artificial debería establecer las relaciones matemáticas que le permitan cuantificar el número de clones y la abundancia relativa de cada alelo presente en la muestra.

6. Material y métodos.

DeepCloneFinder es un software basado en *Deep Learning (*Código disponible en: https://github.com/OPSergio/DeepCloneFinder.git) que permite identificar la composición clonal (en frecuencia). Esta herramienta en un primer modelo debe de ser capaz de identificar el número de mutaciones. Para ello se desarrollará un modelo que se denominará simple, se generará una base de datos con muestras de comunidades sintéticas, de las cuales se extraerán las principales características que se emplearán como *input*, estas son el número de mutaciones, el número de alelos junto con la matriz asociada, finalmente se irán refinando los hiperparámetros modificando la red neuronal.

En un modelo más complejo, se generará una base de datos, con un total de 1000 comunidades sintéticas, se extraerán las características principales, en este caso el número de mutaciones y los alelos presentes en las muestras para obtener un algoritmo capaz de clasificar las muestras en función del estas características.

Este modelo requiere de la inclusión de ruido, por lo que, en un modelo final, se introducirá de forma deliberada, muestras que generen una disminución en la precisión del modelo. La finalidad es realizar una aproximación lo más realista posible para garantizar el buen rendimiento en la práctica.

Finalmente, se desarrollará un modelo final, el cual debe tener la capacidad de predicción del número de clones pertenecientes a cada alelo y su frecuencia relativa en la muestra.

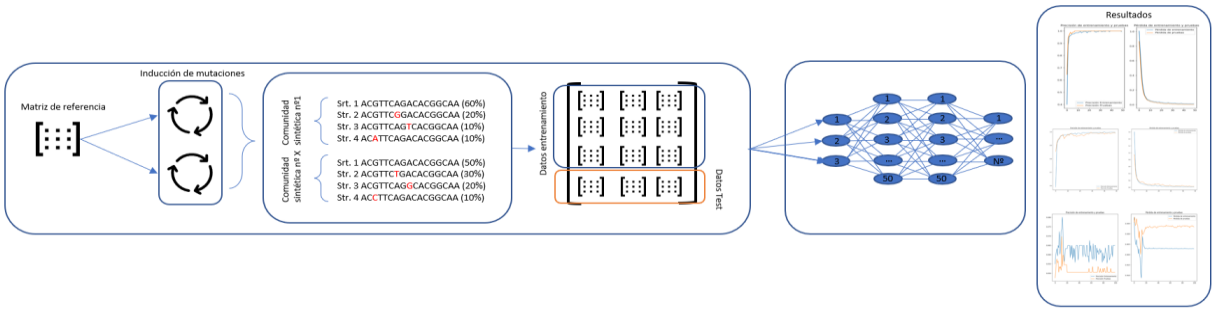

Figura 1 Diagrama de trabajo para el desarrollo del software DeepCloneFinder. Fuente: Elaboración propia.

Para llevar a cabo el entrenamiento del modelo, se partirá de una matriz molde, obtenida de la traducción de una secuencia FASTA de la base de datos *Gene* del *National Center for Biotechnology Information* (NCBI) a una matriz de probabilidad.

Para el desarrollo de *DeepCloneFinder*, se requiere de gran cantidad de datos y para ello se debe obtener un simulador que los genere y cumpla con los requisitos preestablecidos.

6.1 Simulador de comunidades sintéticas.

La finalidad de este simulador consiste en, partiendo de una matriz molde, la cual se obtiene como se detalló anteriormente, obtener una base de datos heterogénea, simulando un número de muestras significativo de comunidades sintéticas adaptado para cada modelo.

Para ello, el simulador de comunidades debe ser capaz de, a lo largo de la matriz de referencia, inducir un número determinado de mutaciones en posiciones totalmente aleatorias, a su vez, debe de generar el número deseado de alelos que compondrán la comunidad simulada. Estas características serán extraídas y facilitadas como inputs a la red neuronal.

Para llevar a cabo el simulador, se emplearán las siguientes librerías:

Pandas: Proporciona herramientas que permiten la lectura de datos en diferentes formatos, la selección y filtrado de datos, así como la unión o fusión de los mismos (McKinney et al., 2010).

NumPy: Da soporte para la creación de vectores y matrices de grandes dimensionalidades, además de la recopilación de funciones matemáticas de alto nivel para operar con ellas (Harris et al., 2020).

Random: Contiene una serie de funciones para generar números pseudoaleatorios (Van Rossum 2020).

En un primer paso, se lee la matriz de probabilidades original (16S rRNA de *Pseudomonas aeruginosa*) y a partir de ella, se genera un array de NumPy para poder operar y trabajar con las diferentes matrices.

Se diseña un bucle anidado, el cual será el producto de dos bucles *for,* el bucle interno, será el encargado de generar las mutaciones, de modo que el número de vueltas será igual al número de mutaciones producidas en la secuencia original.

Para ello las instrucciones definidas implican la generación de una posición aleatoria de la matriz en la que se sustituirá el valor predefinido, para que no se produzcan dobles conteos en una misma posición del gen, eliminando el conteo perteneciente a la secuencia original, sustituyéndolo por el conteo inducido generando el efecto que produciría una mutación sobre la secuencia original.

El bucle externo, se emplea para obtener y guardar la matriz final de probabilidad, de modo que el número de vueltas se corresponde con el número total de matrices generadas y las instrucciones definidas para el mismo implican la compilación de las matrices y su división entre el número de alelos generado. De modo que la matriz resultante, contendrá la probabilidad de cada base nitrogenada en cada posición del gen perteneciente a la comunidad sintética generada.

6.2 Base de datos.

Librerías empleadas para el tratamiento de los datos previo al entrenamiento no definidas en apartados anteriores:

Os: Módulo que permite acceder a funcionalidades dependientes del Sistema Operativo, principalmente aquellas que otorgan información acerca del entorno del mismo y permiten manipular la estructura de directorios.

6.2.1 Base de datos para el Modelo Simple.

Se desarrolla un primer modelo para la predicción del número de mutaciones en una secuencia cuya longitud es de 200 pares de bases.

Para el entrenamiento de este primer modelo, se emplearán 100 muestras para cada categoría. Para la obtención de estos datos, se recurrirá al simulador de comunidades desarrollado y explicado en pasos anteriores.

Se genera un directorio para cada grupo de muestras, en ellos se guardan las matrices de probabilidad de las comunidades generadas para el entrenamiento. Una vez realizado este paso, se requiere del agrupamiento de todos los datos puesto que *TensorFlow* empleará una única matriz que comprenda todos datos.

Se genera un bucle que recorra el directorio en el que se encuentran los datos, al alojarse en directorios diferentes, se repite el proceso para cada grupo de muestras. Terminado el proceso se obtienen tres matrices de dimensión (100,4,200), se agrupan en una única (300,4,200) que constituirá la base de datos que se empleará para el entrenamiento del primer modelo.

Otro de los parámetros requeridos por *TensorFlow* es un vector que incluya las etiquetas de los datos.

Se procede a generar las etiquetas para cada categoría y del mismo modo, se concatenan los vectores con las etiquetas. Se obtiene un vector (1,300), el cual contiene las etiquetas que se entregarán como *input* al modelo.

Finalmente, se dividen los datos de modo que el 70% de ellos serán empleados para el entrenamiento del algoritmo y el 30% restante se empleará para testear la precisión del mismo. Esta división no es el único tratamiento que se da a los datos, se debe aleatorizar el orden puesto que, en la matriz generada, se encuentran ordenados por grupos y existe el riesgo de que la inteligencia artificial aprenda los datos ordenados y no sea útil.

6.2.2 Base de datos Modelo avanzado.

Se desarrolla un algoritmo capaz de predecir tanto el número de mutaciones, como el de número de alelos de una muestra. Para ello, se dará como dato de entrada, una matriz de probabilidades que contenga, la probabilidad de cada

base nitrogenada por posición del gen de un conjunto de comunidades sintéticas. De modo que la base nitrogenada cuya probabilidad se encuentre en mayor proporción, corresponderá a la secuencia inalterada o alelo dominante (mayor abundancia relativa) y las mutaciones se encontrarán en menor proporción en función de su frecuencia en la muestra.

Para ello se debe obtener una base de datos con muestras representativas de cada categoría y para la obtención de estas muestras, se recurre al simulador de comunidades desarrollado para la consecución del proyecto.

Concretamente, la base de datos estará compuesta por 100 muestras para cada una de las nueve categorías. Cada categoría se compone por muestras independientes. En la *Tabla 1* se recogen las categorías con sus respectivas características. Los datos abarcan desde el gen sin mutar, hasta cuatro alelos con tres mutaciones.

		Alelos			
		α	α y β	α, β y γ	α, β, γ y δ
N° mutaciones	1	-	0	3	6
	2	-	1	4	7
	3	-	2	5	8
	0	9	-	-	-

Tabla 1 Características de cada categoría de datos con su respectiva etiqueta. Fuente: elaboración propia.

Se repite el proceso desarrollado en el anterior apartado. La base de datos resultante será de 1000 muestras agrupadas en una única matriz de dimensión (1000,4,200).

La inclusión de ruido para aumentar la precisión del modelo en un entorno real, se lleva a cabo con la herramienta empleada normalmente en clasificación de imagen *ImageDataGenerator,* concretamente, las modificaciones se realizan sobre los datos de entrenamiento y consisten en, desplazamientos horizontales, zooms y modificaciones de brillo, lo que, para el caso, implica modificaciones en la composición de la muestra en porcentaje, y alteraciones del gen expresado.

6.2.3 Base de datos Modelo Final.

La finalidad de la creación de esta base de datos es obtener una base de datos heterogénea de comunidades bacterianas sintéticas empleando el simulador de comunidades desarrollado dentro del marco del proyecto. Las comunidades simuladas serán de diversa naturaleza, en ellas se presentarán hasta cuatro alelos, con un número determinado de clones.

		Diccionario de categorías							
		Alelo α		Alelo β		Alelo γ		Alelo δ	
Categoría	Número de individuos	Nº de clones	Frecuencia relativa	Nº de clones	Frecuencia relativa	Nº de clones	Frecuencia relativa	Nº de clones	Frecuencia relativa
0	1	1	1,00	0	0,00	0	0,00	0	0,00
1	2	1	0,50	1	0,50	0	0,00	0	0,00
2	3	2	0,67	1	0,33	0	0,00	0	0,00
3	3	1	0,33	1	0,33	1	0,33	0	0,00
4	4	3	0,75	1	0,25	0	0,00	0	0,00
5	4	2	0,50	1	0,25	1	0,25	0	0,00
6	4	1	0,25	1	0,25	1	0,25	1	0,25
7	5	4	0,80	1	0,20	0	0,00	0	0,00
8	5	3	0,60	2	0,40	0	0,00	0	0,00
9	5	3	0,60	1	0,20	1	0,20	0	0,00
10	5	2	0,40	1	0,20	1	0,20	1	0,20
11	6	5	0,83	1	0,17	0	0,00	0	0,00
12	6	4	0,67	1	0,17	1	0,17	0	0,00
13	6	3	0,50	3	0,50	0	0,00	0	0,00
14	6	3	0,50	2	0,33	1	0,17	0	0,00
15	6	3	0,50	1	0,17	1	0,17	1	0,17
16	6	4	0,67	2	0,33	0	0,00	0	0,00

Tabla 2 Número de clones y frecuencia relativa de cada alelo presente en la comunidad, en función de la categoría de las muestras. Fuente: Elaboración propia.

En la *tabla 2*, se presentan las categorías a las que pertenecen los datos empleados para el desarrollo del algoritmo. Para todas las categorías, el número de muestras es n=200, salvo para la categoría 0, para esta categoría el número de muestras es n=100. Las dimensiones de la base de datos generada son de (3300,4,200).

6.3 Estructura de la red neuronal.

Librerías empleadas para el diseño y desarrollo de la estructura de la red neuronal.

Keras: Biblioteca de redes neuronales artificiales de código abierto (Chollet 2015).

TensorFlow: Librería de código abierto para *Machine Learning*, permite construir y entrenar redes neuronales para detectar patrones (Abadi et al., 2016).

6.3.1 Capa de entrada.

El primer componente a definir es la capa de entrada, dada la naturaleza de los datos empleados, las dimensiones de esta capa se corresponderán con las de las matrices generadas (4,200). Cada neurona de esta capa se encargará de

almacenar el valor contenido de una posición de la matriz, es por ello que, si la matriz tiene un total de 800 datos, se requiere una capa de entrada de 800 neuronas. La finalidad de este tipo de red neuronal es el análisis individual de cada posición de la matriz, este hecho es interesante para el desarrollo de *DeepCloneFinder* puesto que no existe una relación matemática entre las diferentes posiciones de la matriz y es estudiando cada posición de forma individual, la forma en la que se encuentra la relación entre los alelos presentes en la muestra.

6.3.2 Capas ocultas.

Ambos modelos poseen una estructura idéntica, se componen por un número de capas internas de neuronas de tipo denso que variará en función de las necesidades del algoritmo, estas neuronas establecen relaciones entre sí, de modo que quedan interconectadas. El número de neuronas es inicialmente de 50 neuronas cada capa. La función de activación es ReLU, Unidad Lineal Rectificada Una de las funciones no lineales más usadas en los outputs de las capas ocultas, el éxito de esta función radica en que tiene un rango de capacidad de respuesta más útil. La capacidad de respuesta de una función sigmoidea disminuye relativamente rápido en ambos lados, perdiendo valor en los extremos. Esta función toma una función lineal y todos los valores por debajo de cero lo transforma a cero. Esa transformación puede desembocar en el problema de la muerte de algunas de las neuronas.

Tras la realización del entrenamiento, se observa que el modelo requiere de un mayor número de neuronas, por lo tanto, se aumenta el número a 100 neuronas cada capa.

Tras realizar la comparación con el resto de las funciones de activación, la empleada para el software de clasificación desarrollado es la ReLU.

6.3.3 Capa de salida.

La capa de salida está compuesta por un número dependiente de neuronas, el número será igual al número de categorías contenidas en la base de datos. La función de activación es *softmax*, esta función es la más empleada para algoritmos de clasificación. Si se tratase de una clasificación binaria, la función

más adecuada sería la sigmoidea. Al contener datos pertenecientes a más de 2 categorías, la elección de esta función está justificada.

6.3.4 Compilación de la red neuronal y entrenamiento.

Como optimizador se emplea la función *Adam,* como función de pérdida, se emplea la función *SparseCategoricalCrossentropy*.

El número de épocas se decide en función de los resultados obtenidos, en un primer modelo, se observa que la pérdida por época desciende a prácticamente cero entre las 40 y las 50 épocas, sin embargo, el valor de la precisión en la predicción sufre oscilaciones hasta las 40 épocas. Por lo tanto, para el modelo complejo, se emplea un número superior a 50 épocas. En el caso del modelo final el número de épocas se define en 100 épocas.

7. Resultados.

7.1 Modelo simple.

El desarrollo de este modelo tiene como principal finalidad extraer una serie de características de la comunidad sintética objeto de estudio, concretamente el número de mutaciones presentes en la comunidad. A su vez, se trata de un algoritmo relativamente sencillo que se emplea para comprobar la viabilidad de desarrollar algoritmos con un grado de complejidad superior.

En la *figura 2* se muestran los resultados obtenidos tras 50 épocas de entrenamiento del algoritmo desarrollado para predecir el número de mutaciones en muestras con 2 alelos para el gen 16S rRNA de *pseudomonas aerugionsa.*

Como se observa en la *Figura 2* la precisión inicial es próxima a 0.4, en el transcurso de 10 épocas, el modelo es tiene una precisión superior a 0.98, desde ese instante hasta el final del entrenamiento, se mantiene constante la precisión del modelo.

La tasa de pérdida por época decrece de forma exponencial hasta las 10 épocas, desde ese instante hasta el final del entrenamiento, esta función de pérdida se reduce, lo que significa que el modelo reduce la tasa de aprendizaje, esto se solapa con la pendiente de la precisión, la cual, como se ha explicado anteriormente, se reduce completamente a partir de la época 10.

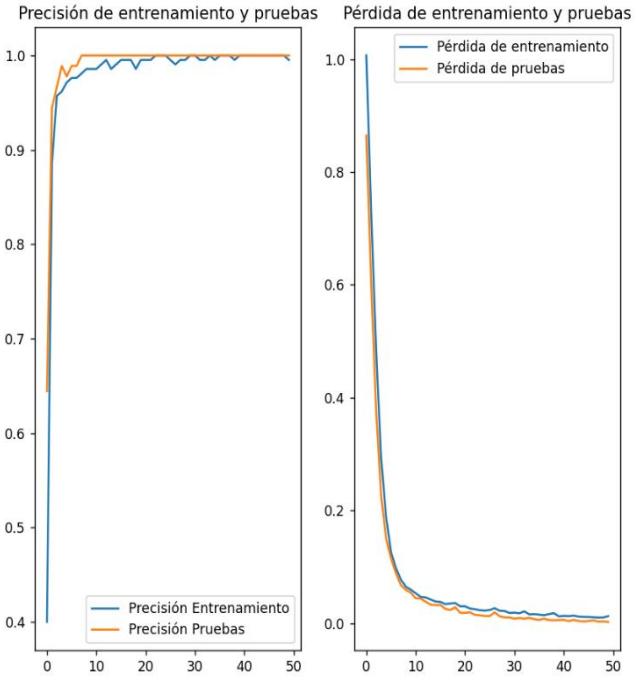

Figura 2 Parámetros del modelo. Se representa la precisión y la pérdida de entrenamiento frente al número de épocas.

El resultado obtenido por el modelo con las pruebas al final del entrenamiento es próximo a 98%, a lo largo de las épocas de entrenamiento, se producen oscilaciones producto de los diferentes ajustes de los pesos entre las neuronas. Por ello se determina que la capacidad de predicción del modelo es alta.

7.2 Modelo Avanzado.

En vista de los resultados obtenidos en el modelo inicial o básico, se desarrolla un algoritmo con capacidad de identificar el número de alelos presentes en una comunidad sintética y el número de mutaciones tomando como referencia el alelo en mayor proporción. Este algoritmo es un paso intermedio entre el inicio del desarrollo y la fase final del software *DeepCloneFinder*, la importancia radica en la sensibilidad de la red neuronal ante la detección de las características extraídas de la comunidad sintética, si esta sensibilidad es baja y por

consiguiente el rendimiento del algoritmo, sería inviable alcanzar un modelo final con la capacidad de cuantificar la abundancia relativa de cada alelo.

En la *figura 3* se muestran los datos obtenidos tras el entrenamiento del modelo avanzado empleando 2 capas ocultas de neuronas con función de activación ReLU, el entrenamiento se realiza durante 50 épocas y como se puede observar, a lo largo de la curva de aprendizaje, se generan oscilaciones en los valores de predicción, estas oscilaciones inicialmente están acentuadas pero a medida que la red ajusta los pesos, se suavizan las oscilaciones y la tendencia de aprendizaje del modelo transcurridas las épocas definidas para el entrenamiento, continúa al alza, lo que indica que si el número de épocas fuese mayor, la precisión alcanzada sería a su vez mayor.

Por ello, se modificarán los parámetros de entrenamiento aumentando el número de épocas y de la estructura de la red buscando un menor número de oscilaciones, para ello se modificará el número de neuronas en las capas ocultas y las funciones de activación.

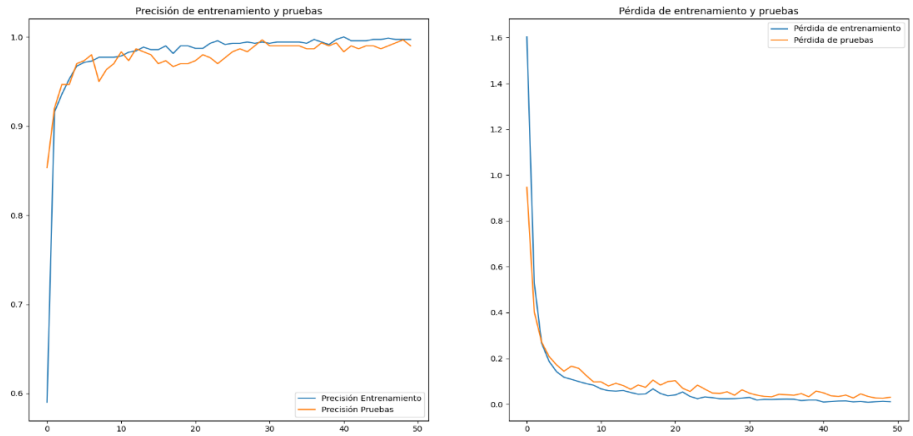

Figura 3 Parámetros del modelo, se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado con 2 capas de 50 neuronas cada una, con función de activación ReLU.

En un nuevo intento, se generará una estructura similar a la indicada para el modelo anterior, las capas ocultas en este caso serán de 100 neuronas cada una y la función de activación será la misma. Se incrementarán las épocas de entrenamiento un 100% alcanzando un total de 100 épocas.

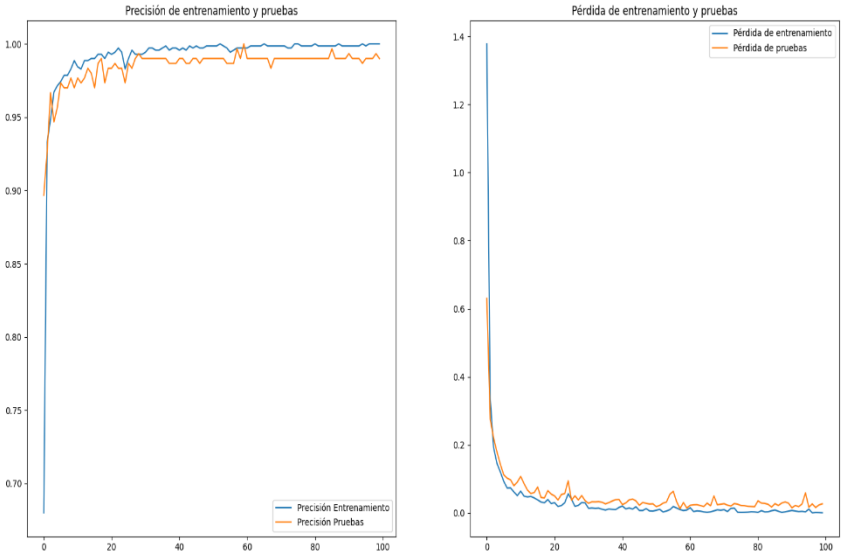

Figura 4 Parámetros del modelo, se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado con 2 capas de 100 neuronas cada una, con función de activación ReLU.

En la *Figura 4* se observa que el número de neuronas no ha conseguido eliminar por completo las oscilaciones en la curva de precisión, en cierto modo se suavizan, pero el valor de precisión en pruebas es ligeramente menor que en el modelo anterior a pesar de tener una mayor complejidad. Lo mismo ocurre con el número de épocas, la tasa de pérdida a partir de la época 40, es muy próxima a cero y, por lo tanto, a partir de ese instante, el aprendizaje del modelo es prácticamente nulo.

Se diseña un modelo cuya estructura es idéntica al modelo anterior, en este caso se modifica la función de activación entre las neuronas. Se sustituirá la función de activación ReLU por exponential.

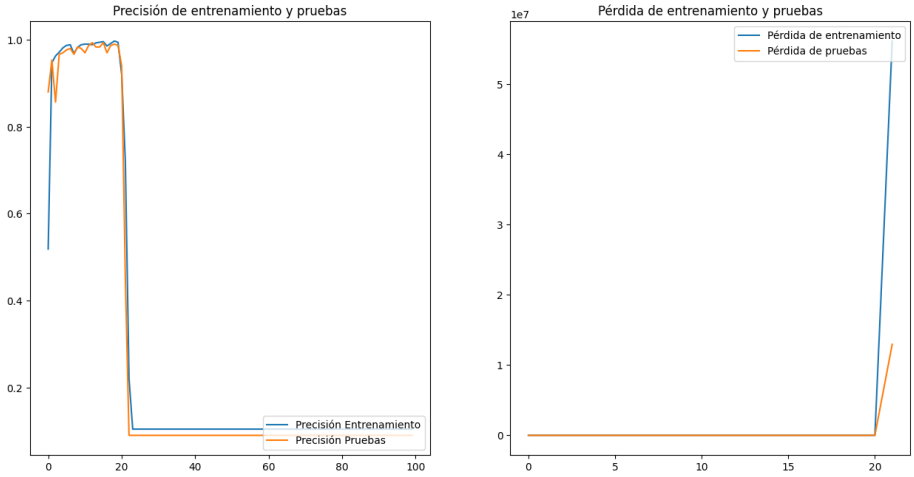

Figura 5 Parámetros del modelo, se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado con 2 capas de 100 neuronas cada una, con función de activación exponential.

En la *Figura 5* se puede observar el efecto producido por la función exponential, inicialmente ofrece una precisión elevada pero rápidamente cae hasta valores próximos a 0, en la pérdida de entrenamiento, se observa que el modelo no aprende, hasta la época 20 dónde se produce una pérdida de entrenamiento muy elevada.

En un siguiente modelo, se modificará la función de activación por la función ELU.

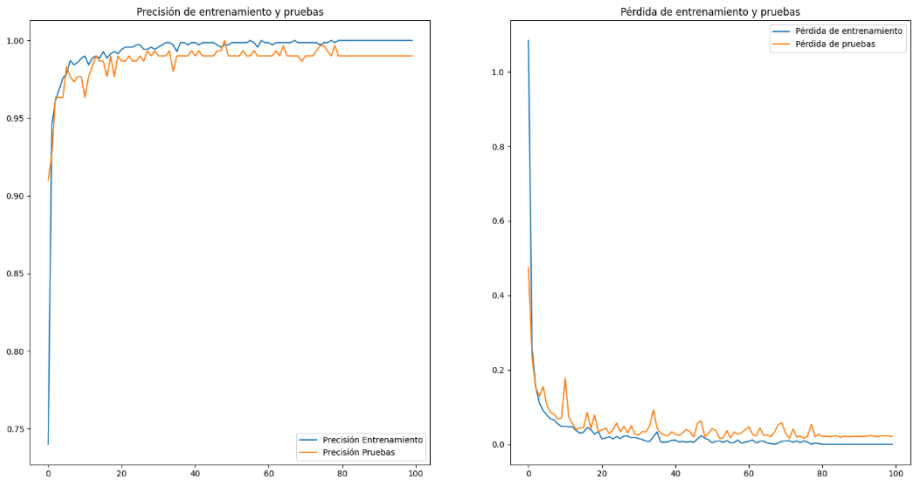

Figura 6 Parámetros del modelo. Se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado 2 capas ocultas con 100 neuronas. función de activación ELU.

En la *figura 6* se observa como el rendimiento del modelo es bueno, nuevamente se presentan las mismas oscilaciones que se observaban en el modelo con neuronas activadas por la función ReLU, en este caso, consigue eliminar este efecto a partir de la época 80. En la tasa de aprendizaje se observa que se produce a gran velocidad, aun así, los resultados soy muy similares y la capacidad de computación requerida para esta función de activación, es mayor al necesario para la función de activación ReLU.

En un último modelo, se empleará la función de activación GeLU.

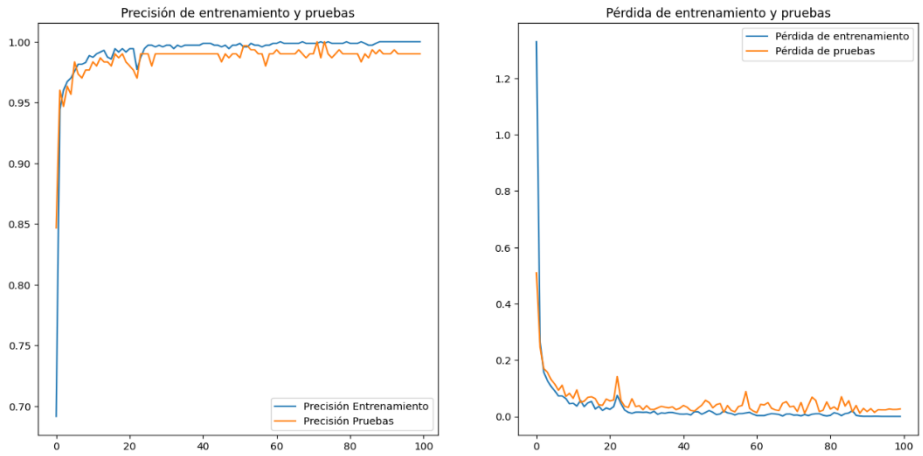

Figura 7 Parámetros del modelo. Se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado 2 capas ocultas de 100 neuronas, función de activación GeLU.

En la *figura 7* se observa que esta función de activación en el modelo genera un efecto similar al que generaba la función de activación ReLU, sin embargo, los resultados obtenidos son similares a modelos anteriores, al igual que ocurre con la función de activación ELU, el coste a nivel computacional es elevado y no supone una mejora sustancial frente a modelos anteriores.

Una vez determinada la estructura óptima, la cual será empleada para el modelo avanzado desarrollado, se genera ruido de forma deliberada para la obtención de un algoritmo con mayor capacidad de predicción en entorno real los resultados se muestran en la *figura 8.*

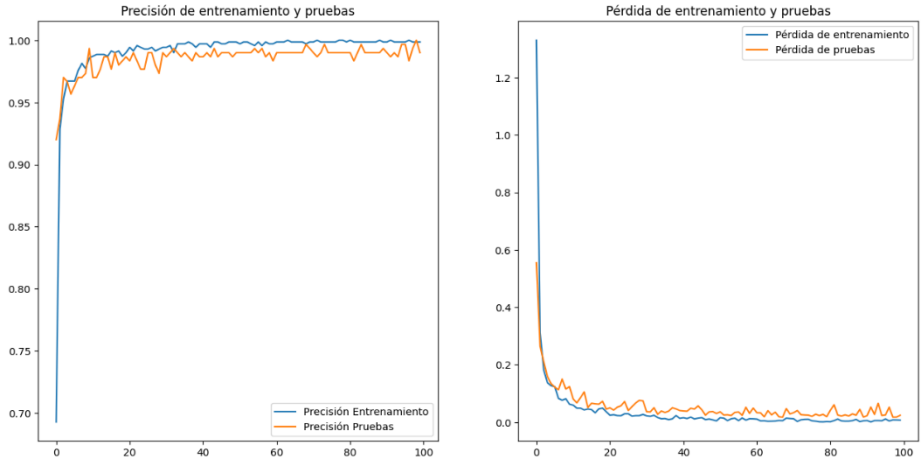

Figura 8 Parámetros del modelo. Se representa la precisión y la pérdida de entrenamiento frente al número de épocas. Modelo avanzado 2 capas ocultas de 100 neuronas, función de activación ReLU. Ruido aplicado a los datos de entrenamiento.

7.3 Modelo final.

Tras el éxito de las fases anteriores del proyecto, la viabilidad de obtener un algoritmo capaz de cuantificar la abundancia de los alelos presentes en una muestra e indicar el número de clones de cada uno de los alelos, es alta. En este sentido, se desarrolla un modelo cuya finalidad es la predicción de, partiendo de una matriz de probabilidad, determinar la abundancia relativa de cada alelo presente en la muestra. La matriz de probabilidad contiene el número total de conteos para cada posición separado por base nitrogenada y dividido entre en número de individuos presentes en la muestra. La intención es que, partiendo de esta matriz, el algoritmo de inteligencia artificial desarrollado sea capaz de indicar la etiqueta correcta.

El diccionario de etiquetas se muestra en la *Tabla 2*, el número de muestras para cada etiqueta es de 200 salvo la categoría perteneciente a la matriz con un único individuo, esta última categoría contará con 100 muestras. De modo que, la base de datos empleada para el entrenamiento del modelo es de (3300,4,200). El método para desarrollar esta base de datos se abordó en apartados anteriores.

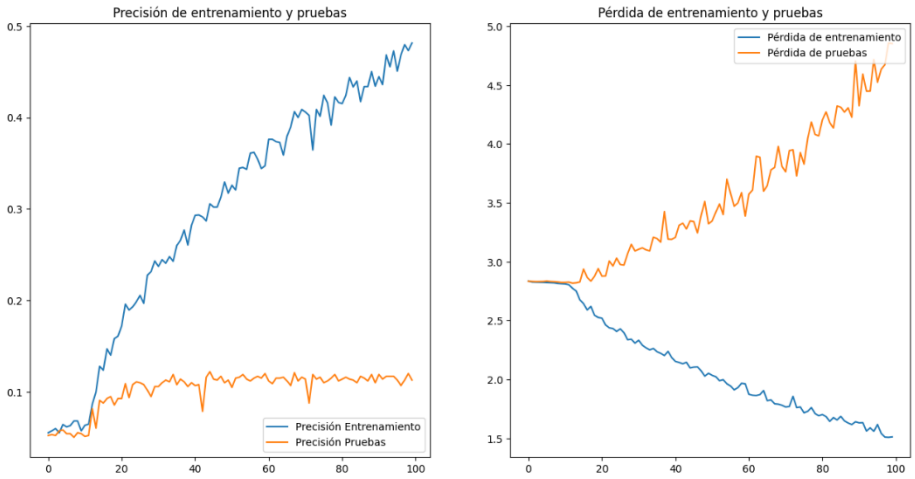

Figura 9 Parámetros del modelo. Se representa la pérdida de entrenamiento frente al número de épocas. Modelo Final 2 capas ocultas de 100 neuronas, función de activación ReLU.

En la Figura 9, se observa el resultado obtenido, la capacidad de predicción del algoritmo es de entorno al 10% en su aplicación para predecir los datos de test, por lo tanto, el rendimiento del algoritmo desarrollado es bajo.

Para la mejora de la eficiencia del modelo, principalmente se trabajó sobre la estructura de la red, se adicionaron neuronas y el resultado no mejoró, se trató de probar con funciones de activación como ELU o GeLU, cuyos requerimientos computacionales son superiores, pero evitan la pérdida de neuronas en el ajuste de los pesos, a pesar de ello el modelo continua sin mejorar el rendimiento. Finalmente se recurrió a adicionar capas ocultas de neuronas, lo cual tampoco desembocó en una mejora del rendimiento.

Cómo último recurso, se desarrolla una estructura de red neuronal convolucional (CNN) adicionando una capa de tipo convolución seguida de una de agrupación, estructura típica de este tipo de redes neuronales, finalmente se encadena con una estructura densa de clasificación.

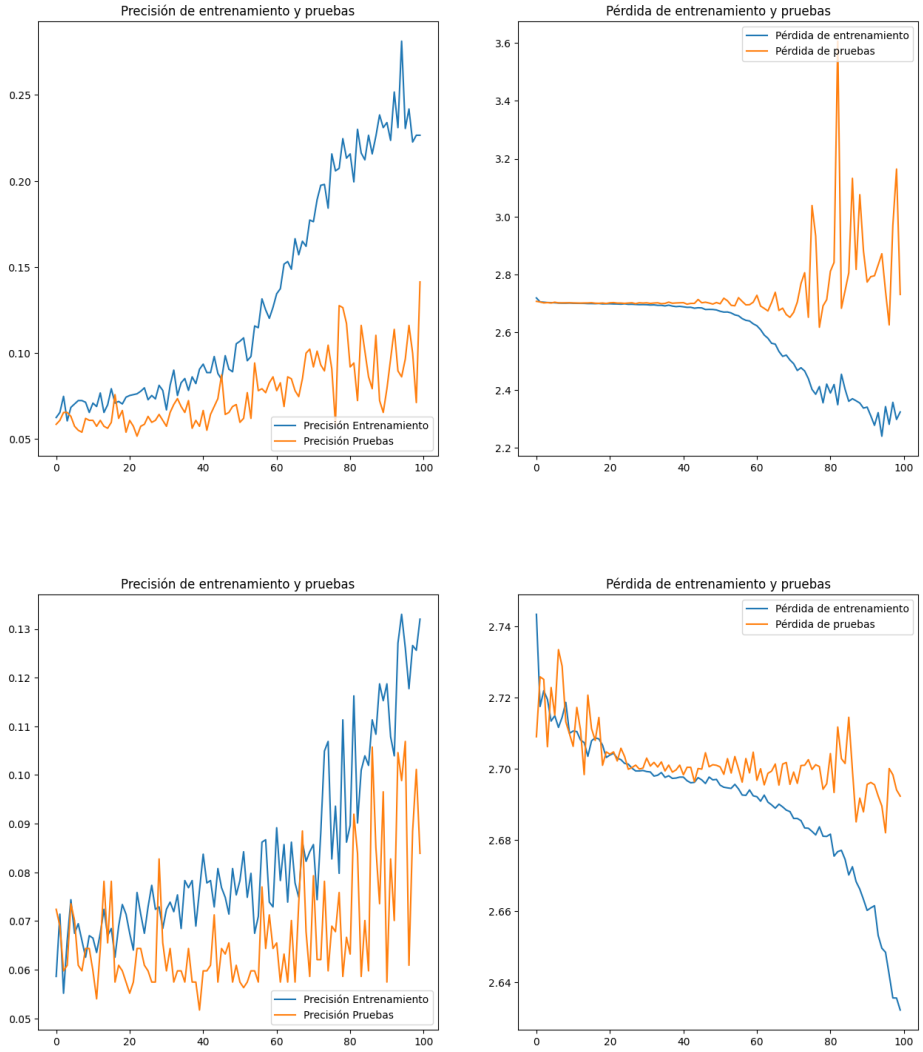

Figura 10 Parámetros del modelo. Se representa la pérdida de entrenamiento frente al número de épocas. Modelo Final, se emplean estructuras de redes neuronales convolucionales.

Si bien se observa una mejora en el rendimiento del algoritmo en comparación con su predecesor, sigue generando un rendimiento razonablemente bajo entorno a un 10%.

La profundidad de la capa se ve limitada por las dimensiones de los datos empleados, por lo tanto, los resultados reflejados en la *figura 10* corresponden a los obtenidos empleando la máxima profundidad posible.

8. Discusión y conclusión.

Los resultados del proyecto llevado a cabo, si bien se encuentran lejos de la obtención de un algoritmo con capacidad de predicción de la abundancia relativa de los alelos presentes en un muestra, se han obtenido dos modelos eficientes de predicción. El primer modelo tiene la capacidad de determinar el número de mutaciones presentes en una muestra a partir de una matriz de probabilidad, la precisión del modelo obtenido es de (val_accuracy 0.98) empleando una estructura de la red neuronal definida en la figura 2. Este método es útil para determinar el número de alelos presentes en la muestra. En un segundo modelo, se potencia esta característica señalada, y se obtiene un algoritmo capaz de determinar el número de alelos y las mutaciones, partiendo nuevamente de una matriz de probabilidad. Tras realizar diferentes modificaciones de la estructura de la red neuronal y de la función de activación de las neuronas, se concluye que la estructura óptima para abordar la clasificación de las muestras en función del número de alelos y mutaciones del gen estudiado es la constituida por dos capas ocultas con 50 neuronas cada una, función de activación ReLU, para generar transformaciones no lineales. El resultado se recoge en la figura 3, (val_accuracy 0.97).

En el desarrollo del algoritmo para la determinación de la abundancia relativa de los alelos y la cuantificación del número de clones de cada alelo, se obtuvieron resultados que requieren de mejora y optimización, se diseñó una estructura de red neuronal de tipo denso, con dos capas ocultas (*Hidden layers*) empleando en todas ellas la función de activación ReLU, se alcanzó un valor de precisión en las pruebas próximo al 10%, los resultados de este modelo se recogen en la figura 9, (val_accuracy 0.052), este valor indica que, la red neuronal, no consigue ajustar los pesos de modo que sea aplicable a otros datos que no sean los de entrenamiento. Dado que la estructura de red neuronal de tipo densa diseñada no ofrece el rendimiento deseado, se abordó el mismo problema empleando una red neuronal convolucional (CNN), este abordaje, no se empleó en las fases iniciales del proyecto debido a que el rendimiento ofrecido por la red neuronal densa era óptimo y el margen de mejora no era destacable. Empleando la red neuronal convolucional, el resultado obtenido por el clasificador, ha sido de una precisión entorno al 10%. Dada la complejidad de los cálculos internos en la red neuronal para la determinación de los pesos y sesgos, es imposible conocer cuál

es la limitación que impide el funcionamiento óptimo de la red puesto que, desde un punto de vista teórico, establecer la relación matemática entre la abundancia relativa y el número de clones por alelo, debería de ser asumible por una red neuronal con las características diseñadas para el desarrollo de este algoritmo y así se reflejaba en fases iniciales del proyecto.

Una de las posibles explicaciones es la similitud entre muestras en las que existe codominancia entre alelos, de modo que la proporción en la que se encuentran los alelos de la muestra no presenta diferencias, es el caso de los datos que por similitud se emparejan de la siguiente forma (etiqueta 2 – etiqueta 13; etiqueta 3 – etiqueta 16). Esto puede generar errores que desemboquen en un ajuste erróneo de los pesos, provocando la falta de precisión del algoritmo.

Existe evidencias de que la estimación y diferenciación se complica cuando se tienen cepas estrechamente relacionadas, por lo tanto, el error en las estimaciones de abundancia relativa es mayor en individuos estrechamente relacionados que en especies cuya relación sea menor (Angly et al., 2009). En el caso de los datos generados, el número de mutaciones inducidas para los diferentes alelos es poco significativo, un número aleatorio de mutaciones entre 1-20, en un gen de longitud 200pb, se traduce en una diferencia genética en el mayor de los casos del 10%. Esto puede repercutir en la precisión obtenida por el modelo para la determinación de la abundancia relativa. Otros proyectos como GASiC, ofrece resultados reduciendo hasta un 60% la tasa de error con respecto a los mejores enfoques existentes (Lindner & Renard, 2013) incluso en organismos relacionados con más del 95% de similitud de secuencia.

Otro abordaje interesante es el empleo de regiones menores, incluso de SNP, como el algoritmo desarrollado por Luo et al., 2015 denominado ConStrains, el cual emplea patrones en SNP como base para la diferenciación y cuantificación de cepas específicas.

En futuros pasos, se identificarán las posibles actuaciones que lleven a *DeepCloneFinder* a un rendimiento óptimo, estas actuaciones pasan por un aumento considerable del número de comunidades sintéticas empleadas para el entrenamiento del algoritmo de clasificación. A su vez, se reestructurará la base de datos para la extracción de las características de interés para el

entrenamiento del algoritmo, aportando datos con una diferenciación notable en relación a lo anteriormente citado, se aumentará la variabilidad genética, induciendo un mayor número de mutaciones y se agruparán los datos por rangos de abundancia relativa para omitir la redundancia de datos cuyas proporciones sean idénticas y pertenezcan a categorías distintas, puesto que desemboca una caída de la precisión del algoritmo desarrollado.

Otro abordaje posible, puede ser el desarrollo de diferentes algoritmos de clasificación, de modo que, de forma similar a lo realizado en las fases iniciales del proyecto, se empleen diferentes algoritmos para obtener características específicas de una comunidad sintética, de modo que finalmente se obtenga un software conformado por una serie de algoritmos que ofrezcan una descripción detallada de la comunidad estudiada.

El software desarrollado y denominado *DeepCloneFinder,* requiere de un mayor desarrollo y así se refleja en los resultados obtenidos en la fase final del presente proyecto, el algoritmo no ofrece el rendimiento deseado y la problemática abordad, concretamente la estimación de la abundancia relativa es compleja (Linder & Renard, 2013; Luo et al.,2015; Angly et al., 2009; Lu et al., 2017). El abordaje de este proyecto ha sido planteado para el asentamiento de unas bases sobre las que se trabajará en el futuro para la obtención de un software cuya precisión sea notable y se proceda con su evaluación en entorno relevante.

9. Bibliografía

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Brain, G. (n.d.). This paper is included in the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16). Open access to the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation is sponsored by USENIX. TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning. Retrieved November 5, 2022, from https://tensorflow.org.

Agrebi, S., & Larbi, A. (2020). Use of artificial intelligence in infectious diseases. In Artificial Intelligence in Precision Health (pp. 415–438). Elsevier. https://doi.org/10.1016/b978-0-12-817133-2.00018-5

Ahn, T.-H., Chai, J., & Pan, C. (2015). Genome analysis Sigma: Strain-level inference of genomes from metagenomic analysis for biosurveillance. Bioinformatics 31(2), 170–177. https://doi.org/10.1093/bioinformatics/btu641

Altschul S, Gish W, Miller W, Myers E, Lipman D: Basic local alignment search tool. J Mol Biol 1990, 215: 403–410.

Angly, F. E., Willner, D., Prieto-Davó, A., Edwards, R. A., & Schmieder, R. (2009). The GAAS Metagenomic Tool and Its Estimations of Viral and Microbial Average Genome Size in Four Major Biomes. PLoS Comput Biol, 5(12), 1000593. https://doi.org/10.1371/journal.pcbi.1000593

Besemer, J., & Borodovsky, M. (1999). Heuristic approach to deriving models for gene finding. Nucleic Acids Research, 27(19), 3911–3920. https://academic.oup.com/nar/article/27/19/3911/1057783

Brosch, R., Gordon, S. v., Marmiesse, M., Brodin, P., Buchrieser, C., Eiglmeier, K., Garnier, T., Gutierrez, C., Hewinson, G., Kremer, K., Parsons, L. M., Pym, A. S., Samper, S., van Soolingen, D., & Cole, S. T. (2002). A new

evolutionary scenario for the Mycobacterium tuberculosis complex. Proceedings of the National Academy of Sciences of the United States of America, 99(6), 3684–3689. https://doi.org/10.1073/PNAS.052548299

Buchfink, B., Xie, C. & Huson, D. Fast and sensitive protein alignment using DIAMOND. Nat Methods 12, 59–60 (2015). https://doi.org/10.1038/nmeth.3176.

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Daniel R (2004): The soil metagenome-a rich resource for the discovery of novel natural products Related papers T he met agenomics of soil The soil metagenome-a rich resource for the discovery of novel natural products. Current Opinion in Biotechnology 2004, 15: 199–204. https://doi.org/10.1016/j.copbio.2004.04.005

Garnier, T., Eiglmeier, K., Camus, J. C., Medina, N., Mansoor, H., Pryor, M., Duthoy, S., Grondin, S., Lacroix, C., Monsempe, C., Simon, S., Harris, B., Atkin, R., Doggett, J., Mayes, R., Keating, L., Wheeler, P. R., Parkhill, J., Barrell, B. G., … Hewinson, R. G. (2003). The complete genome sequence of Mycobacterium bovis. Proceedings of the National Academy of Sciences of the United States of America, 100(13), 7877–7882. https://doi.org/10.1073/PNAS.1130426100

Harper, A., Topol, E. Pharmacogenomics in clinical practice and drug development. Nat Biotechnol 30, 1117–1124 (2012). https://doi.org/10.1038/nbt.2424

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. In Nature (Vol. 585, Issue 7825, pp. 357–362). Nature Research. https://doi.org/10.1038/s41586-020-2649-2

Hassan, S., Sabreena, , Khurshid, Z., Bhat, S.A., Kumar, V. & Ameen, F. et al. (2022) Marine bacteria and omic approaches: A novel and potential

repository for bioremediation assessment. Journal of Applied Microbiology, 133, 2299– 2313. Available from: https://doi.org/10.1111/jam.15711

Hoff, K. J., Tech, M., Lingner, T., Daniel, R., Morgenstern, B., & Meinicke, P. (2008). Gene prediction in metagenomic fragments: A large scale machine learning approach. BMC Bioinformatics 9, 217. https://doi.org/10.1186/1471-2105-9-217

J. Peterson, S. Garges, M. Giovanni, P. McInnes, L. Wang, J.A. Schloss, V. Bonazzi, J.E. McEwen, K.A. Wetterstrand, C. Deal, C.C. Baker, V. Di Francesco, T.K. Howcroft, R.W. Karp, R.D. Lunsford, C.R. Wellington, T. Belachew, M. Wright, C. Giblin, H. David, M. Mills, R. Salomon, C. Mullins, B. Akolkar, L. Begg, C. Davis, L. Grandison, M. Humble, J. Khalsa, A.R. Little, H. Peavy, C. Pontzer, M. Portnoy, M.H. Sayre, P. Starke-Reed, S. Zakhari, J. Read, B. Watson, M. Guyer The NIH human microbiome Project Genome Res., 19 (2009), pp. 2317-2323, 10.1101/gr.096651.109

Li, J., & Qian, J. M. (2020). Artificial intelligence in inflammatory bowel disease: Current status and opportunities. Chinese Medical Journal, 133(7), 757–759. https://doi.org/10.1097/CM9.0000000000000714

Lindner, M. S., & Renard, B. Y. (2013). Metagenomic abundance estimation and diagnostic testing on species level, Nucleic Acids Research, Volume 41, Issue 1, 1 January 2013, Page e10, https://doi.org/10.1093/nar/gks803.

Lu, J., Breitwieser, F. P., Thielen, P., & Salzberg, S. L. (2017). Bracken: Estimating species abundance in metagenomics data. PeerJ Computer Science, 2017(1). https://doi.org/10.7717/peerj-cs.104

Luo, C., Knight, R., Siljander, H., Knip, M., Xavier, R. J., & Gevers, D. (2015). ConStrains identifies microbial strains in metagenomic datasets. Nature Biotechnology, 33(10), 1045–1052. https://doi.org/10.1038/nbt.3319

Ma, Y., Guo, Z., Xia, B., Zhang, Y., Liu, X., Yu, Y., Tang, N., Tong, X., Wang, M., Ye, X., Chen, Y., & Wang, J (2022). Identification of antimicrobial peptides from the human gut microbiome using deep learning. Nat Biotechnol 40, 921–931. https://doi.org/10.1038/s41587-022-01226-0

McDonald, D., Hyde, E., Debelius, J. W., Morton, J. T., Gonzalez, A., Ackermann, G., Aksenov, A. A., Behsaz, B., Brennan, C., Chen, Y., DeRight Goldasich, L., Dorrestein, P. C., Dunn, R. R., Fahimipour, A. K., Gaffney, J., Gilbert, J. A., Gogul, G., Green, J. L., Hugenholtz, P., … Gunderson, B. (2018). American Gut: an Open Platform for Citizen Science Microbiome Research. MSystems, 3(3). https://doi.org/10.1128/MSYSTEMS.00031-18/FORMAT/EPUB

McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

Morgulis, A., Coulouris, G., Raytselis, Y., Madden, T. L., Agarwala, R., & Schäffer, A. A. (2008). Database indexing for production MegaBLAST searches. BIOINFORMATICS, 24(16), 1757–1764. https://doi.org/10.1093/bioinformatics/btn322

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., Mende, D. R., Li, J., Xu, J., Li, S., Li, D., Cao, J., Wang, B., Liang, H., Zheng, H., … Zoetendal, E. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. Nature, 464(7285), 59–65. https://doi.org/10.1038/nature08821

S. Agrebi, A. Larbi. Use of artificial intelligence in infectious diseases Artif. Intell. Precis. Health (2020), pp. 415-438, 10.1016/B978-0-12-817133-2.00018-5

Streit W, Daniel R, Jaeger KE: Prospecting for biocatalysts and drugs in the genomes of non-cultured microorganisms. Current Opinion in Biotechnology 2004, 15: 285–290. 10.1016/j.copbio.2004.05.006

Tringe SG, von Mering C, Kobayashi A, Salamov AA, Chen K, Chang HW, Podar M, Short JM, Mathur EJ, Detter JC, Bork P, Hugenholtz P, Rubin EM (2005). Comparative metagenomics of microbial communities. Science 2005, 308: 554–557. 10.1126/science.1107851

Tyson, GW., Chapman, J., Hugenholtz ,P., Allen, EE., Ram, RJ., Richardson, PM., Solovyev, VV., Rubin, EM., Rokhsar, DS., Banfield, JF., (2004). Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature, 428: 37–43. 10.1038/nature02340

Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.

Wood, D. E., & Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. http://ccb.jhu.edu/software/kraken/.

Wright, R. J., Comeau, A. M., & Langille, M. G. I. (2022). From defaults to databases: parameter and database choice dramatically 1 impact the performance of metagenomic taxonomic classification tools. https://doi.org/10.1101/2022.04.27.489753