CLASIFICACIÓN TAXONÓMICA DE BACTERIAS USANDO MACHINE LEARNING

LAURA GUADALUPE GOLONDRINO FERNÁNDEZ

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
Bogotá, Colombia
2020

CLASIFICACIÓN TAXONÓMICA DE BACTERIAS USANDO MACHINE LEARNING

LAURA GUADALUPE GOLONDRINO FERNÁNDEZ

ASESORES:
FERNANDO ENRIQUE LOZANO Ph. D
CAROLINA HIGUERA M. Sc

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
Bogotá, Colombia

2020

Índice

Índice de cuadros

Índice de figuras

1. INTRODUCCIÓN

Cada ser vivo del planeta contiene material genético que brinda información biológica fundamental para conocerlo y clasificarlo. Ese material genético está constituido por ácido ribonucleico (ADN), el cual es una molécula construida por cadenas de nucleótidos [1]. Dos cadenas de estos nucleótidos se denominan bases y forman el ADN como una estructura de hélice, de esa forma, la secuencia de nucleótidos es lo que caracteriza el genoma de cada ser vivo [2].

El genoma bacteriano está compuesto por una molécula de ADN que contiene entre 500 mil y 10 millones de pares de bases [3]. La importancia de clasificar bacterias es distinguir un organismo de otro y agrupar organismos similares por criterios de interés para los microbiólogos u otros científicos [4]. Esos criterios de interés pueden estar en diferentes áreas como: la biotecnología, el estudio de muestras ambientales, la industria o el diagnóstico de enfermedades producidas por bacterias. Otro aspecto importante es la clasificación de bacterias por su ADN, pues facilita la tarea de aislarlas y cultivarlas, teniendo en cuenta que son dificiles de cultivar en condiciones de laboratorio [5], incluso se puede hacer una clasificación más precisa a nivel taxonómico.

La predicción de secuencias de ADN es una tarea difícil debido a la corta longitud de lectura, la naturaleza incompleta y fragmentada de los datos [6], además de las millones de bacterias que existen en el mundo. Es por eso que se han desarrollado modelos de machine learning para lograr la extracción de características en ese tipo de datos [7]. Este trabajo busca estudiar modelos de aprendizaje profundo para realizar una correcta clasificación taxonómica de bacterias usando secuencias de ADN.

2. OBJETIVOS

2.1. Objetivo general

Crear un modelo basado en machine learning para clasificar taxonómicamente un grupo de bacterias a través de secuencias de ADN.

2.2. Objetivos específicos

- Establecer el uso del aprendizaje automático para la clasificación de bacterias mediante secuencias de ADN.
- Definir la representación vectorial de las secuencias de ADN para la implementación del modelo de machine learning.
- Desarrollar diferentes modelos de machine learning para evaluar su desempeño en la clasificación taxonómica de bacterias.
- Escoger y validar un modelo de machine learning que logre clasificar correctamente todos los taxones, teniendo en cuenta su eficiencia y costo computacional.

3. MARCO TEÓRICO

3.1. Secuencias de ADN

El ADN es una molécula que consiste en dos cadenas que se mantienen unidas por enlaces entre las bases, formando una estructura de doble hélice. Las bases son: adenina (A), citosina (C), guanina (G), y timina (T) [1]. La secuencia de ADN es el orden en el que se encuentran esas bases, además de indicar la clase de información genética que se transporta en un segmento específico de ADN.

3.2. Secuencias del gen 16s:

Es una de las más utilizadas para estudios de filogenia y taxonomía, pues es diferente para cada especie bacteriana indicando al menos el 97 % de identidad. Esta secuencia tiene nueve regiones hipervariables cortas que diferencian los distintos taxones bacterianos [8].

3.3. Taxonomía:

La taxonomía es la ciencia de la clasificación, identificación y nomenclatura en los seres vivos [4].Los taxones siguen un orden jerárquico. En la figura 3.3.1 se muestran los taxones analizados en este trabajo.

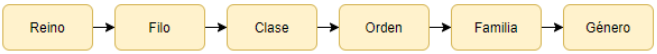


Figura 3.3.1: Taxones

3.4. Red neuronal multicapa:

Las redes neuronales multicapa o perceptrón multicapa son modelos de aprendizaje automático qu recrean modelos simples del cerebro para resolver tareas computacionales difíciles [9]. Este tipo de red está compuesta por:

- **Neuronas artificiales:**

Figura 3.4.1: Neurona artificial

Son unidades computacionales simples que tienen señales de entrada (x_1 y x_2) y un término de intercepción ($+1$), como se muestra en la figura 3.4.1,

estas entradas ponderadas se suman produciendo una señal de salida $h_{W,b}$ que pasa a través de una función de activación [10]:

$$h_{W,b} = f(W^T x) = f(\sum_{i=1}^{2} W_i x_i + b) \tag{1}$$

Donde f es la función de activación y W los pesos de las entradas.

- **Funciones de activación:**
  Relu: La neurona se activará al cumplirse lo siguiente:

$$R(z) = \begin{Bmatrix} z & z > 0 \\ 0 & z \leq 0 \end{Bmatrix} \tag{2}$$

Softmax: Convierte un vector real en un vector de probabilidades categóricas. Los elementos del vetor de salida están en el rango (0,1) con una suma de probabilidades igual a 1 [11]. Esta función se utiliza en la capa de salida para problemas multiclase.

$$f(x) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3}$$

- **Estrcutura de la red neuronal:** Una red neuronal es la conexión de muchas neuronas. A una fila de neuronas se le llama capa. De esa forma, la arquitectura de la red tendrá una capa de entrada, capas ocultas y la capa de salida como se observa en la figura 3.4.2. Las capas ocultas se denominan así porque no están expuestas directamente a la capa de entrada [9].

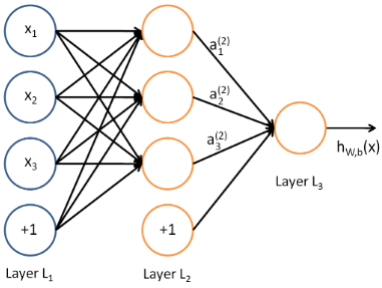


Figura 3.4.2: Estructura de una red neuronal [10].

3.5. Red neuronal convolucional o CNN:

Las CNN son modelos de aprendizaje profundo que recoge y clasifica características de una imagen. Las neuronas dentro de una CNN se dividen en una

estructura, y cada conjunto de neuronas analiza una pequeña región o característica de la imagen [12].

- **Capa convolucional:** Esta capa crea un mapa de características al aplicar un filtro a una entrada (las dimensiones del filtro deben ser menores a las dimensiones de la entrada). Ese mapa indica las ubicaciones y el tipo de caracteristica detectada en una imagen [13].
- **Capa de agrupación:** Esta capa reduce la cantidad de información del mapa de características, dejando la información más esencial [12].
- **Capa de aplanamiento o flatten:** Esta capa convierte la salida de las capas convolucionales en un vector [12].
- **Capa conectada:** Es una capa sencilla de la red neuronal multicapa.

Para analizar la estructura de una CNN, se toma como ejemplo la arquitectura LeNet 5, puesto que más adelante se trabajará con ella.

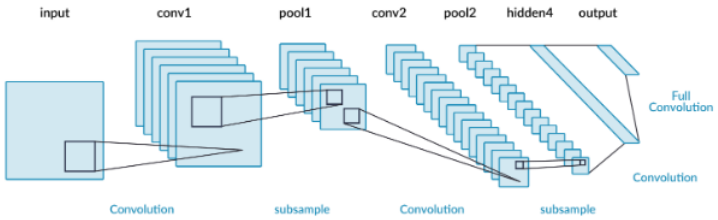
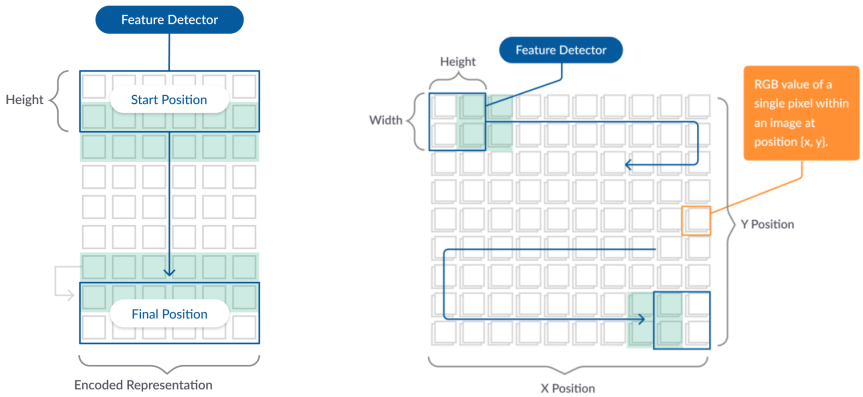


Figura 3.5.1: Estructura de la red neuronal convolucional LeNet-5 [12].

De la figura 3.5.1, la entrada es una imagen de tamaño 32×32, la primera capa convolucional tiene un filtro de 5×5, esta crea los mapas de características de la entrada. La siguiente capa es de agrupamiento, la cual reduce las dimensiones de los mapas. El proceso se repite y entre *pool2* y *hidden4* se encuentra la capa de aplanamiento, convirtiendo los mapas en un vector.

Las diferencias entre una CNN 2D y 1D son: la estructura de los datos de entrada y como el filtro se mueve a través de los datos. En la figura 3.5.2a y 3.5.2b se observa el movimiento del filtro en una CNN 1D y 2D.



(a) Convolucional 1D (b) Convolucional 2D

Figura 3.5.2: Movimiento del filtro en una CNN 1D y 2D [14].

4. MATERIALES Y MÉTODOS

4.1. Software

El procesamiento de los datos, entrenamiento y evaluación de los modelos de aprendizaje profundo se desarrollaran en el ambiente de Python, usando las bibliotecas de TensorFlow y keras.

4.2. Base de datos

La base de datos utilizada[1] hace parte de una investigación desarrollada en el artículo: "Deep learning models for bacteria taxonomic classification of metagenomic data"publicado en *BMC Bioinformatics*. Contiene 28 mil secuencias cortas de ADN del gen 16s de un grupo de bacterias. Las secuencias se generaron usando tecnología de secuenciación de nueva generación (NGS) denominada como amplicón (AMP). Además, el grupo de bacterias pertenece al filo proteobacteria y cada secuencia está etiquetada por taxones de clase a género.El tratamiento de los datos se encuentra disponible en [5]. El cuadro 4.2.1 contiene la información sobre el número de etiquetas de cada taxón.

Taxón	Número de etiquetas
Clase	3
Orden	20
Familia	37
Género	96

Cuadro 4.2.1: Información sobre el número de etiquetas en los diferentes taxones

En las figuras 4.2.1, 4.2.2, 4.2.3 y 4.2.4 se puede observar el número de datos para cada etiqueta de los diferentes taxones. Además del nombre de las bacterias que lo conforman.

[1]Link: `http://tblab.pa.icar.cnr.it/public/BMC-CIBB_suppl/datasets/`

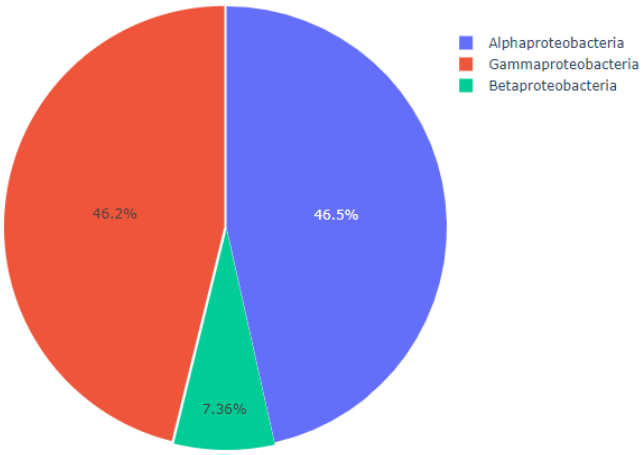


Figura 4.2.1: Distribución de los datos en el taxón clase

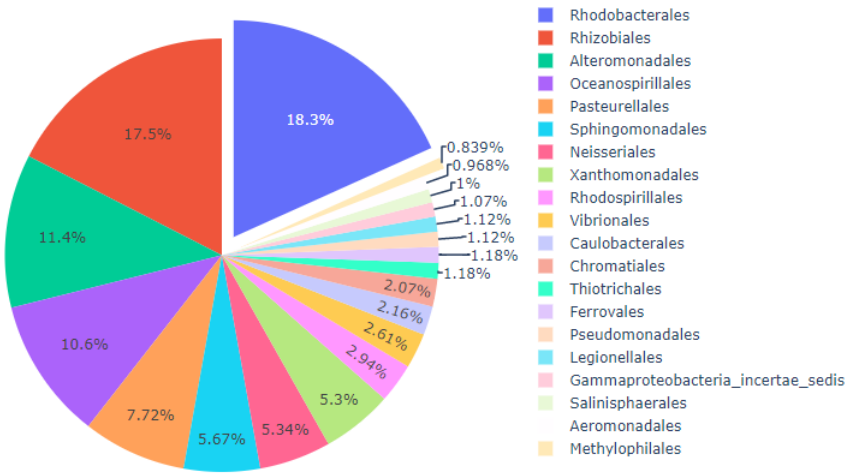


Figura 4.2.2: Distribución de los datos en el taxón orden

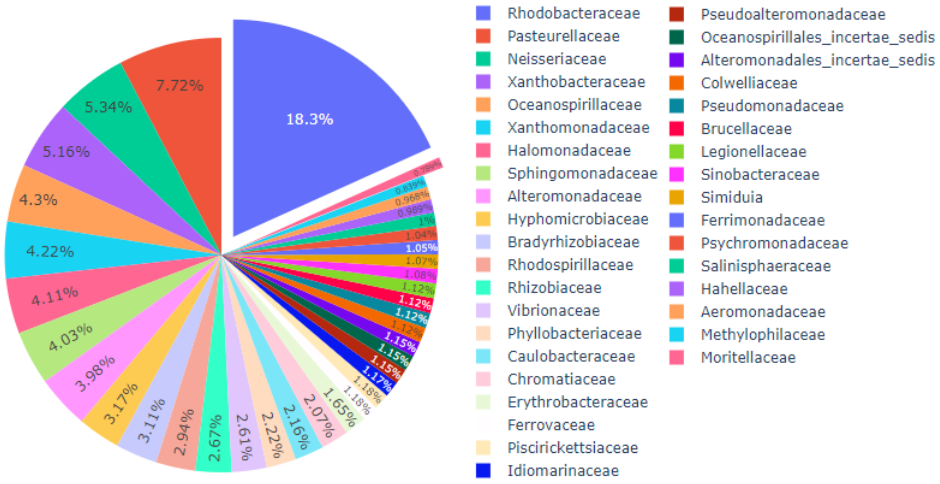


Figura 4.2.3: Distribución de los datos en el taxón familia

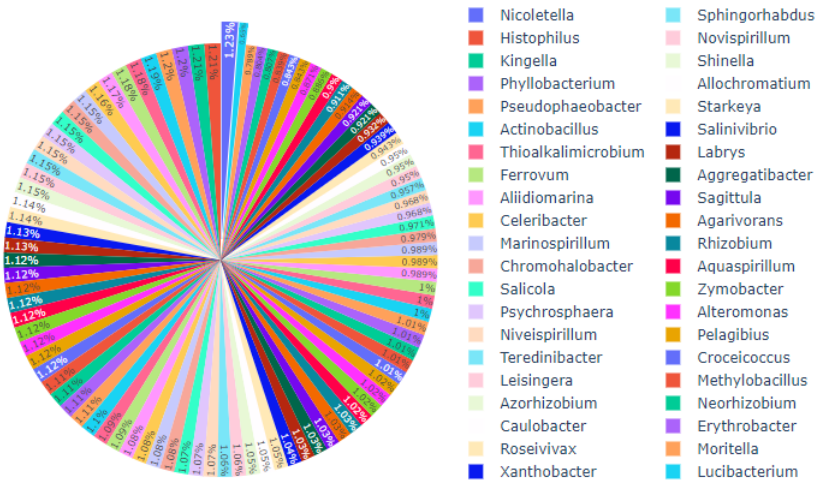


Figura 4.2.4: Distribución de los datos en el taxón género

Con lo anterior, se puede observar que tenemos una base de datos desbalanceada. Esto puede causar un bajo rendimiento en las clases minoritarias, por eso se usó la técnica de sobremuestreo de minorías sintéticas o SMOTE. Esta técnica selecciona una instancia de la clase minoritaria al azar y encuentra sus k vecinos más cercanos, la instancia sintética se crea en un punto seleccionado aleatoriamente entre las dos instancias de la clase minoritaría [15].

4.3. Procesamiento

4.3.1. K-mers

En [5], [16], [17] y [18] utilizan vectores k-mers para obtener una representación vectorial de secuencias de ADN. Esta representación se basa en obtener subcadenas con una cierta longitud k en las secuencias originales. De esa forma, se tiene una secuencia S de n carácteres determinados por el alfabeto: $\lambda = A, G, C, T = 4$, que representan los nucleótidos del ADN. Los k-mers (K) serán todas las posibles combinaciones de λ de tamaño k que pertenecen a S [18]. El vector V contiene todas las subcadenas de K:

$$V = K_1, K_2, K_3 ... K_m \tag{4.3.4}$$

Donde V tendrá un tamaño de:

$$\lambda^k = m \tag{4.3.5}$$

Las subcadenas K se forman por medio de una ventana deslizante de la secuencia S, comenzando en la posición 1 hasta la posición *n-k+1*. Por ejemplo, en la figura 4.3.1 se puede observar los k-mers de tamaño *k=4* obtenidos de la secuencia 'AAGTCAAGT':

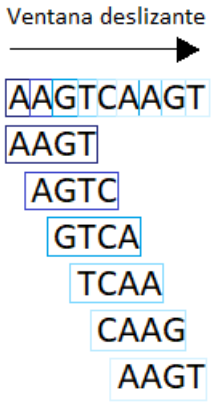


Figura 4.3.1: Formación de las subcadenas K.

Al utilizar este tipo de representación no es importante mantener el orden de los k-mers de la secuencia original, pues el objetivo es encontrar k-mers distintivos. También, en el caso de k-mers que contienen caracteres que no representan un nucleótido de ADN, por ejemplo 'N', se podrán eliminar [16].

Según [5] valores pequeños para la longitud de los k-mers pueden ser suficientes para obtener información de la secuencia, además de evitar definir un espacio vectorial muy grande. Es por eso que, para este estudio, se decidió usar tres valores de k diferentes: 3, 4 y 5. De esa forma se podrá analizar cual tamaño

de k ofrece mejor información de la secuencia de ADN. En el cuadro 4.3.1 está el total número de k-mers para los diferentes tamaños de k.

Tamaño de k	Total número de k-mers
3	64
4	256
5	1024

Cuadro 4.3.1: Longitud del vector V para los tamaños de k utilizados.

4.3.2. Frecuencia K-mer

Una vez obtenidos todos los k-mers de la secuencia, se procede a crear un vector f que contiene el número total de cada k-mer en S:

$$f = c_1, c_2, c_3 ... c_m \tag{4.3.6}$$

Por último, la frecuencia de cada k-mer es igual al número total del k-mer en la secuencia S sobre el total de k-mers de la secuencia [18]:

$$F_i = \frac{c_i}{n - k + 1} \tag{4.3.7}$$

Retomando el ejemplo anterior con la secuencia 'AAGTCAAGT' de tamaño n=*9* y k=*4*, la frecuencia de cada k-mer se observa en el cuadro 4.3.2.

K-mer	Total	Frecuencia
AAGT	2	1/3
AGTC	1	1/6
GTCA	1	1/6
TCAA	1	1/6
CAAG	1	1/6

Cuadro 4.3.2: Ejemplo de la frecuencia k-mer

Además, la longitud del vector F es $m = \lambda^4 = 256$.

4.3.3. Estandarización

La estandarización de un conjunto de datos implica reescalar la distribución de valores para que la media sea cero y la desviación estándar sea 1, de esa forma, el conjunto de datos se ajusta a una distribución gaussiana con media y varianza de buen comportamiento. En una red neuronal profunda es importante reescalar los datos de entrada, pues mejora la estabilidad y el rendimiento de la red [19].

Para la estandarización se requiere la media (μ) y la desviación estándar (θ) de los datos de entrada (X_{in}). De la siguiente ecuación se obtiene los datos estandarizados (X_{out})s:

$$X_{out} = \frac{X_{in} - \mu}{\theta} \tag{4.3.8}$$

4.4. Vector de salida

Para la clasificación de cada taxón se tiene diferente número de etiquetas como se observó anteriormente. Es por eso que el vector de salida dependerá del taxón que se está clasificando. Sin embargo, para todos los taxones se usará la representación de vectores one-hot usando la función de keras *to categorical*. Por ejemplo, en el cuadro 4.4.1 se observa los vectores de salida para el taxón clase.

Clase	Vector one-hot
Alphaproteobacteria	1 0 0
Betaproteobacteria	0 1 0
Gammaproteobacteria	0 0 1

Cuadro 4.4.1: Representación del vector de salida para el taxón clase

4.5. Datos de entrenamiento, validación y prueba

Antes de comenzar un modelo de aprendizaje profundo es necesario dividir el conjunto de datos en entrenamiento, validación y prueba. Los datos de entrenamiento son el 80 % del total de los datos y, como su nombre lo indica, se usan para entrenar todos los modelos planteados. Los datos de validación son el 10 % de los datos de entrenamiento y se usan para observar el rendimiento de los modelos. Por último, los datos de prueba son el 20 % del total de los datos y sólo se usan con el modelo escogido para evaluarlo. El total de los datos se puede observar en el cuadro 4.5.1.

Datos	Número
Entrenamiento	20160
Validación	2240
Prueba	5600
Total datos	28000

Cuadro 4.5.1: Número de datos

4.6. Red neuronal multicapa

En [5], [6], [7] y [16] utilizan redes neuronales profundas para la clasificación de secuencias de ADN. Es por eso que, inicialmente, se probará un modelo de

red neuronal multicapa para evaluar su rendimiento con este tipo de datos.

Se creó un modelo secuencial de dos capas ocultas con las siguientes características:

- Optimizador: Adam.
- Función de pérdida: categorical crossentropy, pues es una red neuronal multiclase.
- Activación de las capas ocultas: Relu.
- Activación de la capa de salida: Softmax.
- Tasa de aprendizaje 0.01.
- Métrica: Exactitud.

Sin embargo, para obtener el número de neuronas adecuado en cada capa oculta, se construyó un modelo diferente para cada longitud de k-mer. Además, se probó diferentes números de neuronas, entre 10 y 150, para obtener el modelo de mayor exactitud. En el cuadro 4.6.1 se observa el número de neuronas escogido para cada capa.

Número de neuronas en cada capa			
Nombre	K-mer 3	K-mer 4	K-mer 5
Capa de entrada	64	256	1024
Capa 1 oculta	120	120	120
Capa 2 oculta	60	84	64
Capa de salida	*	*	*

Cuadro 4.6.1: Número de neuronas para cada capa y k-mer.

(*) La capa de salida depende del taxón que se va a clasificar, en el cuadro 4.6.2 está el número de neuronas en la capa de salida para cada taxón.

Capa de salida	
Taxón	Número de neuronas
Clase	3
Orden	20
Familia	37
Género	96

Cuadro 4.6.2: Número de neuronas para la capa de salida.

Este modelo se usará con la base de datos desbalanceada y balanceada con SMOTE. De esa forma, se desea evaluar si las clases sintéticas creadas con SMOTE mejoran el rendimiento del modelo.

4.7. Red neuronal convolucional (CNN)

Se crean modelos de dos tipos de CNN: 1D y 2D. De esa forma, se logrará encontrar que tipo de red tiene mejores resultados.
Las características usadas en la CNN 1D son:

- El tamaño del kernel en las capas convolucionales es: 3, 4 y 5 para los k-mers 3, 4 y 5 respectivamente.
- Capa de agrupación máxima 1D o *MaxPooling* 1D de tamaño 2 después de cada capa convolucional.

Las características usadas en la CNN 2D son:

- El tamaño del kernel en las capas convolucionales depende del k-mer: 3x3, 4x4 y 5x5 para los k-mers 3, 4 y 5 respectivamente.
- Capa de agrupación máxima 2D o *MaxPooling* 2D de tamaño 2x2 después de cada capa convolucional.

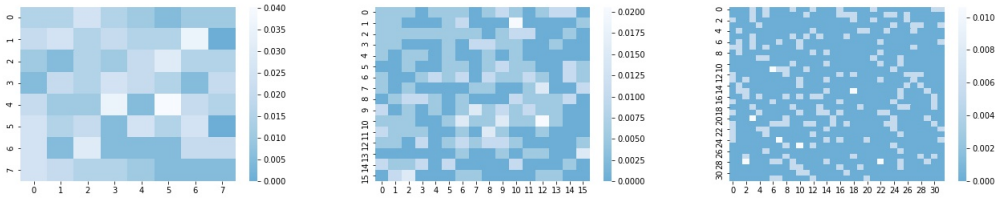
Las características generales usadas en cada CNN:

- Capa de aplanamiento o *Flatten* seguida de las capas convolucionales.
- Optimizador: Adam.
- Función de pérdida: categorical crossentropy, pues es una red neuronal multiclase.
- Activación de las capas: Relu.
- Activación de la capa conectada de salida: Softmax.
- Tasa de aprendizaje 0.01.
- Métrica: Exactitud.

La estructura de los datos de entrada cambia dependiendo si es 1D o 2D. En la CNN 1D la secuencia de datos es unidimensional, por eso se crea una matriz $n \times n$. Por otro lado,la CNN 2D es bidimensional, entonces se crea una imagen con profundidad de 1 (escala de grises) de tamaño $n \times n \times 1$. En ambos casos n depende del tamaño del k-mer:

- K-mer 3: $n = 8$
- K-mer 4: $n = 16$
- K-mer 5: $n = 32$

En la figura 4.7.1 se observa la representación de los datos de entrada como una imagen para cada k-mer.



(a) K-mer 3 (b) K-mer 4 (c) K-mer 5

Figura 4.7.1: Representación de los datos de entrada

Una vez obtenido la estructura de los datos de entrada, se construyeron 7 modelos para cada tipo de red. El número de neuronas en cada capa se determinó de la misma manera que la red neuronal multicapa.

El modelo 1 se basó en una red neuronal convolucional LeNet-5, pues en [16] se usó este tipo de red para clasificar secuencias de ADN. De igual forma, el modelo 3 se basó en la referencia [7]. Las siguientes figuras muestran los modelos construídos con el número de neuronas en cada capa.

Modelo 1	
Capa	Número de neuronas
Convolucional 1	6
Convolucional 2	16
Conectada 1	120
Conectada 2	84

Figura 4.7.2: Modelo 1.

Modelo 2	Número de neuronas					
	CNN 1D			CNN 2D		
Capa	k-mer 3	k-mer 4	k-mer 5	k-mer 3	k-mer 4	k-mer 5
Convolucional 1	60	60	84	32	32	32
Convolucional 2	100	100	100	68	100	36
Conectada 1	120	120	90	60	120	120
Conectada 2	84	84	112	64	84	64

Figura 4.7.3: Modelo 2.

Modelo 3	
Capa	Número de neuronas
Convolucional 1	10
Convolucional 2	20
Conectada 1	500

Figura 4.7.4: Modelo 3.

Modelo 4	Número de neuronas					
	CNN 1D			CNN 2D		
Capa	k-mer 3	k-mer 4	k-mer 5	k-mer 3	k-mer 4	k-mer 5
Convolucional 1	80	50	80	120	20	20
Conectada 1	120	120	90	60	120	120
Conectada 2	112	84	112	64	84	64

Figura 4.7.5: Modelo 4.

Modelo 5	Número de neuronas					
	CNN 1D			CNN 2D		
Capa	k-mer 3	k-mer 4	k-mer 5	k-mer 3	k-mer 4	k-mer 5
Convolucional 1	10	10	10	10	10	10
Convolucional 2	20	20	20	20	20	20
Convolucional 3	30	30	30	30	30	30
Conectada 1	60	120	120	60	120	120
Conectada 2	64	84	84	64	84	64

Figura 4.7.6: Modelo 5.

Modelo 6	Número de neuronas					
	CNN 1D			CNN 2D		
Capa	k-mer 3	k-mer 4	k-mer 5	k-mer 3	k-mer 4	k-mer 5
Convolucional 1	112	112	112	32	64	64
Convolucional 2	90	90	60	60	90	60
Conectada 1	120	100	120	120	100	100

Figura 4.7.7: Modelo 6.

Modelo 7	Número de neuronas					
	CNN 1D			CNN 2D		
Capa	k-mer 3	k-mer 4	k-mer 5	k-mer 3	k-mer 4	k-mer 5
Convolucional 1	64	84	100	64	32	64
Conectada 1	120	100	80	120	120	120

Figura 4.7.8: Modelo 7.

5. RESULTADOS Y ANÁLISIS

5.1. Desempeño del conjunto de datos balanceado y desbalanceado en la red neuronal multicapa

Se entrenó la red neuronal multicapa para cada k-mer y taxón, y con el conjunto de validación se evaluó la red. Las métricas utilizadas para evaluar el modelo son:

- Exactitud: Es el porcentaje de aciertos del modelo para todas las clases.
- Precisión: Define que tan confiable es el modelo para decir que una clase verdaderamente pertenece a esa clase.
- Recall o exhaustividad: Es la respuesta del modelo de que tan bien predice una clase.

Como se obtiene una precisión y recall para cada una de las clases de cada taxón, se tomará la media de las métricas para una observación general.

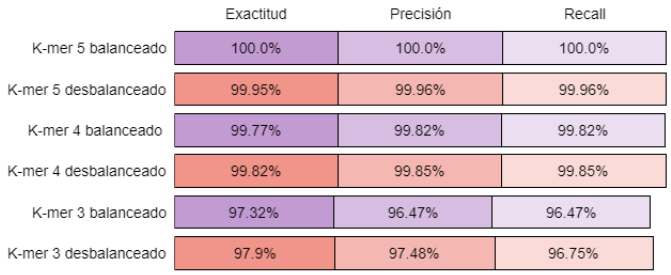


Figura 5.1.1: Desempeño del modelo en el taxón clase.

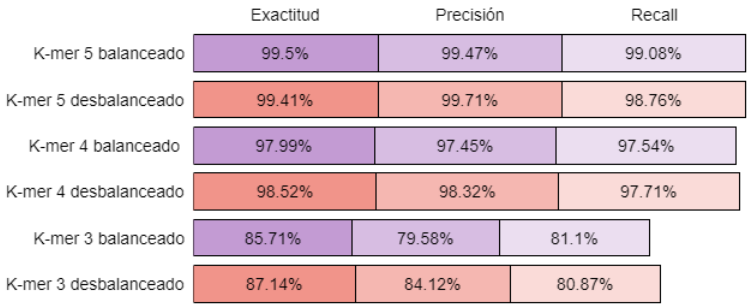


Figura 5.1.2: Desempeño del modelo en el taxón orden.

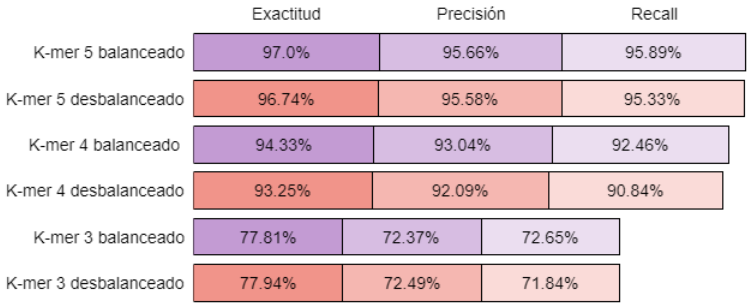


Figura 5.1.3: Desempeño del modelo en el taxón familia.

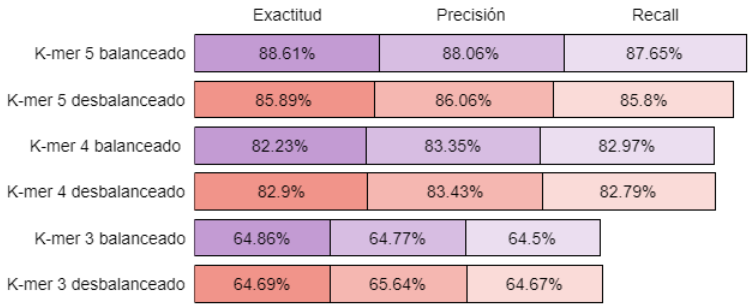


Figura 5.1.4: Desempeño del modelo en el taxón género.

De las figuras 5.1.1, 5.1.2, 5.1.3 y 5.1.4, se puede observar que el desempeño para el conjunto de datos balanceado y desbalanceado es bueno en general y no son muy diferentes, pues las métricas tienen porcentajes altos. En algunos casos el conjunto de datos desbalanceado tiene mejor desempeño, sin embargo, solo con el conjunto de datos balanceado se obtuvo una exactitud del 100 % en el taxón clase con k-mer 5. Lo más acertado, teniendo en cuenta los resultados, es usar el conjunto de datos balanceado para seguir evaluando los siguientes modelos.
Como se logró porcentajes altos en las métricas de precisión y recall, en los siguientes modelos se observará sólo la métrica de exactitud para un diagnóstico más general de su desempeño.

5.2. Resultados de la red neuronal multicapa

Se entrenó el modelo de la red neuronal multicapa en 50 épocas para observar el desempeño en cada taxón. Además de observar el tiempo de entrenamiento.

Taxón	K-mer 3		K-mer 4		K-mer 5	
	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)
Clase	97.32	64.1	99.77	69.42	100	101.29
Orden	85.71	162.33	97.99	182.04	99.5	285.68
Familia	77.81	303.87	94.32	349.01	97	528.81
Género	64.86	56.58	82.23	63.07	88.61	97.89

Figura 5.2.1: Exactitud y tiempo de entrenamiento para cada k-mer y taxón.

De la figura 5.2.1, se puede concluir que hay un mayor desempeño con k-mer 5 y menor con k-mer 3 en cada uno de los taxones. Además, la exactitud también disminuye a medida que los taxones disminuyen de grado, pero sigue teniendo un porcentaje alto. También, el tiempo de entrenamiento no es muy alto.
Anteriormente se mencionó que se requiere la mayor exactitud posible para la clasificación taxonómica de bacterias, por lo cual se buscará mejorar el taxón género con los modelos de redes neuronales convolucionales.

5.3. Desarrollo de diferentes modelos de redes neuronales convolucionales

Se entrenaron los 7 modelos de CNN 1D y 2D para cada k-mer, pero se evaluó únicamente el taxón de género, pues es el de menor exactitud. Los resultados obtenidos se observan en la figura 5.3.1.

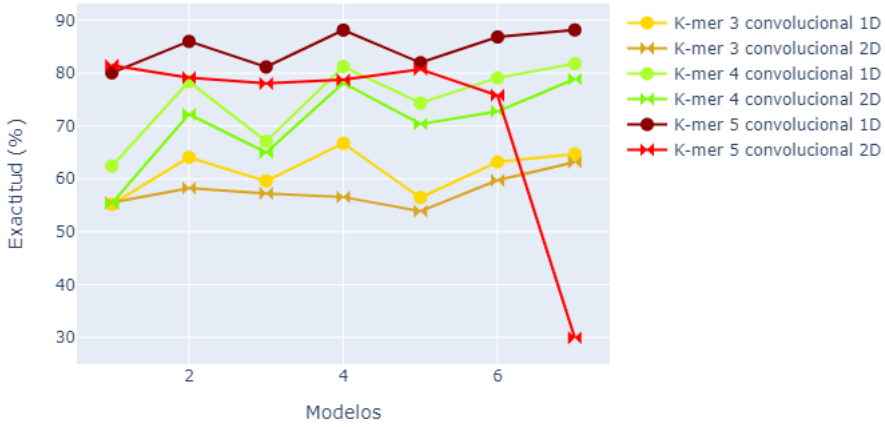


Figura 5.3.1: Exactitud de los modelos para cada k-mer y tipo de CNN.

Teniendo en cuenta los resultados de la red neuronal multicapa, se observa que, igualmente, para el k-mer 5 hay mayor exactitud que para los otros k-mers. Además, los modelos con CNN 2D tienen menor exactitud en comparación con CNN 1D. Con lo anterior, para este tipo de datos se obtiene un mejor desempeño de los modelos convolucionales observando las secuencias de datos como unidimensionales. Si se tiene en cuenta que una CNN 1D es más efectiva para obtener características en segmentos de longitud fija y, además, no importa dónde se encuentra esa característica en el segmento, entonces los resultados son acertados, pues describe la secuencia de k-mers que se usa.

En metodología se mencionó que hay una capa de agrupación después de cada capa convolucional. Sin embargo, se observó que al eliminar las capas de agrupación, el porcentaje de exactitud aumentó para los dos tipos de CNN como se puede observar en las figuras 5.3.2 y 5.3.3.

	Red neuronal convolucional 1D				Red neuronal convolucional 2D			
	Con pooling		Sin pooling		Con pooling		Sin pooling	
Modelos	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)
1	62.41	87.55	77.85	86	55.36	202.69	82.32	283.94
2	78.39	136.19	81.11	187.88	72.23	334.7	81.78	1569.5
3	67.09	114.16	79.95	137.12	64.95	234.25	81.79	735.7
4	81.25	105.29	84.41	119.26	78.12	240.18	84.15	304.3
5	74.36	126.84	77.9	134.1	70.35	384.2	77.36	565.95
6	79.10	163.89	83.70	207.82	72.76	480.4	81.74	748.7
7	81.78	112.6	85.08	125.56	80.26	263.71	84.95	389.7

Figura 5.3.2: Comparación de exactitud y tiempo de entrenamiento de los modelos con y sin capa de agrupamiento para k-mer 4.

	Red neuronal convolucional 1D				Red neuronal convolucional 2D			
	Con pooling		Sin pooling		Con pooling		Sin pooling	
Modelos	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)
1	80.04	155.5	85.4	161.5	81.47	768.6	87.18	1716
2	85.98	340.77	86.91	487.5	81.42	3203	88.34	5936
3	81.16	198.63	84.06	315.9	78.03	1009.8	87.58	5125
4	88.12	241.95	88.7	252.8	78.75	865	88.30	1145
5	81.56	210	84.5	257.8	80.7	2397	85.6	2652
6	86.83	341.25	87.18	467	75.8	3358.1	86.65	13575
7	88.16	238.9	88.25	248.42	29.95	1369	88.43	2699

Figura 5.3.3: Comparación de exactitud y tiempo de entrenamiento de los modelos con y sin capa de agrupamiento para k-mer 5.

Se puede observar que hay una mayor exactitud, tanto para CNN 1D y 2D de cada k-mer al eliminar las capas de agrupamiento. También, el tiempo de entrenamiento aumenta.

La razón de agregar una capa de agrupamiento es reducir el tamaño de las características recibidas de la capa convolucional, de esa forma, el tiempo es menor y las características son más robustas [20]. Al eliminar la capa de agrupación, el tiempo de entrenamiento aumenta, pues no se reduce el tamaño de las características. Es posible que, en este tipo de datos, la capa de agrupamiento no toma en cuenta características que tienen información importante, por eso se obtiene un mejor desempeño al eliminar la capa.

Los porcentajes resaltados son los modelos que obtuvieron mayor exactitud. De igual manera, se puede observar que el k-mer 5 sigue obteniendo un mejor

desempeño que el k-mer 4. Por consiguiente, se trabajará con el k-mer 5.

Además de eliminar la capa de agrupamiento, también se observó un sobreajuste en cada modelo. El modelo 4 de la CNN 1D obtuvo el mayor porcentaje de exactitud con 88.7 % en los datos de validación, pero en los datos de entrenamiento obtuvo un 97.03 %. Una solución al sobreajuste es aplicar penalizaciones a los parámetros de las capas durante la optimización. Esas penalizaciones alentarán a la red a mantener pesos pequeños durante el entrenamiento [21].
De la biblioteca de keras se usa el regularizador *kernel regularizer* con l2 que calcula la suma de los valores al cuadrado de los pesos [22].

Exactitud(%)		
Modelo	CNN 1D	CNN 2D
1	86.78	89.10
2	88.08	88.16
3	88.7	88.74
4	88.61	89.33
5	86.021	87.72
6	89.46	88.03
7	88.39	88.43

Cuadro 5.3.1: Exactitud de cada modelo al aplicar regularización l2.

Del cuadro 5.3.1 se observa que en ambos tipos de CNN la exactitud aumentó hasta un 2 %, y el modelo 6 de la CNN 1D y el modelo 4 de la CNN 2D obtuvieron la mayor exactitud comparada a la de los otros modelos. Por lo anterior, se procede a entrenar y evaluar ambos modelos para cada uno de los taxones.

Taxón	CNN 1D		CNN 2D	
	Exactitud (%)	Tiempo (s)	Exactitud (%)	Tiempo (s)
Clase	99.77	480.96	99.86	1284.26
Orden	97.63	1289.85	99.19	3484.53
Familia	93.7	2399.9	95.66	5904.8
Género	89.46	434.63	89.33	1101.79

Figura 5.3.4: Comparación de exactitud y tiempo de entrenamiento en cada taxón con los modelos escogidos.

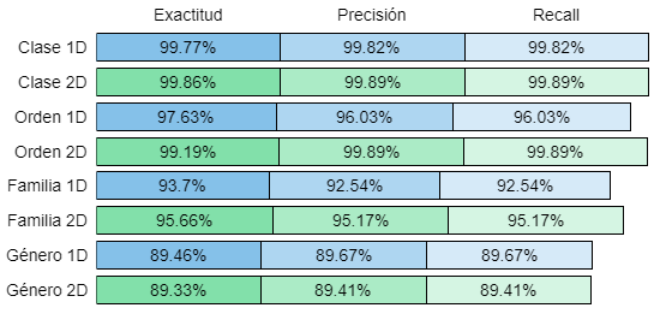


Figura 5.3.5: Resultado de métricas para los modelos escogidos.

En la figura 5.3.5 se puede observar el porcentaje de las métricas de los dos modelos escogidos para cada taxón. A pesar de tener un mejor resultado en el taxón género, la exactitud en los otros taxones no superó a la del modelo de la red neuronal multicapa. Sin embargo, se puede observar que las redes neuronales convolucionales también tienen un buen desempeño en cuanto a precisión, recall y exactitud.

5.4. Modelo escogido para la clasificación taxonómica de bacterias

Por último, se escogió el mejor modelo teniendo en cuenta exactitud, precisión, recall y tiempo de entrenamiento. Primero se tomó en cuenta el tiempo de entrenamiento de todos los modelos. La red neuronal multicapa tiene el menor tiempo, pues es la red más sencilla, seguida de la CNN 1D y 2D. Además, a pesar de que se obtenían mejores resultados al eliminar las capas de agrupamiento, el tiempo aumentaba mucho, en algunos casos se observó el doble de tiempo.
Ahora, al analizar las métricas, la red neuronal multicapa obtiene una mayor exactitud en los tres primeros taxones, mientras que el taxón género obtiene una mayor exactitud en los modelos de CNN 1D y 2D. Sin embargo, la diferencia es del 1 %. Es por eso que se escoge la red neuronal multicapa.

En la figura 5.4.1 se observa la exactitud y las pérdidas del modelo escogido al entrenarlo por 100 épocas.

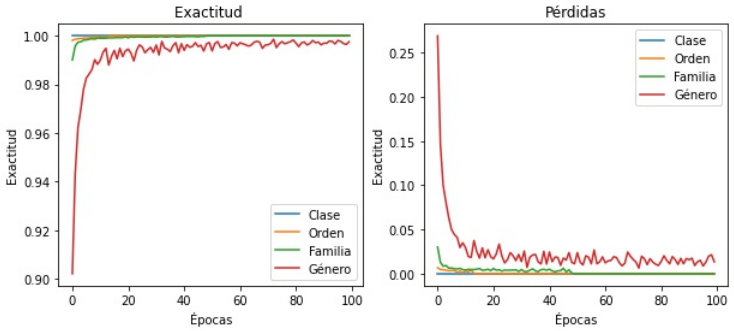


Figura 5.4.1: Exactitud y pérdidas durante el entrenamiento para cada taxón.

Luego del entrenamiento, se evaluó el modelo con los datos de prueba. En el cuadro 5.4.1 se observa la exactitud y el tiempo de entrenamiento para cada taxón.

Taxón	Exactitud (%)	Tiempo (s)
Clase	100	163.64
Orden	99.21	450.68
Familia	96.73	821.3
Genero	88.21	161.2

Cuadro 5.4.1: Exactitud al evaluar los datos de prueba

También se observaron las demás métricas, pero esta vez para cada una de las clases.

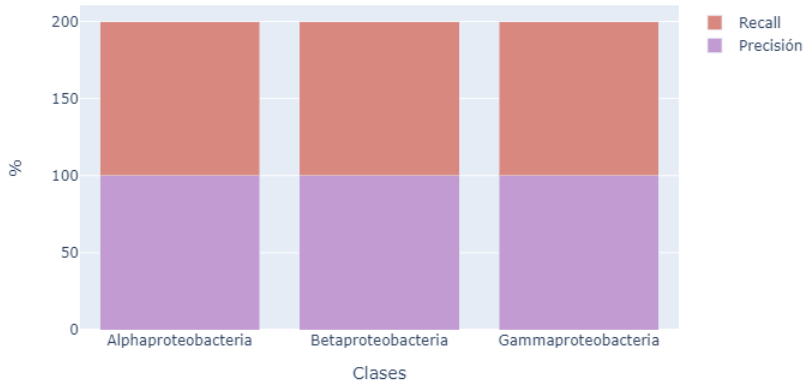


Figura 5.4.2: Precisión y recall en el taxón clase.

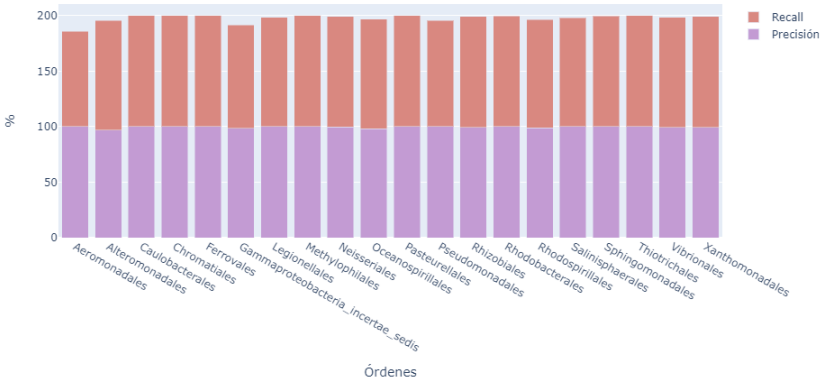


Figura 5.4.3: Precisión y recall en el taxón orden.

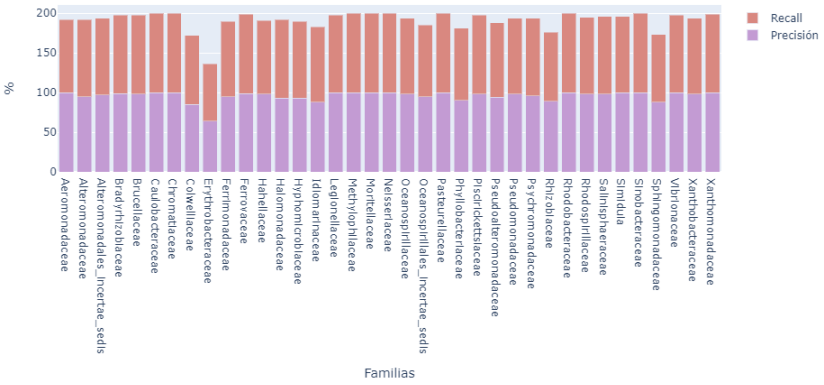


Figura 5.4.4: Precisión y recall en el taxón familia.

Figura 5.4.5: Precisión y recall en el taxón género.

Las figuras 5.4.2, 5.4.3, 5.4.4 y 5.4.5 muestran la precisión y recall de cada una de las bacterias por taxón.
En el taxón clase y orden se observa que ambas métricas son altas, es decir, el modelo clasifica correctamente las clases.
En el taxón familia hay dos clases con precisión y recall más bajos que las demás, estas son: Erythrobacteraceae y Colwelliaceae. Observando la distribución de los datos, esas dos bacterias no son las que menor número de datos tiene. De hecho, la bacteria Moritellaceae es la de menor número de datos y obtuvo una precisión y recall más altos. Entonces, puede existir dos posibles explicaciones. La primera es que los datos sintéticos para las dos bacterias no tuvieron la

información adecuada. La segunda es que la secuencia de ADN no tenía dicha información.
En el taxón género varias bacterias obuvieron bajos porcentajes de precisión y recall, especialmente: Leisingera y Pseudophaeobacter, estas no hacen parte de las familias que tuvieron baja precisión y recall. De igual forma, las dos posibles explicaciones dadas anteriormente pueden ser la causa de obtener resultados bajos.

6. CONCLUSIONES

La representación vectorial usada para las secuencias de ADN condujo a buenos resultados: alto porcentaje de exactitud, precisión y recall en cada taxón. Es decir, los vectores k-mers si logran extraer características importantes de las secuencias, pero depende del tamaño de k. Teniendo en cuenta los resultados, todos los modelos desarrollados presentaron mejor desempeño con $k = 5$, y el desempeño iba disminuyendo a medida que el tamaño del k-mer bajaba. Podría concluirse que a mayor tamaño del k-mer, mejor desempeño. Sin embargo, no se analizaron tamaños de k-mers mayores a 5, por lo cual no es conveniente deducir tal afirmación.

Los modelos de redes neuronales convolucionales 1D tenían un mejor desempeño a comparación de las 2D, además de tener mayor exactitud, el tiempo de entrenamiento era menor. Es decir, la secuencia de datos unidimensionales extraía mayores características que una secuencia de datos bidimensional. Sin embargo, se observó que al eliminar la capa de agrupamiento, la exactitud de los modelos de CNN 2D aumentó hasta un 5 % y un poco menos en la CNN 1D. Es decir, al reducir el mapa de características de las capas convolucionales, también se perdía información importante. Esto puede deberse a que las imágenes de entrada eran muy pequeñas. Por esa razón, las capas de agrupamiento disminuían el desempeño de los modelos. Sin embargo, al eliminarlas, el tiempo de entrenamiento aumentó hasta más del doble en algunos modelos. Ignorando el tiempo de entrenamiento, las CNN 2D sin capa de agrupamiento tenían hasta un 1 % más de exactitud que las CNN 1D, es decir, la secuencia de datos bidimensionales también ofrecen un buen desempeño como los unidimensionales.

El desempeño de los modelos también disminuía a medida que bajaba el orden de los taxones. Esto era de esperarse, puesto que entre más bajo sea el orden del taxón, las características de la secuencia de ADN serán más dificiles de encontrar. Además, el número de datos para cada género de bacteria era muy bajo. Aun así, se logró una exactitud del 88 % en el taxón género, además de que la mayoría de las clases tenían precisión y recall altos. El taxón de mayor orden no presentó dificultades, pues se obtuvo una exactitud del 100 %.

De los modelos desarrollados, el que presenta un mejor desempeño es el de la red neuronal multicapa, el cual es el modelo más sencillo, y por lo tanto, el de menor costo computacional. De la literatura consultada en todo el trabajo, no se encontró que usaran redes neuronales multicapa, la mayoría usaban redes neuronales convolucionales u otro tipo de aprendizaje profundo. Por esa razón, es importante destacar el hecho de que un modelo sencillo sea suficiente para extraer características de secuencias de ADN. Sin embargo, no es algo concluyente, puesto que solo se compara con redes neuronales convolucionales y no otro tipo de algoritmos de machine learning.

7. BIBLIOGRAFÍA

[1] "ADN (Ácido Desoxirribonucleico)", *National Human Genome Research Institute*, 2020. [Online]. Available: https://www.genome.gov/es/genetics-glossary/ADN-acido-Desoxirribonucleico

[2] J. Rodríguez, "Secuenciación de Genomas", España, 2004, pp. 285-310. Available: http://arbor.revistas.csic.es/index.php/arbor/article/viewFile/609/611

[3] "Genomas Bacterianos — Department of Microbiology", *College of Agriculture and Life Science*, 2020. [Online]. Available: https://micro.cornell.edu/research/epulopiscium/espanol/genomas-bacterianos/.

[4] S. Baron, "Classificationïn *Medical Microbiology*, The University of Texas Medical Branch at Galveston, 1996.

[5] Fiannaca, A., La Paglia, L., La Rosa, M. et al. "Deep learning models for bacteria taxonomic classification of metagenomic data". *BMC Bioinformatics 19*, 2018, pp. 198. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2182-6#rightslink

[6] Al-Ajlan, A., El Allali, A. "CNN-MGP: Convolutional Neural Networks for Metagenomics Gene Prediction". *Interdiscip Sci Comput Life Sci 11*, 2019, pp. 628–635. Available: https://doi.org/10.1007/s12539-018-0313-4

[7] Lo Bosco G., Di Gangi M.A. "Deep Learning Architectures for DNA Sequence Classification", in *Fuzzy Logic and Soft Computing Applications*, Petrosino A., Loia V., Pedrycz W. WILF 2016. Lecture Notes in Computer Science, 2017, vol 10147. Springer, Cham.

[8] A. Vázquez, "Técnicas de secuenciación de nueva generación para el estudio del microbioma humano", Universidad Complutense, 2016.

[9] J. Brownlee, "Crash Course On Multi-Layer Perceptron Neural Networks", *Machine Learning Mastery*, 2016. [Online]. Available: https://machinelearningmastery.com/neural-networks-crash-course/

[10] "Multi-Layer Neural Network", *Unsupervised Feature Learning and Deep Learning Tutorial*. [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

[11] "Keras documentation: Layer activation functions", *Keras.io*. [Online]. Available: https://keras.io/api/layers/activations/

[12] "Convolutional Neural Network Architecture: Forging Pathways to the Future", *MissingLink.ai*, 2020. [Online]. Available: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/

[13] J. Brownlee, "How Do Convolutional Layers Work in Deep Learning Neural Networks?", *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[14] "Keras Conv1D: Working with 1D Convolutional Neural Networks in Keras", MissingLink.ai, 2020. [Online]. Available: https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

[15] J. Brownlee, "SMOTE for Imbalanced Classification with Python", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[16] Rizzo, Riccardo & Fiannaca, Antonino & La Rosa, Massimo & Urso, Alfonso, "A Deep Learning Approach to DNA Sequence Classification", in *Computational Intelligence Methods for Bioinformatics and Biostatistics: 12th International Meeting*, CIBB 2015, Naples, Italy, 2016, pp. 129-140.

[17] Chaudhary, N. Sharma, A. Agarwal, P. Gupta, A. Sharma, V, "16S Classifier: A Tool for Fast and Accurate Taxonomic Classification of 16S rRNA Hypervariable Regions in Metagenomic Datasets", 201, Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0116106

[18] E. Weitschek, F. Cunial and G. Felici, "Classifying Bacterial Genomes with Compact Logic Formulas on k-Mer Frequencies", *25th International Workshop on Database and Expert Systems Applications*, Munich, 2014, pp. 69-73.

[19] J. Brownlee, "How to use Data Scaling Improve Deep Learning Model Stability and Performance", *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/

[20] J. Brownlee, "A Gentle Introduction to Pooling Layers for Convolutional Neural Networks", *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[21] J. Brownlee, "Use Weight Regularization to Reduce Overfitting of Deep Learning Models", *Machine Learning Mastery*, 2018. [Online]. Available: https:

//machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/

[22] "Keras documentation: Layer weight regularizers", *Keras.io*. [Online]. Available: https://keras.io/api/layers/regularizers/