UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

APRENDIZAJE AUTOMÁTICO DE LAS REGLAS DE ENSAMBLAJE DE COMUNIDADES MICROBIANAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN TECNOLOGÍA

PRESENTA:

WALTER ANDRÉ ROSALES REYES

DIRECTOR DE TESIS:

DR. MARCO TULIO ANGULO BALLESTEROS

MARZO 2021

JURIQUILLA QUERÉTARO

Jurado calificador:
Secretario
Dr.
Olivares Pinto
Ulises

Presidente
Dr.
Velasco Hernández
Jorge X.

Vocal
Dr.
Santana Cibrian
Mario

Sustituto 1
Dr.
Angulo Ballesteros
Marco Tulio

Sustituto 2
MCIM
González Castro
Carlos Alberto

APRENDIZAJE AUTOMÁTICO DE LAS REGLAS DE ENSAMBLAJE DE COMUNIDADES MICROBIANAS
LICENCIATURA EN TECNOLOGÍA, MARZO 2021

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

BLVD. UNVERSITARIO 3600
76230 JURIQUILLA QUERÉTARO

"Lo mejor de todo era que ya no eran necesarios los trajes espaciales que cubrían todo el cuerpo; la atmósfera, si bien todavía era irrespirable, tenía una densidad que permitiría el empleo de simples mascarillas y tubos de oxígeno. Y aunque los microbiólogos no daban fechas exactas, habían prometido que al cabo de unas pocas décadas más hasta estos equipos se podrían descartar. Cepas de bacterias generadoras de oxígeno ya se habían soltado sobre la superficie de Ganímedes, la mayoría había muerto, pero algunas habían florecido, y la curva que ascendía poco a poco en la gráfica de análisis atmosférico era la primera prueba que se exhibía con orgullo a todos los visitantes de Dárdano."

2061: Odisea Tres (Arthur C. Clarke, 1987)

"It is certain that there may be extraordinary mental activity with an extremely small absolute mass of nervous matter: thus the wonderfully diversified instincts, mental powers, and affections of ants are notorious, yet their cerebral ganglia are not so large as the quarter of a small pin's head. Under this point of view, the brain of an ant is one of the most marvelous atoms of matter in the world, perhaps more so than the brain of a man."

Charles Darwin, 1872

Resumen ejecutivo

Las comunidades microbianas en la biosfera del planeta y el cuerpo humano tienen un rol fundamental en la integridad de los ecosistemas y la salud humana. Es necesario desarrollar un entendimiento predictivo de la dinámica de comunidades microbianas, permitiéndonos entender cómo se ensamblan en primera instancia. Sin embargo, entender la dinámica de comunidades microbianas es un problema difícil debido al gran número de especies que frecuentemente tienen, la gran diversidad e incertidumbre de los procesos que gobiernan su dinámica, y las limitaciones experimentales que impiden realizar mediciones frecuentes de la abundancia de sus especies.

Para contribuir a resolver las dificultades arriba mencionadas, en esta Tesis se construye una nueva arquitectura de Deep Learning para aprender las reglas de ensamblaje de una comunidad microbiana a partir de "experimentos" con únicamente las abundancias iniciales y finales correspondientes a distintas colecciones de especies. Estas reglas de ensamblaje determinan cuándo una colección de especies coexiste y, cuando no, determinan qué especies se extinguen. Se validó el desempeño de la arquitectura propuesta usando datos *in-silico* de sistemas ecológicos teóricos y empíricos de ocho especies bacterianas heterotróficas de suelo. Se muestra que la arquitectura propuesta aprende de manera precisa las reglas de ensamblaje usando un número pequeño de experimentos, y que tiene un desempeño muy superior a comparación de arquitecturas clásicas de Deep Learning. En conclusión, el presente trabajo muestra cómo los algoritmos de Deep Learning pueden ser una herramienta fundamental para avanzar el conocimiento de comunidades microbianas.

Índice general

1 Introducción y motivación

1.1 Motivación

Las comunidades microbianas son conjuntos de distintas especies de microbios (en general es posible usar con clasificaciones taxonómicas especificas) que coexisten e interactúan en un cierto tiempo y espacio [8]. Estas comunidades se encuentran en casi todos los hábitats del planeta, desde aguas termales hasta los lagos de la Antártica y el cuerpo humano. En particular, en años recientes se ha investigado su contribución dentro de huéspedes de otros reinos ecológicos [11]. En todos estos casos se ha encontrado que estas comunidades realizan funciones muy importantes para la sustentabilidad de sus hábitat o huéspedes [11].

Una de las áreas de investigación más importantes en el actualidad sobre la importancia de las comunidades microbianas es el estudio de la microbiota en seres vivos y el impacto sobre sus huéspedes. En particular, en el estudio del efecto de la disbiosis (desbalance en la comunidad microbiana) de la microbiota se ha encontrado una importante asociación con el surgimiento de enfermedades y el estado general de salud del huésped [27]. La alteración de la microbiota consiste en la modificación de las abundancias de las especies que la conforman. Esto lleva al huésped a un estado de disbiosis (dysbiosis en inglés), el cual está altamente relacionado con alteraciones en el metabolismo [27, 9], causantes de padecimientos como la obesidad [34, 35, 7], el envejecimiento [41, 9, 27], así como en propiciar la diabetes [39]. También, es ampliamente investigada su relación con alteraciones en el sistema inmune [4, 23, 9],

padecimientos crónico degenerativos como el cáncer [30, 38, 27], así como afectaciones al sistema nervioso a través del *eje intestino-cerebro* (GBA, del inglés para gut-brain axis) [42, 21, 19].

Estas comunidades microbianas son igualmente relevantes en aplicaciones industriales como los bioreactores [1, 16]. En un bioreactor, las bacterias funcionan como los generadores de energía y por tanto su dinámica se relaciona con la eficiencia y eficacia, así como en oportunidades de optimización. De forma similar, en los campos de cultivo las comunidades microbianas representan una de las áreas de mayor importancia para manipular y regular los procesos de cultivo, ya que son fundamentales para la integración de nutrientes al suelo, así como en propiciar el crecimiento de las plantas cultivadas [12, 33, 11].

Actualmente existen diversas técnicas que buscan manipular las comunidades microbianas con la intención de llevarlas de un cierto estado no deseado a uno deseado [2]. Un claro ejemplo de esto son las terapias microbianas de prebióticos y probióticos, en las cuales se busca integrar a la comunidad un conjunto de cepas o especies particulares. También existen estrategias que buscan implantar a un huésped una comunidad completa (ya existente en otro espécimen) sobre las comunidades que ya poseía a través de trasplantes fecales [15]. Estrategias similares se toman también en áreas como la agricultura, donde la inoculación de cepas bacterianas es una práctica común para agregar sus beneficios al suelo de cultivo, o la búsqueda de trasplantar una microbiota de unas zonas de cultivo a otras [33]. Para que estas estrategias se puedan llevar a cabo con éxito, es necesario conocer la dinámica subyacente a las comunidades microbianas sobre las que se están implementando.

1.2 Reglas de ensamblaje de sistemas ecológicos

Desde los comienzos del estudio de las poblaciones biológicas, Charles Darwin argumentó que las especies en un sistema ecológico se encuentran en una “lucha constante por la sobreviencia” que determina qué especies están presentes en cierto tiempo y lugar. Esta observación fue hecha de forma más precisa en los trabajos pioneros de Jared Diamond en 1975 [10], cuando observó que las especies de aves encontradas en distintas islas parecían obedecer ciertas

reglas de ensamblaje que describían qué colecciones de especies podían ser observadas (i.e., coexistían) y cuales no. En la actualidad, el estudio de las reglas de ensamblaje de sistemas ecológicos es un campo muy activo de investigación [13]. En su formulación moderna, las reglas de ensamblaje tienen el objetivo de determinar cuáles colecciones de especies coexisten —y cuáles y qué especies se extinguen— a partir de conocer qué especies están inicialmente presentes y su abundancia inicial.

Las reglas de ensamblaje pueden describirse de manera más formal como sigue. Consideremos una comunidad de N especies. Denotamos por $\mathbf{x}(0) = (x_1(0), x_2(0), \cdots, x_N(0)) \in \mathbb{R}^N_{\geq 0}$ la abundancia inicial de especies, con $x_i(0)$ la abundancia inicial de la i-ésima especie. La colección de especies iniciales puede describirse como un vector $\mathbf{z}(0) = \mathbf{z}(\mathbf{x}(0)) \in \{0,1\}^N$, donde su i-ésimo componente z_i se determina como

$$z_i = \begin{cases} 1, & \text{si } x_i \neq 0, \\ 0, & \text{si } x_i = 0. \end{cases}$$

Es decir, $z_i(0) = 1$ solo si la i-ésima especie está presente al momento inicial. Las reglas de ensamblaje son entonces el mapeo

$$\varphi : \mathbb{R}^N_{\geq 0} \to \mathbb{R}^N_{\geq 0}, \tag{1.1}$$

que asocia a cada abundancia inicial de especies $\mathbf{x}(0)$ su abundancia "final" o en estado estable $\mathbf{x}_f = \varphi(\mathbf{x}(0)) \in \mathbb{R}^N_{\geq 0}$. En particular, las reglas de ensamblaje determinan la colección final de especies como $\mathbf{z}_f = \mathbf{z}(\mathbf{x}_f) = \mathbf{z}(\varphi(\mathbf{x}_0)) \in \{0,1\}^N$. De esta forma, es posible determinar qué colecciones de especies coexisten (i.e., si $\mathbf{z}_f = \mathbf{z}(0)$), o cuales colecciones no coexisten y qué especies se extinguen (i.e., la i-ésima especie se extingue si $\mathbf{z}_{i,f} \neq \mathbf{z}_i(0)$).

En la Figura 1.1 se ilustra el concepto de reglas de ensamblaje en una comunidad de $N = 3$. En este caso, existen $2^N - 1 = 7$ colecciones distintas de especies. Estas reglas de ensamblaje están relacionadas por la dinámica que describen las especies según el modelo considerado, de forma que de una colección de especies se puede pasar a otra (e.g. si una se extingue).

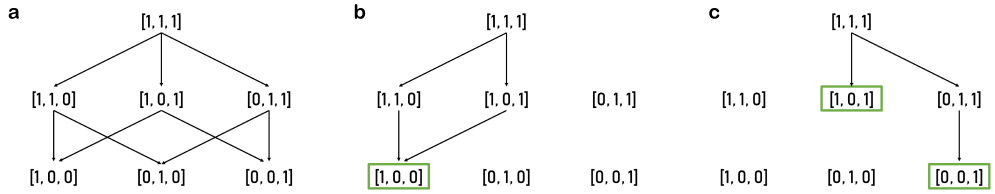

Figura 1.1 Ilustración de las reglas de ensamblaje para todas las colecciones de especies para 3 especies. (a) Muestra todos los posibles caminos entre estados en las coleccione de especies, el fluir de un estado de nivel superior a uno inferior hace referencia a la extinción de una especie, de forma que se pueden comenzar con todas y llegar hasta la dominancia de una única sobre las demás. (b) Ilustración de la existencia de múltiples caminos que puedan llevar a un mismo estado de equilibrio estable final (colección de especies encerrada en un recuadro verde). (c) Se muestra la posibilidad de existencia de múltiples estados de equilibrio estables finales con las mismas colecciones de especies.

Podemos representar todas las posibles colecciones de especies en forma de una red acíclica no dirigida (red tipo árbol). Esto permite visualizar que, del caso donde la colección de especies es aquella donde las N especies están presentes (el nodo superior), se puede fluir a cualquiera de los estados posibles subsecuentes a través del proceso de extinción. Además, es importante notar que no existe un único camino para llegar a un mismo estado, considerando el caso en donde una única especie se puede extinguir a la vez.

1.3 Retos para aprender las reglas de ensamblaje de comunidades microbianas

En sistemas ecológicos, el enfoque clásico para inferir o aprender las reglas de ensamblaje es construir un modelo matemático parametrizado de su dinámica poblacional, usualmente en forma de una ecuación diferencial ordinaria [8]. Este enfoque es de hecho más general, pues no solamente provee la abundancia final asociada a cierta abundancia inicial de especies que requiere la Ec. (1.1), sino toda la historia temporal de cómo la abundancia cambia con el tiempo. Este enfoque clásico ha sido empleado para tratar de inferir las reglas de ensamblaje de comunidades microbianas, tanto en trabajos clásicos [36] como recientes [14, 37]. Sin embargo, este enfoque tiene dos problemas fundamentales que limitan su aplicación a comunidades microbianas [40].

Primero, este enfoque clásico requiere suponer a-priori una modelo parametrizado para la dinámica poblacional del sistema. En comunidades microbianas (y en todas las comunidades ecológicas de forma general), su dinámica poblacional está influenciada por diversos factores muy poco caracterizados, incluyendo las interacciones intra/entre especies, gradientes de temperatura, gradientes de recursos, ubicación geográfica, características espaciales del hábitat entre muchas otras [13]. Por esta razón, exceptuando en las condiciones más simples, se vuelve muy difícil elegir un modelo parametrizado para la dinámica poblacional de comunidades microbianas. Segundo, garantizar una correcta inferencia de los parámetros del modelo requiere mediciones frecuentes en el tiempo de la abundancia de las especies en la comunidad [3], lo cual es muy difícil y costoso de hacer en la mayoría de las comunidades microbianas. De hecho, el procedimiento experimental más usado para estudiar comunidades microbianas reporta únicamente la abundancia inicial y final de especies en la comunidad. Este segundo problema se vuelve particularmente serio en comunidades microbianas con un gran número de especies. Por ejemplo, incluso si se supone que las interacciones entre cada par de especies pueden describirse por un solo parámetro, en una comunidad de N especies existirán N^2 de esos parámetros. Esto muestra que, conforme crece el número de especies, el número de parámetros sólo para un factor aumenta de forma cuadrática.

En este sentido, se requieren nuevos enfoques para aprender las reglas de ensamblaje de comunidades microbianas, que permitan sobrellevar al menos las dos limitaciones arriba mencionadas.

1.4 Contribuciones principales

En este trabajo presentamos el uso de herramientas de Deep Learning para aprender las reglas de ensamblaje de comunidades microbianas a partir únicamente de datos. Deep Learning es una metodología basada en redes neuronales profundas que ha sido muy exitosa para aprender mapeos complicados en diversas áreas de la ciencia [26]. Como datos de entrenamiento se consideraron varios “experimentos” que contienen solamente abundancias iniciales y finales de distintas colecciones de especies. Se comienza mostrando que la arquitectura clásica de Residual Networks (ResNet), que ha sido muy exitosa

para resolver una gran diversidad de problemas [17], tiene un desempeño muy pobre para aprender las reglas de ensamblaje. Este resultado es esperable debido a que las reglas de ensamblaje tienen una estructura muy particular derivada de la dinámica ecológica subyacente. Entonces, proponemos una nueva arquitectura que llamamos *ResNet ecológica* además de funciones de perdida y algoritmos de entrenamiento diseñados específicamente para resolver el problema de aprender las reglas de ensamblaje. Validamos esta arquitectura usando sistemas ecológicos teóricos y empíricos, encontrando que la ResNet ecológica puede aprender con alta precisión las reglas de ensamblaje de sistemas ecológicos con un número pequeño de experimentos, sin necesidad de conocer la dinámica poblacional subyancente.

1.5 Organización de la tesis

El resto de la Tesis está organizada como sigue. El Capítulo 2 presenta una breve descripción de los algoritmos de deep learning, así como las diversas componentes necesarias para entrenarlos. En él también se presenta una introducción al modelo de dinámica poblacional conocido como Lotka-Volterra generalizado y condiciones particulares en sistemas de dos especies.

Los resultados en este trabajo están divididos en dos capítulos. En el Capítulo 3 se presenta el desarrollo de la nueva arquitectura y función de pérdida propuesta para lograr el objetivo de aprender la reglas de ensamblaje. Este capítulo también describe el proceso, algoritmos y parámetros utilizados para el entrenamiento de la ResNet ecológica. El capítulo 4 contiene los resultados de los procesos de validación que fueron aplicados a esta nueva arquitectura, tanto con datos de origen sintético como a partir de parámetros de una comunidad microbiana empírica.

Finalmente, en el capítulo 5 se presentan las conclusiones a los resultados de este trabajo al igual que propuestas para trabajo futuro derivados del modelo de la *ResNet ecológica* y sus aplicaciones.

2 Metodología

2.1 Algoritmos de *deep learning*

2.1.1 Arquitecturas de redes neuronales

Dentro de los métodos de machine learning se encuentran los de deep learning, que consisten en aplicar funciones matemáticas mediante un conjunto de capas sucesivas aplicadas a los datos de entrada. La estructura típica de una red neuronal profunda (DNN, por sus siglas en inglés para deep neural network) es la mostrada en la Fig. 2.1a denominada *feedforward neural network* usualmente denotadas simplemente como DNN (red neuronal profunda) [32], donde se pueden apreciar las capas ocultas (*deep layers*) en color amarillo, mientras que los círculos azules representan los valores de entrada y los de salida en verde.

Otro algoritmo de deep learning son las residual neural networks (ResNet), la Fig. 2.1b muestra una ilustración de la arquitectura convencional de este tipo de algoritmos. En este caso se puede notar que se cuenta con una entrada (círculo azul) a la cual se le suma el resultado de aplicarle una función, de forma que se obtiene un nuevo valor del vector con las mismas dimensiones. Este nuevo vector puede ser considerado tanto como el vector de salida, como el de la nueva entrada, es decir existe una retroalimentación, ya que la salida de una iteración anterior se convierte en la entrada de la iteración siguiente. Dicho ciclo se repite un número fijo de veces hasta que se decide tomar el valor

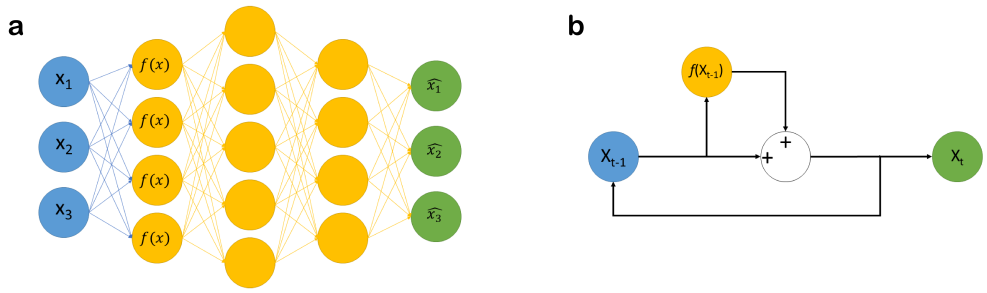

Figura 2.1 Ilustración de algoritmos de deep learning. (a) Estructura convencional de una red neuronal profunda (DNN) del tipo feedforward. Los círculos azules representan las entradas, los amarillos son las "capas ocultas" (*hidden layers*) donde se aplican operaciones a los valores de la capa inmediata anterior, mientras que los círculos verdes representan la salida de la red, en este caso un vector de dimensión 3. (b) Estructura convencional de una residual neural network (ResNet). Al leer el diagrama de izquierda a derecha se puede ver que se tiene un vector de entrada inicial, al cual se aplica una función $f(\mathbf{x})$ y posteriormente se suman los resultados; ese vector resultante es ahora tanto el vector de salida como el nuevo de entrada, de forma que existe una retroalimentación entre la salida y la entrada. Este proceso es cíclico, de forma que el número de veces en que se va a repetir este ciclo interno se define con anterioridad y establece cuándo se tomará el vector de salida (en verde) como la nueva entrada, o como la salida definitiva.

de salida como el valor resultante de la arquitectura [32]. A diferencia del algoritmo de las DNN donde son los pesos de las aristas (las conexiones entre capas) los valores (denominados parámetros) que se ajustan y entrenan para mejorar el rendimiento de la red, en el caso de la ResNet estos parámetros se encuentran dentro de la función. Ambos algoritmos tienen métodos de ajuste diferentes dados los parámetros a entrenar.

El método de las ResNet ofrece una gran ventaja en comparación con las DNN dado que existe una retroalimentación entre la salida de la iteración pasada y la nueva iteración, haciendo una dependencia muy estrecha entre la nueva salida y la entrada, por lo que la red se va quedando con información almacenada de la iteración anterior, mientras que en el caso de la DNN únicamente recibe información de la capa anterior inmediata la cual únicamente contiene la información de una función aplicada a la información en la iteración anterior. En la Ec. (2.1) se observa la ecuación que describe la arquitectura de una ResNet. Este proceso iterativo también puede ser interpretado como el "transcurso del tiempo" dado que se tiene una especie de memoria, a la cual se le añade nueva información con cada nueva iteración (con cada intervalo de tiempo). Esta interpretación resulta ser muy conveniente para nuestro propósito, ya que el objetivo de este trabajo es simular el comportamiento de una dinámica

a lo largo del tiempo, de forma que se tenga una condición inicial que fluye a lo largo del tiempo y llega a un estado final. Esto permite considerar que el algoritmo correcto para esta representación sea el de la ResNet bajo la interpretación del paso del tiempo.

Formalmente, una arquitectura ResNet consta de una función parametrizada $f_\theta : \mathbb{R}^N \to \mathbb{R}^N$, donde θ es el vector de parámetros. Aquí, N es el numero de nodos de cada capa (que es igual en todas las capas de la red). Con esta función, el "estado" $h \in \mathbb{R}^N$ de la i-ésima capa oculta se calcula como

$$h_i = h_{i-1} + f_\theta(h_{i-1}). \tag{2.1}$$

2.1.2 Función de pérdida

Para entrenar los modelos de machine learning es necesario tener una función de pérdida L, la cual le va a indicar al algoritmo qué tan bueno es su ajuste a los datos [32]. Formalmente, dado un conjunto de datos $\mathcal{D}$ y un valor θ de los parámetros de la red, la pérdida es la función: $L : (\theta, \mathcal{D}) \longrightarrow \mathbb{R}$ El valor $L(\theta, \mathcal{D})$ se conoce como la "pérdida" para los datos $\mathcal{D}$ con el parámetro θ. Por definición, la función de pérdida se construye tal que si las predicciones del algoritmo coinciden con los datos, entonces la pérdida es $L = 0$. En general, la función de pérdida es un parámetro de diseño que puede ser utilizado para ajustar el aprendizaje del algoritmo. De esta forma, el proceso de "aprendizaje" de la red consisten en ajustar sus parámetros para minimizar la función de pérdida.

Para nuestro objetivo, es también necesario dividir esta función de pérdida en dos partes, las cuales evalúan distintos aspectos del rendimiento del modelo.

2.1.3 Algoritmos de entrenamiento

El proceso de entrenamiento de un algoritmo de deep learning consiste en el proceso iterativo de cálculo de la predicción del modelo, evaluar el error a partir de la función de pérdida y actualizar los parámetros de la red para la siguiente iteración. Estas iteraciones se conocen como épocas[1] , y consisten en

realizar el proceso anteriormente señalado con el conjunto de datos utilizados para el entrenamiento [32].

Optimizador

El proceso de actualizar los parámetros de la red es llevado a cabo por algoritmos conocidos como optimizadores, los cuales en cada época modifican los valores de estos parámetros según el error que haya tenido el modelo en dicha iteración [32]. Uno de los algoritmos básicos es el conocido como GD (en inglés gradient descent) que consiste en tomar una fracción del error y restarlo a los parámetros del modelo, esta fracción depende del gradiente evaluado en cada parámetro[2] multiplicado por un hiperparámetro[3] conocido como *learning rate*[4] [31]. El algoritmo de GD se puede describir con la siguiente regla iterativa:

$$\theta_{i+1} = \theta_i - \eta \, \nabla_{\theta_i} L(\theta_i),$$

donde $\eta > 0$ es el hiperparámetro conocido como learning rate, $\theta_i \in \mathbb{R}^P$ al conjunto de parámetros a actualizar en la i-ésima época y L es la función de pérdida previamente definida [31].

En la Fig. 2.2 se puede observar el funcionamiento de un optimizador y el efecto de la magnitud del learning rate durante el entrenamiento. En el eje de las ordenadas se grafica el error de la predicción del modelo, cada uno de los puntos amarillos representan un conjunto de parámetros y las flechas

[1] Se denomina época a una iteración durante el proceso de entrenamiento de un modelo de deep learning. Este entrenamiento consiste, de forma general, en los siguientes pasos: ingresar los datos al modelo y calcular la predicción, evaluar el error de esta predicción contra los resultados esperados mediante una función de pérdida, y por último, actualizar los parámetros del modelo a partir del error con un algoritmo de optimización (conocido como optimizador).

[2] En cada época se evalúa el gradiente de la función de pérdida con respecto a los parámetros del modelo. Este proceso se lleva a cabo mediante el uso de una técnica denominada *backpropagation* en inglés, y que consiste en evaluar el gradiente de la función de pérdida con respecto a los parámetros del modelo, propagando el error desde la capa de salida hacia la de entrada a través de las capas ocultas (ver Fig. 2.1a). [31]

[3] Un hiperparámetro es aquel valor que se define antes de comenzar a entrenar y se utiliza para dirigir algunas características del entrenamiento del modelo.

[4] El *learning rate* es aquel valor escalar que determina el tamaño del paso que se va a tomar en dirección del gradiente.

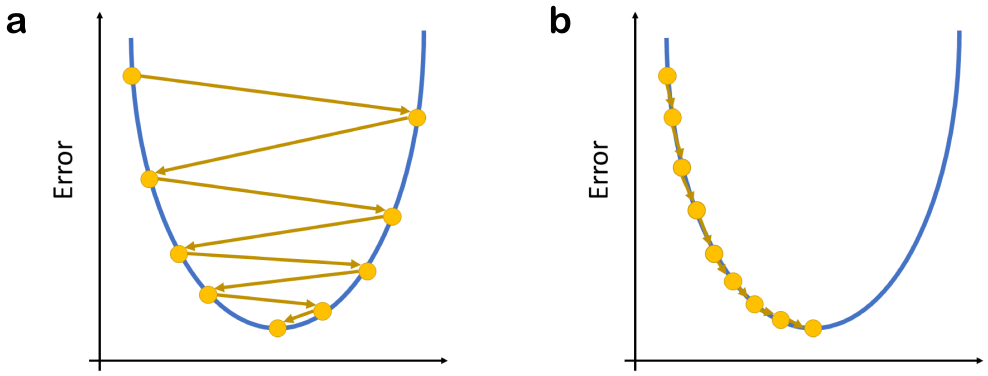

Figura 2.2 Representación gráfica del funcionamiento de un optimizador. (a) Indica el proceso iterativo con un parámetro de learning rate alto, lo que hace que el comportamiento del error sea altamente cambiante. (b) Hace referencia al comportamiento del error cuando se utilizan un parámetro de learning rate chico.

hacen referencia al proceso iterativo de entrenamiento, mientras que las curvas indican el gradiente de los parámetros. En ambas figuras se puede observar que existe un mínimo global, el cual se obtiene a partir del conjunto de parámetros que mejor rendimiento darían al modelo sobre los datos. Ese punto es al que se desea llegar, y el optimizador lo que busca es ir modificando los parámetros de forma que se vayan acercando cada vez más hacia ese mínimo global.

Se puede notar que hay dos posibles extremos para la magnitud del learning rate. Si se utiliza uno muy grande se tienen saltos en los parámetros igualmente grandes, lo que puede llevar a no llegar al mínimo e incluso aumentar el error en la predicción. La Fig. 2.3a muestra que, al utilizar un valor muy grande, es posible que al actualizar los parámetros se de un incremento tan grande que provoque incluso que el error sea mayor (esta iteración se indica con las flechas verdes), lo que a su vez afectaría la siguiente iteración. Esto provocaría un peor desempeño en la siguiente iteración y así sucesivamente. En el caso contrario (ver Fig. 2.3b), cuando se usa un valor pequeño puede entonces que el optimizador llegue a un mínimo local y, al ser tan pequeño tanto el gradiente como el learning rate, los parámetros quedan estancados, sin llegar en ningún momento al óptimo global.

En la literatura existen múltiples optimizadores [31], todos ellos variantes modificadas del GD antes descrito. Dos de los más utilizados son el SGD (del inglés para *stochastic gradient descend*) y ADAM[5]. Sin embargo, las ejempli-

[5]El algoritmo de SGD consiste en la misma regla de actualización de parámetros que el GD, con la particularidad de que, en lugar de evaluar el gradiente y la pérdida con respecto a

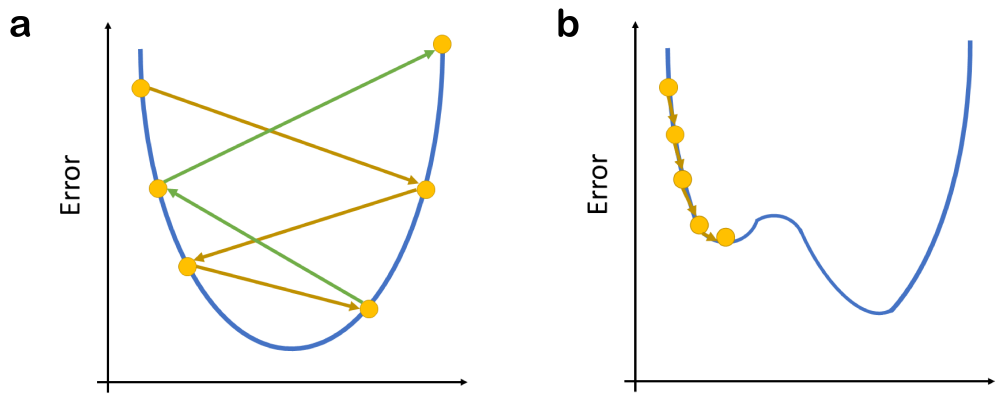

Figura 2.3 Limitaciones de la magnitud del learning rate (a) Uso de un learning rate muy grande puede provocar que el error incremente, alejándose del óptimo global. (b) Magnitudes muy pequeñas de learning rate pueden provocar que el algoritmo se quede estancado en un mínimo local, impidiendo llegar a los parámetros óptimos.

ficaciones en la Fig. 2.3 aplican en todos los casos, de forma que la elección del optimizador, así como el hiperparámetro learning rate tiene que ser con base en prueba y error, comparando los rendimientos y eligiendo aquel que obtenga el mejor o que cumpla con los objetivos esperados.

Hiperparámetros

El ya mencionado learning rate es uno de los varios hiperparámetros utilizados para el entrenamiento de los modelos de deep learning. Consisten en aquellos parámetros que se definen antes del proceso de entrenamiento y que son utilizados para dirigir el entrenamiento del modelo, lo cual determinará el rendimiento del mismo. Otro hiperparámetro a considerar en el proceso de entrenamiento es el número de épocas que se va a iterar. Esto tiene una estrecha relación con el optimizador y el learning rate ya que, como se puede deducir a partir de las Fig. 2.2 y 2.3 el llegar al error óptimo también depende del número de iteraciones que se lleven a cabo.

La selección de estos hiperparámetros, es de forma general arbitraria y en ocasiones se determinan a través de procesos meta-heurísticos [32]. Se llevan

todos los datos de entrada, lo hace con un único par de datos a la vez. ADAM por otro lado, es una versión con mayores modificaciones, que toma en cuenta iteraciones anteriores a partir del cálculo del momento (i.e. la segunda derivada) y ajustes dinámicos del learning rate.
La definición completa y las reglas de actualización para estos y otros optimizadores se pueden encontrar en [31].

a cabo diferentes pruebas para determinarlos, buscando aquellos que en combinación den como resultado el mejor rendimiento del modelo.

Meta-learning: reptile

Uno de los factores más importantes frente al entrenamiento de un algoritmo de deep learning es la inicialización de los parámetros, lo cual tiene un impacto determinante ante el proceso de entrenamiento. Este proceso, al igual que la selección de hiperparámetros es totalmente arbitrario, no existiendo una forma general aplicable a todos los casos. A pesar de que existen varios algoritmos que ofrecen diferentes formas de inicializar los parámetros de un modelo, ninguno puede asegurar el mejor rendimiento en todos los casos, de forma que se han desarrollado algoritmos que mejoraran el entrenamiento a partir de la inicialización de los parámetros y que son conocidos como algoritmos de *meta-learning* [28].

El algoritmo reptile es uno de ellos, el cual consiste básicamente en realizar una optimización en la inicialización de los parámetros cada vez que se vuelve a entrenar la red. Por tanto ahora se tendrá un nuevo hiperparámetro que sería el número de épocas (iteraciones) que se entrenará este nuevo algoritmo [28]. Su funcionamiento es ejemplificado en la Fig. 2.4 donde se puede ver que en cada época de reptile[6]. se actualizan los parámetros iniciales del modelo, de forma que en la siguiente iteración pueda tener un mejor desempaño final al entrenar por el mismo número de épocas. Este proceso se puede ver también en el pseudocódigo 2.1.

Reptile también mejora la capacidad de generalizar, es decir, de no caer en *overfitting*[7] sobre los datos de entrenamiento, puesto que permite que se reordenen antes de volver a entrenar con ellos e incluso modificar el tamaño del conjunto de datos para entrenar. Esto permite que el modelo no se ajuste perfectamente a un conjunto de datos, sino que en cada actualización se ajuste a los datos de forma diferente.

[6] En este trabajo se considera el valor de *epochs* como el número de épocas que se entrena la ResNet ecológica de forma convencional, mientras que el número de épocas reptile se denomina como *repochs*. Se puede apreciar la diferencia entre una y otra en el pseudocódigo 2.1.

[7] El *overfitting* o sobreajuste sucede cuando el modelo logra ajustar de buena manera los datos con los cuales fue entrenado, pero al momento de evaluarlo en otro conjunto de datos no utilizado para entrenar, no obtiene buenos re-

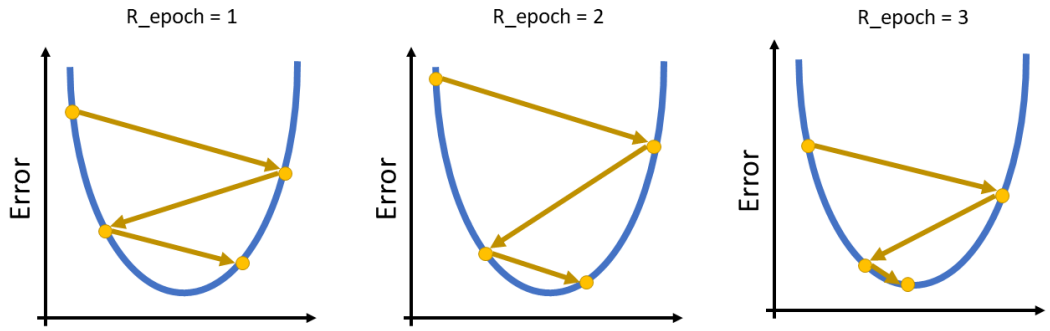

Figura 2.4 Representación de tres épocas del algoritmo reptile (R_epoch hace referencia a una época del reptile). En la primera se inicializa en un cierto conjunto de parámetros por tres épocas, posteriormente se actualizan los parámetros iniciales y se empieza un segundo entrenamiento de tres épocas y así sucesivamente. Se puede observar cómo conforme reptile mejora la inicialización de los parámetros, el error final va disminuyendo.

Algoritmo 2.1: Pseudocódigo reptile

```
for repoch in repochs do
    Guardar los parámetros iniciales;
    for epoch in epochs do
        Entrenar modelo de forma convencional con el optimizador
         ADAM;
    end
    Actualizar parámetros iniciales según rendimiento actual del
     modelo;
end
```

Datasets

Con la intención de evaluar el rendimiento de los modelos de deep learning, resulta necesario dividir el conjunto de datos totales en tres partes, denominadas sets de: train o entrenamiento, validation o validación y test o prueba; no forzosamente teniendo el mismo número de datos (tamaño del set) todos. Una manera de repartirlos es mostrada en la Fig. 2.5 donde primero se divide el conjunto de datos totales entre un set de entrenamiento y otro de prueba, el primero se utilizará para entrenar el modelo, el segundo para evaluar su rendimiento una vez entrenado. En el caso del set de entrenamiento, comúnmente se hace otra división entre el verdadero set de entrenamiento y el de

sultados dado que "aprendió" la distribución paricular de los datos de entrenamiento. Dos sitios con explicaciones a más detalle son las siguientes: `https://en.wikipedia.org/wiki/Overfitting` y también `https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690`

validación, la diferencia es que el conjunto de datos de entrenamiento son aquellos con los que se va a entrenar el modelo realmente, mientras que el de validación se utiliza para medir el rendimiento de las predicciones a lo largo del proceso de entrenamiento.

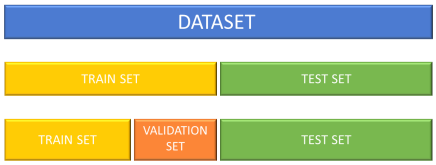

Figura 2.5 División del conjunto de datos (dataset) en tres partes. El conjunto de entrenamiento y prueba pueden tener el mismo tamaño, mientras que validación es un subconjunto de datos de entrenamiento.

Esta división es esencial puesto que se espera que las predicciones del modelo sean generales, y no casos particulares únicamente del conjunto de datos utilizados para entrenar. De esta forma se tienen entonces conjuntos de datos que son "vistos" por el modelo y son sobre los cuales se ajustan los parámetros (set de entrenamiento), otro conjunto con el cual se hace la evaluación (set de validación) del rendimiento de la red durante el proceso de entrenamiento con la intención de obtener un resultado similar al que se busca obtener en el set de prueba, mayormente usado para verificar que la red no esté cayendo en *overfitting* o sobreajuste de los datos de entrenamiento o por el contrario un *underfitting* que sería cuando el modelo no es capaz de ajustar prácticamente ninguno de los datos. Y finalmente el conjunto de datos donde se prueba la capacidad de predicción del modelo sobre datos que jamás fueron utilizados durante su entrenamiento (set de prueba) pero que son aplicados para conocer el rendimiento final del modelo una vez que se haya decidido detener el entrenamiento.

2.2 Modelos matemáticos de dinámica poblacional de sistemas ecológicos

2.2.1 Modelo Lotka-Volterra Generalizado

Los estados de equilibrio estables y los caminos posibles están definidos por las interacciones entre las especies, ya sea con ellas mismas o con otras. Dichas interacciones pueden ser descritas y simuladas a través de un modelo matemá-

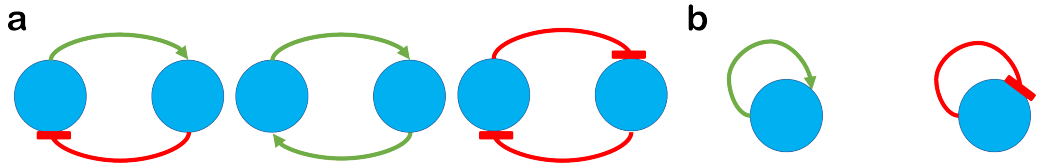

Figura 2.6 Interacciones interespecíficas e intraespecíficas en comunidades ecológicas. (a) Ejemplos de interacciones provenientes de la matriz **A**. (b) Ejemplos de interacciones descritas por el vector **r**. Las aristas rojas indican una inhibición (interacción negativa) y las verdes promoción (interacción positiva).

tico conocido como GLV (del inglés, generalized Lotka-Volterra). Este modelo es una ecuación diferencial de primer orden que relaciona la abundancia de una especie a lo largo del tiempo, su tasa de reproducción así como de las interacciones y abundancias de las demás especies con las que interactúa en la comunidad. Con más precisión, denotando como $x_i(t)$ la abundancia de la i-ésima especie al tiempo t y como $\mathbf{x} = (x_1, x_2, \cdots, x_N) \in \mathbb{R}^N$ el vector con la abundancia de todas las especies del sistema, el modelo GLV toma la forma:

$$\dot{\mathbf{x}}(t) = \mathbf{x}(t) \odot f(\mathbf{x}(t)), \tag{2.2}$$

donde $f(\mathbf{x}) = \mathbf{A}\,\mathbf{x} + \mathbf{r}$, con $\mathbf{A} \in \mathbb{R}^{N\times N}$ y $\mathbf{r} \in \mathbb{R}^N$. Arriba, el símbolo $\mathbf{x} \odot \mathbf{v}$ denota el producto entrada-a-entrada entre vectores $\mathbf{x}, \mathbf{v} \in \mathbb{R}^N$ (i.e., producto de Hadamard). Dadas unas condiciones iniciales de abundancia $\mathbf{x}(0) \in \mathbb{R}^N$ para todas las especies, se pueden simular y determinar sus abundancias en cualquier tiempo resolviendo la Eq. (2.2).

En la Ec. (2.2) se muestra que $f(\mathbf{x})$ depende de una matriz cuadrada **A** que corresponde a la matriz de interacciones existentes entre las N especies consideradas en el modelo, mientras que **r** indica el crecimiento de la especie de forma intrínseca (tasa de reproducción), sin dependencia de las interacciones con las demás especies. La matriz **A** es realmente una matriz de adyacencia de una red que define las interacciones entre todas las especies, que pueden ser de dos tipos: promotoras o inhibitorias (ver Fig. 2.6)

2.2.2 Modelo logístico

El modelo logístico es un modelo derivado del GLV, donde se toma en consideración la capacidad máxima de carga del ambiente, lo que hace referencia a que en la naturaleza las poblaciones no crecen infinitamente, sino que están limitadas por su ambiente (e.g. recursos, espacio disponible, etc.). Es decir, el modelo permite el crecimiento de la población si ésta está por debajo de un cierto parámetro conocido como *capacidad máxima*, o provoca la muerte de los individuos si la población está por encima de dicha capacidad. Este parámetro de capacidad máxima está descrito en el modelo logístico por el vector **k** mostrado en el sistema de ecuaciones siguiente para dos especies:

$$\begin{aligned} \dot{\mathbf{x}}_1 &= x_1(r_1 - \frac{x_1}{k_1} - a_1\frac{x_2}{k_1}) \\ \dot{\mathbf{x}}_2 &= x_2(r_2 - \frac{x_2}{k_2} - a_2\frac{x_1}{k_2}) \end{aligned} \tag{2.3}$$

Fácilmente se puede ver que el sistema en la Ec. (2.3) es un caso particular del de la Ec. (2.2) donde $\mathbf{A} = -\begin{bmatrix} \frac{1}{k_1} & \frac{a_1}{k_1} \\ \frac{a_2}{k_2} & \frac{1}{k_2} \end{bmatrix}$ y $\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$. De esta manera la dinámica poblacional es la misma, pero dados estos parámetros se restringe la población máxima de la especie x_i al valor de la constante k_i. Cabe destacar que este sistema supone que cada especie presente puede sobrevivir de forma independiente de la otra, lo que lleva a que cada especie pueda modificar su abundancia dada únicamente su propia existencia.

2.2.3 Coexistencia y extinción en sistemas de dos especies

En comunidades de dos especies, existen únicamente cuatro posibles escenarios, que en notación del vector de colección de especies serían los mostrados en la Fig. 2.7. En estos vectores se ejemplifican los dos casos posibles en los que puede caer la dinámica del modelo GLV: coexistencia y extinción.

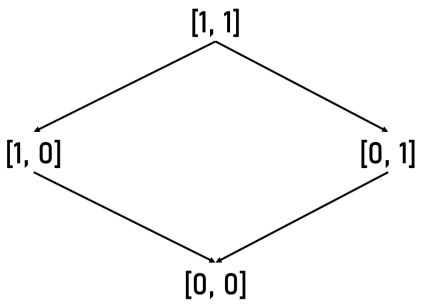

Figura 2.7 Vectores de colección de especies posibles para una comunidad de dos especies.

El caso de coexistencia es aquel donde la dinámica muestra que ambas especies prevalezcan, es decir que ninguna excluye a la otra, sino que ambas coexisten en un estado estable. Sin embargo, como resulta evidente a partir del modelo GLV y el diagrama de la Fig. 2.7, si alguna de las dos especies no está presente (tiene abundancia inicial cero) la dinámica llevará hasta un estado estable correspondiente a la existencia de la única especie inicial. En el diagrama, éste estado está representado por el vector de colección de especies superior (ilustrado como [1, 1]), el cual indica que ambas especies están presentes y coexisten, pudiendo notar que no hay forma de llegar a él a menos que se comience en él.

A partir del diagrama también se puede notar que, dado el vector de colección de especies completo (ambas especies presentes) es posible llegar a los tres estados restantes; esto es a través del proceso de extinción. Este proceso considera aquella dinámica en la que una de las especies reduce su abundancia hasta quedar totalmente extinguida y por tanto quedar eliminada del vector de colección de especies.

3 Diseño de una nueva arquitectura

En el capítulo 4 de esta tesis, mostraremos que la arquitectura clásica de una ResNet presentada en la sección 2.1.1 proporciona un muy pobre desempeño para aprender las reglas de ensamblaje de comunidades ecológicas. Esto nos motiva a construir una nueva arquitectura que llamamos "ResNet ecológica" que subsana estas limitaciones. En el capítulo 4 mostraremos cómo la ResNet ecológica propuesta tiene un aprendizaje muy eficiente en comparación con otras arquitecturas.

3.1 ResNet ecológica

Proponemos que el modelo GLV mostrado en Ec. (2.2), como una ecuación diferencial, se puede ver como un proceso iterativo. De forma que se obtiene la ecuación siguiente:

$$\mathbf{x}_t = \mathbf{x}_{t-1} \odot g(\mathbf{x}_{t-1}),$$

donde $t = 1, 2, \cdots$ se puede considerar como el índice de la t-ésima capa.

Esta nueva arquitectura usa la idea de discretizar una ecuación diferencial de forma que pase de un régimen continuo sobre el tiempo a intervalos discretos. Así mismo, interpreta la arquitectura de una ResNet como un proceso iterativo

a lo largo del tiempo. Sin embargo, dado que se trabaja con comunidades ecológicas, existen ciertas restricciones que se tienen que considerar, esencialmente sobre la función $g(\mathbf{x})$ para que cumpla con las restricciones mismas del modelo GLV.

Proponemos que la arquitectura de la "ResNet ecológica" luzca de la forma siguiente:

$$\mathbf{x}_t = \mathbf{x}_{t-1} \odot e^{f_\theta(\mathbf{x}_{t-1})}, \tag{3.1}$$

donde $f_\theta : \mathbb{R}^N \to \mathbb{R}^N$ es una función arbitraria que depende de los parámetros $\theta \in \mathbb{R}^P$ a ajustar. Esta propuesta consta de dos ingredientes claves, los cuales asimilan las dos restricciones indispensables que se deben de cumplir al trabajar con comunidades ecológicas y que resultan de la naturaleza misma del fenómeno:

1. la multiplicación de vectores entrada-a-entrada. Esto permite que si una especie está originalmente ausente, ella permanecerá ausente para cualquier valor de t;

2. la función f_θ entra en la arquitectura como $e^{f_\theta(\mathbf{x})}$, asegurando que nunca existirá un valor negativo para la abundancia de una especie.

De forma que nuestro modelo es una modificación novedosa sobre el algoritmo de una ResNet, pero adecuado a la forma matemática que un modelo GLV presenta.

En el resto de este trabajo, se eligió una función afín $f_\theta(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, de forma que los parámetros a actualizar θ son: $\mathbf{W} \in \mathbb{R}^{N\times N}$ y $\mathbf{b} \in \mathbb{R}^N$ los cuales serán ajustados a lo largo del entrenamiento. Como se muestra en capítulos posteriores, esta elección es suficientemente buena para aprender las reglas de ensamblaje en los sistemas estudiados.

Una ilustración de cómo resulta la arquitectura de la recién propuesta ResNet ecológica se puede ver en la Fig. 3.1. El círculo de la izquierda representa

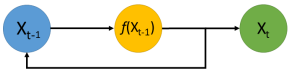

Figura 3.1 Ilustración del modelo de la ResNet ecológica. Se observa que la entrada es multiplicada elemento a elemento consigo misma después de aplicarle una cierta función f_θ. El resultado de esa operación ahora es la entrada de la siguiente iteración, tal y como lo indica la retroalimentación. Cabe destacar que el círculo amarillo donde la función f_θ está contenida puede representar cualquier conjunto de operaciones, incluso una NN.

la entrada, el amarillo la función que se multiplica elemento a elemento (producto de Hadamard) con la entrada y que posteriormente pasa a la salida representada por el círculo verde. Como anteriormente se mencionó, tiene su ciclo de retroalimentación iterativo similar al de las ResNet convencionales.

Cabe destacar que la función f_θ puede ser tan compleja como se desee, incluso llegar a ser una DNN, sin embargo, para los propósitos de este trabajo se ocupó la función mostrada en Ec. (3.1), aunque queda de manifiesto la escalabilidad en complejidad del modelo de la ResNet ecológica.

3.2 Función de pérdida

3.2.1 MSD: Distancia entre distribuciones

Como ya se mencionó en secciones anteriores, una de las funciones de pérdida más usuales es el error cuadrático medio (MSE), el cual calcula la norma dos entre la predicción del modelo y el valor correcto, denotada de la siguiente forma:

$$MSE = \frac{1}{M}\sum_{i=0}^{M}(\hat{y}_i - y_i)^2$$

donde M es la cantidad de muestras, $\hat{y}_i$ es la predicción del modelo para la i-ésima muestra mientras que y_i es el valor real de la i-ésima muestra. De esta forma se calcula la diferencia que existe entre el valor predicho por el modelo

y el valor real. Con el entrenamiento del modelo se busca minimizar el error, de forma que el modelo entrenado sea capaz de replicar lo mejor posible el modelo de los datos de entrenamiento. Este caso es altamente útil cuando la comparación es entre valores escalares, sin embargo, en nuestro modelo estamos trabajando con vectores, por lo que una diferencia escalar no puede ser aplicada.

Cabe destacar que la función aplicada en el MSE puede ser interpretada como el cuadrado de la distancia euclidiana entre dos puntos en una dimensión (escalares), de ser así, es posible aplicar la misma idea a nuestra arquitectura, utilizando la función más general posible de la distancia, considerando entonces la distancia entre puntos de un hiper-espacio para vectores de dimensión N.

Proponemos entonces que la función de pérdida que mida la distancia entre los puntos (estados) finales y los predichos por nuestra arquitectura sea la distancia euclidiana, la cual está definida para vectores de dimensión igual o mayor a 1, lo cual la hace escalable para cualquier dimensión de nuestro vector de especies. Esta función de pérdida en adelante será mencionada como MSD que podría ser entendida como (Mean Squared Distance) y se encuentra definida por la Ec. (3.2).

$$MSD = \frac{1}{M}\sum_{i=0}^{M} d(\hat{\mathbf{y}}_i - \mathbf{y}_i) \tag{3.2}$$

3.2.2 $Loss_{\varepsilon}$: Predicción de colecciones de especies

Al ser la predicción de las reglas de ensamblaje uno de los objetivos esenciales de nuestra arquitectura, consideramos fundamental el agregar un factor en la función de pérdida que penalice el comportamiento de estas reglas durante el entrenamiento. Como fue mencionado, las reglas de ensamblaje hacen referencia a la presencia o ausencia de una especie, denotado por el vector de colección de especies **z** definido en el capítulo 1.2. En la naturaleza y en experimentos *in vitro* definir este vector resulta relativamente simple puesto que se cuenta con un número entero de especímenes por especie, que únicamente en

el en que una especie tenga población nula sería considerada como ausente. Sin embargo, eso implica que la distribución de poblaciones para toda especie es entera para cualquier tiempo, lo cual no se cumple en la mayoría de los casos cuando se trabaja con el modelo GLV, puesto que es un modelo continuo de ecuaciones diferenciales que produce una distribución de poblaciones en los reales positivos. Esto se convierte entonces en un problema práctico que encontramos fundamental resolver, proponiendo utilizar un parámetro ε que sirva como umbral (o *threshold* en inglés) de forma que ahora el vector $\mathbf{z}_\varepsilon$. Formalmente, definimos esta función como $\delta_\varepsilon : \mathbb{R}^N \to \{0,1\}^N$, donde la i-ésima entrada esta dada por

$$\delta_{i,\varepsilon}(\mathbf{x}) = \begin{cases} 1, & \text{si } x_i \geq \varepsilon, \\ 0, & \text{si } x_i < \varepsilon. \end{cases} \tag{3.3}$$

Usualmente, denotaremos $\mathbf{z}_\varepsilon = \delta_\varepsilon(\mathbf{x})$.

Esto implica ahora que el parámetro ε va a ser el que defina si una población está presente o no. Esto tiene fuertes implicaciones ya que ahora se considera que existe una población si el modelo predice un valor mayor a un cierto umbral ε, y que no existe en caso contrario. El parámetro ε debería determinarse *a priori*, pero para lograrlo sería necesario llevar a cabo experimentos y compararlos con los resultados de las ecuaciones y evaluar el valor "experimental" para cada conjunto de especies. Realizar estas validaciones requeriría trabajo experimental adicional, lo cual no es siempre posible de hacer. Para resolver (parcialmente) esta cuestión, se propone considerar un "barrido" de parámetros ε, como se explicada a continuación.

Por ello proponemos implementar una función de pérdida que considere la cuestión relacionada con el parámetro ε. En el apéndice A1 se muestra un ejemplo del cálculo de la función de pérdida $loss_\varepsilon$ sobre un conjunto de datos. El objetivo de esta función de pérdida consiste en hacer un barrido sobre un dominio $[\varepsilon_{min}, \varepsilon_{max}]$ determinado para cada entrenamiento y conjunto de datos[1], de forma que podamos penalizar la predicción de nuestra arquitectura ante cualquier valor de ε sobre dicho dominio. De manera formal la función de pérdida está dada por:

[1] A lo largo de este trabajo se consideró el rango como [0, *valmax*], donde *valmax* es el valor de la abundancia máxima obtenida en estado estable para el conjunto de especies y parámetros presentes en la simulación de los datos.

$$
\begin{aligned}
loss_{\varepsilon}(\mathcal{D}, \hat{\mathcal{D}}) &= \frac{1}{\varepsilon_{max} - \varepsilon_{min}} \int_{\varepsilon_{min}}^{\varepsilon_{max}} d_{\varepsilon}(\mathcal{D}, \hat{\mathcal{D}}) d\varepsilon, \\
&\approx \frac{1}{\lfloor \frac{\epsilon_{max} - \epsilon_{min}}{\Delta\varepsilon} \rfloor + 1} \sum_{\varepsilon} d_{\varepsilon}(\mathcal{D}, \hat{\mathcal{D}}) \Delta\varepsilon,
\end{aligned}
\tag{3.4}
$$

aproximando el barrido a través de una discretización, donde $\mathcal{D}$ es el conjunto de estados finales para cada especie por cada experimento y $\hat{\mathcal{D}}$ es el conjunto de datos con las predicciones de nuestro modelo para cada especie por cada experimento. Por tanto $\mathcal{D}$ y $\hat{\mathcal{D}}$ son matrices de $N \times M$ siendo N el número de especies y M el de muestras.

La función d_{ϵ} mostrada a continuación permite medir la fracción de predicciones incorrectas del modelo entre el total de predicciones realizadas, permitiendo evaluar el rendimiento del modelo en el intervalo $[\varepsilon_{min}, \varepsilon_{max}]$ y posteriormente ajustar el modelo con respecto a este resultado:

$$
d_{\epsilon}(\mathcal{D}, \hat{\mathcal{D}}) = \frac{1}{M} \sum_{M} \frac{1}{N} \sum_{N} |\delta_{\varepsilon}(\mathcal{D}) - \delta_{\varepsilon}(\hat{\mathcal{D}})|, \tag{3.5}
$$

donde δ_{ε} hace referencia a la Ec. (3.3).

3.2.3 Cálculo completo del error

Como se mencionó al principio de esta sección, la función de pérdida que utilizamos consiste en dos partes: MSD y $Loss_{\varepsilon}$. Para hacer el cálculo del error entre las predicciones de nuestro modelo y los valores reales se tienen entonces dos contribuciones: distancia entre los puntos (abundancias) y predicciones incorrectas en la colección de especies. Por último, es necesario sumar ambas contribuciones para calcular el error total del modelo, lo que resulta en la siguiente ecuación:

$$
error = MSD(\mathcal{D}, \hat{\mathcal{D}}) + \lambda Loss_{\varepsilon}(\mathcal{D}, \hat{\mathcal{D}}), \tag{3.6}
$$

que es aquella utilizada como función de perdida para el resto de este trabajo.

En esta función utilizamos λ como un parámetro de regularización, el cual sirve para modificar la importancia que tiene el error en las predicciones de las colecciones de especies frente a qué tan cerca se encuentran del valor real. Esto permite contar con un parámetro para regular el objetivo de aprendizaje de la ResNet ecológica durante el entrenamiento.

3.3 Entrenamiento

3.3.1 Tamaño de dataset

Uno de los objetivos de este trabajo es encontrar el número de experimentos[2] (datos) necesarios para que las predicciones por parte de la ResNet ecológica sean correctas para un conjunto de especies. En particular identificar si existe una relación entre el número de datos utilizados para entrenar y el número de especies involucradas en la dinámica poblacional. Para lograrlo se utilizó como estrategia el entrenamiento de diferentes modelos de ResNet ecológica para diferentes tamaños de set de entrenamiento, tal y como se ilustra en la Fig. 3.2 en la cual se puede apreciar que el dataset completo, se dividió en dos conjuntos: set de entrenamiento y set de prueba. El tamaño del dataset fue de 1100 experimentos, de los cuales 1000 fueron destinados al set de prueba y 100 al set de entrenamiento[3]. De este último, se crearon siete subconjuntos con diferente cantidad de experimentos tomados del set de entrenamiento de forma aleatoria, siendo el set de prueba común entre todos los subconjuntos generados.

[2] Un experimento se considera como el equivalente a una simulación de las ecuaciones del modelo GLV con un conjunto de parámetros, o un experimento (*in vivo* o *in vitro*) de un conjunto de especies con una cierta condición inicial. Es decir, para un experimento con N especies, se tienen dos vectores: $\mathbf{x}_0$ y $\mathbf{x}_f$, $\mathbf{x} \in \mathbb{R}^N$, los cuales hacen referencia al vector de abundancias iniciales, y el vector de abundancias en el estado final resultantes, respectivamente. Cada uno de estos experimentos funciona como "un dato" para el entrenamiento y evaluación de la ResNet ecológica.

[3] Dado que en este trabajo los experimentos fueron creados *in silico*, se optó por tener un set de validación mucho mayor al de entrenamiento, con la finalidad de obtener una validación más exhaustiva. Hay que considerar que en casos con datos experimentales es posible que obtener esta gran cantidad de datos de validación sea inviable, sin embargo, estos no son relevantes al momento de entrenar la ResNet ecológica.

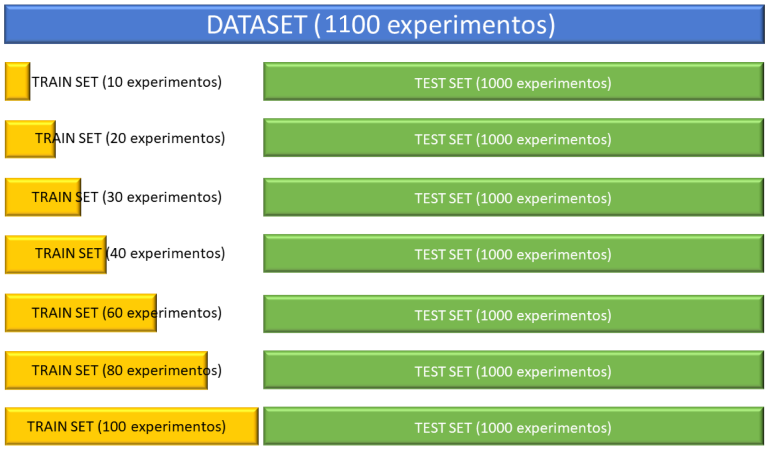

Figura 3.2 División del dataset completo (1100 experimentos) para los diferentes sets de entrenamiento y el set de prueba. Se puede apreciar que el set de prueba es igual en todos los casos, mientras que los de entrenamiento modifican la cantidad de experimentos utilizados para entrenar en cada uno de los casos.

3.3.2 Algoritmo de entrenamiento

En el entrenamiento de la ResNet ecológica se utilizó el algoritmo de meta-learning de reptile (ver Pseudocódigo 2.1), donde el modelo fue entrenado por un número *epochs* de épocas internas por cada época reptile (*repoch*).

Los resultados iniciales a partir de entrenar con un número fijo de épocas se ilustran en la Fig. 3.3, quedando en evidencia que 10 repochs son insuficientes para entrenar la ResNet, existiendo un underfitting evidente y un error prácticamente total en todos los casos (ver Fig. 3.3a-b). Mientras que el entrenamiento con 20 repochs, ilustrado en las Fig. 3.3c-d muestra un mejor rendimiento, reduciendo considerablemente el error frente al caso de 10. Sin embargo en estas figuras se nota un comportamiento particular en donde los mejores resultados se obtienen con un menor número de experimentos tanto en el set de entrenamiento como en el de prueba, lo que resulta contraintuitivo debido a que, a mayor cantidad de datos se espera una mejor capacidad de aprendizaje por parte del modelo. Aunado a esto, los resultados con 20 repochs presentan una dispersión muy amplia, abarcando prácticamente todos los valores de error; un indicativo del posible underfitting en el entrenamiento en gran parte de los modelos, especialmente en los de mayor número de experimentos.

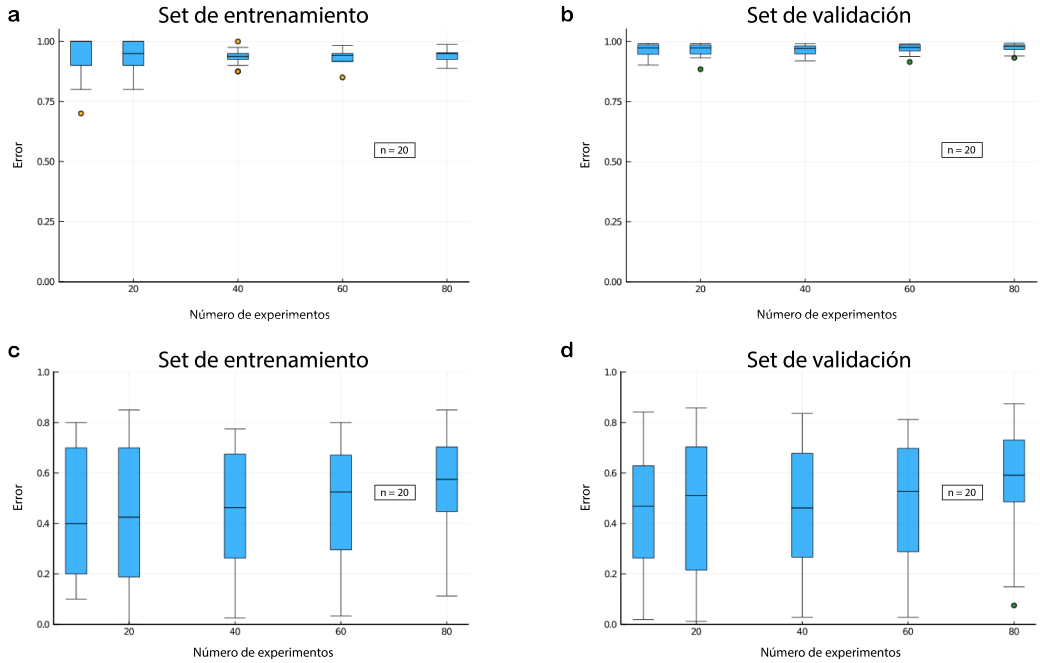

Figura 3.3 Rendimiento de la ResNet ecológica entrenada por un número de repochs independiente del número de experimentos. (a) y (b) son los resultados de entrenar cada modelo por 10 repochs para el set de entrenamiento y el de prueba respectivamente. En (c) y (d) es entrenando cada modelo por 20 repochs.

Dadas estas observaciones, se propuso que, para obtener un mejor desempeño por parte de la ResNet, el número de épocas a entrenar podría estar en función del número de experimentos. Para comprobar esto se llevaron a cabo dos pruebas: en una de ellas se entrenó la ResNet por un número fijo de *repochs*, en el segundo se dejó entrenar el modelo hasta cumplir con un valor de error mínimo. Ambas estrategias fueron aplicadas al set de entrenamiento con 10 experimentos, los resultados se muestran en la Fig. 3.4, donde (a) y (b) hacen referencia al caso en el que se entrenó por 10 repochs, mientras que (c) y (d) muestran el caso en que se dejó entrenar el modelo hasta obtener un error menor a a un valor deseado $e_{\text{máx}}$. En este trabajo elegimos $e_{\text{máx}} = 0.2$, aunque este valor no modifica cualitativamente la forma heurística de nuestros resultados durante el entrenamiento.

Una comparativa entre (a) y (c) deja ver que la red con un valor fijo de 10 repochs no logró entrenar totalmente, existiendo una gran dispersión en los resultados de entre los 20 modelos entrenados en comparación con la segunda estrategia. Para esta segunda estrategia se obtuvo como resultado que en promedio para lograr un error menor a 0.2 era necesario entrenar por 20 repochs. Aunado a estas estrategias, para los tamaños de set de entrenamiento

mayores a 10 se les dejó entrenar hasta obtener un error menor al error promedio del tamaño de set de entrenamiento inmediatamente anterior (e.g. en (a) para 20 experimentos cada modelo se entrenó hasta obtener un error menor a 0.48, que fue el error promedio de los resultados sobre el set de entrenamiento con 10 experimentos).

Al analizar los datos resultantes de ambas estrategias se encontró una tendencia entre el número de repochs y la cantidad de experimentos del set de entrenamiento, descrita por una función afín de la forma $ax + b$ donde b es el número base de repochs necesarias para el entrenamiento, x el número de experimentos en el set de entrenamiento y a el número de repochs que se aumentan conforme aumentan los experimentos. Como se observa en la Fig. 3.4, esta función permite a la ResNet ecológica obtener resultados similares e incluso mejores en el set de prueba que el de entrenamiento, comportamiento deseado, evitando complicaciones como el *overfiting* y el *underfitting* del modelo. En comparación con casos en los que únicamente se establecen valores fijos de repochs, donde se encontró el fenómeno de underfitting[4], la aproximación de una función afín presenta mejores resultados y sirve como una forma heurística inferida a partir de los datos para ajustar el número de repochs conforme al tamaño del set de entrenamiento.

Con base en los resultados obtenidos, se propone para las pruebas subsecuentes hacer uso de la ecuación afín antes descrita. De forma que para obtener el número de *repochs* a entrenar se puede utilizar la siguiente ecuación:

$$repochs = \lceil \frac{1}{10} experimentos + 21 \rceil \tag{3.7}$$

de forma que por cada 10 experimentos en el set de entrenamiento, es necesario entrenar una época reptile más el modelo, partiendo con 21 repochs como base, parámetros obtenidos a partir de los resultados sobre los modelos ilustrados en la Fig. 3.4.

[4]Los resultados del entrenamiento son iguales e incluso peores en el set de prueba, además de presentar una gran variabilidad en ambos casos, situación que se observa en la Fig. 3.3.

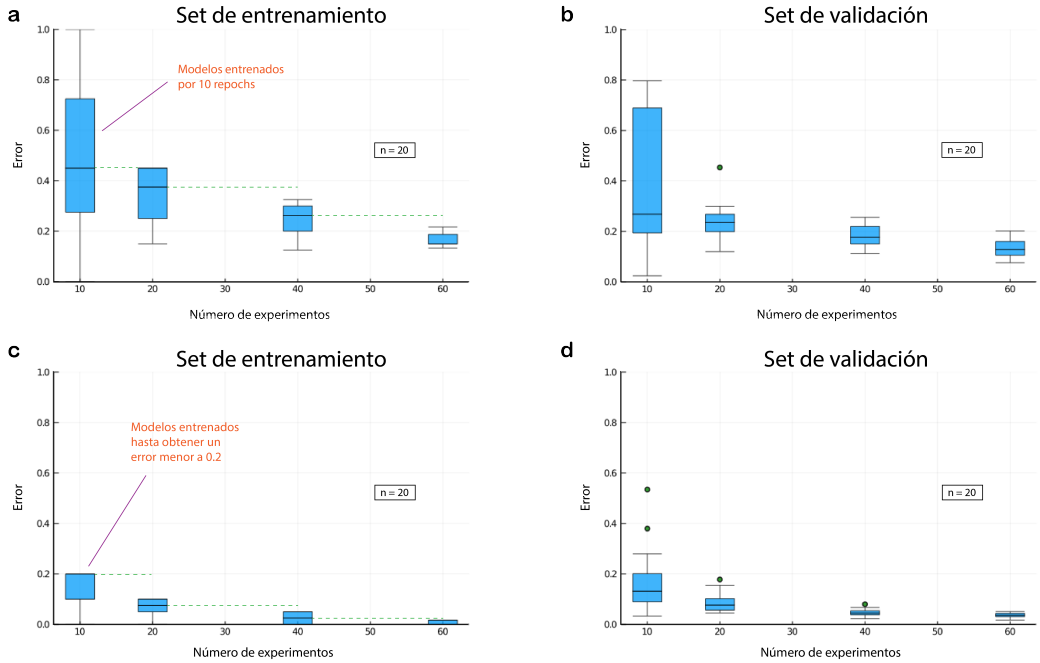

Figura 3.4 Determinación de repochs a partir del número de experimentos en el set de entrenamiento. (a) y (b) muestran los resultados de entrenar sobre el set de entrenamiento, fijando el número de repochs a 10 en el set de 10 experimentos. (c) y (d) contienen los resultados de entrenar sobre el set de entrenamiento hasta obtener un valor de error mínimo de 0.2. En ambos casos, los set de mayor número de experimentos fueron entrenados sin límite de épocas hasta obtener un error menor al error promedio del tamaño de set inmediatamente anterior.

3.3.3 Parámetros

Además del número de repochs, el modelo requiere de un conjunto de hiperparámetros necesarios para el proceso de entrenamiento. En la Tabla 3.1 se indican los utilizados para entrenar las ResNet ecológicas presentadas en este trabajo.

En la tabla se puede identificar que el learning rate tanto de reptile como de las épocas internas es inversamente proporcional a la época de entrenamiento, partiendo de un valor constante diferente para cada uno. Esto debido a que, al mantener el learning rate constante, el modelo no es capaz de llegar a un mínimo, sino que queda oscilando entre un cierto conjunto de valores de pérdida[5]

[5]Como se mencionó en la sección 2.1.3, el objetivo del optimizador es minimizar el error considerando el gradiente de la función de pérdida. Para lograrlo, el learning rate funciona como un parámetro que regula qué tanto se desea avanzar en dirección del gradiente. Es bien conocido el caso en que, si el learning rate es muy grande, el optimizador no es capaz de llegar a un mínimo global, sino que oscila alrededor de él (ellos, en caso de haber mínimos locales) [31]. Para impedir esto es que en este trabajo se toma el learning rate

En ambos casos se utilizó ADAM [25] como algoritmo de optimización dado que en pruebas preliminares se obtuvo un mejor rendimiento en comparación con otros. Este resultado coincide con otros estudios que muestran que ADAM es un algoritmo muy flexible de optimización [25, 31].

Hiperparámetro	Descripción	Valor
γ_r	Constante de proporcionalidad en épocas reptile	0.05
$\eta_{reptile}$	Ecuación del learning rate para épocas reptile (repoch)	$\frac{\gamma_r}{repoch}$
γ_e	Constante de proporcionalidad en épocas internas	0.01
η_{epochs}	Ecuación del learning rate para épocas internas (epoch)	$\frac{\gamma_e}{epoch}$
λ	Parámetro de regularización función de pérdida	1
epochs	Épocas de entrenamiento internas	100

Tabla 3.1 Hiperparámetros utilizados para entrenamiento de la ResNet ecológica.

También se encuentra el valor del parámetro de regularización para la función de pérdida en Ec. (3.6), siendo el parámetro que regula el objetivo de la red entre predecir las colecciones de especies correctamente y minimizar la distancia entre sus abundancias. Este valor se estableció como la unidad dado que se buscaba encontrar el rendimiento del modelo buscando ambos objetivos, de forma que ninguno de los dos tomara prioridad. Acompañado de estos, se encuentra el número de épocas internas de entrenamiento *epochs* que es el número de épocas que se entrena el modelo dentro de cada *repoch*, igualado a un valor fijo obtenido de forma heurística a través de diferentes pruebas, en las cuales se observa la convergencia hacia un valor de forma asintótica (ver Fig. 3.5).

3.3.4 Hardware y software

El entrenamiento de los modelos de la ResNet ecológica se llevó a cabo en una computadora con Procesador Intel®Core™i7-8750H CPU @ 2.20GHz, 2208 Mhz con 6 procesadores principales y 12 procesadores lógicos, 16GB de memoria DRAM DDR4 y Windows 10 de 64-bits como sistema operativo.

como un valor inversamente proporcional al número de épocas que se haya entrenado, de forma que conforme se vaya avanzando en el entrenamiento se disminuya el tamaño del paso en dirección del gradiente.

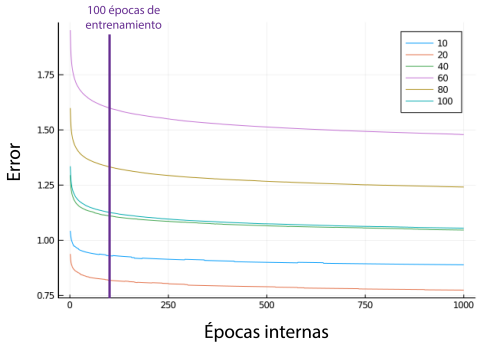

Figura 3.5 Error del modelo en función de la época de entrenamiento y el número de experimentos en el set de entrenamiento.

Se utilizó el lenguaje de programación Julia [5] en su versión 1.4.2 de 64-bits. La arquitectura de la ResNet ecológica fue construida haciendo uso de la versión 0.10.4 del paquete Flux [22] para Julia.

4 Validación de arquitectura

4.1 Limitaciones de arquitecturas existentes para aprender reglas de ensamblaje microbianas

El aprendizaje de reglas de ensamblaje microbianas es un tema poco explorado hasta el momento de la escritura de esta tesis, por tanto para poder evaluar el rendimiento de una arquitectura de deep learning para el aprendizaje de las reglas de ensamblaje de una comunidad de dos especies ($N = 2$) se implementó un modelo de DNN del tipo feedforward (ver Fig. 2.1) del cual posteriormente se evaluó su rendimiento.

La arquitectura de la DNN implementada consistió de cuatro capas ocultas del tipo afín ($f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$) donde $\mathbf{x}$ es el vector de entradas, $\mathbf{W}$ y $\mathbf{b}$ la matriz y el vector de parámetros a entrenar respectivamente para cada capa. Cada capa tiene dos neuronas, es decir que $\mathbf{W} \in \mathbb{R}^{N \times N}, \mathbf{b} \in \mathbb{R}^N$, con una función de activación del tipo *ReLU*[1] [32]. Como optimizador se ocupó ADAM [25] con un learning rate $\eta = 0,01$ y como función de pérdida se utilizó el error cuadrático medio, MSE (Mean Squared Error en inglés).

[1] La función ReLU (del inglés para rectified linear unit) está definida como $f(z) = max(0, z), z \in \mathbb{R}$. Esta función de activación lo que hace es convertir todos los valores negativos en cero, y todos los positivos (y el cero) los deja iguales. De esta forma podemos asegurarnos que no haya valores negativos en las salidas de nuestras capas.

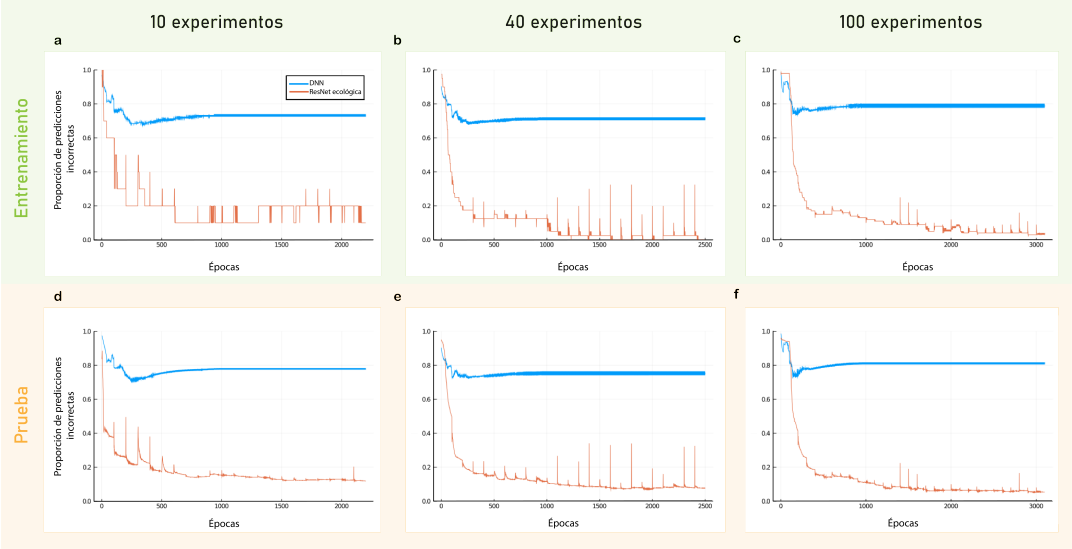

Figura 4.1 Comparativo entre el rendimiento para el aprendizaje de las reglas de ensamblaje de una comunidad de dos especies por una arquitectura de DNN y la arquitectura propuesta de la ResNet ecológica. Se puede apreciar que nuestra propuesta presenta saltos abruptos, estos corresponden con las actualizaciones realizadas por el algoritmo de reptile, el cual al reajustar los parámetros provoca una alta variación, misma que no se observa en el caso de la DNN a la que no se le aplicó reptile durante el entrenamiento.

Se simularon 1100 experimentos del modelo logístico de la Ec. (2.3) para dos especies[2]. Los parámetros del modelo fueron $\mathbf{A} = -\begin{bmatrix} -0.125 & -0.25 \\ -0.2 & -0.1 \end{bmatrix}$ y $\mathbf{r} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Estos datos se utilizaron para entrenar ambas arquitecturas y fueron divididos en un set de validación con 1000 experimentos y un set de entrenamiento de 100, del cual se formaron subconjuntos con 10, 40 y 100 experimentos cada uno, tomados al azar entre los 100 posibles.

El comparativo entre el rendimiento en el set de entrenamiento y en el de prueba se puede observar en la Fig. 4.1, donde se muestra la proporción de vectores de colección de especies predichos de forma correcta por ambos modelos para los diferentes tamaños de set de entrenamiento. En estas figuras se puede apreciar el comportamiento de ambas arquitecturas durante el

[2]Las condiciones iniciales de cada simulación fueron generadas de forma aleatoria con una distribución uniforme. Para poder determinar el rango de esta distribución se simuló un primer caso con condiciones aleatorias en el rango [0, 1] y se tomó del estado final la abundancia máxima entre ambas especies y se estableció como el valor máximo *valmax*. Se utilizó este valor para definir el valor máximo de las condiciones iniciales, de forma que el rango de la distribución de condiciones iniciales para cada especie fue [0, valmax). Esto se mantiene para todas las condiciones iniciales generadas en el presente trabajo.

proceso de entrenamiento, donde se puede observar que en todas las situaciones la arquitectura de la DNN no es capaz de predecir más del 25 % de los experimentos tanto en el set de entrenamiento como en el set de prueba de forma correcta; además de que es apreciable en todos los casos que de forma asintótica converge rápidamente a un desempeño constante con apenas la primera mitad del entrenamiento, indicando que ya no es capaz de mejorar sus predicciones a partir de los datos del set de entrenamiento. En comparación, nuestro modelo es capaz de predecir, al final del entrenamiento, con una precisión mayor al 90 % los vectores de colección de especies tanto del set de entrenamiento como del de prueba. Además, se aprecia que es capaz de seguir aprendiendo y mejorando su desempeño durante todo el proceso de entrenamiento, siendo hasta el final donde comienza a converger lentamente a un rendimiento estable.

La arquitectura de DNN entrenada fue diseñada con la intención de tener características similares a nuestra propuesta, tanto del número de capas, como de neuronas por capa. Igualmente, las funciones de activación utilizadas fueron tipo ReLU dado que no pueden existir especies con valores negativos, de forma que la red únicamente posea valores positivos. También el learning rate η coincide con el utilizado para entrenar nuestro modelo, así como el número de épocas de entrenamiento para cada arquitectura.

A partir de lo visto en la comparación de la Fig. 4.1 se propone que mientras mayor similitud exista entre el fenómeno y la arquitectura del algoritmo de deep learning que se va a entrenar, los resultados tienden a mejorar, tal y como ya se había encontrado en trabajos anteriores, principalmente en el aprendizaje de fenómenos biológicos complejos [6]. Partiendo de estos resultados, en las secciones siguientes presentamos la arquitectura de la ResNet ecológica, así como un conjunto de validaciones más amplia donde se muestra su desempeño ante diversas comunidades microbianas y sus diversos casos.

4.2 Poblaciones con dos especies

Las comunidades compuestas por dos especies representan uno de los casos base de las dinámicas poblacionales, en donde se pueden presentar únicamente

cuatro estados finales, mismos que se agrupan en dos casos: coexistencia y extinción como fue mostrado en la sección 2.2.3.

4.2.1 Caso de extinción

Consideremos el caso de extinción, donde al menos una de las dos especies va reduciendo su abundancia hasta quedar totalmente extinguida.

Para la obtención de datos, se llevaron a cabo simulaciones del modelo logístico descrito por la Ec. (2.3). En cada una de ellas se comenzó con un conjunto de condiciones iniciales uniformemente distribuidas en el rango (0, 12][3] para cada una de las especies representadas en la Fig. 4.2. Se añadieron a estos resultados 24 pares de condiciones iniciales ubicadas en los ejes, esto dado que se crearon 12 condiciones iniciales donde solo una de las especies tuviera un valor distribuido uniformemente en el rango ya mencionado, mientras que la otra tuviera condición inicial cero; esto para cada una de las especies.

Este conjunto de datos luce de forma similar a lo mostrado en la Tabla 4.1, en la cual se puede observar que para cada par de condiciones iniciales se obtiene un par de condiciones finales, las cuales corresponden al estado estable al que llega la población dada su dinámica. Es decir, se obtienen dos arreglos de dimensión $N \times M$ donde N es el número de especies (en este caso 2) y M es el número de experimentos que se simularon, uno de los arreglos corresponde a los valores de condiciones iniciales $\mathbf{x}(0)$ y el otro a estados finales $\mathbf{x}(t_f)$.

Para la simulación se utilizaron los parámetros siguientes:

$$r = [1, 1], k = [8, 10], a = [2, 2]$$

La dinámica del modelo logístico con estos parámetros es la que determina los estados de equilibrio estables a los que la dinámica poblacional converge, estos estados de equilibrio se pueden identificar a través de la visualización del campo vectorial del modelo mostrado en la Fig. 4.3a. Existen dos casos finales:

[3]Rango obtenido para el esquema de condiciones iniciales (0, *valmax*]. Donde *valmax* es el valor de la abundancia máxima obtenida en estado estable al realizar una simulación tal y como se describe a principios de este capítulo en [2].

$x_1(0)$	$x_2(0)$
4.6	8.9
6.3	1.5
11.2	1.3
2.4	7.5

$x_1(t_f)$	$x_2(t_f)$
0.0	10.0
8.0	0.0
8.0	0.0
0.0	10.0

Tabla 4.1 Formato de matrices a utilizar para entrenar a la red, estas son de dimensión $N \times M$ donde N es la cantidad de columnas (especies) y M la cantidad de experimentos (simulaciones). La tabla de la izquierda representa las condiciones iniciales de cada experimento, mientras que las del lado derecho corresponden a los estados finales obtenidos a partir de su simulación en el modelo logístico.

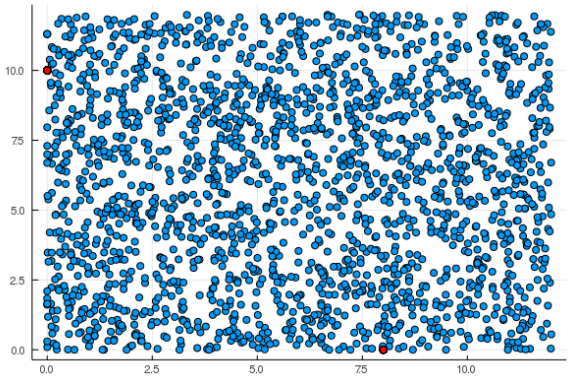

Figura 4.2 Distribución de condiciones iniciales de los datos de entrenamiento para el caso de extinción. Se muestran en color rojo los estados finales resultantes de la simulación, con los puntos azules como condiciones iniciales del modelo logístico.

en el que prevalece la primera especie y la segunda se extingue, o viceversa. En este mismo diagrama se puede apreciar la presencia de una separatriz, misma que cuenta con la característica de dividir el diagrama en dos secciones, dependiendo del lado de la separatriz en la que se encuentren las condiciones iniciales será el estado final al que lleve la dinámica.

El proceso de entrenamiento se llevó a cabo conforme a lo mencionado en el Capítulo 3. Dividiendo el dataset completo (1124 experimentos generados conforme a la descripción de los párrafos anteriores) en un set de prueba (1024 experimentos) y uno de entrenamiento (100 experimentos), este último para crear los siete *subsets* con diferente número de experimentos para entrenar (ver Fig. 3.2). El algoritmo de reptile fue utilizado para entrenar, aplicando la fórmula heurística Ec. (3.7) anteriormente discutida. Así como los parámetros mostrados en la Tabla 3.1.

Los resultados del rendimiento del modelo para cada *subset* del set de entrenamiento se muestra en la gráfica de cajas y bigotes ilustrada en la Fig. 4.3b donde se aprecian los resultados finales del modelo ya entrenado sobre su respectivo set de entrenamiento. En la gráfica se muestran los resultados

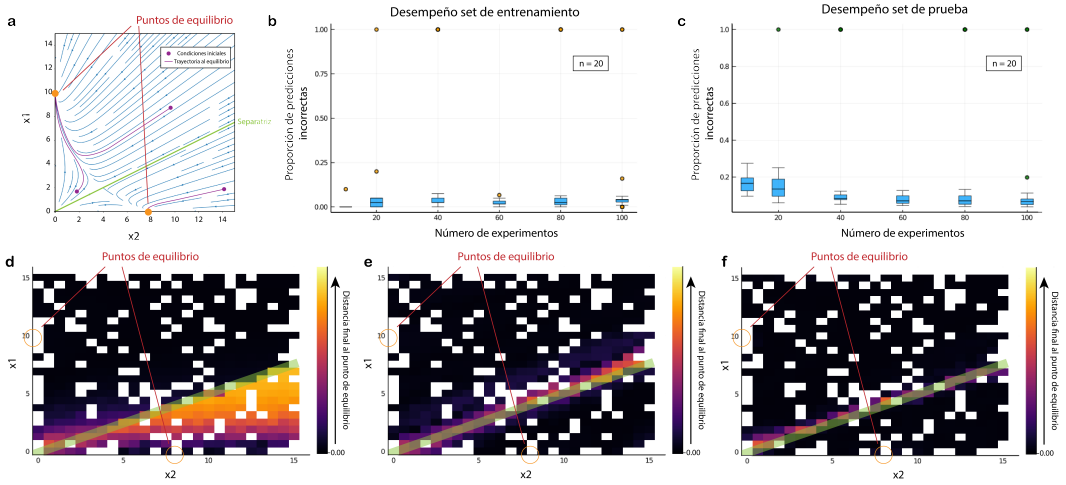

Figura 4.3 Caso de extinción. En (a) se muestra el campo vectorial correspondiente a la dinámica del modelo GLV con un conjunto de parámetros que producen un caso de extinción. Se indican ejemplos de trayectorias sobre el campo a partir de un conjunto de condiciones iniciales hasta el estado final en su respectivo punto de equilibrio, los cuales igualmente están indicados. Una línea verde representa la separatriz, misma que permite identificar de forma visual el punto de equilibrio al que llegará un conjunto de condiciones iniciales dependiendo de su ubicación. En (b) se presenta un gráfico de cajas y bigotes con la proporción de predicciones del vector de colección de especies incorrectas de la ResNet en el set de entrenamiento para diferentes tamaños del mismo. Para cada tamaño de set de entrenamiento se realizaron 20 entrenamientos independientes. Un gráfico similar se muestra en (c) para los resultados de las mismas ResNet sobre su respectivo set de prueba. Las figuras (d) a (f) muestran heatmaps donde se indica la distancia final entre la predicción de la ResNet ecológica y el valor real del estado final, donde el negro indica una diferencia nula y el blanco una ausencia de experimentos en esa región. Estos gráficos corresponden con los resultados sobre el set de prueba de un modelo entrenado con 10 (d), 40 (e) y 100 (f) experimentos. En todas se indica la ubicación de los puntos de equilibrio así como la separatriz equivalentes a los indicados en (a).

de 20 entrenamientos diferentes para cada tamaño de set de entrenamiento, mientras que los resultados mostrados en la Fig. 4.3c corresponden con los respectivos rendimientos de los modelos sobre el set de prueba.

En las Fig. 4.3d-f se muestran heatmaps de la distancia entre los estados finales simulados y los predichos por la red, siendo los ejes las abundancias absolutas de cada especie. Una explicación más detallada del razonamiento detrás de estas figuras se puede encontrar en el apéndice A2. En estas figuras, los colores más cercanos al amarillo corresponden a mayores distancias, mientras que los más cercanos al negro indican menor distancia, siendo el negro el cero; el color blanco hace referencia a la ausencia de experimentos a evaluar en esa zona. En ellas se muestran los resultados sobre el set de prueba de una ResNet

ecológica entrenada con 10, 40 y 100 experimentos respectivamente. También se indica en color verde la separatriz ilustrada en el campo vectorial de la Fig. 4.3a, así como la ubicación de los estados de equilibrio.

Estos resultados ilustran dos fenómenos, el primero de ellos es que la zona con puntos de mayor distancia final en todos los casos coincide con la posición de la separatriz, siendo que a mayor distancia de ella menor el error final de la ResNet ecológica. El segundo parte de un caso particular, el del set de entrenamiento con 10 experimentos (Fig. 4.3)d, donde se aprecia que la zona por debajo de la separatriz es notoriamente distinta a la presente en los casos con más experimentos. Esta observación es soportada por los resultados mostrados en la gráfica de cajas y bigotes y los resultados sobre el set de prueba (Fig. 4.3c), donde con 10 experimentos se encontró un peor desempeño comparado con el resto, aumentando en comparación con los de más de 40 experimentos. Comportamiento inverso al observado sobre el set de entrenamiento (Fig. 4.3b), en el cual el de mejor desempeño promedio es el de 10 experimentos.

Se puede apreciar que los resultados sobre los sets a partir de 40 experimentos ilustran un comportamiento muy similar, tanto en rendimiento en el set de entrenamiento y prueba, como en la forma de entrenar, denotado en los heatmap (Fig. 4.3e-f), donde la única diferencia notoria es la reducción del grueso de los puntos con mayor distancia sobre la separatriz, así como una disminución en los valores de sus distancias finales.

Estos gráficos ilustran también que, en promedio, la ResNet ecológica puede predecir correctamente más del 80 % de los casos sobre el set de prueba a partir de entrenarla con 10 experimentos, y con sets de entrenamiento con 40 experimentos o más, es capaz de predecir al menos el 90 % de los vectores de colección de especies finales del set de prueba correctamente (Fig. 4.3).

4.2.2 Caso de coexistencia

Ahora retomemos el caso de coexistencia para dos especies como se mencionó en la sección 2.2.3, donde ambas especies prevalecen.

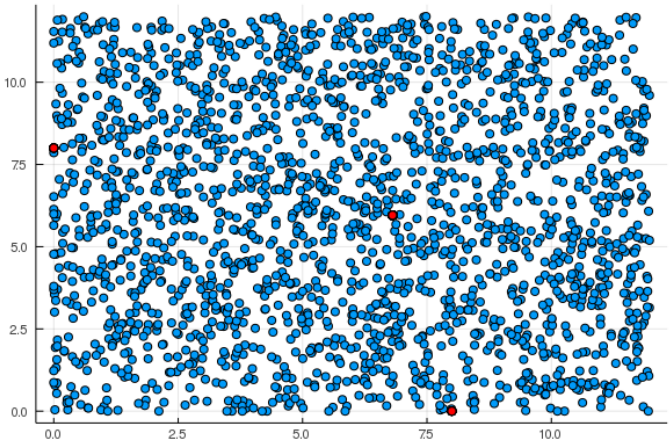

Figura 4.4 Distribución de condiciones iniciales (puntos azules) y los estados finales (puntos rojos) para los experimentos del caso de coexistencia.

$x_1(0)$	$x_2(0)$
3.5	2.7
0.0	1.5
11.2	0.0
2.4	7.5

$x_1(t_f)$	$x_2(t_f)$
6.0	7.0
0.0	8.0
8.0	0.0
6.0	7.0

Tabla 4.2 Representación de los arreglos de $N \times M$ de los datos sintéticos, a la izquierda las condiciones iniciales, a la derecha sus respectivos estados finales.

Los datos sintéticos fueron simulados a partir del modelo GLV, utilizando los parámetros siguientes conforme al modelo logístico Ec. (2.3):

$$r = [1, 1], k = [8, 8], a = [0.2, 0.3]$$

Fueron simulados 1100 experimentos con condiciones iniciales uniformemente distribuidas dentro del rango (0, 12] para cada especie, así como 12 casos a lo largo de cada eje, donde únicamente una especie posee abundancia inicial (ver Fig. 4.4). Estos casos sobre el eje llevan a los estados finales donde únicamente la especie con abundancia inicial prevalece, por tanto la importancia de evaluar el rendimiento del modelo sobre de ellos. Los arreglos de dimensión $N \times M$ con los datos sintéticos lucen de la forma presentada en la Tabla 4.2, donde se observan los tres estados finales mencionados, en los cuales cualquier condición inicial donde estén presentes las dos especies llega al estado final de coexistencia, mientras que si únicamente hay una especie inicial, prevalece.

Este fenómeno queda evidenciado de forma gráfica con el diagrama del campo vectorial de la dinámica con los parámetros de coexistencia ilustrado en la Fig.

4.5a, mismo en el que se indica con un punto naranja el estado de equilibrio de coexistencia y tres ejemplos de trayectorias sobre el campo vectorial para un conjunto de condiciones iniciales.

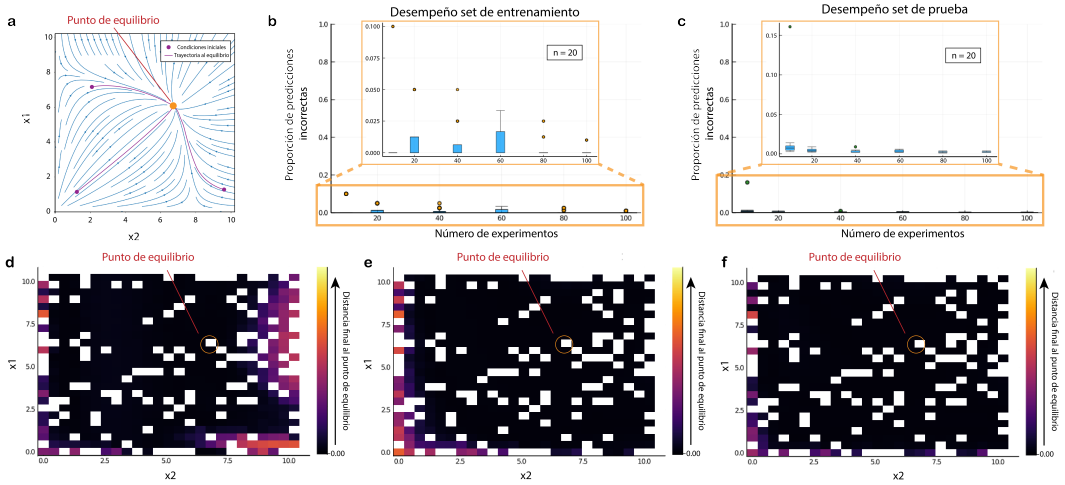

Figura 4.5 Caso de coexistencia. (a) Ilustra el campo vectorial correspondiente a la dinámica del modelo GLV con un conjunto de parámetros que producen un caso de coexistencia. Se indica el punto de equilibrio estable de coexistencia. (b) Muestra la proporción de predicciones del vector de colección de especies incorrectas de la ResNet en el set de entrenamiento para diferentes tamaños del mismo. Se ilustra el gráfico con un rango [0, 1] y sobre de él (dentro del recuadro amarillo) una ”ampliación” de los resultados con un rango menor. En (c) se muestra un gráfico similar para los resultados de las mismas ResNet sobre su respectivo set de prueba, de nueva cuenta con la imagen ”ampliada” en el recuadro amarillo. En las figuras (d) a (f) se ilustran los heatmaps donde se indica la distancia final entre la predicción de la ResNet ecológica y el valor real del estado final, donde el negro indica una diferencia nula y el blanco una ausencia de experimentos en esa región. Estos gráficos corresponden con los resultados sobre el set de prueba de un modelo entrenado con 10 (d), 40 (e) y 100 (f) experimentos. En ellos se indica la posición del estado de equilibrio de coexistencia marcado en (a).

Para el entrenamiento se empleó la misma estrategia de división del dataset ilustrada en la Fig. 3.2, así como los hiperparámetros de la Tabla 3.1. Como en el caso de extinción, se entrenaron 20 modelos de ResNet ecológica diferentes para cada tamaño de set. El rendimiento de los modelos sobre el set de entrenamiento se muestra en la Fig. 4.5b, la cual indica la fracción de vectores de colección de especies predichas de forma incorrecta por los modelos entrenados. En ella se muestra sobre del gráfico original (con rango [0, 1]) una “ampliación” de los mismos resultados pero sobre un rango menor ([0, 0.1]) en la cual se puede apreciar a mayor detalle las diferencias entre el número de experimentos utilizados para el entrenamiento. De forma similar, en la Fig. 4.5c se muestran los resultados de los modelos entrenados sobre el

set de prueba, nuevamente con la ”ampliación” para una mejor apreciación de los resultados.

En las Fig. 4.5d-f se muestran heatmaps que denotan el comportamiento de los modelos entrenados con 10, 40 y 100 experimentos respectivamente, sobre su set de prueba. En ellas se indica la ubicación del punto de equilibrio de coexistencia con un círculo color anaranjado. El color del recuadro ilustra la distancia final de los experimentos en él, siendo el color negro el cero y el blanco la ausencia de experimentos en esa zona.

4.3 Validación en comunidades microbianas empíricas

En la sección 4.2 se llevaron a cabo las validaciones sobre parámetros de comunidades microbianas sintéticas, es decir, asignados de forma manual. Posterior a esta validación, en este trabajo se buscó entrenar el modelo sobre parámetros de comunidades microbianas empíricas, es decir reales. Se utilizaron datos de ocho especies de microbios de suelo obtenidos en [14], ver Tabla 4.3.

Símbolo	Nombre científico
Ea	*Enterobacter aerogenes*
Pa	*Pseudomonas aurantiaca*
Pch	*Pseudomonas chlororaphis*
Pci	*Pseudomonas citronellolis*
Pf	*Pseudomonas fluorescens*
Pp	*Pseudomonas putida*
Pv	*Pseudomonas veronii*
Sm	*Serratia marcescens*

Tabla 4.3 Nombres y símbolos de las ocho especies de la comunidad microbiana empírica utilizada para nuestra validación.

La dinámica de esta población se ilustra en la Fig. 4.6. En ella, cada nodo (círculo) y símbolo corresponde con una especie de la comunidad (en la Tabla 4.3 se muestra la correspondencia entre el símbolo y el nombre de cada especie).

La Fig. 4.6a corresponde a la matriz de interacciones del modelo GLV mostrado en Ec. (2.2) que contiene los valores de las interacciones entre especies de forma pareada. Esta matriz es equivalente a una matriz de adyacencia de una red dirigida ponderada, donde $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{N \times N}$ y el elemento a_{ij} hace

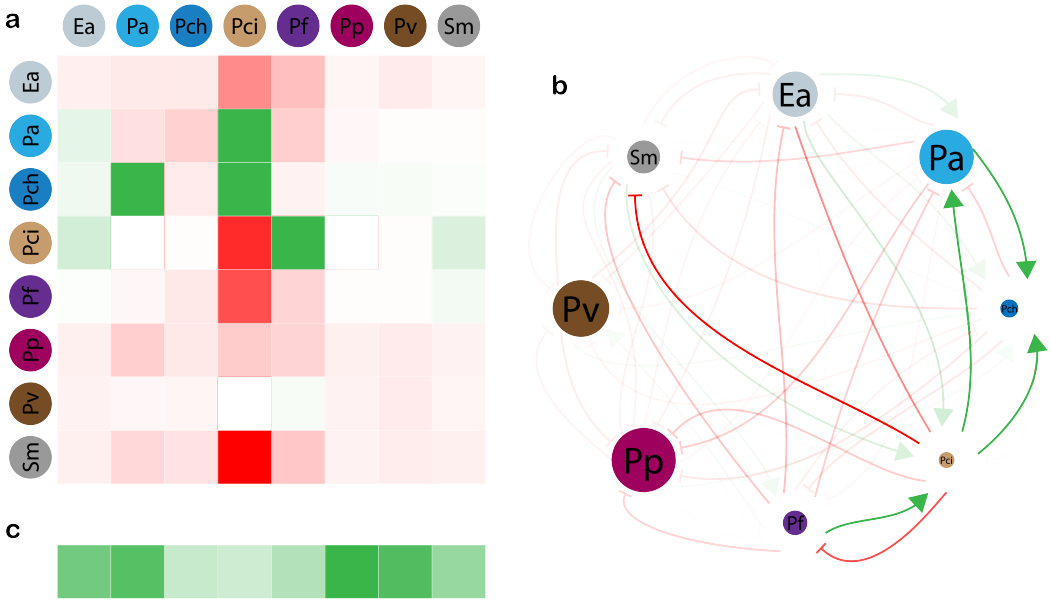

Figura 4.6 Dinámica poblacional de ocho especies microbianas de suelo obtenidas de [14]. En (a) se ilustra la matriz **A** que corresponde a los parámetros de interacción entre especies del modelo GLV. En ella el color verde hace referencia a la promoción (valores positivos) entre especies, y el rojo la inhibición (valores negativos), siendo la intensidad de la interacción consistente con la intensidad del color. A partir de (a), se puede ilustrar la red dirigida subyacente a la población la cual se muestra en (b), siguiendo la misma estrategia de la intensidad de colores con las interacciones. En (c) hay un renglón que hace referencia al vector **r** del modelo GLV, que es la tasa de reproducción para cada especie de las columnas en (a), este a su vez está representado por el tamaño de los nodos de cada especie en (b).

referencia a la interacción ejercida por la especie *j* sobre la especie *i*. Siguiendo la interpretación como matriz de adyacencia, es posible crear una red dirigida que ilustre las interacciones entre las especies (ver Fig.4.6b) siendo cada uno de los nodos de la red (círculos) una especie de la comunidad, y los vértices (flechas) las interacciones entre ellas. En esta red hay dos tipos de interacciones: promotoras e inhibitorias, las cuales indican el tipo de interacción entre cada par de especies (su nodo origen y su nodo destino), mientras que la tonalidad del color hace referencia a su intensidad. De forma similar a la matriz **A**, el vector $\mathbf{r} = (r_i) \in \mathbb{R}^N$ es el correspondiente al del modelo GLV en Ec. (2.2), donde r_i hace referencia a la tasa de crecimiento de la i-ésima especie.

4.3.1 Desempeño para predecir coexistencia y extinción en sistemas de dos especies

Se evaluó el rendimiento de la ResNet ecológica sobre la comunidad microbiana empírica de la Fig. 4.6. Se tomaron pares de especies y se simuló la dinámica al estado estable de cada uno con el modelo GLV y sus respectivos parámetros empíricos. Se obtuvieron 28 diferentes colecciones de especies. De ellos se obtuvieron 11 casos de coexistencia y 17 casos de extinción.

Caso de coexistencia

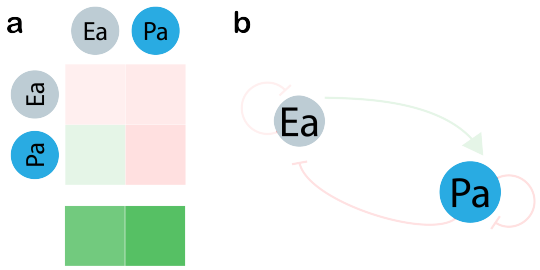

Figura 4.7 Parámetros de la dinámica entre dos especies (Ea y Pa) de la comunidad microbiana empírica para el caso de coexistencia. En (a) se ilustran los parámetros de interacción pareados en la matriz cuadrada superior, y el vector de crecimiento per capita por especie en la parte inferior. En (b) una ilustración de la red subyacente a los parámetros de interacción en (a).

En la Fig. 4.7 se muestra un ejemplo de un par de especies tomadas de la comunidad de ocho especies. En la Fig. 4.7a se muestra la matriz de adyacencia **A**, y en la parte inferior el vector **r** con las tasas de crecimiento de cada especie, y en la Fig. 4.7b la red subyacente derivada de estos parámetros. Se puede apreciar que para esta dinámica se ha denotado un *self-loop* para cada especie, esta interacción hace referencia a la diagonal de la matriz **A**, mientras que se ha representado la importancia del vector **r** con el tamaño del nodo de cada especie. Es importante recalcar que estos valores dependen de la abundancia de la especie, en particular que la dependencia sobre **r** es lineal ($x_i r_i$) y cuadrática para los de la diagonal ($x_i{}^2 a_{ii}$), por tanto la importancia de la diagonal aumenta conforme mayor sea la abundancia de la especie i-ésima. Esto resulta en que, para abundancias reducidas la población tiende a aumentar dado el crecimiento por **r**, mientras que para abundancias grandes la diagonal de **A** reduce la tasa de crecimiento poblacional, esto sucede de forma dinámica hasta llegar a un equilibrio.

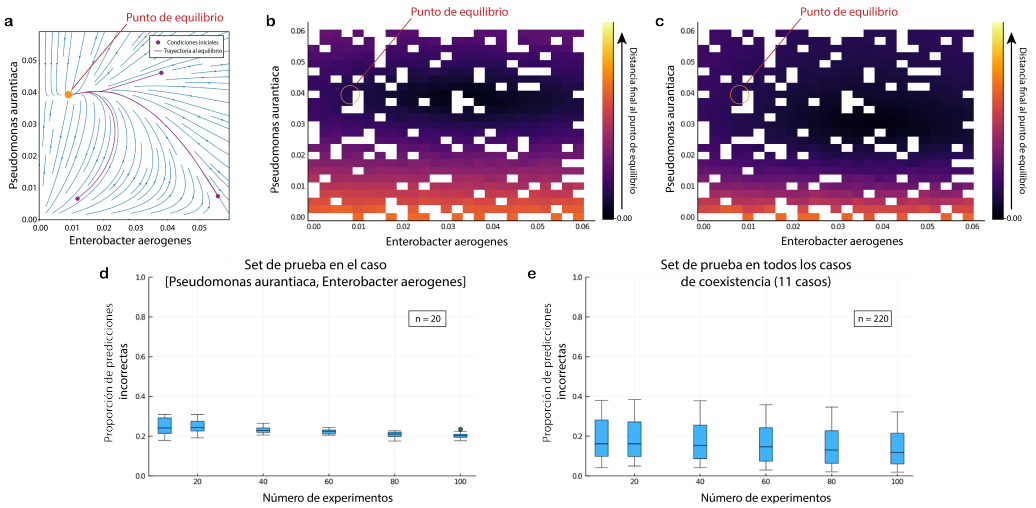

Figura 4.8 Resultados en el caso de coexistencia sobre los datos empíricos para una población con dos especies. En (a) se ilustra el campo vectorial de la dinámica entre *Enterobacter aerogenes* (Ea) y *Pseudomonas aurantiaca* (Pa). En (b) y (c) el resultado sobre el set de prueba de una ResNet ecológica entrenada con 10 experimentos y 100 experimentos respectivamente. (d) Muestra el rendimiento de las arquitecturas entrenadas en el caso de la colección de especies [Ea, Pa] sobre diferentes tamaños de set de entrenamiento, y (e) el rendimiento de todos los casos de coexistencia entre las diferentes combinaciones de dos especies de la comunidad microbiana en [14].

La dinámica de la comunidad con ambas especies se aprecia en la Fig. 4.8a en el campo vectorial resultante de los parámetros empíricos, en ella se puede apreciar que representa un caso de coexistencia, con el punto de equilibrio estable indicado con un círculo amarillo y tres posibles trayectorias ilustradas sobre la figura. Se entrenaron modelos de ResNet ecológica de forma equivalente a con los parámetros sintéticos de la sección 4.2. En las Fig. 4.8b-c se muestra el rendimiento de un entrenamiento con 10 experimentos (ver Fig. 4.8b) y otro con 100 experimentos (ver Fig. 4.8c) sobre el set de prueba. Se muestran también los resultados del rendimiento sobre el set de prueba de las 20 ResNet ecológicas entrenadas para cada tamaño de set de entrenamiento (ver Fig. 4.8d) en el caso ilustrado en la Fig. 4.7, así como el resultado general de entrenar cada uno de los 11 casos de coexistencia existentes entre las combinaciones de poblaciones de dos especies a partir de las ocho originales (ver Fig. 4.8e) donde para cada una de las combinaciones se entrenaron 20 modelos por cada tamaño de set de entrenamiento.

Caso de extinción

De forma análoga se presentaron los casos de extinción, uno de ellos es el correspondiente al vector de colección de especies [*Enterobacter aerogenes*, *Pseudomonas citronellolis*] que se ilustra en la Fig. 4.9. Donde la matriz de interacciones y el vector de tasa de crecimiento interespecífica se ilustran en la Fig. 4.9a en la parte superior e inferior respectivamente, mientras que la red subyacente a estos parámetros se muestra en la Fig. 4.9b, donde el tamaño las interacciones de la matriz **A** son ilustradas en la red, y el tamaño del nodo de cada especie hace referencia a la tasa de crecimiento dada por el vector **r**.

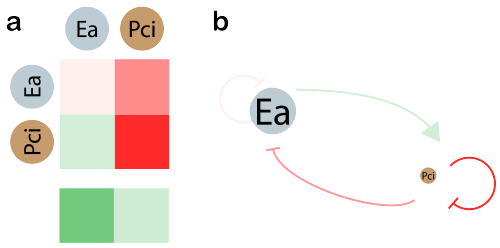

Figura 4.9 Parámetros de la dinámica entre dos especies (Ea y Pci) de la comunidad microbiana empírica para el caso de extinción. En (a) se ilustran los parámetros de interacción pareados en la matriz cuadrada superior, y el vector de crecimiento per capita por especie en la parte inferior. En (b) una ilustración de la red subyacente a los parámetros de interacción en (a).

A partir de estos parámetros, la dinámica resultante es la mostrada en la Fig. 4.10a, donde se puede apreciar que consiste en un caso de extinción, donde Ea extingue a Pci llegando al punto de equilibrio marcado. Los resultados sobre los set de prueba de los modelos entrenados con 10 y 100 experimentos se ilustran en las Fig. 4.10b-c respectivamente. De igual forma, el rendimiento de la ResNet ecológica sobre el vector de colección de especies [Ea, Pci] mostrado en Fig. 4.10d y el correspondiente al análisis del entrenamiento sobre todos los casos de extinción de las posibles combinaciones de dos especies (17 casos) evaluados sobre su set de prueba (ver Fig. 4.10e).

4.3.2 Desempeño en sistemas de más de dos especies

De forma similar a lo mostrado en la sección 4.3.1, se entrenaron modelos sobre comunidades con más de dos especies. En las Fig. 4.11a-f se pueden apreciar los resultados de entrenar modelos con 3, 4, 5, 6, 7, y las 8 especies de la comunidad respectivamente. En todos los casos excepto el de 8 especies se

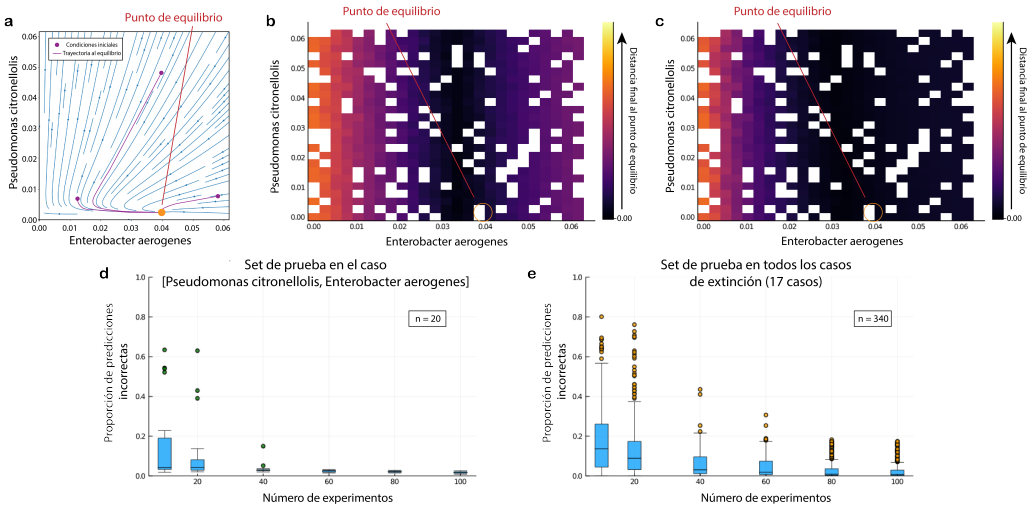

Figura 4.10 Resultados sobre el caso de extinción de los datos empíricos para una población con dos especies. En (a) se ilustra el campo vectorial de la dinámica entre *Enterobacter aerogenes* (Ea) y *Pseudomonas citronellolis* (Pci). En (b) y (c) el resultado sobre el set de prueba de una ResNet ecológica entrenada con 10 experimentos y 100 experimentos respectivamente. (d) muestra el rendimiento de las arquitecturas entrenadas en el caso de la colección de especies [Ea, Pci] sobre diferentes tamaños de set de entrenamiento, y (e) el rendimiento de todos los casos de extinción entre las diferentes combinaciones de dos especies de la comunidad microbiana en [14].

tomaron seis vectores de colección de especies diferentes al azar de entre todas las posibles combinaciones para el número de especies determinado. En el caso de 8 especies se muestran los resultados de las ResNet ecológicas entrenadas con los datos simulados con la comunidad completa y sus parámetros (ver Fig. 4.6). Los resultados corresponden al rendimiento sobre el set de prueba de las 20 arquitecturas entrenadas por cada colección de especies en sus diferentes tamaños de set de entrenamiento.

4.4 Discusión

La validación de la ResNet ecológica sobre datos *sintéticos* (ver sección 4.2) así como sobre los datos a partir de los parámetros de una comunidad empírica (ver secciones 4.3.1 y 4.3.2) muestran que la arquitectura es capaz de obtener resultados con una precisión mayor al 80 % sobre el set de prueba, considerando que en ambos casos el set de entrenamiento es considerablemente *chico* comparado con otros trabajos de deep learning aplicados en áreas de biología [6]. A pesar de ello, en ninguno de nuestros resultados se presentó una evidencia contundente de *overfitting*, dado a que en la mayoría de los

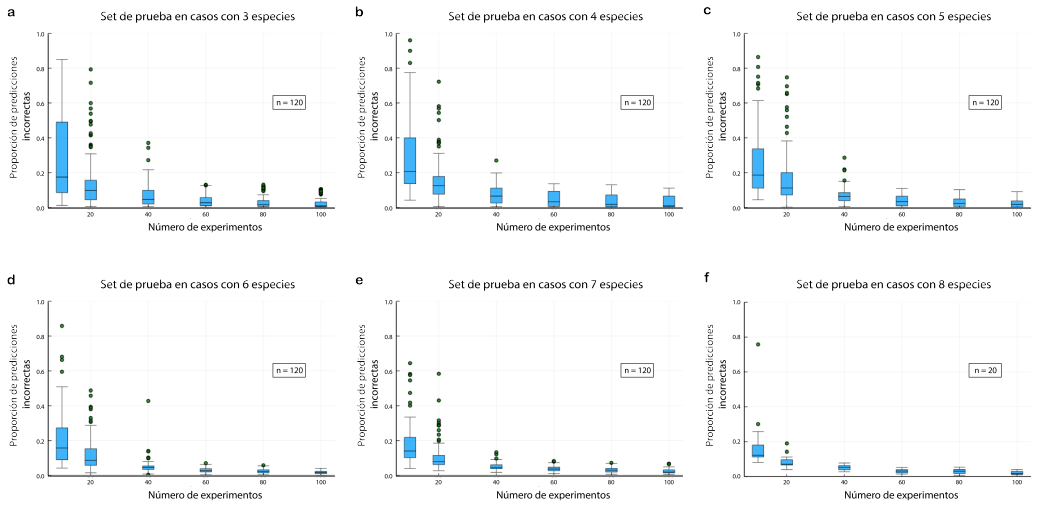

Figura 4.11 Rendimiento sobre el set de prueba de la ResNet ecológica con diferentes tamaños de vector de colección de especies. De (a) a (e) se presentan los resultados de 6 vectores de colecciones de especies diferentes para cada número de especies presentes (de 3 a 7 respectivamente). En (f) es el rendimiento de la ResNet ecológica sobre los datos de la comunidad completa.

entrenamientos el desempeño de la arquitectura fue mejor en el set de prueba que en el de entrenamiento.

Aunado a esto, se entrenaron 20 modelos con parámetros iniciales y experimentos diferentes para corroborar que los resultados correspondieran al desempeño general de la arquitectura, y no únicamente aplicables a un conjunto de parámetros iniciales o datos particulares. Este hecho puede surgir a partir de la implementación del algoritmo de meta-learning reptile, el cual permite a la arquitectura ajustarse al comportamiento general de los datos y no de forma particular, evitando fuertemente la presencia de overfitting y logrando que la red mantuviera la capacidad de predicción generalizada sobre el fenómeno y no de forma particular sobre los datos de entrenamiento. Un claro ejemplo de esto son los resultados sobre los datos sintéticos del caso de extinción (ver Fig. 4.3) donde con apenas 10 experimentos en el set de entrenamiento, la ResNet ecológica es capaz de identificar la ubicación de la separatriz. Este fenómeno se ve resaltado con los resultados con 40 y 100 experimentos en el set de entrenamiento, donde se nota de forma más clara que el ajuste de la red se hace cada vez más fino con respecto a esta zona de *decisión*.

La validación de la arquitectura de la ResNet ecológica presentada en este trabajo compara los resultados sobre comunidades simuladas con parámetros sintéticos así como obtenidos de una comunidad empírica. Los resultados muestran que la capacidad de aprendizaje de la red en ambos casos es comparable, de forma que es capaz de ajustarse a comunidades creadas *in silico* tanto como *in vitro*, considerando un mismo modelo para la dinámica poblacional (el modelo GLV). En ambos casos, la comparación de los campos vectoriales de la dinámica poblacional contra los resultados mostrados en los heatmaps muestran la capacidad de la red para ajustarse a la dinámica misma que generó los datos. Por ejemplo, en los casos de extinción, resulta notorio que las zonas de mayor *complejidad* para aprender (mayor distancia de la predicción al estado final) se ubica en las zonas de decisión; siendo para caso de datos sintéticos de la Fig. 4.3 la separatriz y para la comunidad empírica sobre el eje de las *Pseudomonas citronellolis* que van a ser extintas por las *Enterobacter aerogenes*. Este último fenómeno, donde la zona de mayor error está cerca del eje puede deberse a que estos valores donde únicamente Pci están presentes permiten que prevalezca su especie, ya que no hay Ea, de forma que mientras más cerca del valor crítico, mayor complejidad para la red de decidir si se trata del caso de extinción de las Ea o de Pci.

Un comportamiento muy similar se encuentra en el caso de coexistencia, donde nuevamente la sección de los ejes es la que la predicción cuenta con distancias finales más grandes con respecto al valor correcto. En especial, el caso de coexistencia en la comunidad sintética (ver Fig. 4.5) muestra este comportamiento, el cual se va reduciendo conforme más datos se otorgan en el entrenamiento.

Los resultados en la sección 4.3 muestran también un comportamiento poco esperado, que es el de que a mayor número de especies mejor rendimiento presenta el modelo. Este hecho parece ser algo sorprendente y poco plausible. Cabe resaltar que, al observar los resultados entre casos de coexistencia y extinción para dos especies, se aprecia una diferencia en rendimiento, siendo los de extinción mejores en comparación con los de coexistencia. Por tanto, es posible que dicha mejora en los resultados conforme aumenta el número de especies sea resultado de que la proporción de casos de coexistencia frente a los de extinción vaya disminuyendo conforme aumenta el número de especies, y por tanto los resultados generales de todos los posibles casos vayan

acercándose más al comportamiento presentado por únicamente los casos de extinción para dos especies.

En gran parte de los resultados del rendimiento de la ResNet ecológica sobre el set de prueba se puede encontrar una tendencia común: cualitativamente el modelo tiene resultados similares con sets de entrenamiento con 40 experimentos o más. Esta tendencia es mayormente notoria en los casos de extinción tanto con datos sintéticos (ver Fig. 4.3) como en los procedentes de la simulación de una comunidad empírica (ver Fig. 4.10). Esto se ve replicado también en los resultados con más de 2 especies (ver Fig. 4.11). Dichos resultados permiten argumentar que, con alrededor de 40 experimentos, la red es capaz de obtener un rendimiento similar al encontrado si se hubiera entrenado con 100 experimentos o más. Esta observación es de alta importancia ya que gran parte de los algoritmos de deep learning son conocidos por la necesidad de consumir una gran cantidad datos, así como recursos computacionales para lograr su entrenamiento, lo cual lleva mayormente a tiempos de entrenamiento prolongados y requerimientos de hardware elevados. Sin embargo, para lograr el entrenamiento de una ResNet ecológica sobre decenas de experimentos es suficiente contar con una PC comercial como la utilizada en nuestro trabajo (ver sección 3.3.4), en la cual son necesarios unos minutos para tener totalmente entrenado el modelo y listo para usarlo en validaciones.

Un área poco explorada en este trabajo fue la importancia de los parámetros, en particular del número de capas de la ResNet ecológica, así como del parámetro de regularización en la función de pérdida λ. Estos parámetros, como se planteó en la sección 3.1, tienen como objetivo aumentar la complejidad de la arquitectura en el caso del número de capas, y definir el objetivo de entrenamiento de la función de pérdida con el parámetro λ. Ambos parámetros fueron ajustados de forma heurística, y existe la posibilidad de que modificarlos permita obtener un mejor rendimiento por parte de la arquitectura, pero su exploración requeriría un trabajo exhaustivo para su validación, mismo que sale de los objetivos del presente proyecto. De forma que encontramos posible que al modificar la complejidad de las interacciones entre las especies de una comunidad microbiana, así como aumentar el número de especies presentes causen que modular y evaluar el papel de estos parámetros pueda resultar fundamental para lograr un alto desempeño por parte de la ResNet ecológica, pero nuevamente se deja esto para trabajos futuros.

5 Conclusiones y trabajo futuro

El uso de algoritmos de deep learning en fenómenos complejos ha sido ampliamente explorado recientemente, en particular en el área de la medicina y la biología con gran éxito [6, 29]. Sin embargo, encontramos que es fundamental diseñar arquitecturas que permitan aprender las características del fenómeno objetivo de forma más eficiente para obtener mejores resultados.

En este trabajo mostramos cómo nuestra propuesta de la ResNet ecológica es capaz de obtener un mejor rendimiento comparado con un modelo de DNN convencional con características similares. Para lograrlo, la arquitectura de la ResNet ecológica fue diseñada con alta similitud con el modelo al que se busca entrenar, así como dotarla con propiedades para cumplir las restricciones del modelo en su propio diseño. De forma paralela, la creación de una función de pérdida que sea capaz de revelar el objetivo de entrenamiento a la arquitectura fue predominante para el rendimiento de nuestra propuesta. Siendo la unión de los dos factores anteriores lo que mostró ser determinante para obtener resultados que no fueron logrados con modelos sin un diseño pensado en el fenómeno a entrenar.

Además del diseño de una arquitectura y una función de pérdida enfocado en el fenómeno para el cual fue entrenado el modelo, el uso de algoritmos de optimización y entrenamiento es clave durante el entrenamiento de modelos de deep learning. En este trabajo encontramos relevante el uso de algoritmos de entrenamiento del tipo *meta-learning*, en particular de *reptile*, mismo que

permitió dotar a la ResNet ecológica con la capacidad de generalizar sobre los datos de entrenamiento, evitando de esta forma el overfitting, el cual considerando el tamaño tan reducido de experimentos en el set de train (unas decenas de datos), resultaba un problema latente. De forma equivalente, para la ResNet ecológica se encontró que existe una forma heurística de encontrar el número de épocas de entrenamiento para este tipo de algoritmos de *meta-learning*, el cual depende del número de experimentos en el set de train.

También se mostró que el modelo de la ResNet ecológica, a través del entrenamiento descrito en este trabajo, es capaz de predecir con una precisión promedio mayor al 80 % el vector de colección de especies de comunidades microbianas tanto *in silico* como de comunidades empíricas, estas últimas con hasta 8 especies, mientras que un modelo de DNN logró una precisión menor al 25 %. Aunado a esto, el rendimiento de la ResNet ecológica mejora hasta un 90 % de precisión al entrenarla sobre un set de train de 40 experimentos, a partir de los cuales conforme se aumenta el set de train, los resultados finales obtienen cada vez menor variabilidad. Cabe destacar que a nuestros datos no les fueron agregados componentes de ruido, lo cual podría tomar importancia al momento de trabajar con datos experimentales. Se plantea abordar este punto a detalle en un trabajo posterior, donde se tome en cuenta los posibles errores experimentales para lo obtención de datos empíricos, y por tanto se puedan hacer validaciones con ellos y emular comportamientos similares en los datos *in silico*. También, para llevar a cabo validaciones con datos experimentales es necesario contar varios experimentos para la misma colección de especies. Por ejemplo, en los datos experimentales reportados por Friedman et al. [14], se reportan entre 3 y 5 experimentos con las mismas colecciones de especies. Este numero de experimentos es menor al requerido por nuestra arquitectura de acuerdo al análisis desarrollado en esta tesis (la arquitectura propuesta ocupaba al menos 10 experimentos). Concebimos dos posibles soluciones a este problema. La primera es obtener una mayor cantidad de experimentos para cada colección de especies. Avances experimentales recientes en tecnologías de co-cultivo automático de comunidades microbianas sugieren que este tipos de datos estará disponible en un futuro muy cercano [24]. La otra alternativa es realizar el entrenamiento y validación usando datos con experimentos que contienen distintas colecciones de especies. Esperamos que usar este tipo de datos permita entrenar la ResNet ecológica al menos tan bien que al utilizar datos con la misma colección de especies, pero es necesario realizar la validación numérica correspondiente. Notamos también

que este tipo de datos con experimentos de distintas colecciones de especies ya se encuentran disponibles actualmente [14]. En este sentido, esta segunda alternativa es muy prometedora para explorarse en trabajo futuro.

Finalmente, en este trabajo se validó el rendimiento de la ResNet ecológica haciendo uso de un conjunto de parámetros identificados para el propósito del mismo, sin embargo, tanto su arquitectura como la función de pérdida implementada poseen diferentes parámetros que permiten modular los diferentes objetivos, así como su complejidad. Parámetros como el número de capas de la arquitectura, la forma de la función $f_\theta(\mathbf{x})$ de cada capa y el parámetro λ de la función de pérdida fueron poco explorados, y se dejan como oportunidad para próximos desarrollos. Estos parámetros es posible que sean fundamentales para el entrenamiento sobre datos que presenten dinámicas más complejas, con interacciones tipo Holling II y III u otras formas funcionales [18, 20], las cuales no fueron exploradas, ya que se optó por comenzar las validaciones sobre las interacciones Holling tipo I, base del modelo GLV [18]. Otra situación donde estos parámetros podrían tener mayor relevancia es durante el entrenamiento con datos experimentales de comunidades microbianas, propuesta que se extiende para trabajos futuros donde se cuente con el número suficiente de experimentos para lograr llevar a cabo las validaciones de forma detallada tal y como se presentó en este trabajo con los parámetros de la comunidad empírica ocupada en [14], la cual tiene reportados únicamente dos datos por cada combinación de especies, insuficientes para entrenar cualquier modelo de deep learning.

Bibliografía

6

[1] Abdelhamid Ajbar y Mohamed Asif. „On the Existence of Complex Dynamics in Pure and Simple Microbial Competition in Bioreactors“. En: *Arabian Journal for Science and Engineering* 39.11 (nov. de 2014), págs. 7495-7501.

[2] Marco Tulio Angulo, Claude H Moog y Yang-Yu Liu. „A theoretical framework for controlling complex microbial communities“. En: *Nature communications* 10.1 (2019), págs. 1-12.

[3] Marco Tulio Angulo, Jaime A Moreno, Gabor Lippner, Albert-László Barabási y Yang-Yu Liu. „Fundamental limitations of network reconstruction from temporal data“. En: *Journal of the Royal Society Interface* 14.127 (2017), pág. 20160966.

[4] Y. Belkaid y O. J. Harrison. „Homeostatic Immunity and the Microbiota“. En: *Immunity* 46.4 (abr. de 2017), págs. 562-576.

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski y Viral B Shah. *Julia: A fresh approach to numerical computing*. 2017.

[6] Diogo M. Camacho, Katherine M. Collins, Rani K. Powers, James C. Costello y James J. Collins. „Next-Generation Machine Learning for Biological Networks“. En: *Cell* 173.7 (jun. de 2018), págs. 1581-1592.

[7] P. D. Cani, R. Bibiloni, C. Knauf, A. Waget, A. M. Neyrinck, N. M. Delzenne y R. Burcelin. „Changes in gut microbiota control metabolic endotoxemia-induced inflammation in high-fat diet-induced obesity and diabetes in mice“. En: *Diabetes* 57.6 (jun. de 2008), págs. 1470-1481.

[8] Ted J Case. „Illustrated guide to theoretical ecology“. En: *Ecology* 80.8 (1999), págs. 2848-2848.

[9] J. C. Clemente, L. K. Ursell, L. W. Parfrey y R. Knight. „The impact of the gut microbiota on human health: an integrative view“. En: *Cell* 148.6 (mar. de 2012), págs. 1258-1270.

[10] Jared M Diamond. „The island dilemma: lessons of modern biogeographic studies for the design of natural reserves“. En: *Biological conservation* 7.2 (1975), págs. 129-146.

[11] N. Dubilier, M. McFall-Ngai y L. Zhao. „Microbiology: Create a global microbiome effort“. En: *Nature* 526.7575 (oct. de 2015), págs. 631-634.

[12] Roger East. „Microbiome: Soil science comes to life“. En: *Nature* 501.7468 (sep. de 2013), S18-S19.

[13] *Ecological Assembly Rules: Perspectives, Advances, Retreats*. Cambridge University Press, 1999.

[14] J. Friedman, L. M. Higgins y J. Gore. „Community structure follows simple assembly rules in microbial microcosms“. En: *Nat Ecol Evol* 1.5 (mar. de 2017), pág. 109.

[15] A. Gagliardi, V. Totino, F. Cacciotti, V. Iebba, B. Neroni, G. Bonfiglio, M. Trancassini, C. Passariello, F. Pantanella y S. Schippa. „Rebuilding the Gut Microbiota Ecosystem“. En: *Int J Environ Res Public Health* 15.8 (ago. de 2018).

[16] A. Gaki, A. Theodorou, D. V. Vayenas y S. Pavlou. „Complex dynamics of microbial competition in the gradostat“. En: *J. Biotechnol.* 139.1 (ene. de 2009), págs. 38-46.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun. „Deep residual learning for image recognition“. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, págs. 770-778.

[18] C. S. Holling. „The Functional Response of Predators to Prey Density and its Role in Mimicry and Population Regulation“. En: *Memoirs of the Entomological Society of Canada* 97.S45 (1965), págs. 5-60.

[19] E. Y. Hsiao, S. W. McBride, S. Hsien y col. „Microbiota modulate behavioral and physiological abnormalities associated with neurodevelopmental disorders“. En: *Cell* 155.7 (dic. de 2013), págs. 1451-1463.

[20] Sze-Bi Hsu, Tzy-Wei Hwang y Yang Kuang. „Global dynamics of a predator-prey model with Hassell-Varley type functional response“. En: *Discrete Contin. Dyn. Syst. Ser. B* 10.4 (2008), págs. 857-871.

[21] H. K. Hughes, D. Rose y P. Ashwood. „The Gut Microbiota and Dysbiosis in Autism Spectrum Disorders“. En: *Curr Neurol Neurosci Rep* 18.11 (sep. de 2018), pág. 81.

[22] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal y Viral Shah. „Fashionable Modelling with Flux“. En: *CoRR* abs/1811.01457 (2018).

[23] J. Karczewski, B. Poniedzia?ek, Z. Adamski y P. Rzymski. „The effects of the microbiota on the host immune system“. En: *Autoimmunity* 47.8 (dic. de 2014), págs. 494-504.

[24] Jared Kehe, Anthony Kulesa, Anthony Ortiz, Cheri M. Ackerman, Sri Gowtham Thakku, Daniel Sellers, Seppe Kuehn, Jeff Gore, Jonathan Friedman y Paul C. Blainey. „Massively parallel screening of synthetic microbial communities“. En: *Proceedings of the National Academy of Sciences* 116.26 (2019), págs. 12804-12809. eprint: `https://www.pnas.org/content/116/26/12804.full.pdf`.

[25] Diederik Kingma y Jimmy Ba. „Adam: A Method for Stochastic Optimization“. En: *CoRR* abs/1412.6980 (2017). arXiv: `1412.6980`.

[26] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. „Deep learning“. En: *nature* 521.7553 (2015), págs. 436-444.

[27] Won-Jae Lee y Koji Hase. „Gut microbiota–generated metabolites in animal health and disease“. En: *Nature Chemical Biology* 10.6 (mayo de 2014), págs. 416-424.

[28] Alex Nichol, Joshua Achiam y John Schulman. „On First-Order Meta-Learning Algorithms“. En: *CoRR* abs/1803.02999 (2018). arXiv: `1803.02999`.

[29] Nariman Noorbakhsh-Sabet, Ramin Zand, Yanfei Zhang y Vida Abedi. „Artificial Intelligence Transforms the Future of Health Care“. En: *The American journal of medicine* 132.7 (jul. de 2019), págs. 795-801.

[30] S. Pushalkar, M. Hundeyin, D. Daley y col. „The Pancreatic Cancer Microbiome Promotes Oncogenesis by Induction of Innate and Adaptive Immune Suppression“. En: *Cancer Discov* 8.4 (abr. de 2018), págs. 403-416.

[31] Sebastian Ruder. „An overview of gradient descent optimization algorithms“. En: *CoRR* abs/1609.04747 (2016). arXiv: `1609.04747`.

[32] Ajay Shrestha y Ausif Mahmood. „Review of Deep Learning Algorithms and Architectures“. En: *IEEE Access* PP (abr. de 2019), págs. 1-1.

[33] Chunxu Song, Feng Zhu, Víctor J. Carrión y Viviane Cordovez. „Beyond Plant Microbiome Composition: Exploiting Microbial Functions and Plant Traits via Integrated Approaches“. En: *Frontiers in Bioengineering and Biotechnology* 8 (ago. de 2020).

[34] P. J. Turnbaugh, R. E. Ley, M. A. Mahowald, V. Magrini, E. R. Mardis y J. I. Gordon. „An obesity-associated gut microbiome with increased capacity for energy harvest“. En: *Nature* 444.7122 (dic. de 2006), págs. 1027-1031.

[35] N. Vallianou, T. Stratigou, G. S. Christodoulatos y M. Dalamaga. „Understanding the Role of the Gut Microbiome and Microbial Metabolites in Obesity and Obesity-Associated Metabolic Disorders: Current Evidence and Perspectives“. En: *Curr Obes Rep* 8.3 (sep. de 2019), págs. 317-332.

[36] John H Vandermeer. „The competitive structure of communities: an experimental approach with protozoa“. En: *Ecology* 50.3 (1969), págs. 362-371.

[37] Ophelia S Venturelli, Alex V Carr, Garth Fisher, Ryan H Hsu, Rebecca Lau, Benjamin P Bowen, Susan Hromada, Trent Northen y Adam P Arkin. „Deciphering microbial interactions in synthetic human gut microbiome communities“. En: *Molecular systems biology* 14.6 (2018), e8157.

[38] M. Y. Wei, S. Shi, C. Liang, Q. C. Meng, J. Hua, Y. Y. Zhang, J. Liu, B. Zhang, J. Xu y X. J. Yu. „The microbiota and microbiome in pancreatic cancer: more influential than expected“. En: *Mol. Cancer* 18.1 (mayo de 2019), pág. 97.

[39] L. Wen, R. E. Ley, P. Y. Volchkov y col. „Innate immunity and intestinal microbiota in the development of Type 1 diabetes“. En: *Nature* 455.7216 (oct. de 2008), págs. 1109-1113.

[40] Yandong Xiao, Marco Tulio Angulo, Jonathan Friedman, Matthew K Waldor, Scott T Weiss y Yang-Yu Liu. „Mapping the ecological networks of microbial communities“. En: *Nature communications* 8.1 (2017), págs. 1-12.

[41] H. J. Zapata y V. J. Quagliarello. „The microbiota and microbiome in aging: potential implications in health and age-related diseases“. En: *J Am Geriatr Soc* 63.4 (abr. de 2015), págs. 776-781.

[42] S. Zhu, Y. Jiang, K. Xu, M. Cui, W. Ye, G. Zhao, L. Jin y X. Chen. „The progress of gut microbiome research related to brain disorders“. En: *J Neuroinflammation* 17.1 (ene. de 2020), pág. 25.

Apéndice

A

A1 Ejemplo función de pérdida

Se muestra el pseudocódigo que corresponde con el algoritmo utilizado para calcular la función de pérdida $loss_{\varepsilon}$ sobre el conjunto de datos $\mathcal{D}$ el cual contiene los vectores de distribución de especies $\mathbf{x}_f$ de los datos para las M muestras y $\hat{\mathcal{D}}$ contiene los vectores $\hat{\mathbf{x}}_f$ que son el conjunto de predicciones generadas por el modelo.

Algoritmo A.1: Pseudocódigo $loss_{\epsilon}$

```
for ε_i in ε_min : ε_max do
    Convertir D y D̂ al vector de colección de especies con ε_i como
     threshold;
    for experimento in experimentos do
        Sumar el número de especies que fueron predichas de forma
         incorrecta;
    end
    Sacar el promedio de especies predichas de forma incorrecta de
     entre todos los experimentos con el parámetro ε_i;
end
Sacar el promedio de especies predichas incorrectamente para el
 barrido realizado sobre el parámetro ε desde ε_min hasta ε_max ;
```

En la Fig. A1.2 se muestra gráficamente los pasos que sigue el algoritmo sobre un conjunto de datos $\mathcal{D}$ y $\hat{\mathcal{D}}$ con dimensiones N = 2 y M = 3, lo que equivale a dos especies y tres muestras por cada especie. Además se puede observar que dichos datos corresponden con la representación mostrada en la Fig. A1.1 la

cual contiene la gráfica de los puntos en un plano, donde cada eje representa una de las especies.

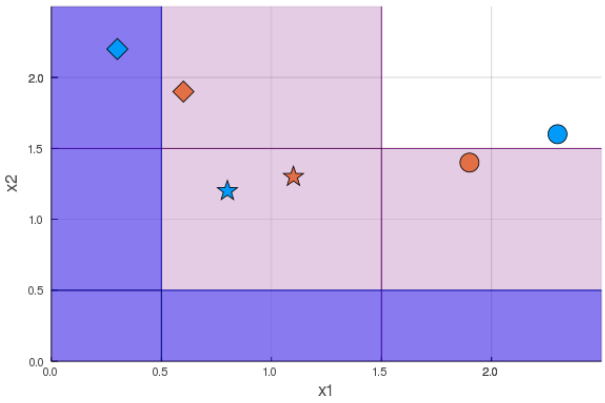

Figura A1.1 Representación del vector de distribuciones de las especies para cada una de las muestras de los datasets $\mathcal{D}$ (azul) y $\hat{\mathcal{D}}$ (rojo), siendo pares entre símbolos semejantes. Se muestran en forma de rectángulos las zonas donde el valor de la abundancia de la especie es menor al parámetro ε (0.5 para morado, 1.5 para rosa), por tanto se considera ausente en el vector de colección de especies.

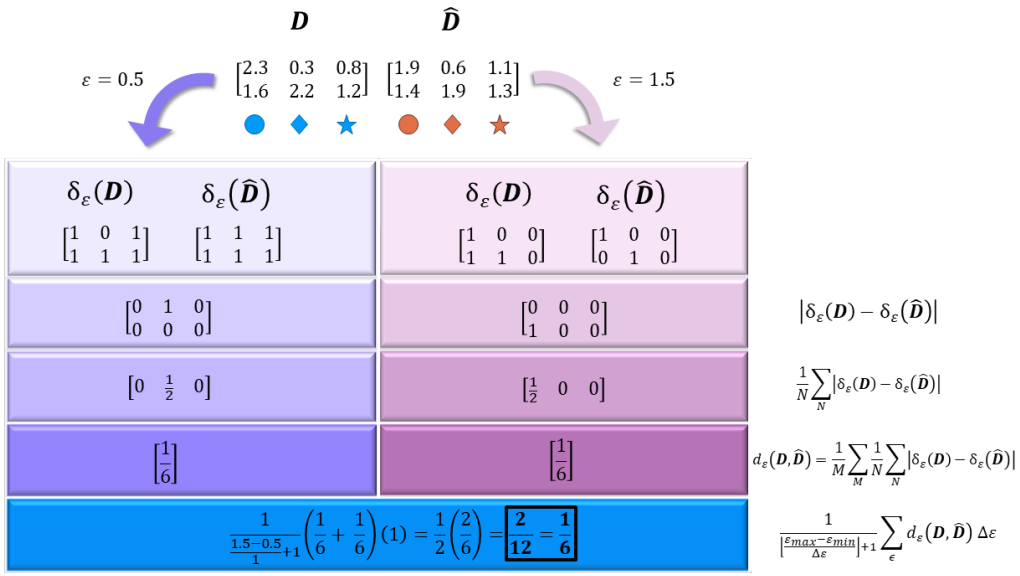

Figura A1.2 Procedimiento que lleva a cabo el algoritmo de la $loss_\varepsilon$ para calcular la función de pérdida entre el dataset $\mathcal{D}$ con las distribuciones finales de las N especies para los M experimentos y el dataset $\hat{\mathcal{D}}$ que es la predicción del modelo. En la parte inferior, en un recuadro color negro se señala el valor final de la función, el cual es un escalar y corresponde con el número total de predicciones incorrectas entre el número total de predicciones realizadas.

Una forma de interpretar la Fig. A1.1 es que, el valor ε delimita el plano donde se encuentran las especies, de forma que dibuja una línea recta sobre su valor, todo aquel valor que sea menor a esta línea será transformado en 0 (e.g. en el rombo azul, el valor de x_1 en esa muestra es menor a ε de forma que, al transformar esa muestra a su correspondiente colección de especies, se considerará como no presente, mientras que la componente de la especie x_2 al ser mayor a ε en el eje vertical, se considerará como

presente). De forma que se puede observar que todas aquellas observaciones ubicadas en el cuadrado inferior izquierdo formado por las limitaciones de ε tendrán el vector de colección de especies [0, 0], mientras que los rectángulos verticales y horizontales ubicados en los ejes tendrán los vectores [0, 1] y [1, 0] respectivamente y toda la sección restante del plano tendrá el vector [1, 1] que corresponderá a la zona donde ambas especies se consideran existentes. Esto igualmente se puede apreciar en la Fig. A1.2 cuando se aplica la función δ_ε al conjunto de datos.

De esta forma, el siguiente paso consiste en identificar si las predicciones están en la misma zona que los valores reales, lo cual consiste en restar ambos conjuntos de datos, dado que se trabaja con números binarios es posible obtener el número de predicciones incorrectas por muestra con el valor absoluto. Posteriormente se divide el total de predicciones incorrectas entre el número total de predicciones por muestra para conocer la fracción de estas con respecto al total.

Finalmente se hace un promedio entre las fracciones resultantes para conocer la fracción de incorrectas entre todas las muestras, para promediarse con respecto al intervalo en que se barrió el parámetro ε. El resultado refleja la fracción de predicciones que fueron hechas de forma incorrecta con respecto a las predicciones totales. Esta fracción se puede encontrar en el recuadro negro señalado en la parte inferior de la Fig. A1.2 la cual indica en este caso particular que el modelo predijo 2 de las 12 muestras de forma incorrecta. Este análisis se puede corroborar de forma visual con la Fig. A1.1 donde se aprecia que en dos ocasiones (una por cada valor de ε) que una de las especies queda de un lado de la zona marcada por ε y la otra del contrario, lo cual resulta en un valor opuesto en el vector de colección de especies y por tanto en una predicción incorrecta del modelo.

A2 Figuras de color falso (heatmaps)

Los heatmaps presentados para demostrar los resultados del entrenamiento del modelo representan la distancia final de los experimentos en una región de la representación de las abundancias absolutas. Para comprender la información que contienen podemos tomar el ejemplo de la Fig. A2.1.

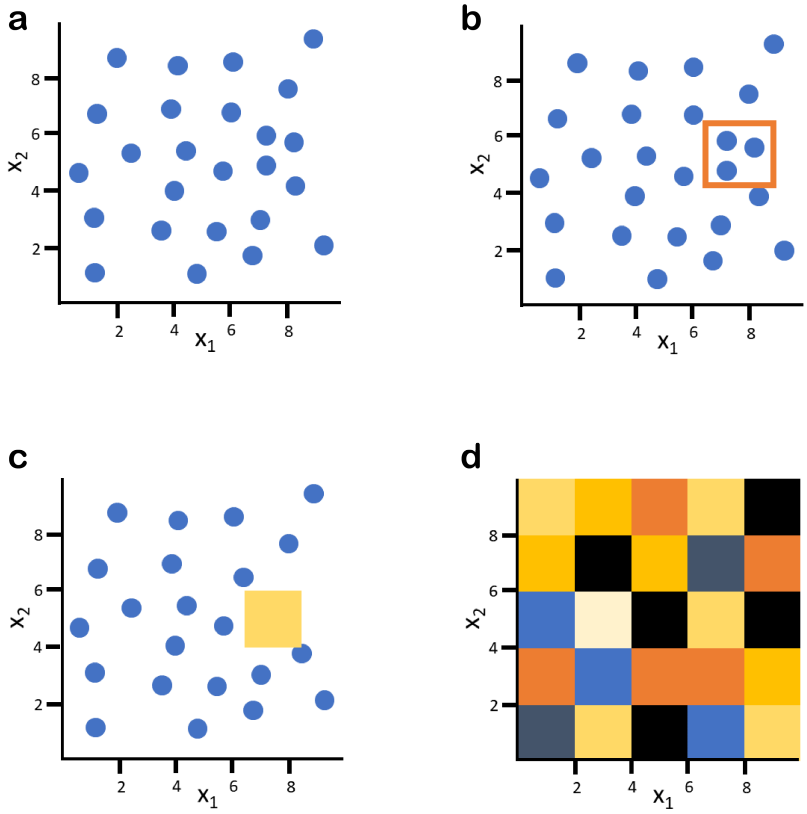

Figura A2.1 Método para evaluar y crear las gráficas de heatmap para medir el rendimiento del modelo sobre los datos. (a) Distribución de condiciones iniciales. (b) Selección de una región del heatmap a evaluar. (c) Coloreado de la zona seleccionada según la distancia final promedio de la predicción de los puntos contenidos. (d) Heatmap final al repetir el proceso en cada sección de las condiciones iniciales.

Se empieza con la distribución de condiciones iniciales (ver Fig. A2.1a). Posteriormente se crea una "cuadrícula ficticia" sobre esas condiciones iniciales. Para cada uno de esos recuadros se toman las muestras ahí contenidas (ver el ejemplo de la Fig. A2.1b donde se toma una región de 2×2 que contiene 3 muestras) y se evalúan usando el modelo entrenado. Una vez obtenidos los resultados del modelo, se evalúa la distancia final de cada punto. Se toma el promedio de todos los puntos contenidos y se interpola a una función que le da color según el valor (Fig. A2.1c). Se repite el proceso para cada uno de los

recuadros de la cuadrícula y se obtiene como resultado el heatmap de la Fig. A2.1d donde cada color hace referencia al promedio de las distancias finales de todos los puntos ahí contenidos.

Hay dos factores a considerar al utilizar esta técnica: el tamaño de la cuadrícula, y la posibilidad de espacios vacíos. Estas dos cuestiones están relacionadas entre ellas ya que a menor tamaño de cuadrícula, mayor probabilidad de que no haya un experimento en dicha zona.

Resulta sencillo identificar que diferentes tamaños de cuadrícula pueden dar una perspectiva diferente del fenómeno a evaluar, de forma que si se toma una tamaño de cuadro muy grande es posible que se pierdan detalles sutiles. De forma contraria, al tomar tamaños de cuadrícula muy pequeños, como ya se mencionó, la probabilidad de tener un espacio sin experimentos a evaluar aumenta, lo que lleva a tener una mayor aparición de espacios vacíos (o sin valores). Aunado a esto, se llega un punto en el que, sin importar qué tan pequeña sea la cuadrícula, la apreciación del fenómeno no cambia en lo absoluto. Recordando que estos diagramas de color falso son una herramienta cualitativa, para visualizar fenómenos de agrupaciones, más que evaluar de forma cuantitativa las diferencias; por lo que pasado un cierto umbral de resolución de cuadrícula, la información obtenida realmente será la misma sin importar qué tanto se reduzca su tamaño.

De forma que obtener buenos resultados con los heatmaps en estas situaciones es más una cuestión de prueba y error que de una formulación matemática, pero resultan una herramienta muy útil para visualizar de forma cualitativa fenómenos como agrupamientos o *clusters* en el conjunto de datos.