UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

GRADO DE INGENIERÍA EN INFORMÁTICA

Machine Learning para corrección de errores en datos de secuenciación de ADN

Autor
Amin Kasrou Aouam

Directores
Carlos Cano Gutiérrez

Mentores
María Soledad Benítez Cantos

Facultad de Educación, Tecnología y Economía de Ceuta

—

Ceuta, Julio de 2021

Resumen

Las nuevas técnicas de secuenciación de ADN (NGS) han revolucionado la investigación en genómica. Estas tecnologías se basan en la secuenciación de millones de fragmentos de ADN en paralelo, cuya reconstrucción se basa en técnicas de bioinformática. Aunque estas técnicas se apliquen de forma habitual, presentan tasas de error significantes que son detrimentales para el análisis de regiones con alto grado de polimorfismo. En este estudio se implementa un nuevo método computacional, locimend, basado en Deep Learning para la corrección de errores de secuenciación de ADN. Se aplica al análisis de la región determinante de complementariedad 3 (CDR3) del receptor de linfocitos T (TCR), generada in silico y posteriormente sometida a un simulador de secuenciación con el fin de producir errores de secuenciación. Empleando estos datos, entrenamos una red neuronal profunda (CNN) con el objetivo de generar un modelo computacional que permita la detección y corrección de los errores de secuenciación. Los resultados obtenidos demuestran que locimend es un modelo que identifica y corrige los patrones de errores de secuenciación de ADN, obteniendo una precisión de 0,89 y un área debajo de la curva (AUC) de 0,98. La implementación incluye una API REST que realiza la inferencia de la secuencia correcta de ADN, a partir de una secuencia de ADN con errores con el modelo pre-entrenado, con el objetivo de popularizar su uso en la comunidad científica.

Palabas clave: deep learning, corrección de errores, receptor de linfocitos T, secuenciación de ADN, inmunología

Abstract

Next generation sequencing (NGS) techniques have revolutionised genomic research. These technologies perform sequencing of millions of fragments of DNA in parallel, which are pieced together using bioinformatics analyses. Although these techniques are commonly applied, they have non-negligible error rates that are detrimental to the analysis of regions with a high degree of polimorphism. In this study we propose a novel computational method, locimend, based on a Deep Learning algorithm for DNA sequencing error correction. It is applied to the analysis of the complementarity determining region 3 (CDR3) of the T-cell receptor (TCR) found on the surface of lymphocytes, generated in silico and subsequently subjected to a sequencing simulator in order to produce sequencing errors. Using these data, we trained a depp neural network with the aim of generating a computational model that allows the detection and correction of sequencing errors. Our results show that locimend is a model that identifies and corrects DNA sequencing error patterns, obtaining an accuracy of 0,89 and an area under the curve (AUC) of 0,98. The implementation includes a REST API that performs the inference of the correct DNA sequence, from a DNA sequence with errors with the pre-trained model, in order to popularise its use in the scientific community.

Keywords: deep learning, error correction, DNA sequencing, T-cell receptor, immunology

Agradecimientos

Este proyecto no podría haber sido posible sin el apoyo de numerosas personas. En particular, quiero agradecer especialmente a Carlos Cano Gutiérrez por depositar su voto de confianza al asignarme un proyecto de investigación, el cual no era una propuesta de Trabajo de Fin de Grado. Y a María Soledad Benítez Cantos por su mentorización invaluable a lo largo de este trabajo. Su afán por el conocimiento, sus revisiones y comentarios de retroalimentación, su habilidad para exponer un concepto complejo en una frase y su dedicación incondicional al proyecto han sido el pilar central que ha permitido un desenlace favorable de la investigación.

Índice general

Índice de cuadros

Índice de figuras

1 Introducción

El ácido desoxirribonucleico (ADN) y el ácido ribonucleico (ARN) son los repositorios moleculares de la información genética. La estructura de cada proteína, y en última instancia de cada biomolécula y componente celular, es producto de la información programada en la secuencia de nucleótidos de una célula. La capacidad de almacenar y transmitir la información genética de una generación a otra es una condición fundamental para la vida. Un segmento de una molécula de ADN que contiene la información necesaria para la síntesis de un producto biológico funcional, ya sea una proteína o un ARN, se denomina gen. El almacenamiento y la transmisión de información biológica son las únicas funciones conocidas del ADN [1].

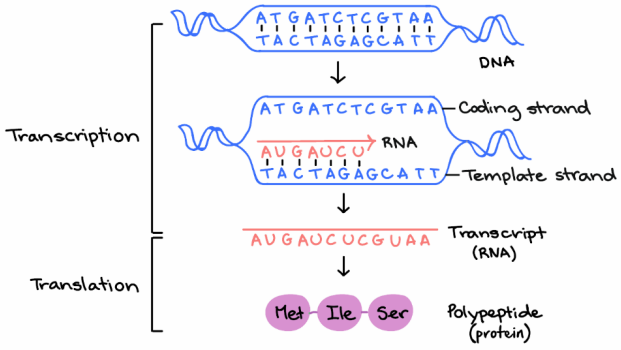

Figura 1.1: Dogma central de la biología molecular

Tanto el ADN como el ARN son ácidos nucleicos compuestos por unidades monoméricas llamadas nucleótidos, que a su vez contienen una base nitrogenada. El ADN está compuesto por las bases adenina (A), guanina (G) citosina (C) y timina (T), mientras que el ARN contiene uracilo (U) en lugar de timina (T) [2].

Hay muy pocos principios firmes en biología. A menudo se dice, de una forma u otra, que la única regla real es que no hay reglas, es decir, que se pueden encontrar excepciones a cada principio fundamental si se busca lo suficiente. El principio conocido como el "Dogma Central de la Biología Molecular"parece ser una excepción a esta regla de excepción ubicua [3]. El Dogma Central establece que, una vez que la información ha pasado a proteína, no puede volver a salir; i.e. la

transferencia de información de ácido nucleico a ácido nucleico, o de ácido nucleico a proteína puede ser posible, pero la transferencia de proteína a proteína, o de proteína a ácido nucleico es imposible [4].

Las proteínas se producen mediante el proceso de traducción, que tiene lugar en los ribosomas y está dirigido por el ARN mensajero (ARNm). El mensaje genético codificado en el ADN se transcribe primero en ARNm, y la secuencia de nucleótidos del ARNm se traduce en la secuencia de aminoácidos de la proteína. El ARNm que especifica la secuencia de aminoácidos de la proteína se lee en codones, que son conjuntos de tres nucleótidos que especifican aminoácidos individuales [2]. El código genético se muestra a continuación:

First Base	Second Base				Third Base
(5′)	U	C	A	G	(3′)
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	Stop	Stop	A
	Leu	Ser	Stop	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Figura 1.2: El código genético

Por lo tanto, si elucidamos la información contenida en el ADN, obtenemos información sobre las biomoléculas que realizan las diferentes tareas fisiológicas y metabólicas (e.g. ARN, proteínas).

1.1. Secuenciación de ADN

La secuenciación de ADN es el proceso mediante el cual se determina el orden de los nucleótidos en una secuencia de ADN. En los años 70, Sanger et al. desarrollaron métodos para secuenciar el ADN mediante técnicas de terminación de cadena [5]. Este avance revolucionó la biología, proporcionando las herramientas necesarias para descifrar genes, y posteriormente, genomas completos. La demanda creciente de un mayor rendimiento llevó a la automatización y paralelización de las tareas de secuenciación. Gracias a estos avances, la técnica de Sanger permitió determinar la primera secuencia del genoma humano en 2004 (Proyecto Genoma Humano). Cabe destacar que se estima que

la secuencia final cubría aproximadamente el 92 % del genoma debido a las limitaciones de la propia técnica de secuenciación y ensamblado [6].

Sin embargo, el Proyecto Genoma Humano requerió una gran cantidad de tiempo y recursos, y era evidente que se necesitaban tecnologías más rápidas, de mayor rendimiento y más baratas. Por esta razón, en el mismo año (2004) el *National Human Genome Research Institute* (NHGRI) puso en marcha un programa de financiación con el objetivo de reducir el coste de la secuenciación del genoma humano a 1000 dólares en diez años [7]. Esto estimuló el desarrollo y la comercialización de las tecnologías de secuenciación de alto rendimiento o *Next-Generation Sequencing* (NGS), en contraposición con el método automatizado de Sanger, que se considera una tecnología de primera generación.

1.2. Técnicas de secuenciación de alto rendimiento

Estos nuevos métodos de secuenciación proporcionan tres mejoras importantes: en primer lugar, no requieren la clonación bacteriana de los fragmentos de ADN, sino que se basan en la preparación de bibliotecas de moléculas en un sistema sin células. En segundo lugar, en lugar de cientos, se producen en paralelo de miles a muchos millones de reacciones de secuenciación. Finalmente, estos resultados de secuenciación se detectan directamente sin necesidad de una técnina experimental adicional llamada electroforesis [8].

Actualmente, se encuentran en desarrollo las tecnologías de tercera generación de secuenciación (*Third-Generation Sequencing*). Existe un debate considerable sobre la diferencia entre la segunda y tercera generación de secuenciación; la secuenciación en tiempo real y la divergencia simple con respecto a las tecnologías anteriores deberían ser las características definitorias de la tercera generación. En este trabajo se considera que las tecnologías de tercera generación son aquellas capaces de secuenciar moléculas individuales, negando el requisito de amplificación del ADN que comparten todas las tecnologías anteriores [9].

Estas nuevas técnicas han demostrado su valor con avances que han permitido secuenciar el genoma humano completo, incluyendo las secuencias repetitivas (de telómero a telómero) que no pudieron

dilucidarse en el Proyecto Genoma Humano. Combinando los aspectos complementarios de las tecnologías Oxford Nanopore y PacBio HiFi, 2111 nuevos genes, de los cuales 140 son codificantes, fueron descubiertos en el genoma humano en el año 2021 [10].

1.3. Limitaciones de los métodos paralelos

Aunque las tecnologías de secuenciación paralelas (NGS) han revolucionado el estudio de la variedad genómica entre especies y organismos individuales, la mayoría tiene una capacidad limitada para detectar mutaciones con baja frecuencia. Este tipo de análisis es esencial para detectar mutaciones en oncogenes (genes responsables de la transformación de una célula normal a maligna), pero se ve restringido por una tasa de errores de secuenciación no despreciables, tal y como ilustra la siguiente tabla:

Cuadro 1.1: Longitudes de lectura y tasas de error de secuenciación aproximados de diferentes tecnologías de secuenciación de alto rendimiento a mediados de 2014 [11]

Tecnología	Longitud de lectura (bp)	Tasa de error (%)
Sanger	~1,000	0.1–1
Illumina	2×125	≥0.1
SOLiD	35-50	>0.06
454	400	1
SMRT	~10,000	16
Ion Torrent	400	1

Para contrarrestar este obstáculo, varias técnicas mitigatorias se han puesto en marcha. Una de las más populares es el uso de una secuencia de consenso, que es un perfil estadístico a partir de un alineamiento múltiple de secuencias. Es una forma básica de descubrimiento de patrones, en la que un alineamiento múltiple de secuencias más amplio se resume en las características que se conservan. Este tipo de análisis permite determinar la probabilidad de cada base en cada posición de una secuencia [12].

Todas las técnicas de consenso monocatenarias reducen los errores en dos o tres órdenes de magnitud, lo que es mucho mayor que cual-

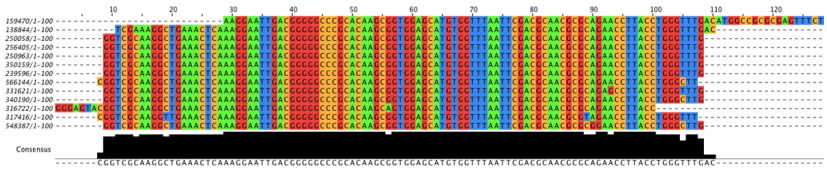

Figura 1.3: Alineamiento múltiple de secuencias. La secuencia de consenso aparece en la parte inferior y está formada por los nucleótidos con más ocurrencias en cada posición [13]

quier enfoque computacional o bioquímico anterior, y permiten identificar con precisión variantes raras por debajo del 0.1 % de abundancia. Sin embargo, persisten algunos errores. Los errores que se producen durante la primera ronda de amplificación pueden propagarse a todas las demás copias escapando la corrección [14].

Este problema se agrava en el análisis de repertorios inmunológicos, debido a nuestra limitada capacidad para distinguir entre la verdadera diversidad de los receptores de los linfocitos T (TCR) e inmunoglobulinas (IG) de los errores de PCR y secuenciación que son inherentes al análisis del repertorio. Las variantes resultantes de estos procesos biológicos pueden tener concentraciones drásticamente diferentes, lo que hace que aquellas con menor frecuencia sean indistinguibles de errores de secuenciación [15].

1.4. Variedad genética en el sistema inmunitario

La capacidad del sistema inmunitario adaptativo para responder a cualquiera de los numerosos antígenos extraños potenciales a los que puede estar expuesta una persona depende de los receptores altamente polimórficos expresados por las células B (inmunoglobulinas o anticuerpos) y las células T (receptores de células T [TCR]). La especificidad de las células T viene determinada principalmente por la secuencia de aminoácidos codificada en los bucles de la tercera región determinante de la complementariedad (CDR3) [16].

En el timo, durante el desarrollo de los linfocitos T, se selecciona al azar un segmento génico de cada familia (V, D y J) mediante un proceso conocido como recombinación somática o recombinación V(D)J, de modo que se eliminan del genoma del linfocito los no seleccionados y los segmentos V(D)J escogidos quedan contiguos, determinando la

secuencia de las subunidades del TCR y, por tanto, la especificidad de antígeno de la célula T. La selección aleatoria de segmentos junto con la introducción o pérdida de nucleótidos en sus uniones son los responsables directos de la variabilidad de TCR, cuya estimación es del orden de 10^{15} posibles especies distintas o clonotipos [17].

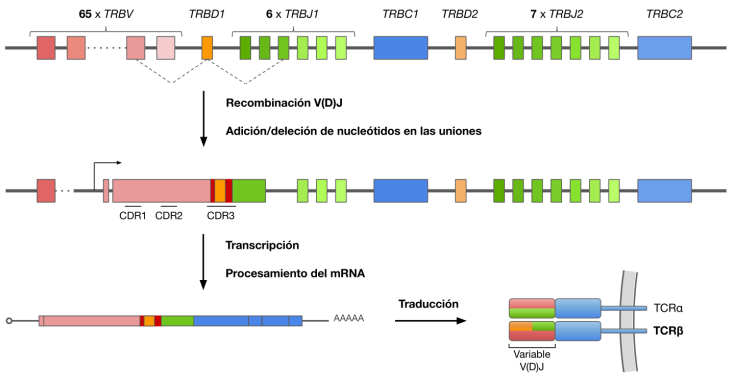

Figura 1.4: Generación de diversidad en el TCR α β. Durante el desarrollo de los linfocitos T se reordenan los segmentos génicos V (rosa), D (naranja) y J (verde) a través del proceso de recombinación V(D)J. Durante este proceso se pueden añadir o eliminar nucleótidos en la unión de los segmentos (rojo), contribuyendo a la diversidad de la secuencia. Se señalan las 3 regiones CDR, estando CDR3 localizada en la unión V(D)J [17]

Debido a la diversidad generada en esta recombinación aleatoria, las moléculas de anticuerpos y TCR muestran una enorme variabilidad de secuencias CDR3. De hecho, el número de secuencias de aminoácidos que están presentes en las regiones CDR3 de las moléculas de anticuerpos y TCR es mucho mayor que las que pueden ser codificadas por segmentos de genes de la línea germinal [18].

Frente a la evidencia recaudada, diversos métodos computacionales basados en la inteligencia artificial se han desarrollado para aliviar estos impedimentos e intentar discernir la verdadera variabilidad de secuencia de los errores de secuenciación inherentes a las tecnologías actuales.

1.5. Inteligencia artificial

La inteligencia artificial (IA) es uno de los campos más nuevos de la ciencia y la ingeniería. La investigación empezó después de la Segunda Guerra Mundial, y el término se acuñó en 1956, en la conferencia de Dartmouth College. La definición de inteligencia artificial sigue generando debate a día de hoy. Por ende acotaremos la definición de inteligencia artificial al estudio de los agentes inteligentes [19].

Un agente es cualquier elemento capaz de percibir su entorno mediante sensores y actuar en consecuencia en ese entorno mediante actuadores. Un agente inteligente es aquel que actúa para conseguir el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado. En términos matemáticos, formulamos que el comportamiento de un agente se describe por la función de agente que asigna a cualquier entrada una acción [19].

1.5.1. Historia inicial de la inteligencia artificial

Los comienzos de la inteligencia artificial (1956-1969) se caracterizan por un entusiasmo y optimismo generalizado. Los rápidos avances en *hardware*, junto con el desarrollo de sistemas que redefinían las posibilidades de los ordenadores excedían las expectativas de los especialistas [19]. La atmósfera frenética, durante este periodo, se puede recapitular a partir de de la propuesta de la conferencia de Dartmouth:

> El estudio se basa en la conjetura de que cada aspecto del aprendizaje o cualquier otro rasgo de la inteligencia puede ser, en principio, precisamente descifrado. Se intentará encontrar un método para hacer que las máquinas utilicen el lenguaje, formen abstracciones y conceptos, resuelvan problemas que ahora están reservados a los humanos, y que se mejoren a sí mismas [20].

A finales de los años 1960, los investigadores se toparon con numerosos obstáculos al tratar de usar sus sistemas para resolver problemas más complejos. Debido a una carencia de resultados decisivos, la financiación en este ámbito académico fue cancelada [19].

Los sistemas de conocimiento (década de 1970) aprovecharon conocimiento específico de una rama para escalar los sistemas inteligentes,

intercambiando sistemas generales con bajo rendimiento por sistemas específicos al problema con mejor rendimiento. Estos sistemas novedosos se empezaron a comercializar en los años 80, creando la industria de la IA [19].

1.6. Redes neuronales artificiales

Una red neuronal artificial es un modelo de computación bioinspirado, formado por capas de neuronas artificiales. Comenzaremos definiendo el concepto de neurona artificial, con el fin de introducir la noción de red neuronal artificial de forma clara y concisa.

Una neurona artificial es un modelo de una neurona biológica, donde cada neurona recibe un conjunto de señales y, al dispararse, transmite una señal a las neuronas interconectadas. Las entradas (*inputs*) se inhiben o amplifican mediante unos pesos numéricos asociados a cada conexión. El disparo, i.e. activación, se controla a través de la función de activación. La neurona recoge todas las señales entrantes y calcula una señal de entrada neta en función de los pesos respectivos. La señal de entrada neta sirve de entrada a la función de activación que calcula la señal de salida [21].

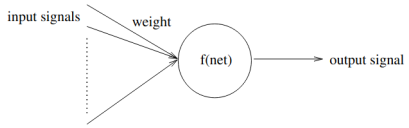

Figura 1.5: Diagrama de una neurona artificial [21]

El proceso de activación se puede expresar como un modelo matemático:

$$y = f\left(\sum_{i=0}^{n} w_i x_i - T\right) \tag{1.1}$$

donde y es la salida del nodo, f es la función de activación, w_i es el peso de la entrada x_i , y T es el valor del umbral [22].

Una red neuronal artificial (ANN, por sus siglas en inglés) es una red de capas de neuronas artificiales. Una ANN está formada por una capa de entrada, capas ocultas y una capa de salida. Las neuronas de una

capa están conectadas, total o parcialmente, a las neuronas de la capa siguiente. También son posibles las conexiones de retroalimentación con las capas anteriores [21]. La estructura típica de una ANN es la siguiente:

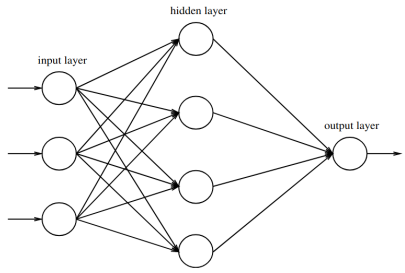

Figura 1.6: Estructura de una red neuronal artificial [21]

Los principios básicos de las redes neuronales artificiales fueron formulados por primera vez en 1943, y el perceptrón, que históricamente es posiblemente la primera neurona artificial, se propuso en 1958 [23]. Sin embargo, estos modelos no fueron populares hasta mediados de la década de 1980, cuando se reinventó el algoritmo de *back-propagation* [19].

1.6.1. Back propagation

El algoritmo de *back propagation* permite establecer los pesos y, por tanto, entrenar los perceptrones multicapa. Esto abrió el camino para el uso de las redes neuronales multicapa. La disponibilidad de un método riguroso para fijar los pesos intermedios, i.e. para entrenar las capas ocultas, impulsó el desarrollo de las ANN, superando las deficiencias de la capa única propuesta por Minsky [23]. Los autores de la publicación original que describió este algoritmo sintetizan su funcionamiento a alto nivel:

> Este algoritmo ajusta iterativamente los pesos de cada conexión, con el objetivo de minimizar la función de pérdida (i.e. obtener la menor diferencia entre la salida de la ANN y el valor esperado). Como resultado de ello, las capas ocultas llegan a representar características importantes del dominio del problema, y las regularidades de

la tarea son captadas por las interacciones de estas unidades. La capacidad de crear nuevas características útiles distingue a la retropropagación de otros métodos anteriores más sencillos, como el procedimiento de convergencia del perceptrón [24].

Asimismo el algoritmo de *back propagation* es un enfoque para calcular los gradientes de forma algorítmica, que es especialmente eficaz para los modelos que emplean redes neuronales multicapa [25]. Procedemos a la exposición del algoritmo de optimización más usado con la *back propagation*, el *gradient descent*.

1.6.2. Gradient descent

El *gradient descent* es una forma de minimizar una función objetivo parametrizada por las variables de un modelo (e.g. pesos de una neurona artificial) mediante la actualización de los éstas en la dirección opuesta al gradiente de la función objetivo con respecto a los parámetros, i.e. siguiendo la dirección de la pendiente de la superficie creada por la función objetivo hacia abajo hasta llegar a un mínimo local o global [26].

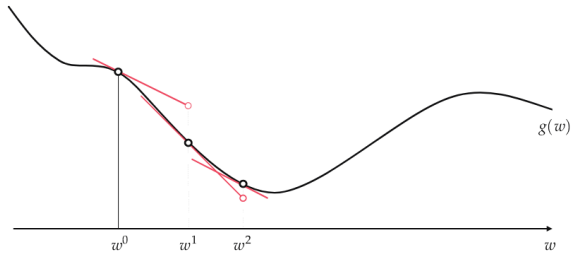

Figura 1.7: Diagrama del algoritmo de *gradient descent*. Comenzando en el punto inicial w^0 hacemos nuestra primera aproximación a $g(w)$ en el punto $(w^0, g(w^0))$ en la función (mostrada como un punto negro hueco) con la aproximación en serie de Taylor de primer orden dibujada en rojo. Moviéndonos en la dirección de descenso del gradiente negativo proporcionada por esta aproximación llegamos a un punto $w^1 = w^0 - \alpha \frac{d}{dw} g(w^0)$. A continuación, repetimos este proceso en w^1, moviéndonos en la dirección del gradiente negativo allí, y así sucesivamente. [25]

Esta secuencia de pasos se puede formular, de forma matemática:

$$w^k = w^{k-1} - \alpha \nabla g(w^{k-1}) \tag{1.2}$$

donde $-\nabla g(w)$ es el gradiente negativo de una función $g(w)$ en un punto [25].

Este proceso, junto con sus variantes, es el algoritmo de optimización local más utilizado en *machine learning* actualmente. Esto se debe en gran medida al hecho de que la dirección de descenso proporcionada a través del gradiente es más fácil de computar (particularmente cuando la dimensión del *input* aumenta) [25].

1.7. Hacia el Deep Learning

El *Deep Learning* es un subconjunto del *machine learning*, en el cual se utiliza como modelo de computación las redes neuronales artificiales (ANN) con múltiples capas ocultas. Este enfoque trata de resolver el problema de una buena capacidad expresiva de las redes neuronales, sin el aumento exponencial del número de neuronas [27].

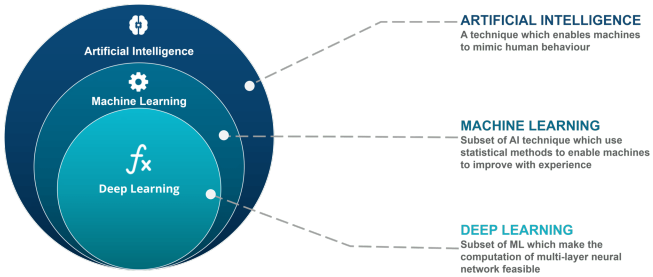

Figura 1.8: Diferencia entre inteligencia artificial, *machine learning* y *deep learning*

La cuestión sobre cómo mantener esta capacidad expresiva de la red y al mismo tiempo reducir el número de unidades de cálculo se ha planteado durante años. Intuitivamente, se sugirió que es natural buscar redes más profundas porque 1) el sistema neuronal humano es una arquitectura profunda y 2) los humanos tienden a representar conceptos en un nivel de abstracción como la composición de conceptos en niveles inferiores. Actualmente, la solución es construir arquitecturas más profundas, i.e. redes neuronales con más capas ocultas [27].

En la actualidad, los avances tanto en potencia de cálculo del *hardware*, especialmente en las tarjetas gráficas (GPU) [28], junto con la disponibilidad de grandes *datasets* [19] han permitido que el aprendizaje profundo prospere y mejore drásticamente el estado del arte en muchas tareas de inteligencia artificial, como la detección de objetos, el reconocimiento del habla o la traducción automática. Su naturaleza de arquitectura profunda otorga a esta disciplina la posibilidad de resolver muchas tareas de IA más complicadas [27]. Como resultado, los investigadores están extendiendo el aprendizaje profundo a una variedad de diferentes dominios y tareas modernas:

- Eliminación del ruido en las señales de voz
- Descubrimiento de patrones de agrupación de expresiones genéticas
- Generación de imágenes con distintos estilos
- Análisis de sentimientos de diferentes fuentes simultáneas

1.8. Bioinformática

La bioinformática es un campo interdisciplinar en el que intervienen principalmente las áreas de biología molecular, genética, informática, matemáticas y estadística. Esta disciplina permite que los estudios biológicos a gran escala y con gran cantidad de datos se aborden desde un punto de vista informático. Los problemas más comunes son el modelado de procesos biológicos a nivel molecular y la realización de inferencias a partir de los datos recogidos, [29] en particular:

Cuadro 1.2: Problemas clásicos de la bioinformática

Problema	Fuente de datos
Alineamiento de secuencias múltiples	ADN
Búsqueda de patrones de secuencias	ADN
Análisis evolutivos	ADN
Análisis de la función	Proteínas
Predicción de las estructuras	Proteínas
Estudio de las vías metabólicas	Redes biológicas

Es tentador atribuir los orígenes de la bioinformática a la reciente convergencia de la secuenciación del ADN, los proyectos genómicos a gran escala, Internet y los superordenadores. Sin embargo, algunos científicos que afirman que la bioinformática se encuentra en su infancia reconocen que los ordenadores eran herramientas importantes en la biología molecular una década antes de que la secuenciación del ADN se convirtiera en algo factible (década de los 1960) [30].

En los ámbitos de la investigación biológica y médica, la labor del análisis computacional ha aumentado de forma espectacular. La primera ola de esta disciplina se centró en el análisis de secuencias, en la cual aún quedan muchos problemas importantes sin resolver, las necesidades actuales y futuras se centran en la integración sofisticada de conjuntos de datos extremadamente diversos. Estos nuevos tipos de datos proceden de una variedad de técnicas experimentales, muchas de las cuales son capaces de producir datos a nivel de células enteras, órganos, organismos o incluso poblaciones. La principal fuerza impulsora de estos cambios ha sido la llegada de nuevas y eficientes técnicas experimentales, principalmente la secuenciación del ADN, que han conducido a un crecimiento exponencial de las descripciones de las moléculas de proteínas, ADN y ARN [31].

2 Estado del arte

Procedemos a realizar un estudio de las metodologías actuales en los ámbitos, introducidos previamente, del *Deep Learning* y de la bioinformática, con el objetivo de identificar las técnicas que se utilizan a nivel académico y en la industria.

2.1. Deep Learning

El gran potencial de las ANN es la alta velocidad de procesamiento que ofrecen en una implementación paralela masiva, lo que ha aumentado la necesidad de investigar en este ámbito. Hoy en día, las ANN se utilizan sobre todo para la aproximación de funciones universales en paradigmas numéricos debido a sus excelentes propiedades de autoaprendizaje, adaptabilidad, tolerancia a los fallos, no linealidad y avance en el mapeo de la entrada a la salida. Las ANN son capaces de resolver aquellos problemas que no pueden resolverse con la capacidad de cómputo de los procedimientos tradicionales y las matemáticas convencionales [32].

Los métodos de *Deep Learning* han resultado ser adecuados para el estudio de big data con un éxito notable en su aplicación al reconocimiento del habla, *computer vision*, el reconocimiento de patrones, los sistemas de recomendación y el procesamiento del lenguaje natural (NLP) [33]. En la actualidad, la innovación del *Deep Learning* en la identificación de imágenes, la detección de objetos, la clasificación de imágenes y las tareas de identificación de rostros tienen un gran éxito [32].

En nuestro estudio, evaluaremos 3 arquitecturas de *Deep Learning*: *autoencoder*, CNN y *Deep Feedforward Networks*. Estableceremos una comparación entre estas diferentes estructuras de ANN, además de mencionar avances recientes en estos algoritmos.

2.1.1. Autoencoder

Un *autoencoder* (AE) es un tipo de ANN. Se trata de un algoritmo de aprendizaje no supervisado que se utiliza para codificar eficazmente un conjunto de datos con el fin de reducir la dimensionalidad. Durante las últimas décadas, los AE han estado a la vanguardia en el ámbito

del *Deep Learning*. Los datos de entrada se convierten primero en una representación abstracta que, a continuación, la función codificadora vuelve a convertir en el formato original. En concreto, se entrena para codificar la entrada en alguna representación, de modo que la entrada pueda reconstruirse a partir de la misma [33]

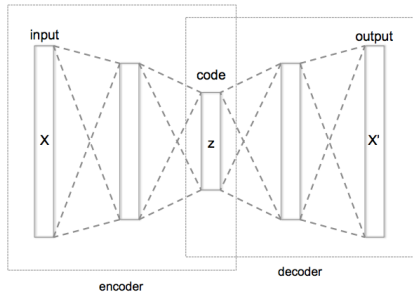

Figura 2.1: Diagrama de un *autoencoder*. Internamente presenta una capa (z) que describe un código para representar el input [34]

Un autoencoder se compone de 2 elementos:

Cuadro 2.1: Elementos de un *autoencoder*

Elemento	Función
Encoder	$h = f(x)$
Decoder	$r = g(h)$

El proceso de aprendizaje se describe como una optimización de una función de pérdida:

$$L(x, g(f(x))) \tag{2.1}$$

Donde L es la función de pérdida que penaliza $g(f(x))$ por ser distinto de x [35].

Tradicionalmente, los *autoencoders* se utilizaban para reducir la dimensionalidad o *feature learning*. Recientemente, ciertas teorías que

conectan los AE y los modelos de variables latentes han llevado a los autocodificadores a la vanguardia del modelado generativo [35].

En la actualidad, los *autoencoders* se utilizan para la reducción de ruido tanto en texto [36] como en imágenes [37], *clustering* no supervisado [38], generación de imágenes sintéticas [39], reducción de dimensionalidad [38] y predicción de secuencia a secuencia para la traducción automática [40].

2.1.2. Redes neuronales convolucionales

Una red neuronal convolucional (CNN, por sus siglas en inglés) es un tipo de red neuronal especializada en el procesamiento de datos que tienen una topología en forma de cuadrícula (*grid*). El nombre de red neuronal convolucionalïndica que la red emplea una operación matemática denominada convolución. Las redes convolucionales han tenido un enorme éxito en las aplicaciones prácticas [35].

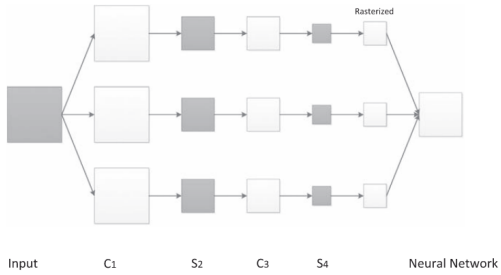

Figura 2.2: Diagrama de una CNN. Una CNN es una red neuronal multicapa, compuesta por dos tipos diferentes de capas, a saber, las capas de convolución (capas C) y las capas de submuestreo (capas S) [33]

En el contexto de una CNN, una convolución es una operación lineal que implica la multiplicación de un conjunto de pesos con la entrada, al igual que sucede en una red neuronal tradicional. La multiplicación se realiza entre una matriz de datos de entrada y una matriz bidimensional de pesos, llamada filtro o núcleo (*kernel*). El filtro es más pequeño que los datos de entrada, dado que permite que el mismo filtro (conjunto de pesos) se multiplique por la matriz de entrada varias veces en diferentes puntos de la entrada [41]. La operación queda representada de forma gráfica en la siguiente figura:

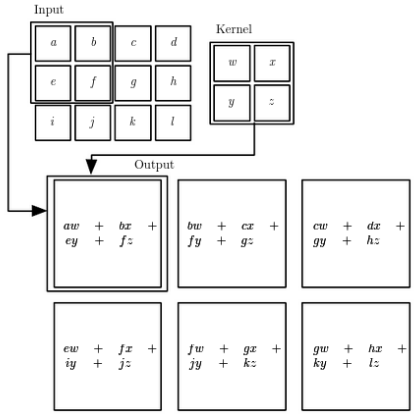

Figura 2.3: Representación de una convolución bidimensional. Restringimos la salida a sólo las posiciones en las que el núcleo se encuentra completamente dentro de la imagen. Los recuadros con flechas indican cómo se forma el tensor de salida, aplicando el núcleo a la correspondiente región superior izquierda del tensor de entrada [35]

Las capas de convolución (capas C) se utilizan para extraer características y las capas de submuestreo (capas S) son esencialmente capas de mapeo de características (*feature mapping*). Sin embargo, cuando la dimensionalidad de las entradas es igual a la de la salida del filtro, debido a la alta dimensionalidad, la aplicación de un clasificador puede provocar un *overfitting* o sobreajuste. Para resolver este problema, se introduce un proceso de *pooling*, i.e. submuestreo o *down-sampling*, para reducir el tamaño total de la señal [33].

En la actualidad, las CNN se utilizan para *computer vision*, tanto para la clasificación de imágenes [42] como para la segmentación [43], sistemas de recomendación [44] y análisis de sentimientos [45].

2.1.3. *Deep feedforward networks*

Las *deep feedforward networks*, i.e. *deep multilayer perceptrons*, son los modelos de *Deep Learning* por excelencia. El objetivo de una red *feedforward* es aproximar una función f, definiendo un mapeo:

$$y = f(x; \theta) \tag{2.2}$$

donde y es la categoría que deseamos como salida y θ es el valor de los parámetros que resultan en la mejor aproximación de la función. Estos modelos se denominan *feedforward* porque la información fluye a través de la función que se evalúa desde x, a través de los cálculos intermedios utilizados para definir f y, finalmente, a la salida y. No hay conexiones de retroalimentación (*feedback*) en las que las salidas del modelo se retroalimenten a sí mismas [35].

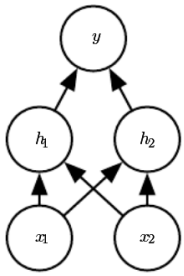

Figura 2.4: Diagrama de una *feedforward network* que contiene dos unidades, con una única capa oculta [35]

Las *deep feedforward networks* se pueden aplicar a una miríada de problemas, dado que se puede considerar la arquitectura de ANN más general. En la actualidad, las *deep feedforward networks* se pueden utilizar para la identificación automática del idioma [46], para la modelización de la propagación de enfermedades infecciosas [47] y para la predicción de la demanda de energía eléctrica [48].

2.2. Bioinformática

El estudio de las ómicas en biología molecular se beneficia de una serie de nuevas tecnologías que pueden ayudar a explicar vías, redes y procesos celulares, tanto normales como anormales, mediante el seguimiento simultáneo de miles de componentes moleculares. Las ómicas abarcan un conjunto cada vez más amplio de ramas, desde la **genómica** (el estudio cuantitativo de los genes codificantes de proteínas, los elementos reguladores y las secuencias no codificantes), la **transcriptómica** (ARN y expresión de genes), la **proteómica** (por ejemplo, centrada en la abundancia de proteínas) y la **metabolómica** (metabolitos y redes metabólicas) hasta los avances en la era de la biología

y la medicina postgenómica: farmacogenómica (estudio cuantitativo de cómo la genética afecta a la respuesta del huésped a los fármacos) y fisiómica (dinámica y funciones fisiológicas de organismos enteros) [49].

Los métodos de la bioinformática han demostrado ser eficaces para resolver los diversos problemas de las ómicas, concretamente para la obtención del estado transcriptómico de una célula (RNA-seq) [50], reconstrucción de las secuencias de ADN [51], anotación de genomas [52] y predicción de la estructura tridimensional de las proteínas [53]. Sin embargo, el problema de las tasas de error no negligibles en las tecnologías de secuenciación de ADN de segunda y tercera generación ha impulsado el desarrollo de múltiples técnicas bioinformáticas para paliar este contratiempo.

La corrección de errores con respecto a una posición genómica específica puede lograrse disponiendo todas las lecturas de forma horizontal, una tras otra, y examinando la base en cada posición específica de todas estas lecturas. Como los errores son infrecuentes y aleatorios, las lecturas que contienen un error en una posición específica pueden corregirse seleccionando la base más probable inferida a partir de las demás lecturas [54]. Entre estos métodos, destacamos *Coral* [55], *Quake* [56] y *MEC* [57].

El uso de *Deep Learning* para la corrección de errores de secuenciación es un área de investigación novedosa, en la cual cabe mencionar *NanoReviser*, un algoritmo recientemente publicado que emplea CNN y *Long Short-Term Memory network* (LSTM) para intentar corregir el patrón de error característico de la tecnología de tercera generación *Oxford Nanopore* [58].

3 Objetivos

1. Introducción al dominio de un problema de biología molecular: secuenciación de ADN y análisis de receptores de linfocitos T (TCR)
2. Introducción al análisis bioinformático de secuencias de ADN: preprocesamiento de lecturas, alineamiento y otros análisis bioinformáticos asociados
3. Revisión de la bibliografía científica reciente perteneciente al dominio del problema
4. Creación de un repositorio software para la generación *in silico* de secuencias de TCR y la simulación de la secuenciación de las mismas
5. Introducción al uso de Tensorflow y Keras para *Deep Learning*
6. Estudio de aplicación de Tensorflow/Keras a la corrección de errores de secuenciación en base a los datos sintetizados previamente

4 Planificación del proyecto

El presente proyecto pertenece al ámbito de la bioinformática, lo cual implica que es un trabajo interdisciplinar. Por lo tanto, se requiere una formación previa en áreas del conocimiento muy diversas. En particular, fue necesario un estudio de las bases bioquímicas, las *NGS*, la recombinación VDJ de los linfocitos T, el lenguaje de programación R, el *Deep Learning* y el funcionamiento de Tensorflow y Keras.

El proyecto se divide en 2 partes:

- locigenesis: Generación y secuenciación *in silico* de CDR3
- locimend: Corrección de los errores de secuenciación del ADN

La razón de esta segmentación se expone en la sección siguiente.

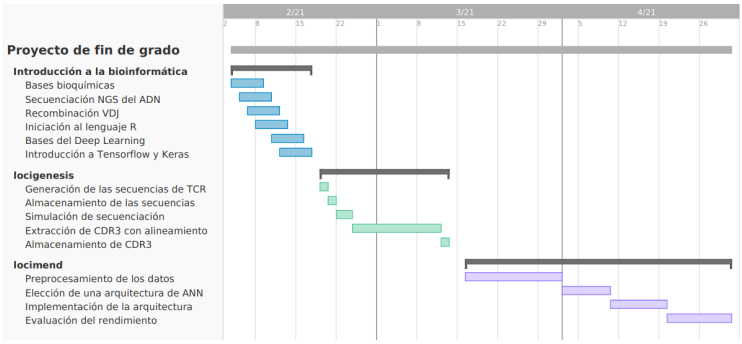

Figura 4.1: Cronograma del proyecto

La estimación temporal inicial del proyecto no se vio representada por la implementación real del mismo. Tanto la extracción de secuencias CDR3 mediante alineamiento múltiple, como el preprocesamiento de las secuencias para el uso de Tensorflow ocuparon la mayor parte del desarrollo del sistema.

Estas observaciones no son sorprendentes; en la práctica se ha comprobado que el preprocesamiento de los datos supone aproximadamente el 80 % del esfuerzo total de los proyectos de *machine learning* [59]. Incorporamos esta información a nuestro bagaje académico, como una forma de *feedback* positivo para mejorar la estimación temporal de los futuros proyectos relacionados con técnicas de aprendizaje automático.

5 Diseño y descripción del sistema

La finalidad de este proyecto es el desarrollo de un *pipeline*, con el objetivo de crear un algoritmo de *Deep Learning* capaz de corregir errores de secuenciación en secuencias de ADN, en particular, en la región CDR3 del TCR. Por ende, el trabajo consiste en el desarrollo *end-to-end* de un sistema de *machine learning*.

El sistema se compone de 2 partes, dado que el algoritmo de *Deep Learning* es generalizable y se podría proceder al entrenamiento de éste con otro conjunto de datos.

Cuadro 5.1: Descripción del proyecto

Elemento	Finalidad	Lenguaje de programación
locigenesis	Generación y secuenciación de CDR3	R
locimend	Corrección de errores de secuenciación	Python

El diseño del sistema queda plasmado a continuación:

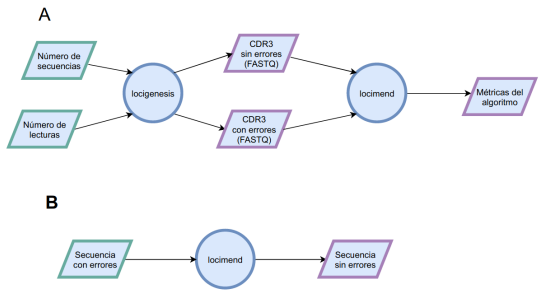

Figura 5.1: Diseño del sistema. (**A**) Entrenamiento del algoritmo de *Deep Learning*. Como *input* proporcionamos el número de secuencias, junto con el número de lecturas que deseamos que se simulen. Locigenesis generará 2 archivos en formato FASTQ, que contienen CDR3 con y sin errores de secuenciación, que son el *input* de locimend, cuya salida es el conjunto de métricas del algoritmo. (**B**) Inferencia del modelo de *Deep Learning* previamente entrenado y desplegado. Se provee como entrada una secuencia de ADN con errores de secuenciación, el algoritmo procesa ésta y devuelve una secuencia de ADN sin errores

Procedemos a la exposición de cada parte del *pipeline*, por separado, resaltando las tecnologías usadas y las funcionalidades.

5.1. locigenesis

locigenesis es una herramienta que genera un receptor de células T (TCR) humano, lo pasa por una herramienta de simulación de lectura de secuencias y extrae las regiones CDR3.

El objetivo de este proyecto es generar tanto secuencias de CDR3 con y sin errores de secuenciación, con el fin de crear *datasets* para entrenar un algoritmo de *Deep Learning*.

5.1.1. Tecnologías

- immuneSIM: generación *in silico* de repertorios de BCR y TCR, humanos y de ratón [60]
- CuReSim: simulador de secuenciación que emula la tecnología *Ion Torrent* [61]
- Biostrings: manipulación de secuencias biológicas [62]

5.1.2. Funcionamiento

El programa realiza, parametrizado por 2 parámetros de entrada (número de secuencias diferentes y número de lecturas por el simulador de secuenciación), los siguientes pasos:

1. Generación de diversas secuencias de la cadena β del TCR
2. Exportación de las secuencias a un archivo en formato FASTQ (tanto CDR3 como la secuencia VDJ completa)
3. Simulación de una secuenciación mediante CuReSim, y almacenamiento de las secuencias con errores
4. Alineamiento de las secuencias completas con errores, y extracción de CDR3 a partir de una heurística
5. Exportación de las secuencias de CDR3 con errores y sin errores en archivos con formato FASTQ

5.2. locimend

locimend es un algoritmo de *Deep Learning* que corrige errores de secuenciación de secuencias de ADN.

El objetivo de este proyecto es crear un modelo que pueda inferir la secuencia correcta de ADN, a partir de una secuencia de ADN con errores. Se trata de una reducción de ruido aplicada a un problema de genómica.

La arquitectura del modelo es una *deep feedforward network*, formada por:

- Capa de entrada
- Capa de *masking*: ignora ciertos valores que se añaden a cada tensor para homogenizar el tamaño del *input*
- 3 capas densas, seguidas de una capa de *dropout*: el *dropout* es un mecanismo que disminuye el *overfitting* o sobreajuste
- Capa densa

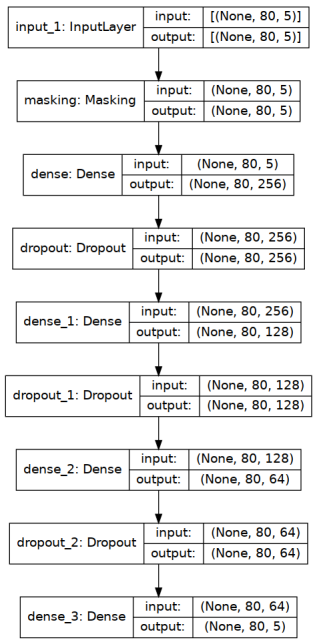

Figura 5.2: Arquitectura de la red neuronal

La interacción con el modelo se puede realizar mediante:

- Interfaz por línea de comandos (CLI)
- API REST

5.2.1. Tecnologías

- Tensorflow: creación y ejecución de algoritmos de *machine learning* [63]
- Biopython: manipulación de secuencias biológicas [64]
- FastAPI: *framework* web para la creación de APIs

5.2.2. Entrenamiento

El entrenamiento del modelo se realiza, a partir de 2 archivos FASTQ, en diferentes pasos:

1. Lectura de dos archivos FASTQ (secuencia de CDR3 correcta y secuencia de CDR3 con errores) simultáneamente para obtener las *features* y el *label*
2. Alineamiento de la secuencia correcta y de la secuencia con errores, para obtener entradas de la misma longitud
3. Codificación basada en el índice de las secuencias de ADN, para obtener secuencias numéricas
4. Conversión de las secuencias numéricas al formato binario TFRecord (basado en *Protocol Buffers*)
5. Separación y almacenamiento del *dataset* en conjunto de entrenamiento, validación y test
6. Lectura paralela de los conjuntos del *dataset*
7. *One-hot encoding* de las secuencias numéricas, las cuales representan las secuencias de ADN
8. Entrenamiento del algoritmo de *Deep Learning*, con los datos preprocesados
9. Obtención de las métricas que miden el rendimiento del modelo

5.2.3. Inferencia

La inferencia consiste en la predicción de nuevos resultados a partir de un modelo pre-entrenado. Se llevan a cabo los siguientes pasos:

1. Carga del modelo pre-entrenado
2. Lectura de la secuencia de ADN con errores
3. Codificación basada en el índice de la secuencia de ADN, para obtener une secuencia numérica

4. *One-hot encoding* de las secuencia numérica, la cual representa la secuencia de ADN
5. Predicción de la secuencia correcta de ADN
6. Descodificación de la secuencia numérica hasta obtener una secuencia de ADN
7. Presentación de la secuencia de ADN correcta inferida por el modelo

5.3. Reproducibilidad

La reproducibilidad de los experimentos en la ciencia es un elemento esencial en el método científico, el cual asegura que una técnica novedosa ofrece resultados verídicos. Actualmente, nos encontramos en una etapa de crisis de reproducibilidad, donde el 70 % de los investigadores han fallado al tratar de replicar el estudio de otro científico [65].

Asimismo, en el ámbito de la informática pocos experimentos computacionales son documentados de forma precisa. Por lo general, no existe un registro del flujo de trabajo, la configuración del *hardware* y el *software* del equipo, la configuración de los parámetros o las secuencias de invocación de funciones. El código fuente a menudo se revisa sin dejar constancia de ello. Además de dificultar la reproducibilidad de los resultados, estas prácticas acaban impidiendo la productividad de los propios investigadores [66].

Recientemente han surgido nuevos enfoques para lidiar con este problema desde el punto de vista de la resolución de dependencias de un proyecto *software*. En el presente trabajo, usamos el gestor de paquetes Nix [67], para garantizar que los resultados que obtenemos son reproducibles en cualquier máquina.

6 Resultados

El algoritmo de *Deep Learning* fue entrenado con un *dataset* sintético de las secuencias de la región CDR3 del TCR. En concreto, se generó un dataset de 20,000 secuencias, procedentes de una simulación de secuenciación (reproducida durante 100 iteraciones), de 200 secuencias únicas. Este *dataset* se proporciona en el repositorio de locimend.

Dado que los conjuntos de datos de entrenamiento estaban completamente anotados (i.e. aprendizaje supervisado), el problema se formuló como una tarea de clasificación binaria supervisada de predicción de una base dada como error/no error, y el rendimiento se midió en unos *datasets* de validación y de test.

El entrenamiento del modelo de *Deep Learning* se efectuó en un PC con un procesador Ryzen 5 2600X (6 núcleos, 12 hebras) y 16 GB de RAM. El tiempo de ejecución fue de 23 minutos.

Presentamos las métricas obtenidas al finalizar el entrenamiento de locimend:

Cuadro 6.1: Rendimiento de locimend con cada *dataset*

Dataset	Accuracy	AUC
Validación	0.89	0.98
Test	0.89	0.98

En definitiva, locimend adquirió una gran capacidad para discernir entre nucleótidos erróneos y correctos, en ambos *datasets*, con una precisión de 0,89 y un área bajo la curva (AUC) de 0,98.

7 Conclusiones

8 Futuras mejoras

Los resultados del algoritmo de *Deep Learning* son satisfactorios aunque presenta ciertas limitaciones.

En primer lugar, el modelo dado que la red neuronal se entrenó exclusivamente con datos que emulan una secuenciación de Ion Torrent, es probable que las características que contribuyen a las posiciones erróneas sean específicas al proceso de secuenciación de Ion Torrent. Por lo tanto, la inferencia a partir de datos de secuenciación procedentes de otras tecnologías podría ser menos eficaz. Un entrenamiento con varios *dataset* provenientes de diferentes tecnologías aumentaría la capacidad de generalización del modelo.

Asimismo, el entrenamiento del modelo se realizó con un conjunto de datos de 20,000 secuencias, un tamaño bastante limitado para una tarea de aprendizaje automático. La generación de *datasets* de mayor tamaño no era viable debido a limitaciones en las capacidades de cómputo de los recursos disponibles. Por ende, la elaboración de un *dataset* de mayor envergadura es una tarea pendiente.

Uno de los factores que más afecta el rendimiento de una red neuronal es el valor de los hiperparámetros. Los hiperparámetros son variables inherentes al modelo, como el optimizador usado, la distribución de los valores iniciales en los pesos del modelo, la tasa de aprendizaje o el *batch size*. Todos estos parámetros son interdependientes [68], por lo que es importante optimizarlos de forma combinatoria. Una optimización más a fondo de los hiperparámetros podría incrementar la eficiencia del modelo.

Finalmente, el desarrollo y despliegue de un *frontend* web para la API REST permitiría mejorar la accesibilidad del modelo para los investigadores que deseen usar este modelo.

Bibliografía

[1] M. M. C. Albert Lehninger David L. Nelson, *Lehninger-principles of biochemistry*, 5th Edition. W. H. Freeman, 2008, p. 276.

[2] A. M. Michael A. Lieberman, *Mark's basic medical biochemistry a clinical approach*, Third. Lippincott Williams & Wilkins, 2008, pp. 209, 260.

[3] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, no. 5258, pp. 561–563, Aug. 1970, doi: 10.1038/227561a0.

[4] F. H. Crick, "On protein synthesis," in *Symp soc exp biol*, 1958, vol. 12, p. 8.

[5] F. Sanger, S. Nicklen, and A. R. Coulson, "DNA sequencing with chain-terminating inhibitors," *Proceedings of the National Academy of Sciences*, vol. 74, no. 12, pp. 5463–5467, 1977, doi: 10.1073/pnas.74.12.5463.

[6] I. H. G. S. Consortium, "Finishing the euchromatic sequence of the human genome," *Nature*, vol. 431, no. 7011, pp. 931–945, Oct. 2004, doi: 10.1038/nature03001.

[7] J. A. Schloss, "How to get genomes at one ten-thousandth the cost," *Nature Biotechnology*, vol. 26, no. 10, pp. 1113–1115, Oct. 2008, doi: 10.1038/nbt1008-1113.

[8] E. L. van Dijk, H. Auger, Y. Jaszczyszyn, and C. Thermes, "Ten years of next-generation sequencing technology," *Trends in Genetics*, vol. 30, no. 9, pp. 418–426, Sep. 2014, doi: 10.1016/j.tig.2014.07.001.

[9] J. M. Heather and B. Chain, "The sequence of sequencers: The history of sequencing DNA," *Genomics*, vol. 107, no. 1, pp. 1–8, 2016, doi: https://doi.org/10.1016/j.ygeno.2015.11.003.

[10] S. Nurk *et al.*, "The complete sequence of a human genome," *bioRxiv*, 2021, doi: 10.1101/2021.05.26.445798.

[11] K. Reinert, B. Langmead, D. Weese, and D. J. Evers, "Alignment of next-generation sequencing reads," *Annual Review of Genomics and Human Genetics*, vol. 16, no. 1, pp. 133–151, 2015, doi: 10.1146/annurev-genom-090413-025358.

[12] C. Lee, "Generating consensus sequences from partial order multiple sequence alignment graphs," *Bioinformatics*, vol. 19, no. 8, pp. 999–1008, May 2003, doi: 10.1093/bioinformatics/btg109.

[13] A. Nagar and M. Hahsler, "Fast discovery and visualization of conserved regions in DNA sequences using quasi-alignment," *BMC Bioinformatics*, vol. 14, no. 11, p. S2, Sep. 2013, doi: 10.1186/1471-2105-14-S11-S2.

[14] J. J. Salk, M. W. Schmitt, and L. A. Loeb, "Enhancing the accuracy of next-generation sequencing for detecting rare and subclonal mutations," *Nature Reviews Genetics*, vol. 19, no. 5, pp. 269–285, May 2018, doi: 10.1038/nrg.2017.117.

[15] M. Shugay *et al.*, "Towards error-free profiling of immune repertoires," *Nature Methods*, vol. 11, no. 6, pp. 653–655, Jun. 2014, doi: 10.1038/nmeth.2960.

[16] H. S. Robins *et al.*, "Comprehensive assessment of T-cell receptor beta-chain diversity in alphabeta T cells," *Blood*, vol. 114, no. 19, pp. 4099–4107, Nov. 2009.

[17] M. S. B. Cantos, "Análisis de repertorios de receptores de células t a partir de datos de secuenciación masiva," Master's thesis, Universidad de Granada, 2019.

[18] A. K. Abbas, A. H. Lichtman, and S. Pillai, in *Cellular and molecular immunology*, 9th ed., Elsevier, 2017, p. 204.

[19] P. N. Stuart Russell, *Artificial intelligence: A modern approach*, 3rd ed. Prentice Hall, 2010, pp. 38–45, 48–49, 55–56.

[20] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI Magazine*, vol. 27, no. 4, p. 12, 2006, doi: 10.1609/aimag.v27i4.1904.

[21] A. P. Engelbrecht, *Computational intelligence. An introduction*, 2nd ed. Wiley, 2007, pp. 39–40.

[22] J. Zou, Y. Han, and S.-S. So, "Overview of artificial neural networks," in *Artificial neural networks: Methods and applications*, D. J. Livingstone, Ed. Totowa, NJ: Humana Press, 2009, pp. 14–22. doi: 10.1007/978-1-60327-101-1_2.

[23] D. Graupe, *Principles of artificial neural networks*, 3. ed. World Scientific Publ, 2013, pp. 28–31.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[25] A. K. K. Jeremy Watt Reza Borhani, *Machine learning refined: Foundations, algorithms, and applications*, 2nd ed. Cambridge University Press, 2020.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[27] H. Wang, B. Raj, and E. P. Xing, "On the origin of deep learning," *CoRR*, vol. abs/1702.07800, 2017,Available: http://arxiv.org/abs/1702.07800

[28] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010, doi: 10.1162/NECO_a_00052.

[29] T. Can, "Introduction to bioinformatics," in *miRNomics: MicroRNA biology and computational analysis*, M. Yousef and J. Allmer, Eds. Totowa, NJ: Humana Press, 2014, pp. 51–71. doi: 10.1007/978-1-62703-748-8_4.

[30] J. B. Hagen, "The origins of bioinformatics," *Nature Reviews Genetics*, vol. 1, no. 3, pp. 231–236, Dec. 2000, doi: 10.1038/35042090.

[31] S. B. Pierre Baldi, *Bioinformatics: The machine learning approach*, 2nd ed. The MIT Press, 2001, p. 12.

[32] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018, doi: https://doi.org/10.1016/j.heliyon.2018.e00938.

[33] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017, doi: https://doi.org/10.1016/j.neucom.2016.12.038.

[34] Chervinskii, "Autoencoder structure," *Wikimedia*. Dec. 2015.Available: https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[36] M. Lewis *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, doi: 10.18653/v1/2020.acl-main.703.

[37] S. A. Bigdeli and M. Zwicker, "Image restoration using auto-encoding priors," *CoRR*, 2017,Available: http://arxiv.org/abs/1703.09964v1

[38] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *CoRR*, 2015,Available: http://arxiv.org/abs/1511.05644v2

[39] J. Yoo, H. Lee, and N. Kwak, "Unpriortized autoencoder for image generation," *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, doi: 10.1109/icip40778.2020.9191173.

[40] Ł. Kaiser and S. Bengio, "Discrete autoencoders for sequence models," *CoRR*, 2018,Available: http://arxiv.org/abs/1801.09797v1

[41] J. Brownlee, "How do convolutional layers work in deep learning neural networks?" *Machine Learning Mastery*. Apr. 2020.Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[42] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, 2017,Available: http://arxiv.org/abs/1704.04861v1

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, 2015,Available: http://arxiv.org/abs/1505.04597v1

[44] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," *CoRR*, 2018,Available: http://arxiv.org/abs/1808.05163v4

[45] H. Sadr, M. N. Solimandarabi, M. M. Pedram, and M. Teshnehlab, "A novel deep learning method for textual sentiment analysis," *CoRR*, 2021,Available: http://arxiv.org/abs/2102.11651v1

[46] I. Lopez-Moreno, J. Gonzalez-Dominguez, D. Martinez, O. Plchot, J. Gonzalez-Rodriguez, and P. J. Moreno, "On the use of deep feedforward neural networks for automatic language identification," *Computer Speech & Language*, vol. 40, pp. 46–59, Nov. 2016, doi: 10.1016/j.csl.2016.03.001.

[47] S. Chakraborty, A. K. Choudhary, M. Sarma, and M. K. Hazarika, "Reaction order and neural network approaches for the simulation of COVID-19 spreading kinetic in india," *Infectious Disease Modelling*, vol. 5, pp. 737–747, 2020, doi: 10.1016/j.idm.2020.09.002.

[48] M. Mansoor, F. Grimaccia, S. Leva, and M. Mussetta, "Comparison of echo state network and feed-forward neural networks in electrical load forecasting for demand response programs," *Mathematics and Computers in Simulation*, vol. 184, pp. 282–293, Jun. 2021, doi: 10.1016/j.matcom.2020.07.011.

[49] M. V. Schneider and S. Orchard, "Omics technologies, data and bioinformatics principles," *Bioinformatics for Omics Data*, pp. 3–30, 2011, doi: 10.1007/978-1-61779-027-0_1.

[50] S. Peri *et al.*, "Read mapping and transcript assembly: A scalable and high-throughput workflow for the processing and analysis of ribonucleic acid sequencing data," *Frontiers in Genetics*, vol. 10, Jan. 2020, doi: 10.3389/fgene.2019.01361.

[51] D. R. Zerbino and E. Birney, "Velvet: Algorithms for de novo short read assembly using de bruijn graphs," *Genome Research*, vol. 18, no. 5, pp. 821–829, Feb. 2008, doi: 10.1101/gr.074492.107.

[52] G. Spudich, X. M. Fernandez-Suarez, and E. Birney, "Genome browsing with ensembl: A practical overview," *Briefings in Functional Genomics and Proteomics*, vol. 6, no. 3, pp. 202–219, Aug. 2007, doi: 10.1093/bfgp/elm025.

[53] Y. Liu, Q. Ye, L. Wang, and J. Peng, "Learning structural motif representations for efficient protein structure search," *Bioinformatics*, vol. 34, no. 17, pp. i773–i780, Sep. 2018, doi: 10.1093/bioinformatics/bty585.

[54] X. Yang, S. P. Chockalingam, and S. Aluru, "A survey of error-correction methods for next-generation sequencing," *Briefings in Bioinformatics*, vol. 14, no. 1, pp. 56–66, Apr. 2012, doi: 10.1093/bib/bbs015.

[55] L. Salmela and J. Schroder, "Correcting errors in short reads by multiple alignments," *Bioinformatics*, vol. 27, no. 11, pp. 1455–1461, Apr. 2011, doi: 10.1093/bioinformatics/btr170.

[56] D. R. Kelley, M. C. Schatz, and S. L. Salzberg, "Quake: Quality-aware detection and correction of sequencing errors," *Genome Biology*, vol. 11, no. 11, p. R116, 2010, doi: 10.1186/gb-2010-11-11-r116.

[57] L. Zhao, Q. Chen, W. Li, P. Jiang, L. Wong, and J. Li, "MapReduce for accurate error correction of next-generation sequencing data," *Bioinformatics*, vol. 33, no. 23, pp. 3844–3851, Feb. 2017, doi: 10.1093/bioinformatics/btx089.

[58] L. Wang, L. Qu, L. Yang, Y. Wang, and H. Zhu, "NanoReviser: An error-correction tool for nanopore sequencing based on a deep learning algorithm," *Frontiers in Genetics*, vol. 11, p. 900, 2020, doi: 10.3389/fgene.2020.00900.

[59] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining," *Applied Artificial Intelligence*, vol. 17, no. 5–6, pp. 375–381, May 2003, doi: 10.1080/713827180.

[60] C. R. Weber *et al.*, "immuneSIM: Tunable multi-feature simulation of b- and t-cell receptor repertoires for immunoinformatics benchmarking," *Bioinformatics*, vol. 36, no. 11, pp. 3594–3596, Apr. 2020, doi: 10.1093/bioinformatics/btaa158.

[61] S. Caboche, C. Audebert, Y. Lemoine, and D. Hot, "Comparison of mapping algorithms used in high-throughput sequencing: Application to ion torrent data," *BMC Genomics*, vol. 15, no. 1, p. 264, 2014, doi: 10.1186/1471-2164-15-264.

[62] H. Pagès, P. Aboyoun, R. Gentleman, and S. DebRoy, "Biostrings: Efficient manipulation of biological strings." 2019.

[63] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems." 2015.Available: https://www.tensorflow.org/

[64] P. J. A. Cock *et al.*, "Biopython: Freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, Mar. 2009, doi: 10.1093/bioinformatics/btp163.

[65] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, May 2016, doi: 10.1038/533452a.

[66] V. Stodden, J. Borwein, and D. H. Bailey, "Publishing standards for computational science: 'Setting the default to reproducible'." 2013.

[67] E. Dolstra, M. Jonge, and E. Visser, "Nix: A safe and policy-free system for software deployment." Jan. 2004, pp. 79–92.

[68] F. Chollet, *Deep learning with python*. Manning, 2017.