UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE
INGENIERÍA
UDELAR

Estudio y Aplicación del Modelo HyenaDNA en la Detección de Ancestría Genética

Gonzalo Cameto

Programa de Posgrado en Ingeniería Matemática
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Mayo de 2026

UNIVERSIDAD DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERÍA
UDELAR

Estudio y Aplicación del Modelo HyenaDNA en la Detección de Ancestría Genética

Gonzalo Cameto

Tesis de Maestría presentada al Programa de Posgrado en Ingeniería Matemática, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magister en Ingeniería Matemática.

Director de tesis:
D.Sc. Prof. María Inés Fariello

Codirector:
D.Sc. Prof. Federico Lecumberry

Director académico:
D.Sc. Prof. Marcelo Fiori

Montevideo – Uruguay
Mayo de 2026

Cameto, Gonzalo

Estudio y Aplicación del Modelo HyenaDNA en la Detección de Ancestría Genética / Gonzalo Cameto. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2026.

XIII, 101 p. 29, 7cm.

Director de tesis:

María Inés Fariello

Codirector:

Federico Lecumberry

Director académico:

Marcelo Fiori

Tesis de Maestría – Universidad de la República, Programa de Ingeniería Matemática, 2026.

Referencias bibliográficas: p. 91 – 95.

1. Deep Learning, 2. Genómica, 3. HyenaDNA, 4. Ancestría genética. I. Fariello, María Inés *et al.* II. Universidad de la República, Programa de Posgrado en Ingeniería Matemática. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Dr. Guillermo Moncecchi

Dr. Bernardo Marenco

Dr. Miguel Pérez-Enciso

Montevideo – Uruguay
Mayo de 2026

Agradecimientos

A mis tutores y director académico, María Inés Fariello, Federico Lecumberry y Marcelo Fiori, por la guía constante, la paciencia frente a mis preguntas elementales sobre genómica, y por acercarme a un campo de investigación tan fascinante.

A la Facultad de Ingeniería de la Universidad de la República, por la formación recibida y por haber sido el espacio donde esta tesis fue posible.

A mi pareja, familia y amigos, por el apoyo constante a lo largo de este proceso, y por escuchar con paciencia mis explicaciones de *machine learning* y de genómica.

Biology at its most fundamental level can be thought of as an information processing system, albeit a phenomenally complex and emergent one.

Demis Hassabis, *Nobel Lecture*, 2024

RESUMEN

La inferencia de ancestría local (*Local Ancestry Inference*, LAI) consiste en determinar el origen ancestral de cada segmento del genoma de un individuo, una tarea relevante para la medicina personalizada, los estudios de genética de poblaciones y la comprensión de la historia demográfica humana. Métodos establecidos como RFMix (Maples et al., 2013) y Gnomix (Hilmarsson et al., 2022) abordan esta tarea mediante modelos estadísticos y de aprendizaje automático. En este contexto, resulta relevante explorar si arquitecturas de aprendizaje profundo con complejidad subcuadrática pueden abordar eficazmente este tipo de tarea genómica.

Los modelos de lenguaje basados en Transformers (Vaswani et al., 2017) han demostrado capacidad para modelar secuencias biológicas, pero su complejidad cuadrática en la longitud de secuencia limita su aplicación a genomas completos. HyenaDNA (Nguyen et al., 2023) es un modelo genómico preentrenado que utiliza convoluciones implícitas para procesar secuencias de hasta 1 millón de nucleótidos con complejidad $O(N \log N)$, permitiendo capturar dependencias de largo alcance de forma eficiente.

El objetivo principal de esta tesis es estudiar y aplicar HyenaDNA en la detección de ancestría genética. Se realiza un análisis de su arquitectura y funcionamiento, con especial énfasis en las convoluciones implícitas y su escalabilidad subcuadrática. Se compara su rendimiento con modelos basados en Transformers y se evalúa su aplicación práctica en tareas genómicas, incluyendo clasificación de especies y detección de ancestría continental a partir de polimorfismos de nucleótido único (SNPs).

Palabras claves:
Deep Learning, Genómica, HyenaDNA, Ancestría genética.

ABSTRACT

Local Ancestry Inference (LAI) consists of determining the ancestral origin of each segment of an individual's genome, a task relevant to personalized medicine, population genetics studies, and the understanding of human demographic history. Established methods such as RFMix (Maples et al., 2013) and Gnomix (Hilmarsson et al., 2022) address this task using statistical and machine learning models. In this context, it is relevant to explore whether deep learning architectures with sub-quadratic complexity can effectively tackle this type of genomic task.

Transformer-based language models (Vaswani et al., 2017) have demonstrated the ability to model biological sequences, but their quadratic complexity in sequence length limits their application to whole genomes. HyenaDNA (Nguyen et al., 2023) is a pre-trained genomic model that uses implicit convolutions to process sequences of up to 1 million nucleotides with $O(N \log N)$ complexity, enabling the efficient capture of long-range dependencies.

The main goal of this thesis is to study and apply HyenaDNA to genetic ancestry detection. An analysis of its architecture and functioning is carried out, with special emphasis on implicit convolutions and their sub-quadratic scalability. Its performance is compared to Transformer-based models, and its practical application in genomic tasks is evaluated, including species classification and continental ancestry detection from single nucleotide polymorphisms (SNPs).

Keywords:
Deep Learning, Genomics, HyenaDNA, Genetic Ancestry.

Tabla de contenidos

Capítulo 1

Introducción y Objetivos

1.1. Introducción

La detección de ancestría genética busca determinar el origen poblacional de los segmentos del genoma de un individuo. En particular, la inferencia de ancestría local (*Local Ancestry Inference*, LAI) asigna un origen ancestral a cada segmento cromosómico, lo que resulta relevante para la medicina personalizada —donde la respuesta a fármacos puede depender del trasfondo genético (Relling and Evans, 2015)— y para estudios de genética de poblaciones que buscan reconstruir historias demográficas y patrones migratorios. Herramientas como RFMix (Maples et al., 2013) y Gnomix (Hilmarsson et al., 2022) son métodos establecidos que abordan esta tarea mediante pipelines de múltiples etapas: clasificadores locales por ventana seguidos de modelos de suavizado (como CRF o XGBoost), requiriendo además datos faseados, mapas genéticos y paneles de referencia. En este contexto, resulta relevante explorar si un modelo unificado *end-to-end* basado en arquitecturas de aprendizaje profundo con complejidad subcuadrática puede abordar eficazmente este tipo de tarea genómica, simplificando el pipeline de procesamiento. Los resultados de esta comparación se presentan en el Capítulo 4.

La secuenciación genómica codifica una gran cantidad de información crucial para la regulación de genes y la síntesis de proteínas. Los modelos de lenguaje basados en Transformers (Vaswani et al., 2017) han demostrado capacidad para modelar secuencias biológicas, pero su complejidad cuadrática $O(N^2)$ en la longitud de secuencia limita severamente la ventana de contexto utilizable. Por ejemplo, DNABERT (Ji et al., 2021) opera con un máximo de

512 tokens, y modelos previos basados en Transformers típicamente no superan los 4096 tokens de contexto. Arquitecturas híbridas como Enformer (Avsec et al., 2021) extienden el campo receptivo a ∼200 kb mediante convoluciones dilatadas combinadas con atención, pero aún no alcanzan la escala de un genoma completo.

HyenaDNA (Nguyen et al., 2023) es un modelo genómico pre-entrenado en un genoma humano que utiliza convoluciones implícitas para permitir longitudes de contexto de hasta 1 millón de tokens a nivel de nucleótido único. Este modelo se entrena hasta 160 veces más rápido que los Transformers, superándolo en diversas tareas.

El modelo combina la arquitectura Hyena (Poli et al., 2023) con el procesamiento de secuencias de ADN. La arquitectura Hyena fue diseñada específicamente para resolver el problema de escalado cuadrático en la longitud de secuencia que presentan los Transformers, respondiendo a la pregunta: "¿Existen operadores subcuadráticos que puedan igualar la calidad de la atención a escala?". La solución propuesta fue "Hyena Hierarchy", un operador definido por una recurrencia de dos primitivas subcuadráticas eficientes: una convolución larga y una compuerta multiplicativa elemento a elemento.

Las secuencias de ADN presentan desafíos particulares para su modelado computacional. A diferencia de las secuencias procesadas por modelos de lenguaje natural, cuya longitud típica rara vez supera unos miles de tokens, las secuencias genómicas son órdenes de magnitud más largas: el genoma humano tiene 3200 millones de nucleótidos. Además, presentan dependencias e interacciones de largo alcance que pueden abarcar más de 100 000 nucleótidos, lo que hace inviable el uso directo de arquitecturas con complejidad cuadrática en la longitud de secuencia.

En este contexto, las arquitecturas subcuadráticas como HyenaDNA resultan de particular interés. A diferencia de los Transformers, cuyo mecanismo de atención escala como $O(N^2)$, HyenaDNA mantiene una complejidad de $O(N \log N)$ mediante convoluciones implícitas, lo que permite procesar secuencias extensas con resolución de un nucleótido. Esta tesis busca evaluar si este tipo de arquitectura es apta para tareas genómicas donde el contexto largo es relevante, como la detección de ancestría a partir de SNPs distribuidos a lo largo de segmentos cromosómicos extensos.

1.2. Objetivos

1.2.1. Objetivos Generales

Los objetivos generales de esta tesis son comprender en profundidad el funcionamiento de HyenaDNA y su aplicación en genómica, incluyendo un análisis detallado desde el punto de vista matemático del modelado de secuencias mediante convoluciones implícitas; y analizar la arquitectura del modelo y su escalabilidad subcuadrática.

1.2.2. Objetivos Específicos

Los objetivos específicos son: implementar y evaluar HyenaDNA en tareas de clasificación de especies como primer acercamiento al modelo; estudiar las convoluciones implícitas y su implementación en HyenaDNA, validando matemáticamente su comportamiento en escenarios controlados; aplicar HyenaDNA a la detección de ancestrías genéticas, evaluando su rendimiento con diferentes representaciones de entrada y longitudes de secuencia; y comparar HyenaDNA con modelos basados en Transformers, evaluando las diferencias y similitudes en términos de rendimiento y complejidad computacional.

1.3. Estructura del trabajo

La investigación se articuló en cuatro bloques:

- **Fundamentos teóricos** (Capítulo 2): se describieron las bases de HyenaDNA, las limitaciones de Transformers para secuencias largas y las principales estrategias de codificación posicional aplicables a genómica.
- **Metodología experimental** (Capítulo 3): se definió el *pipeline* experimental, incluyendo datasets, estrategias de muestreo, configuraciones de entrenamiento y métricas de evaluación.
- **Experimentos progresivos** (Capítulo 4): se desarrolló una serie de experimentos que fueron desde validaciones básicas hasta un sistema optimizado de detección de ancestría.
- **Análisis e interpretación**: se estudiaron las representaciones internas del modelo y se realizaron comparaciones controladas con arquitecturas Transformer equivalentes.

Capítulo 2

Fundamentos teóricos

Este capítulo presenta los fundamentos teóricos que sustentan la aplicación de modelos fundacionales de largo alcance a la genómica, con particular énfasis en la detección de ancestría genética. Se inicia con los principios del modelado computacional de secuencias genómicas, continúa con las limitaciones de los Transformers tradicionales, describe las estrategias para manejar contexto largo en genómica, y culmina con la presentación del operador *Hyena* y el modelo *HyenaDNA* empleado en esta tesis.

2.1. Modelado computacional de secuencias genómicas

El ADN puede representarse como una secuencia de símbolos provenientes de un alfabeto mínimo $\mathcal{A} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, donde cada letra corresponde a una base nitrogenada específica: adenina (A), citosina (C), guanina (G) y timina (T). Estas cuatro bases constituyen los bloques fundamentales de la información genética, complementándose en pares (A-T y C-G) para formar la estructura de doble hélice del ADN.

En el contexto del procesamiento computacional de secuencias, cada símbolo del alfabeto se denomina *token*, la unidad mínima de información que el modelo puede procesar (Sennrich et al., 2016). En genómica, un token puede corresponderse con un nucleótido individual (Nguyen et al., 2023), o pueden utilizarse estrategias alternativas donde se agrupan múltiples haplotipos en tokens denominados k-mers (Ji et al., 2021). Adicionalmente a A, C, G y T, se incluye un token "N" que representa ambigüedad cuando la identidad de la

base no puede determinarse con certeza (Nguyen et al., 2023).

Desde la perspectiva computacional, modelar estas secuencias implica construir representaciones numéricas y arquitecturas algorítmicas capaces de aprender patrones útiles para tareas específicas.

Al igual que en Procesamiento de Lenguaje Natural (PLN), los modelos fundacionales en genómica buscan aprender representaciones que capturen dependencias de corto y largo alcance, sirviendo como punto de partida para múltiples tareas descendentes (*fine-tuning* o *prompting*). Sin embargo, el dominio genómico presenta retos particulares que lo distinguen de otras aplicaciones de aprendizaje de secuencias.

El desafío más evidente es la longitud extrema: las secuencias genómicas pueden alcanzar hasta 3200 millones de nucleótidos en el caso del genoma humano, varios órdenes de magnitud mayor que documentos típicos en PLN. Esta escala se ve agravada por las dependencias de largo alcance, donde las interacciones regulatorias pueden abarcar más de 100 000 nucleótidos, requiriendo arquitecturas capaces de mantener información a lo largo de contextos extensos.

Simultáneamente, el dominio presenta baja densidad de señal: solo una pequeña fracción de la secuencia contiene información discriminativa (como las variantes SNPs detalladas en la siguiente sección), mientras que la mayoría del contenido es altamente conservado entre individuos. Esta situación plantea el desafío de identificar señales sutiles en las secuencias a procesar. Finalmente, la estructura contextual resulta crítica, ya que la posición relativa de los elementos y su contexto genómico son fundamentales para comprender su función e impacto biológico (Avsec et al., 2021).

2.1.1. SNPs y detección de ancestría genética

Los polimorfismos de un solo nucleótido (SNPs) son variaciones en la secuencia de ADN donde un nucleótido difiere entre individuos en la misma posición genómica. Estos representan la forma más común de variación genética humana, ocurriendo aproximadamente una vez cada $600 - 700$ nucleótidos a lo largo del genoma humano (The 1000 Genomes Project Consortium, 2015).

Los SNPs constituyen marcadores especialmente valiosos para la detección de ancestría genética debido a tres características fundamentales. En primer lugar, actúan como marcadores poblacionales: las poblaciones humanas en di-

ferentes regiones del mundo exhiben frecuencias alélicas —proporción de una de las variantes genéticas en la población— que son distintivas respecto de las de otras poblaciones, reflejando su historia evolutiva particular (Patterson et al., 2006).

Adicionalmente, los patrones de ligamiento resultan informativos, ya que los haplotipos —combinaciones específicas de alelos heredadas conjuntamente— preservan información sobre la historia demográfica y los patrones migratorios de las poblaciones ancestrales (The International HapMap Consortium, 2005; Patterson et al., 2012).

Finalmente, proporcionan una señal ancestral acumulativa, dado que el análisis conjunto de múltiples SNPs en ventanas genómicas permite reconstruir el origen ancestral de segmentos cromosómicos específicos con alta precisión (Maples et al., 2013).

El desafío computacional radica en procesar eficientemente estos patrones dispersos de variación manteniendo la resolución de nucleótido individual, crucial para detectar señales ancestrales sutiles.

2.1.2. Representación computacional del ADN

Los modelos fundacionales en genómica emplean *embeddings* entrenables, que asignan a cada nucleótido o variante un vector continuo en un espacio de alta dimensionalidad. Estas representaciones numéricas permiten que las arquitecturas de aprendizaje automático procesen eficientemente la información genética y capturen patrones complejos presentes en las secuencias de ADN.

La calidad de estas representaciones es crítica para el éxito de tareas como la clasificación de especies o la detección de ancestría, ya que determinan qué información está disponible para que el modelo aprenda patrones discriminativos relevantes.

2.2. Transformers y el mecanismo de atención

El modelo Transformer (Vaswani et al., 2017) representa un cambio de paradigma respecto a las arquitecturas recurrentes. Sin embargo, su complejidad computacional y de memoria cuadrática $\mathcal{O}(L^2)$ con respecto a la longitud de la secuencia L limita su aplicabilidad a secuencias muy largas, como se observa en modelos como DNABERT (Ji et al., 2021), que suelen limitarse a longitudes

de hasta aproximadamente 4000 tokens. Matemáticamente, el Transformer se define como una función de mapeo que transforma una secuencia de vectores de entrada en una secuencia de vectores de salida mediante la composición de funciones de atención y transformaciones no lineales puntuales, preservando la dimensión del espacio latente a través de conexiones residuales.

2.2.1. Proyecciones Lineales y Espacios Latentes

Sea $\mathbf{X} \in \mathbb{R}^{L \times D}$ la matriz de entrada, donde L es la longitud de la secuencia y D la dimensión del modelo. En el mecanismo de *Self-Attention* (Autoatención), la entrada se proyecta a tres subespacios distintos mediante transformaciones lineales parametrizadas, generando las matrices de *Query* ($\mathbf{Q}$), *Key* ($\mathbf{K}$) y *Value* ($\mathbf{V}$).

En la configuración de atención multi-cabezal (detallada en la Sección 2.2.4), cada cabezal i realiza sus propias proyecciones independientes:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V, \tag{2.1}$$

donde $\mathbf{W}_i^Q, \mathbf{W}_i^K \in \mathbb{R}^{D \times d_k}$ y $\mathbf{W}_i^V \in \mathbb{R}^{D \times d_v}$ son matrices de pesos entrenables específicas para el cabezal i. Aquí, d_k representa la dimensión del subespacio donde se calcula la similitud (o alineación) entre tokens.

2.2.2. Atención de Producto Punto Escalada

El núcleo del mecanismo es la función de atención, que mapea una consulta y un conjunto de pares clave-valor a una salida. La operación se formula como:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}\,. \tag{2.2}$$

Interpretación Geométrica y Estadística

El término $\mathbf{Q}\mathbf{K}^\top$ computa el producto punto entre cada vector fila de $\mathbf{Q}$ (representando el token i) y cada vector fila de $\mathbf{K}$ (token j). Geométricamente, el producto punto $\mathbf{q}_i \cdot \mathbf{k}_j$ funciona como un *score* de similitud no normalizado.

El factor de escalado $\frac{1}{\sqrt{d_k}}$ surge de una consideración estadística crítica. Como señalan Vaswani et al. (2017), asumiendo que las componentes de $\mathbf{Q}$ y $\mathbf{K}$ son variables aleatorias independientes con media 0 y varianza 1, su pro-

ducto punto $\sum_{i=1}^{d_k} q_i k_i$ tendría media 0 y varianza d_k. Sin el escalado, para valores grandes de d_k, los productos punto pueden crecer significativamente en magnitud, empujando a la función *softmax* hacia regiones de gradiente extremadamente pequeño (problema de *vanishing gradients*). Dividir por $\sqrt{d_k}$ normaliza la varianza del producto punto a 1, estabilizando el entrenamiento.

La función softmax$(\cdot)$ se aplica a lo largo de la última dimensión, transformando los puntajes en una distribución de probabilidad. Finalmente, la multiplicación por $\mathbf{V}$ calcula la esperanza matemática de los vectores de valor, ponderada por estas probabilidades de atención.

2.2.3. Optimización de Parámetros

Las matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$, junto con las matrices de proyección de salida $\mathbf{W}^O$ y los pesos de las redes *feed-forward*, constituyen el conjunto de parámetros Θ del modelo. El entrenamiento se realiza mediante aprendizaje supervisado (o auto-supervisado en el caso de BERT/DNABERT), minimizando una función de costo $\mathcal{L}(\Theta)$. Esta función de costo es la *entropía cruzada* (*Cross-Entropy*) entre la distribución predicha y la real.

La actualización de los pesos se realiza mediante el algoritmo de *Backpropagation* y el optimizador Adam (Vaswani et al., 2017) (o sus variantes, como AdamW). Para una matriz de pesos genérica $\mathbf{W} \in \Theta$, la actualización se basa en el cálculo del gradiente de la pérdida respecto a los pesos en la iteración t:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_t}. \tag{2.3}$$

El optimizador Adam utiliza los primeros y segundos momentos de estos gradientes para calcular pasos de actualización adaptativos. Debido a que todas las operaciones en la Ecuación 2.2 (multiplicación matricial, escalado, exponencial en softmax) son diferenciables, los gradientes $\nabla_\Theta \mathcal{L}$ fluyen desde la salida hasta las matrices de proyección iniciales, permitiendo que el modelo aprenda qué características del espacio de entrada deben proyectarse para maximizar la atención en contextos relevantes.

2.2.4. Atención Multi-cabezal (Multi-Head Attention)

Para permitir que el modelo atienda conjuntamente a información de diferentes subespacios de representación en diferentes posiciones, se emplea la

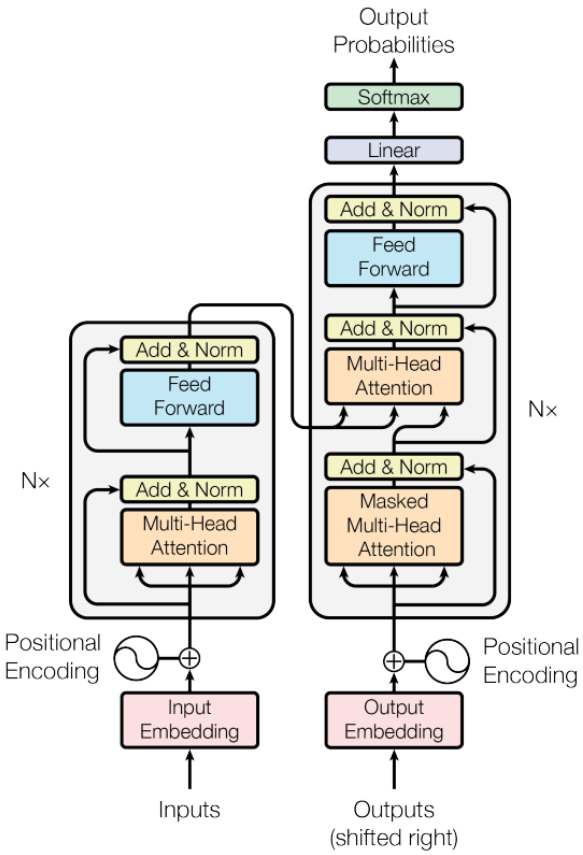


Figura 2.1: Arquitectura del modelo Transformer (Vaswani et al., 2017). Se observa el flujo de tensores a través de los bloques de atención y las redes *feed-forward*. Las conexiones residuales (Add & Norm) evitan el desvanecimiento del gradiente en redes profundas.

atención multi-cabezal. Si definimos H como el número de cabezales, la operación se define formalmente como:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}^O \\ \text{head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)\,. \end{aligned} \tag{2.4}$$

Aquí, las proyecciones son específicas para cada cabezal: $\mathbf{W}_i^Q, \mathbf{W}_i^K \in \mathbb{R}^{D \times d_k}$ y $\mathbf{W}_i^V \in \mathbb{R}^{D \times d_v}$. $\mathbf{W}^O \in \mathbb{R}^{Hd_v \times D}$ es una proyección lineal final que mezcla las representaciones de los diferentes cabezales para recuperar la dimensión original del modelo.

2.2.5. Redes Feed-Forward y Normalización

Como se muestra en la Figura 2.1, cada capa de atención es seguida por una red neuronal *feed-forward* (FFN) aplicada independientemente a cada posición. La FFN tiene la siguiente estructura:

$$\text{FFN}(\mathbf{x}) = \text{máx}(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\,. \tag{2.5}$$

La función $\text{máx}(0, \cdot)$ corresponde a la activación *Rectified Linear Unit* (ReLU). Además, se aplica normalización de capa (*Layer Normalization*) y conexiones residuales alrededor de cada sub-capa. Si denotamos a una sub-capa (ej. Atención o FFN) como $\mathcal{F}$, la salida es $\text{LayerNorm}(x + \mathcal{F}(x))$. Esto facilita la optimización al mantener la media y varianza de las activaciones estables a través de la profundidad de la red.

2.2.6. Limitaciones computacionales de la atención

El mecanismo de atención presenta una complejidad computacional y de memoria $\mathcal{O}(L^2)$ respecto a la longitud de la secuencia L, debido a la necesidad de calcular y almacenar la matriz de scores $\mathbf{Q}\mathbf{K}^\top$ de dimensión $L \times L$. Optimizaciones como *FlashAttention* (Dao et al., 2022) abordan este problema mediante técnicas de *tiling* que evitan materializar la matriz completa en memoria, procesando bloques y aprovechando la jerarquía de memoria de las GPUs. Si bien estas optimizaciones reducen significativamente el uso de memoria y aceleran el cómputo, la complejidad cuadrática permanece, motivando el desarrollo de arquitecturas alternativas como las presentadas en secciones posteriores.

2.3. Fundamentos de operaciones sobre secuencias

Esta sección presenta las operaciones fundamentales empleadas en arquitecturas de procesamiento de secuencias. Estas definiciones matemáticas son necesarias para comprender tanto las estrategias de contexto largo discutidas posteriormente como los operadores subcuadráticos que constituyen el núcleo de esta tesis.

2.3.1. Convolución discreta estándar

La convolución es una operación esencial en el procesamiento de señales discretas y el análisis de sistemas. Matemáticamente, la convolución discreta entre dos secuencias x y h de soporte infinito se define como:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]\,. \tag{2.6}$$

Para trabajar con secuencias de longitud finita, se reformula de la siguiente manera: Sea $\mathbf{x} \in \mathbb{R}^L$ la secuencia de entrada de longitud L y $\mathbf{h} \in \mathbb{R}^K$ el filtro (o kernel) de longitud K. La convolución discreta finita se puede expresar como:

$$y[n] = \sum_{k=0}^{K-1} h[k]x[n-k]\,, \tag{2.7}$$

donde n indexa la posición en la secuencia de salida. El filtro $\mathbf{h}$ actúa como un detector de patrones local que se desliza sobre la entrada. La región de la entrada que afecta a un elemento de salida se conoce como *campo receptivo* (*receptive field*) (Goodfellow et al., 2016).

2.3.2. Convolución dilatada

Para aumentar el campo receptivo sin incrementar el costo computacional, se utiliza la *convolución dilatada* (Yu and Koltun, 2016). Esta técnica permite integrar información de rangos extendidos en la secuencia, lo cual es fundamental para modelar interacciones de largo alcance en genómica. Fue una técnica clave en modelos predecesores importantes como Basenji2 (Kelley, 2020).

La convolución dilatada introduce un parámetro de *dilatación* d que define el espaciado entre los puntos del kernel aplicados a la entrada. Matemáticamente, para una entrada $\mathbf{x}$ y un filtro $\mathbf{h}$ de longitud K, la operación se define como:

$$y[n] = \sum_{k=0}^{K-1} h[k]x[n-d\,k]\,. \tag{2.8}$$

Una dilatación $d = 1$ corresponde a la convolución estándar (Ecuación 2.7). Al incrementar d, el filtro cubre un área mayor de la entrada sin aumentar el número de parámetros. Una propiedad clave es que, al apilar capas con factores de dilatación crecientes (por ejemplo, incrementando exponencialmente

en cada capa), el campo receptivo efectivo crece con la profundidad de la red (van den Oord et al., 2016).

Sin embargo, esta estrategia mostró limitaciones en la práctica para el modelado genómico. Los modelos basados exclusivamente en convoluciones dilatadas, como Basenji2, lograban considerar elementos de secuencia solo hasta ~ 20 kb de distancia. Esta restricción llevó al desarrollo de arquitecturas híbridas como Enformer (Avsec et al., 2021), que combinan "torres" convolucionales con módulos de autoatención para alcanzar campos receptivos de hasta ~ 100 kb.

2.3.3. Convolución causal

En el contexto de modelos autoregresivos para secuencias, la *causalidad* es una propiedad fundamental que garantiza que la salida en una posición k dependa únicamente de las entradas en posiciones anteriores o iguales a k, nunca de posiciones futuras (van den Oord et al., 2016). Esta restricción es esencial para el entrenamiento de modelos de lenguaje, donde se busca predecir el siguiente token basándose exclusivamente en el contexto previo.

Una *convolución causal* es aquella donde el filtro $\mathbf{h}$ satisface:

$$h_k = 0 \quad \text{para todo } k < 0\,. \tag{2.9}$$

Bajo esta condición, la operación de convolución (Ecuación 2.7) se reduce a:

$$y[n] = \sum_{k=0}^{\min(n,K-1)} h[k]\, x[n-k]\,, \tag{2.10}$$

donde la suma incluye únicamente términos donde $n - k \geq 0$, asegurando que $y[n]$ dependa solo de $\{x[0], x[1], \ldots, x[n]\}$.

Desde la perspectiva matricial, la matriz de Toeplitz S_h correspondiente a un filtro causal es *triangular inferior*:

$$\mathsf{S}_h = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{L-1} & h_{L-2} & h_{L-3} & \cdots & h_0 \end{bmatrix}. \tag{2.11}$$

Esta estructura triangular es análoga a la máscara causal utilizada en los mecanismos de atención de Transformers autoregresivos (Vaswani et al., 2017), donde la matriz de atención se enmascara para ser triangular inferior, impidiendo que cada posición atienda a posiciones futuras.

2.3.4. Submuestreo (pooling)

El *submuestreo* (*pooling*) es una operación complementaria a las convoluciones que reduce la dimensionalidad espacial de una secuencia mediante funciones de agregación aplicadas a ventanas locales (LeCun et al., 1998; Krizhevsky et al., 2012). A diferencia de la convolución dilatada que mantiene la longitud de la secuencia, el *pooling* produce una salida de dimensionalidad reducida.

Para una secuencia $\mathbf{x} \in \mathbb{R}^L$, el *pooling* con tamaño de ventana K y paso s genera una salida $\mathbf{y}$ de longitud $L' = \lfloor (L-K)/s \rfloor + 1$. Las dos variantes más comunes son:

Max Pooling:

$$y[n] = \max\{x[i] : ns \leq i < ns + K\}\,. \tag{2.12}$$

Average Pooling:

$$y[n] = \frac{1}{K} \sum_{i=ns}^{ns+K-1} x[i]\,. \tag{2.13}$$

Esta reducción de resolución resulta problemática en genómica. Mientras que en visión computacional los píxeles vecinos suelen codificar información redundante, en secuencias de ADN cada nucleótido puede ser discriminativo. Un SNP individual puede perderse si cae en una ventana de *max pooling* donde valores vecinos dominan, o diluirse mediante *average pooling*. Modelos como Basenji (Kelley et al., 2018) utilizaron *pooling* para escalar a regiones de $\sim$ 20 kb, pero inevitablemente sacrificaron la resolución necesaria para detectar variantes puntuales críticas para aplicaciones como la inferencia de ancestría local (Maples et al., 2013).

2.3.5. Compuertas multiplicativas elemento a elemento

El *element-wise multiplicative gating* (compuerta multiplicativa elemento a elemento) es un mecanismo que permite modular el flujo de información mediante productos elemento a elemento entre señales (Dauphin et al., 2017).

A diferencia de los mecanismos de suma utilizados en conexiones residuales, las compuertas multiplicativas proporcionan un control directo sobre la amplitud y polaridad de la información que se propaga a través de la red.

Siguiendo la convención de Poli et al. (2023), consideramos el caso de un solo canal ($D = 1$) para claridad de exposición. Dada una señal $\mathbf{z} \in \mathbb{R}^L$ y una compuerta $\mathbf{x} \in \mathbb{R}^L$, la operación de compuerta multiplicativa se define como:

$$y_t = x_t z_t, \quad t = 1, \ldots, L\,. \tag{2.14}$$

La compuerta $\mathbf{x}$ se genera como una proyección lineal de la entrada $\mathbf{u}$:

$$\mathbf{x} = \mathbf{u}\mathbf{W}\,, \tag{2.15}$$

donde $\mathbf{W} \in \mathbb{R}^{D\times D}$ es una matriz de pesos aprendible.[1] La señal $\mathbf{z}$ puede provenir de diversas fuentes según la arquitectura: otra proyección de la entrada, el resultado de una convolución, o la salida de una capa anterior.

Esta operación admite una representación matricial compacta, ilustrada en la Figura 2.2. Definiendo la matriz diagonal $\mathsf{D}_x = \mathrm{diag}(\mathbf{x}) \in \mathbb{R}^{L\times L}$, cuya diagonal principal contiene los elementos del vector $\mathbf{x}$, la compuerta multiplicativa se expresa como:

$$\mathbf{y} = \mathsf{D}_x \mathbf{z}\,. \tag{2.16}$$

El comportamiento de la compuerta depende de cómo se genere $\mathbf{x}$. En las *Gated Linear Units* (GLU) de Dauphin et al. (2017), la compuerta pasa por una función sigmoide que restringe sus valores al intervalo $(0, 1)$, limitando la operación a atenuación o supresión de la señal. Sin embargo, cuando no se aplican funciones de activación que restrinjan el rango, como en arquitecturas más recientes (Poli et al., 2023), las compuertas pueden además amplificar ($|x_t| > 1$) o invertir la polaridad ($x_t < 0$) de la señal, proporcionando mayor flexibilidad.

Las compuertas multiplicativas presentan características que las hacen especialmente útiles para el procesamiento de secuencias: dependencia contextual, al generarse dinámicamente a partir de la entrada; eficiencia computacional, requiriendo únicamente $\mathcal{O}(LD)$ operaciones elemento a elemento; y preservación estructural de las dimensiones, facilitando su integración en ar-

[1] En la implementación, $\mathbf{u}$ y $\mathbf{x}$ son matrices en $\mathbb{R}^{L\times D}$ y la operación se aplica independientemente por posición.

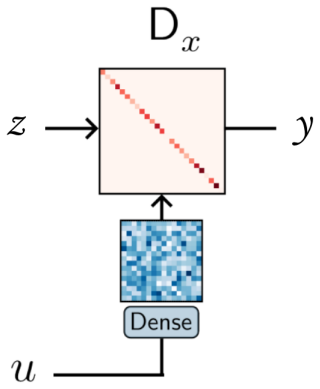


Figura 2.2: Representación de la compuerta multiplicativa. La entrada **u** se transforma mediante una capa densa (proyección lineal) para generar los elementos de la matriz diagonal D_x. La señal **z** se multiplica por esta matriz diagonal, produciendo la salida $\mathbf{y} = \mathsf{D}_x\mathbf{z}$. Adaptado de Poli et al. (2023).

quitecturas profundas.

2.4. Estrategias para contexto largo en genómica

Dos estrategias principales han intentado sortear la restricción cuadrática de los Transformers en función de la longitud de la secuencia para el modelado genómico.

2.4.1. Tokenización basada en k-mers

Esta estrategia, empleada en arquitecturas tipo BERT enmascaradas como DNABERT (Ji et al., 2021), agrupa k nucleótidos consecutivos en un solo token (k-mer). Si bien este enfoque reduce linealmente el número de tokens y, por ende, la complejidad cuadrática, compromete la resolución fina. La pérdida de sensibilidad a variaciones puntuales como los SNPs es una limitación crítica, ya que estos polimorfismos son fundamentales para muchas aplicaciones genómicas, incluyendo la detección de ancestría (The 1000 Genomes Project Consortium, 2015; The International HapMap Consortium, 2005).

2.4.2. Uso de Convoluciones para contexto largo

La segunda estrategia aprovecha las propiedades de escalabilidad de las convoluciones para procesar secuencias largas de manera eficiente. Los dos enfoques principales empleados en genómica son las convoluciones dilatadas (Sección 2.3.2) y el submuestreo mediante *pooling* (Sección 2.3.4) (Avsec et al., 2021; Kelley et al., 2018).

Estas técnicas presentan un compromiso crítico para aplicaciones genómicas: las convoluciones dilatadas permiten expandir el campo receptivo manteniendo resolución de nucleótido individual, alcanzando ventanas de contexto de $\sim$ 20 kb en modelos como Basenji2 (Kelley, 2020). En contraste, el *pooling* reduce la complejidad computacional pero sacrifica irreversiblemente la resolución fina, diluyendo la información de nucleótidos individuales. Esta pérdida es particularmente problemática para detectar SNPs y sus patrones de ocurrencia, limitando su aplicabilidad en tareas de genómica fina como la inferencia de ancestría local (Maples et al., 2013; Patterson et al., 2006, 2012).

La Tabla 2.1 resume las características principales de los modelos genómicos más relevantes para el análisis de ancestría y secuencias. HyenaDNA destaca por su escalabilidad subcuadrática, que permite contextos hasta 1000$\times$ más largos que los modelos basados en Transformers, manteniendo resolución de nucleótido individual.

Tabla 2.1: Comparación de modelos genómicos para análisis de ancestría y secuencias

Modelo	**Arquitectura**	**Complejidad**	**Contexto**	**Resolución**
DNABERT	Transformer	$\mathcal{O}(L^2)$	512 tokens	k-mer (6 bp)
Enformer	CNN + Transf.	$\mathcal{O}(L^2)$	$\sim$200 kb	Binned (128 bp)
Gnomix	XGBoost	$\mathcal{O}(N \log N)$	$\sim$4587 SNPs	SNP individual
HyenaDNA	Hyena	$\mathcal{O}(L \log L)$	1M bp	Nucleótido ind.

2.4.3. Conclusión sobre Limitaciones

Ambos enfoques (tokenización por k-mers y convoluciones con submuestreo) comprometen la resolución fina necesaria para detectar *polimorfismos de un solo nucleótido* (SNPs). Este desafío motivó el desarrollo de modelos que buscan mantener la precisión a nivel de nucleótido mientras escalan el contexto, como se discute en las secciones posteriores.

2.5. Convoluciones de largo alcance y evaluación eficiente

Los modelos convolucionales de largo alcance representan una alternativa prometedora a los Transformers para el procesamiento de secuencias largas. La clave de su eficiencia radica en la evaluación mediante la Transformada Rápida de Fourier y la evolución desde convoluciones explícitas tradicionales hacia formulaciones implícitas optimizadas. Esta sección presenta estas técnicas que constituyen el fundamento de los operadores subcuadráticos empleados en esta tesis.

2.5.1. Transformada de Fourier y el Teorema de Convolución

Las operaciones de convolución presentadas en la Sección 2.3, aunque fundamentales para el procesamiento de secuencias, presentan un desafío computacional significativo cuando se aplican a filtros de longitud comparable a la secuencia de entrada. La evaluación directa de una convolución entre una secuencia de longitud L y un filtro de longitud K requiere $\mathcal{O}(LK)$ operaciones. Para filtros largos donde $K \approx L$, esto resulta en complejidad cuadrática $\mathcal{O}(L^2)$, lo cual dificulta el procesamiento de secuencias largas como son las genómicas.

La Transformada de Fourier proporciona un marco matemático que permite evaluar convoluciones de manera eficiente mediante el teorema de convolución, reduciendo la complejidad computacional a $\mathcal{O}(L \log L)$. Esta reducción de complejidad es fundamental para la viabilidad práctica de arquitecturas como *Hyena* que emplean filtros convolucionales de longitud extendida.

Transformada Discreta de Fourier

La Transformada Discreta de Fourier (DFT, por sus siglas en inglés) es una operación que descompone una secuencia discreta finita en sus componentes de frecuencia (Oppenheim et al., 1999). Para una secuencia $\mathbf{x} \in \mathbb{C}^L$ de longitud L, la DFT se define como:

$$X[\omega] = \sum_{n=0}^{L-1} x[n] e^{-2\pi i \omega n / L}, \quad \omega = 0, 1, \ldots, L-1\,, \tag{2.17}$$

donde $i = \sqrt{-1}$ es la unidad imaginaria, ω es el índice de frecuencia, y $X[\omega] \in \mathbb{C}$ representa la amplitud compleja de la componente de frecuencia ω. La exponencial compleja $e^{-2\pi i \omega n/L}$ describe sinusoides de diferentes frecuencias que forman una base ortogonal del espacio de señales discretas.

La DFT mapea la señal desde el *dominio temporal* (o espacial, en el caso de secuencias de ADN) al *dominio de frecuencia*. En el dominio de frecuencia, cada coeficiente $X[\omega]$ cuantifica la contribución de una frecuencia específica, ω, a la señal original. Esta representación alternativa resulta particularmente útil para ciertas operaciones, como la convolución, que se simplifican significativamente en este dominio.

Transformada Rápida de Fourier

El cálculo directo de la DFT según la Ecuación 2.17 requiere $\mathcal{O}(L^2)$ operaciones, ya que cada uno de los L coeficientes de frecuencia $X[\omega]$ involucra una suma sobre los L elementos de la secuencia original. La Transformada Rápida de Fourier (FFT, *Fast Fourier Transform*) (Cooley and Tukey, 1965) es una familia de algoritmos que calculan la DFT con complejidad $\mathcal{O}(L \log L)$.

El algoritmo más conocido es el de Cooley-Tukey, basado en una estrategia de *divide-and-conquer* que explota la simetría y periodicidad de las exponenciales complejas. La idea fundamental consiste en descomponer recursivamente el problema de tamaño L en subproblemas más pequeños. Para longitudes L que son potencias de 2, el algoritmo subdivide la DFT en dos DFTs de tamaño $L/2$ (índices pares e impares), aplica recursivamente la FFT a cada mitad, y combina los resultados.

Esta reducción de complejidad tiene implicaciones prácticas profundas: para una secuencia de $L = 2^{20} \approx 10^6$ elementos (tamaño típico en genómica), la evaluación directa requeriría aproximadamente 10^{12} operaciones, mientras que la FFT requiere solo $\sim 2 \times 10^7$ operaciones, una mejora de cinco órdenes de magnitud.

Transformada Inversa de Fourier

La Transformada Inversa Discreta de Fourier (IDFT o IFFT cuando se calcula eficientemente) recupera la señal temporal original a partir de su representación en frecuencia:

$$x[n] = \frac{1}{L} \sum_{\omega=0}^{L-1} X[\omega] e^{2\pi i \omega n / L}, \quad n = 0, 1, \ldots, L-1 \,. \tag{2.18}$$

La única diferencia con la DFT (Ecuación 2.17) radica en el signo del exponente y el factor de normalización $1/L$. Esta operación es igualmente eficiente cuando se implementa mediante FFT, manteniendo complejidad $\mathcal{O}(L \log L)$.

Una propiedad fundamental es la reversibilidad perfecta: aplicar la IFFT a la salida de la FFT recupera exactamente la señal original, es decir, $\text{IFFT}(\text{FFT}(x)) = x$, salvo errores numéricos en la implementación computacional.

Teorema de Convolución

El teorema de convolución (Oppenheim et al., 1999; Brigham, 1988) establece una equivalencia fundamental entre operaciones en el dominio temporal y el dominio de frecuencia. Para dos secuencias $\mathbf{x}, \mathbf{h} \in \mathbb{C}^L$, el teorema afirma que:

$$\mathcal{F}\{\mathbf{x} * \mathbf{h}\} = \mathcal{F}\{\mathbf{x}\} \odot \mathcal{F}\{\mathbf{h}\} \,, \tag{2.19}$$

donde $\mathcal{F}$ denota la Transformada de Fourier, $*$ representa la convolución (Ecuación 2.7), y $\odot$ denota la multiplicación elemento a elemento (producto de Hadamard).

La intuición detrás de este teorema radica en que las convoluciones implementan operaciones de filtrado lineal que modifican selectivamente las componentes de frecuencia de una señal. Mientras que en el dominio temporal esta operación requiere combinar múltiples valores desplazados (suma ponderada), en el dominio de frecuencia se reduce a una simple modulación independiente de cada componente de frecuencia.

La ventaja computacional es directa: la multiplicación elemento a elemento $\odot$ requiere $\mathcal{O}(L)$ operaciones, mientras que la convolución directa requiere $\mathcal{O}(L\,K)$ operaciones donde K es la longitud del filtro. Para filtros largos donde $K \approx L$, esto representa una diferencia de $\mathcal{O}(L^2)$ versus $\mathcal{O}(L)$.

Consecuentemente, cualquier convolución puede evaluarse mediante:

$$\mathbf{y} = \mathbf{x} * \mathbf{h} = \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{x}\} \odot \mathcal{F}\{\mathbf{h}\}\} \,, \tag{2.20}$$

o, en notación algorítmica:

$$\mathbf{y} = \text{IFFT}\big(\text{FFT}(\mathbf{x}) \odot \text{FFT}(\mathbf{h})\big). \tag{2.21}$$

El costo computacional total de este enfoque es $2 \times \mathcal{O}(L \log L)$ para las dos FFTs, $\mathcal{O}(L)$ para la multiplicación elemento a elemento, y $\mathcal{O}(L \log L)$ para la IFFT, resultando en complejidad dominante $\mathcal{O}(L \log L)$.

Convolución Circular y Zero-Padding

Un aspecto técnico importante es que la FFT, por su naturaleza basada en funciones periódicas, implementa naturalmente la *convolución circular* en lugar de la convolución lineal estándar definida en la Ecuación 2.7 (Oppenheim et al., 1999; Brigham, 1988). En la convolución circular, la secuencia se trata como si sus extremos estuvieran conectados en un anillo, de modo que los índices "envuelven" (*wrap around*) cuando exceden la longitud de la secuencia.

Para secuencias finitas donde se desea la convolución lineal estándar, este comportamiento circular produce resultados incorrectos debido al aliasing temporal. La solución consiste en aplicar *zero-padding*: extender tanto la señal de entrada $\mathbf{x}$ como el filtro $\mathbf{h}$ con ceros hasta una longitud de al menos $L_{\text{pad}} = L + K - 1$, donde L y K son las longitudes originales de $\mathbf{x}$ y $\mathbf{h}$ respectivamente.

En la práctica, para filtros largos donde $K \approx L$, se utiliza típicamente $L_{\text{pad}} = 2L$, lo que garantiza que los efectos de convolución circular no contaminen los primeros L elementos del resultado. Tras evaluar la convolución mediante FFT sobre las secuencias extendidas, se descartan los últimos L elementos, conservando únicamente los primeros L que corresponden a la convolución lineal deseada.

Aplicación a Convoluciones Largas en Genómica

La evaluación eficiente de convoluciones mediante FFT es fundamental para el modelado de secuencias genómicas largas. El proceso involucra extender ambas secuencias mediante zero-padding, calcular las FFTs de ambas, multiplicar elemento a elemento en el dominio de frecuencia, y aplicar la transformada inversa. La complejidad total es $\mathcal{O}(L \log L)$, drásticamente inferior a la evaluación directa $\mathcal{O}(L^2)$ para filtros largos. El algoritmo completo con análisis

detallado de complejidad se presenta en la subsección siguiente (2.5.2), donde se combina con la generación implícita de filtros.

Esta eficiencia computacional permite aplicar filtros convolucionales de longitud $L \sim 10^5$ a 10^6 en secuencias genómicas, haciendo viable el procesamiento de cromosomas completos o grandes regiones intergénicas.

2.5.2. De convoluciones explícitas a implícitas

En la subsección anterior se presentaron los fundamentos de la Transformada de Fourier y el Teorema de Convolución, que permiten evaluar convoluciones con complejidad $\mathcal{O}(L \log L)$. Esta subsección describe cómo combinar esta evaluación eficiente con parametrizaciones implícitas para crear filtros convolucionales de longitud arbitraria con un número reducido de parámetros entrenables (Poli et al., 2023; Nguyen et al., 2023).

Formulación explícita y sus limitaciones

En la formulación *explícita* tradicional, los coeficientes del filtro convolucional $\mathbf{h} = [h_0, h_1, \ldots, h_{L-1}]$ se almacenan directamente como parámetros entrenables. La operación de convolución se expresa matricialmente mediante la matriz de Toeplitz S_h definida en la Ecuación 2.11:

$$\mathbf{y} = \mathsf{S}_h \mathbf{x}\,, \tag{2.22}$$

donde $\mathbf{x} \in \mathbb{R}^L$ es la secuencia de entrada y $\mathbf{y} \in \mathbb{R}^L$ la salida. Cada elemento y_i se obtiene como la suma ponderada $\sum_{k=0}^{i} h_k\, x_{i-k}$.

Esta formulación presenta una limitación fundamental para secuencias largas: el número de parámetros entrenables crece linealmente con L.

Parametrización implícita mediante redes neuronales

Las *convoluciones implícitas* (Poli et al., 2023) proponen una solución al problema del crecimiento lineal de parámetros respecto al largo de la secuencia: en lugar de almacenar el filtro $\mathbf{h}$ como un vector de L parámetros independientes, se parametriza como la salida de una función continua entrenable. Esta función, denotada f_θ, es típicamente un perceptrón multicapa (MLP) con parámetros θ:

$$h_k = f_\theta(k), \quad k \in \{0, 1, \ldots, L-1\}\,, \tag{2.23}$$

donde $f_\theta : \mathbb{R} \to \mathbb{R}$ es la red neuronal generadora de filtros, k es el índice de posición en la secuencia usado como entrada, y h_k es el valor del filtro en la posición k generado por la red. Los parámetros de la red se denotan $\theta \in \mathbb{R}^M$, donde M representa el número total de pesos y sesgos de la arquitectura. El aspecto crucial es que $M \ll L$: el número de parámetros de la red es independiente de la longitud de la secuencia y generalmente mucho menor.

Como señalan Poli et al. (2023), esta parametrización implícita permite *"desacoplar la memoria de cada filtro del número de parámetros"*, lo cual es fundamental para el modelado de secuencias largas.

Arquitectura de la función generadora

En la práctica, la función generadora f_θ en el operador Hyena tiene una arquitectura específica compuesta por tres componentes (Poli et al., 2023):

$$h_k = \text{Window}(k)(\text{FFN} \circ \text{PositionalEncoding})(k)\,, \tag{2.24}$$

donde $\text{PositionalEncoding} : \mathbb{R} \to \mathbb{R}^{D_e}$ transforma cada índice de posición k en un vector de dimensión D_e, $\text{FFN} : \mathbb{R}^{D_e} \to \mathbb{R}^{ND}$ es una red *feed-forward* que genera los valores del filtro, y $\text{Window} : \mathbb{R} \to \mathbb{R}$ es una función de ventana que modula la amplitud del filtro según la posición. El operador $\circ$ denota la composición de funciones. La FFN utiliza activaciones periódicas (funciones seno) que permiten capturar contenido de alta frecuencia, abordando el sesgo de baja frecuencia inherente a las redes neuronales convencionales. La función *Window* implementa un decaimiento exponencial, lo que sesga los filtros hacia patrones donde las posiciones cercanas tienen mayor influencia que las lejanas. La Figura 2.3 ilustra esta arquitectura, mostrando cómo cada componente transforma progresivamente el índice de posición k hasta generar el filtro $\mathbf{h}$.

Aprendizaje *end-to-end*

Un aspecto central de las convoluciones implícitas es que todos los componentes de f_θ (codificación posicional, FFN y *Window*) se entrenan conjuntamente con el resto del modelo mediante *backpropagation*. Durante el paso *forward*, la red genera el filtro completo evaluando h_k para cada posición

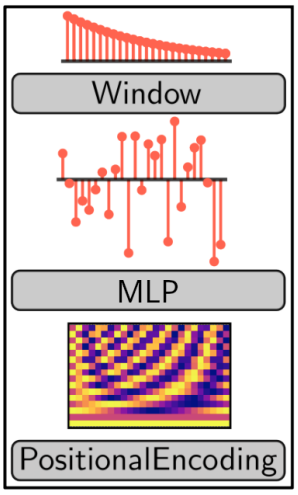


Figura 2.3: Arquitectura de generación de filtros implícitos en Hyena. De abajo hacia arriba: (1) *PositionalEncoding* codifica cada posición k mediante patrones de frecuencia; (2) el MLP transforma estas representaciones en valores intermedios del filtro; (3) la función *Window* aplica un decaimiento exponencial. El filtro resultante $\mathbf{h}$ permite modelar dependencias de largo alcance con $M \ll L$ parámetros. Adaptado de Poli et al. (2023).

$k \in \{0, \ldots, L-1\}$; luego, el filtro se aplica a la secuencia de entrada mediante FFT (Ecuación 2.21). Durante el paso *backward*, el gradiente del error se propaga a través de la convolución y la FFT, actualizando los parámetros θ.

Este proceso permite que f_θ "descubra" automáticamente qué forma de filtro es óptima para la tarea. A medida que el entrenamiento progresa, la red aprende a generar filtros que capturan patrones relevantes, por ejemplo motivos regulatorios en secuencias genómicas (Nguyen et al., 2023).

2.6. El operador *Hyena*

El operador *Hyena* (Poli et al., 2023) constituye un reemplazo subcuadrático del mecanismo de atención, construido alternando convoluciones largas parametrizadas implícitamente (Sección 2.5.2) y compuertas multiplicativas controladas por los datos (Sección 2.3.5). Hyena preserva las propiedades computacionales clave del mecanismo de atención, ademas de presentar un escalamiento sublineal de parámetros respecto a la longitud de secuencia, y la capacidad de modelar dependencias entre cualquier par de posiciones en la entrada. Las convoluciones largas, implementadas mediante matrices de Toeplitz S_h, proporcionan la mezcla de información a través del tiempo, mientras

que las compuertas multiplicativas, representadas como matrices diagonales D_x cuyos valores dependen de la entrada, permiten la selección dinámica de características en cada posición.

2.6.1. Arquitectura y definición formal

La arquitectura de Hyena se fundamenta en la descomposición del mapa de atención densa en una recurrencia de operaciones lineales que desacoplan el modelado de dependencias del costo computacional. Siguiendo la convención de Poli et al. (2023), presentamos las operaciones para un solo canal.[2] Para una secuencia de entrada $\mathbf{u} \in \mathbb{R}^L$, el operador genera primero $N+1$ proyecciones: una señal de valores $\mathbf{v}$ y un conjunto de señales de control o compuertas $\mathbf{x}^1, \ldots, \mathbf{x}^N$. Estas proyecciones se obtienen aplicando una capa densa seguida de una convolución corta, como se detalla más adelante.

Matemáticamente, el operador Hyena puede expresarse como una factorización matricial que alterna entre dos tipos de matrices estructuradas:

$$\mathbf{y} = \mathsf{H}(\mathbf{u})\mathbf{v} = \mathsf{D}_x^N \mathsf{S}_h^N \cdots \mathsf{D}_x^2 \mathsf{S}_h^2 \mathsf{D}_x^1 \mathsf{S}_h^1 \mathbf{v}\,. \tag{2.25}$$

Esta formulación revela la naturaleza dual del procesamiento en Hyena (ver Figura 2.4):

1. **Mezcla de Tiempo Global (S_h):** Las matrices de Toeplitz S_h^n implementan convoluciones causales largas. Estas operaciones permiten que la información de pasos de tiempo anteriores influya en el estado actual, proporcionando el contexto global necesario. Al utilizar el Teorema de la Convolución, estas operaciones se evalúan en $\mathcal{O}(L \log L)$ mediante FFT.
2. **Selección de Información Local (D_x):** Las matrices diagonales $\mathsf{D}_x^n = \mathrm{diag}(\mathbf{x}^n)$ implementan las compuertas multiplicativas elemento a elemento descritas en la Sección 2.3.5. Como se formalizó en la Ecuación 2.16, estas matrices son funciones de la entrada actual $\mathbf{u}$ (proyecciones lineales sin restricción de rango), permitiendo al modelo modular dinámicamente la amplitud de la señal —amplificando, atenuando o invirtiendo— y seleccionar características específicas en cada posición de la secuencia.

[2] En la implementación general, la entrada es una matriz $\mathbf{U} \in \mathbb{R}^{L \times D}$ y las operaciones se aplican independientemente por canal.

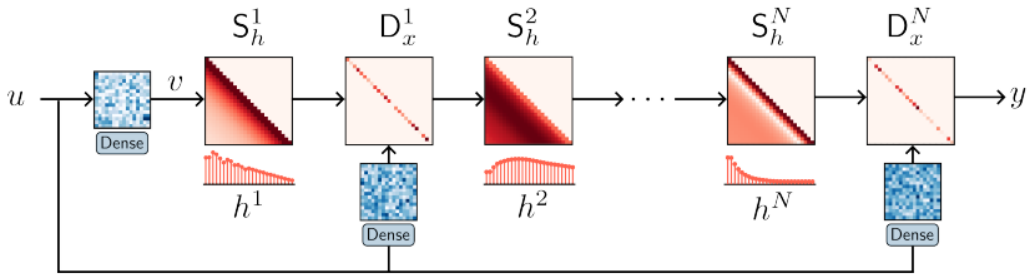


Figura 2.4: Definición del operador Hyena como recurrencia de primitivas subcuadráticas. La entrada se proyecta en valores $\mathbf{v}$ y compuertas $\mathbf{x}$. El procesamiento alterna entre mezcla de tiempo mediante matrices de Toeplitz S_h^n (convoluciones largas) y selección de información mediante matrices diagonales D_x^n (compuertas). Esta estructura logra un costo computacional sublineal en parámetros respecto a la longitud de secuencia. Adaptado de Poli et al. (2023).

2.6.2. Implementación jerárquica y Dualidad de Señales

Operativamente, la recurrencia definida en la Ecuación 2.25 se implementa sin materializar las matrices, utilizando una estructura jerárquica (Poli et al., 2023). Para un operador de orden N, la entrada se proyecta en un vector de valores $\mathbf{v} \in \mathbb{R}^L$ y N vectores de compuertas $\mathbf{x}^n \in \mathbb{R}^L$ para $n = 1, \dots, N$. La salida en cada posición $t \in \{1, \dots, L\}$ de la secuencia se computa como:

$$\begin{aligned} z_t^1 &= v_t \\ z_t^{n+1} &= x_t^n \left(h^n * z^n\right)_t \quad \text{para } n = 1, \dots, N \\ y_t &= z_t^{N+1}, \end{aligned} \tag{2.26}$$

donde $\mathbf{z}^n \in \mathbb{R}^L$ representa el estado intermedio en el nivel n de la recurrencia, $\mathbf{h}^n \in \mathbb{R}^L$ es el filtro de convolución implícito que define la matriz de Toeplitz S_h^n, y $*$ denota la convolución causal.

En la configuración estándar de orden 2 ($N = 2$) utilizada en modelos como HyenaDNA, expandiendo la recurrencia y expresándola en notación vectorial se obtiene:

$$\mathbf{y} = \mathbf{x}^2 \odot \left(\mathbf{h}^2 * \left(\mathbf{x}^1 \odot \left(\mathbf{h}^1 * \mathbf{v}\right)\right)\right), \tag{2.27}$$

donde todas las variables son vectores en $\mathbb{R}^L$.

Interpretación de Procesamiento de Señales: Una propiedad fundamental de esta arquitectura es su comportamiento en el dominio de la frecuencia. Dado que la multiplicación elemento a elemento en el dominio del

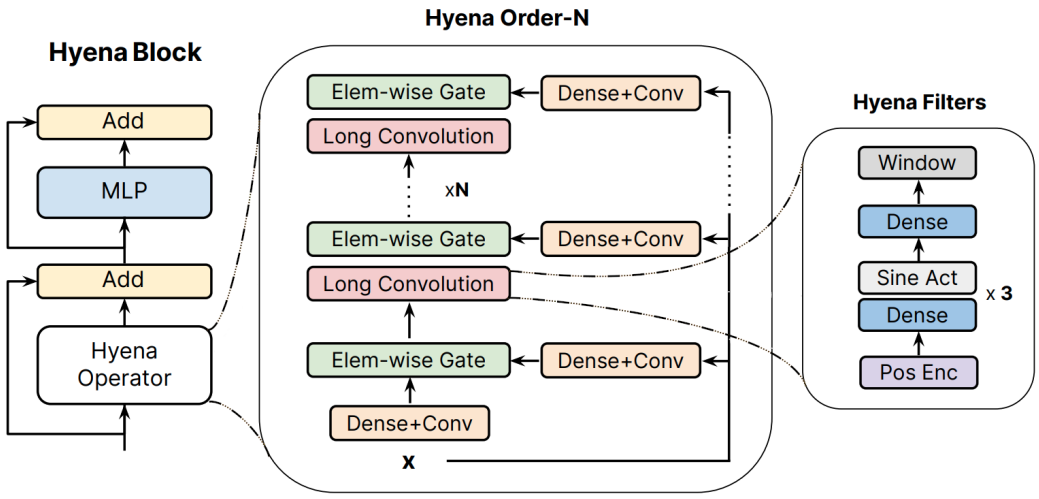


Figura 2.5: Arquitectura del bloque HyenaDNA. Se observa el flujo de las proyecciones a través de las convoluciones implícitas (parametrizadas por una red neuronal auxiliar) y las compuertas multiplicativas. La evaluación mediante FFT garantiza la eficiencia temporal. Adaptado de Nguyen et al. (2023).

tiempo equivale a una convolución en el dominio de la frecuencia (y viceversa, como establece el Teorema de Convolución en la Sección 2.5.1), Hyena alterna efectivamente entre el filtrado en tiempo (vía convoluciones) y la mezcla de frecuencias (vía compuertas multiplicativas). Esta capacidad de *mixing* espectral permite al modelo capturar patrones periódicos y estructurales complejos, esenciales en el análisis de señales biológicas.

La Figura 2.5 ilustra el bloque constructivo completo. Como se mencionó anteriormente, las proyecciones $\mathbf{v}, \mathbf{x}^1, \ldots, \mathbf{x}^N$ se obtienen aplicando primero una capa densa y luego convoluciones cortas (de tamaño 3). Este segundo componente fue introducido en la arquitectura H3 (Dao et al., 2023) para resolver deficiencias en tareas de *associative recall* (recuperación asociativa), donde el modelo debe extraer un valor dado una clave. La capa densa opera de manera independiente por posición, sin mezclar información entre tokens adyacentes. La convolución corta actúa como un operador de desplazamiento que permite al modelo comparar y relacionar información entre posiciones vecinas antes de las compuertas multiplicativas y las convoluciones largas. Esta combinación permite procesar eficientemente tanto patrones de corto como de largo alcance.

En términos de rendimiento, como se muestra en la Figura 2.6, Hyena mantiene una complejidad $\mathcal{O}(L \log L)$, superando a métodos de atención opti-

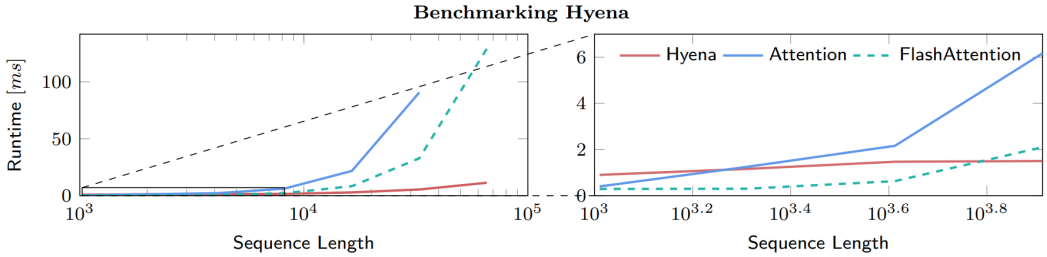


Figura 2.6: Comparativa de complejidad computacional. Hyena (rojo) exhibe un escalamiento subcuadrático, permitiendo contextos de millones de tokens donde la atención estándar (azul) y FlashAttention (verde) se vuelven computacionalmente inviables. Adaptado de Poli et al. (2023).

mizada como *FlashAttention* (Sección 2.2.6), que si bien reducen significativamente el uso de memoria y aceleran el cómputo, mantienen una complejidad cuadrática $\mathcal{O}(L^2)$. La diferencia se vuelve evidente para secuencias largas: para $L = 10^5$ tokens, Hyena muestra un tiempo de ejecución considerablemente menor que los métodos basados en atención, lo que habilita su uso para secuencias genómicas extremadamente largas.

2.7. HyenaDNA: modelado genómico subcuadrático

HyenaDNA (Nguyen et al., 2023) adapta el operador Hyena al dominio de la genómica, abordando el desafío de modelar interacciones de largo alcance inherentes al ADN, como la relación entre *enhancers* (secuencias regulatorias que potencian la transcripción génica) y promotores (regiones del ADN donde se inicia la transcripción) que pueden estar separados por cientos de miles de pares de bases.

La arquitectura aprovecha la eficiencia del operador Hyena para entrenar modelos con ventanas de contexto de hasta 1 millón de tokens con resolución de un solo nucleótido (single-nucleotide resolution). Esto representa un avance significativo respecto a modelos previos basados en Transformers (como DNABERT o Nucleotide Transformer), que típicamente están limitados a contextos de 512 a 4096 tokens debido a la restricción cuadrática.

El diseño de HyenaDNA introduce varias adaptaciones específicas para el dominio biológico:

1. **Tokenización Minimalista:** Utiliza un vocabulario a nivel de carácter

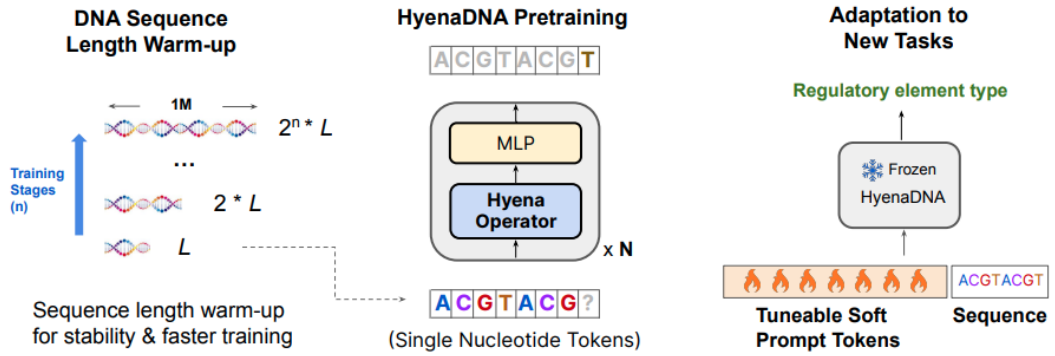


Figura 2.7: Pipeline de HyenaDNA para modelos fundacionales genómicos. (Izquierda) El entrenamiento utiliza un esquema de *warm-up* de longitud de secuencia para estabilizar la convergencia en contextos largos. (Derecha) Capacidad de adaptación a nuevas tareas mediante *In-Context Learning* y *Soft Prompts*, aprovechando el contexto extendido sin necesidad de fine-tuning completo. Adaptado de Nguyen et al. (2023).

(A, C, G, T, N), evitando la tokenización por k-mers. Esto preserva la semántica exacta de las secuencias y permite detectar mutaciones puntuales (SNPs) críticas para estudios de ancestría y patogenicidad.

2. **Escalabilidad de Contexto:** Implementa un currículum de entrenamiento (*Sequence Length Warm-up*) que incrementa progresivamente la longitud de la ventana de contexto durante el entrenamiento, estabilizando la optimización de las convoluciones largas (Figura 2.7, izquierda).
3. **Adaptación eficiente:** Aprovecha la ventana de contexto extendida para dos estrategias de adaptación complementarias. El *In-Context Learning* permite realizar tareas nuevas sin modificar los parámetros del modelo, proporcionando ejemplos demostrativos directamente en el contexto de entrada. Alternativamente, los *Soft Prompts* son vectores continuos entrenables que se concatenan a la entrada; durante la adaptación, solo estos vectores se optimizan mientras los parámetros del modelo permanecen congelados, reduciendo significativamente el costo computacional respecto al *fine-tuning* completo (Figura 2.7, derecha).

El modelo se distribuye en diversas escalas, desde *HyenaDNA-tiny* (1k contexto) hasta *HyenaDNA-xlarge* (1M contexto). Para esta tesis, la relevancia de HyenaDNA radica en su capacidad para procesar regiones cromosómicas enteras, permitiendo al modelo "ver" patrones de ancestría global y local simultáneamente, algo inalcanzable con arquitecturas de ventana corta.

2.8. Representaciones posicionales para genómica

La codificación posicional constituye un componente esencial en arquitecturas que procesan secuencias, ya que permite distinguir elementos que aparecen en distintas ubicaciones dentro de una secuencia. Sin esta información, modelos como *Transformers* (Vaswani et al., 2017) no podrían diferenciar subsecuencias idénticas ubicadas en distintas regiones del genoma, perdiendo la noción de contexto estructural. En este apartado se revisan los principales mecanismos de codificación posicional empleados en el aprendizaje profundo, junto con las adaptaciones necesarias para su aplicación en secuencias genómicas ultralargas.

2.8.1. Codificación posicional sinusoidal

El *positional encoding* original del Transformer (Vaswani et al., 2017) inyecta información posicional mediante funciones sinusoidales fijas que dependen de la posición *pos* y la dimensión i del embedding:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d}}\right), \end{aligned} \tag{2.28}$$

donde d es la dimensionalidad del embedding. Este esquema es determinístico, no requiere parámetros adicionales y permite al modelo aprender relaciones posicionales relativas de manera explícita.

2.8.2. Embeddings posicionales

Una alternativa ampliamente empleada son los *embeddings posicionales* (Gehring et al., 2017), que representan cada posición mediante una fila en una matriz paramétrica $\mathbf{E} \in \mathbb{R}^{P\times d}$, donde P es el número máximo de posiciones posibles. Esta aproximación ofrece máxima *expresividad* —entendida como la capacidad del modelo para representar funciones o patrones complejos— y flexibilidad de adaptación al dominio, pero presenta una limitación crítica en genómica: el tamaño de $\mathbf{E}$ crece linealmente con P. Para un cromosoma humano ($P \approx 2.5 \times 10^8$) y $d = 128$, la matriz completa requiere aproxima-

damente 60 GB de memoria (en `float16`), lo que impide su uso en hardware convencional. Por tanto, aunque expresivos, estos embeddings son inviables para secuencias genómicas completas sin algún tipo de compresión o reducción dimensional.

2.8.3. Embeddings posicionales factorizados (FPE)

Los *factorized positional embeddings* (FPE) abordan el problema de escalabilidad mediante factorización matricial de bajo rango, una técnica inspirada en la factorización de embeddings propuesta en ALBERT (Lan et al., 2019). En lugar de una matriz monolítica $\mathbf{E} \in \mathbb{R}^{P \times d}$, se emplea la descomposición:

$$\mathbf{E} = \mathbf{AB}\,, \tag{2.29}$$

donde $\mathbf{A} \in \mathbb{R}^{P \times k}$ y $\mathbf{B} \in \mathbb{R}^{k \times d}$, con $k \ll d$. Esto permite reducir drásticamente el consumo de memoria manteniendo una diferenciación posicional útil. Por ejemplo, con $k = 16$, $d = 128$ y $P = 2.5 \times 10^8$, la memoria se reduce de ~ 60 GB a ~ 7.5 GB. El valor de k controla el rango efectivo de representación: valores mayores preservan más detalle posicional, mientras que valores pequeños favorecen eficiencia computacional.

2.8.4. Rotary positional embeddings (RoPE)

Los *rotary embeddings* (RoPE) (Su et al., 2024) incorporan la información posicional mediante rotaciones en el espacio de embeddings, codificando la distancia relativa entre tokens en lugar de su posición absoluta. Para una posición m y una dimensión par i:

$$\begin{pmatrix} q_i^{(m)} \\ q_{i+1}^{(m)} \end{pmatrix} = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix} \begin{pmatrix} q_i \\ q_{i+1} \end{pmatrix}, \tag{2.30}$$

donde $\theta_i = 10000^{-2i/d}$. Este enfoque conserva invarianza relativa, generaliza a secuencias más largas y no introduce parámetros adicionales, por lo que resulta especialmente adecuado para modelos genómicos que requieren extrapolación a regiones no vistas durante el entrenamiento.

2.8.5. Embeddings de Fourier

Los *Fourier positional embeddings* (Zheng et al., 2021) representan las posiciones como combinaciones de funciones trigonométricas de múltiples frecuencias:

$$\gamma(p) = \left[\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{F-1}\pi p), \cos(2^{F-1}\pi p)\right], \tag{2.31}$$

donde F controla la resolución espectral. A diferencia de los métodos discretos, esta representación continua permite interpolar posiciones y manejar secuencias de longitud variable, siendo particularmente útil en genómica, donde las posiciones no siempre están uniformemente distribuidas (por ejemplo, SNPs dispersos).

2.8.6. Factores de escala para reducción de memoria

En contextos genómicos, donde P puede alcanzar cientos de millones, se introduce un *factor de escala* α que agrupa posiciones contiguas para reducir el tamaño efectivo del embedding:

$$P_{\text{efectivo}} = \frac{P_{\text{cromosoma}}}{\alpha}. \tag{2.32}$$

Por ejemplo, con $\alpha = 100$, el número de posiciones distintas disminuye de 2.5×10^8 a 2.5×10^6, reduciendo el uso de memoria en un 99 % a costa de una pérdida de resolución local. Factores mayores permiten procesar cromosomas completos en hardware limitado, aunque sacrifican precisión espacial en regiones cercanas.

2.8.7. Representaciones posicionales en HyenaDNA

En los modelos *HyenaDNA*, la codificación posicional no se aplica de forma explícita durante el preentrenamiento, ya que la arquitectura Hyena puede capturar dependencias largas mediante convoluciones jerárquicas eficientes (Nguyen et al., 2023). Sin embargo, para tareas que dependen fuertemente del contexto genómico —como la detección de ancestría— pueden integrarse embeddings posicionales explícitos durante la adaptación o el fine-tuning. Esta flexibilidad arquitectural permite combinar la eficiencia de Hyena con

representaciones posicionales adaptadas a la escala y naturaleza del ADN.

2.9. Métricas de evaluación

La evaluación de modelos en tareas genómicas requiere métricas específicas que permitan cuantificar tanto el rendimiento de clasificación como la calidad del modelado de secuencias. Esta sección presenta las principales métricas y técnicas de interpretabilidad empleadas en los experimentos de esta tesis.

2.9.1. Métricas de clasificación

Para tareas de clasificación como detección de ancestría, se emplean métricas estándar derivadas de la matriz de confusión. Para un problema de clasificación multiclase, definimos para cada clase i los verdaderos positivos (TP_i) como muestras de la clase i correctamente clasificadas, los falsos positivos (FP_i) como muestras de otras clases incorrectamente clasificadas como clase i, y los falsos negativos (FN_i) como muestras de la clase i incorrectamente clasificadas como otras clases.

La *exactitud* mide la proporción de predicciones correctas sobre el total de muestras:

$$\text{Exactitud} = \frac{\sum_{i=1}^{C} \mathrm{TP}_i}{\sum_{i=1}^{C} (\mathrm{TP}_i + \mathrm{FN}_i)}\,, \tag{2.33}$$

donde C es el número de clases. En detección de ancestría con tres poblaciones (Africana, Asiática, Europea), una exactitud de 0.93 indica que el 93 % de las predicciones son correctas.

La *precisión* para la clase i mide la proporción de predicciones correctas entre todas las predicciones para esa clase:

$$\text{Precisión}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i}\,. \tag{2.34}$$

La precisión macro-promediada se calcula como:

$$\text{Precisión}_{\text{macro}} = \frac{1}{C} \sum_{i=1}^{C} \text{Precisión}_i\,. \tag{2.35}$$

El *recall* para la clase i mide la proporción de muestras de esa clase que

fueron correctamente identificadas:

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}\,. \tag{2.36}$$

El recall macro-promediado se calcula como:

$$\text{Recall}_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C} \text{Recall}_i\,. \tag{2.37}$$

El *F1-score* combina precisión y recall en una métrica balanceada mediante la media armónica:

$$\text{F1}_i = 2\,\frac{\text{Precisión}_i \times \text{Recall}_i}{\text{Precisión}_i + \text{Recall}_i}\,. \tag{2.38}$$

El F1-score macro-promediado es:

$$\text{F1}_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C} \text{F1}_i\,. \tag{2.39}$$

Estas métricas son especialmente importantes en conjuntos de datos desbalanceados de ancestría, donde diferentes poblaciones pueden estar representadas en proporciones distintas.

2.9.2. Perplejidad

La perplejidad (Jurafsky and Martin, 2024) es una métrica fundamental para evaluar modelos de lenguaje y predicción de secuencias. Para una secuencia de nucleótidos $\mathbf{x} = (x_1, x_2, \ldots, x_L)$, la perplejidad se define como:

$$\text{Perplejidad}(\mathbf{x}) = \exp\left(-\frac{1}{L}\sum_{t=1}^{L} \log P(x_t|x_1, \ldots, x_{t-1})\right), \tag{2.40}$$

donde $P(x_t|x_1, \ldots, x_{t-1})$ es la probabilidad que asigna el modelo al nucleótido x_t dado el contexto anterior. Alternativamente, puede expresarse en términos de la entropía cruzada promedio:

$$\text{Perplejidad} = \exp(\text{Entropía Cruzada})\,. \tag{2.41}$$

La perplejidad tiene una interpretación intuitiva como el número efectivo de opciones que el modelo considera en cada posición. Una perplejidad igual

a 1 indica un modelo perfecto con predicción determinística correcta. Valores entre 1 y 4 sugieren que el modelo está aprendiendo patrones útiles, mientras que una perplejidad de 4 equivale a predicción aleatoria uniforme entre los 4 nucleótidos (A, C, G, T).

La perplejidad es especialmente valiosa en genómica debido a su sensibilidad a patrones, detectando cuándo el modelo aprende estructuras biológicas reales versus ruido. Además, permite comparar diferentes arquitecturas y técnicas de codificación de manera objetiva, facilitando la selección del enfoque más adecuado. Finalmente, funciona como una herramienta de diagnóstico de entrenamiento, identificando problemas de convergencia o configuraciones subóptimas durante el desarrollo del modelo.

2.9.3. Técnicas de interpretabilidad

La interpretabilidad de modelos genómicos es crucial para validar predicciones y extraer conocimiento biológico significativo. Esta sección describe las principales técnicas empleadas para interpretar el comportamiento de modelos como HyenaDNA.

Los *mapas de saliencia* calculan la importancia de cada nucleótido para la predicción final mediante el gradiente de la función de pérdida respecto a la entrada. Matemáticamente, para una secuencia de entrada $\mathbf{x} = [x_1, x_2, \ldots, x_L]$ y una función del modelo $f(\mathbf{x})$, la saliencia del elemento i se define como:

$$S_i = \left| \frac{\partial f(\mathbf{x})}{\partial x_i} \right|. \tag{2.42}$$

Estos mapas permiten identificar qué regiones genómicas contribuyen más significativamente a las decisiones del modelo, revelando patrones en la secuencia de entrada. La Figura 2.8 muestra un ejemplo de mapa de saliencia para una secuencia clasificada como de ancestría europea.

El *análisis de embeddings y estados ocultos* examina las representaciones internas aprendidas por el modelo mediante técnicas de reducción de dimensionalidad para visualizar la estructura del espacio latente. El análisis de componentes principales (PCA) proporciona una proyección lineal que preserva la máxima varianza en las primeras componentes principales, útil para identificar las direcciones de mayor variabilidad en los embeddings. Por otro lado, t-SNE (t-Distributed Stochastic Neighbor Embedding) es una técnica no lineal que preserva la estructura local del espacio de alta dimensión, efectiva para reve-

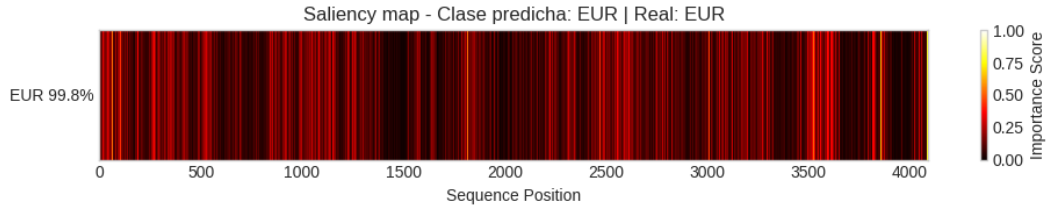


Figura 2.8: Mapa de saliencia para una secuencia genómica clasificada como ancestría europea (EUR) con 99.8 % de confianza. Las barras verticales indican la importancia de cada posición de la secuencia para la predicción del modelo, donde colores más brillantes (amarillo/blanco) corresponden a mayor relevancia. Se observa que ciertas posiciones contribuyen significativamente más que otras, sugiriendo la presencia de variantes informativas para la clasificación de ancestría.

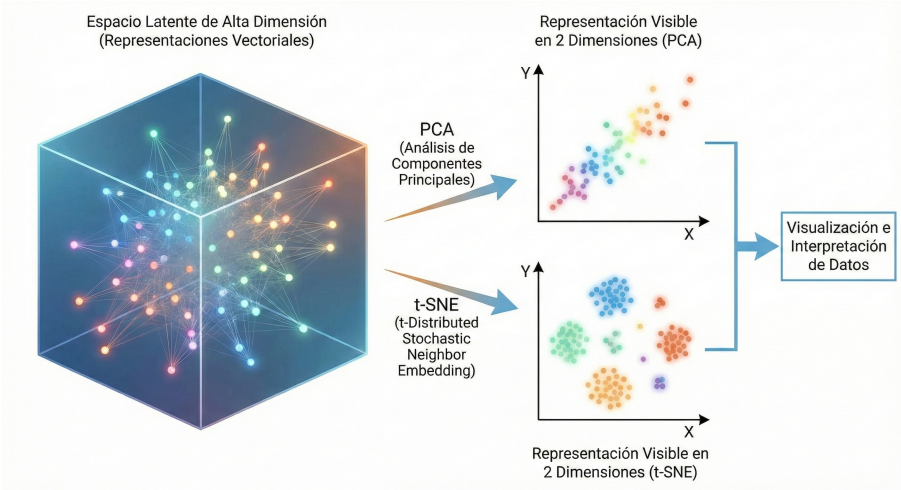


Figura 2.9: Esquema de reducción de dimensionalidad aplicado a representaciones internas del modelo. A partir del espacio latente de alta dimensión (izquierda), se emplean PCA (arriba) y t-SNE (abajo) para obtener proyecciones bidimensionales que permiten visualizar e interpretar la estructura de los datos. PCA preserva las direcciones de máxima varianza global, mientras que t-SNE captura la estructura local, revelando clusters correspondientes a diferentes poblaciones ancestrales.

lar clusters y separaciones entre poblaciones ancestrales. Estas visualizaciones permiten verificar si el modelo ha aprendido representaciones que reflejan agrupamientos biológicamente significativos, como la separación entre poblaciones ancestrales en el espacio de embeddings. La Figura 2.9 ilustra el proceso de reducción de dimensionalidad desde el espacio latente de alta dimensión hacia representaciones bidimensionales mediante PCA y t-SNE.

Los *mapas de ancestría cromosómica* son visualizaciones específicas del dominio genómico que muestran las predicciones de ancestría a lo largo de cada cromosoma. Estos mapas revelan segmentos de ancestría homogénea, donde el

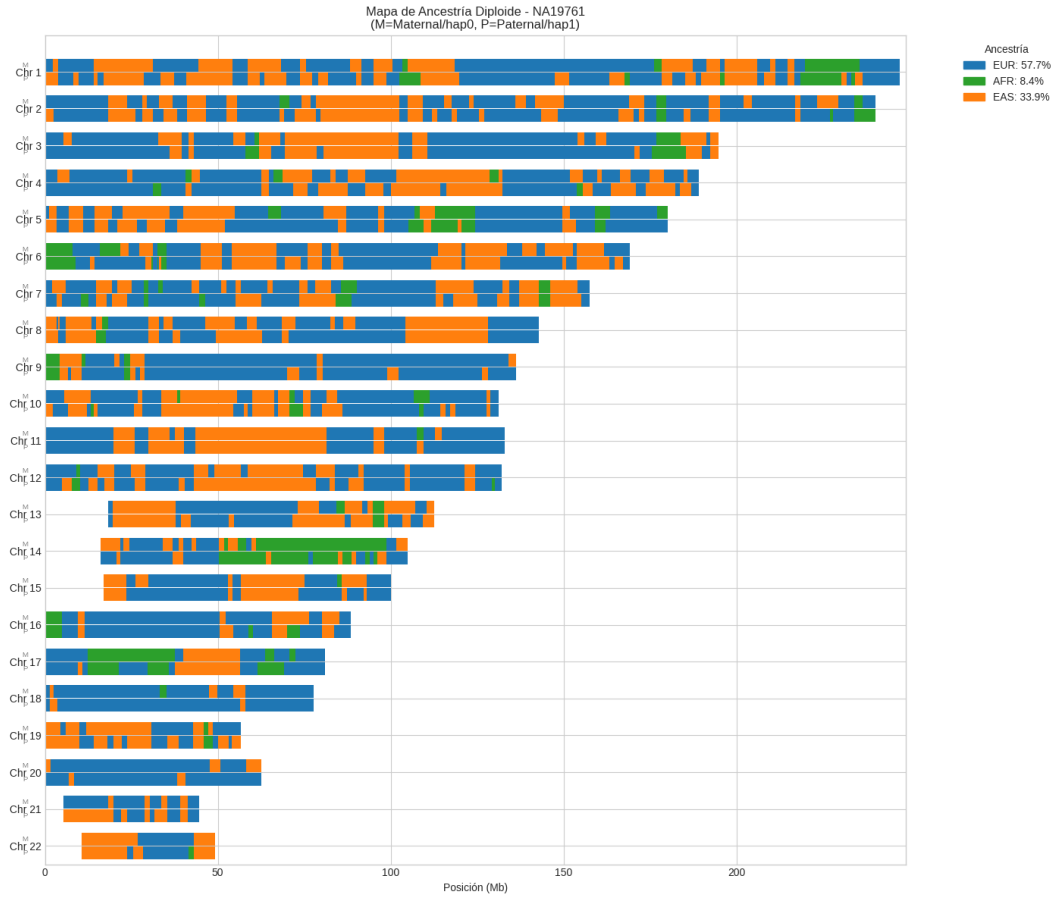


Figura 2.10: Mapa de ancestría diploide para un individuo del conjunto de datos, mostrando los 22 cromosomas autosómicos (ambos haplotipos). Cada cromosoma se representa como una barra horizontal donde pequeñas ventanas genómicas son clasificadas según su ancestría predicha (europea, africana o asiática), revelando la composición ancestral a lo largo del genoma.

modelo predice consistentemente el mismo origen ancestral; puntos de recombinación, correspondientes a transiciones entre diferentes ancestrías que pueden reflejar eventos históricos de mezcla poblacional; y regiones de incertidumbre, zonas donde el modelo muestra baja confianza, potencialmente indicando secuencias ambiguas o datos de entrenamiento insuficientes. Estos mapas proporcionan insights biológicos interpretables sobre patrones de herencia, estructura poblacional y eventos demográficos históricos, conectando las predicciones del modelo con procesos evolutivos conocidos. La Figura 2.10 presenta un ejemplo de este tipo de visualización.

2.10. Síntesis y relevancia para la tesis

Los fundamentos expuestos justifican la adopción de *HyenaDNA* como arquitectura principal para la detección de ancestría genética. Esta elección se fundamenta en tres aspectos críticos que abordan directamente los desafíos del dominio genómico:

- **Eficiencia computacional**: La complejidad subcuadrática $\mathcal{O}(L \log L)$ —frente a $\mathcal{O}(L^2)$ de los Transformers— permite procesar extensas regiones cromosómicas con recursos computacionales limitados, democratizando el acceso a análisis genómicos de gran escala.
- **Preservación de resolución**: Mantiene sensibilidad a nucleótidos individuales en contextos ultralargos (hasta 1 M tokens), característica esencial para detectar patrones sutiles de SNPs distribuidos a lo largo de extensas regiones cromosómicas.
- **Flexibilidad arquitectural**: La disponibilidad de modelos preentrenados escalables, combinada con estrategias especializadas de codificación posicional para genómica, proporciona un marco adaptable para diferentes escenarios experimentales y limitaciones de hardware.

Esta convergencia de capacidades posiciona a *HyenaDNA* como una excelente herramienta para abordar la detección de ancestría genética a resolución de nucleótido individual, especialmente cuando se requiere analizar señales ancestrales dispersas en grandes regiones genómicas donde los métodos tradicionales encuentran limitaciones de escalabilidad.

Capítulo 3

Materiales y métodos

Este capítulo describe la metodología experimental empleada para evaluar la aplicación de *HyenaDNA* en tareas de clasificación genómica y detección de ancestría. La investigación se enmarca dentro de un enfoque experimental y aplicado, con el objetivo de demostrar la efectividad de arquitecturas subcuadráticas en el procesamiento de secuencias genómicas largas y patrones dispersos de SNPs.

La metodología se estructura en cuatro componentes principales: la descripción y procesamiento de los conjuntos de datos utilizados, los enfoques de entrenamiento implementados, la caracterización del entorno computacional, y las técnicas de análisis y evaluación empleadas para interpretar los resultados.

3.1. Conjuntos de datos

Los experimentos realizados en esta tesis emplean dos tipos principales de conjuntos de datos: datasets de clasificación de especies proporcionados por el proyecto *HyenaDNA*, y datos genómicos humanos del proyecto *1000 Genomes* (The 1000 Genomes Project Consortium, 2015) procesados específicamente para tareas de detección de ancestría.

3.1.1. Datasets de clasificación de especies

Para los experimentos de clasificación de especies se utilizan genomas completos de múltiples organismos proporcionados por el proyecto *HyenaDNA*. Los experimentos específicos de esta tesis se enfocan en la clasificación entre cinco especies representativas: humano, lémur, hipopótamo, ratón y cerdo.

Los genomas de cada especie se almacenan en formato FASTA (Pearson and Lipman, 1988), un formato estándar de texto plano para secuencias biológicas donde cada entrada comienza con una línea de encabezado (precedida por el carácter >) seguida de las líneas de la secuencia de nucleótidos (ver Apéndice 1). Los archivos se organizan por cromosoma individual. El procesamiento sigue un pipeline específico para garantizar la calidad y reproducibilidad, utilizando una división predefinida de cromosomas para entrenamiento, validación y test que asegura que no haya solapamiento entre conjuntos de datos.

El muestreo de secuencias para entrenamiento implementa una estrategia de tres niveles diseñada para maximizar la diversidad genómica. Primero se realiza la selección aleatoria de especies asegurando representación balanceada durante el entrenamiento. Dentro de cada especie seleccionada, se realiza la selección aleatoria de cromosomas. Finalmente, se selecciona aleatoriamente una posición de inicio en el cromosoma elegido, seguida de la extracción de una ventana contigua de longitud predefinida (típicamente entre 1k y 32k nucleótidos).

Las características principales de los datasets incluyen ventanas de longitud fija configurables desde 1k hasta 32k nucleótidos, con relleno de nucleótidos "N" cuando la secuencia extraída es más corta que la ventana predefinida (por ejemplo, cerca de los extremos de un cromosoma o en regiones con datos faltantes). La cobertura abarca mamíferos con diferentes distancias evolutivas, proporcionando diversidad taxonómica. La división cromosómica sigue una separación estricta por cromosomas, asignando cromosomas completos a un único conjunto (entrenamiento, validación o test). Esta estrategia es fundamental para evitar fuga de información (*data leakage*): las regiones cercanas dentro de un mismo cromosoma comparten estructura de ligamiento y patrones de variación correlacionados, por lo que incluir regiones del mismo cromosoma en entrenamiento y test podría inflar artificialmente las métricas de rendimiento. Por ejemplo, para el humano se utiliza la división donde los cromosomas 2, 4, 6, 8, 14-22, X e Y se asignan a entrenamiento, los cromosomas 1, 3, 12 y 13 a validación, y los cromosomas 5, 7, 9-11 a test. El preprocesamiento emplea tokenización compatible con modelos pre-entrenados usando codificación por caracteres con un vocabulario de 5 tokens: los cuatro nucleótidos (A, C, G, T) y el carácter especial N para posiciones desconocidas o ambiguas.

3.1.2. Datos del proyecto 1000 Genomes

A diferencia de la clasificación de especies, donde las diferencias genómicas entre organismos son amplias y se distribuyen a lo largo de todo el genoma, la detección de ancestría entre humanos requiere un enfoque diferente. Los humanos comparten aproximadamente el 99,9 % de su secuencia genómica, y solo el ∼0,1 % restante —concentrado en los SNPs— contiene información discriminativa sobre el origen ancestral. Por esta razón, los experimentos de ancestría trabajan directamente con SNPs en lugar de secuencias completas de nucleótidos.

Para estos experimentos se emplean datos genómicos humanos del proyecto *1000 Genomes* (The 1000 Genomes Project Consortium, 2015), que proporcionan información de variación genética de poblaciones humanas globalmente distribuidas. Este conjunto de datos es particularmente valioso para estudios de ancestría debido a su cobertura poblacional diversa.

El procesamiento de los datos sigue un pipeline estructurado. Primero se obtienen archivos VCF (*Variant Call Format*) (Danecek et al., 2011), un formato estándar que registra las variantes genéticas detectadas respecto de un genoma de referencia (ver Apéndice 1). Los archivos VCF contienen diversos tipos de variantes: SNPs, inserciones, deleciones y variantes estructurales. Para los experimentos de esta tesis se filtran únicamente los SNPs, descartando inserciones, deleciones y otros tipos de variantes, dado que los SNPs constituyen los marcadores más informativos para ancestría (Sección 2.1.1). Luego se realiza la selección de poblaciones representativas de tres grupos continentales principales: Europa (británicos de Inglaterra y Escocia, poblaciones ibéricas de España, toscanos de Italia, finlandeses de Finlandia), África (gambianos de la División Occidental, Luhya de Webuye en Kenia, Mende de Sierra Leona), y Asia (Han chinos de Pekín, japoneses de Tokio, Kinh de Ciudad Ho Chi Minh en Vietnam). Finalmente, se realiza la conversión a formato HDF5 (The HDF Group, 2025), un formato binario jerárquico que permite acceder a subconjuntos de datos sin necesidad de cargar el archivo completo en memoria RAM. Esto resulta esencial dado el volumen de datos genómicos involucrados. El formato HDF5 resultante organiza los datos jerárquicamente, facilitando el acceso rápido a subconjuntos específicos de SNPs por cromosoma, posición genómica, o población de origen.

Dado el volumen masivo de datos disponibles (millones de SNPs por indi-

viduo), se implementa una estrategia de muestreo diseñada para crear muestras representativas y balanceadas. El proceso comienza con el sorteo aleatorio del grupo continental (Europa, África o Asia), seguido del sorteo aleatorio de individuos dentro de cada población ancestral, asegurando representación equilibrada entre grupos. Para cada individuo seleccionado, se realiza el sorteo aleatorio del cromosoma de origen, evitando sesgos hacia cromosomas específicos. Luego se determina aleatoriamente la posición inicial en el cromosoma seleccionado, garantizando cobertura uniforme del genoma. Finalmente, se obtiene una ventana contigua de SNPs de longitud predefinida (típicamente entre 1k y 10k SNPs) comenzando en la posición seleccionada. Esta estrategia asegura que cada muestra de entrenamiento contenga información genómica diversa y representativa, minimizando el riesgo de overfitting a regiones genómicas específicas.

Los datos procesados se dividen en conjuntos de entrenamiento, validación y test siguiendo una estrategia estratificada que preserva la distribución ancestral en cada subconjunto. El conjunto de entrenamiento (70 %) se utiliza para el entrenamiento o *fine-tuning* de modelos y optimización de parámetros. El conjunto de validación (15 %) se emplea para monitoreo durante entrenamiento y selección de hiperparámetros. El conjunto de test (15 %) se reserva exclusivamente para evaluación final de rendimiento, nunca utilizado durante el desarrollo del modelo. La división se realiza a nivel de individuo para evitar que datos del mismo individuo aparezcan en múltiples conjuntos, disponibilizando todos los cromosomas en cada uno de los conjuntos.

3.2. Estrategias de entrenamiento

Los experimentos implementan tres estrategias principales de entrenamiento, cada una diseñada para abordar aspectos específicos del modelado genómico y la detección de ancestría.

3.2.1. Fine-tuning de modelos pre-entrenados

Para experimentos que utilizan secuencias completas de ADN, se emplea *fine-tuning* de los modelos *HyenaDNA* pre-entrenados. Esta estrategia aprovecha las representaciones aprendidas durante el pre-entrenamiento en genomas completos, adaptándolas específicamente para tareas de clasificación.

Los pesos del modelo se inicializan con valores pre-entrenados en genomas humanos completos. Se añade una capa MLP de clasificación específica para la tarea, ya sea clasificación de especies o detección de ancestría. La optimización emplea tasas de aprendizaje diferentes para las capas pre-entrenadas (menor tasa) y las nuevas capas de clasificación (mayor tasa), permitiendo una adaptación gradual de las representaciones aprendidas.

3.2.2. Predicción del siguiente token

Para modelado de secuencias de SNPs, dado que los modelos pre-entrenados de *HyenaDNA* fueron entrenados con secuencias completas, se realiza entrenamiento desde cero mediante predicción del siguiente token (*next token prediction*). Este enfoque permite al modelo aprender relaciones entre SNPs y patrones de ligamiento sin supervisión explícita.

El objetivo de entrenamiento consiste en predecir el SNP en la posición $t+1$ dada la secuencia de SNPs en posiciones 1 a t. La función de pérdida emplea cross-entropy entre la distribución predicha y el SNP verdadero. A través de este proceso, el modelo desarrolla representaciones internas que capturan dependencias entre SNPs y patrones poblacionales, estableciendo una base sólida para tareas posteriores de clasificación.

3.2.3. Clasificación supervisada

La estrategia final implica entrenamiento supervisado directo para clasificación de ancestría, añadiendo una capa de clasificación al modelo entrenado mediante predicción del siguiente token. Se agrega una MLP de clasificación al final del encoder para mapear representaciones internas a probabilidades de clase. La función de pérdida utiliza cross-entropy multiclase para optimización directa de la tarea de clasificación. Se emplean técnicas de regularización como early stopping y weight decay para prevenir overfitting, asegurando generalización adecuada a datos no vistos.

3.3. Entorno computacional y herramientas

Los experimentos se realizan en un entorno computacional de escala reducida, pero suficiente para realizar los experimentos con el modelo *HyenaDNA* dado su arquitectura subcuadrática.

La infraestructura de hardware incluye una GPU NVIDIA GeForce RTX 4060 Ti con 16 GB de memoria VRAM para aceleración de entrenamiento con soporte CUDA, un procesador Intel Core i5-12400F de 12va generación con 12 hilos de ejecución, 32 GB de RAM, y una unidad SSD de 1 TB para almacenamiento.

El desarrollo se basa en las siguientes herramientas y bibliotecas:

- **PyTorch** (Paszke et al., 2019): framework principal de aprendizaje automático.
- **PyTorch Lightning** (Falcon and The PyTorch Lightning team, 2019): manejo eficiente de entrenamiento y evaluación de modelos.
- **Hydra** (Yadan, 2019): manejo de configuraciones y parámetros de experimentos.
- **Transformers (Hugging Face)** (Wolf et al., 2020): manejo de arquitecturas de atención y modelos pre-entrenados.
- **HDF5/h5py** (The HDF Group, 2025; Collette, 2013): manejo eficiente de grandes volúmenes de datos genómicos.
- **NumPy** (Harris et al., 2020) y **Pandas** (McKinney, 2010): manipulación y análisis de datos numéricos.
- **bcftools** (Danecek et al., 2021): procesamiento de archivos VCF.
- **pyfaidx** (Shirley et al., 2015): manejo de archivos FASTA.

El desarrollo se estructura usando Visual Studio Code (Microsoft, 2015) con configuraciones (*launch configurations*) que permiten ejecutar y debuggear experimentos de forma organizada. Cada configuración corresponde a un experimento específico con parámetros claramente definidos. Las configuraciones incluyen semillas aleatorias y parámetros completos para garantizar reproducibilidad. Se integran herramientas de logging y monitoreo para seguimiento del progreso de entrenamiento, facilitando la organización y el análisis sistemático de los experimentos realizados.

3.4. Métricas y técnicas de análisis

La evaluación de los modelos entrenados emplea un conjunto de métricas cuantitativas y técnicas de visualización para interpretar tanto el rendimiento como el comportamiento interno de los modelos.

Para evaluar modelos entrenados mediante predicción del siguiente token se utiliza **perplejidad**, que mide la calidad del modelo de lenguaje genómico y su capacidad para predecir SNPs. En tareas de clasificación de ancestría, la **exactitud (*Accuracy*)** constituye la métrica principal, reportando la proporción de predicciones correctas. Adicionalmente, se calculan **métricas balanceadas** como F1-score, precisión y recall por clase para evaluar rendimiento en datasets potencialmente desbalanceados, proporcionando una visión más completa del comportamiento del modelo.

Las técnicas de interpretabilidad incluyen **saliency maps**, que visualizan regiones de la secuencia que contribuyen más significativamente a las predicciones de clasificación, permitiendo identificar qué SNPs o regiones genómicas son más informativos para determinar ancestría. Los **mapas de ancestría cromosómica** muestran las predicciones de ancestría a lo largo de cada cromosoma, revelando segmentos de ancestría homogénea y transiciones entre diferentes ancestrías. El **análisis de embeddings y estados ocultos** visualiza las representaciones aprendidas para comprender el comportamiento interno del modelo.

Para el análisis de representaciones se emplea **PCA (Análisis de Componentes Principales)** (Goodfellow et al., 2016), que reduce la dimensionalidad de los estados ocultos del modelo para visualizar la estructura de las representaciones aprendidas y detectar agrupamientos poblacionales. **t-SNE (t-Distributed Stochastic Neighbor Embedding)** (van der Maaten and Hinton, 2008) es una técnica de visualización no lineal aplicada tanto a estados ocultos como a embeddings posicionales para revelar estructura de clusters y relaciones entre poblaciones.

3.5. Consideraciones éticas y reproducibilidad

La investigación adhiere a principios éticos estándar para estudios genómicos. Todos los datos utilizados provienen de conjuntos públicos con consentimiento apropiado para investigación. Los datos de individuos están completamente anonimizados y no permiten identificación personal. El código completo, configuraciones experimentales y datos procesados están disponibles en un repositorio público[1] para facilitar la reproducción independiente de resultados,

[1] https://github.com/gonzalocameto/hyena-dna

asegurando transparencia y verificabilidad del trabajo realizado.

Capítulo 4

Presentación de los datos, análisis y discusión

En este capítulo se presentan los resultados experimentales obtenidos con *HyenaDNA* aplicados a tareas de clasificación de especies y detección de ancestrías a partir de datos genómicos. A partir de la metodología desarrollada en los capítulos anteriores, se diseñó una secuencia progresiva de experimentos que permiten evaluar, de forma controlada, el impacto de cada componente arquitectónico y de codificación en el rendimiento final del modelo.

Los experimentos se organizan desde escenarios de validación básica hasta configuraciones optimizadas de alto desempeño, de la siguiente manera:

1. **Clasificación de especies**: Validación inicial del modelo *HyenaDNA* en una tarea estándar de clasificación genómica.
2. **Detección de ancestría con secuencias completas**: Primer acercamiento a la identificación de patrones ancestrales empleando secuencias genómicas completas, destacando las limitaciones fundamentales del enfoque.
3. **Detección de ancestría usando solo SNPs**: Evaluación del modelo considerando únicamente posiciones polimórficas, eliminando regiones conservadas del genoma.
4. **Detección de ancestría con SNPs y codificación contextual**: Incorporación explícita de información posicional y cromosómica para restaurar el contexto genómico perdido al filtrar SNPs.
5. **Optimizaciones y entrenamientos mejorados**: Ajustes arquitectónicos y de entrenamiento orientados a maximizar la eficiencia y precisión

del sistema.

6. **Comparación HyenaDNA vs. Transformers**: Análisis comparativo entre arquitecturas subcuadráticas y atención densa en el dominio genómico.
7. **Análisis de embeddings posicionales y representaciones internas**: Exploración cualitativa de las representaciones ocultas generadas por el modelo y su relación con la estructura poblacional.

Cada sección presenta la metodología experimental, los resultados cuantitativos y una discusión interpretativa de los hallazgos en relación con los objetivos planteados. Finalmente, se sintetizan las conclusiones generales del capítulo.

4.1. Clasificación de especies

El primer experimento consiste en utilizar los pesos del backbone de *HyenaDNA* proporcionados por Nguyen et al. (2023) y entrenar desde cero un clasificador (*MLP*) para distinguir entre cinco especies a partir de secuencias de ADN. El objetivo es familiarizarse con el modelo y establecer una línea base que valide su capacidad en una tarea de clasificación genómica estándar, antes de abordar el tema central de esta tesis: la detección de ancestrías.

4.1.1. Configuración experimental

Se partió del backbone de *HyenaDNA* con contexto de 32k, cuyos pesos preentrenados en el genoma humano fueron publicados por Nguyen et al. (2023). Sobre este backbone se agregó una capa de clasificación (MLP) inicializada desde cero. Durante el *fine-tuning* se entrenan conjuntamente el backbone y el clasificador, de modo que las representaciones genómicas aprendidas en el preentrenamiento se adaptan a la tarea de clasificación de especies.

La arquitectura empleada consta de tres componentes principales:

- **Entrada**: Secuencias de ADN de longitud fija (32k nucleótidos)
- **Backbone LM**: Modelo de lenguaje *HyenaDNA* pre-entrenado que incluye una capa de embeddings (GPT2Embeddings) para mapear nucleótidos a vectores de dimensión 256, seguida de 2 bloques Hyena (cada

uno conteniendo HyenaOperator, LayerNorm y MLP) que producen representaciones genómicas de dimensión [32768, 256]

- **Decoder de secuencia**: Capa lineal que realiza clasificación utilizando el estado oculto de la última posición del Backbone LM (dim. 256) y mapea esta representación a las 5 clases de especies

El modelo completo contiene 1.69 millones de parámetros totales, todos entrenables durante el *fine-tuning*.

El entrenamiento se realizó durante 100 épocas (aproximadamente 4 horas) utilizando el optimizador AdamW (lr: 6e-5, weight decay: 0.1), con secuencias de 32 768 nucleótidos, tamaño de batch de 4 secuencias, regularización mediante gradient clipping (1.0) y dropout en embeddings (0.1), scheduler cosine con warmup, y entrenamiento en precisión mixta (16-bit).

4.1.2. Dataset y especies evaluadas

El experimento incluyó cinco especies seleccionadas para evaluar la capacidad del modelo de distinguir entre organismos evolutivamente distantes: humano, ratón, hipopótamo, cerdo y lémur. Durante el entrenamiento, se selecciona una especie al azar, luego un cromosoma, y por último una ventana de 32k nucleótidos, asegurando uniformidad en la selección de secuencias a nivel de especie, cromosoma y posición.

4.1.3. Resultados

El modelo alcanzó una exactitud (*accuracy*) de 0.92 en el conjunto de test, demostrando una buena capacidad para distinguir entre las cinco especies evaluadas. Este resultado confirma que las representaciones aprendidas por *HyenaDNA* durante el pre-entrenamiento capturan información relevante para tareas de clasificación.

El Tabla 4.1 resume las métricas de rendimiento por especie. Cerdo alcanza el mejor rendimiento global, con un F1-score de 0.98. Ratón muestra excelente recall (0.97) con precisión algo menor (0.86). Humano presenta alta precisión (0.99) pero recall menor (0.80), lo que indica que el modelo es conservador al clasificar secuencias como humanas. Hipopótamo muestra la mayor confusión, particularmente con lémur (162 casos mal clasificados), como se observa en la Figura 4.1.

Tabla 4.1: Métricas de rendimiento por especie en clasificación genómica.

Especie	Precisión	Recall	F1-score
Cerdo	0.99	0.98	0.98
Lémur	0.97	0.97	0.97
Ratón	0.86	0.97	0.91
Humano	0.99	0.80	0.88
Hipopótamo	0.82	0.87	0.84
Promedio	—	—	0.92

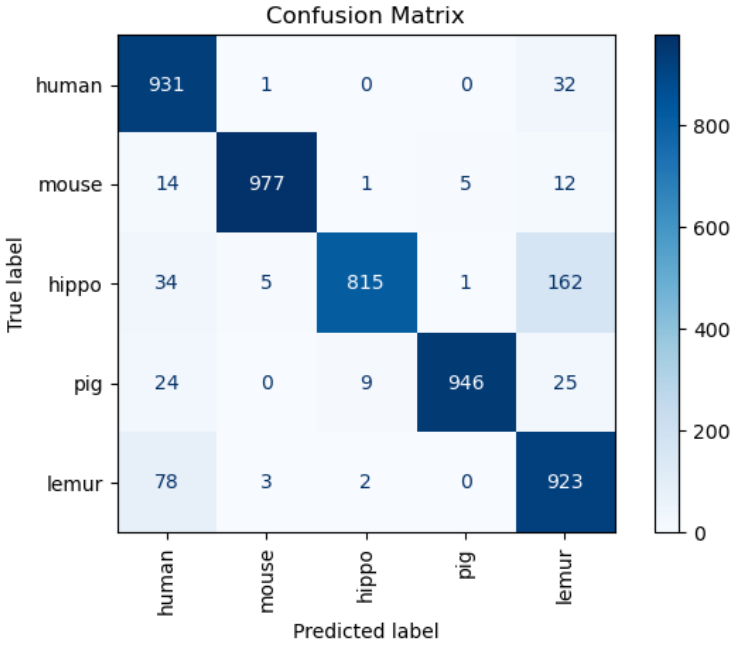


Figura 4.1: Matriz de confusión para clasificación de especies. Los valores en la diagonal principal representan clasificaciones correctas.

La Figura 4.2 muestra el mapa de calor de probabilidades de clase a lo largo de un cromosoma de un lémur. El modelo exhibe alta confianza (regiones amarillas) en segmentos específicos de la secuencia, mientras que la variabilidad en la confianza a lo largo del genoma sugiere que diferentes regiones contienen niveles variables de información taxonómica. Las zonas de baja confianza pueden deberse a limitaciones del modelo o a similitudes genómicas entre especies.

Para comprender qué características de las secuencias utiliza el modelo para realizar clasificaciones, se generaron *saliency maps* que identifican las posiciones más influyentes en las decisiones de clasificación. La Figura 4.3 presenta un *saliency map* para una secuencia aleatoria de lémur de 32k nucleótidos, donde

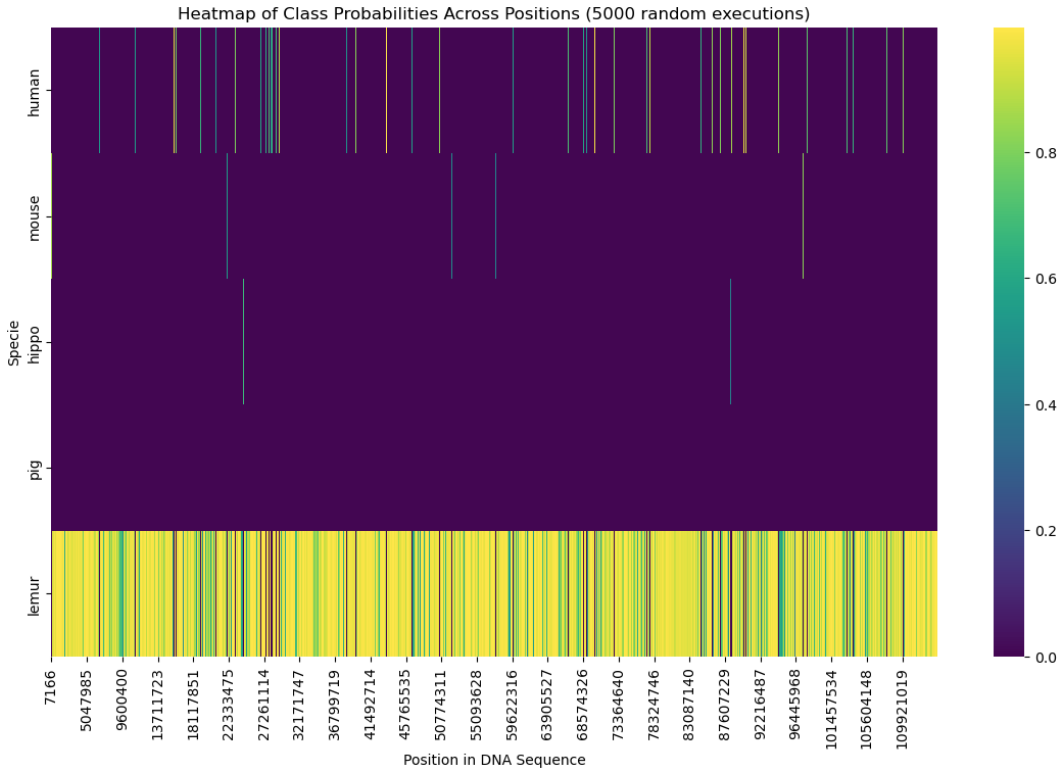


Figura 4.2: Mapa de calor de probabilidades de clase a través de 5000 muestras de largo 32 000 nucleótidos del cromosoma 5 de un lémur. Las regiones amarillas indican alta confianza en la predicción correcta, mientras que las regiones oscuras muestran incertidumbre o predicciones incorrectas.

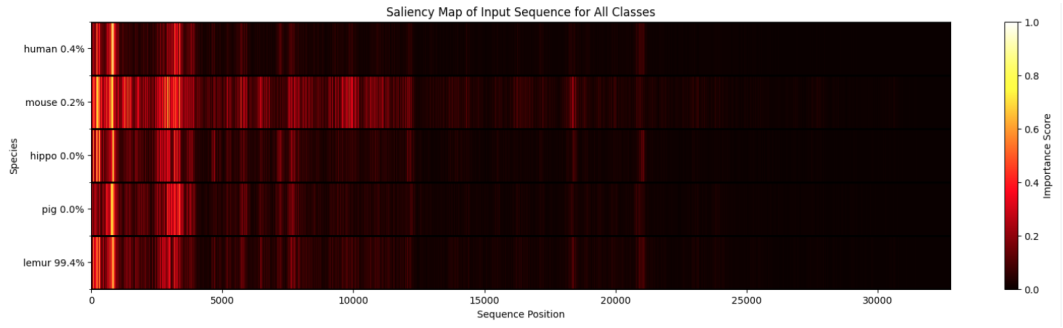


Figura 4.3: *Saliency maps* para una secuencia aleatoria de lémur. La intensidad del color indica la importancia de cada posición para la decisión de clasificación. Los resultados muestran alta confianza con lémur 99.4 %.

se observa que las primeras posiciones muestran consistentemente mayor importancia, sugiriendo que secuencias más cortas podrían ser suficientes para la clasificación. Además, las regiones con mayor importancia se distribuyen de manera no uniforme, concentrándose en motivos específicos potencialmente relacionados con características taxonómicas distintivas.

4.1.4. Discusión

Los resultados del experimento de clasificación de especies confirman la eficacia de *HyenaDNA* para tareas de clasificación genómica estándar. La exactitud de 0.92 demuestra que las representaciones pre-entrenadas capturan información taxonómica discriminativa, mientras que el *fine-tuning* requiere únicamente 4 horas partiendo del modelo pre-entrenado, evidenciando la eficiencia del enfoque. Este experimento establece la viabilidad de *HyenaDNA* para tareas de clasificación genómica, proporcionando confianza para su aplicación en el problema más complejo de detección de ancestría genética humana, donde las diferencias entre clases son más sutiles que las observadas entre especies evolutivamente distantes.

4.2. Detección de ancestría con secuencias completas

El segundo experimento marca la transición hacia el objetivo principal de esta tesis: la detección de ancestría genética humana. A diferencia del experimento de clasificación de especies, donde las diferencias genómicas son amplias, la detección de ancestría requiere identificar variaciones mucho más sutiles entre poblaciones humanas que comparten la gran mayoría de su secuencia.

Este primer enfoque emplea secuencias completas de nucleótidos para evaluar si *HyenaDNA* es capaz de detectar patrones ancestrales directamente desde la secuencia genómica, sin preprocesamiento o selección previa de variantes.

4.2.1. Diseño experimental

El experimento se basa en el conjunto de datos del proyecto *1000 Genomes* descrito en el Capítulo 3, considerando tres poblaciones representativas de distintos continentes: **Gambia (África)**, **China y Japón (Asia)** y **Gran Bretaña (Europa)**.

Para cada muestra de entrenamiento se aplicó un muestreo aleatorio en cuatro etapas: selección de la población, del individuo dentro de la población, del cromosoma (entre los 22 autosómicos) y de la posición inicial de la ventana de secuencia. Esta estrategia garantiza una cobertura uniforme del genoma y evita sesgos hacia regiones específicas.

El entrenamiento se realizó con un número de secuencias por época variable (entre 1000 y 10 000) y una estrategia de *batch* dinámico, reduciendo progresivamente el tamaño del lote cada 10 épocas (de 64 a 8 muestras). Esta técnica permitió estabilizar el entrenamiento utilizando tamaños de batch mayor durante las primeras épocas donde el *learning-rate* es alto. Se evaluaron los modelos *HyenaDNA-1k* y *HyenaDNA-32k*.

4.2.2. Resultados y análisis

Los resultados obtenidos fueron significativamente inferiores a las expectativas derivadas del experimento de especies, con una exactitud promedio de $\sim$0.33, equivalente al rendimiento esperado de un clasificador aleatorio para tres clases. El modelo no logró aprender patrones discriminativos, indicando que las secuencias completas carecen de suficiente señal ancestral cuando se muestrean de forma aleatoria.

Este comportamiento se explica por la baja densidad de polimorfismos de nucleótido único (SNPs) en el genoma humano. En promedio, solo el 0.15 % de las posiciones corresponden a SNPs, lo que equivale aproximadamente a un SNP cada 700 nucleótidos. De este modo, una secuencia de 1k nucleótidos contiene en promedio $\sim$1.46 SNPs, mientras que una de 32k contiene alrededor de 47 SNPs informativos. Procesar millones de posiciones conservadas para obtener unas pocas variantes relevantes conduce a una fuerte dilución de la señal de ancestría.

4.2.3. Discusión

El bajo rendimiento de este experimento muestra una limitación clara del uso de secuencias completas para detectar ancestría. Los humanos comparten aproximadamente el 99,9 % de su secuencia genómica, con variaciones (SNPs) ocurriendo solo cada $\sim$600–1000 nucleótidos (The 1000 Genomes Project Consortium, 2015). La información útil está muy dispersa en el genoma, ya que la mayoría de las posiciones son idénticas entre individuos y solo unas pocas aportan diferencias relevantes. Estas variantes quedan ocultas entre regiones altamente conservadas, lo que dificulta que el modelo las aprenda.

Como resultado, gran parte del cálculo se destina a procesar datos repetidos que no aportan valor real. Esto contrasta con la tarea de clasificación de

especies (exactitud 0.92), donde las diferencias evolutivas son mucho mayores y generan señales más fáciles de distinguir.

Estos resultados indican que es necesario cambiar la estrategia de entrada para aprovechar mejor la información genética. En lugar de usar la secuencia completa, conviene enfocarse en las posiciones variables —los SNPs—, que concentran la señal ancestral. El siguiente experimento explora este enfoque, evaluando si *HyenaDNA* puede mejorar la detección de ancestría humana usando esta representación más compacta.

4.3. Detección de ancestría usando solo SNPs

A partir de los resultados del experimento anterior, este tercer experimento explora una alternativa más enfocada: trabajar únicamente con polimorfismos de nucleótido único (SNPs). El objetivo es evaluar si concentrar la información en posiciones variables permite mejorar la detección de ancestría y el rendimiento general del modelo.

4.3.1. Diseño experimental

En este caso se construyó una representación condensada del genoma, que incluye exclusivamente las posiciones variables. Este procedimiento redujo los $\sim$3.2 mil millones de nucleótidos del genoma humano a aproximadamente 4.5 millones de SNPs informativos. Las muestras de entrenamiento se generaron muestreando directamente sobre este array condensado, siguiendo la secuencia: población $\rightarrow$ individuo $\rightarrow$ cromosoma $\rightarrow$ posición dentro del array de SNPs del cromosoma seleccionado. A partir de allí, se extrajeron ventanas de longitud fija (1k o 32k SNPs según el modelo), sin incorporar coordenadas genómicas absolutas ni distancias físicas entre variantes.

La configuración experimental se mantuvo coherente con la del experimento anterior en cuanto a las poblaciones utilizadas y la división de datos. Cada muestra contiene únicamente SNPs consecutivos en el array condensado, preservando la identidad alélica y el orden relativo, pero sin incluir información posicional ni cromosómica absoluta. Los modelos empleados —*HyenaDNA-1k* y *HyenaDNA-32k*— fueron los mismos utilizados previamente, sin reajuste de pesos, lo que implica una compatibilidad parcial entre ambas representaciones.

4.3.2. Resultados y análisis

El uso exclusivo de SNPs produjo una mejora marginal pero consistente respecto a las secuencias completas. La exactitud alcanzó valores entre 0.40 y 0.45, superando el rendimiento aleatorio (0.33) y mostrando una convergencia estable durante el entrenamiento. A diferencia del experimento anterior, la función de pérdida se redujo de manera sostenida y no se observó sobreajuste evidente, lo que sugiere que el modelo logró capturar parcialmente la señal ancestral.

Aunque el desempeño sigue lejos de los niveles observados en la clasificación de especies, este resultado confirma empíricamente que los SNPs contienen información discriminativa suficiente para distinguir poblaciones humanas, y que eliminar regiones conservadas mejora efectivamente la relación señal-ruido del modelo.

4.3.3. Discusión

Este experimento confirma que focalizarse en los SNPs mejora el aprovechamiento de la información ancestral y reduce el ruido derivado de las regiones conservadas. Sin embargo, los resultados también muestran que la identidad alélica por sí sola no basta para una clasificación precisa.

El siguiente paso consiste en incorporar información posicional que permita al modelo aprender dependencias entre variantes, sin perder eficiencia computacional.

4.4. Detección de ancestría usando SNPs con codificación contextual

Este experimento busca superar las limitaciones observadas al trabajar con SNPs sin contexto posicional. El objetivo es evaluar si combinar información alélica concentrada con contexto cromosómico y posicional permite recuperar el poder discriminativo perdido y mejorar el rendimiento en detección de ancestría.

La Figura 4.4 resume el flujo completo de procesamiento, desde la secuencia cromosómica completa hasta el tensor de entrada al modelo, integrando los pasos de selección y condensación de SNPs descritos en la Sección 4.3 con las

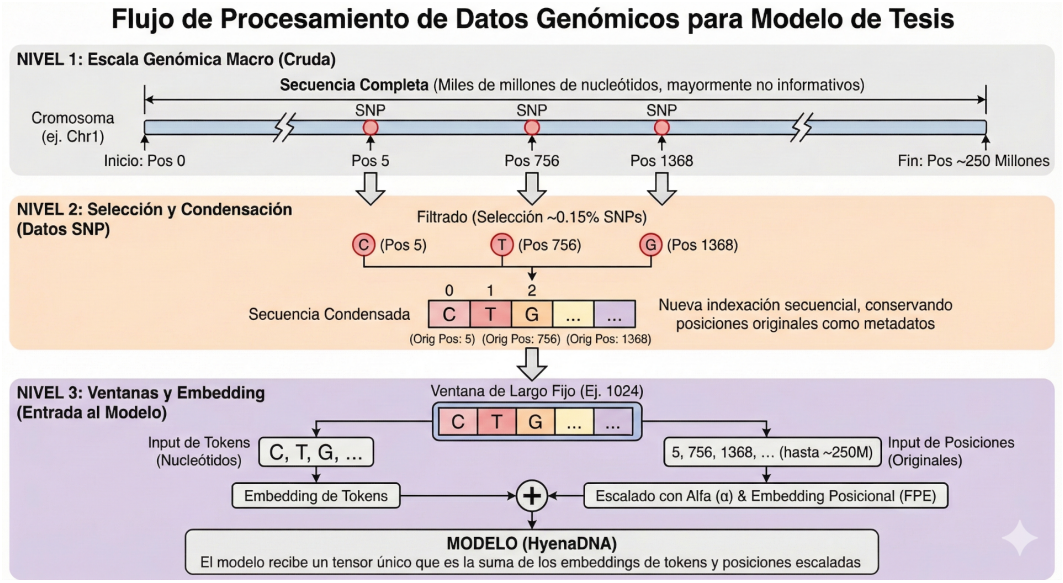


Figura 4.4: Flujo de procesamiento de datos genómicos. Nivel 1: selección de SNPs desde la secuencia cromosómica completa. Nivel 2: condensación en una secuencia indexada secuencialmente, conservando posiciones originales como metadatos. Nivel 3: extracción de ventanas de largo fijo y generación del tensor de entrada mediante la suma de embeddings de tokens y embeddings posicionales factorizados (FPE) escalados con α.

codificaciones posicionales introducidas en este experimento.

4.4.1. Incorporación de contexto genómico

El modelo *HyenaDNA* se extendió para integrar dos tipos de información contextual: codificación cromosómica y codificación posicional (Nivel 3 de la Figura 4.4). Estas extensiones se diseñaron para mantener la eficiencia computacional mientras se aumenta la sensibilidad a dependencias de largo alcance.

Codificación cromosómica. A cada muestra se le agregó un prefijo binario que identifica el cromosoma de origen. Este vector, de cinco bits, se concatena a la secuencia de entrada y permite distinguir patrones específicos de cada cromosoma sin introducir nuevos parámetros a entrenar. Ejemplo de codificación:

- Cromosoma 1: [0,0,0,0,1] + secuencia de SNPs
- Cromosoma 2: [0,0,0,1,0] + secuencia de SNPs
- Cromosoma 22: [1,0,1,1,0] + secuencia de SNPs

Codificación posicional. Se evaluaron distintas estrategias para incorporar información sobre la ubicación de los SNPs en el genoma. Las posiciones utilizadas son absolutas dentro de cada cromosoma: cada SNP se identifica por su posición genómica real en el cromosoma de origen (por ejemplo, la posición 1 234 567 del cromosoma 3). Estas posiciones absolutas se utilizan como entrada para las distintas codificaciones posicionales evaluadas, ajustando cada método al rango posicional cromosómico (∼250 millones de posiciones) y a las limitaciones de memoria disponibles:

- **Sinusoidal Encoding (PE):** versión clásica no parametrizada, optimizada para el rango genómico completo. Sin costo adicional en memoria.
- **Positional Embeddings (PosEmb):** embeddings entrenables, efectivos pero con alto costo de memoria a resolución de nucleótido.
- **Factorized Positional Embeddings (FPE):** factorización $E = A \times B$ con $k = 32$, reduciendo drásticamente los parámetros sin perder capacidad expresiva.
- **Rotary y Fourier Embeddings:** métodos sin parámetros entrenables, incluidos para comparación.
- **Convolución implícita Hyena:** uso directo de posiciones reales escaladas ($\alpha = 1000$), aprovechando la convolución implícita del operador Hyena.

4.4.2. Configuración experimental

Las pruebas combinaron las distintas estrategias posicionales con una codificación cromosómica binaria en los modelos *HyenaDNA-1k* y *HyenaDNA-32k*. Las condiciones de entrenamiento fueron unificadas para asegurar comparabilidad:

- **Dataset:** 45 individuos (15 por continente) del proyecto 1000 Genomes. En la Sección 4.5.1 se describe la expansión posterior a 590 individuos.
- **Genoma de referencia:** hg38.
- **Tarea:** Clasificación de ancestría continental: Europa (Reino Unido), África (Gambia) y Asia (China, Japón).
- **Longitud de secuencia:** 1024 SNPs.
- **Rango posicional:** $P = 2.5 \times 10^8$.
- **Dimensión de embeddings:** $d = 128$.

- **Optimizador:** AdamW en GPU de 16 GB.

4.4.3. Resultados

Las codificaciones no entrenables (PE, Rotary, Fourier) mostraron mejoras marginales, mientras que las variantes parametrizadas resultaron significativamente superiores, destacándose la factorización FPE (Sección 2.8.3). En FPE, el factor de escala α (Sección 2.8.6) controla la resolución posicional: valores altos ($\alpha = 25\,000$) degradan la precisión, mientras que $\alpha = 100$ ofrece un buen equilibrio entre precisión y uso de memoria.

La Tabla 4.2 resume la exactitud obtenida con FPE y $\alpha = 100$ para distintas longitudes de contexto.

Tabla 4.2: Exactitud (%) en detección de ancestría con FPE ($\alpha = 100$) según longitud de contexto.

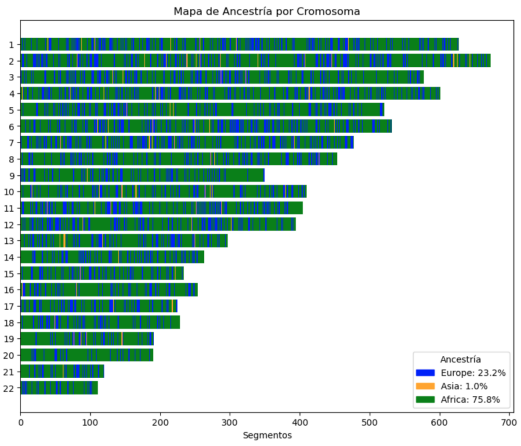
Modelo	**Exactitud (%)**
HyenaDNA-128	68
HyenaDNA-1k	83
HyenaDNA-32k	93

El rendimiento mejora con la longitud de contexto, indicando que las dependencias ancestrales se extienden a lo largo de regiones genómicas amplias. Esta configuración se adopta en los experimentos posteriores; en la Sección 4.5 se presenta un análisis más detallado de las distintas técnicas de codificación posicional, valores de α y longitudes de secuencia.

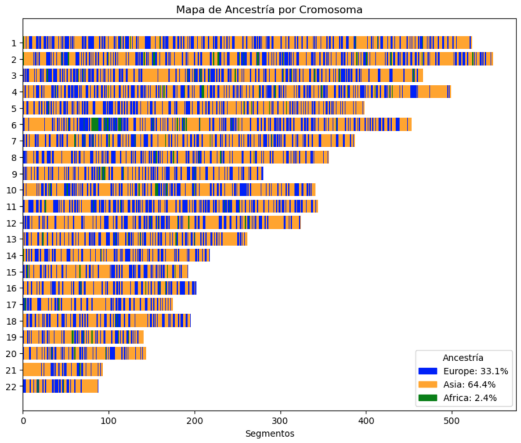
4.4.4. Mapas de ancestría cromosómica

Para una interpretación visual se generaron mapas de ancestría con la configuración óptima (*HyenaDNA-1k + FPE* con $\alpha = 100$). La Figura 4.5 muestra las predicciones para tres individuos representativos (Gambia, Japón y Gran Bretaña).

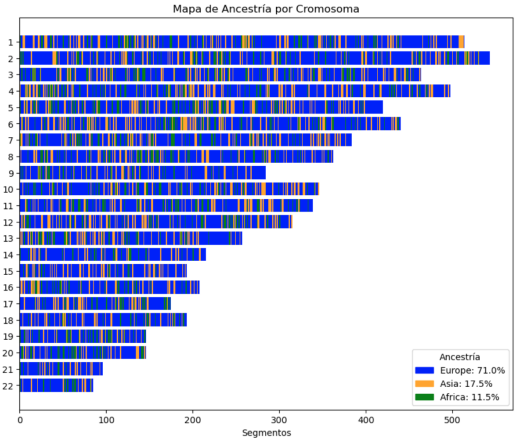
Observaciones. El modelo asigna correctamente la ancestría principal de cada individuo, mostrando coherencia con la población esperada. Los mapas reflejan patrones cromosómicos consistentes a lo largo del genoma, lo que indica que el modelo capta dependencias de largo alcance entre regiones. También aparecen segmentos mixtos en algunos individuos, lo que puede deberse tanto



(a) Gambia (HG02816)



(b) Japón (NA18948)



(c) Gran Bretaña (HG00252)

Figura 4.5: Mapas de ancestría cromosómica para tres individuos de poblaciones africana, asiática y europea. Los colores representan la proporción de ancestría estimada por el modelo.

a la variación biológica real como a limitaciones del modelo en zonas con poca información.

4.4.5. Discusión

Los resultados confirman que la integración de información posicional y cromosómica recupera la capacidad discriminativa perdida al trabajar con SNPs aislados. La arquitectura *HyenaDNA*, gracias a su convolución implícita y complejidad subcuadrática, permite ampliar el contexto sin incrementar el costo computacional de manera significativa.

Los *Factorized Positional Embeddings* emergen como la estrategia más efec-

tiva, alcanzando 93 % de exactitud con *HyenaDNA-32k*. Este desempeño es competitivo con modelos estado del arte como *RFMix* y *G-Nomix*, manteniendo resolución de nucleótido y capacidad para procesar regiones genómicas extensas. En conjunto, estos resultados demuestran que las señales de ancestría requieren una representación posicional de alta resolución y confirman la capacidad de *HyenaDNA* para modelar relaciones genómicas de largo alcance con eficiencia.

4.5. Optimizaciones y entrenamientos mejorados

Este quinto experimento incorpora optimizaciones sistemáticas para maximizar el rendimiento del sistema de detección de ancestría. Las mejoras abarcan: expansión del conjunto de datos, refinamiento de las estrategias de codificación, optimización de la arquitectura del modelo y una estrategia de entrenamiento en dos etapas.

4.5.1. Expansión del conjunto de datos

Cobertura poblacional Se expandió el dataset para mejorar robustez y generalización, incorporando múltiples poblaciones por continente:

- **Europa (240 individuos)**: Reino Unido, Italia, España, Finlandia.
- **África (160 individuos)**: Gambia, Kenia, Sierra Leona.
- **Asia (190 individuos)**: China, Japón, Vietnam.

Total: 590 individuos, un incremento de 13× respecto de experimentos anteriores (45 individuos).

Almacenamiento y partición Se adoptó HDF5 con estructura jerárquica por cromosoma e individuo, lo que acelera el acceso y reduce el tiempo de carga durante el entrenamiento. Cada individuo se encuentra en un archivo separado de aproximadamente 12 MB, lo que representa un total de 7 GB para los 590 individuos. La partición 70/15/15 % preserva el balance poblacional en entrenamiento, validación y test.

4.5.2. Refinamiento de estrategias de codificación

Codificación cromosómica Se reemplazó la codificación binaria inicial (p. ej., chr 1: [0,0,0,0,1]) por un *embedding* cromosómico entrenable de dimensión 22×128: una matriz donde cada fila es un vector denso de 128 dimensiones que representa un cromosoma. Por ejemplo, el cromosoma 3 se representa como un vector $\mathbf{e}_3 \in \mathbb{R}^{128}$ aprendido durante el entrenamiento, que se suma a los embeddings de todas las posiciones de la secuencia de entrada correspondiente a ese cromosoma. Esto aumenta la capacidad expresiva y permite una integración más rica con la representación secuencial respecto de la codificación binaria original.

Parámetros posicionales Se exploraron sistemáticamente los hiperparámetros de la codificación posicional, variando el factor de escala α para mapear ~ 250 millones de posiciones genómicas a espacios representables y el factor k, que controla el rango de las matrices A y B en *Factorized Positional Embeddings* (FPE), con el fin de equilibrar capacidad expresiva y eficiencia paramétrica. El análisis comparativo se presenta en la Sección 4.5.5.

4.5.3. Optimización de la arquitectura

Se expandió el clasificador final (MLP) desde `Linear [256] → [3]` a:

- `Linear [256] → [512]`
- `GELU [512] → [512]`
- `Linear [512] → [256]`
- `GELU [256] → [256]`
- `Linear [256] → [3]`

Esta ampliación aumenta la expresividad del clasificador: la adición de capas ocultas con activaciones no lineales incrementa la capacidad de la red para aproximar funciones de decisión complejas (Hornik et al., 1989), lo que resulta relevante cuando las representaciones del encoder requieren transformaciones no triviales para separar las clases de ancestría.

4.5.4. Entrenamiento en dos etapas

La estrategia apunta a un preentrenamiento directo sobre SNPs con información posicional, superando la limitación planteada en la Sección 4.3.1

sobre la incompatibilidad de reutilizar modelos preentrenados en secuencias completas al trabajar con representaciones basadas en SNPs.

Etapa 1: Predicción del siguiente token En esta primera etapa preentrenamos un modelo de lenguaje (LM) genómico para predecir el siguiente SNP en la secuencia. Utilizamos entre 1000 y 10 000 secuencias por época, con *batch size* de 128, y evaluamos el progreso mediante *perplexity* como métrica principal.

Etapa 2: *Fine-tuning* para clasificación Se inicializa el encoder Hyena con los pesos previamente preentrenados en la etapa 1 y se entrena conjuntamente con el MLP clasificador expandido.

4.5.5. Análisis comparativo de codificación posicional

Se evaluaron técnicas de codificación posicional considerando dinámicas de entrenamiento, uso de memoria y rendimiento.

Dinámicas de entrenamiento y perplejidad Las curvas de *perplexity* para la tarea de *next-token* (1024 SNPs) se muestran en la Figura 4.6. FPE (α=10) alcanza la menor perplejidad final con un valor de 1.41, seguido por FPE (α=100) con 1.72 y FPE (α=1000) con 2.79. Técnicas no parametrizadas (PE, RoPE, Fourier) ofrecen mejoras marginales o nulas sobre el *baseline* sin posición.

Importancia relativa de nucleótidos vs. posición El experimento *FPE 100 No Word Emb* (sólo posición, sin *embeddings* de nucleótidos) arroja perplejidad 4.00 —idéntica al *baseline*—, demostrando que la información posicional por sí sola es insuficiente y que **posición y alelos son complementarios y críticos**.

Uso de memoria y viabilidad Se analizaron resoluciones FPE y su impacto en memoria, incluyendo el factor 4× de AdamW requerido para el entrenamiento del modelo.

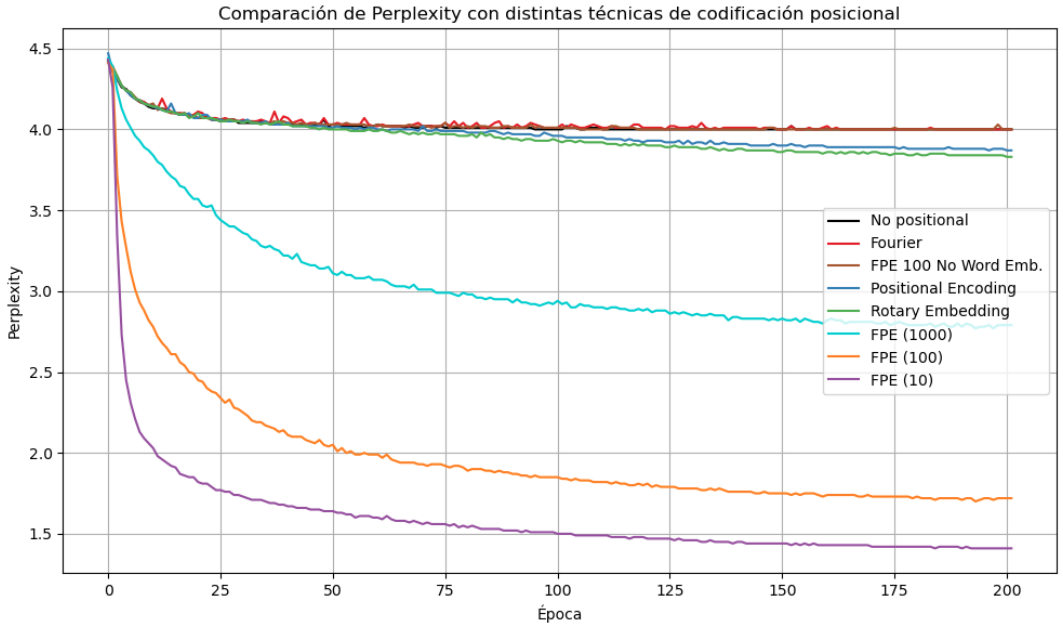


Figura 4.6: Evolución de la perplejidad con distintas codificaciones posicionales (secuencias de 1024 SNPs). FPE (α=10) es la mejor; PE y RoPE sólo superan levemente al *baseline* sin posición.

Tabla 4.3: Resolución FPE y memoria

Resolución	α	P_{ef}	**Mem+AdamW**	**Eval.**
1 nucleótido	1	2.5×10^8	59.6 GiB	No
10 SNPs	10	2.5×10^7	5.96 GiB	Sí
100 SNPs	100	2.5×10^6	0.6 GiB	Sí
1000 SNPs	1000	2.5×10^5	0.04 GiB	Sí

Tabla 4.4: Rendimiento en *next-token prediction*

Configuración	**Perplejidad**	**Memoria+AdamW**
Sin posición	4.00	-
Positional Encoding	3.87	-
Rotary Embeddings	3.83	-
Fourier Features	4.00	-
Convolución implícita	4.00	-
FPE (α=1000)	2.79	0.04 **GiB**
FPE (α=100)	1.72	0.6 **GiB**
FPE (α=10)	1.41	5.96 **GiB**
FPE 100 sin *word emb.*	4.00	0.6 GiB

Selección final **FPE con** α=100 ofrece el mejor equilibrio entre rendimiento (1.72) y memoria (0.6 GiB), habilitando contextos largos y arquitecturas con más capas. La memoria base FPE se estima como $(P_{\text{efectivo}} \cdot k + k \cdot d) \cdot 4$ bytes, con k=16, d=128.

Interpretación biológica de la importancia posicional La criticidad del FPE tiene una explicación biológica clara. Los SNPs son marcadores dispersos extraídos de coordenadas genómicas específicas. Sin información posicional, el modelo procesa únicamente colecciones desordenadas de alelos (A, C, G, T), perdiendo contexto biológico fundamental:

1. **Desequilibrio de ligamiento (LD)**: Las asociaciones no aleatorias entre alelos en diferentes loci varían característicamente según la población debido a su historia demográfica, cuellos de botella y eventos de mestizaje.
2. **Estructura haplotípica**: Las combinaciones de alelos heredadas juntas en segmentos cromosómicos reflejan historias de recombinación específicas de cada población.
3. **Paisajes de recombinación**: Las regiones genómicas con alta o baja recombinación moldean la diversidad genética de forma diferente según la población.

Sin FPE, la precisión de clasificación colapsa a 34.6 % (cerca del 33.3 % aleatorio para 3 clases), confirmando que la información posicional no es solo beneficiosa sino *esencial* para tareas basadas en SNPs.

4.5.6. Resultados finales

Una vez definidas las optimizaciones en dataset, codificación y arquitectura, se entrenó el modelo en la configuración seleccionada para evaluar el impacto conjunto de todas estas mejoras. En esta subsección se resume dicha configuración final, la dinámica de entrenamiento en las dos etapas propuestas y el rendimiento alcanzado, tanto a nivel de métricas globales de clasificación como de patrones de ancestría cromosómica.

Configuración experimental optimizada Los resultados finales se obtuvieron utilizando una configuración única, derivada de los análisis anteriores.

Arquitectura del modelo:

- **Modelo base**: HyenaDNA con 4 capas Hyena.
- **Dimensión del modelo**: $d = 128$.
- **Longitudes de contexto evaluadas**: 128, 512, 1024, 2048 y 4096 SNPs.
- **Codificación posicional**: FPE con $\alpha = 100$ para secuencias ≥ 512 SNPs y $\alpha = 10$ para 128 SNPs.

Parámetros de entrenamiento:

- **Estrategia**: entrenamiento en dos etapas (predicción del siguiente token + *fine-tuning* para clasificación).
- **Optimizador**: AdamW (`lr`=6e-5, `weight_decay`=0.1, `betas`=[0.9, 0.999]).
- **Scheduler**: *Cosine Warmpup* (80 pasos de *warmup*, `lr_min`=3e-4).
- **Batch size**: 256 durante 50 épocas, 128 durante otras 50 y 64 en las 200 épocas restantes.
- **Épocas máximas**: 300, con *early stopping* basado en *validation accuracy*.
- **Precisión numérica**: *mixed precision* (16 bits) para mejorar la eficiencia.

Esta configuración fija el contexto sobre el que deben interpretarse los resultados que se describen a continuación.

Resultados del entrenamiento El proceso de entrenamiento se llevó a cabo siguiendo la estrategia en dos etapas introducida anteriormente, evaluando sistemáticamente el impacto de la longitud de secuencia y el factor de escala α en el rendimiento del modelo.

La Tabla 4.5 resume los resultados obtenidos para cada configuración, incluyendo ambas etapas del entrenamiento.

Los resultados revelan dos patrones importantes. Primero, existe una clara correlación entre la longitud de secuencia y el rendimiento de clasificación: la configuración con 32 768 SNPs alcanza el mejor *accuracy* de validación (99.89 %) y la menor perplejidad (1.62), mientras que secuencias más cortas obtienen rendimientos progresivamente inferiores. El modelo con 4096 SNPs alcanza 99.34 % de *accuracy*, demostrando que contextos de longitud moderada ya capturan la mayor parte de la señal ancestral.

Segundo, la comparación de las dos configuraciones con 128 SNPs ilustra el impacto del factor α. Con α=100 (menor resolución posicional), el modelo

Tabla 4.5: Resultados comparativos por longitud de secuencia y factor α

Secuencia	α	Etapa 1 (NTP)		Etapa 2 (Clasif.)		Tiempo
(SNPs)		Tiempo	PPL Val	Tiempo	Acc Val	Total
128	100	26 min	2.125	43 min	75.11 %	69 min
128	10	59 min	1.722	134 min	83.24 %	193 min
512	100	60 min	1.842	91 min	92.07 %	151 min
1024	100	55 min	1.864	153 min	96.64 %	208 min
2048	100	183 min	1.675	305 min	98.38 %	488 min
4096	100	187 min	1.663	552 min	99.34 %	739 min
32 768	**100**	**28:00**	1.62	**22:00**	99.89 %	23.6**h**

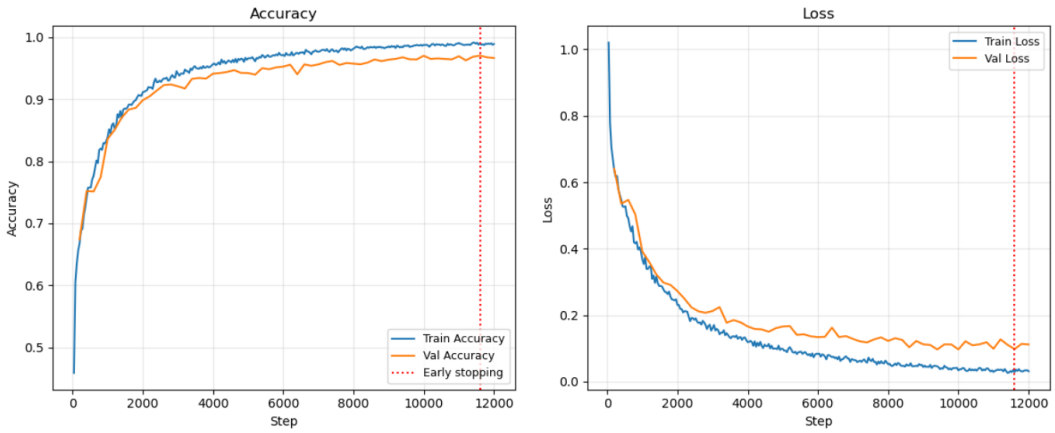


Figura 4.7: Curvas de entrenamiento del modelo optimizado durante la etapa de *fine-tuning* para clasificación de ancestría. (Izquierda) Evolución de *accuracy* en entrenamiento (azul) y validación (naranja), mostrando convergencia consistente hasta alcanzar 97.04 % en validación. La línea roja punteada indica el punto de *early stopping* en el `step` 11599 (época ∼290). (Derecha) Evolución de la pérdida (*loss*) en entrenamiento y validación, demostrando convergencia suave sin signos de sobreajuste marcado.

alcanza 75.11 % de *accuracy*, mientras que con α=10 (mayor resolución) mejora a 83.24 %. Esto demuestra que para secuencias cortas, una codificación posicional más fina es beneficiosa. Sin embargo, para secuencias largas ($\geq$512 SNPs), α=100 ofrece un equilibrio óptimo entre rendimiento y eficiencia de memoria.

Las curvas de entrenamiento del modelo de 1024 SNPs se muestran en la Figura 4.7. Se observa una mejora sostenida del *accuracy* en entrenamiento y validación hasta el punto de *early stopping*, así como una disminución suave de la pérdida sin indicios claros de sobreajuste.

Rendimiento de clasificación El modelo optimizado con contexto de 4096 SNPs alcanza 99.34 % de *accuracy* en la tarea de clasificación de ancestría continental, con métricas muy equilibradas entre las tres clases y F1-score de 0.99 para cada una. La matriz de confusión normalizada (Tabla 4.6) muestra que los errores de clasificación entre poblaciones son escasos y se concentran en pocos casos frontera.

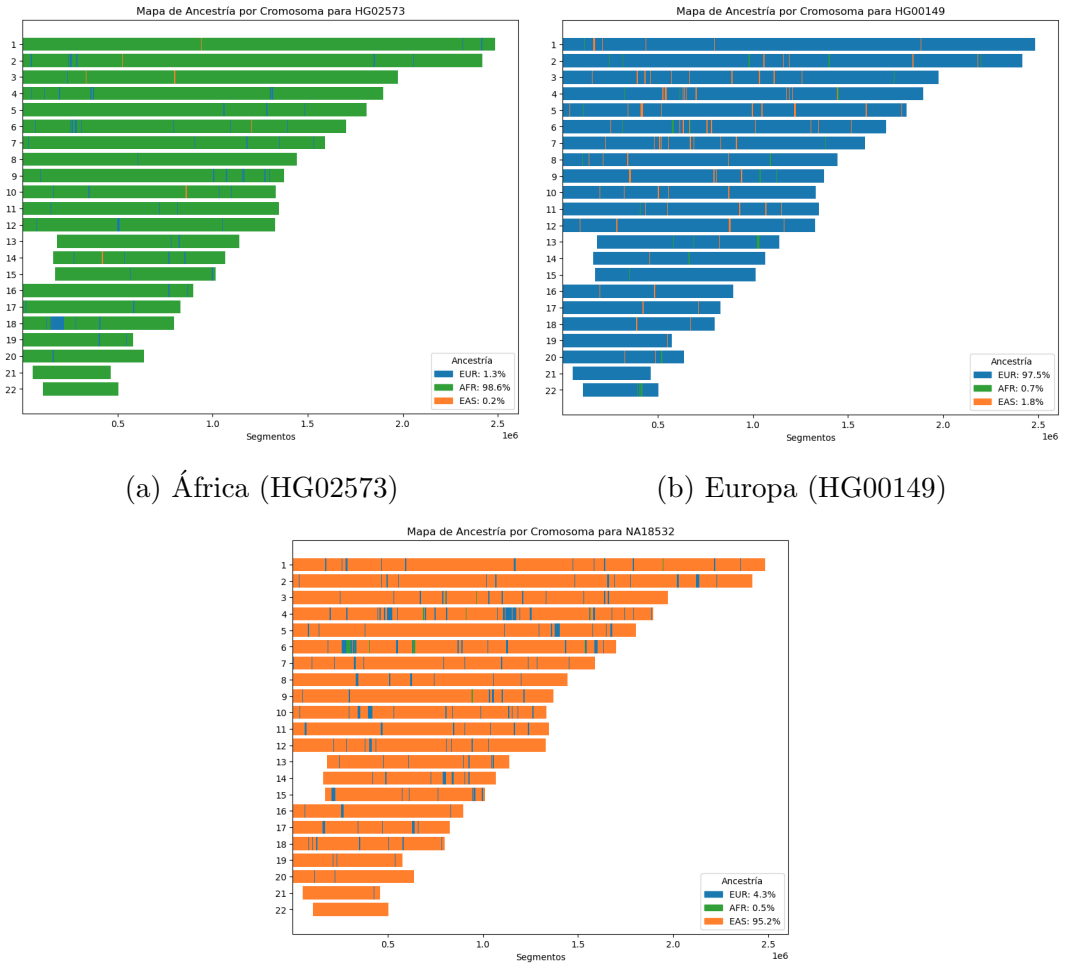
Tabla 4.6: Matriz de confusión normalizada (4096 SNPs, *accuracy* 99.34 %)

Real/Pred.	EUR	AFR	EAS
EUR	0.993	0.004	0.003
AFR	0.006	0.991	0.003
EAS	0.005	0.001	0.994

Comparación con métodos del estado del arte Gnomix (Hilmarsson et al., 2022), un método reciente para inferencia de ancestría local, alcanza 98.92 % de *accuracy* distinguiendo 8 poblaciones utilizando 22 modelos XGBoost específicos por cromosoma con ventanas de aproximadamente 4587 SNPs. Nuestro modelo unificado *end-to-end* alcanza 99.34 % de *accuracy* con contextos de 4096 SNPs y 99.89 % con 32 768 SNPs, igualando o superando a Gnomix con una arquitectura más simple que no requiere entrenar modelos separados por cromosoma. Esta comparación demuestra que las arquitecturas subcuadráticas pueden competir con métodos especializados del estado del arte para inferencia de ancestría.

Mapas de ancestría cromosómica Además de las métricas agregadas, es relevante analizar la coherencia de las predicciones a nivel cromosómico. La Figura 4.8 muestra mapas de ancestría cromosómica para tres individuos representativos.

En comparación con versiones previas del modelo, estos mapas muestran patrones de ancestría menos fragmentados y más coherentes a lo largo de los cromosomas, lo que sugiere una mejor captura de la estructura haplotípica subyacente. La reducción de segmentos ancestrales inconsistentes es coherente con las optimizaciones introducidas (entrenamiento en dos etapas, dataset expandido y arquitectura mejorada) y refuerza la validez biológica de las predicciones más allá de las métricas globales de clasificación.



(a) África (HG02573)

(b) Europa (HG00149)

(c) Asia (NA18532)

Figura 4.8: Mapas de ancestría cromosómica generados por el modelo optimizado para tres individuos representativos. (a) Individuo africano (HG02573) muestra ancestría predominantemente africana (98.6 % verde) con mínimos segmentos europeos (1.3 % azul) y asiáticos (0.2 % naranja). (b) Individuo europeo (HG00149) presenta ancestría mayoritariamente europea (97.5 % azul) con pequeñas proporciones africanas (0.7 % verde) y asiáticas (1.8 % naranja). (c) Individuo asiático (NA18532) exhibe ancestría predominantemente asiática (95.2 % naranja) con segmentos europeos (4.3 % azul) y mínimos africanos (0.5 % verde).

4.5.7. Rendimiento computacional

Además del rendimiento predictivo, es importante analizar cómo se distribuyen los parámetros del modelo final, ya que esto condiciona los requisitos de memoria y el costo computacional de entrenamiento e inferencia. A continuación se detalla la estructura paramétrica del modelo optimizado.

Parámetros por componente El modelo presenta la siguiente descomposición de parámetros por bloque funcional:

Tabla 4.7: Distribución de parámetros en el modelo optimizado

Componente	**Parámetros**	**Porcentaje**
Embeddings de nucleótidos	2048	0.005 %
FPE ($\alpha = 100$)	39 833 248	**96.30 %**
Embedding cromosómico	2944	0.007 %
Encoder Hyena (4 capas)	1 462 400	3.54 %
MLP clasificador	66 307	0.16 %
Total	41 366 563	**100 %**

Observaciones De esta distribución se desprenden varios puntos relevantes:

- **Dominancia de FPE**: El 96.3 % de los parámetros corresponden a los *Factorized Positional Embeddings*. El costo paramétrico del modelo está fuertemente concentrado en la codificación posicional optimizada.
- **Eficiencia del encoder**: Las 4 capas HyenaDNA representan sólo el 3.54 % de los parámetros totales. Esto ilustra la eficiencia de la arquitectura subcuadrática: la mayor parte de la capacidad paramétrica no reside en el mecanismo de atención/convolución, sino en la representación posicional.
- **Clasificador compacto**: El MLP final utiliza únicamente 66k parámetros (0.16 %), manteniendo la capa de decisión simple en comparación con el bloque de codificación posicional. Esto sugiere que la mayor parte del poder representacional proviene del encoder y de FPE, mientras que el clasificador actúa principalmente como una capa de lectura.

En conjunto, estos resultados muestran que el modelo logra alto rendimiento con una arquitectura de cómputo relativamente ligera en el encoder y el clasificador.

Eficiencia en inferencia Además de la eficiencia en entrenamiento, el modelo demuestra alta velocidad en inferencia. Un genoma completo, procesado mediante aproximadamente 1600 ventanas deslizantes a lo largo de los 22 cromosomas autosómicos, se analiza en aproximadamente 15 segundos utilizando

el hardware descrito previamente, lo que equivale a ∼106 ventanas por segundo. Esta velocidad permite aplicaciones de detección de ancestría en tiempo real y análisis a gran escala de cohortes poblacionales.

4.6. Comparación HyenaDNA vs Transformers

Una vez fijada la configuración óptima de codificación posicional y factores de escala, se realizó un experimento comparativo entre *HyenaDNA* y una arquitectura basada en *Transformers* para cuantificar las ventajas de los modelos subcuadráticos en la tarea de detección de ancestría genética.

La comparación utiliza la misma configuración optimizada identificada en los experimentos anteriores: *Factorized Positional Embedding* FPE($\alpha = 100$), codificación cromosómica por *embedding* y entrenamiento en dos etapas sobre el dataset expandido de 590 individuos.

4.6.1. Diseño experimental

Arquitecturas evaluadas Para garantizar una comparación justa, se construyeron dos modelos con configuraciones equivalentes en términos de tamaño y datos de entrada. La Tabla 4.8 resume las configuraciones arquitecturales:

Tabla 4.8: Configuraciones arquitecturales comparadas

Modelo	Parámetros	Contexto	Embedding dim	Bloques
HyenaDNA	41.23M	1024	128	4
Transformer	41.15M	1024	128	4

Ambos modelos utilizan ∼ 41M de parámetros, longitud de contexto 1024 tokens, dimensión de embedding 128 y 4 bloques. En los dos casos, el **Factorized Positional Embedding (FPE)** concentra la mayoría de los parámetros (∼ 39.8M). Si se excluye FPE, HyenaDNA tiene 1.39M parámetros y el Transformer 1.32M, una diferencia ligera atribuible a los componentes específicos del *HyenaOperator* (filtros Hyena y modulación exponencial).

Diferencias de diseño Aunque el tamaño total de parámetros es comparable, los mecanismos internos son fundamentalmente distintos.

En el **Transformer**, el bloque básico se organiza alrededor de *multi-head attention* (MHA), con complejidad temporal y espacial cuadrática $\mathcal{O}(L^2)$ en la longitud de secuencia L. Cada capa aplica proyecciones lineales para construir matrices $\mathbf{Q}$, $\mathbf{K}$ y $\mathbf{V}$ por cabeza de atención, calcula el producto escalar $\mathbf{QK}^\top$ escalado, aplica *softmax* y combina los valores $\mathbf{V}$. Esta formulación permite dependencia global explícita entre posiciones, pero a costa de un uso de memoria que crece como L^2, lo que se convierte en un cuello de botella para secuencias largas.

En **HyenaDNA**, el mecanismo central es el *HyenaOperator*, cuyo costo es subcuadrático $\mathcal{O}(L \log L)$, al reemplazar la atención explícita por convoluciones implícitas largas evaluadas con FFT. Los filtros de convolución se generan dinámicamente mediante MLPs, reduciendo el número de parámetros de L a $M \ll L$. Además, la señal pasa por compuertas multiplicativas jerárquicas (matrices diagonales $D_{x_1}, D_{x_2}, \dots$) que modulan la información por canal. La alternancia entre filtrado convolucional y *gating* multiplicativo da lugar a un rango receptivo global con mejor comportamiento en memoria que la atención cuadrática y se adapta bien a la paralelización en GPU.

Protocolo de entrenamiento Las dos arquitecturas se entrenaron bajo el mismo protocolo, con el objetivo de aislar el efecto del mecanismo secuencial:

- **Entrada**: secuencias de SNPs con FPE($\alpha = 100$) y codificación cromosómica por embedding.
- **Batch size**: 64, elegido para que el Transformer pueda entrenarse dentro de los límites de memoria de la GPU.
- **Estrategia**: entrenamiento en dos etapas:
  1. Predicción del siguiente token (*next-token prediction*).
  2. *Fine-tuning* para clasificación de ancestría.

La comparación se restringe a secuencias de longitud 1024 debido a las limitaciones de memoria del Transformer en este entorno de hardware.

4.6.2. Convergencia y rendimiento predictivo

Convergencia en predicción del siguiente token La Figura 4.9 muestra la evolución de la perplejidad en validación durante el entrenamiento de predicción del siguiente SNP para ambas arquitecturas.

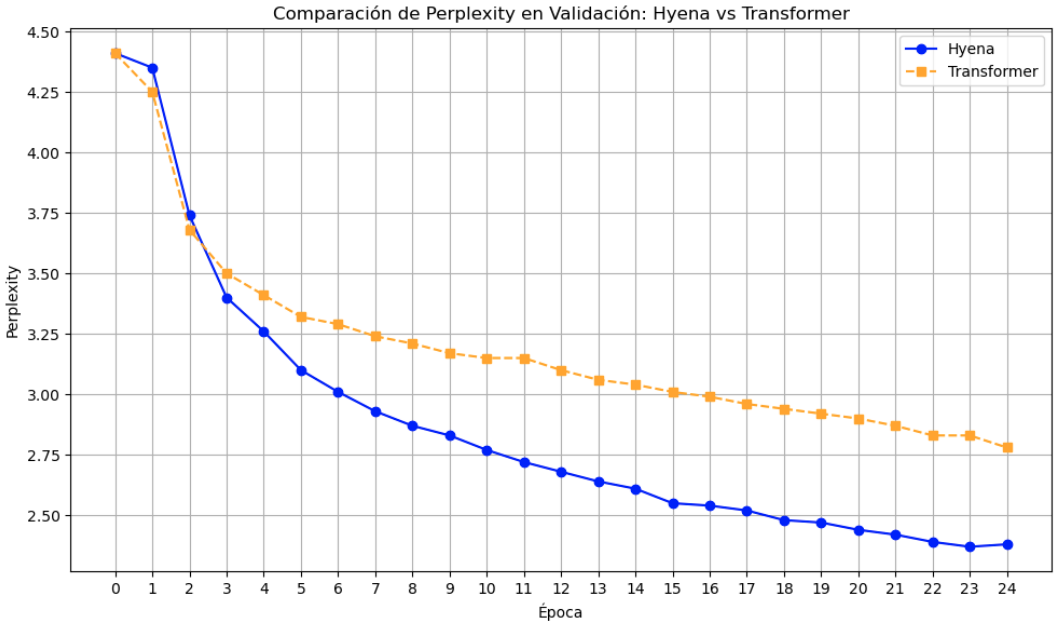


Figura 4.9: Evolución de la perplejidad en validación durante el entrenamiento de predicción del siguiente token. HyenaDNA (azul) converge más rápido y a valores de perplejidad inferiores que Transformer (naranja).

Los resultados pueden resumirse de la siguiente forma:

- **Velocidad de convergencia**: HyenaDNA reduce la perplejidad de forma más rápida, especialmente en las primeras 20 épocas.
- **Perplejidad final**: HyenaDNA converge a 2.38, mientras que Transformer alcanza 2.78 en el mismo número de épocas. El Transformer requiere 22 épocas adicionales para alcanzar la perplejidad de HyenaDNA ($\sim$2.38).
- **Estabilidad**: Ambos modelos muestran curvas suaves, sin oscilaciones grandes, lo que indica que el proceso de optimización es estable en ambos casos.

Combinando esta convergencia más rápida con el menor tiempo por época, HyenaDNA logra el mismo nivel de perplejidad con un costo de entrenamiento significativamente menor.

Rendimiento en clasificación de ancestría Para la comparación de clasificación, se extendió el entrenamiento de predicción del siguiente token del Transformer durante 22 épocas adicionales, de modo que ambos modelos partieran de una perplejidad similar ($\sim$2.38). La Figura 4.10 muestra la evolución

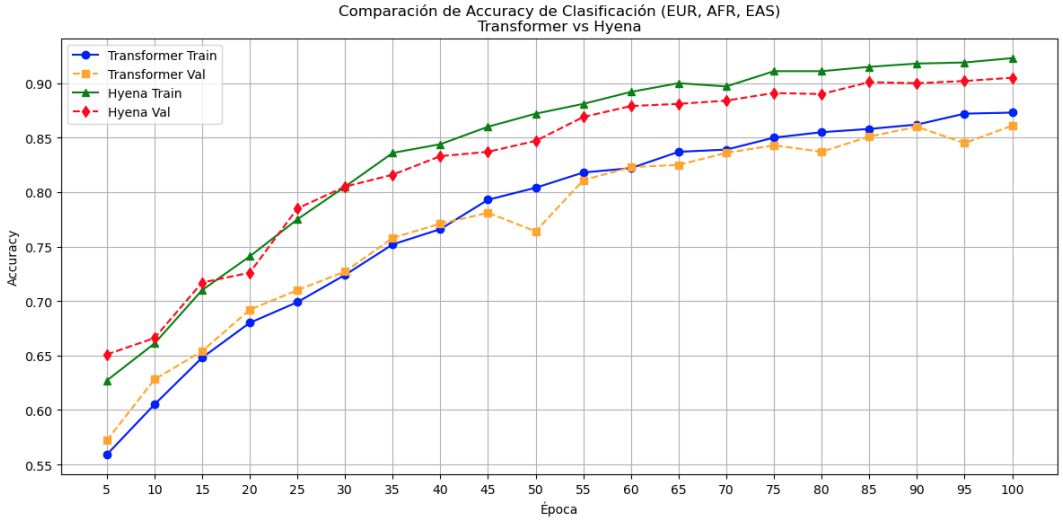


Figura 4.10: Evolución de la *accuracy* de clasificación durante el entrenamiento. HyenaDNA (líneas verde/roja) supera de forma consistente al Transformer (líneas azul/naranja) tanto en entrenamiento como en validación.

de la *accuracy* durante el *fine-tuning*.

Resultados de clasificación:

- **Accuracy final**: HyenaDNA alcanza ∼90 % de *accuracy*, frente a ∼86 % del Transformer.
- **Gap consistente**: A lo largo de todo el entrenamiento, HyenaDNA se mantiene aproximadamente 5 puntos porcentuales por encima.
- **Brecha train–validation**: Ambos modelos muestran brechas comparables entre entrenamiento y validación, lo que sugiere niveles de generalización similares.

Nota: Estos valores de *accuracy* (∼90 %) son inferiores al resultado final optimizado reportado en el experimento anterior (∼99 %). La diferencia se debe principalmente a los tiempos de entrenamiento y algunos hiperparámetros, por ejemplo el tamaño de batch, que fue ajustado en esta comparación para permitir el entrenamiento del modelo Transformer con la memoria disponible de GPU.

4.6.3. Eficiencia computacional

Predicción del siguiente token En la primera etapa (predicción del siguiente SNP), se observaron diferencias claras en uso de memoria y tiempo de entrenamiento por época:

Tabla 4.9: Eficiencia en entrenamiento de *next-token prediction*

Métrica	Transformer	HyenaDNA	Mejora
Memoria GPU (MB)	14 786	3940	3.75× **menor**
Tiempo por época	1:40	0:39	2.56× **más rápido**
Parámetros totales	41.15M	41.23M	≈ equivalente
Parámetros sin FPE	1.32M	1.39M	≈ equivalente

De la Tabla 4.9 se desprende que HyenaDNA utiliza sólo ∼26.6 % de la memoria GPU que requiere el Transformer, manteniendo un número de parámetros equivalente. Además, el tiempo por época es aproximadamente 2.5× menor, lo que reduce considerablemente el costo de explorar diferentes configuraciones.

***Fine-tuning* de clasificación** En la segunda etapa (clasificación de ancestría), las diferencias de eficiencia se mantienen:

Tabla 4.10: Eficiencia en *fine-tuning* de clasificación

Métrica	Transformer	HyenaDNA	Mejora
Memoria GPU (MB)	14 786	4400	3.36× **menor**
Tiempo por época	1:40	0:39	2.56× **más rápido**
Parámetros totales	41.15M	41.23M	≈ equivalente
Parámetros sin FPE	1.32M	1.39M	≈ equivalente

El hecho de que las arquitecturas subyacentes tengan prácticamente la misma capacidad paramétrica y, aun así, HyenaDNA resulte mucho más eficiente en memoria y tiempo, sugiere que las ventajas provienen del diseño subcuadrático del *HyenaOperator*, y no de una reducción de capacidad del modelo.

4.6.4. Escalabilidad con tamaño de batch

Para estudiar el comportamiento bajo distintas cargas, se evaluó HyenaDNA con tamaños de *batch* más grandes en la tarea de clasificación de ancestría:

Al cuadruplicar el *batch size* de 64 a 256, el tiempo por época se mantiene casi constante, mientras que el uso de memoria crece de forma esperable pero sigue dentro de los límites del hardware (12.6 GB). Esto indica que HyenaDNA aprovecha bien la paralelización en GPU y permite aumentar el tamaño de

Tabla 4.11: Impacto del tamaño de *batch* en HyenaDNA

Configuración	Batch 64	Batch 256	Relación
Memoria GPU (MB)	4400	12 612	2.87× **más**
Tiempo por época	0:39	0:40	≈ equivalente

batch para acelerar la recolección de gradientes sin penalizaciones significativas en tiempo por época.

4.6.5. Escalabilidad de longitud máxima de contexto

Para caracterizar los límites de cada arquitectura, evaluamos la longitud máxima de secuencia que puede procesarse con *batch size* de 1 en una GPU de 16 GB de memoria.

Tabla 4.12: Longitud máxima de contexto por arquitectura

Modelo	SNPs máximos	Ratio
Transformer	∼8500	1×
HyenaDNA	∼210 000	**24**×

El Transformer alcanza su límite de memoria a aproximadamente 8500 SNPs, mientras que HyenaDNA puede procesar hasta 210 000 SNPs en el mismo hardware, una ventaja de 24× en longitud máxima de contexto. Esta diferencia demuestra que para aplicaciones genómicas que requieren contextos extendidos —como el análisis de cromosomas completos (50k–350k SNPs) o la detección de interacciones regulatorias de largo alcance— las arquitecturas subcuadráticas no son solo más eficientes sino a menudo **necesarias**.

4.6.6. Discusión comparativa

Los experimentos controlados muestran una ventaja consistente de HyenaDNA frente al Transformer en el contexto de genómica computacional, incluso cuando se igualan la capacidad paramétrica y la codificación posicional. A igualdad de longitud de secuencia y número de parámetros, HyenaDNA utiliza varias veces menos memoria GPU, reduce el tiempo por época aproximadamente a la mitad, alcanza menor perplejidad en menos épocas y obtiene una *accuracy* de clasificación de ancestría superior. Estas diferencias se man-

tienen tanto en la fase de predicción del siguiente SNP como en el *fine-tuning* para clasificación.

Implicaciones para genómica computacional En conjunto, estos resultados indican que las arquitecturas subcuadráticas como HyenaDNA constituyen una alternativa práctica a Transformers para modelar secuencias genómicas largas. Permiten entrenar modelos competitivos en GPUs con memoria limitada, reducen el tiempo de entrenamiento por experimento y facilitan la exploración de arquitecturas y configuraciones de codificación más variadas. Además abren la puerta a longitudes de contexto y tamaños de dataset mayores, lo cual es muy importante en genómica computacional.

4.7. Análisis de embeddings posicionales y representaciones ocultas

Para comprender cómo *HyenaDNA* procesa y representa información ancestral, se analizaron las representaciones internas del modelo. Este análisis estudia tanto los *embeddings* posicionales como los *hidden states* generados durante la inferencia, con el objetivo de caracterizar la organización interna del conocimiento ancestral aprendido.

4.7.1. Metodología de análisis

Configuración experimental Para el análisis de representaciones se utilizó la configuración óptima identificada en experimentos previos: un modelo HyenaDNA con contexto 1k y FPE($\alpha = 100$), entrenado sobre el dataset expandido de 590 individuos pertenecientes a tres poblaciones continentales. El análisis se llevó a cabo sobre 5000 secuencias sorteadas aleatoriamente, cada una de longitud 1024 SNPs.

Técnicas de reducción dimensional Las representaciones de alta dimensión se proyectaron a espacios bidimensionales mediante dos técnicas complementarias: *Principal Component Analysis* (PCA), que aplica una proyección ortogonal sobre los ejes (componentes principales) que maximizan la varianza explicada, y *t-distributed Stochastic Neighbor Embedding* (t-SNE), que preserva

principalmente la estructura local y resulta especialmente útil para visualizar clusters.

Niveles de representación analizados Se extrajeron y compararon distintas capas de representación:

- **Embeddings posicionales**: vectores generados por el *Factorized Positional Embedding* (FPE).
- **Representaciones internas**: *hidden states* finales del encoder, antes de la capa de clasificación.
- **Salidas del modelo**: probabilidades de salida por clase ancestral.

Esta combinación permite estudiar conjuntamente el espacio posicional, las representaciones internas y las decisiones de clasificación resultantes.

4.7.2. Embeddings posicionales

El análisis PCA de los *factorized positional embeddings* muestra las propiedades globales del espacio posicional aprendido:

Las primeras componentes principales de la figura 4.11 no presentan clusters separados por cromosoma, sino una distribución aproximadamente uniforme en torno a una estructura circular. Posiciones cercanas en el genoma se mapean a vectores cercanos en el espacio latente, lo que indica que FPE aprende un mapa posicional suave y continuo.

4.7.3. Representaciones durante la predicción del siguiente SNP

Durante la primera etapa del entrenamiento en dos fases (Subsec. 4.5.4), el modelo se entrena para predecir el siguiente SNP en secuencias de 1024 tokens. Esta etapa define un modelo de lenguaje genómico auto-supervisado. El análisis de los *hidden states* en este punto permite entender qué estructura desarrolla el modelo cuando su objetivo es predecir nucleótidos individuales.

Organización por nucleótido objetivo La Figura 4.12 muestra una proyección t-SNE de los *hidden states* asociados a la predicción del siguiente token, coloreados según el nucleótido objetivo (A, C, G, T).

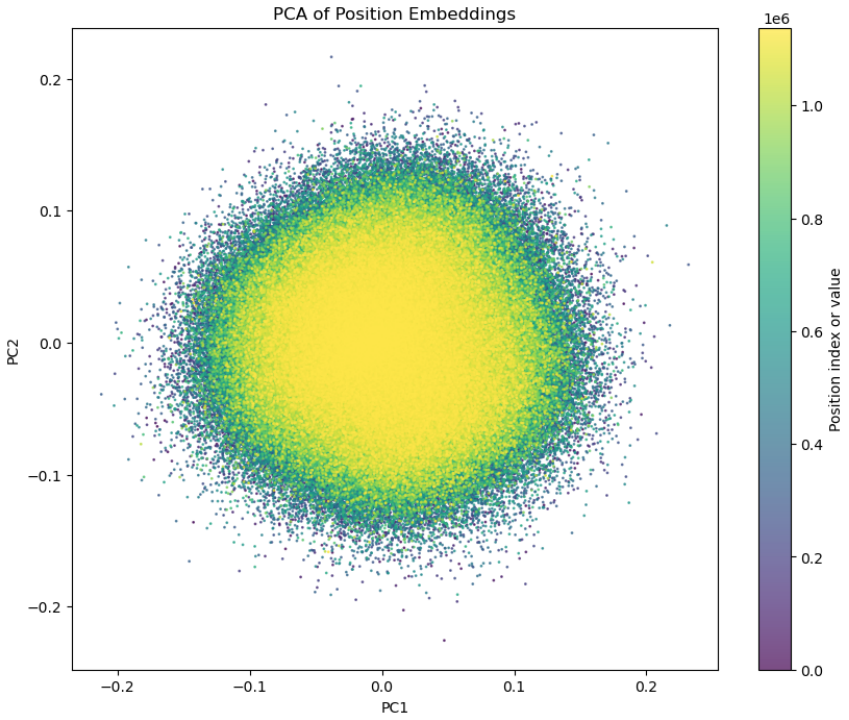


Figura 4.11: Visualización PCA de *factorized positional embeddings*. Cada punto corresponde a una posición genómica, coloreada por su valor de posición. La distribución presenta una estructura aproximadamente circular y suave en las primeras componentes principales.

Las representaciones adoptan una estructura en estrella, con cuatro regiones bien definidas asociadas a cada nucleótido. El centro de la figura concentra muestras en las que el contexto previo no permite una predicción clara, mientras que los puntos más alejados del centro corresponden a contextos que permiten predicciones más confiables. Las transiciones suaves entre las puntas de la estrella indican que el modelo organiza las secuencias en un espacio continuo de probabilidad sobre los cuatro nucleótidos.

Implicaciones del preentrenamiento Esta geometría refleja varios aspectos del aprendizaje auto-supervisado: el modelo desarrolla representaciones especializadas para los contextos que preceden a cada nucleótido, de manera análoga a cómo un modelo de lenguaje natural diferencia contextos que favorecen distintas palabras. La separación clara de regiones por nucleótido indica que el preentrenamiento capta dependencias locales relevantes para la secuencia genómica y proporciona una base informativa para la etapa de *fine-tuning*. En otras palabras, el modelo aprende a organizar el espacio de secuencias según

Figura 4.12: Visualización t-SNE de *hidden states* durante la predicción del próximo token. Cada punto representa una secuencia, coloreada por el próximo nucleótido a predecir: Adenina (A, azul), Citosina (C, naranja), Guanina (G, verde) y Timina (T, rojo).

el "siguiente nucleótido probable", y esa organización actúa luego como punto de partida para la tarea de clasificación de ancestría.

4.7.4. Representaciones del modelo de clasificación de ancestría

Clusters poblacionales tras el *fine-tuning* El análisis de los *hidden states* finales del modelo ajustado para clasificación muestra una reorganización completa del espacio de representación. La Figura 4.13 presenta una proyección t-SNE de estas representaciones, coloreadas por ancestría verdadera (AFR, EAS, EUR).

En contraste con la organización en estrella por nucleótidos, aquí se observan tres clusters poblacionales bien separados y con mínima superposición. El modelo conserva su capacidad para agrupar representaciones de forma coherente, pero ahora el criterio de agrupamiento está alineado con la tarea de

Figura 4.13: Visualización t-SNE de *hidden states* finales del modelo *fine-tuned* para clasificación de ancestría. Los puntos representan secuencias individuales coloreadas por ancestría: África (AFR, verde), Asia del Este (EAS, naranja) y Europa (EUR, azul).

clasificación continental. La baja intersección entre clusters coincide con el alto rendimiento cuantitativo obtenido en las métricas de clasificación.

Geometría inter-poblacional La disposición relativa de los clusters también aporta información cualitativa. Los clusters de África y Europa presentan zonas de proximidad, al igual que Europa y Asia, mientras que África y Asia se mantienen más alejados sin regiones de transición directa. En el espacio latente aprendido, Europa ocupa una posición intermedia que actúa como "puente" entre las otras dos poblaciones. Este patrón es compatible con una interpretación genética y geográfica (Europa conectando físicamente África y Asia y habiendo servido como corredor migratorio), pero también puede ser simplemente la forma natural en que el modelo organiza tres clases con distintos grados de similitud estadística. Distinguir entre estos escenarios requeriría análisis adicionales específicos de estructura poblacional.

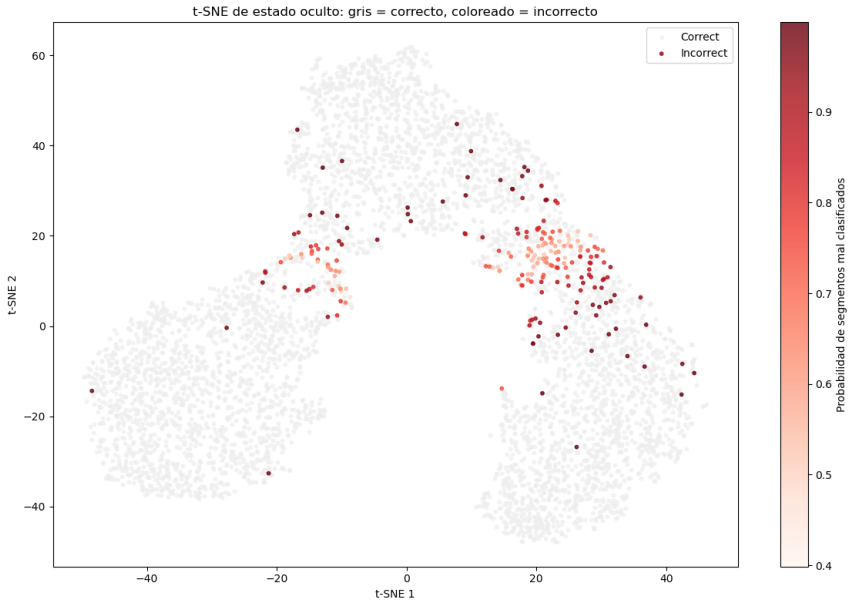


Figura 4.14: Mapa de confianza y errores en el espacio t-SNE. Los puntos grises indican predicciones correctas; los puntos rojos corresponden a muestras mal clasificadas. La intensidad refleja la probabilidad asignada a la clase predicha incorrectamente.

4.7.5. Relación entre representaciones y confianza del modelo

Para estudiar cómo se relaciona la posición en el espacio de representaciones con el comportamiento del clasificador, se proyectaron sobre el t-SNE final las predicciones correctas e incorrectas, junto con la confianza (*softmax*) asignada a la clase predicha.

Las muestras mal clasificadas se concentran cerca de las fronteras entre clusters, justo donde las distribuciones de probabilidad entre clases son menos extremas. En cambio, las regiones centrales de cada cluster están dominadas por predicciones correctas con alta confianza. Esta correlación entre distancia al centro del cluster, nivel de confianza y tasa de acierto indica que la geometría del espacio latente es informativa: la posición de una muestra proporciona una estimación razonable de la incertidumbre del modelo sin necesidad de calibración adicional.

4.7.6. Discusión

El análisis de las representaciones internas de HyenaDNA muestra una transición clara a lo largo del entrenamiento en dos etapas. Durante el preentrenamiento, el modelo organiza las secuencias en torno al nucleótido siguiente, generando una estructura en estrella por (A, C, G, T). Tras el *fine-tuning* para clasificación de ancestría, este mismo espacio se reorganiza en tres clusters poblacionales bien definidos (AFR, EAS, EUR). Esta evolución ilustra cómo el *transfer learning* en genómica puede reutilizar representaciones aprendidas para una tarea auto-supervisada y adaptarlas eficazmente a una tarea supervisada de mayor nivel.

La geometría del espacio de *hidden states* es además interpretable: las muestras cercanas al centro de cada cluster tienden a asociarse con predicciones de alta confianza, mientras que las situadas en regiones de frontera acumulan la mayoría de los errores. Esta relación entre distancia a los clusters y precisión del modelo proporciona un criterio natural para identificar casos ambiguos que podrían requerir revisión o análisis adicional, sin introducir mecanismos específicos de calibración.

Por último, la organización relativa de las poblaciones en el espacio latente sugiere patrones de similitud entre AFR, EUR y EAS que son plausibles desde el punto de vista genético y geográfico, aunque confirmar esta correspondencia requeriría análisis poblacionales dedicados. En conjunto, los resultados respaldan las decisiones arquitecturales adoptadas: los *factorized positional embeddings* proporcionan un espacio posicional suave y rico, el entrenamiento en dos etapas facilita la transferencia de conocimiento desde predicción de nucleótidos hacia clasificación de ancestría y HyenaDNA muestra capacidad para formar y reorganizar representaciones coherentes. Además de alcanzar alta precisión, el modelo produce representaciones internas que son estructuradas e interpretables, lo que agrega valor para el análisis de datos genómicos más allá de las métricas de rendimiento estándar.

4.8. Conclusiones del capítulo

Este capítulo presentó una progresión experimental sistemática que validó la efectividad de *HyenaDNA* en tareas de clasificación genómica y detección de ancestría. Los resultados demuestran que la combinación de filtrado de SNPs,

codificación posicional avanzada y arquitectura subcuadrática constituye un sistema eficiente y preciso para el análisis genómico a gran escala.

La configuración final combina SNPs filtrados, embeddings posicionales factorizados FPE($\alpha = 100$), codificación cromosómica y entrenamiento en dos etapas sobre arquitectura *HyenaDNA*. Como muestra la Tabla 4.5, el rendimiento escala con la longitud de contexto, alcanzando 99.89 % de *accuracy* con 32 768 SNPs y superando a métodos especializados del estado del arte como Gnomix (Hilmarsson et al., 2022). La comparación con Transformers equivalentes (Tabla 4.9, Tabla 4.12) muestra que HyenaDNA ofrece ventajas tanto en eficiencia computacional como en escalabilidad, con hasta 24× más capacidad de contexto.

Capítulo 5

Consideraciones finales

Este capítulo resume los resultados de la investigación, evalúa el cumplimiento de los objetivos planteados y discute el alcance de aplicar HyenaDNA a tareas de clasificación genómica, con foco en detección de ancestría. También se presentan las principales limitaciones del trabajo y posibles líneas de investigación futura.

5.1. Síntesis de la investigación

Esta tesis estudió el uso de HyenaDNA, una arquitectura subcuadrática de contexto largo, para tareas de clasificación genómica, con énfasis en la detección de ancestría. El trabajo se organizó como una progresión experimental que permitió aislar los componentes claves para obtener buenos resultados en este tipo de problemas.

5.1.1. Progresión experimental

Los experimentos siguieron una secuencia que permitió ir refinando el diseño. La Tabla 5.1 resume la evolución de los resultados a lo largo de los cinco experimentos realizados.

El punto de partida fue la clasificación de especies (Tabla 4.1), que validó la capacidad de HyenaDNA para resolver tareas genómicas estándar. El intento de detección de ancestría con secuencias completas produjo rendimiento aleatorio, evidenciando que el enfoque "todo el genoma" sin selección de posiciones no era adecuado. El uso de solo SNPs aportó una mejora marginal, todavía insuficiente para la tarea.

Tabla 5.1: Progresión experimental: *accuracy* obtenida en cada experimento.

Exp.	Enfoque	Accuracy
1	Clasificación de especies	0.92
2	Ancestría con secuencias completas	∼0.33
3	Ancestría con solo SNPs	<0.50
4	SNPs con contexto posicional (32k)	∼0.93
5	Optimizaciones sistemáticas (4096 SNPs)	0.9934

La incorporación de SNPs con contexto posicional y cromosómico marcó el punto de inflexión, con una mejora sustancial al aumentar la longitud de contexto (Tabla 4.2). Las optimizaciones sistemáticas —dataset expandido, codificación posicional refinada, arquitectura mejorada y entrenamiento en dos etapas— llevaron a un modelo final con *accuracy* superior al 99 % (Tabla 4.5).

Finalmente, la comparación controlada con Transformers equivalentes mostró ventajas claras de HyenaDNA tanto en eficiencia computacional (Tabla 4.9, Tabla 4.10) como en escalabilidad (Tabla 4.12).

5.2. Cumplimiento de objetivos

5.2.1. Objetivos generales

Los objetivos generales planteados en el Capítulo 1 se cumplieron de forma satisfactoria:

- **Comprender HyenaDNA en el contexto genómico**: se analizó la arquitectura subcuadrática, el operador Hyena y su comportamiento específico en genómica, destacando el papel central de los *embeddings* posicionales.
- **Aplicar HyenaDNA a tareas genómicas**: se implementó HyenaDNA para clasificación de especies (92 % de *accuracy*) y detección de ancestría continental (∼99 % de *accuracy*), mostrando que la arquitectura es efectiva en problemas genómicos reales.
- **Realizar una evaluación comparativa**: se llevó a cabo una comparación controlada con Transformers, demostrando ventajas en uso de memoria, tiempo de entrenamiento y *accuracy* final en condiciones equivalentes.

- **Analizar la escalabilidad**: se verificó la capacidad de procesar secuencias largas (hasta 32k tokens en los experimentos, con soporte teórico hasta 1M) manteniendo eficiencia computacional gracias a la complejidad subcuadrática del operador Hyena.

5.2.2. Objetivos específicos

Los objetivos específicos también fueron alcanzados:

- **Clasificación de especies**: HyenaDNA alcanzó 92 % de *accuracy*, lo que valida su aplicación en un escenario genómico estándar.
- **Detección de ancestría**: se logró 99.34 % de *accuracy* para clasificación de ancestría continental con secuencias de 4096 SNPs.
- **Evaluación con distintas longitudes de contexto**: se observó una mejora consistente del rendimiento al aumentar el contexto, pasando de 68 % a 83 % y luego a 93 % de *accuracy* con contextos de 128, 1k y 32k tokens, respectivamente (usando configuraciones no optimizadas).

5.3. Contribuciones de la investigación

La investigación aporta resultados en tres planos: metodológico, técnico e interpretativo, orientados al desarrollo de sistemas genómicos basados en arquitecturas subcuadráticas.

En el plano **metodológico**, se propone un *pipeline* concreto que combina SNPs y *Factorized Positional Embeddings*, junto con un protocolo de entrenamiento en dos etapas específico para SNPs. Se diseñó además una infraestructura de almacenamiento HDF5 escalable para manejar los datos genómicos.

En el plano **técnico**, se cuantificó el rol crítico de los *embeddings* posicionales (96.3 % de los parámetros totales), se mostró la complementariedad entre información alélica y posicional mediante experimentos controlados y se midieron las ventajas de HyenaDNA frente a Transformers tanto en eficiencia como en rendimiento. También se documentó la mejora con contexto largo (68 % $\rightarrow$ 83 % $\rightarrow$ 93 %) para modelos no optimizados.

Finalmente, en el plano de **interpretabilidad**, se analizó la transformación del espacio de representaciones internas desde una organización por nucleótidos hacia clusters poblacionales tras el *fine-tuning*. Se mostró que la distancia en

el espacio latente correlaciona con la confianza de predicción, lo que aporta una lectura geométrica de la certeza del modelo.

5.4. Implicaciones de los resultados

5.4.1. Implicaciones para la genómica computacional

Los resultados de esta tesis apoyan la idea de que las arquitecturas subcuadráticas especializadas son particularmente adecuadas para secuencias genómicas largas. La eficiencia computacional de HyenaDNA permite entrenar modelos competitivos en GPUs con recursos limitados y hace viable trabajar con contextos que serían muy costosos para Transformers estándar. La combinación de operador Hyena y FPE ofrece ventajas que no se explican únicamente por el conteo de parámetros, sino por el diseño específico del mecanismo secuencial.

La capacidad de procesar hasta 210k tokens en los experimentos (y hasta 1M según Nguyen et al. (2023)) abre la puerta a análisis que requieren contexto extenso, como secuencias cromosómicas completas. La estrategia de usar SNPs filtrados y luego contextualizarlos posicionalmente se perfila como un patrón reutilizable en otros dominios donde la señal es dispersa en secuencias largas pero necesita contexto espacial para ser interpretada.

5.4.2. Implicaciones para genética de poblaciones

En el ámbito de genética de poblaciones, la precisión alcanzada ($\sim$99 % de *accuracy* en ancestría continental) sugiere que modelos como HyenaDNA pueden servir como herramientas complementarias a métodos tradicionales de detección de ancestría. No obstante, el estudio está limitado a tres grandes grupos continentales derivados de 1000 Genomes. Antes de pensar en aplicaciones más amplias, será necesario validar el enfoque en poblaciones adicionales, escenarios de ancestría mixta compleja y condiciones de muestreo más diversas.

5.5. Limitaciones del estudio

5.5.1. Limitaciones metodológicas

El trabajo presenta varias limitaciones que condicionan el alcance de las conclusiones:

- **Cobertura poblacional**: el análisis se restringe a tres poblaciones continentales (Europa, África, Asia). Extender el estudio a subpoblaciones, mezclas más finas de ancestría y otros continentes requiere trabajo adicional.
- **Rango de contexto explorado**: los experimentos se limitaron a contextos de hasta 32k tokens. La prueba con contexto de 210k se limitó solamente a evaluar la capacidad del modelo con el hardware disponible, sin profundizar en los resultados de este experimento. Entrenamientos completos con contextos más largos (32k–1M) podrían revelar patrones adicionales o nuevos cuellos de botella computacionales.
- **Comparación con Transformers**: la comparación empírica se realizó únicamente con secuencias de 1k tokens debido a restricciones de memoria para la arquitectura Transformer, por lo que no se midieron directamente las diferencias en el régimen de contexto donde HyenaDNA es más ventajoso.

5.5.2. Limitaciones técnicas

Desde el punto de vista técnico, las principales limitaciones son:

- **Dependencia de FPE**: los *Factorized Positional Embeddings* concentran 96.3 % de los parámetros (39.8M de 41M) y su eliminación colapsa el rendimiento. Esto sugiere la necesidad de explorar variantes más eficientes en memoria.
- **Sensibilidad al factor de escala**: el factor α en FPE requiere un balance cuidadoso entre resolución posicional y coste computacional. En este trabajo, $\alpha = 100$ resultó óptimo, pero no es evidente que esta elección generalice a otros datasets o tareas.
- **Escalado exponencial de memoria FPE**: la reducción del factor α mejora el rendimiento pero aumenta la memoria de forma exponencial

(100× aumento de $\alpha = 1000$ a $\alpha = 10$). Esto limita la exploración de resoluciones posicionales muy finas en hardware con memoria limitada.

- **Límites de escalabilidad del Transformer**: experimentos de escalabilidad máxima (*batch size* 1) muestran que el Transformer alcanza su límite a ∼8500 SNPs, mientras que HyenaDNA escala a ∼210 000 SNPs (ventaja de 24×). Sin embargo, no se realizaron entrenamientos completos en estos regímenes extremos para verificar si las ventajas se mantienen.

5.6. Direcciones de trabajo futuro

5.6.1. Extensiones inmediatas

Una extensión natural es aumentar la complejidad del problema de ancestría. Esto incluye escalar a 10–20 grupos ancestrales, incorporar subpoblaciones y estimar componentes de ancestría mixta de forma cuantitativa. Finalmente, sería útil evaluar de forma sistemática contextos más largos (64k–1M tokens) y caracterizar con mayor detalle los *trade-offs* entre rendimiento, memoria y tiempo de entrenamiento.

Codificaciones posicionales alternativas Dado que FPE concentra 96.3 % de los parámetros del modelo (39.8M de 40.9M), desarrollar codificaciones más eficientes en memoria que mantengan el poder expresivo de FPE constituye una prioridad de investigación. Los SNPs aparecen aproximadamente cada 600–700 pares de bases, lo que significa que solo una fracción de las posiciones codificadas se utilizan realmente. Codificaciones que exploten esta dispersión —por ejemplo, mediante factorizaciones adicionales, codificaciones relativas, o embeddings continuos basados en la posición física real— podrían reducir los requisitos de memoria manteniendo la capacidad discriminativa.

5.6.2. Aplicaciones avanzadas

En un plano más aplicado, HyenaDNA podría utilizarse para procesar cromosomas completos, por ejemplo secuencias de 50k–350k SNPs por cromosoma, evitando ventanas deslizantes y capturando interacciones de más largo alcance. Otra dirección es la predicción de fenotipos complejos, donde el con-

texto cromosómico amplio podría ayudar a modelar interacciones de largo alcance. También resulta interesante explorar tareas de regresión en coordenadas geodésicas, utilizando un dataset con poblaciones globalmente distribuídas, para estudiar si el modelo puede mapear directamente genotipos a coordenadas geográficas.

5.6.3. Large Genomic Model (LGM)

A más largo plazo, los resultados apuntan hacia el desarrollo de un *Large Genomic Model* (LGM) especializado en SNPs, análogo a los grandes modelos del lenguaje natural (LLM), pero adaptado al dominio genómico. Un modelo de este tipo podría preentrenarse en grandes volúmenes de datos y luego ajustarse a tareas específicas (ancestría, fenotipos, detección de variantes raras, etc.). En este contexto, sería interesante investigar el uso de *soft prompts* y otras técnicas de adaptación sobre modelos preentrenados en SNPs como proponen los autores de HyenaDNA (Nguyen et al., 2023).

5.7. Reflexiones finales

5.7.1. Hallazgo principal

El resultado más claro de esta tesis es la **importancia crítica del embedding posicional** en tareas genómicas basadas en SNPs. Su eliminación reduce el rendimiento a niveles aleatorios, independientemente de la calidad del modelo base. De forma complementaria, los experimentos que utilizan solo posiciones sin información alélica también colapsan el rendimiento, lo que indica que la información posicional y la información de alelos son ambas necesarias. La conclusión es que, cuando se trabaja con SNPs, la codificación posicional es un requerimiento indispensable del modelo.

5.7.2. Validación de arquitecturas subcuadráticas

La comparación directa con Transformers de tamaño equivalente mostró que HyenaDNA ofrece ventajas significativas en eficiencia computacional (hasta 3.75× menos memoria GPU y 2.56× más velocidad por época) y también en rendimiento (90 % vs 86 % de *accuracy* en el escenario comparativo). Estas ventajas aparecen incluso en contextos relativamente cortos (1k tokens), donde

ambos modelos son entrenables. En el régimen de contextos largos (32k–1M tokens), donde los Transformers estándar dejan de ser viables por restricciones de memoria, es razonable esperar que las diferencias, principalmente en memoria, sean aún mayores.

5.7.3. Metodología transferible

El *pipeline* desarrollado —basado en SNPs filtrados y codificación posicional— resultó más efectivo que el procesamiento directo de secuencias completas. Este patrón es conceptualmente transferible a otros dominios con señal dispersa en secuencias extensas que requieren contexto espacial y dependencias de largo alcance. El entrenamiento en dos etapas confirma, además, el valor del *transfer learning* específico de dominio: un modelo de lenguaje genómico preentrenado puede servir como base sólida para tareas supervisadas posteriores.

5.7.4. Cierre

En conjunto, esta tesis muestra que HyenaDNA es una arquitectura adecuada para la detección de ancestría genética, capaz de alcanzar $\sim$99 % de precisión con un uso eficiente de recursos y con representaciones internas que admiten, en cierta medida, una interpretación geométrica. Las arquitecturas subcuadráticas de contexto largo se presentan como alternativas viables a Transformers en el dominio genómico, donde las dependencias de largo alcance son la regla.

Los resultados obtenidos y las líneas futuras propuestas señalan un camino hacia modelos genómicos de propósito general que aprovechen estas propiedades para acercar análisis avanzados a más grupos de investigación y acelerar el descubrimiento en genómica computacional.

Referencias bibliográficas

Avsec, v., Agarwal, V., Visentin, D., Ledsam, J. R., Grabska-Barwińska, A., Taylor, K. R., Assael, Y., Jumper, J., Kohli, P., and Kelley, D. R. (2021). Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203.

Brigham, E. O. (1988). *The Fast Fourier Transform and its Applications*. Prentice Hall, Englewood Cliffs, NJ.

Collette, A. (2013). *Python and HDF5: Unlocking Genomic and Large-Scale Data*. O'Reilly Media, Inc.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., et al. (2011). The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158.

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., and Li, H. (2021). Twelve years of samtools and bcftools. *GigaScience*, 10(2):giab008.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359.

Dao, T., Fu, D. Y., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. (2023). Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941. PMLR.

Falcon, W. and The PyTorch Lightning team (2019). Pytorch lightning.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

Hilmarsson, H., Kumar, A. S., Siu, R., Yooseph, S., Sandhu, R., and Sankararaman, S. (2022). High resolution ancestry deconvolution for next generation genomic data. *Nature Communications*, 13(1):2149.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Ji, Y., Zhou, Z., Liu, H., and Davuluri, R. V. (2021). Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120.

Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing*. Prentice Hall, 3 edition. Disponible en https://web.stanford.edu/~jurafsky/slp3/.

Kelley, D. R. (2020). Cross-species regulatory sequence activity prediction. *PLoS Computational Biology*, 16(7):e1008050.

Kelley, D. R., Reshef, Y. A., Bileschi, M., Belanger, D., McLean, C. Y., and Snoek, J. (2018). Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Research*, 28(5):739–750.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Maples, B. K., Gravel, S., Kenny, E. E., and Bustamante, C. D. (2013). Rfmix: a discriminative modeling approach for rapid and robust local-ancestry inference. *The American Journal of Human Genetics*, 93(2):278–288.

McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61.

Microsoft (2015). Visual Studio Code.

Nguyen, E., Poli, M., Faizi, M., Thomas, A., Wornow, M., Birch-Sykes, C., Massaroli, S., Patel, A., Rabideau, C., Bengio, Y., Ermon, S., Baccus, S. A., and Ré, C. (2023). Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in Neural Information Processing Systems*, 36.

Oppenheim, A. V., Schafer, R. W., and Buck, J. R. (1999). *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Patterson, N., Moorjani, P., Luo, Y., Mallick, S., Rohland, N., Zhan, Y., Genschoreck, T., Webster, T., and Reich, D. (2012). Ancient admixture in human history. *Genetics*, 192(3):1065–1093.

Patterson, N., Price, A. L., and Reich, D. (2006). Population structure and eigenanalysis. *PLoS genetics*, 2(12):e190.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. (2023). Hyena hierarchy: Towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28043–28078. PMLR.

Relling, M. V. and Evans, W. E. (2015). Pharmacogenomics in the clinic. *Nature*, 526(7573):343–350.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1715–1725.

Shirley, M. D., Ma, Z., Pedersen, B. S., and Wheelan, S. J. (2015). Efficient "pythonic" access to fasta files using pyfaidx. *PeerJ PrePrints*, 3:e1196.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

The 1000 Genomes Project Consortium (2015). A global reference for human genetic variation. *Nature*, 526(7571):68–74.

The HDF Group (1997–2025). Hierarchical Data Format, version 5.

The International HapMap Consortium (2005). A haplotype map of the human genome. *Nature*, 437(7063):1299–1320.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yadan, O. (2019). Hydra – a framework for elegantly configuring complex applications.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*.

Zheng, J., Ramasinghe, S., and Lucey, S. (2021). Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*.

APÉNDICES

Apéndice 1

Formatos de datos genómicos

Este apéndice describe los formatos de datos utilizados en esta tesis para el almacenamiento y procesamiento de información genómica.

1.1. Formato FASTA

El formato FASTA (Pearson and Lipman, 1988) es un estándar de texto plano para representar secuencias biológicas (ADN, ARN o proteínas). Cada entrada consta de una línea de encabezado que comienza con el carácter >, seguida del identificador y descripción de la secuencia, y a continuación las líneas con la secuencia de nucleótidos (típicamente en líneas de 70–80 caracteres).

El siguiente ejemplo muestra las primeras entradas de un archivo FASTA utilizado en los experimentos de clasificación (dataset de *enhancers*):

```
>***|0
AATTTTCTCATTTTCTCATAAAGTTTAACAGTTGTTTATTTGAGTCAGAATTCAAATAAGCTTC
TGTACATTACAATTGGTTTTAAGTTCTTATAAGACTCTATAGGTTTTCCCTTCATAATTTTTCT
TGCAATTTATTTGTTAAAGAAATTGGGTCATTTGTCCTATTGAGTGCTCCACTGTCTGTTTTTA
TTATTGTA
>***|0
ACTGGTTATCTTTTAGGACTAGTTAATATAACCCATTCTCTAACCAACAGATAACTCAACCAGG
TTCAGCACCTGATGGGTTACTCTTCAAGGACTCCCTTCTAAATCTCACTTTGCTGTGTCCACAA
TTCTAAATTGCTATACAATAGCATTTTCTCACTCTCATTCAGTATTTTACACAGAAAGATATGC
CTTGAACC
```

Cada entrada comienza con > seguido de un identificador y la clase (en

este caso `0` para no-*enhancer*). Las líneas siguientes contienen la secuencia de nucleótidos compuesta por los caracteres A, C, G y T.

1.2. Formato VCF (*Variant Call Format*)

El formato VCF (Danecek et al., 2011) es un estándar para almacenar variantes genéticas respecto de un genoma de referencia. El archivo se compone de líneas de metadatos (precedidas por `##`), una línea de encabezado de columnas (precedida por `#`), y las líneas de datos donde cada fila representa una variante. Las columnas principales son:

- **CHROM**: cromosoma donde se ubica la variante.
- **POS**: posición en el cromosoma (coordenada absoluta, base 1).
- **ID**: identificador de la variante (ej: rs12345).
- **REF**: alelo de referencia.
- **ALT**: alelo(s) alternativo(s).
- **QUAL**: calidad de la llamada de variante.
- **FILTER**: resultado de los filtros de calidad.
- **INFO**: información adicional en formato clave=valor.
- **FORMAT** y columnas de muestras: genotipos de cada individuo (ej: `0|1` indica un alelo de referencia y uno alternativo).

El siguiente ejemplo muestra un fragmento de un archivo VCF del cromosoma 14 de una población asiática del proyecto 1000 Genomes. Primero las últimas líneas de metadatos y encabezado, y luego las primeras variantes:

```
1  ##fileformat=VCFv4.3
2  ##FILTER=<ID=PASS,Description="All filters passed">
3  ##fileDate=27022019_15h52m43s
4  ##source=IGSRpipeline
5  ##reference=ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/
   GRCh38_reference_genome/GRCh38_full_analysis_set_plus_decoy_hla.fa
6  ##FORMAT=<ID=GT,Number=1,Type=String,Description="Phased Genotype">
7  ##contig=<ID=14>
8  ##INFO=<ID=AF,Number=A,Type=Float,Description="Estimated allele frequency
   in the range (0,1)">
9  ##INFO=<ID=AC,Number=A,Type=Integer,Description="Total number of alternate
    alleles in called genotypes">
10 ##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of samples with
   data">
11 ##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles
   in called genotypes">
```

```
12 ##INFO=<ID=EAS_AF,Number=A,Type=Float,Description="Allele frequency in the
    EAS populations calculated from AC and AN, in the range (0,1)">
13 ##INFO=<ID=EUR_AF,Number=A,Type=Float,Description="Allele frequency in the
    EUR populations calculated from AC and AN, in the range (0,1)">
14 ##INFO=<ID=AFR_AF,Number=A,Type=Float,Description="Allele frequency in the
    AFR populations calculated from AC and AN, in the range (0,1)">
15 ##INFO=<ID=AMR_AF,Number=A,Type=Float,Description="Allele frequency in the
    AMR populations calculated from AC and AN, in the range (0,1)">
16 ##INFO=<ID=SAS_AF,Number=A,Type=Float,Description="Allele frequency in the
    SAS populations calculated from AC and AN, in the range (0,1)">
17 ##INFO=<ID=VT,Number=.,Type=String,Description="indicates what type of
   variant the line represents">
18 ##INFO=<ID=EX_TARGET,Number=0,Type=Flag,Description="indicates whether a
   variant is within the exon pull down target boundaries">
19 ##INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth;
   some reads may have been filtered">
20 ##bcftools_viewVersion=1.21+htslib-1.21
21 ##bcftools_viewCommand=view -c1 -Ov -s NA18956 -o asia_chr14.vcf ALL.chr14
   .shapeit2_integrated_snvindels_v2a_27022019.GRCh38.phased.vcf; Date=Wed
   Oct 23 21:49:28 2024
22 #CHROM  POS     ID       REF      ALT      QUAL     FILTER  INFO     FORMAT
   NA18956
23 14      16076810        .        AC       A        .        PASS     AC=1;AN=2;
   DP=29388;AF=0.41;EAS_AF=0.38;EUR_AF=0.41;AFR_AF=0.42;AMR_AF=0.4;SAS_AF
   =0.44;VT=INDEL;NS=2548  GT       1|0
24 14      16093253        .        C        G        .        PASS     AC=1;AN=2;
   DP=14547;AF=0.65;EAS_AF=0.49;EUR_AF=0.77;AFR_AF=0.65;AMR_AF=0.69;SAS_AF
   =0.66;VT=SNP;NS=2548    GT       1|0
25 14      16093569        .        C        A        .        PASS     AC=1;AN=2;
   DP=28688;AF=0.3;EAS_AF=0.17;EUR_AF=0.48;AFR_AF=0.18;AMR_AF=0.38;SAS_AF
   =0.36;VT=SNP;NS=2548    GT       1|0
```

En este ejemplo, cada línea de datos indica un SNP en el cromosoma 14: la columna `POS` muestra la posición genómica, `REF` y `ALT` los alelos de referencia y alternativo, y el campo `GT` (`1|0`) indica que el individuo NA18956 porta un alelo alternativo en el primer haplotipo y el de referencia en el segundo.

1.3. Formato HDF5

HDF5 (*Hierarchical Data Format version 5*) (The HDF Group, 2025) es un formato binario diseñado para almacenar grandes volúmenes de datos numéricos de forma eficiente. A diferencia de los formatos de texto como FASTA y VCF, HDF5 permite acceso aleatorio a subconjuntos de datos sin necesidad de cargar el archivo completo en memoria RAM, lo que resulta esencial para los volúmenes de datos genómicos manejados en esta tesis.

Los datos se organizan jerárquicamente en grupos (análogos a directorios) y datasets (análogos a archivos), permitiendo estructurar la información por

cromosoma, individuo y tipo de dato. En esta tesis, cada individuo se almacena en un archivo HDF5 separado de aproximadamente 12 MB, conteniendo los genotipos de SNPs organizados por cromosoma.

El siguiente ejemplo muestra la estructura interna de un archivo HDF5 para el individuo NA18550 (población asiática). El archivo contiene dos grupos principales:

```
positions/              (grupo)
  positions/1           Dataset (270502,) uint32
  positions/2           Dataset (281095,) uint32
  ...
  positions/22          Dataset (45581,)  uint32
snps/                   (grupo)
  snps/1/               (grupo)
    snps/1/hap0         Dataset (270502,) |S1
    snps/1/hap1         Dataset (270502,) |S1
  snps/2/               (grupo)
    snps/2/hap0         Dataset (281095,) |S1
    snps/2/hap1         Dataset (281095,) |S1
  ...
```

El grupo `positions` almacena las coordenadas genómicas de cada SNP por cromosoma, y el grupo `snps` contiene los alelos para cada haplotipo (`hap0` y `hap1`). La notación `uint32` indica enteros sin signo de 32 bits (utilizados para las posiciones genómicas), mientras que `|S1` corresponde a la notación de NumPy/h5py para cadenas de un único byte (un carácter ASCII), suficiente para representar cada alelo (A, C, G o T). Por ejemplo, los primeros 10 SNPs del cromosoma 1:

```
positions/1:  [55299, 55326, 66507, 69511, 77961,
               84002, 87190, 88169, 263722, 770988]

snps/1/hap0:  [C, T, A, G, G, A, G, C, G, G]
snps/1/hap1:  [T, C, T, G, A, G, A, T, C, G]
```

Cada posición en `positions/1` corresponde a la ubicación genómica absoluta del SNP en el cromosoma 1, y los valores en `hap0` y `hap1` representan

los alelos del individuo en cada haplotipo. Esta estructura permite acceder eficientemente a cualquier subconjunto de SNPs por cromosoma sin cargar el archivo completo.