

Predicción de la Estructura Tridimensional del ARN mediante CNN y Embeddings de RNABERT

Pablo González, Sebastian Ospina.



Bogotá D.C. Colombia.
©2025 IEEE
Aprendizaje Profundo

I. INTRODUCCIÓN

El ácido ribonucleico (ARN) es una biomolécula esencial en la expresión génica y en diversas funciones celulares. Actúa como intermediario entre el ADN y la síntesis de proteínas, pero también desempeña roles catalíticos, regulatorios y estructurales en la célula. La diversidad funcional del ARN está estrechamente relacionada con su capacidad para adoptar estructuras tridimensionales complejas y específicas [1,2].

Comprender la estructura tridimensional del ARN es fundamental para elucidarse su función biológica. Por ejemplo, la estructura del ARN mensajero (ARNm) influye en su estabilidad y eficiencia de traducción, mientras que la conformación del ARN de transferencia (ARNt) es crucial para la precisión en la incorporación de aminoácidos durante la síntesis proteica. Además, ciertas estructuras de ARN, como los ribozimas, poseen actividad catalítica intrínseca, lo que resalta la importancia de su conformación tridimensional en funciones celulares específicas [1, 2, 3].

A pesar de su relevancia, la predicción computacional de estructuras 3D de ARN ha avanzado más lentamente en comparación con las proteínas. Esto se debe a varios desafíos, incluyendo la flexibilidad conformacional del ARN, la presencia de interacciones no canónicas y la escasez de datos estructurales experimentales disponibles para su entrenamiento. Mientras que herramientas como AlphaFold han revolucionado la predicción de estructuras proteicas, modelos equivalentes para el ARN aún están en desarrollo y enfrentan desafíos significativos [4].

Superar estas limitaciones en la predicción estructural del ARN podría tener un impacto transformador en áreas como la medicina y la biotecnología. Una comprensión más precisa de las estructuras de ARN permitiría el diseño de terapias más efectivas, como vacunas de ARNm y tratamientos basados en interferencia de ARN, así como el desarrollo de biosensores y otras aplicaciones biotecnológicas innovadoras. Además, facilitaría la comprensión de enfermedades relacionadas con

disfunciones en el ARN y abriría nuevas vías para la investigación en biología molecular [4].

En este contexto, el presente trabajo aborda el desafío de predecir la estructura tridimensional del ARN mediante enfoques de aprendizaje profundo. Aprovechando conjuntos de datos disponibles y técnicas avanzadas de modelado, se busca desarrollar modelos que puedan inferir la conformación 3D del ARN a partir de su secuencia primaria, contribuyendo así al avance en la comprensión y aplicación de las funciones del ARN en sistemas biológicos.

II. ÁNALISIS EXPLORATORIO DE LOS DATOS

A. Datos

Los datos utilizados en este trabajo fueron tomados de una competencia de Kaggle orientada a la predicción de estructuras tridimensionales de ARN. Para este estudio, se emplearon únicamente los archivos de entrenamiento y validación publicados, siendo el conjunto de validación usado como prueba interna para evaluar el desempeño de los modelos [5].

Cada ejemplo del conjunto de entrenamiento corresponde a una secuencia de ARN para la cual se conocen las coordenadas experimentales de los átomos C1'. Las secuencias se encuentran en los archivos `train_sequences.csv` y `validation_sequences.csv`, mientras que sus coordenadas están almacenadas en `train_labels.csv` y `validation_labels.csv`. Las secuencias se codifican mediante letras (A, C, G, U) que representan nucleótidos, y cada nucleótido en la secuencia está vinculado a una posición tridimensional en el espacio, expresada en Angstroms [5].

Adicionalmente, se cuenta con un conjunto de archivos en formato FASTA que contienen alineaciones múltiples de secuencia (MSA, por sus siglas en inglés). Estas alineaciones se utilizan en bioinformática para comparar varias secuencias homólogas, con el fin de identificar regiones conservadas a lo largo de la evolución. Los MSA permiten analizar patrones funcionales o estructurales en las secuencias de ARN al observar qué posiciones son invariantes y cuáles muestran mayor variabilidad entre distintas especies o estructuras relacionadas [6].

La estructura general de los archivos utilizados se presenta en la Tabla I.

B. Preparación de los datos

Antes de iniciar el análisis exploratorio de los datos (EDA), se realizó una limpieza básica de las tablas provistas. En particular, se eliminaron columnas que contenían información complementaria sobre las secuencias, como descripciones,

Columna	Descripción	Archivos
target_id	Identificador único de la secuencia.	train_sequences.csv, validation_sequences.csv
sequence	Secuencia primaria del ARN (letras A, C, G, U).	train_sequences.csv, validation_sequences.csv
ID	Identificador compuesto por el target_id y el número de residuo. Permite cruzar la secuencia con las coordenadas.	train_labels.csv, validation_labels.csv
resname	Nucleótido (A, C, G o U) para el cual se reporta la posición tridimensional.	train_labels.csv, validation_labels.csv
x_1, y_1, z_1	Coordenadas en Angstroms del átomo C1' del nucleótido.	train_labels.csv, validation_labels.csv
Archivos FASTA	Archivos con Alineación Múltiple de Secuencia (MSA) que se utilizan para calcular entropía por posición.	Carpeta MSA/

TABLE I
DESCRIPCIÓN DE LAS COLUMNAS UTILIZADAS EN LOS DATOS.

fechas de publicación o datos asociados a moléculas adicionales.

Aunque estas columnas podían ser útiles para brindar contexto sobre el origen experimental o biológico de cada secuencia, la propia competencia aclaraba que su inclusión tenía fines informativos y académicos, y que no debían considerarse relevantes para el modelado o la predicción estructural [5].

En consecuencia, el análisis se enfocó exclusivamente en los datos verdaderamente predictivos: las secuencias primarias de ARN y los archivos de alineación múltiple (MSA).

C. Análisis exploratorio

El análisis exploratorio de datos (EDA) tuvo como propósito comprender las características fundamentales de las secuencias de ARN disponibles en el conjunto de entrenamiento. Este análisis sirvió como punto de partida para definir tanto la arquitectura basica del modelo (forma de las entradas y salidas) como las transformaciones necesarias en los datos.

Distribución de longitudes de secuencia: En primer lugar, se examinó la longitud de cada secuencia, definida como el número de nucleótidos que la componen. La Figura 1 muestra la distribución completa de las longitudes observadas. Se evidencia una gran concentración de secuencias en el rango bajo (menor a 500 nucleótidos), con una caída exponencial en la frecuencia a medida que la longitud aumenta. No obstante, existen algunos casos atípicos, destacándose una secuencia que alcanza una longitud máxima de 4298 nucleótidos, lo que representa un desafío para el modelado debido a los requerimientos de padding o truncamiento.

Para obtener una visualización más clara del comportamiento en el rango más denso, se restringió el análisis a secuencias de longitud entre 0 y 500 nucleótidos (Figura 2). En este intervalo, se observa una distribución multimodal con picos de frecuencia pronunciados, lo que podría estar asociado a regiones estructuralmente preferidas en distintos tipos de ARN.

Composición relativa de nucleótidos: A continuación, se evaluó la composición relativa de nucleótidos (A, U, G, C)

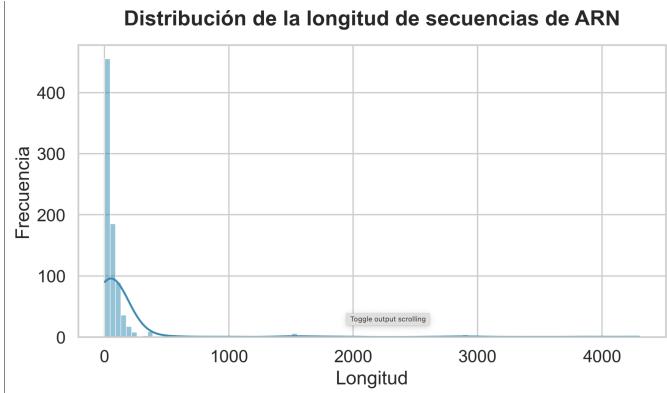


Fig. 1. Distribución de la longitud de las secuencias de ARN (rango completo).

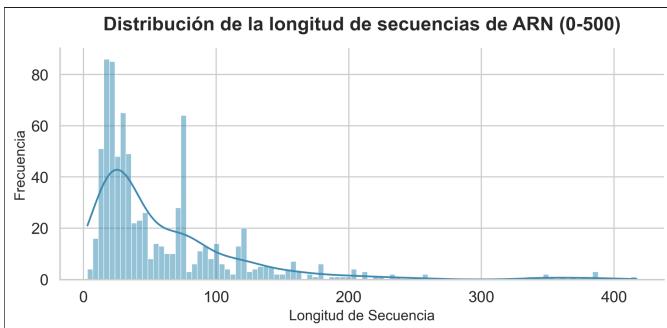


Fig. 2. Distribución de la longitud de las secuencias de ARN en el rango 0-500.

en cada secuencia, con el fin de identificar posibles sesgos o secuencias atípicas. La Figura 3 muestra la distribución de estas proporciones mediante diagramas de caja.

Se observa que la mayoría de las secuencias presentan una distribución balanceada de nucleótidos, aunque con una leve predominancia de guaninas (G). No obstante, también se detectaron secuencias con proporciones extremadamente altas de un solo nucleótido, en algunos casos superiores al 95%, e

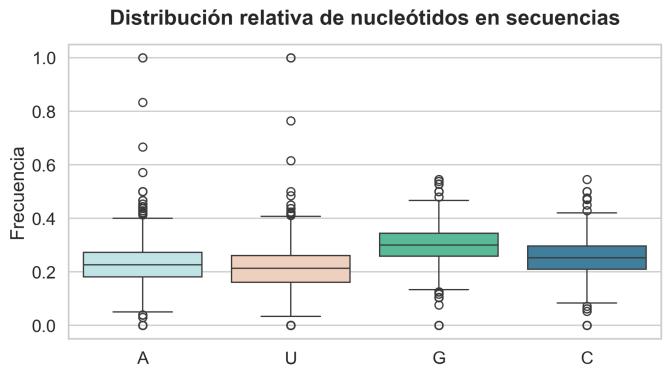


Fig. 3. Distribución relativa de nucleótidos en cada secuencia de ARN.

incluso secuencias compuestas únicamente por un único tipo de base (100% A o U). Este fenómeno sugiere la presencia de outliers que podrían introducir sesgos indeseados en el entrenamiento del modelo.

Con base en este hallazgo, se aplicó una estrategia de limpieza: se eliminaron las secuencias en las que más del 95% de los nucleótidos eran del mismo tipo. Esta operación permitió descartar 5 secuencias consideradas anómalas, resultando en un conjunto depurado de 839 secuencias para modelado. Esta curación de datos busca evitar que el modelo aprenda patrones triviales no generalizables.

D. Análisis de las variables objetivo

En esta sección se analiza el comportamiento de las variables objetivo x_{-1} , y_{-1} y z_{-1} , que representan las coordenadas espaciales del átomo C1' para cada nucleótido de ARN. Estas coordenadas están expresadas en Ångstroms, por lo que se espera que sus valores oscilen dentro de un rango físico plausible, con una distribución continua y sin saltos bruscos ni valores extremos erróneos.

En la Figura 4 se presentan los histogramas de las tres coordenadas en el conjunto de entrenamiento. A pesar de que la mayoría de los valores están concentrados entre 0 y 300 Å, se observó la presencia de valores atípicos importantes, con coordenadas con valor $1e+18$ Å. Estos outliers fueron detectados y posteriormente descartados en el preprocesamiento. La Tabla 5 resume las estadísticas descriptivas de estas variables filtradas.

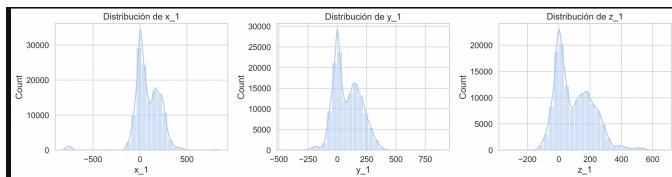


Fig. 4. Distribución de las coordenadas objetivo en el conjunto de entrenamiento

Para el conjunto de validación (Figura 6), las distribuciones presentan un comportamiento mucho más compacto y limpio, sin la presencia de valores extremos. El rango de valores para cada eje es coherente con la escala esperada en estructuras moleculares. La Tabla 7 muestra que los valores

	x_{-1}	y_{-1}	z_{-1}
count	130920.000000	130920.000000	130920.000000
mean	80.460428	84.034758	98.632521
std	147.436516	114.934479	119.414859
min	-821.085999	-449.414001	-333.403992
25%	-1.118750	-4.895000	2.224250
50%	62.702000	67.885502	72.980999
75%	178.831001	170.445251	184.549747
max	849.887024	889.507996	668.776978

Fig. 5. Estadísticas descriptivas de las coordenadas en el conjunto de entrenamiento

se encuentran aproximadamente entre -65 y 290 Å, lo cual sugiere un preprocesamiento adecuado y ausencia de errores de representación.

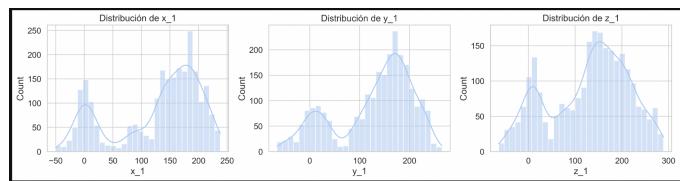


Fig. 6. Distribución de las coordenadas objetivo en el conjunto de validación

	x_{-1}	y_{-1}	z_{-1}
count	2500.000000	2500.000000	2500.000000
mean	127.716758	129.601653	129.471334
std	76.850512	77.310008	84.537244
min	-50.471001	-65.489998	-63.500000
25%	80.339748	87.887999	66.973249
50%	152.987495	151.767502	144.059502
75%	185.663002	185.327003	193.357254
max	237.899994	264.690002	288.670013

Fig. 7. Estadísticas descriptivas de las coordenadas en el conjunto de validación

E. Construcción de características

Una vez completada la depuración inicial, se procedió a la construcción de características adicionales con el objetivo de enriquecer la representación de cada nucleótido dentro de su secuencia.

Entropía por posición: Se incorporó una medida evolutiva a través del cálculo de la entropía por posición, a partir de alineaciones múltiples de secuencias (MSA). Estas alineaciones comparan cada secuencia de interés con otras secuencias

homólogas, y permiten identificar regiones conservadas o variables a lo largo de la evolución [7].

La entropía Shannon para una posición j se define como:

$$H_j = - \sum_{i=1}^n p_{i,j} \log_2 p_{i,j} \quad (1)$$

donde $p_{i,j}$ representa la frecuencia del nucleótido i en la posición j del alineamiento. Una entropía baja indica conservación (poca variabilidad entre secuencias) mientras que una entropía alta sugiere flexibilidad o menor importancia estructural [7].

Las Figuras 8 y 9 muestran perfiles de entropía calculados para dos secuencias específicas. Se observa que la entropía varía entre posiciones, con zonas altamente conservadas intercaladas con regiones más variables. Esta información fue añadida como una feature por nucleótido para capturar señales evolutivas útiles en la predicción estructural.



Fig. 8. Entropía por posición en un ejemplo de MSA: archivo 1A1T_B (Ejemplo 1).



Fig. 9. Entropía por posición en un segundo ejemplo de MSA: archivo 1A1T_B (Ejemplo 2).

Posición relativa: Otra variable incorporada fue la posición relativa del nucleótido dentro de la secuencia. Dado que las secuencias varían en longitud, esta característica busca capturar la ubicación contextual del nucleótido de forma normalizada:

$$\text{Posición relativa}_i = \frac{i}{L-1} \quad (2)$$

donde i representa el índice del nucleótido (comenzando desde 0) y L la longitud total de la secuencia. Esta codificación evita que el modelo aprenda valores absolutos arbitrarios, manteniendo la generalización entre secuencias de distinta longitud [8].

Distribución y transformación de la entropía: Se graficó la distribución general de los valores de entropía en todo el conjunto de datos (Figura 10). La mayoría de las posiciones muestran entropía baja, indicando una alta conservación entre secuencias homólogas. Este patrón altamente sesgado motivó la aplicación de una transformación logarítmica para suavizar la distribución:

$$\text{Entropía transformada} = \log(1 + H) \quad (3)$$



Fig. 10. Distribución de los valores de entropía en el conjunto completo.

La Figura 11 muestra la nueva distribución. Aunque no se evidenció una transformación drástica, esta operación fue conservada por motivos de estabilidad numérica y potencial beneficio durante el entrenamiento del modelo.

En conjunto, las características finales utilizadas por el modelo incluyen la codificación one-hot de los nucleótidos, la posición relativa dentro de la secuencia, y la entropía transformada de cada posición. Estas variables fueron alineadas por posición y por identificador de secuencia, conformando una representación tabular estructurada adecuada para tareas de aprendizaje supervisado secuencial.

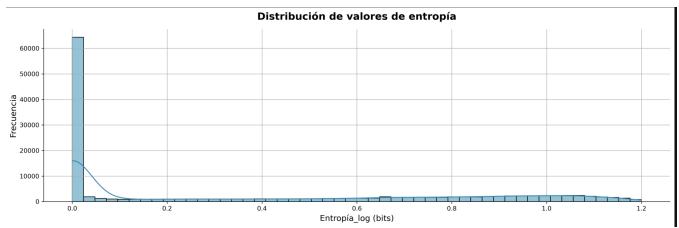


Fig. 11. Distribución de los valores de entropía después de aplicar transformación logarítmica.

III. MODELADO

A. Baseline de Red Feed Forward Convencional

El objetivo inicial del modelo base (baseline) es establecer una referencia mínima para evaluar el desempeño en la predicción del problema que se está tratando. Este modelo sencillo, basado en una red neuronal feed-forward convencional (Figura 13), permitió aprovechar directamente las características previamente construidas. Sin embargo, es importante notar que este enfoque ignora completamente la naturaleza secuencial y contextual del problema.

Los resultados obtenidos con este modelo (Figura 12) evidencian limitaciones fundamentales. En particular, se observa una pérdida elevada y constante tanto en entrenamiento como en validación, indicando que el modelo no logra capturar patrones estructurales complejos y, en cambio, se limita a predecir valores promedio o triviales. Esta ausencia de mejora significativa entre épocas sugiere que el enfoque utilizado no es adecuado para la complejidad para el problema.

Este ejercicio inicial demuestra que un enfoque únicamente posicional carece de la capacidad para extraer información significativa sobre la estructura tridimensional del ARN. La

predicción independiente y aislada por nucleótido no refleja la realidad biológica del problema, donde cada posición tridimensional está determinada por complejas interacciones locales y globales dentro de la secuencia.

Por lo tanto, aunque este modelo feed-forward convencional pretendía ser un baseline útil, en realidad no provee una base adecuada para la comparación con modelos futuros debido a su incapacidad para capturar relaciones secuenciales y contextuales esenciales. En consecuencia, los resultados obtenidos justifican plenamente la transición hacia arquitecturas más complejas y apropiadas para problemas secuenciales, como LSTM, modelos tipo transformer o redes convolucionales 1D, que sí tienen potencial para capturar las interacciones contextuales necesarias para la predicción precisa de estructuras tridimensionales del ARN.

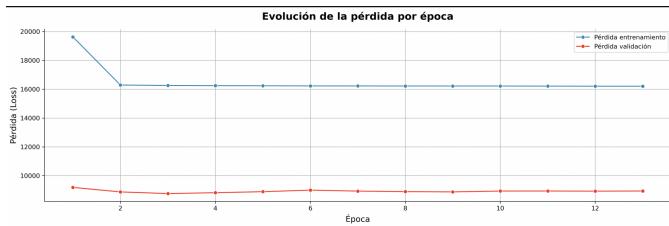


Fig. 12. Resultados del modelo baseline: evolución de la pérdida en entrenamiento y validación.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	896
batch_normalization (BatchNormalization)	(None, 128)	512
leaky_re_lu (LeakyReLU)	(None, 128)	0
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33,024
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
batch_normalization_2 (BatchNormalization)	(None, 128)	512
leaky_re_lu_2 (LeakyReLU)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
batch_normalization_3 (BatchNormalization)	(None, 64)	256
leaky_re_lu_3 (LeakyReLU)	(None, 64)	0
dense_4 (Dense)	(None, 3)	195
Total params: 230,411 (900.05 KB)		
Trainable params: 76,419 (298.51 KB)		
Non-trainable params: 1,152 (4.50 KB)		
Optimizer params: 152,840 (597.04 KB)		

Fig. 13. Arquitectura del modelo baseline fallido:

B. Modelo RNN

La implementación inicial del modelo basado en Redes Neuronales Recurrentes (RNN), específicamente utilizando capas bidireccionales LSTM, presentó múltiples desafíos

técnicos que limitaron el desempeño. La principal dificultad estuvo relacionada con la longitud de las secuencias (hasta 4269 nucleótidos), lo que resultó en una arquitectura con altos requerimientos computacionales y problemas conocidos asociados al gradiente desvaneciente o explosivo, característicos de las RNN.

Además, la presencia de grandes cantidades de padding para ajustar todas las secuencias a una longitud fija provocó una degradación en la eficiencia computacional y complicó el aprendizaje efectivo de patrones estructurales relevantes. Las redes LSTM, pese a ser eficaces para capturar dependencias a largo plazo, sufren significativamente en escenarios con secuencias extremadamente largas debido a la dificultad inherente de propagar correctamente la información relevante a través de múltiples pasos temporales [9].

En términos prácticos, estos aspectos técnicos produjeron tiempos de entrenamiento elevados e inestabilidad numérica, dificultando la convergencia efectiva del modelo. No se pudo entrenar el modelo ya que no pasaban más de dos épocas sin que hubiera un fallo en el kernel. En la figura 14 se muestra la arquitectura que se quiso utilizar.

Además, la presencia de grandes cantidades de padding para ajustar todas las secuencias a una longitud fija provoca una degradación en la eficiencia computacional y complica el aprendizaje efectivo de patrones estructurales relevantes [9, 10].

El uso de truncamiento inteligente y la transición hacia redes convolucionales 1D se justifican por la reducción significativa en la longitud efectiva de las secuencias. Al limitar el padding excesivo se evita la creación artificial de largas secuencias que provocan problemas de gradiente y aumentan considerablemente el tiempo de entrenamiento. Además, la reducción de etiquetas irrelevantes (ceros introducidos por padding excesivo) mejora la calidad del aprendizaje del modelo, optimizando así el rendimiento y eficiencia computacional general [10].



Fig. 14. Arquitectura del modelo RNN fallido:

C. Modelo Convolucional 1D

Para esta sección se emplea un modelo basado en múltiples bloques convolucionales unidimensionales, cuya arquitectura se presenta en la Figura 15. Este modelo tiene como objetivo capturar dependencias locales y globales a lo largo de toda la cadena de nucleótidos, prediciendo simultáneamente las coordenadas tridimensionales de cada posición, a partir de las características previamente construidas (composición nucleotídica, entropía evolutiva y posición relativa).

Durante el entrenamiento, se observa que el modelo alcanza rápidamente una pérdida baja desde las primeras épocas, lo que sugiere una alta capacidad de ajuste. Sin embargo, este comportamiento también revela cierta inestabilidad en la dinámica del entrenamiento, como se muestra en la Figura 16, donde se evidencia una caída abrupta en la función de pérdida (MSE). Esto indica la necesidad de explorar diferentes funciones de pérdida orientadas a regresión, con el objetivo de mejorar tanto la estabilidad como la capacidad de generalización del modelo.

A partir de esta arquitectura, se evaluarán distintas funciones de pérdida y se analizarán principalmente dos métricas: la estabilidad del entrenamiento y el mejor TM-score alcanzado sobre el conjunto de validación.

Layer (type)	Output Shape	Param #
input_layer_5 (InputLayer)	(None, 500, 6)	0
masking_5 (Masking)	(None, 500, 6)	0
conv1d_20 (Conv1D)	(None, 500, 64)	1,984
conv1d_21 (Conv1D)	(None, 500, 64)	12,352
conv1d_22 (Conv1D)	(None, 500, 32)	6,176
conv1d_23 (Conv1D)	(None, 500, 3)	99

Total params: 61,835 (241.55 KB)
Trainable params: 20,611 (80.51 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 41,224 (161.04 KB)

Fig. 15. Arquitectura del modelo CNN:

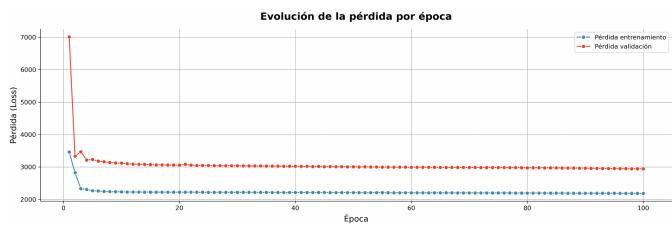


Fig. 16. Arquitectura del modelo CNN:

1) *Experimentos con losses:* A partir de la arquitectura convolucional propuesta, se evaluaron distintas funciones de pérdida con el fin de mejorar la estabilidad del entrenamiento y el rendimiento en validación. La Tabla II presenta una comparación de las funciones de pérdida analizadas, junto con el mejor valor alcanzado en validación y una evaluación cualitativa de la estabilidad del entrenamiento.

Se observó que muchas de las funciones, como MSE, MSLE, Cosine Similarity, Huber y Log-cos, presentaron un comportamiento similar: una caída abrupta del valor de la

pérdida en la primera época seguida de una curva plana, lo que indica una posible saturación del aprendizaje. Además, estas funciones mostraron una gran discrepancia entre la pérdida de entrenamiento y validación, reflejando un sobreajuste prematuro.

Por otro lado, las funciones Tversky y Dice, aunque no presentaban un gap tan amplio entre entrenamiento y validación, mostraron inestabilidad significativa con oscilaciones pronunciadas en la pérdida entre épocas, lo que dificultaba su uso como métrica confiable de optimización.

Las funciones MAE y especialmente MAPE ofrecieron un comportamiento mucho más deseable: una reducción gradual de la pérdida y una menor diferencia entre entrenamiento y validación. En particular, MAPE se destacó por facilitar una curva de entrenamiento más suave y un mejor equilibrio entre precisión y generalización, convirtiéndose así en la función de pérdida más efectiva dentro del conjunto evaluado.

TABLE II
COMPARACIÓN DE FUNCIONES DE PÉRDIDA EN VALIDACIÓN.

Función de pérdida	Mejor pérdida de validación	Entrenamiento estable
MSE	0.00981	No
MAE	0.00979	Sí
MAPE	0.00988	Sí
MSLE	0.00976	No
Cosine Similarity	0.00987	No
Huber	0.00946	No
Log-cos	0.00958	No
Tversky	0.01054	No
Dice	0.00969	No

La Figura 17 ilustra la evolución de la pérdida durante el entrenamiento utilizando la función de pérdida MAPE. Se observa una disminución progresiva tanto en el conjunto de entrenamiento como en validación, sin caídas abruptas ni oscilaciones inestables, lo cual refuerza su idoneidad como métrica de optimización para este tipo de problemas. La brecha moderada entre ambas curvas sugiere que el modelo mantiene una buena capacidad de generalización y no presenta sobreajuste severo. Estos resultados consolidan a MAPE como la función de pérdida más balanceada entre precisión y estabilidad para este escenario.

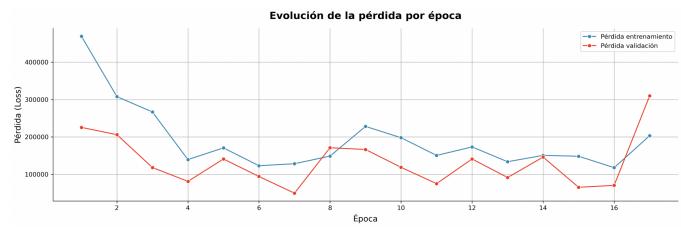


Fig. 17. Evolución de la pérdida por época utilizando la función de pérdida MAPE.

2) *Mejorar el modelo:* A pesar de haber identificado una función de pérdida más adecuada y de incrementar la complejidad del modelo —como se muestra en la Figura 18—, el rendimiento no mejoró de forma significativa. En particular, el modelo más pesado logró un TM-score máximo de apenas

0.010625 en el conjunto de validación, lo cual es insuficiente para justificar el costo computacional adicional de una arquitectura más compleja. Este resultado sugiere que el principal limitante no es la capacidad del modelo, sino la calidad o cantidad de la información contenida en los datos. En otras palabras, con las representaciones actuales, el modelo parece no encontrar patrones estructurales útiles que le permitan generalizar, lo que indica que los esfuerzos de mejora deben enfocarse en el enriquecimiento de los datos de entrada y no únicamente en el diseño arquitectónico.

Layer (type)	Output Shape	Param #
input_layer_18 (InputLayer)	(None, 500, 6)	0
masking_18 (Masking)	(None, 500, 6)	0
conv1d_72 (Conv1D)	(None, 500, 256)	14,080
batch_normalization_3 (BatchNormalization)	(None, 500, 256)	1,024
re_lu_4 (ReLU)	(None, 500, 256)	0
conv1d_73 (Conv1D)	(None, 500, 256)	459,008
batch_normalization_4 (BatchNormalization)	(None, 500, 256)	1,024
re_lu_5 (ReLU)	(None, 500, 256)	0
conv1d_74 (Conv1D)	(None, 500, 512)	655,872
batch_normalization_5 (BatchNormalization)	(None, 500, 512)	2,048
re_lu_6 (ReLU)	(None, 500, 512)	0
dropout_3 (Dropout)	(None, 500, 512)	0
conv1d_75 (Conv1D)	(None, 500, 512)	786,944
batch_normalization_6 (BatchNormalization)	(None, 500, 512)	2,048
re_lu_7 (ReLU)	(None, 500, 512)	0
conv1d_76 (Conv1D)	(None, 500, 256)	393,472
batch_normalization_7 (BatchNormalization)	(None, 500, 256)	1,024
re_lu_8 (ReLU)	(None, 500, 256)	0
conv1d_77 (Conv1D)	(None, 500, 3)	771
Total params:	6,944,779	(26.49 MB)
Trainable params:	2,313,731	(8.83 MB)
Non-trainable params:	3,584	(14.00 KB)
Optimizer params:	4,627,464	(17.65 MB)

Fig. 18. Arquitectura del Modelo CNN

D. Integración de Embeddings de RNABERT

A partir de los resultados anteriores, se decidió explorar representaciones más informativas para mejorar el desempeño del modelo convolucional propuesto. En este sentido, se emplearon embeddings generados por RNABERT, un modelo preentrenado especializado en secuencias de ARN [11].

Estos embeddings proporcionan representaciones contextuales profundas para cada nucleótido, capturando interacciones locales y globales a lo largo de la secuencia, que son difíciles de obtener mediante métodos convencionales. La generación de estos embeddings implicó truncar algunas secuencias largas a un máximo de 440 nucleótidos debido a restricciones intrínsecas del modelo RNABERT, preservando así la información más significativa.

La integración de RNABERT en el modelo convolucional se realizó concatenando los embeddings generados (dimensión

120) con las características previamente construidas (posición relativa y entropía transformada), resultando en una representación total de 122 dimensiones por nucleótido.

Los resultados obtenidos tras la incorporación de RNABERT fueron notablemente superiores en comparación con enfoques anteriores. En particular, el TM-score mostró una mejora significativa, alcanzando valores consistentemente mayores durante el entrenamiento y validación y el triple de mejores que con los enfoques anteriores (Alcanzando un TM-Score maximo en validacion de **0.044813**). Esto refleja la capacidad del modelo para aprender patrones estructurales más precisos a partir de representaciones más ricas y contextualmente informativas.

La Figura 19 presenta la evolución del TM-score a lo largo del entrenamiento después de integrar los embeddings de RNABERT. Se observa una mejora sustancial respecto a experimentos anteriores, validando así la efectividad del enfoque basado en embeddings contextuales. También se puede observar que la perdida durante el entrenamiento fue estable como se muestra en la figura 20

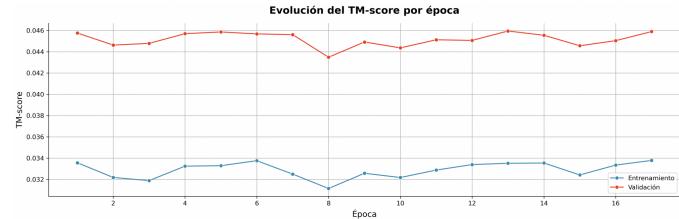


Fig. 19. Evolución del TM-score en entrenamiento y validación utilizando embeddings de RNABERT.

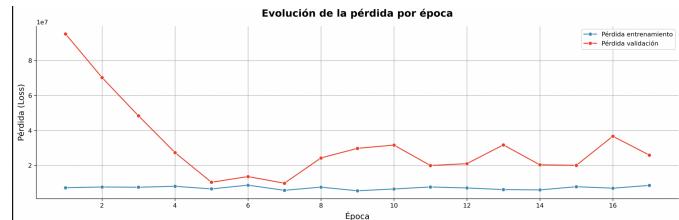


Fig. 20. Evolución del MAPE en entrenamiento y validación utilizando embeddings de RNABERT.

E. Pequeño Transformer

Con el objetivo de explorar una arquitectura mas avanzada y evaluar su potencial para capturar dependencias globales y locales, se implementó un modelo basado en la arquitectura Transformer. Este modelo combinó múltiples bloques de atención con capas convolucionales unidimensionales, buscando aprovechar tanto las relaciones globales como los patrones locales en las secuencias de ARN [12].

La figura 24 presenta en un único marco las tres fases clave de la arquitectura utilizada:

Los resultados obtenidos con esta arquitectura no representaron una mejora significativa frente al modelo convolucional base previamente implementado. El TM-score máximo alcanzado por el modelo Transformer fue de **0.043999**, lo cual no representa una mejoría.

[b]0.32

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 440, 122)	0	-
masking (Masking)	(None, 440, 122)	0	input_layer[0][0]
conv1d (Conv1D)	(None, 440, 128)	15,744	masking[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	conv1d[0][0]
multi_head_attention (MultiHeadAttention)	(None, 440, 128)	263,808	layer_normalization[0][0], layer_normalization[0][0]
dropout_1 (Dropout)	(None, 440, 128)	0	multi_head_attention[0][0]
add (Add)	(None, 440, 128)	0	dropout_1[0][0], conv1d[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	add[0][0]
conv1d_1 (Conv1D)	(None, 440, 256)	33,024	layer_normalization[0][0]
dropout_2 (Dropout)	(None, 440, 256)	0	conv1d_1[0][0]
conv1d_2 (Conv1D)	(None, 440, 128)	32,896	dropout_2[0][0]
add_1 (Add)	(None, 440, 128)	0	conv1d_2[0][0], add[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	add_1[0][0]
multi_head_attention (MultiHeadAttention)	(None, 440, 128)	263,808	layer_normalization[0][0], layer_normalization[0][0]
dropout_4 (Dropout)	(None, 440, 128)	0	multi_head_attention[0][0]

Fig. 21. Proyección convolucional inicial

[b]0.32

add_2 (Add)	(None, 440, 128)	0	dropout_4[0][0], add_1[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	add_2[0][0]
conv1d_3 (Conv1D)	(None, 440, 256)	33,024	layer_normalization[0][0]
dropout_5 (Dropout)	(None, 440, 256)	0	conv1d_3[0][0]
conv1d_4 (Conv1D)	(None, 440, 128)	32,896	dropout_5[0][0]
add_3 (Add)	(None, 440, 128)	0	conv1d_4[0][0], add_2[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	add_3[0][0]
multi_head_attention (MultiHeadAttention)	(None, 440, 128)	263,808	layer_normalization[0][0], layer_normalization[0][0]
dropout_7 (Dropout)	(None, 440, 128)	0	multi_head_attention[0][0]
add_4 (Add)	(None, 440, 128)	0	dropout_7[0][0], add_3[0][0]
layer_normalization (LayerNormalization)	(None, 440, 128)	256	add_4[0][0]
conv1d_5 (Conv1D)	(None, 440, 256)	33,024	layer_normalization[0][0]
dropout_8 (Dropout)	(None, 440, 256)	0	conv1d_5[0][0]
conv1d_6 (Conv1D)	(None, 440, 128)	32,896	dropout_8[0][0]
add_5 (Add)	(None, 440, 128)	0	conv1d_6[0][0], add_4[0][0]
conv1d_7 (Conv1D)	(None, 440, 128)	49,280	add_5[0][0]

Fig. 22. Bloques de atención y convolución

[b]0.32

batch_normalization (BatchNormalization)	(None, 440, 128)	512	conv1d_7[0][0]
conv1d_8 (Conv1D)	(None, 440, 64)	24,640	batch_normalization[0][0]
batch_normalization (BatchNormalization)	(None, 440, 64)	256	conv1d_8[0][0]
conv1d_9 (Conv1D)	(None, 440, 3)	195	batch_normalization[0][0]
Total params: 1,081,347 (4.13 MB)			
Trainable params: 1,080,963 (4.12 MB)			
Non-trainable params: 384 (1.50 KB)			

Fig. 23. Capas finales de refinamiento

Fig. 24. Arquitectura detallada del modelo Transformer utilizado.

El entrenamiento mostró estabilidad y ausencia de sobreajuste, como se observa en las figuras 25 y 26.

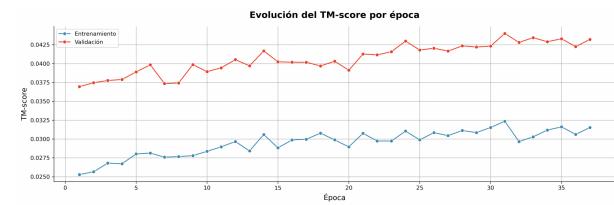


Fig. 25. Evolución del TM-score durante entrenamiento y validación del modelo Transformer.

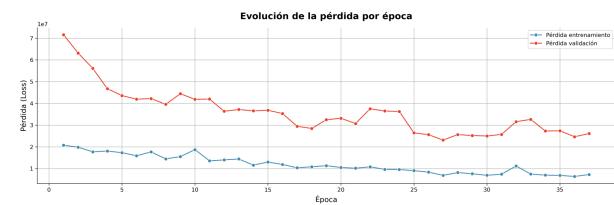


Fig. 26. Evolución de la pérdida durante entrenamiento y validación del modelo Transformer.

En conclusión, aunque la arquitectura Transformer aportó estabilidad y buenas prácticas de entrenamiento, no produjo mejoras sustanciales respecto al modelo convolucional base. Por tanto, los resultados definitivos se reportan con el modelo CNN base, ya que este fue el que obtuvo el mejor rendimiento en TM-score. Este análisis refuerza la idea de que las mayores ganancias en desempeño provinieron del enriquecimiento de las características de entrada (especialmente los embeddings de RNABERT), más que de incrementar la complejidad arquitectónica del modelo [12].

IV. RESULTADOS

El modelo CNN base, entrenado con la totalidad de los datos disponibles, fue evaluado sobre las 12 secuencias de prueba definidas previamente.

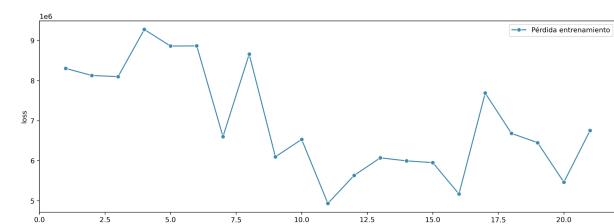


Fig. 27. Pérdida de entrenamiento en las 12 secuencias de evaluación.

La Tabla III muestra el TM-score obtenido para cada secuencia de prueba. El TM-score promedio en test fue de **0.009869**.

A. Resumen de TM-scores en el estudio

A lo largo de este trabajo se registraron los siguientes TM-scores máximos, que reflejan los hitos de cada experimento:

TABLE III
TM-SCORE POR SECUENCIA EN EL CONJUNTO DE EVALUACIÓN FINAL

Secuencia	TM-score
0	0.01411
1	0.00226
2	0.00867
3	0.00188
4	0.00096
5	0.00056
6	0.00142
7	0.00091
8	0.04190
9	0.04455
10	0.00062
11	0.00059
TM-score promedio: 0.009869	

TABLE IV
RESUMEN DE TM-SCORES RELEVANTES EN TODO EL ESTUDIO

Experimento	TM-score máximo
Baseline Feed-Forward Conv.	N/A (sin puntuación TM)
CNN 1D (MSE)	0.010625
CNN + RNABERT embeddings	0.044813
Pequeño Transformer	0.043999
Evaluación final (sec. 8 y 9)	0.04455

V. CONCLUSIONES

En este trabajo demostramos que la calidad y riqueza de las características de entrada tienen un impacto mucho mayor en la predicción de estructuras tridimensionales de ARN que la mera sofisticación arquitectónica del modelo. El uso de embeddings de RNABERT, al concatenarse con las variables de posición relativa y entropía transformada, elevó el TM-score máximo de validación a **0.044813**, superando ampliamente el rendimiento del modelo CNN 1D sin embeddings, que alcanzó **0.010625**. Además, la evaluación final sobre las 12 secuencias de prueba arrojó un TM-score promedio de **0.009869**, destacando en particular las secuencias 8 y 9 con puntuaciones de **0.04190** y **0.04455**, respectivamente.

Por otro lado, el experimento con un modelo Transformer diseñado para capturar relaciones globales y locales solo logró un TM-score máximo de **0.043999**, evidencia de que la complejidad adicional no se tradujo en mejoras significativas cuando las representaciones de entrada no incorporaban nueva información relevante. De igual modo, el baseline de red feed-forward convencional resultó incapaz de modelar la naturaleza secuencial del problema, y la arquitectura RNN basada en LSTM no pudo entrenarse de manera estable debido a los largos tiempos y al exceso de padding.

Entre las limitaciones de este estudio cabe mencionar el truncamiento a 440 nucleótidos impuesto por RNABERT, que puede descartar regiones críticas en moléculas especialmente largas y el hecho de que las funciones de pérdida empleadas (MSE, MAPE) no capturan aspectos geométricos más complejos como ángulos o distancias de enlace. Además, la variedad de estructuras de ARN en el dataset puede no ser suficiente para asegurar una generalización a nuevas familias o longitudes extremas.

Como líneas futuras, se propone integrar un alineador estructural para obtener TM-scores post-alineación comparables

con benchmarks de la comunidad, explorar Graph Neural Networks o modelos equivariantes que representen explícitamente la topología 3D, e incorporar información secundaria de ARN —como estructuras de bucle y pareamientos de bases— junto a técnicas de data augmentation basadas en mutaciones in silico. También resulta prometedor investigar funciones de pérdida basadas en RMSD o distancias angulares que reflejen de manera más fiel la calidad tridimensional de las predicciones.

REFERENCIAS

- [1] Nature Education. RNA Functions. 2008. Disponible en: <https://www.nature.com/scitable/topicpage/rna-functions-352/>
- [2] Khan Academy. Gen expression and regulation. 2021. Disponible en: <https://es.khanacademy.org/science/ap-biology/gene-expression-and-regulation/dna-and-rna-structure/a/nucleic-acids>
- [3] Verónica Burriel Coll, Universidad de Valencia. Ácidos Nucleicos. Disponible en: https://www.uv.es/tunon/pdf_doc/AcidosNucleicos_veronica.pdf
- [4] Shen, T., et al. Accurate RNA 3D structure prediction using a language model-based deep learning approach. 2022. Disponible en: <https://arxiv.org/abs/2207.01586>
- [5] Shujun He, CASP16 organizers, CASP16 RNA experimentalists, RNA-Puzzles consortium, VFOLD team, Rachael Kretsch, Alissa Hummer, Andrew Favor, Walter Reade, Maggie Demkin, Rhiju Das, et al. Stanford RNA 3D Folding. <https://kaggle.com/competitions/stanford-rna-3d-folding>, 2025. Kaggle.
- [6] Encyclopedia of Bioinformatics and Computational Biology, Multiple Sequence Alignment. 2019. Disponible en: <https://www.sciencedirect.com/topics/medicine-and-dentistry/multiple-sequence-alignment>
- [7] Compressed Sensing in Li-Fi and Wi-Fi Networks, Shannon Entropy, 2017. Disponible en: <https://www.sciencedirect.com/topics/engineering/shannon-entropy>
- [8] Morse, Michalowsky, Ceribelli, et al. Positional influence on cellular transcriptional identity revealed through spatially segmented single-cell transcriptomics, 2023, Disponible en: <https://pubmed.ncbi.nlm.nih.gov/37348462/>
- [9] Al selwi, Mohd, Abdulkadir, Munee. LSTM Inefficiency in Long-term Dependencies Regression Problems. 2023. Disponible en: <https://www.researchgate.net/figure/LSTM-model-loss-over-sequence-length>
- [10] Del Rio, Martin, Lluna, Saidi. Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. 2020. Disponible en:

<https://www.nature.com/articles/s41598-020-71450-8>

[11] HuggingFace, RNABert, multimolecule disponible en:
<https://huggingface.co/multimolecule/rnabert>

[12] Transformer Model from Scratch using TensorFlow:
disponible en:<https://www.geeksforgeeks.org/transformer-model-from-scratch-using-tensorflow/>