Optimización de Redes Neuronales mediante Computación ADN

Michael Steven Jiménez Contreras[1] & Oscar David Rubiano Diaz[1]

***Abstract*— Se propone un algoritmo basado en lógica de computación ADN para optimizar automáticamente las topologías de redes neuronales. Cada cadena de ADN representa una configuración de capas ocultas, preservando las de entrada, salida e hiperparámetros clave. Implementado con TensorFlow y Keras, el método busca minimizar el error mediante exploración automática. Las pruebas realizadas con el dataset Iris, validan su eficacia para identificar arquitecturas óptimas.**

I. INTRODUCCIÓN

En este trabajo, se presenta un algoritmo que utiliza lógica de computación ADN para buscar topologías óptimas a problemas de redes neuronales.

Las redes neuronales son herramientas que han evolucionado significativamente en los últimos años, consolidándose como herramientas poderosas para la toma de decisiones. Estas permiten analizar el impacto de diversas variables sobre un fenómeno observado, lo que las convierte en instrumentos valiosos para determinar políticas y acciones adecuadas frente a un suceso específico. En este contexto, la capacidad de probar múltiples topologías de modelos de manera automática, representa una gran oportunidad para agilizar el desarrollo de modelos precisos.

La lógica de la computación basada en ADN resulta ser una estrategia versátil y aplicable en múltiples contextos. Al tratarse de un método de búsqueda, permite identificar posibles soluciones válidas para un problema dado. Al combinarlas con las capacidades de clasificación de las redes neuronales abre nuevas posibilidades para la optimización de topologías que minimicen la función de pérdida.

II. ESTADO DEL ARTE

II-A. Computación ADN

La computación de ADN es un campo emergente que emplea moléculas de ADN como sustrato para realizar procesos computacionales o como medio de almacenamiento de información. Esta técnica se basa en las propiedades únicas del ADN, como su capacidad para formar pares de bases de manera predecible y específica (adenina con timina, y citosina con guanina). Estas propiedades permiten la construcción de redes de reacción química a gran escala, donde el ADN actúa como un material programable para la computación y el almacenamiento de datos [1].

En la computación de ADN, la información se codifica en la secuencia de nucleótidos. Las operaciones computacionales se realizan mediante interacciones moleculares, como el desplazamiento de hebras de ADN (DSD, por sus siglas en inglés), en el que una hebra de ADN de entrada reemplaza a otra en un complejo de doble hebra, liberando una hebra de salida. Estas reacciones pueden controlarse con precisión ajustando la longitud y la secuencia de las zonas de inicio de las hebras. Además, se pueden utilizar nanostructuras de ADN autoensambladas para implementar tareas computacionales y algoritmos complejos [2].

Una de las aplicaciones más prometedoras de la computación de ADN es la construcción de redes neuronales moleculares. Estas redes pueden ser entrenadas con datos y luego convertidas en complejos de ADN que replican funciones específicas en el laboratorio. Por ejemplo, se han utilizado para analizar firmas de expresión genética y para el diagnóstico de enfermedades. En el ámbito médico, la computación de ADN ha demostrado ser particularmente útil para la detección de biomarcadores, incluso en concentraciones extremadamente bajas [3].

Además de la computación, el ADN se utiliza como medio de almacenamiento de datos. La información se puede codificar en la secuencia de nucleótidos o en la estructura del ADN. La lectura de datos almacenados en ADN se realiza mediante técnicas de secuenciación, que identifican la secuencia de bases nitrogenadas. Los datos codificados en la estructura del ADN se pueden leer usando técnicas como la microscopía de fuerza atómica o la microscopía electrónica.

El ADN ofrece ventajas significativas, como una densidad de almacenamiento extremadamente alta y la capacidad de conservar datos durante períodos prolongados [4]. Sin embargo, la lectura de datos almacenados en ADN sigue siendo un proceso costoso y lento. Para abordar estos desafíos, se están desarrollando métodos innovadores que incluyen la compartimentación del ADN en diferentes recipientes, el etiquetado de datos con códigos de barras o etiquetas de ADN, y la recuperación selectiva de datos mediante técnicas como el uso de imanes o la reacción en cadena de la polimerasa (PCR).

II-B. Redes neuronales feedforward

Las redes neuronales feedforward son arquitecturas totalmente conectadas donde el procesamiento es unidireccional hacia las capas posteriores [5].

Estas redes, al momento de ser entrenadas, utilizan algoritmos como descenso del gradiente, el cual permite reducir su función de error utilizando su derivada con respecto a los pesos; sin embargo, puede ocurrir que se estanquen en mínimos locales o en puntos de silla, en lugar del mínimo global, pues depende de la complejidad del problema [6].

Las redes neuronales feedforward son arquitecturas populares debido a su capacidad para generalizar bien, lo que significa que pueden producir resultados precisos con datos

Universidad Distrital Francisco José de Caldas

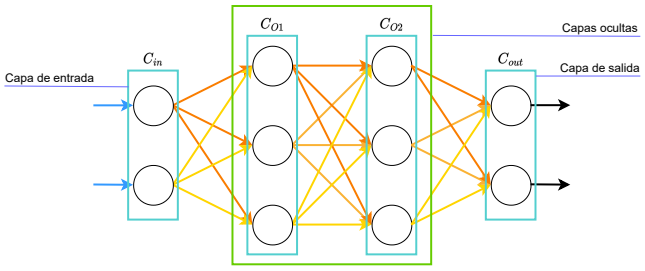

Fig. 1. Diagrama de una red neuronal, elaboración propia basada en [5].

que no se vieron durante el entrenamiento [7]. Estas redes tienen una estructura en capas donde la información fluye en una sola dirección, desde la capa de entrada hasta la de salida, pasando por una o más capas ocultas. En una red feedforward en capas, cualquier camino desde un nodo de entrada a un nodo de salida atraviesa el mismo número de arcos.

El entrenamiento de estas redes se realiza mediante algoritmos de optimización y aprendizaje que ajustan los pesos y sesgos para minimizar una función de error. Un algoritmo común es backpropagation, una variación de la búsqueda de gradiente que utiliza un criterio de optimalidad de mínimos cuadrados. El algoritmo calcula el gradiente del error con respecto a los pesos propagando el error hacia atrás a través de la red. Sin embargo, backpropagation puede tener problemas de escalado, especialmente en problemas complejos, pues puede quedarse atrapado en mínimos locales [7]. Esto se debe a que el espacio de soluciones puede tener muchos mínimos locales, y backpropagation puede no encontrar el mínimo global. Además, backpropagation requiere funciones de transferencia diferenciables, lo que limita su uso con algunos tipos de nodos o criterios de optimalidad.

Como alternativa a backpropagation, se pueden utilizar algoritmos genéticos. Estos algoritmos se inspiran en la evolución biológica y exploran el espacio de soluciones de manera inteligente para encontrar valores cercanos al óptimo global. Un algoritmo genético comienza con una población de cromosomas, cada uno representando un conjunto de pesos de red neuronal. Los cromosomas son evaluados usando una función que retorna una puntuación para cada uno [7]. Luego, los operadores genéticos, como la mutación y el cruce, son aplicados a los cromosomas padres para crear descendientes. Los hijos son evaluados e insertados en la población. Los algoritmos genéticos no tienen el mismo problema de escala que backpropagation y no se ven afectados por los mínimos locales. Además, pueden trabajar con funciones de transferencia discontinuas.

En cuanto a la capacidad de las redes feedforward, se ha demostrado que una red con una sola capa oculta puede aproximar cualquier función continua con una precisión arbitraria, siempre que se utilice un número suficiente de nodos en la capa oculta. Sin embargo, en la práctica, se utilizan más capas ocultas para resolver problemas de manera más eficiente. Si bien una red con una capa oculta es teóricamente suficiente, una red con dos capas ocultas y menos nodos puede resolver el mismo problema de forma más eficiente [8]. Sin embargo, la elección de la arquitectura de red apropiada, incluyendo el número de capas y nodos, sigue siendo en parte un arte.

Además de la arquitectura de red, existen técnicas para mejorar la capacidad de generalización de una red modificando tanto las conexiones como la arquitectura durante el entrenamiento. Estas técnicas se dividen en dos categorías: métodos de poda y métodos constructivos. Los métodos de poda comienzan con una red grande y eliminan gradualmente los nodos o conexiones innecesarias. En cambio, los métodos constructivos comienzan con una red pequeña y añaden gradualmente nodos o conexiones según sea necesario. Estos métodos buscan escapar de los mínimos locales. Además, se pueden combinar los métodos constructivos con los de poda para obtener mejores resultados.

II-C. Algoritmos genéticos con ADN

Los algoritmos genéticos son técnicas de resolución de problemas inspiradas en la biología que optimizan una función de aptitud, partiendo de un grupo inicial seleccionan unos cuantos para empezar a cruzar y mutar, luego los evalúan y determinan cuáles pueden pasar a la siguiente generación [9].

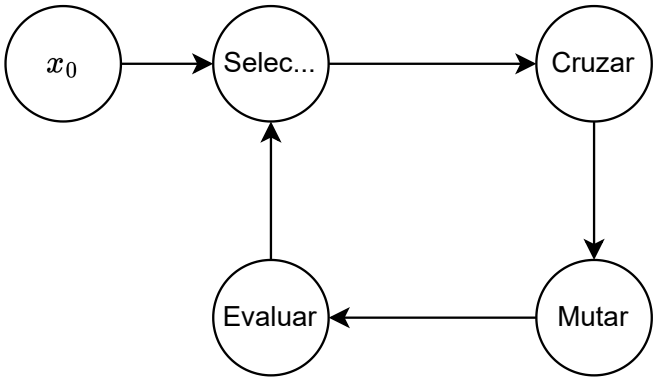

Fig. 2. Algoritmo genético utilizado, elaboración propia basada en [9]

La eficiencia de estos algoritmos depende de la cantidad de soluciones que se permiten por generación, en las primeras generaciones se espera un rendimiento bajo [9].

Se busca representar el problema en un formato que permita hacer las operaciones de forma sencilla, en este caso se utilizó ADN. El procedimiento se puede observar en Fig 3.

Todas las operaciones pueden variar ligeramente, en el caso de esta estrategia se plantea utilizando una selección elitista y cruce de un punto, lo cual significa que se conservan los mejores individuos y se divide cada cromosoma en dos partes y se combinan [10].

II-D. Redes neuronales basadas en ADN

Las redes neuronales artificiales (RNA) basadas en ADN son una forma de computación molecular que utiliza las propiedades del ADN para realizar cálculos y procesar información. Estas redes se inspiran en la estructura y el

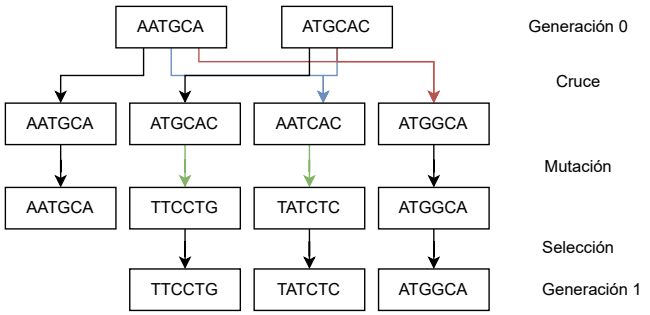

Fig. 3. Algoritmo genético con ADN. Elaboración propia.

funcionamiento del cerebro humano, pero en lugar de usar componentes electrónicos, emplean moléculas de ADN para llevar a cabo las operaciones [11]. El objetivo es crear sistemas que puedan procesar información, aprender y tomar decisiones [12].

El ADN se compone de subunidades llamadas nucleótidos, cada uno con una base nitrogenada: Adenina (A), Guanina (G), Timina (T) o Citosina (C). La secuencia de estas bases a lo largo de una cadena de ADN puede utilizarse para almacenar y codificar información. En las redes neuronales de ADN, la información se representa mediante la concentración de moléculas específicas de ADN, y las operaciones se llevan a cabo mediante reacciones químicas entre estas moléculas [13]. Una de las principales ventajas de la computación con ADN es su capacidad de realizar cálculos masivamente paralelos, ya que muchas moléculas de ADN pueden operar simultáneamente. Esto permite resolver problemas complejos en menos tiempo que con la computación electrónica tradicional.

La construcción de redes neuronales con ADN implica la creación de circuitos moleculares que imiten el funcionamiento de las neuronas artificiales. Esto incluye la implementación de operaciones como la multiplicación ponderada (MAC) y la suma de entradas, así como funciones de activación no lineales como la sigmoide, ReLU y softmax. La multiplicación ponderada se realiza mediante reacciones de desplazamiento de cadenas de ADN donde la concentración de las moléculas de ADN que representan los pesos determina el resultado. La suma se implementa mediante reacciones en las que múltiples especies se convierten en una única especie de suma ponderada.

En cuanto a las funciones de activación, se han desarrollado métodos para implementar la función sigmoide, incluso para valores mayores a uno, que se logran mediante un nuevo divisor molecular, que permite superar el cuello de botella del escalado. Las funciones ReLU y softmax también han sido implementadas usando reacciones moleculares y de ADN [14].

Una arquitectura de puerta de conmutación permite el control independiente de las funciones de transmisión de señal y de asignación de peso durante el cálculo. Variando la secuencia del dominio de reconocimiento se permite la conexión a diferentes puertas posteriores, mientras que variando la secuencia del dominio de ajuste de peso se pueden determinar los pesos asignados a las entradas. De este modo, se pueden implementar operaciones MAC de compartición de pesos a nivel molecular y construir redes neuronales convolucionales moleculares [15].

Las redes neuronales convolucionales (ConvNets) de ADN pueden implementarse codificando el kernel de convolución en la secuencia del dominio de ajuste de peso, y ajustando los valores de los pesos con la concentración de las moléculas sustrato de peso. El patrón de entrada se codifica con cadenas simples de ADN, donde la presencia o ausencia de una cadena representa un 1 o un 0, respectivamente. Se pueden usar moléculas de ajuste de peso para activar selectivamente conjuntos de pesos y permitir que las mismas moléculas de ADN se utilicen para diferentes tareas. Un proceso de max-pooling se aplica para reducir el tamaño del mapa de características mediante la aniquilación del píxel más pequeño entre dos.

III. Metodología

La estrategia a utilizar está basada en [16] simplificada con Tensorflow y Keras; cada cadena de ADN representa una topología de las capas ocultas de la red neuronal, sin embargo, esta propuesta solo pretende optimizar las capas ocultas, pues se debe preservar la forma de las capas de entrada y salida, tampoco se modifican los hiperparámetros, pues deben preservar características como el algoritmo de entrenamiento y la función de pérdida. Véase Fig 4.

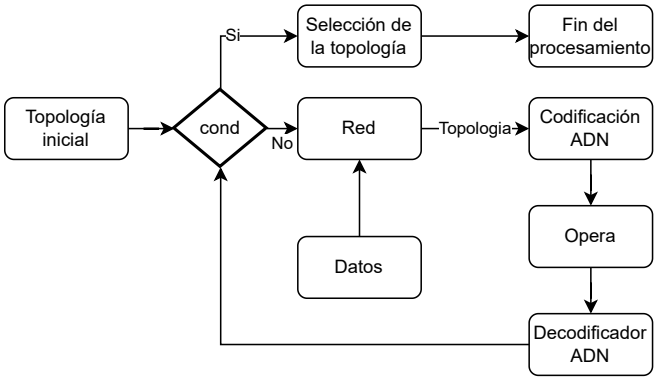

Fig. 4. Estrategia de construcción de topología. Elaboración propia.

Posteriormente se va a validar la capacidad de construcción del algoritmo para diferentes problemas, las pruebas iniciales se basan en el dataset Iris [17] y posteriormente en un dataset de datos abiertos de Colombia, y se verificará la estructura de la red y los pesos de las neuronas de entrada.

IV. Desarrollo

El código fuente, junto con los modelos desarrollados, está disponible públicamente en el repositorio de GitHub [1].

La codificación del problema se hizo con una técnica de etiquetado secuencial base 4 ADN, $A \rightarrow 0, T \rightarrow 1, C \rightarrow 2, G \rightarrow 3$.

[1] https://github.com/Michael-Jimenez-C/Redes-neuronales-basadas-en-ADN

En todos los modelos generados se utilizan callbacks para reducir el tiempo de entrenamiento, entre ellos *EarlyStopping* y *TerminateOnNaN*, que son funciones que se ejecutan en cada época de entrenamiento y realizan funciones específicas como detener el entrenamiento, guardar puntos de control y restaurar pesos [18].

IV-A. Métricas

Se utiliza principalmente la función de pérdida como métrica (*val_loss*) de forma que siempre se obtengan modelos que generalicen los datos correctamente, para el ejemplo se utilizó *Categorical Crossentropy*.

IV-B. Optimización de topología con ADN

Para codificar una red, se decidió utilizar frames, estos frames constan de una tupla representada en ADN que contiene la cantidad de neuronas en la capa y un indice que apunta a una función de activación, véase Fig 5.

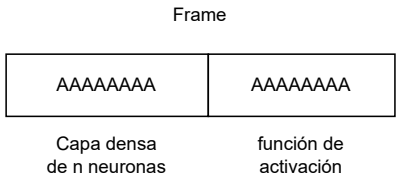

Fig. 5. Ejemplo de un Frame. Elaboración propia.

De forma que para representar una red se pueden encadenar frames, obteniendo algo similar a lo propuesto por [16], por ejemplo, AATGAAAT indica una red de 4 neuronas con función de activación ReLu. Véase Fig 6.

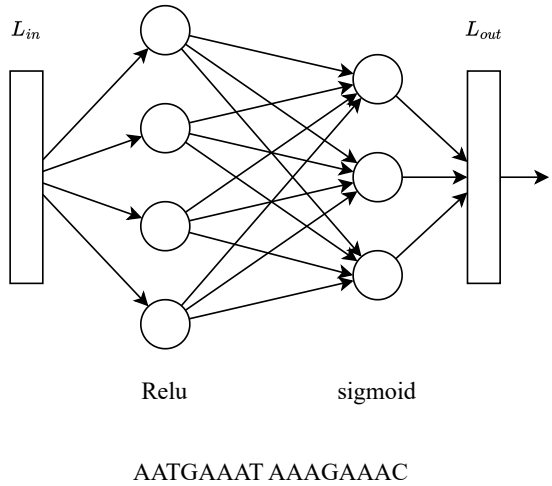

Fig. 6. Representación de una red. Elaboración propia.

Luego se construye la red con la API de Keras añadiendo sus capas de entrada y salida; y sus configuraciones. Al entrenar la red se toma el valor de la función de pérdida como función de aptitud. Finalmente, para recuperar la mejor topología se re-utiliza la función de construcción de redes para obtener la red óptima obtenida.

En este caso no es posible reducir la complejidad almacenando en cache la función de aptitud de las diferentes redes pues depende también del resultado del entrenamiento, por lo cual lo mejor será almacenar aquel modelo que tenga el error más bajo en cada generación.

IV-C. Pruebas realizadas

Se llevaron a cabo pruebas de funcionamiento utilizando Iris y la función de error *Categorical Crossentropy* para evaluar la estrategia. En este caso la entrada está sin transformaciones mientras que la salida está en formato categórico. Posteriormente, los resultado se transformaron mediante la función *argmax* (setosa:0,versicolor:1, virginica:2). Como resultado de esta estrategia se reduce el error, pasando de $0{,}98$ a $0{,}04$ Fig 7

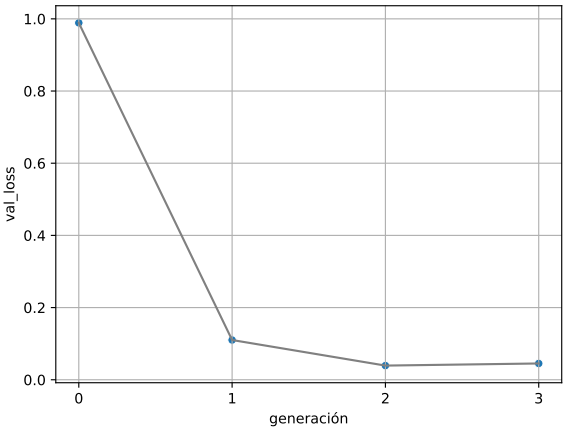

Fig. 7. Error por generación en el dataset Iris. Elaboración propia.

La topología lograda es $(26, linear)$, $(104, tanh)$, $(417, relu)$, $(644, linear)$,$(529, relu)$ lo cual codificado es "AATCCATATC AACAGTTATC TTCTCTTATC TATTTAAAAT TAAGGAATTG"

Su tabla de confusión es

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	46	4
virginica	0	4	46

TABLE I
MATRIZ DE CONFUSIÓN

Con lo cual se puede concluir que esta estrategia puede encontrar topologías óptimas para un problema de forma rápida y precisa, demostrado con la tabla I en la cual se puede ver que falla únicamente 8 en este caso, más por el dataset que por la arquitectura de la red.

En un intento posterior, se definió un valor de tolerancia de error igual a $0{,}8$, y se obtuvo una red valida en la primera iteración: $(10, relu)$, $(40, sigmoid)$, $(160, relu)$, $(640, linear)$, $(512, sigmoid)$, con error de $0{,}0806$

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	0
virginica	0	3	50

TABLE II

MATRIZ DE CONFUSIÓN SEGUNDO INTENTO

De lo cual se espera que conforme se asigne un mayor tiempo de entrenamiento y una menor tolerancia podrá encontrar eventualmente modelos más precisos.

V. CONCLUSIONES

- La técnica propuesta demuestra ser eficaz al identificar una topología que minimiza el error, como puede observarse en las figuras 7.
- La propuesta planteada aporta al desarrollo de herramientas basadas en algoritmos de optimización (ADN) que permitan la creación de topologías más complejas, incorporando diferentes tipos de capas y arquitecturas. Este enfoque facilita la exploración de configuraciones que podrían mejorar el rendimiento en tareas específicas.

REFERENCIAS

[1] S. Yang, B. Bögels, F. Wang, *et al.*, "Dna as a universal chemical substrate for computing and data storage," *Nature Reviews Chemistry*, vol. 8, pp. 179–194, Mar. 2024.

[2] G. Rozenberg and A. Salomaa, "Dna computing: New ideas and paradigms," in *Automata, Languages and Programming* (J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds.), vol. 1644 of *Lecture Notes in Computer Science*, pp. 295–311, Springer, Berlin, Heidelberg, 1999.

[3] V. Manca, C. Martín-Vide, and G. Paun, "New computing paradigms suggested by dna computing: computing by carving," *Biosystems*, vol. 52, no. 1, pp. 47–54, 1999.

[4] T. Kumar and S. Namasudra, "Chapter one - introduction to dna computing," in *Perspective of DNA Computing in Computer Science* (S. Namasudra, ed.), vol. 129 of *Advances in Computers*, pp. 1–38, Elsevier, 2023.

[5] IBM, "¿qué es el descenso del gradiente?." `https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model`, 2021.

[6] IBM, "¿qué es el descenso del gradiente?." `https://www.ibm.com/mx-es/topics/gradient-descent`, S.F.

[7] D. J. Montana, L. Davis, *et al.*, "Training feedforward neural networks using genetic algorithms.," in *IJCAI*, vol. 89, pp. 762–767, 1989.

[8] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Ieee Potentials*, vol. 13, no. 4, pp. 27–31, 1994.

[9] F. S. Caparrini, "Algoritmos genéticos." https://www.cs.us.es/~fsancho/Blog/posts/Algoritmos_Geneticos.md, S.F.

[10] M. Gestal, D. Rivero, J. R. Rabuñal, J. Dorado, and A. Pazos, *Introducción a los Algoritmos Genéticos y la Programación Genética*. Digitalia, 2010.

[11] A. M. Cuello, *Aprendizaje Automático con Algoritmos Genéticos y Redes Neuronales*. Proyecto fin de grado, Universidad politécnica de Madrid, June 2020.

[12] J. E. Ortiz Triviño and E. Acosta Martin, "Modelo general de codificación de rna en secuencias de adn y su aprendizaje basado en selección natural," *Revista de la Facultad de Medicina*, vol. 48, no. 4, pp. 224–231, 2000.

[13] J. E. Ortiz Triviño, "Computadores moleculares: una tecnología prometedora," *Revista de la Facultad de Medicina*, vol. 47, no. 2, pp. 98–101, 1999.

[14] X. Liu and K. K. Parhi, "Molecular and dna artificial neural networks via fractional coding," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 3, pp. 490–503, 2020.

[15] H. Pei, X. Xiong, T. Zhu, Y. Zhu, M. Cao, J. Xiao, L. Li, F. Wang, and C. Fan, "Molecular convolutional neural networks with dna regulatory circuits," *Nature Machine Intelligence*, vol. 4, no. 7, pp. 502–510, 2022.

[16] A. Sedeghat, "Dna-inspired representations for multi-dimensional architectures." https://medium.com/@amin32846/dna-inspired-representations-for-multi-dimensional-architectures-b1e83d89b642, 2024.

[17] D. Dua and C. Graff, "Uci machine learning repository." https://archive.ics.uci.edu/dataset/53/iris, 2019.

[18] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.