Análisis comparativo de redes de Kolmogorov-Arnold y perceptrones multicapa para la representación de funciones especiales

Luis Enrique Ramon-Pedrero, José Adán Hernández-Nolasco, Pablo Pancardo

Universidad Juárez Autónoma de Tabasco, México

{242H21005, Adan.hernandez, pablo.pancardo}@alumno.ujat.mx

Resumen. Este artículo presenta un análisis comparativo entre las redes neuronales de Kolmogorov-Arnold (KAN) y el perceptrón multicapa (MLP) para tareas de representación de funciones continuas unidimensionales. El objetivo fue evaluar ambas arquitecturas bajo condiciones de entrenamiento homogéneas, considerando no solo la precisión predictiva, sino también la eficiencia computacional durante la inferencia. Para ello, se construyó un conjunto de datos sintéticos a partir de cinco funciones objetivo: *bessel*, *bessel_esferica*, *mathieu_je*, *sin_20x* y *sin_30x_exp*. Ambos modelos se entrenaron utilizando el mismo pipeline de aprendizaje supervisado y se compararon mediante las métricas MSE_{test}, R^2_{test} y *flops_forward*. Los resultados muestran que KAN logró consistentemente un menor error y un mejor ajuste que MLP en todas las funciones evaluadas. Además, KAN requirió un menor costo de inferencia, con 704 FLOPs frente a 832 FLOPs para el modelo MLP. En general, la evidencia experimental sugiere que KAN ofrece una relación precisión-costo más favorable que MLP en problemas de representación funcional. Sin embargo, la generalización de estos hallazgos debe validarse con mayor profundidad en otras categorías de funciones y configuraciones experimentales.

Palabras clave: Redes neuronales, redes de Kolmogorov-Arnold, perceptrón multicapa, aproximación de funciones, eficiencia computacional.

Comparative Analysis of Kolmogorov-Arnold Networks and Multilayer Perceptrons for Representing Special Functions

Abstract. This paper presents a comparative analysis between Kolmogorov-Arnold Networks (KAN) and Multi-Layer Perceptron (MLP) neural networks for one-dimensional continuous function representation tasks. The objective was to evaluate both architectures under homogeneous training conditions, considering not only predictive

accuracy but also computational efficiency during inference. To this end, a synthetic dataset was constructed from five target functions: *bessel*, *bessel_esferica*, *mathieu_je*, *sin_20x*, and *sin_30x_exp*. Both models were trained using the same supervised learning pipeline and compared through the metrics MSE_{test}, R^2_{test}, and *flops_forward*. The results show that KAN consistently achieved lower error and better fit than MLP across all evaluated functions. In addition, KAN required a lower inference cost, with 704 FLOPs versus 832 FLOPs for the MLP model. Overall, the experimental evidence suggests that KAN provides a more favorable accuracy-cost tradeoff than MLP in functional representation problems. However, the generalization of these findings should be further validated on additional function categories and experimental configurations.

Keywords: Neural networks, Kolmogorov-Arnold networks, multi-layer perceptron, function approximation, computational efficiency.

1. Introducción

Las redes Perceptrón Multicapa (MLP) son redes neuronales de alimentación hacia adelante compuestas por múltiples capas de neuronas [4,10]. Se utilizan ampliamente para modelar relaciones no lineales complejas mediante entrenamiento basado en descenso del gradiente y retropropagación, y su capacidad de aproximación universal ha sustentado su uso en clasificación, regresión y representación funcional [3].

Por otra parte, el teorema de representación de Kolmogorov-Arnold establece que una función continua multivariable puede expresarse como superposición finita de funciones continuas univariadas y sumas [6,2]. A partir de esta base, Liu et al. propusieron las redes de Kolmogorov-Arnold (KAN), donde las funciones univariadas asociadas a las conexiones son aprendibles, a diferencia de las MLP, que emplean activaciones fijas y ajustan principalmente pesos y sesgos [9].

Estudios recientes han mostrado resultados mixtos al comparar KAN y MLP, por lo que resulta pertinente realizar evaluaciones controladas en tareas específicas de representación funcional [9,15,11]. En este trabajo se compara el desempeño de una KAN respecto a una MLP en términos de exactitud y eficiencia computacional al representar funciones especiales y funciones elementales de alta frecuencia.

El estudio se plantea como una evaluación controlada inicial, limitada a funciones continuas unidimensionales. Esta delimitación permite aislar el efecto de la arquitectura bajo condiciones homogéneas de entrenamiento, aunque no pretende generalizar los resultados a problemas multivariables, de mayor dimensionalidad o escenarios aplicados, los cuales se consideran líneas futuras de validación.

2. Trabajo relacionado

Las redes KAN se inspiran en el teorema de representación de Kolmogorov-Arnold y difieren de las MLP en que emplean funciones aprendibles en las aristas, mientras que las MLP utilizan activaciones fijas en los nodos. Liu et al. presentan su arquitectura, fundamento matemático, leyes de escala y métodos de simplificación, mostrando que KAN puede superar a MLP en diversas tareas [9].

Diversos trabajos han extendido o evaluado KAN en distintos escenarios. En detección de intrusiones para entornos IoT, Abd Elaziz et al. reemplazan capas MLP por capas basadas en KAN, mejorando el desempeño del sistema [1]. En ecuaciones diferenciales y aprendizaje de operadores, Shukla et al. comparan KAN y MLP, reportando que algunas variantes de KAN pueden superar a MLP, aunque también pueden presentar inestabilidad en funciones polinomiales de orden elevado [11].

También se han propuesto aplicaciones de KAN en arquitecturas híbridas y datos reales. Li et al. integran una capa KAN en U-Net para segmentación y generación de imágenes médicas, obteniendo mejoras frente a variantes previas [7]. Asimismo, Vaca-Rubio et al. aplican KAN al pronóstico de tráfico satelital, reportando mayor precisión y eficiencia con menos parámetros que MLP [14].

Estos antecedentes muestran el interés reciente por KAN y motivan comparaciones controladas frente a MLP en tareas específicas de aproximación funcional, como las funciones especiales y señales de alta frecuencia consideradas en este trabajo.

3. Metodología

La metodología empleada en este trabajo se estructura en siete etapas.

3.1. Definición del problema de representación funcional

Se delimitó el problema como una tarea de representación de funciones continuas en una variable, con el propósito de comparar dos familias de modelos neuronales: KAN y MLP. La formulación metodológica se orientó a evaluar, bajo condiciones homogéneas, la capacidad de cada modelo para reproducir funciones objetivo mediante entrenamiento supervisado y validación en datos no vistos.

Desde el punto de vista matemático, una MLP aproxima una función mediante composiciones de transformaciones afines y activaciones no lineales fijas:

$$f_{\mathrm{MLP}}(x) = W_L \sigma \left(W_{L-1} \sigma \left(\cdots \sigma(W_1 x + b_1) \right) \right) + b_L. \tag{1}$$

En contraste, una KAN reemplaza los pesos escalares por funciones univariadas aprendibles en las aristas, de modo que una capa puede expresarse como:

$$x_j^{(l+1)} = \sum_i \phi_{ij}^{(l)} \left(x_i^{(l)} \right), \tag{2}$$

donde $\phi_{ij}^{(l)}$ representa una función aprendible, usualmente parametrizada mediante splines. Esta diferencia estructural permite que KAN ajuste localmente la forma de la función objetivo, mientras que MLP depende de activaciones fijas combinadas con pesos entrenables.

3.2. Construcción del conjunto de funciones y datos de evaluación

Se seleccionó un conjunto de funciones representativas para el estudio: `bessel`, `bessel_esferica`, `mathieu_je`, `sin_20x` y `sin_30x_exp`. Para cada función se generaron datos sintéticos en el dominio definido, con muestreo uniforme, partición entrenamiento/prueba y preprocesamiento mediante normalización. Esta etapa aseguró consistencia entre experimentos y comparabilidad entre modelos.

El propósito de utilizar estas funciones es evaluar la capacidad expresiva de ambas arquitecturas frente a datos de alta complejidad matemática, determinando cuál red tiene una mayor eficiencia y exactitud al modelar este tipo de datos.

3.3. Implementación y configuración de los modelos KAN y MLP

Se implementaron ambos modelos dentro de un mismo pipeline de entrenamiento y evaluación, manteniendo una configuración estable para todos los experimentos. Se fijaron las arquitecturas base y los hiperparámetros principales de optimización, con control de semilla para reproducibilidad. Esta decisión metodológica permitió que las diferencias observadas respondieran al modelo y no a variaciones del procedimiento.

3.4. Ejecución del entrenamiento y evaluación comparativa

Para cada función objetivo se entrenó una KAN y una MLP bajo el mismo esquema supervisado. Las entradas se normalizaron a $[-1, 1]$ y, cuando fue necesario, también se normalizó la salida para estabilizar el entrenamiento. Las predicciones se desnormalizaron antes de calcular las métricas en la escala original.

Durante el entrenamiento se registró la pérdida por época y se generaron gráficas de valor real contra predicción e histogramas de residuos. Estas evidencias permitieron evaluar la convergencia, la forma del error y la capacidad de cada modelo para reproducir la función objetivo.

Las mayores dificultades aparecieron en funciones de alta frecuencia, especialmente en `sin_30x_exp`, donde la combinación de oscilaciones rápidas y modulación exponencial incrementó la complejidad de aproximación.

3.5. Análisis de la relación precisión–costo computacional

La comparación final se planteó bajo un enfoque multicriterio que considera precisión predictiva, costo estimado de inferencia y tiempo de entrenamiento. Se

analizaron conjuntamente `mse_test`, `r2_test`, `flops_forward` y t_{train}, con el fin de identificar el equilibrio entre calidad de representación, costo de inferencia y costo práctico de entrenamiento.

3.6. Integración y documentación de los resultados para su interpretación

Los resultados se consolidaron en un esquema de reporte unificado (tablas y figuras) que facilita la trazabilidad de cada experimento. Esta integración permitió estructurar de forma clara la discusión y las conclusiones, además de dejar una base metodológica reproducible para futuras extensiones del estudio.

3.7. Alcance experimental y limitaciones

La evaluación se restringió a funciones continuas unidimensionales para comparar KAN y MLP bajo condiciones homogéneas de dominio, partición de datos, función de pérdida y entrenamiento. Por ello, los resultados no deben interpretarse como una validación general en problemas multivariables, de mayor dimensionalidad o escenarios aplicados con ruido, restricciones físicas o datos reales. Asimismo, aunque se reporta `flops_forward` como métrica controlada de costo de inferencia, futuras evaluaciones deberán complementar este criterio con tiempo de inferencia, consumo de memoria y estabilidad numérica en hardware específico.

4. Datos y configuración experimental

4.1. Datos y partición

Los datos se generaron sintéticamente evaluando las funciones objetivo del estudio (`bessel`, `bessel_esferica`, `mathieu_je`, `sin_20x` y `sin_30x_exp`) sobre el dominio $[0, 20]$. Para cada función se muestrearon $n\_\text{samples} = 1000$ puntos mediante muestreo uniforme. La partición se realizó de forma aleatoria con `train_split`=0.8 para entrenamiento y el resto para prueba, utilizando `seed`=42 para asegurar reproducibilidad.

Las entradas se normalizaron a $[-1, 1]$. La salida se normalizó únicamente cuando su magnitud superó el umbral definido (`threshold`=50), conservando los parámetros de normalización para desescalar las predicciones durante la evaluación y reportar métricas en la escala original.

4.2. Modelos y entrenamiento

Se compararon dos arquitecturas compactas para mantener un presupuesto computacional comparable: KAN con `width=[1,16,1]`, `grid=19`, `k=3`; y MLP con capas `[1,16,16,1]` y activación `tanh`. Ambos modelos se entrenaron minimizando `MSELoss` con `Adam`, `epochs`=1200, `learning_rate`=0.001 y `batch_size=full`.

La configuración base busca equilibrar capacidad de aproximación y costo de inferencia: MLP usa dos capas ocultas de 16 neuronas, mientras que KAN usa una capa oculta de ancho 16 con funciones por arista parametrizadas mediante splines cúbicos. Dado que no se realizó una búsqueda exhaustiva, el análisis de sensibilidad sobre hiperparámetros clave se plantea como trabajo futuro.

4.3. Evaluación y criterio de costo

La evaluación reportó métricas en la escala original: `mse_test` y `r2_test`. Además, se calculó `flops_forward` como estimación del costo computacional de una pasada hacia adelante y se midió el tiempo total de entrenamiento, denotado como t_{train}, en segundos. Esta métrica complementa el análisis de FLOPs con una medida práctica del costo observado durante el entrenamiento.

Las métricas se definieron formalmente como:

$$MSE_{test} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3}$$

$$R^2_{test} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4}$$

$$\text{FLOPs}_{\text{forward}} = \sum_{l=1}^{L}\text{FLOPs}_l, \tag{5}$$

donde y_i es el valor real, $\hat{y}_i$ la predicción, $\bar{y}$ el promedio de los valores reales, n el número de muestras de prueba y FLOPs_l el costo estimado de la capa l. Adicionalmente, t_{train} corresponde al tiempo total, en segundos, requerido para completar las épocas de entrenamiento de cada modelo bajo el mismo entorno de ejecución.

La interpretación de los resultados combinó métricas cuantitativas y evidencia gráfica. Un menor `mse_test` indica menor error promedio, mientras que un `r2_test` cercano a 1 refleja mayor capacidad para explicar la variabilidad de la función objetivo. Valores negativos de `r2_test` indican un desempeño inferior al de una predicción basada en el promedio. Además, los histogramas de residuos permiten observar si los errores se concentran cerca de cero, lo cual evidencia un ajuste más estable.

5. Resultados

5.1. Comparación KAN vs MLP bajo presupuesto computacional

Se compararon KAN y MLP usando métricas de precisión en prueba, costo estimado de inferencia y tiempo de entrenamiento. La comparación consideró `mse_test`, `r2_test`, `flops_forward` y t_{train}. La Tabla 1 resume los resultados para `bessel`, `bessel_esferica`, `mathieu_je`, `sin_20x` y `sin_30x_exp`.

Tabla 1. Desempeño en prueba, costo estimado de inferencia y tiempo de entrenamiento para MLP y KAN.

Función	**MLP**				**KAN**			
	MSE_{test}	R^2_{test}	FLOPs	t_{train} (s)	MSE_{test}	R^2_{test}	FLOPs	t_{train} (s)
bessel	8.92e-03	-0.0405	832	10.33	4.93e-06	0.9994	704	50.27
bessel_esferica	2.26e-03	0.9700	832	12.80	3.48e-07	0.9999	704	55.61
mathieu_je	2.99e-06	0.9885	832	10.09	5.12e-09	0.9999	704	52.75
sin_20x	5.25e-01	-0.0075	832	10.31	2.82e-03	0.9945	704	50.07
sin_30x_exp	1.19e-01	-0.0611	832	10.03	6.30e-02	0.4367	704	50.48

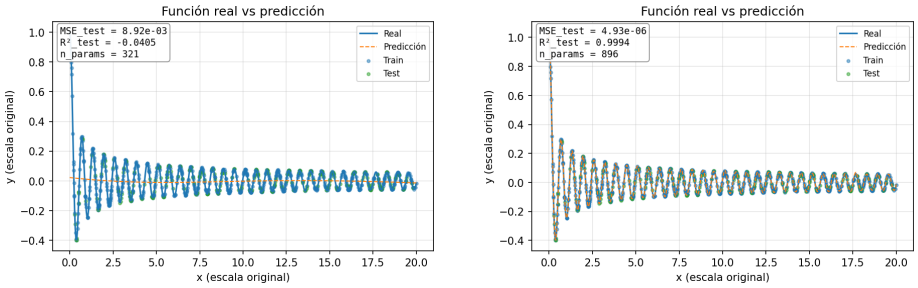

Fig. 1. Ajuste real vs. predicción para `bessel`. Izquierda: MLP. Derecha: KAN.

Los resultados muestran que KAN mejora consistentemente a MLP en `mse_test` y `r2_test`, alcanzando valores de R^2_{test} cercanos a 1 en `bessel`, `bessel_esferica`, `mathieu_je` y `sin_20x`. Además, KAN reduce el costo de inferencia estimado de 832 a 704 FLOPs. Sin embargo, su tiempo de entrenamiento es mayor: aproximadamente 50–56 s frente a 10–13 s de MLP. Por tanto, KAN presenta ventaja en precisión e inferencia, mientras que MLP resulta más eficiente durante el entrenamiento.

5.2. Resultados por función

bessel Para la función `bessel`, KAN supera claramente a MLP en generalización con menor costo estimado de inferencia. En test, MLP obtiene $MSE_{\text{test}} = 8{,}92 \times 10^{-3}$ y $R^2_{\text{test}} = -0{,}040517$ con 832 FLOPs, mientras que KAN alcanza $MSE_{\text{test}} = 4{,}93 \times 10^{-6}$ y $R^2_{\text{test}} = 0{,}999425$ con 704 FLOPs. Como se observa en la Fig. 1, KAN sigue con mayor fidelidad la relación real–predicción. Además, la Fig. 2 muestra una mayor concentración de residuos alrededor de cero para KAN.

bessel_esferica En `bessel_esferica`, KAN vuelve a presentar mejor rendimiento que MLP con menor costo de inferencia. MLP reporta $MSE_{\text{test}} = 2{,}26 \times 10^{-3}$ y $R^2_{\text{test}} = 0{,}970043$ (832 FLOPs), mientras que KAN obtiene $MSE_{\text{test}} = 3{,}48 \times 10^{-7}$ y $R^2_{\text{test}} = 0{,}999995$ (704 FLOPs). La Fig. 3 confirma

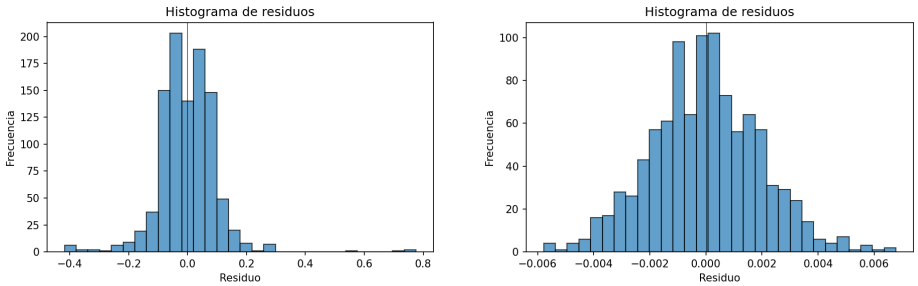

Fig. 2. Histograma de residuos para `bessel`. Izquierda: MLP. Derecha: KAN.

el mejor ajuste de KAN. De forma consistente, la Fig. 4 evidencia residuos de menor magnitud para KAN.

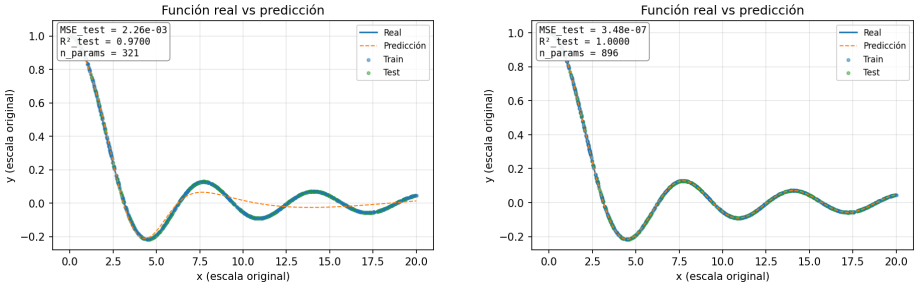

Fig. 3. Ajuste real vs. predicción para `bessel_esferica`. Izquierda: MLP. Derecha: KAN.

mathieu_je Para `mathieu_je`, KAN mantiene una ventaja clara frente a MLP en el conjunto de prueba. MLP alcanza $MSE_{\text{test}} = 2{,}99 \times 10^{-6}$ y $R^2_{\text{test}} = 0{,}988568$ (832 FLOPs), mientras que KAN logra $MSE_{\text{test}} = 5{,}12 \times 10^{-9}$ y $R^2_{\text{test}} = 0{,}999980$ (704 FLOPs). La Fig. 5 muestra la mayor fidelidad del ajuste con KAN. Asimismo, la Fig. 6 confirma una distribución de errores más concentrada cerca de cero.

sin_20x En `sin_20x`, la diferencia entre arquitecturas es especialmente marcada. MLP presenta $MSE_{\text{test}} = 5{,}25 \times 10^{-1}$ y $R^2_{\text{test}} = -0{,}007591$ con 832 FLOPs, mientras que KAN obtiene $MSE_{\text{test}} = 2{,}82 \times 10^{-3}$ y $R^2_{\text{test}} = 0{,}994583$ con 704 FLOPs. La Fig. 7 evidencia que KAN captura mejor el comportamiento global de la señal. En la Fig. 8 se observa también una reducción clara de residuos para KAN.

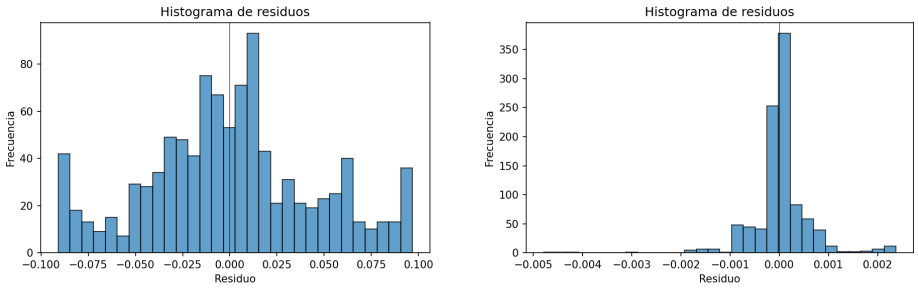

Fig. 4. Histograma de residuos para `bessel_esferica`. Izquierda: MLP. Derecha: KAN.

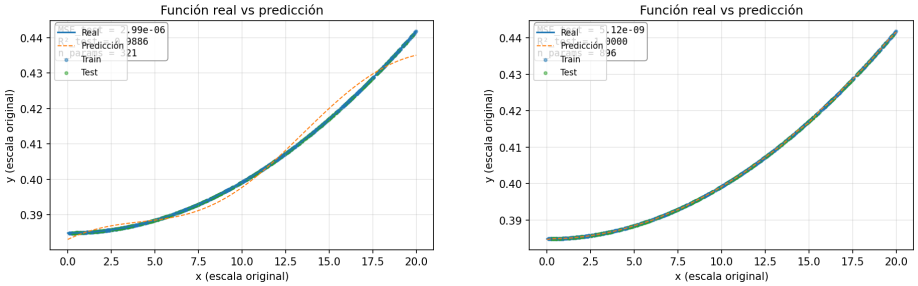

Fig. 5. Ajuste real vs. predicción para `mathieu_je`. Izquierda: MLP. Derecha: KAN.

`sin_30x_exp` En `sin_30x_exp`, KAN también mejora frente a MLP, aunque con una brecha menor que en `bessel` o `mathieu_je`. En test, MLP obtiene $MSE_{\text{test}} = 1{,}19 \times 10^{-1}$ y $R^2_{\text{test}} = -0{,}061152$ (832 FLOPs), mientras que KAN alcanza $MSE_{\text{test}} = 6{,}30 \times 10^{-2}$ y $R^2_{\text{test}} = 0{,}436762$ (704 FLOPs). La comparación de la Fig. 9 sugiere una mejora del ajuste con KAN, aunque aún con desviaciones apreciables. Esto es consistente con la Fig. 10, donde KAN reduce la dispersión residual respecto a MLP, pero sin la concentración observada en los casos de mejor desempeño.

6. Discusión

KAN mostró una mejora consistente frente a MLP, al reducir `mse_test` y aumentar `r2_test` en todas las funciones evaluadas. Las mayores ventajas se observaron en `bessel`, `bessel_esferica`, `mathieu_je` y `sin_20x`; en cambio, `sin_30x_exp` presentó una mejora más moderada, asociada a su mayor complejidad oscilatoria.

En términos computacionales, KAN obtuvo menor costo estimado de inferencia, con 704 FLOPs frente a 832 FLOPs de MLP. Sin embargo, MLP requirió

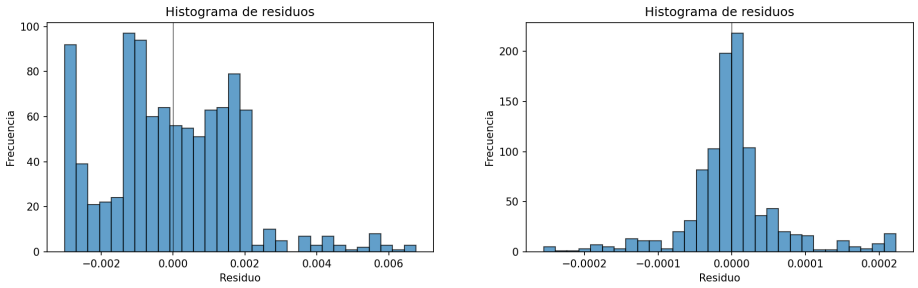

Fig. 6. Histograma de residuos para `mathieu_je`. Izquierda: MLP. Derecha: KAN.

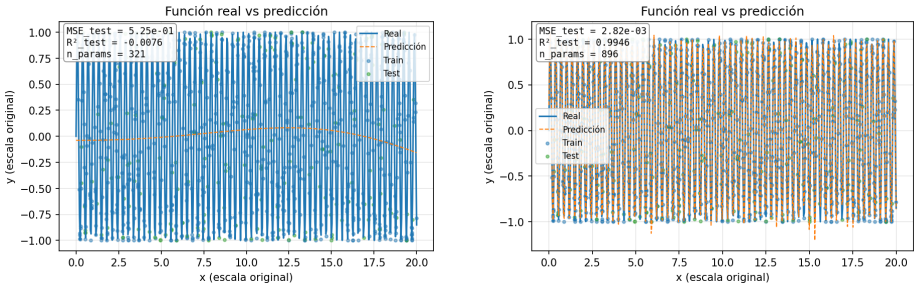

Fig. 7. Ajuste real vs. predicción para `sin_20x`. Izquierda: MLP. Derecha: KAN.

menor tiempo de entrenamiento, por lo que la ventaja de KAN se concentra principalmente en precisión e inferencia, no en costo computacional global.

Este comportamiento puede explicarse por las funciones univariadas aprendibles de KAN, que permiten ajustar variaciones locales de funciones estructuradas u oscilatorias con mayor flexibilidad que las activaciones fijas de MLP. No obstante, los resultados deben interpretarse dentro del alcance definido, ya que el estudio se limita a funciones unidimensionales, una configuración experimental fija y métricas parciales de eficiencia.

El caso `sin_30x_exp` sugiere que funciones con alta frecuencia y modulación de amplitud pueden requerir mayor capacidad, una malla más fina o estrategias adaptativas de muestreo para mejorar la aproximación.

7. Conclusiones y trabajos futuros

En este trabajo se compararon MLP y KAN en tareas de representación funcional, considerando precisión, costo estimado de inferencia mediante `flops_forward` y tiempo de entrenamiento t_{train}. Los resultados muestran que KAN obtuvo menor `mse_test`, mayor R^2_{test} y menor número de FLOPs que MLP en las funciones evaluadas.

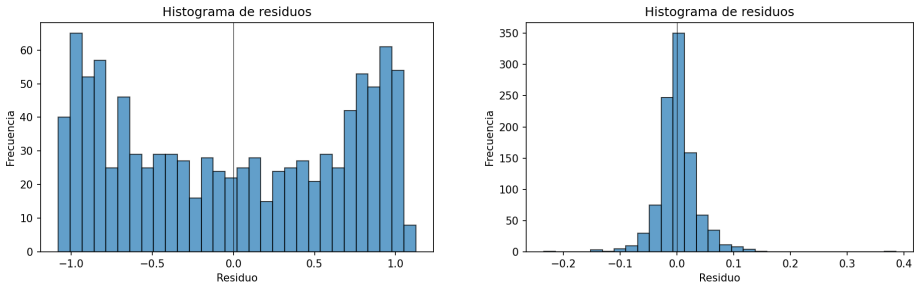

Fig. 8. Histograma de residuos para `sin_20x`. Izquierda: MLP. Derecha: KAN.

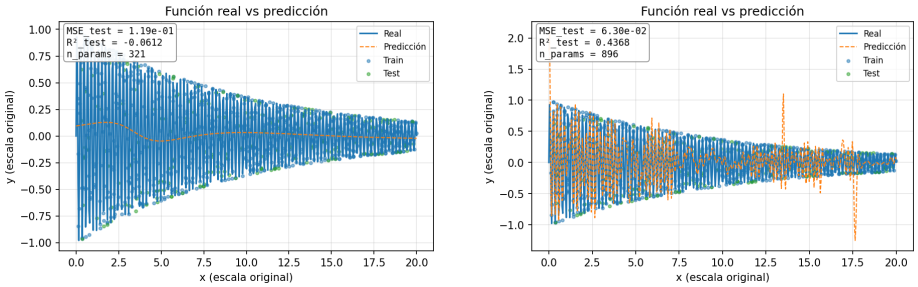

Fig. 9. Ajuste real vs. predicción para `sin_30x_exp`. Izquierda: MLP. Derecha:KAN.

Sin embargo, KAN requirió mayor tiempo de entrenamiento, por lo que su ventaja se concentra en calidad de aproximación e inferencia, no en eficiencia computacional global. Así, KAN presenta una relación precisión–inferencia favorable, mientras que MLP conserva ventaja en entrenamiento.

Como trabajo futuro, se propone validar estos hallazgos en funciones multivariables, problemas de mayor dimensionalidad y configuraciones adicionales. También se plantea ampliar el análisis de sensibilidad, incorporar tiempo de inferencia, memoria y estabilidad numérica, e incluir arquitecturas modernas de aproximación funcional, como Fourier Features [13], SIREN [12], redes residuales [5] y operadores neuronales [8], con el fin de contextualizar la ventaja observada de KAN frente a modelos especializados.

Referencias

1. Abd Elaziz, M., Ahmed Fares, I., Aseeri, A.O.: CKAN: Convolutional kolmogorov-arnold networks model for intrusion detection in IoT environment. IEEE Access 12, 134837–134851 (2024)
2. Arnold, V.I.: On functions of three variables. Doklady Akademii Nauk SSSR 114(4), 679–681 (1957)

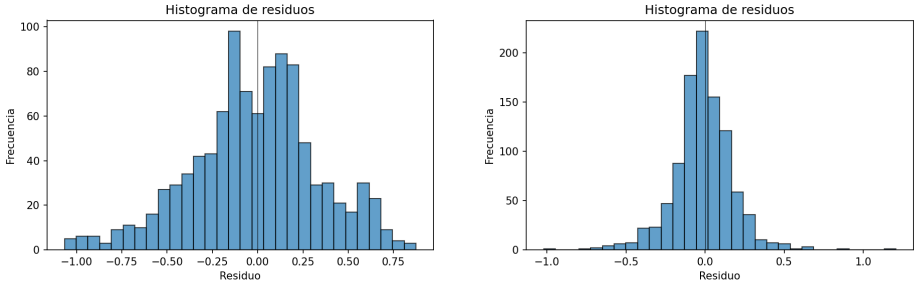

Fig. 10. Histograma de residuos para `sin_30x_exp`. Izquierda: MLP. Derecha: KAN.

3. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems 2(4), 303–314 (1989)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA (2016)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
6. Kolmogorov, A.N.: On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In: Doklady Akademii Nauk. vol. 114, pp. 953–956. Russian Academy of Sciences (1957)
7. Li, C., Liu, X., Li, W., Wang, C., Liu, H., Liu, Y., Chen, Z., Yuan, Y.: U-KAN makes strong backbone for medical image segmentation and generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 4652–4660 (2025)
8. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. In: International Conference on Learning Representations (2021)
9. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T.Y., Tegmark, M.: Kan: Kolmogorov-Arnold networks. arXiv preprint arXiv:2404.19756 (2024)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
11. Shukla, K., Toscano, J.D., Wang, Z., Zou, Z., Karniadakis, G.E.: A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. Computer Methods in Applied Mechanics and Engineering 431, 117290 (2024)
12. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Advances in Neural Information Processing Systems. vol. 33, pp. 7462–7473 (2020)
13. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. In: Advances in Neural Information Processing Systems. vol. 33, pp. 7537–7547 (2020)
14. Vaca-Rubio, C.J., Blanco, L., Pereira, R., Caus, M.: Kolmogorov-Arnold networks (Kans) for time series analysis. arXiv preprint arXiv:2405.08790 (2024)

15. Yu, R., Yu, W., Wang, X.: KAN or MLP: A fairer comparison. arXiv preprint arXiv:2407.16674 (2024)