CONTRIBUCIÓN AL DISEÑO DE PATRONES DE CARGA DE UN NÚCLEO PWR ASISTIDO POR INTELIGENCIA ARTIFICIAL

PABLO PALLARÉS FONT DE MORA
Estudiante de Doctorado
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ENRIQUE QUINTANA-ORTÍ
Profesor
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

RAFAEL MIRÓ HERRERO
Profesor
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

JOSE GARCÍA SANJUAN
Técnico Superior Investigación
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TERESA BARRACHINA CELDA
Profesora
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

JAVIER JORGE LÓPEZ
Ingeniero nuclear
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

INTRODUCCIÓN

El diseño del núcleo es el estudio a priori del comportamiento del núcleo de un reactor a lo largo de un ciclo entre dos recargas de combustible. Se trata de un cálculo termohidráulico y neutrónico en estado estacionario cuasi-estático, cuyo objetivo es obtener los valores de las variables de control (p. ej., concentración de boro, posición de barras de control, potencia, nivel de quemado, duración del ciclo) para operar el reactor en estado estacionario (k_{eff}=1) a lo largo de un ciclo de una planta. Para llevar a cabo este estudio, el ciclo se divide entre un cierto número de periodos de tiempo definidos por el aumento del quemado del combustible durante cada etapa. El grado de quemado de combustible es una medida de la cantidad de energía que ya se ha extraído de la fuente primaria de combustible nuclear y se expresa como la energía liberada por masa inicial de combustible en GWd/tHM (en GigaWatts-día/tonelada métrica de metal pesado). Una parte importante de este tipo de estudios es la gestión de los archivos que contienen el histórico de simulación. De hecho, para determinar las secciones transversales en una determinada etapa del ciclo, es necesario conocer las condiciones de estado del reactor durante todas las etapas anteriores (historial).

METODOLOGÍA

Los diferentes códigos informáticos utilizados para este trabajo han sido: PARCS (v3.4.2) [1] y PATHS (v1.08) [2]. PATHS es un código termohidráulico que se puede acoplar con PARCS. Este último calcula la parte neutrónica del problema. PATHS utiliza un modelo de *drift flux* para caudal bifásico, en lugar del típico modelo de caudal bifásico de 6 ecuaciones utilizado por los códigos de sistema. También se han utilizado Matlab© y Python para tareas de apoyo a la realización del trabajo, como la generación automática de archivos de entrada para PATHS y PARCS, con una distribución aleatoria del patrón de carga (igual en ambos códigos), mostrándose un ejemplo en la **Figura 1**.

```
GEOM
      geo_dim          17 17 28  2  2   !nasyx,nasyy,nz
      rad_conf
      0  0  0  0  0  0 22 22 22 22 22  0  0  0  0  0  0
      0  0  0  0 22 22 22  7  7  7 22 22 22  0  0  0  0
      0  0  0 22 22  7  8  6  2  6  8  7 22 22  0  0  0
      0  0 22 22  7  6  3  2  4  2  3  6  7 22 22  0  0
      0 22 22  7  2  3  2  3  2  3  2  3  2  7 22 22  0
      0 22  7  6  3  2  3  1  3  1  3  2  3  6  7 22  0
     22 22  8  3  2  3  2  3  2  3  2  3  2  3  8 22 22
     22  7  6  2  3  1  3  2  3  2  3  1  3  2  6  7 22
     22  7  2  3  2  3  2  3  2  3  2  3  2  3  2  7 22
     22  7  6  2  3  1  3  2  3  2  3  1  3  2  6  7 22
     22 22  8  3  2  3  2  3  2  3  2  3  2  3  8 22 22
      0 22  7  6  3  2  3  1  3  1  3  2  3  6  7 22  0
      0 22 22  7  2  3  2  3  2  3  2  3  2  7 22 22  0
      0  0 22 22  7  6  3  2  4  2  3  6  7 22 22  0  0
      0  0  0 22 22  7  8  6  2  6  8  7 22 22  0  0  0
      0  0  0  0 22 22 22  7  7  7 22 22 22  0  0  0  0
      0  0  0  0  0  0 22 22 22 22 22  0  0  0  0  0  0
```

Figura 1. Ejemplo de patrón de carga de un reactor PWR (entrada PARCS).

El acceso a PARCS y PATHS se ha obtenido gracias a la participación en el proyecto CAMP-España, patrocinado por el CSN en colaboración con la NRC.

La planta analizada es de un diseño Westinghouse, con 2587 MW de potencia nominal térmica, con 157 elementos combustibles de dos diseños mecánicos diferentes y distintos grados de enriquecimiento en U-235, trabajando en su primer ciclo. Los conjuntos de secciones eficaces para estos segmentos de combustible se encontraban en formato PMAXS (con *branches* históricas e instantáneas) que es uno de los formatos que permite PARCS.

A medida que el reactor está en funcionamiento, el quemado del combustible aumenta progresivamente y la concentración de boro debe de disminuir con cada paso de tiempo para mantener el reactor en estado crítico. Así, hay un momento en el que la reactividad positiva debida a la reducción de boro en el sistema no puede ser compensada por la pérdida de reactividad debida al quemado de combustible (se suponen las barras de control extraídas como condición de diseño). En ese momento, el código detecta el tiempo (y los GWd/tHM) en el que el valor de k_{eff} es inferior a 1 y, por tanto, el reactor se vuelve subcrítico, lo que confirma que ha llegado al punto final de su ciclo de operación.

Los resultados de PATHS/PARCS permiten acceder a muchas variables de especial interés para desarrollar un metamodelo, que puede ser utilizado en futuros trabajos de optimización de patrones (con técnicas de *Simulated Annealing*, por ejemplo), como los de interés en el trabajo, ya que presentan restricciones desde el punto de vista económico, de seguridad y de licenciamiento. A saber: Duración del ciclo, Concentración de Boro, Potencia de Pico, correlaciones de Flujo de Calor Crítico (CHF), Temperatura de Pico de Vaina (PCT) y su posición, DNBR o *Dryout* y su posición.

Existe un creciente interés en utilizar herramientas de Inteligencia Artificial (IA) para acelerar este tipo de cálculos [3,4,5,6]. En este trabajo se ha desarrollado un programa en Matlab que genera automáticamente 25000 patrones aleatorios de carga (con ciertas restricciones) y por ende 25000 archivos de entrada para PARCS/PATHS con estos patrones. La **Figura 2** muestra un esquema de las restricciones aplicadas al programa generador aleatorio (no debe haber dos elementos combustibles iguales uno al lado del otro), aprovechando la simetría de cuarto de núcleo, cada número representa un tipo de combustible diferente (mecánico, enriquecimiento, etc.).

Los ficheros de entrada se ejecutan en el cluster Linux de computación del ISIRYM. Este representa un problema de gestión importante, ya que los archivos de salida ocupan mucho espacio en el disco duro y deben comprimirse una vez finalizado cada caso, borrando los innecesarios, ya que de lo contrario se agota el almacenamiento. Esto implica un proceso de automatización con *scripts* de Linux. El tiempo de ejecución de los 25000 casos utilizando 64 núcleos al mismo tiempo toma alrededor de 10 días (unos 20 minutos

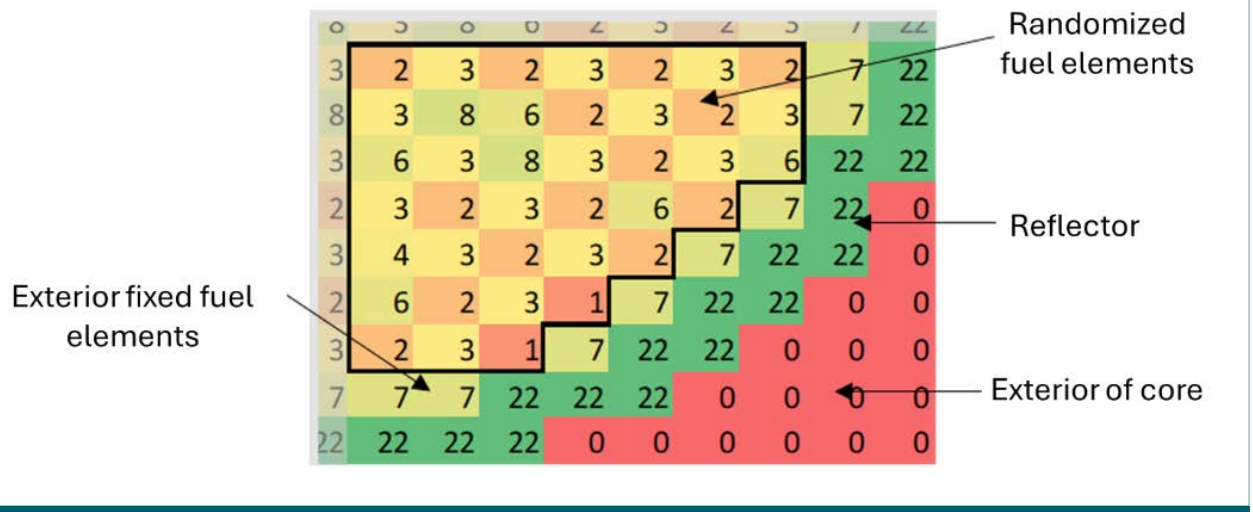

Figura 2. Cuadrante inferior derecho y posiciones variables de los elementos combustibles en el núcleo.

de ejecución por caso). El postprocesamiento de los resultados de PARCS (con Matlab) también es un proceso bastante rápido, realizándose también en el cluster. Lo mismo para el postprocesamiento de PATHS.

La gran cantidad de datos (~5 TB), grandes archivos de salida de datos 3D, generados con las 25000 simulaciones se deben gestionar automáticamente para extraer los datos necesarios para poder entrenar una red neuronal (RN) con el fin de obtener un metamodelo, que se utilizará para determinar en próximos trabajos diferentes configuraciones óptimas de los elementos combustibles en función de ciertos criterios.

La generación de un número determinado de configuraciones de combustible diferentes de forma aleatoria debe cumplir varios criterios (número correcto de cada tipo de elemento combustible, imposibilidad de que dos elementos combustibles idénticos estén pegados entre sí, etc.). Las salidas de los programas de postprocesado automático de la ingente cantidad de resultados obtenidos (concentración de Boro, duración del ciclo, k_{eff}, PCT, Potencia Máxima, etc.), que formarán la base de datos que servirá de entrada para el entrenamiento y verificación de la red neuronal generada (Keras) [7], cuyo objetivo es obtener un gemelo digital con el que abordar en futuros trabajos la aceleración de los cálculos para la optimización de la configuración óptima de combustible en función de unos criterios.

RESULTADOS DE PATHS/PARCS

La **Figura 3** resume los resultados del PARCS, mostrando el punto en el que termina el ciclo, cuya duración exacta debe obtenerse por interpolación entre el último punto con ppm de boro añadido al refrigerante (39.8 ppm), y el primer punto sin ppm de boro añadido (0.1 ppm).

La **Figura 4** presenta la evolución de la concentración de boro (izquierda) y la relación entre el quemado y el tiempo keff (derecha) para uno de los 25000 casos simulados.

El caso número 17603, con una potencia pico de 2.5512 fue el de ciclo más largo con una duración de 662.95 días frente a los 661.89 días del caso más corto.

summary:

PT	RE	DAYs	Keff	Pxyz(1fa,kz)	Pxy	Pz	B(GWd/T)	Bmax	notch	ppm	Tf(K)	Tm(K)	d(g/cc)
58	28	634.8	1.000000	1.262(27, 6)	1.143	1.104	21.430	29.7	0	86.9	780.5	580.6	0.70936
59	29	634.8	1.000000	1.262(23, 6)	1.143	1.104	21.430	29.7	0	86.9	780.5	580.6	0.70936
60	29	649.6	1.000000	1.260(27, 6)	1.142	1.104	21.930	30.3	0	39.8	780.5	580.6	0.70936
61	30	649.6	1.000000	1.260(27, 6)	1.142	1.104	21.930	30.3	0	39.8	780.5	580.6	0.70936
62	30	664.4	0.999418	1.255(23, 6)	1.140	1.101	22.430	30.8	0	0.1	780.5	580.6	0.70938
63	31	664.4	0.999418	1.255(27, 6)	1.140	1.101	22.430	30.8	0	0.1	780.5	580.6	0.70938
64	31	679.2	0.995470	1.234(23, 6)	1.138	1.085	22.930	31.3	0	0.1	780.3	580.5	0.70960
65	32	679.2	0.995470	1.234(27, 6)	1.138	1.085	22.930	31.3	0	0.1	780.3	580.5	0.70960
66	32	694.0	0.991541	1.216(23, 6)	1.137	1.072	23.430	31.9	0	0.1	780.2	580.4	0.70978
67	33	694.0	0.991541	1.217(27, 6)	1.137	1.072	23.430	31.9	0	0.1	780.2	580.4	0.70978
68	33	708.8	0.987641	1.202(23, 6)	1.135	1.061	23.930	32.4	0	0.1	780.1	580.3	0.70992
69	34	708.8	0.987641	1.202(27, 6)	1.135	1.061	23.930	32.4	0	0.1	780.1	580.3	0.70992
70	34	723.6	0.983777	1.196(25,23)	1.133	1.053	24.430	33.0	0	0.1	780.0	580.3	0.71004
71	35	723.6	0.983776	1.196(25,23)	1.133	1.053	24.430	33.0	0	0.1	780.0	580.3	0.71004
72	35	738.4	0.979946	1.200(25,23)	1.131	1.057	24.930	33.6	0	0.1	779.9	580.2	0.71014

Figura 3. Días, k_{eff}, Potencia máxima (3D, radial, axial), Quemado (GWd/tHM), Concentración de boro (ppm).

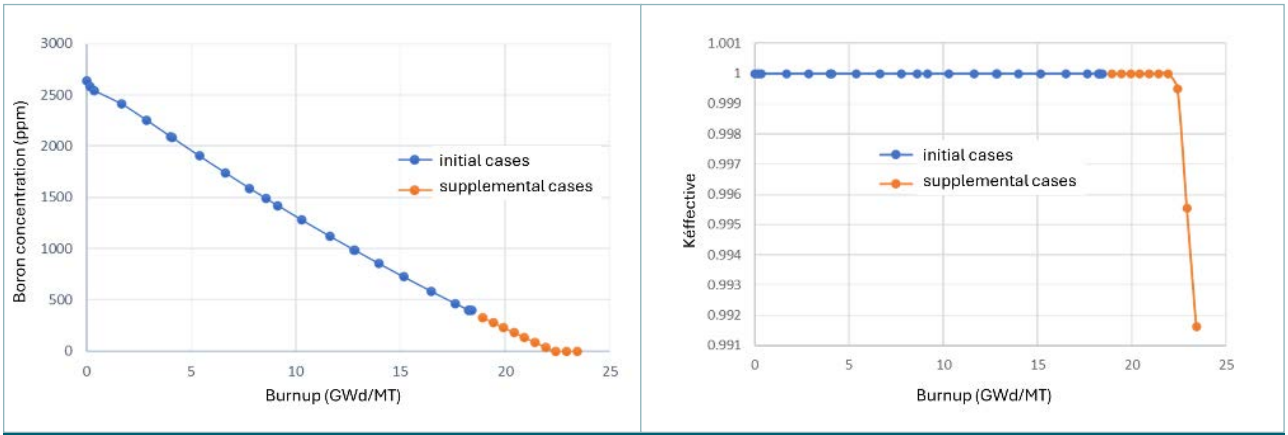

Figura 4. Concentración de boro (ppm) (izquierda) y k_{eff} (derecha) vs. quemado (GWd/tHM).

REDES NEURONALES

Se ha desarrollado un modelo de regresión que utiliza redes neuronales (RRNN) para predecir variables esenciales para el diseño del núcleo de reactores de agua a presión (PWR). Estas predicciones se basan en simulaciones derivadas de cálculos neutrónico-termohidráulicos acoplados en 3D. El objetivo principal de la regresión es pronosticar el valor de una variable numérica en función de los valores de una o más variables, que pueden ser numéricas o categóricas. En esencia, se pretende determinar la relación entre las entradas (x) y las salidas (y) a través de un modelo de aprendizaje profundo (DL).

En esta sección se describen las tres etapas del proceso: 1) preparación y preprocesamiento del conjunto de datos; 2) diseño, entrenamiento, ajuste y validación del modelo neuronal; y 3) evaluación del modelo o modelos.

Preparación y preprocesamiento de conjuntos de datos

Como se ha mencionado en el punto anterior, el conjunto de datos objetivo consta de 25000 casos. Las entradas (x) corresponden a las 39 posiciones ubicadas en el cuadrante inferior de la cuadrícula (patrón de carga, ver **Figura 1**) del reactor PWR. Cada posición está asociada con un valor categórico, que representa el nivel de quemado y el tipo de combustible, que se agrupan en 8 tipos o clases. Las salidas (y) que pretendemos predecir son los valores numéricos de la concentración inicial de boro del reactor, la duración exacta del ciclo y el pico de potencia máxima alcanzado en una ubicación específica (celda) en el modelo del núcleo.

Para procesar las entradas (x), cada una de las 39 posiciones, asociadas con un valor categórico que representa las 8 clases, se ha codificado como un vector de codificación *one-hot* de 8 dimensiones. Esto da como resultado 39 vectores, cada uno con 8 dimensiones, que representan las entradas. Es importante señalar que más adelante, analizaremos la utilización de una capa de incrustación dentro del modelo neuronal para esta transformación de datos.

Respecto a los parámetros de salida (y), se aplicó un escalamiento Min-Max a cada una de las 3 variables de forma independiente. El escalamiento Min-Max se implementa a través del *MinMaxScaler* del módulo *scikit-learn* de Python. Los datos escalados se encuentran dentro de un rango específico, en este caso, entre 0.02 y 0.98. Se calcula de la siguiente manera:

$$z = \frac{(x - x_{min})}{(x_{max} - x_{min})} \cdot (\nu_{max} - \nu_{min}) + \nu_{min} \quad (1)$$

donde x representa el valor original y x_{min} y x_{max} denotan los valores mínimo y máximo del conjunto de datos, respectivamente. Mientras tanto, ν_{min} y ν_{max} representan los límites del rango al que se desea escalar los datos, en este caso entre 0.02 y 0.98; se profundiza en la lógica detrás de esto más adelante.

Posteriormente, el conjunto de datos se dividió en tres subconjuntos: entrenamiento (80 %, 20000 casos), validación (10 %, 2500 casos) y prueba (10 %, 2500 casos).

Diseño, entrenamiento, ajuste y validación de modelos neuronales

Durante la fase experimental, realizamos una exploración exhaustiva de hiperparámetros y técnicas de entrenamiento para la red neuronal final, utilizando los subconjuntos de entrenamiento y validación. Esta sección describe los experimentos más significativos realizados. Todos los experimentos se realizaron usando Python, utilizando el marco de trabajo TensorFlow de código abierto (*tensorflow-gpu v.2.5.0*) [8] con soporte de GPU, junto con el marco de trabajo Keras de alto nivel. Estas bibliotecas se basan en otras bibliotecas auxiliares como *NumPy v.1.23.4*, *Matplotlib v.1.23.4*, etc. Además, se emplea *keras-tuner* para una búsqueda inicial de hiperparámetros para guiar la experimentación posterior.

En nuestra fase experimental, realizamos una serie de pruebas para abordar el desafío de la "no generalización" en las redes neuronales. Inicialmente, exploramos diferentes configuraciones de capas y unidades por capa para combatir la tendencia de las redes neuronales a memorizar durante el entrenamiento, lo que a menudo conduce a una baja capacidad de generalización. Para superar esto, investigamos la necesidad de la regularización L1 y L2 y el impacto de agregar ruido gaussiano durante el entrenamiento. Las técnicas como la regularización L1 y L2 se utilizan comúnmente para evitar

el sobreajuste en las redes neuronales al agregar un término de penalización a la función de pérdida, lo que reduce efectivamente la complejidad del modelo. La incorporación de regularización mejoró significativamente los resultados al mejorar la capacidad de generalización del modelo. La adición de ruido gaussiano durante el entrenamiento se utiliza a menudo para introducir aleatoriedad y aumentar la robustez en el modelo. Sin embargo, en nuestros experimentos, descubrimos que la introducción de ruido gaussiano empeoró los resultados.

Se analizó el efecto de la normalización por lotes en las tasas de error, tanto antes como después de las funciones de activación. La normalización por lotes mejora los resultados al estabilizar el proceso de entrenamiento y acelerar la convergencia. En particular, la normalización por lotes antes de las funciones de activación mostró las mejores mejoras. Además, exploramos el uso de la eliminación durante el entrenamiento, que implica eliminar aleatoriamente algunas neuronas para evitar la coadaptación de los detectores de características, mejorando así la robustez del modelo y evitando el sobreajuste. Sin embargo, la eliminación afectó negativamente a los resultados.

Para abordar los desafíos de optimización como el estancamiento de mínimos locales, evaluamos varios algoritmos con diferentes tasas de aprendizaje, incluidos Adam (estimación de momento adaptativo) y SGD (descenso de gradiente estocástico). Se realizaron pruebas adicionales con Adagrad (gradiente adaptativo) y Adadelta. Finalmente, se seleccionó Adam por sus características integrales. Determinar una tasa de aprendizaje adecuada fue crucial, y exploramos varias técnicas de recocido, como la descomposición lineal, los ajustes escalonados y el sobrecalentamiento. Finalmente, los mejores resultados se lograron con la descomposición sinusoidal, como se evidencia en la **Figura 5**.

Además, probamos diferentes funciones de activación para capas ocultas y de salida, como ReLU, SoftPlus, tanh, sigmoid y funciones lineales para combatir problemas como el desvanecimiento (*fading*) y la explosión del gradiente. La activación de SoftPlus en capas ocultas mostró mejoras con respecto a ReLU, especialmente cuando se combinó con una función de activación de salida sigmoidea. Como los mejores resultados se obtuvieron finalmente con la función de activación sigmoidea, las variables de salida (concentración de boro, duración del ciclo y potencia máxima) se normalizaron de forma independiente entre 0.02 y 0.98. Este rango específico se eligió para mitigar los problemas con la función sigmoidea cerca de los valores límite de 0 y 1.

Después de experimentar con todas las técnicas e hiperparámetros mencionados, se determinó que el mejor modelo neuronal era un Perceptrón Multicapa (MLP) con una capa de *Embedding* en la entrada de la red. La capa de *Embedding* sirve para mapear valores discretos a representaciones vectoriales continuas, lo que permite a la red comprender las relaciones semánticas entre valores categóricos. Agregar esta capa, que transforma los valores categóricos en vectores de 8 características, mejoró considerablemente los resultados. Anteriormente, era necesario dividir la tarea de regresión en tres modelos neuronales separados porque separarlos mejoraba los resultados. Con la adición de esta capa de *Embedding*, las diferencias son mínimas y, en algunas variables, incluso mejores cuando se usa un modelo combinado, lo que nos permite obtener los tres resultados con una sola inferencia. Inicialmente, cargamos la capa de *Embedding* con una matriz identidad de 8 dimensiones y congelamos esta capa durante el entrenamiento, asegurando que las incrustaciones de palabras permanecieran sin cambios. Sin embargo, se descubrió que permitir que estos vectores se entrenaran mejoraba ligeramente los resultados. A continuación de la capa de incrustación, hay una capa de aplanamiento que reúne los 39 vectores de 8 dimensiones y los aplana en un único vector. A partir de este punto, colocamos capas densas ocultas, todas con activación

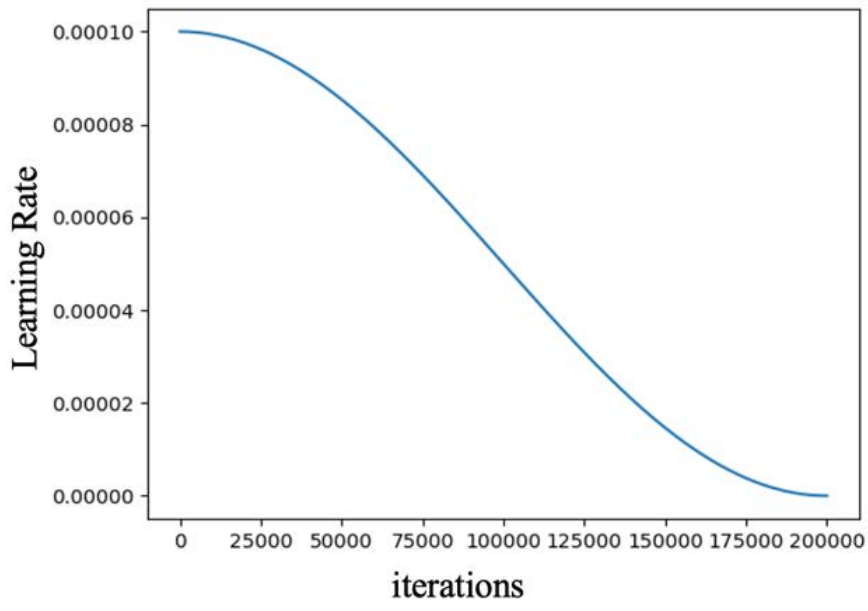

Figura 5. Decaimiento sinusoidal de la tasa de aprendizaje.

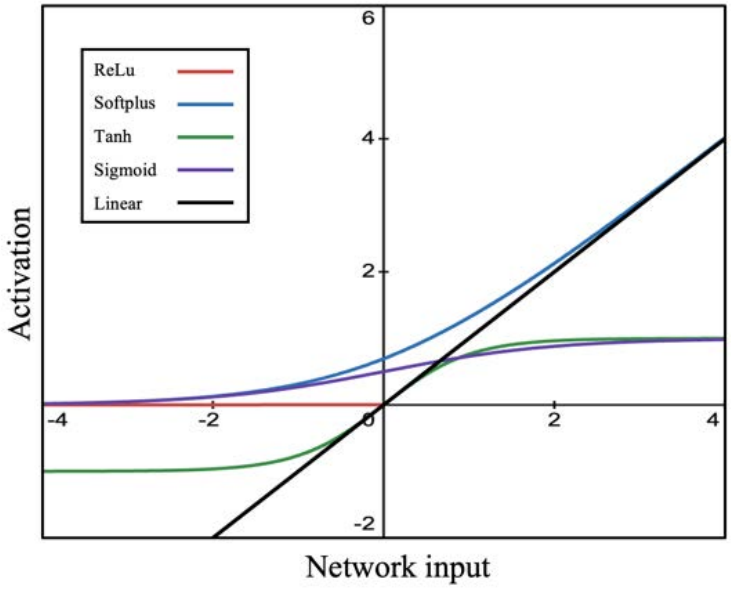

Figura 6. Funciones de activación.

softPlus, regularización L1 y L2 y normalización por lotes antes de la función de activación. La capa final es una capa densa de salida con 3 neuronas, una por cada variable a predecir, con activación sigmoidea.

Evaluación y resultados de redes neuronales

En esta etapa, el subconjunto de prueba se utilizó para evaluar la precisión del modelo de red neuronal como modelo de regresión. Para lograr este propósito, se emplearon varias métricas conocidas y ampliamente adoptadas, entre ellas:

- **Error cuadrático medio (RMSE):** mide la magnitud media de los errores entre los valores previstos y los reales. Es la raíz cuadrada de las diferencias cuadráticas medias entre estos valores.
- **Error cuadrático medio (EMM):** representa la diferencia cuadrática media entre los valores previstos y los reales. Da una idea de la varianza de los errores.
- **R cuadrado (R^2):** indica la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Proporciona una medida de qué tan bien coinciden las predicciones con los datos reales.
- **Error absoluto medio (MAE):** mide la magnitud media de los errores en un conjunto de predicciones, sin tener en cuenta su dirección. Es el promedio de las diferencias absolutas entre los valores previstos y los reales.
- **Desviación estándar del error (σ):** proporciona información sobre la dispersión de los errores con respecto al error medio. Indica la variabilidad de los errores de predicción.
- **Tasa de error porcentual (PER):** calcula el error porcentual relativo dividiendo el RMSE por la media de los valores reales y multiplicando por 100. Esta métrica expresa el error como un porcentaje del valor real medio.
- **Error normalizado de rango (RNE):** esta métrica, una variación del PER, divide el RMSE por el rango (la diferencia entre los valores máximo y mínimo) de los valores reales en lugar de por la media. Este enfoque se centra en las diferencias entre los valores, lo que lo hace más coherente en los casos en los que los valores reales tienen rangos amplios, como los ciclos en los que las diferencias se expresan en horas, pero la media de los ciclos es de alrededor de un año.

Utilizando estas métricas, evaluamos exhaustivamente el rendimiento de nuestro modelo de RRNN.

Los resultados de la red neuronal en la **Tabla I** demuestran un sólido funcionamiento de la misma en la duración del ciclo (horas), la concentración de boro (ppm) y la potencia local pico (valores normalizados). Los bajos valores de RMSE (0.457, 0.125 y 0.701e-03, respectivamente) indican errores de predicción promedio mínimos. Los valores de MSE confirman una baja varianza de error, en particular para la potencia local pico (4.919e-07). Los altos valores de R cuadrado (cercanos a 1.0) muestran un ajuste excelente del modelo, lo que explica casi toda la varianza en los valores reales.

Los bajos valores de MAE (0.377, 0.087 y 0.501e-03) reflejan magnitudes de error promedio mínimas. La desviación estándar de los errores también es baja, lo que indica una precisión de predicción constante. Las métricas PER (0.287e-02% y 0.473e-02%) revelan errores relativos muy bajos, y los valores de RNE, aunque más altos, aún indican una precisión razonable.

En general, el modelo logra una alta precisión y errores mínimos, prediciendo eficazmente variables clave con una sola inferencia. La incorporación de la capa de integración

Tabla I. Resultados de redes neuronales

	Duración del ciclo (horas)	Concentración de boro (ppm)	Potencia pico local (normalizada)
RMSE	0.457	0.125	0.701e-03
MSE	0.208	0.015	4.919 e-07
R^2	0.988	0.999	0.999
MAE	0.377	0.087	0.501e-03
Desviación estándar	0.258	0.088	0.490e-03
POR (%)	0.287e-02	0.473e-02	0.027
RNE (%)	1.977	0.397	0.274

ha mejorado notablemente los resultados, reduciendo la necesidad de modelos separados y mejorando la precisión general de la predicción.

CONCLUSIONES

Después de la experimentación exhaustiva en este trabajo, podemos concluir que el modelo óptimo para predecir los días del ciclo del reactor comprende 39 neuronas en la capa de entrada y 7 capas ocultas con 512-1024-2048-4096-2048-1024-512 neuronas. Este modelo utiliza la función de activación SoftPlus y la normalización por lotes antes de la función de activación, junto con las técnicas de regularización L1 y L2. La capa de salida consta de una neurona que emplea la función de activación sigmoidea. El tamaño de lote que produjo el mejor rendimiento fue 256 muestras.

El error promedio para la duración del ciclo obtenido con la Red Neuronal desarrollada es de solo 1.2077 horas (sobre una duración de ciclo de 662.366 días) con un σ de 0.8880.

Contamos con una red neuronal robusta que predice con precisión la duración del ciclo, la concentración inicial de boro y la potencia pico local para configuraciones de reactores no observadas durante el entrenamiento. El siguiente paso en trabajos futuros es utilizar esta información para obtener el diseño de patrón óptimo. Para identificar esta configuración óptima en el gemelo digital, necesitamos diseñar un sistema inteligente gobernado por una búsqueda multicriterio que combine una concentración inicial de boro más baja, maximice la duración del ciclo y mantenga la potencia pico dentro de límites seguros. Debe especificarse que se le da la máxima prioridad a la duración del ciclo, se probarán diferentes pesos dados a cada criterio de optimización.

El metamodelo obtenido a partir de estos resultados, así como las nuevas RRNN desarrolladas incluyendo más variables de diseño y licenciamiento, como DNBR, PCT, Entalpía, etc., serán utilizadas en estudios futuros para implementar un cálculo rápido del patrón de carga del núcleo basado en técnicas de optimización como Algoritmos Genéticos y más prometedoramente con *Simulated Annealing*.

REFERENCIAS

[1] T. Downar, A. Ward, V. Seker, N. Hudson, D. Barber, L. Larsen, B. Neykov, G. Roth, Y. Xu. PARCS v3.4.2. Universidad de Michigan, Comisión Reguladora Nuclear, Laboratorios de Sistemas de Información Inc., Universidad de Purdue (2022).

[2] A. Wysocki, Y. Xu, B. Collins, A. Manera, T. Downar, AM Ward, N. Hudson, G. Roth, M. Bradbury, L. Larsen, D. Barber. PATHS, PARCS Advanced Thermal Hydraulic Solver 1.08. Universidad de Michigan, Comisión Reguladora Nuclear, Laboratorios de Sistemas de Información Inc. (2022).

[3] E. Fernandes Faria, C. Pereira, C. Optimización del pa-

trón de carga de combustible nuclear utilizando una red neuronal. Annals of Nuclear Energy 30, 603-613 (2003).
[4] FCM Verhagen, HPM Gibcus, PH Wakker, D. Janin, M. Seidl. Desarrollos recientes del código PWR de Rosa y una aplicación especial de diseño de patrones de carga. Advances in Nuclear Management (ANFM 2015), Hilton Head Island, Carolina del Sur, EE. UU. (2015).
[5] L. Gilli y PH Wakker, BR Elder. Desarrollo de una herramienta de gestión de combustible en el núcleo para reactores de agua en ebullición. Actas de ICAPP 2017, Fukui y Kioto, Japón (2017).
[6] RA Saleem, MI Radaideh, T. Kozlowski . Aplicación de redes neuronales profundas para la neutrónica de núcleos de reactores de agua de gran tamaño y alta dimensión. Ingeniería nuclear y tecnología, 52. (2020).
[7] F. Chollet y otros. Keras . (2015). Disponible en: https://github.com/fchollet/keras.
[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, GS Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp , G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: aprendizaje automático a gran escala en sistemas heterogéneos, 2015. Software disponible en tensorflow.org. ■

AGRADECIMIENTOS

Los autores desean agradecer al Consejo de Seguridad Nuclear (CSN) su apoyo financiero a través del proyecto de I+D+i “CLPD-IA. Diseño Optimizado del Patrón de Carga del Núcleo de Reactores LWR Asistido por Inteligencia Artificial” (SUB-36/2021), 2022-2024. Pablo Pallarés también contó con una beca predoctoral de la Generalitat Valenciana (CIACIF/2021/217).