XX Simposio CEA de Control Inteligente
25-27 de junio de 2025, Huelva

Optimización computacional y energética de la metodología de entrenamiento para Redes Neuronales de Impulsos (SNNs) para predicción un paso adelante.

Lucas, S.[1,*], Portillo, E.[1]

Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, Universidad del País Vasco (UPV/EHU), C/ Plaza Ingeniero Torres Quevedo nº 1, 48013, Bilbao, España.

To cite this article: Lucas, S., Portillo, E., 2025. Computational and energy optimization of the training methodology for Spiking Neural Networks (SNNs) one step-ahead forecasting. XX Simposio CEA de Control Inteligente, Huelva (Spain), 2025.

Resumen

Las Redes Neuronales de Impulsos (*Spiking Neural Networks*, SNNs) son modelos computacionales reconocidos por ser capaces de procesar información espacio-temporal con un consumo energético extremadamente bajo. Para aprovechar sus ventajas energéticas y computacionales, las SNNs se deben utilizar junto con algoritmos de codificación-decodificación lo suficientemente eficientes. Así, en este trabajo se presentan dos nuevas propuestas de la metodología de entrenamiento de SNNs para problemas de predicción un paso adelante (*forecasting*), mediante la aplicación de dos versiones optimizadas del algoritmo original de codificación-decodificación basado en la modulación por ancho de pulso (*Pulse Width Modulation*, PWM). Estas dos nuevas propuestas se han validado en series temporales correspondientes a una señal senoidal y 4 bases de datos de acceso público. Los resultados muestran que las nuevas propuestas consiguen eliminar casi por completo los costes computacionales y energéticos de la codificación y decodificación, manteniendo la precisión de la metodología original.

Palabras clave: Diseño de metodologías, Validación de modelos, Redes Neuronales, Modelado de series temporales, Diseño de experimentos.

Computational and energy optimization of the training methodology for Spiking Neural Networks (SNNs) one step-ahead forecasting

Abstract

Spiking Neural Networks (SNNs) are computational models recognized for their capability to process spatiotemporal information with ultra-low energy consumption. To boost their efficiency advantages, it is essential to employ efficient encoding-decoding algorithms alongside SNNs. In this regard, this paper introduces two novel approaches of the training methodology for SNNs in one-step-ahead forecasting tasks, where the original Pulse Width Modulation (PWM) based encoding-decoding algorithm is replaced by its two optimized variants. The proposed methodologies are validated employing five different time series: a sine-wave and four publicly available benchmarks datasets. The results show that the new proposals succeed in almost completely reducing the computational and energy costs of encoding and decoding, while still keeping the accuracy of the original methodology.

Keywords: Design methodologies, Model validation, Neural Networks, Time series modelling, Experiment design.

1. Introducción

La irrupción de aplicaciones como *ChatGPT* o *DeepSeek* ha incrementado la preocupación de la comunidad científica por el consumo energético de las herramientas de Inteligencia Artificial (IA) (de Vries, 2023). Así, el uso de hardware y algoritmos de IA más eficientes resulta cada vez más necesario en el medio-largo plazo.

Una de las técnicas de IA que mayor popularidad ha ga-

*Autor para correspondencia: sergio.lucas@ehu.eus

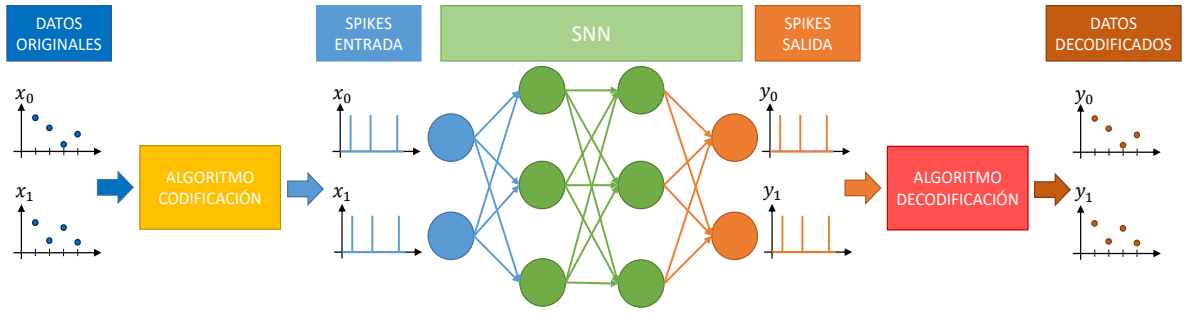

Figura 1: Tratamiento de los datos al usar una SNN.

nado en los últimos años debido a su elevada eficiencia son las Redes Neuronales de Impulsos (*Spiking Neural Networks*, SNNs), también conocidas como la tercera generación de redes neuronales. Esta nueva generación de redes neuronales ha demostrado (Fang et al., 2023) ser más eficiente energéticamente y computacionalmente que sus predecesoras, las Redes Neuronales Artificiales tradicionales (*Artificial Neural Networks*, ANNs). A diferencia de las ANNs, las SNNs codifican la información en series de impulsos o *spikes*, tratando de imitar con mayor fidelidad el comportamiento de las neuronas biológicas. El hecho de emplear *spikes* (secuencias de 0 y 1) como unidades de proceso, implica que: (i) la información en las SNNs no se encuentra en la forma de la acción potencial, sino en cuándo se dan los *spikes* (Gerstner and Kistler, 2002); (ii) las neuronas que en un determinado instante de tiempo no reciban un *spike*, es decir, que su valor de entrada sea nulo, se pueden considerar *inactivas*, reduciendo así el coste energético del algoritmo; y (iii) en las implementaciones hardware, el coste de las multiplicaciones de las entradas por los pesos se puede sustituir directamente por sumas, reduciendo significativamente el coste computacional (Suetake et al., 2023). Además, por todas estas razones, las SNNs presentan ventajas significativas para ser implementadas en hardware neuromórfico de consumo ultrabajo (Bu et al., 2022).

Para poder emplear las SNNs en aplicaciones reales es necesario el uso de algoritmos de codificación y decodificación. La mayoría de las unidades/magnitudes de los procesos reales son de tipo real, por tanto, antes de introducir los datos en la SNN se requiere de una etapa previa donde se codifiquen los datos de tipo real en *spikes* (ver Figura 1). Del mismo modo, si la SNN se está empleando para una tarea de monitorización o control es necesario introducir una etapa posterior para decodificar los *spikes* en las unidades/magnitudes reales.

A pesar de poseer características intrínsecas para procesar información temporal, la inmensa mayoría de los trabajos con SNNs se han aplicado a problemas de clasificación. De hecho, apenas existen trabajos de SNNs en problemas de regresión o predicción (*forecasting*) (Lucas and Portillo, 2024). Esto se debe a que históricamente las SNNs han presentado dos limitaciones: (1) carencia de algoritmos capaces de codificar y decodificar con precisión (nótese que en los problemas de regresión o *forecasting* es necesario decodificar, pero en los de clasificación no); y (2) imposibilidad de implementar directamente en las SNNs estrategias consolidadas de entrenamiento supervisado como la retropropagación, debido a los problemas de diferenciabilidad que presentan las ecuaciones diferenciales que procesan la información en las SNNs en el momento de la emisión de los *spikes* (Lucas and Portillo, 2024).

Así, en (Lucas and Portillo, 2024) se presentó la primera metodología de entrenamiento supervisado de SNNs en problemas de predicción un paso adelante (*forecasting one-step ahead*) que es independiente de las características del campo de aplicación. Para solventar las limitaciones mencionadas, esta metodología aplica: (1) un novedoso algoritmo de codificación y decodificación temporal basada en la modulación por ancho de pulso (*Pulse Width Modulation*, PWM) (Arriandiaga et al., 2020), y (2) un método de gradiente subrogado (Neftci et al., 2019) que permite aplicar la retropropagación en las SNNs.

El algoritmo de codificación-decodificación basado en PWM (Arriandiaga et al., 2020) fue originalmente diseñado para ofrecer ventajas significativas frente a sus algoritmos predecesores en cuanto a la precisión a la hora de codificar y decodificar series temporales, si bien no se tuvo en cuenta su eficiencia. En este sentido, en (Lucas et al., 2025) se presentan dos propuestas (V_{opt} y V_{ext}) para mejorar tanto el coste computacional como energético del algoritmo original (V_{or}).

Asimismo, la metodología de entrenamiento para SNNs en problemas de *forecasting* (Lucas and Portillo, 2024) se diseñó empleando el algoritmo original V_{or} para codificar y decodificar. No emplear algoritmos de codificación y decodificación lo suficientemente eficientes puede contrarrestar la eficiencia que ofrecen las SNNs. Por ello, en este trabajo se presentan dos propuestas optimizadas de la metodología de entrenamiento para SNNs en problemas de *forecasting*, donde el algoritmo V_{or} se sustituye por los algoritmos optimizados V_{opt} y V_{ext}. Además, se comparan los resultados obtenidos con los de la metodología original.

El resto del artículo está estructurado de la siguiente forma: en la Sección 2 se resume la metodología de entrenamiento para SNNs. En la Sección 3 se detallan las principales modificaciones en los algoritmos V_{opt} y V_{ext} para que puedan ser incluidos en la metodología. En la Sección 4 se validan las nuevas propuestas y en la Sección 5 se exponen las conclusiones.

2. Metodología original de entrenamiento de SNNs para problemas de forecasting

La metodología original se encuentra detallada en (Lucas and Portillo, 2024). Los aspectos más importantes de dicha metodología son: (a) el algoritmo de codificación-decodificación, (b) el modelo neuronal, (c) la estrategia de entrenamiento, y (d) la estructura de la SNN.

2.1. *Algoritmo de codificación-decodificación*

En la metodología original de entrenamiento de SNN se usa el algoritmo original de codificación-decodificación basado en PWM (V_{or}) (Arriandiaga et al., 2020). Este algoritmo se basa en los principios del PWM. Durante la fase de codificación (ver Figura 2) emite *spikes* cuando se produce una intersección entre la señal original (serie temporal) y la señal portadora (diente de sierra). Por otro lado, durante la fase de decodificación calcula los valores decodificados mediante la intersección entre la señal portadora y los *spikes* (salida de la SNN). Así, la señal portadora sirve tanto en la codificación como en la decodificación como base temporal. Para poder definir correctamente la señal portadora se utilizan dos hiperparámetros: (i) Número de dientes de sierra (nc), directamente relacionado con la frecuencia de la señal; y (ii) Número de puntos por diente de sierra (npc), el cual define la resolución de la codificación y decodificación.

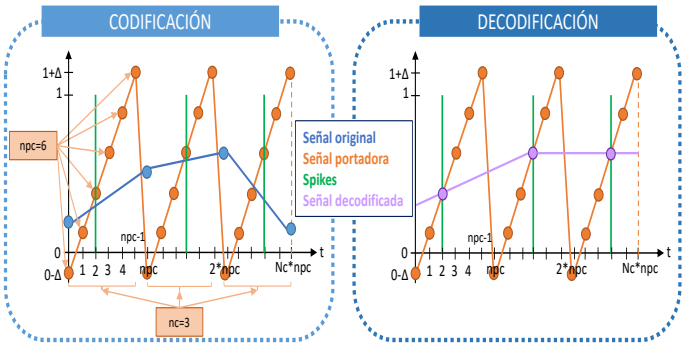

Figura 2: Procesos de codificación y decodificación con V_{or} (donde Δ normalmente es igual al 10 % de la amplitud de la señal)

.

2.2. *Modelo neuronal*

Los modelos neuronales de las SNNs se definen mediante ecuaciones diferenciales. Existe una gran variedad de modelos neuronales de SNNs, los cuales varían según su eficiencia y verosimilitud con las neuronas biológicas.

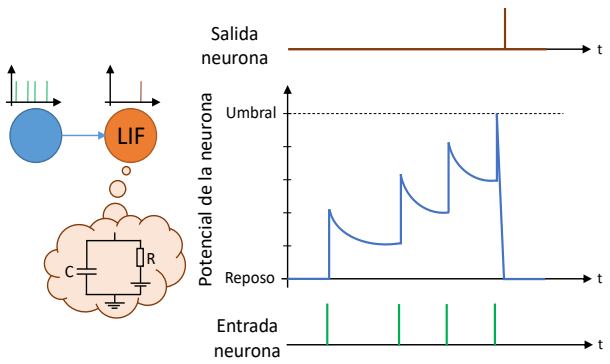

Figura 3: Funcionamiento modelo LIF.

El modelo neuronal utilizado en la metodología de entrenamiento de SNNs es el modelo *Leaky Integrate-and-fire* (LIF) (Gerstner et al., 2014), ya que es el modelo más utilizado debido a su simplicidad y bajo coste computacional (Yamazaki et al., 2022). El modelo LIF se caracteriza por tener un término "*leaky*"que hace que su descarga se realice en forma de exponencial (ver Figura 3), de manera que su comportamiento es el de un sistema de primer orden ante entrada impulso. De hecho, la representación típica del modelo LIF es un circuito RC.

2.3. *Estrategia de entrenamiento*

Las ecuaciones diferenciales de las SNNs presentan problemas de diferenciabilidad en el momento de la emisión del *spike*. Por esta razón, la estrategia supervisada de entrenamiento por excelencia en las ANNs, la retropropagación, no se puede aplicar directamente en las SNNs.

Para solventar esta limitación la metodología de entrenamiento de SNNs presentada en (Lucas and Portillo, 2024) se basa en el método del gradiente subrogado (Neftci et al., 2019). Este método aplica durante la propagación de las muestras las ecuaciones diferenciales propias de las SNNs. Sin embargo, durante la retropagación cambia estas ecuaciones por una función subrogada continua en el tiempo y, por tanto, diferenciable, permitiendo la aplicación de la retropropagación. Más concretamente, la función subrogada de la metodología es la función sigmoidal.

En cuanto a la función de pérdida, la metodología emplea el error cuadrático calculado de la siguiente forma:

$$E = (\delta_{target} - \delta_{salida\ SNN})^2 \tag{1}$$

donde δ_{target} es el instante de tiempo donde se da el *spike* en el target y $\delta_{salida\ SNN}$ es el instante de tiempo donde se da el primer *spike* en la salida de la SNN. Para calcular estos instantes se usa la señal portadora (ver Figura 4), la cual define una base temporal mediante el hiperparámetro npc (en cada diente de sierrra hay npc posibles instantes de emisión de spikes, por tanto, $\delta \in [0, ..., npc - 1]$). Nótese que si no hay *spike* a la salida de la SNN, el error se calcula desde el inicio del diente de sierra. Por tanto, es necesario asegurar que en esa posición no se pueda emitir ningún *spike* a la salida de la SNN.

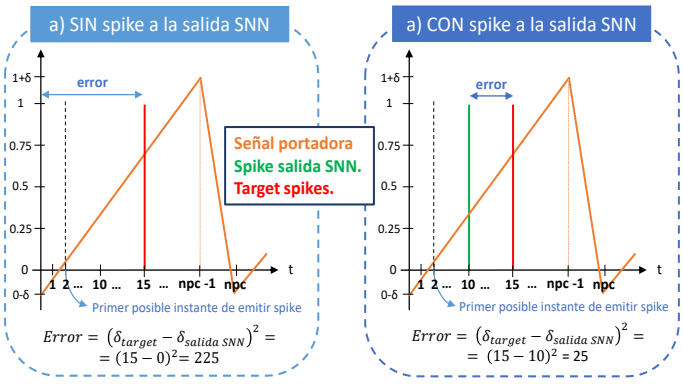

Figura 4: Cálculo función error.

2.4. *Estructura SNN*

La metodología de entrenamiento está diseñada para la SNN más simple posible, esto es, sin capa oculta alguna. Además, al estar específicamente diseñada para un problema de *forecasting*, el número de neuronas en la capa de entrada es igual al número de instantes anteriores (Np) necesarios para la predicción multiplicado por el valor de npc seleccionado. El número de neuronas de la capa de salida es igual al valor de npc, ya que el horizonte de predición es 1 (*one-step ahead*).

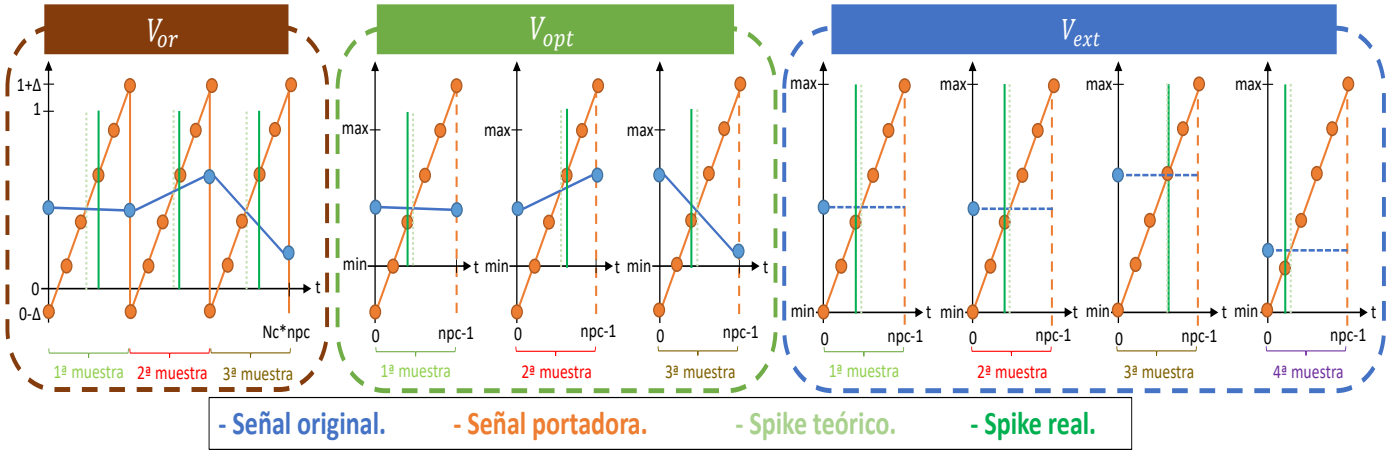

Figura 5: Diferencias en el proceso de codificación entre V_{or}, V_{opt} y V_{ext}.

3. Nuevas propuestas de entrenamiento de SNNs para problemas de forecasting

En esta sección se presentan dos nuevas propuestas para la metodología de entrenamiento de SNNs en problemas de *forecasting*. Estas propuestas se basan en la sustitución del algoritmo de codificación-decodificación V_{or} por dos algoritmos con mayor grado de optimización y eficiencia, V_{opt} y V_{ext}.

3.1. Propuesta con V_{opt}

Los procesos de codificación y decodificación con V_{opt} se encuentran detallados en (Lucas et al., 2025). V_{opt} es la versión optimizada del algoritmo V_{or} y cuenta con dos principales mejoras. En cuanto a la primera mejora, V_{or} interpola señales completas (señales formadas por $nc \cdot npc$ puntos) y las analiza punto a punto, mientras que V_{opt} codifica cada instante por separado mediante el cálculo de la intersección entre dos rectas genéricas. Esta simplificación en las operaciones conlleva importantes ventajas computacionales y energéticas. En cuanto a la segunda mejora, V_{or} emite el *spike* en el instante npc posterior a la intersección entre la señal original y la portadora , tal y como se puede comprobar en la Figura 5. Sin embargo, V_{opt} lo emite en el instante más próximo a la intersección, lo cual mejora la precisión en la codificación.

En este sentido, como V_{opt} optimiza el algoritmo V_{or} sin modificar los principios de éste, la metodología de entrenamiento de SNNs no precisa de ningún cambio para poder incluir este nuevo algoritmo.

3.2. Propuesta con V_{ext}

Los procesos de codificación y decodificación con V_{ext} se encuentran detallados en (Lucas et al., 2025). A diferencia de V_{opt}, V_{ext} sí presenta variaciones de diseño frente a V_{or}, ya que su finalidad es ampliar el campo de aplicación del algoritmo de codificación-decodificación basado en PWM a otras áreas donde no existan relaciones cronológicas. Con dicho fin, V_{ext} no emplea dos puntos consecutivos de la serie temporal para codificar, si no que codifica un *spike* por cada punto, tal y como se puede ver en la Figura 5.

Además, V_{ext} permite definir el diente de sierra de la señal portadora aprovechando al máximo la amplitud de la serie temporal original, lo cual permite que la resolución del algoritmo sea máxima. Sin embargo, tal y como se ha explicado anteriormente, la función de error de la metodología de entrenamiento (ver Ecuación 1) requiere que en la primera posición del diente de sierra no se pueda emitir ningún *spike*. Esta condición necesaria no se cumple con V_{ext} y, por tanto, se debe introducir una modificación adicional en el algoritmo para poder implementarlo en la metodología de entrenamiento de SNNs. Esta modificación se visualiza en la Figura 6 y trata de dejar un único valor de npc libre al inicio del diente de sierra, tratando de perder así la mínima resolución posible.

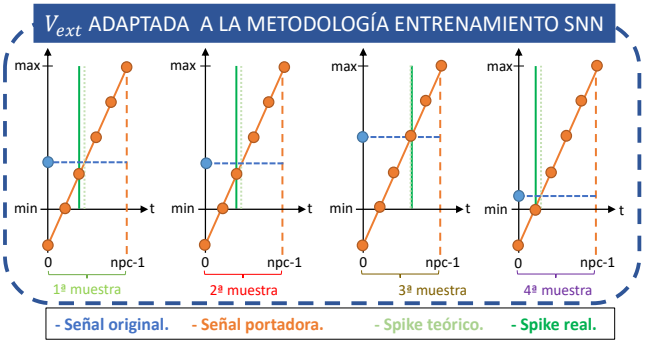

Figura 6: Algoritmo V_{ext} adaptado a la metodología.

4. Validación

En esta sección se presenta la validación de las dos nuevas propuestas para la metodología de entrenamiento de SNNs en problemas de *forecasting one-step ahead*. Las series temporales utilizadas en esta validación son las mismas que las consideradas en (Lucas and Portillo, 2024): (1) una señal senoidal periódica de frecuencia $f_o = 100Hz$, amplitud $A = 3$, frecuencia de muestreo $f_s = 60 \cdot f_o$ y número de ciclos 20; (2) la serie temporal Mackey-Glass Time Series Dataset (MGTSD); (3) la serie temporal formada por el consumo energético de la zona 3 de Tetouan entre las 00:00 del 01/01/2017 y las 23:50 del 31/01/2017; (4) la serie temporal formada por la concentración de PM_{10} entre las 10:00 del 21/02/2017 y las 20:00 del 03/06/2017 en el área metropolitana de Londres y medido por la estación de Blomsburry; y (5) la serie temporal ARCTIC formada por el rango de muestras [11600, 14750] de voz del fiche-

ro "*arctic_a0001.wav*" en la carpeta "*US English bdl (made)*". Todas las referencias necesarias para acceder a estas series temporales se encuentran en (Lucas and Portillo, 2024).

Además, con el objetivo de poder comparar los resultados en igualdad de condiciones, los hiperparámetros a modificar durante el entrenamiento son los mismos que los de la metodología original. En la Tabla 1, se muestra un resumen del rango de valores que pueden adquirir los hiperparámetros. Nótese que la validación consiste en entrenar una SNN para cada una de las posibles combinaciones de estos hiperparámetros y con cada una de las series temporales anteriormente descritas. En la Tabla 1 también se detalla el porcentaje de muestras de entrenamiento, validación y test para cada serie temporal.

Tabla 1: Hiperparámetros utilizados durante la validación

	PWM		Forecasting		División
	nc	npc	Np	H	subconjuntos
V_{opt}	N-1	32			Entrenamiento = 70 %
		64	[1...10]	1	Validación = 20 %
V_{ext}	N	128			Test = 10 %

Las nuevas propuestas se validan comparando sus resultados con los de la metodología original desde dos puntos de vistas diferentes: (i) la eficiencia y (ii) la precisión. Para analizar la eficiencia, cada una de las SNNs entrenadas se ejecuta 50 veces y se calcula la media del coste computacional (CC) y del coste energético (CE). Para ello, se emplea la librería *PyJoules* de Python en un ordenador PC Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz Linux 6.2.0-32-generic. En cuanto a la precisión, se emplea el Error Cuadrático Medio Normalizado (*Normalized Mean Square Error*, NMSE), el cual permite comparar señales en diferentes escalas y se define mediante la siguiente ecuación:

$$NMSE = \frac{\frac{1}{nc*npc}\sum_{i=1}^{nc*npc}(y_i - \hat{y}_i)^2}{var(y)} \quad (2)$$

donde y_i es el valor de cada una de las muestras de la serie temporal original, $\hat{y}_i$ es el valor de cada una de las muestras de la señal decodificada y $var(y)$ es la varianza de la serie temporal original.

4.1. Resultados validación

En la Tabla 2 se muestra para cada serie temporal y para cada metodología: (1) la combinación de hiperparámetros con las que se obtienen los mejores resultados en cuanto a precisión (menor NMSE), y (2) las métricas obtenidas con dicha configuración. Además, se detalla el grado de aleatoriedad de cada serie temporal mediante su valor de entropía de Shannon (H).

En cuanto a la precisión de las nuevas propuestas, se puede observar que con series temporales menos aleatorias (senoidal y MGTSD) se consiguen mejores resultados con las propuestas V_{opt} y V_{ext}. Sin embargo, cuando se aumenta el grado de aleatoriedad de las señales (Tetouan, Air pollution y ARCTIC), V_{or} consigue errores más bajos. Esto implica que emplear *targets* más precisos durante el entrenamiento (casos de V_{opt} y V_{ext}), no siempre conlleva mejoras en los procesos de entrenamiento de SNNs. Una posible causa ante este fenómeno puede ser que los algoritmos V_{opt} y V_{ext} introduzcan más ruido en el entrenamiento, produciendo un sobreajuste en la SNN. No obstante, las diferencias en cuanto a precisión de las metodologías son poco significativas, tal y como se ve en la Figura 7. En esta figura se visualizan las señales decodificadas con cada una de las metodologías y empleando la serie temporal Tetouan con 1 instante anterior y un npc de 64. Nótese que para visualizar en una misma gráfica todas las señales decodificadas, éstas deben tener el mismo valor de npc (mismo número de puntos).

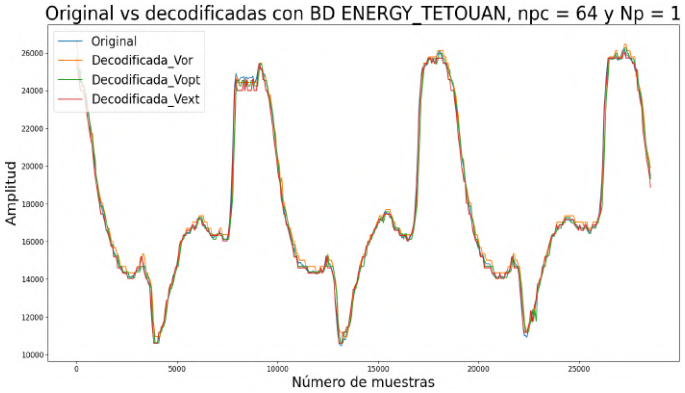

Figura 7: Resultados para la base de datos Tetouan con Np=1 y npc=64.

Para este mismo caso de estudio, la Figura 8 muestra la distribución de los costes computacionales (CC) y energéticos (CE) para cada una de las metodologías y diferenciando los diferentes procesos que ocurren en ellas (codificación, uso de la SNN y decodificación). Se puede apreciar que con el uso de los algoritmos V_{opt} y V_{ext} prácticamente se eliminan los costes computacionales y energéticos de los procesos de codificación y decodificación. Este hecho se corrobora también para el resto de casos de estudio, tal y como se puede ver en la Tabla 2.

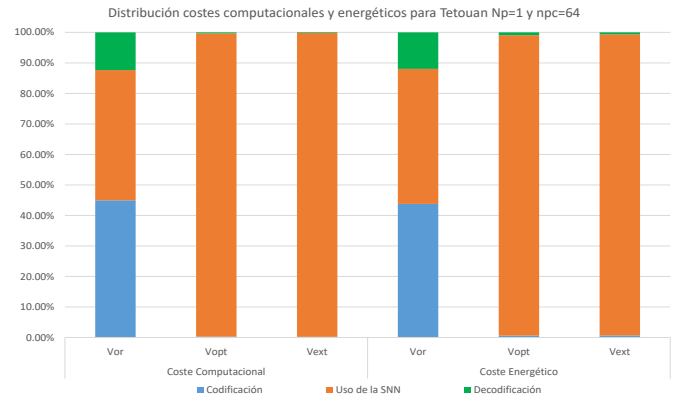

Figura 8: Comparación de la eficiencia de las metodologías para la base de datos Tetouan con Np=1 y npc=64.

5. Conclusiones

En este trabajo se presentan dos nuevas propuestas para la metodología de entrenamiento de SNN en problemas de *forecasting one-step ahead* (Lucas and Portillo, 2024). Ambas propuestas consisten en la sustitución del algoritmo original de codificación-decodificación basado en PWM por sus dos nuevas versiones optimizadas.

Tabla 2: Mejores resultados obtenidos con cada metodología

		Hiperparámetros		Métricas						
				NMSE	Codificación		Propagación		Decodificación	
		npc	Np		CC (ms)	CE (J)	CC (ms)	CE (J)	CC (ms)	CE (J)
SENOIDAL (H = 6.59)	V_{or}	128	3	0,00020	84,21	3,08	56,54	2,38	21,52	0,750
	V_{opt}	128	3	0,00003	0,404	0,030	56,58	2,34	0,257	0,039
	V_{ext}	128	3	0,00004	0,336	0,028	59,18	2,58	0,189	0,036
MGTSD (H = 7.09)	V_{or}	64	1	0,032	39,58	1,29	39,94	1,57	11,71	0,374
	V_{opt}	128	1	0,028	0,217	0,031	57,91	2,36	0,235	0,042
	V_{ext}	128	1	0,032	0,169	0,029	58,14	2,52	0,163	0,031
TETOUAN CONSUMPTION (H = 7.29)	V_{or}	128	1	0,009	302,01	10,04	201,46	8,55	84,58	2,74
	V_{opt}	64	1	0,010	0,417	0,030	141,23	4,70	0,470	0,044
	V_{ext}	32	1	0,011	0,282	0,028	112,92	3,81	0,251	0,040
AIR POLLUTION (H = 7.69)	V_{or}	64	1	0,183	80,23	2,68	75,60	2,62	22,16	0,727
	V_{opt}	32	1	0,221	0,294	0,032	59,52	2,14	0,258	0,030
	V_{ext}	64	1	0,274	0,208	0,028	75,73	2,66	0,188	0,034
ARCTIC (H = 7.82)	V_{or}	64	1	0,017	1107,8	36,43	97,88	4,12	31,50	0,988
	V_{opt}	64	1	0,027	1,454	0,047	99,05	3,20	0,391	0,031
	Vext	32	1	0,038	1,180	0,033	76,81	2,58	0,274	0,031

Los resultados reflejan que las nuevas propuestas: (1) mantienen o mejoran los resultados en cuanto a precisión de la metodología original, y (2) reducen casi por completo los costes computacionales y energéticos de los procesos de codificación y decodificación, siendo los costes principales los asociados a la SNN.

Como trabajo futuro, se está tratando de implementar las metodologías de entrenamiento, tanto con V_{opt} como con V_{ext}, directamente en hardware embebido, más concretamente en una FPGA. El objetivo es lograr que la metodología de entrenamiento funcione en hardware dedidacado, así como analizar la reducción del consumo energético.

Agradecimientos

Este trabajo ha sido financiado por el Departamento de Educación del Gobierno Vasco (ref. IT1726-22), el FEDER/Ministerio de Ciencia e Innovación - Agencia Estatal de Investigación/Proyecto (ref. PID2020-112667RB-I00) y por la Ayuda RED2022-134588-T financiada por MICIU/AEI /10.13039/501100011033.

Referencias

Arriandiaga, A., Portillo, E., Espinosa-Ramos, J. I., Kasabov, N. K., oct 2020. Pulsewidth Modulation-Based Algorithm for Spike Phase Encoding and Decoding of Time-Dependent Analog Data. IEEE Transactions on Neural Networks and Learning Systems 31 (10), 3920–3931.
DOI: 10.1109/TNNLS.2019.2947380

Bu, T., Ding, J., Yu, Z., Huang, T., jun 2022. Optimized Potential Initialization for Low-Latency Spiking Neural Networks. Proceedings of the AAAI Conference on Artificial Intelligence 36 (1), 11–20.
DOI: 10.1609/AAAI.V36I1.19874

de Vries, A., oct 2023. The growing energy footprint of artificial intelligence. Joule 7 (10), 2191–2194.
DOI: 10.1016/J.JOULE.2023.09.004

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., Huang, L., Zhou, H., Li, G., Tian, Y., oct 2023. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances 9 (40).
DOI: 10.1126/SCIADV.ADI1480

Gerstner, W., Kistler, W. M., aug 2002. Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press.
DOI: 10.1017/CBO9780511815706

Gerstner, W., Kistler, W. M., Naud, R., Paninski, L., jan 2014. Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press.
DOI: 10.1017/CBO9781107447615

Lucas, S., Portillo, E., may 2024. Methodology based on spiking neural networks for univariate time-series forecasting. Neural Networks 173, 106171.
DOI: 10.1016/J.NEUNET.2024.106171

Lucas, S., Portillo, E., Cabanes, I., 2025. Optimización y extensión del algoritmo de codificación-decodificación basado en PWM para Redes Neuronales de Impulsos. Revista Iberoamericana de Automática e Informática industrial 22 (1), 21–32.
DOI: 10.4995/RIAI.2024.20836

Neftci, E. O., Mostafa, H., Zenke, F., nov 2019. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine 36 (6), 51–63.
DOI: 10.1109/MSP.2019.2931595

Suetake, K., ichi Ikegawa, S., Saiin, R., Sawada, Y., feb 2023. S3NN: Time step reduction of spiking surrogate gradients for training energy efficient single-step spiking neural networks. Neural Networks 159, 208–219.
DOI: 10.1016/J.NEUNET.2022.12.008

Yamazaki, K., Vo-Ho, V. K., Bulsara, D., Le, N., jun 2022. Spiking Neural Networks and Their Applications: A Review. Brain Sciences 12 (7), 863.
DOI: 10.3390/BRAINSCI12070863