ENTRENAMIENTO SUPERVISADO DE REDES NEURONALES DE IMPULSOS

Sergio Lucas, Eva Portillo, Asier Zubizarreta, Itziar Cabanes
Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, Universidad del País Vasco (UPV/EHU)
slucas004@ikasle.ehu.eus

Resumen

En este trabajo se explora una nueva estrategia de entrenamiento supervisado con Redes Neuronales de Impulsos (Spiking Neural Network, SNN) para forecasting en series temporales. En la actualidad, la inmensa mayoría de los trabajos en SNN se centran principalmente en problemas de clasificación, muy especialmente de imágenes. En este sentido, el trabajo aquí presentado es uno de los primeros trabajos en aplicar SNN para forecasting de series temporales, siendo los resultados muy prometedores. Para validar la metodología se han empleado dos bases de datos: información bursátil de IBM, y señales EEG. Entre los resultados, se demuestra que el rendimiento de la SNN depende, como cabía esperar, de la dinámica de la señal o serie temporal a predecir.

Palabras clave: Redes Neuronales de Impulsos, PWM, series temporales, regresión, forecasting.

1. INTRODUCCIÓN

La pandemia del COVID-19 ha traído consigo la aparición de nuevas necesidades a nivel empresarial y cotidiano. Uno de los principales cambios que se han producido ha sido el aumento de la digitalización de los procesos, lo cual ha provocado un crecimiento exponencial del uso de la Inteligencia Artificial (IA).

Dentro de las tecnologías asociadas a la IA, destacan especialmente las Redes Neuronales Artificiales (*Artificial Neural Networks*, ANN). Como es bien sabido, este paradigma de aprendizaje y procesamiento automático está especialmente dirigido al aprendizaje basado en la experiencia, y por ello es habitual su aplicación en aquellos problemas en los que se desconocen las relaciones explícitas. Sin embargo, a pesar de haber sido aplicadas con éxito en múltiples campos, las neuronas artificiales utilizadas distan mucho de las reales, de manera que el coste energético y de datos que requieren es muy superior al de las redes neuronales biológicas.

En este sentido, las Redes Neuronales de Impulsos (*Spiking Neural Networks*, SNN), también conocidas como tercera generación de redes neuronales artificiales, intentan imitar con mayor fidelidad a las unidades biológicas. La principal diferencia frente a ANN es que en este tipo de redes la información no se codifica en valores analógicos, sino en series de impulsos o *spikes*, tratando de imitar al sistema nervioso. Esto significa que la información va contenida en cuándo suceden los spikes y no en su forma o valor.

Debido a que su codificación se basa en la distribución temporal de los impulsos, las SNN poseen características intrínsecas para manipular información temporal. A pesar de esta importante cualidad, la inmensa mayoría de los trabajos han centrado sus esfuerzos en aplicar las SNN en problemas de clasificación, como pueden ser el reconocimiento de imágenes [3], la supervisión de electrocardiogramas [14], el reconocimiento de patrones [7] o de eventos [15], y la detección de objetos [8]. De hecho, apenas existen trabajos dedicados a modelar evoluciones temporales mediante SNN en problemas de regresión [13] [12]. Es más, en la mayoría de estos trabajos se aplican estrategias de clasificación en lugar de regresión, dando lugar a errores de codificación de la información muy significativos. Además, la mayoría de los trabajos presentes en la literatura se centran exclusivamente en imágenes, siendo especialmente empleadas las bases de datos de *benchmark* MNIST y CIFAR-10 [3] [7] [8] [12] [15].

El principal motivo de estas limitaciones reside en la falta de algoritmos capaces de codificar y decodificar con precisión la señal analógica a *spikes*, o viceversa. Actualmente existen dos enfoques: codificación basada en frecuencia (*rate coding*) y codificación temporal (*temporal coding*). Trabajos recientes [9] han señalado que la codificación temporal contiene mayor cantidad de información relevante que la basada en frecuencia. Sin embargo, las técnicas de codificación basadas en tiempo también presentan una serie de limitaciones, debido a que o bien carecen de proceso de reconstrucción de la señal (decodificación) o los errores que presentan son elevados. En este sentido, en este trabajo se emplea un novedoso algoritmo temporal de codificación basado en PWM [1] [10], el cual

 https://doi.org/10.17979/spudc.9788497498418.0216

supera con creces la precisión en la reconstrucción de la señal original de los algoritmos predecesores y permite ajustar mediante un parámetro la relación entre la precisión y el coste computacional y en memoria.

Otra de las limitaciones existentes en el uso de las SNN se presenta con los algoritmos de entrenamiento supervisado. En las ANN y en las Redes Neuronales Reccurentes (*Recurrent Neural Networks*, RNN) los algoritmos por excelencia de entrenamiento supervisados son *BackPropagation* (BP) y *BackPropagation Through Time* (BPTT), respectivamente. Sin embargo, la dificultad principal a la hora de implementar dichos algoritmos de entrenamiento radica en que la función de activación de las SNN no es diferenciable.

En los últimos años se ha hecho un importante esfuerzo para desarrollar diferentes técnicas para poder implementar el entrenamiento supervisado en las SNN [11], pudiéndose diferenciar principalmente dos enfoques: el entrenamiento supervisado indirecto, que se basa en entrenar primero una ANN y luego transformarla en una SNN; y el entrenamiento supervisado directo. Dentro de este último enfoque ha irrumpido con fuerza el entrenamiento basado en *surrogate gradient*, el cual se basa en que en la propagación de las muestras se utilizan las funciones de cómputo propias de las neuronas de las SNN, pero en el cálculo de la retropropagación se sustituyen estas funciones por unas similares a las funciones de activación de las ANN y RNN, permitiendo así solventar los problemas de diferenciabilidad y pudiendo aplicar el BP a las SNN.

En este contexto, en este artículo se presenta una prueba de concepto de un entrenamiento supervisado con Redes Neuronales de Impulsos para *forecasting* de series temporales. Más concretamente se va a realizar un *forecasting* de tipo *one step-ahead*, siendo el hiperparámetro objeto de estudio el número de instantes anteriores que se van a introducir en la SNN como variables de entrada. Por tanto, se trata de uno de los pocos trabajos presentes en la literatura enfocados a problemas de regresión, alejándose así del uso extendido de la clasificación de imágenes.

2. METODOLOGÍA

Para la realización del entrenamiento se ha utilizado el lenguaje de programación Python y principalmente las librerías Pytorch y SpikingJelly [2]. Pytorch es actualmente una de las herramientas más conocidas de Python en el campo de las SNN, debido a que es una herramienta que facilita la diferenciación automática de funciones. En cuanto a SpikingJelly, se trata de un entorno de código abierto basado en Pytorch, donde se implementan todos los algoritmos de entrenamientos de las SNN de este trabajo.

2.1. BASES DE DATOS

Para evaluar el resultado del entrenamiento de las SNN se hace uso de dos series temporales de dos bases de datos distintas con el fin de poder valorar la influencia de la dinámica de la señal en el aprendizaje de la SNN según la estrategia planteada en este trabajo. Por una parte, se emplea una base de datos de IBM, que hace referencia a la evolución del cierre de las acciones de IBM en el período comprendido entre el 17/05/1961 hasta el 2/11/1962. Esta serie temporal viene descrita en [4] y se ha podido descargar desde *Time Series Data Library* [5]. Se trata de una serie temporal de dinámica lenta con un total de 369 muestras.

Por otro lado, también se va a emplear otra base de datos con una dinámica más rápida que la de IBM. Se trata de la base de datos EEG, descrita en [6], que se basa en la evolución temporal de las señales cerebrales de una persona cuando mueve la muñeca arriba y abajo o la mantiene quieta. Esta señal está constituida por 4097 muestras.

2.2. CODIFICACIÓN MEDIANTE PWM

El punto de partida para el entrenamiento de las SNN es la codificación de las señales de IBM y EEG en *spikes* mediante el algoritmo PWM. Tal y como se detalla en [10], este algoritmo se basa en dos hiperparámetros: el número de puntos por señal portadora o *carriers* (npc) y el número total de señal portadoras o carriers (nc). Para el análisis se han definido valores para estos hiperparámetros de manera que van a permanecer invariables a lo largo de toda la metodología con el objetivo de poder analizar la influencia en el aprendizaje del número de instantes anteriores. Concretamente, los criterios y valores finales escogidos para ambos hiperparámetros son los siguientes:

- Nc. Para forecasting es necesario que haya un *spike* por cada señal portadora, por tanto, el valor del nc va a ser igual al número de muestras de cada señal menos 1. Así, en el caso de la señal IBM nc adquiere un valor de 368, mientras que para EEG es de 4096.
- Npc = 64. El valor npc es el encargado de establecer la relación entre la precisión de la de/codificación y el coste computacional y en memoria. Se considera que con un valor de 64 se consigue una precisión suficiente para esta

https://doi.org/10.17979/spudc.9788497498418.0216

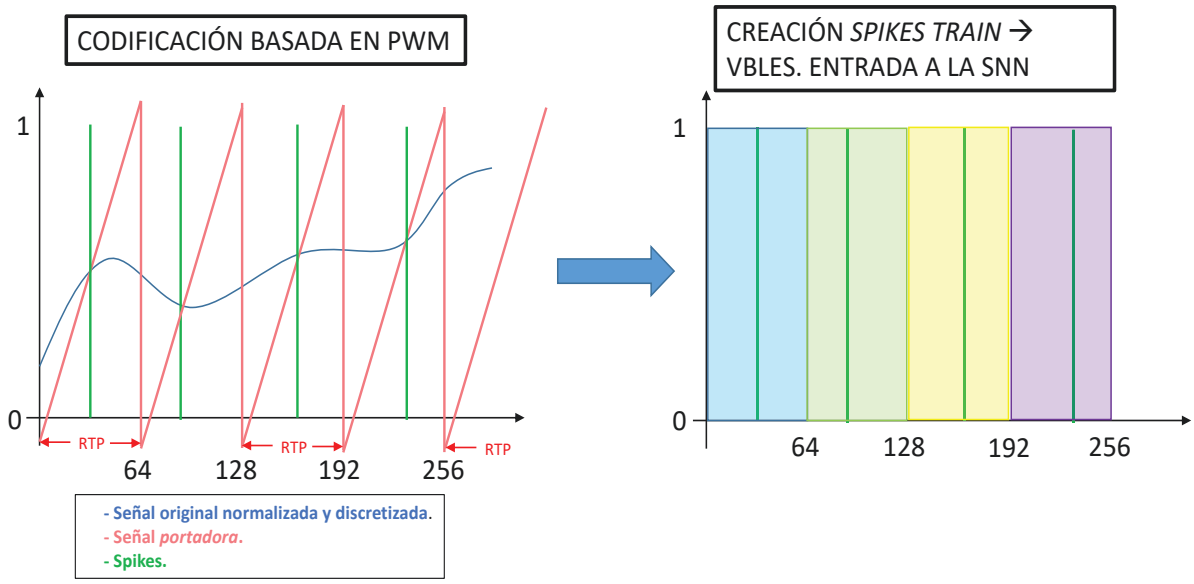

Figura 1: Creación de los *spikes train* como variables de entrada a la red

primera aproximación, siendo el coste computacional y en memoria muy reducido.

En la Figura 1 se muestra el proceso de codificación. Una vez aplicado el algoritmo PWM, se consigue dividir los datos de entrenamiento en ventanas, siendo cada una de estas ventanas las que van a conformar las muestras de entrenamiento de la SNN. El tamaño de estas ventanas está directamente relacionado con el valor npc, por tanto, en el caso del presente trabajo, al haber definido un npc de 64 puntos, cada muestra de entrenamiento está formada por 63 ceros y un uno (el spike).

2.3. INTRODUCCIÓN DE LOS DATOS

En la metodología que se define en este trabajo se plantea que los trenes de impulsos o *spikes train* vengan dados por cada una de las ventanas obtenidas con el algoritmo de codificación basado en PWM.

La forma de introducir los datos va a venir determinada por el tipo de modelo de red empleado. En el caso del trabajo se va a optar por un modelo *stateless* donde se van a introducir todos los datos de entrada en único instante de tiempo o *timestep* [12].

En este sentido, si se emplea la variable i para identificar a cada una de las ventanas de la serie temporal, siendo $i \in [0, ..., Nc-1]$, se debe señalar que para realizar el *forecasting* es necesario tener información de todas las ventanas anteriores que se vayan a emplear. En otras palabras, en el caso de que se empleen 3 instantes anteriores para los 3 primeros instantes (i=0,i=1,i=2) no se puede realizar la predicción, debido a que no se tiene información de los instantes i=-3, i=-2 y i=-1. Así, tal y como se señala en la Figura 2, para el caso de análisis de 3 instantes anteriores, el primer instante a predecir es i=3. Además, en la figura anterior también se señala cómo se configura el conjunto de entradas en el entrenamiento (*train data*, X(i)) y el *target* o salida deseada del entrenamiento (*target data*, Y(i)), donde este último es siempre un valor real de la serie temporal.

2.4. ESTRUCTURA

Al tratarse de una primera aproximación se ha querido emplear un modelo lo más simple posible. Así, el modelo de red no va a disponer de ninguna capa oculta, estando solo constituido por una capa de entrada y una capa de salida. El número de neuronas en la capa de entrada varía según el número de instantes que se quiera introducir, siendo su valor igual al número de instantes anteriores multiplicado por el valor npc. Esto es debido a que, al emplear un modelo *stateless*, es necesario procesar toda la información en un único instante de tiempo o *timestep*. La Figura 2 ilustra la aplicación de este criterio en un ejemplo ficticio. En concreto, para tres instantes anteriores el vector con los datos de entrenamiento, X(i), está formado por 192 muestras ($instantes\ anteriores * npc = 3 * 64 = 192$). A su vez, como la finalidad de la red es realizar un *forecasting* tipo *one step-ahead*, la capa de salida de la red debe estar formada por un número de neuronas igual al valor npc, lo cual también se puede observar en la Figura 2 en la que Y(i) tiene un tamaño igual a 64.

https://doi.org/10.17979/spudc.9788497498418.0216

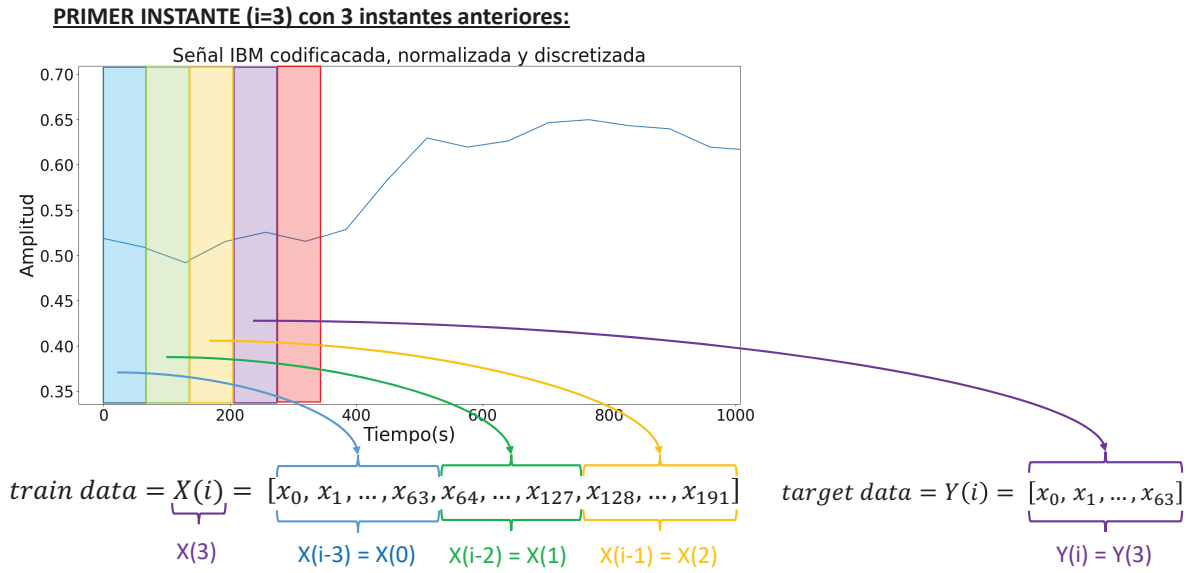

Figura 2: Formación conjuntos *train data* y *target* para el primer instante de la señal IBM

Todas las neuronas que se van a emplear están basadas en el modelo *Leaky Integrate and Fire* (LIF) con $\tau = 100{,}0$, $V_{threshold} = 1{,}0$ y $Vreset = 0{,}0$. Además, durante el entrenamiento se emplea un tamaño de *batch* de 1 y ninguna neurona requiere del término *bias*.

2.5. FUNCIÓN DE ERROR

La función de error es uno de los puntos críticos de cualquier entrenamiento con redes neuronales. En este trabajo se considera que el tipo de función de error debe ser coherente con el algoritmo de codificación de *spikes*. En este sentido, al tratarse el algoritmo de codificación PWM de un método de codificación basado en tiempo, la función de error escogida se basa en comparar el instante de tiempo donde se produce el *spike* en la salida de la red con el instante de tiempo donde se produce el *spike* en el valor deseado. Por simplicidad, si en la salida de la red se emite más de un *spike* sólo se compara el primero de todos ellos, y de hecho es el que se utilizará para la decodificación de la señal. Para detectar los instantes de disparo dentro de cada señal portadora se utiliza una máscara.

En este trabajo se opta por definir una función de error estricta en cuanto a la diferencia de instantes de disparo, de tal forma que:

- Si el instante de disparo del valor deseado coincide con el primer instante de disparo de la red $\rightarrow Error = 0$.
- Si los instantes no coinciden $\rightarrow Error = (instante_{target})^2$

2.6. METODOLOGÍA DE ENTRENAMIENTO

En el presente trabajo se emplea el algoritmo SpikingJelly [2] para poder implementar el entrenamiento supervisado, el cual está basado en *surrogate gradient*. Más concretamente, en SpikingJelly se aplican las ecuaciones diferenciales de las neuronas LIF para hacer la propagación hacia adelante de las muestras introducidas en la SNN, mientras que en la retropropagación se emplea la función *Sigmoid* para solventar los problemas de diferenciabilidad. Además, se hace uso del optimizador Adam.

El proceso de entrenamiento se basa en BP y se puede resumir en los siguientes pasos: i) Inicialización de los potenciales de las neuronas a cero; ii) Propagación de una muestra de entrenamiento; iii) Cálculo de la función de error y retropropagación; iv) Actualización de los pesos; v) Introducción de una nueva muestra de entrenamiento.

Además, el entrenamiento se realiza con 100 épocas, un ratio de aprendizaje igual a 10^{-3} y variando el número de instantes anteriores de 1 hasta 10.

Mencionar que el algoritmo actualmente no tiene implementado una técnica de regulación, haciendo que la red escogida para testear sea siempre la obtenida en la de la última época. Además, al tratarse de una primera aproximación se ha centrado el esfuerzo en que la red aprenda, dejando la generalización para más adelante. Así, se ha decidido no someter a las series temporales a ninguna división que cree los conjuntos de entrenamiento, valida-

https://doi.org/10.17979/spudc.9788497498418.0216

ción y test, haciendo que el 100 % de las muestras se introduzcan en la etapa de entrenamiento y en la etapa de testeo se vuelva a introducir todo este conjunto para comprobar el poder de aprendizaje de la SNN.

3. RESULTADOS

A continuación, se muestran los resultados obtenidos tanto para la serie temporal de IBM como para la de EEG.

Con el objetivo de valorar el rendimiento del aprendizaje de las redes se van a emplear un conjunto de métricas ampliamente utilizadas en problemas de *forecasting*: distancia euclídea, error MSE y error MAE. Con ellas se va a comparar con precisión la señal original normalizada con la predicción realizada por la SNN. Para ello, la salida de la red se debe decodificar con el algoritmo PWM.

3.1. IBM

Los resultados obtenidos para la base de datos IBM son lo que se muestran en la Tabla 1. Esta tabla muestra la variación de la métricas en función del valor del número de instantes anteriores que se introducen a la red. Se puede observar que con 9 instantes anteriores se consiguen los mejores resultados.

Tabla 1: **Análisis resultados señal IBM**

	RESULTADOS TEST		
Inst. Ant.	**Distancia Euclídea**	**Error MSE**	**Error MAE**
1	6.7037	0.004	0.046
2	17.442	0.0145	0.0675
3	11.217	0.0057	0.0429
4	11.0342	0.0056	0.0464
5	10.6149	0.005	0.0477
6	8.1816	0.003	0.0378
7	9.151	0.0037	0.0417
8	9.3607	0.0039	0.0418
9	8.1348	0.0029	0.0392
10	8.2802	0.003	0.0411

En la Figura 3 se muestra la comparación entre la señal original normalizada y la señal decodificada aprendida con la SNN con los valores de 1 instante anterior y 9 instantes anteriores, respectivamente. La mejora en el aprendizaje es muy significativa cuando se aumenta el número de instantes anteriores introducidos en las neuronas de entrada de la SNN, lo cual se puede deber a que la base de datos IBM presenta una dinámica relativamente lenta y, por tanto, un aumento del número de instantes anteriores fomenta que la red aprenda mejor su dinámica y se adapte mejor a los cambios producidos en la serie, pudiendo realizar así una predicción más acertada. Además, también se aumenta el número de *spikes* emitidos por la red, lo que facilita que haya más puntos para realizar la regresión del algoritmo de decodificación PWM.

3.2. EEG

En la Tabla 2 se indican los resultados obtenidos con la base de datos EEG. A diferencia del caso anterior, los mejores resultados se obtienen con 1 instante anterior.

Tabla 2: **Análisis resultados señal EEG**

	RESULTADOS TEST		
Instantes Anteriores	**Distancia Euclídea**	**Error MSE**	**Error MAE**
1	27.0592	0.0028	0.0407
2	42.7113	0.007	0.066
3	61.8138	0.01459	0.0988
4	79.6522	0.0242	0.1316
5	93.8176	0.0336	0.1568
6	104.5737	0.0418	0.1785
7	113.5066	0.0492	0.1955
8	120.1141	0.0552	0.2087
9	125.8145	0.0605	0.2202
10	131.1931	0.0658	0.2306

En la Figura 4 se puede observar la diferencia existente entre la señal original y la decodificada para el caso de 1 instante anterior y 10 instantes anteriores, respectivamente. La diferencia principal entre ambas predicciones es que en el caso de 10 instantes anteriores el número de *spikes* producidos por cada conjunto de muestras de entrenamiento es muy superior, haciendo que cuando se decodifique con el primer *spike* emitido éste esté más cerca del *Reference Time Point* (RTP) de la señal portadora del PWM (ver Figura 1) y la magnitud al decodificar sea inferior a la de la señal original.

En comparación con la serie temporal de IBM, la serie temporal de la base de datos EEG presenta una dinámica mucho más rápida, haciendo que los instantes siguientes no dependan tanto de los instantes anteriores y se produzcan peores predicciones conforme se va aumentando el número de instantes anteriores introducidos en la SNN. Esto se puede observar en la evolución de las métricas de la Tabla 2, donde se produce un aumento significativo de los errores a medida que se aumenta el número de instantes anteriores.

4. CONCLUSIONES

En este trabajo se propone una nueva estrategia de entrenamiento supervisado con Redes Neuronales

https://doi.org/10.17979/spudc.9788497498418.0216

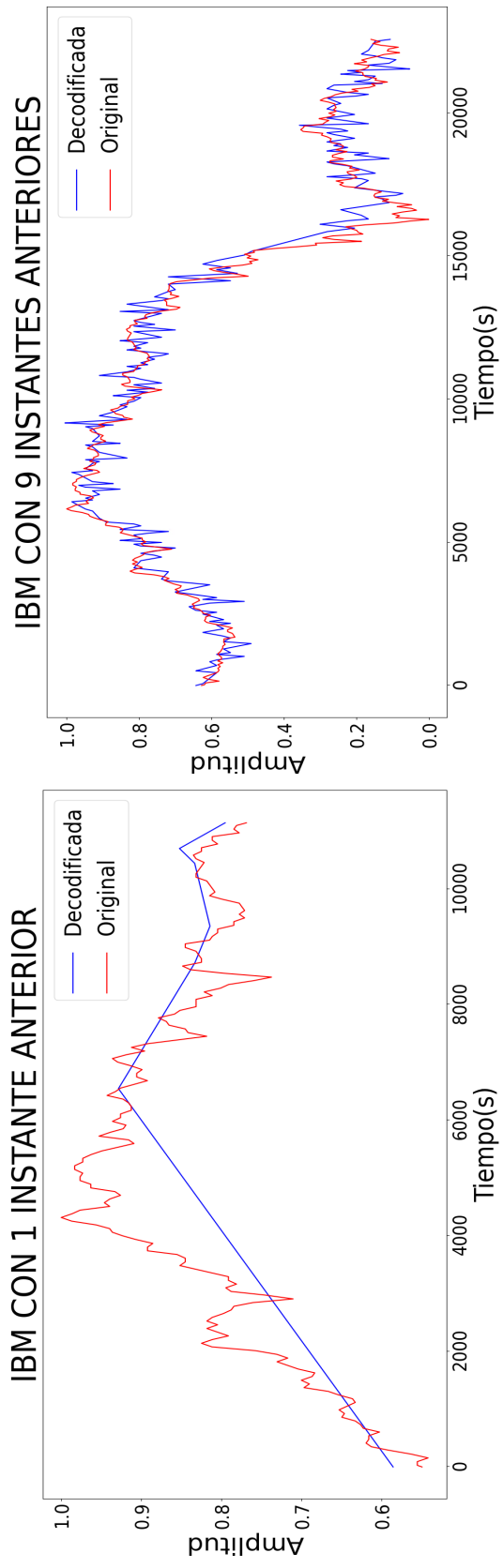

Figura 3: Comparación señal IBM original vs decodificada con 1 y 9 instantes anteriores

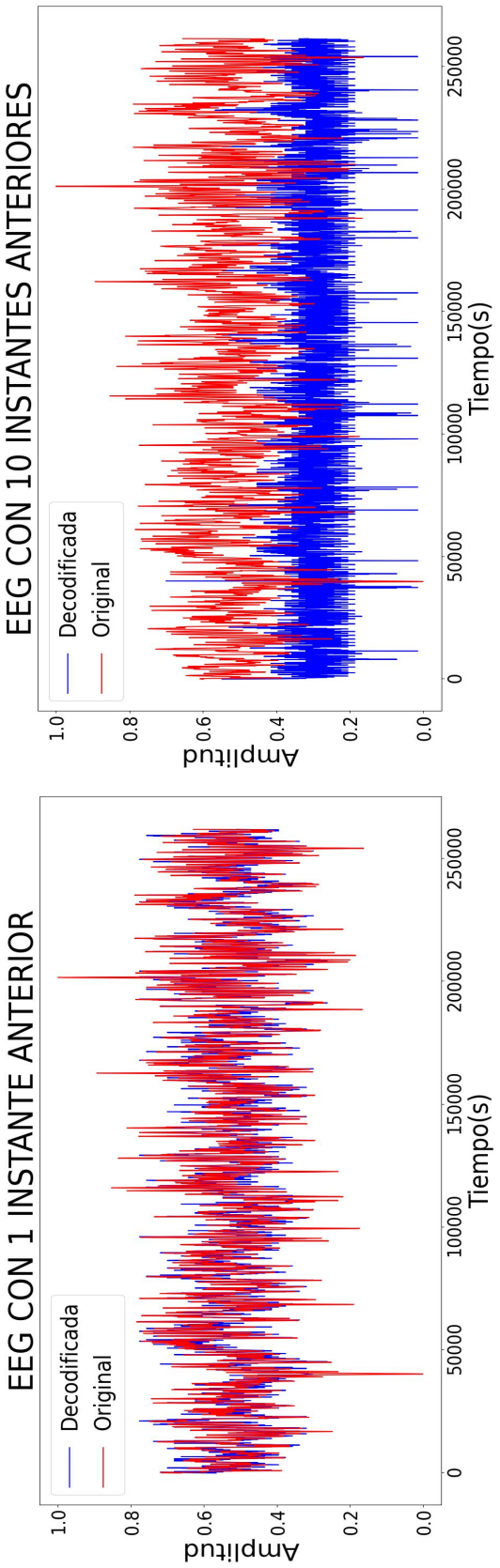

Figura 4: Comparación señal EEG original vs decodificada con 1 y 10 instantes anteriores

https://doi.org/10.17979/spudc.9788497498418.0216

de Impulsos para *forecasting* de series temporales, siendo uno de los primeros trabajos realizados en este campo. Aunque sea una primera aproximación, este trabajo ha permitido extraer una serie de conclusiones muy importantes:

1. La función de error definida así como la forma de introducir los datos en la red han permitido validar un posible método de introducción de los datos en las SNN. Además, los resultados son muy prometedores en cuanto aprendizaje de la red.
2. El número de instantes anteriores que hay que introducir a la red depende de la dinámica de las series temporales. Así, con dinámicas más lentas se requiere un número mayor de instantes anteriores, debido a que al tener más información de la serie el ajuste es mejor. Por otro lado, con dinámicas rápidas, como la señal EEG, con un instante anterior es suficiente para adaptarse a los cambios producidos en la serie temporal.
3. A pesar de no haberse aplicado técnica de regulación y, por tanto, no haberse optimizado la selección de la SNN, los resultados obtenidos son satisfactorios, lo cual se refleja en los reducidos valores obtenidos tanto en el error MSE como en el error MAE, y que se pueden observar en los mejores casos de las Figuras 3 y 4.

Como trabajo futuro, actualmente se está implementando una técnica de regulación en el algoritmo de entrenamiento. Además, esta modificación en el algoritmo permitiría verificar el poder de generalización de las redes entrenadas. Por último, se está trabajando tanto en nuevas propuestas de funciones de error como en diferentes alternativas que permitan mejorar el entrenamiento de las SNN. En cuanto a la mejora de los valores de las métricas de error de los resultados se va a estudiar la inclusión de capas ocultas en la estructura de las redes, así como el incremento del valor npc del algoritmo de codificación.

Agradecimientos

Este trabajo ha sido financiado por el Departamento de Educación del Gobierno Vasco (proyectos ref. PIBA_2020_1_0008 y ref. IT1726-22), la Universidad del País Vasco UPV/EHU (ref. GIU19/045) y el FEDER/Ministerio de Ciencia e Innovación - Agencia Estatal de Investigación/Proyecto (ref. PID2020-112667RB-I00).

English summary

SUPERVISED TRAINING FOR SPIKING NEURAL NETWORK

Abstract

This article proposes a new supervised training strategy with Spiking Neural Networks (SNN) for time series forecasting. Currently, the vast majority of work on SNN is focused on classification tasks, especially image classification. In this sense, this paper is one of the first works to apply SNN for time series forecasting, whose results are very promising. Two benchmark datasets have been used to validate the methodology: IBM stock market information and EEG signals. Among the results, it is shown that the performance of SNN depends, as expected, on the dynamic of the signal or time series to be forecast.

Keywords: Spiking Neural Network, PWM, time series, regression, forecasting.

Referencias

[1] A. Arriandiaga, E. Portillo, J. I. Espinosa-Ramos, and N. K. Kasabov. Pulse-width 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, oct 2020.

[2] W. Fang, Y. Chen, J. Ding, D. Chen, Z. Yu, H. Zhou, Y. Tian, and other contributors. Available online: Spikingjelly. https://github.com/fangwei123456/spikingjelly, accessed June 2022.

[3] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2661–2671, 2021.

[4] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control* . 2015.

[5] R. Hyndman. *Time Series Data Library.* 2015.

[6] N. K. Kasabov. *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence.* Springer, Heidelberg, Berlin, 2019.

 https://doi.org/10.17979/spudc.9788497498418.0216

[7] H. Kim, S. Hwang, J. Park, S. Yun, J. H. Lee, and B. G. Park. Spiking Neural Network Using Synaptic Transistors and Neuron Circuits for Pattern Recognition with Noisy Images. *IEEE Electron Device Letters*, 39(4):630–633, apr 2018.

[8] S. Kim, S. Park, B. Na, and S. Yoon. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11270–11277, apr 2020.

[9] V. Lopes-dos Santos, S. Panzeri, C. Kayser, M. E. Diamond, and R. Quian Quiroga. Extracting information in spike time patterns with wavelets and information theory. *Journal of Neurophysiology*, 113(3):1015–1033, 2015.

[10] S. Lucas, A. Arriandiaga, E. Portillo, A. Zubizarreta, and I. Cabanes. Compresión de datos de tipo real basada en un novedoso algoritmo de codificación para redes neuronales de impulsos. *XLII JORNADAS DE AUTOMÁTICA : LIBRO DE ACTAS*, pages 175–182, aug 2021.

[11] E. O. Neftci, H. Mostafa, and F. Zenke. 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, nov 2019.

[12] U. Rançon, J. Cuadrado-Anibarro, B. R. Cottereau, and T. Masquelier. StereoSpike: Depth Learning with a Spiking Neural Network. sep 2021.

[13] D. Reid, H. Tawfik, A. J. Hussain, and H. Al-Askar. Forecasting weather signals using a polychronous spiking neural network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9225:116–123, 2015.

[14] Z. Yan, J. Zhou, and W. F. Wong. Energy efficient ECG classification with spiking neural network. *Biomedical Signal Processing and Control*, 63:102170, jan 2021.

[15] M. Yao, H. Gao, G. Zhao, D. Wang, Y. Lin, Z. Yang, and G. Li. Temporal-wise Attention Spiking Neural Networks for Event Streams Classification. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10221–10230, 2021.

https://doi.org/10.17979/spudc.9788497498418.0216