COMPRESIÓN DE DATOS DE TIPO REAL BASADA EN UN NOVEDOSO ALGORITMO DE CODIFICACIÓN PARA REDES NEURONALES DE IMPULSOS

Sergio Lucas, Ander Arriandiaga, 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 propone la compresión de datos de tipo real basada en un nuevo algoritmo de codificación para Redes Neuronales de Impulsos inspirado en la conocida modulación por ancho de pulso (PWM). Esta propuesta presenta una serie de ventajas como la simplicidad del algoritmo. Así, permite al usuario establecer el compromiso deseado entre calidad y consumo de recursos de memoria mediante la selección de un sencillo parámetro.

Palabras clave: Redes Neuronales de Impulsos, codificación, decodificación, PWM, compresión.

1 Introducción

En los últimos años se ha producido un incremento en el protagonismo de las técnicas de Inteligencia Artificial (IA), no sólo a nivel industrial, donde esta tecnología lleva usándose desde 1980 [10], sino en una gran variedad de campos de aplicación.

Una de las técnicas más reconocidas dentro de la IA son las Redes Neuronales Artificiales (*Artificial Neural Networks*, ANN), las cuales tratan de imitar la estructura de las redes biológicas. El principal atractivo de las ANN reside en su capacidad de aprender y generalizar a partir de datos reales. Así, han sido ampliamente utilizadas de forma satisfactoria en problemas de clasificación y regresión. Sin embargo, las neuronas artificiales tradicionales quedan muy lejos de imitar a las reales, siendo su coste energético y de memoria muy superior al de las redes neuronales biológicas.

Las neuronas biológicas se comunican mediante series de impulsos eléctricos. Todos estos impulsos son iguales entre sí, por lo que se concluye que la información no está codificada en la forma de la acción potencial, sino en el número y la distribución temporal de los impulsos [5], es decir, la importancia radica en cuándo suceden los impulsos. Con el objetivo de imitar con mayor fidelidad a estas neuronas biológicas ha surgido la tercera generación de redes o también conocidas como Redes Neuronales de Impulsos (*Spiking Neural Networks*, SNN). A diferencia de los modelos ANN, en este tipo de redes la información no se codifica en valores analógicos, sino en series de impulsos o *spikes*.

Debido a que su codificación se basa en la distribución temporal de los impulsos, las SNN poseen características innatas para manipular la información temporal. Así, las SNN han sido ampliamente utilizadas en la clasificación de datos temporales en múltiples campos, como pueden ser la compresión y reconstrucción de imágenes [14] o la detección y clasificación visual de objetos [6]. Sin embargo, apenas se ha producido algún avance para modelar evoluciones temporales mediante SNN en problemas de regresión, estando los pocos trabajos [15][16] realizados hasta la fecha más enfocados a la clasificación y sin que se realice en ellos una reconstrucción de los impulsos en las señales analógicas originales.

La razón principal de estas limitaciones radica en la falta de algoritmos capaces de codificar y reconstruir o decodificar con precisión las señales analógicas en impulsos, o viceversa. Por tanto, la codificación y decodificación se trata de uno de los principales retos de las SNN. Así, entre los diferentes enfoques de codificación existentes, se destaca la codificación basada en frecuencia (*rate coding*) y la codificación temporal (*temporal coding*).

La codificación frecuencial se basa en el número de impulsos sucedidos en ventanas de tiempo móviles. Aunque está ampliamente extendido su uso, trabajos recientes [12][13] sostienen que la codificación basada en tiempo contiene información relevante sobre la señal original, a diferencia de este tipo de codificación. La codificación basada en tiempo se basa en la distancia temporal entre impulsos. Sin embargo, los métodos propuestos también presentan una serie de inconvenientes, debido a que, o bien presentan errores significativos en el proceso de decodificación, o directamente no contemplan este proceso, el cual es necesario para afrontar los problemas de regresión.

En este sentido, en este artículo se propone una novedosa estrategia de compresión de datos de

https://doi.org/10.17979/spudc.9788497498043.175

tipo real basada en un nuevo algoritmo de codificación para Redes Neuronales de Impulsos. El algoritmo de codificación y decodificación es un algoritmo basado en tiempo y está inspirando en la conocida modulación por ancho de pulso (*PulseWidth Modulation, PWM*).

El resto del artículo está estructurado de la siguiente forma: en la sección 2, se presenta el algoritmo de codificación basado en PWM, incluyendo su diseño, sus parámetros y un ejemplo de aplicación. En la sección 3 se detalla la propuesta de compresión objeto de este artículo. Por último, en la sección 4 se exponen las conclusiones.

2 Algoritmo de codificación basado en PWM (PWM-SNN)

2.1 Diseño del algoritmo PWM-SNN

El PWM se trata de una de las técnicas más comunes en electrónica de potencia para realizar la conversión analógica a digital. Para ello, se comparan dos señales: a) la señal de referencia r(t), que es la señal a convertir o modular; b) una señal portadora c(t), la cual es periódica y donde habitualmente se emplea una onda triangular o diente de sierra. La forma de calcular la señal PWM se puede resumir en:

$$PWM(t) = f(x) = \begin{cases} 1, \ si \ r(t) \ \geq \ c(t) \\ 0, \ si \ r(t) \ \leq \ c(t) \end{cases} \qquad (1)$$

La idea subyacente en el algoritmo de codificación que se presenta en este trabajo se basa en emplear la metodología PWM tanto en la codificación de los impulsos o spikes, como en la decodificación o reconstrucción de la señal original. Así, en la codificación, primero, se calcula la señal PWM correspondiente a la señal de referencia analógica y, posteriormente, se considera cada uno de los flancos ascendentes como un spike, es decir, el spike se produce en la intersección entre la señal de referencia y la portadora periódica (este proceso se puede ver en la Figura 1a). Visto el proceso de codificación, se puede deducir que cada spike se genera respecto a un punto de referencia en el tiempo (*Reference Time Point*, RTP), punto en el que la señal portadora c(t) adquiere el valor 0. Consecuentemente, cuanto mayor sea la amplitud de la señal de referencia r(t), más lejos del RTP se genera el spike. Por el contrario, a menor amplitud de r(t), más cerca del RTP se genera el spike.

En cuanto a la decodificación se aplica el método explicado anteriormente pero a la inversa. Esto se puede realizar debido a que se emplea la misma señal periódica para codificar y decodificar. En el caso de la decodificación se parte de los spikes y de la señal portadora, se calculan los valores discretos de la señal original reconstruida mediante la intersección entre los spikes y la señal portadora y, por último, una vez calculados todos los valores discretos, se calculan el resto de los valores de la señal original mediante interpolación. Este proceso se visualiza en la Figura 1b.

Los métodos existentes de codificación para SNN no permiten calcular un spike por cada paso de tiempo [3], siendo ésta una condición indispensable para afrontar problemas de predicción o regresión. Mediante el método PWM-SNN propuesto siempre y cuando la señal de referencia se encuentre en el rango de la portadora, se produce una intersección entre las señales y, por tanto, se produce un spike en cada paso de tiempo, solventando uno de los principales problemas presentes en el campo de las SNN. Además, este método dispone de una serie de parámetros que permiten aumentar o disminuir la precisión de los resultados y el coste computacional [2].

2.2 Parámetros del algoritmo PWM-SNN

Una de las principales ventajas del método de codificación PWM-SNN presentado es que se trata de un método fácilmente parametrizable, pudiéndose reconstruir la señal con gran precisión.

En el método PWM-SNN se emplean dos señales. La señal de referencia se considera a la señal que se debe convertir o modular y viene impuesta por el problema o proceso con el que se esté trabajando. La segunda señal se trata de la señal portadora o *carrier*, la cual es una señal periódica y donde habitualmente se emplea una señal triangular o diente de sierra, como es el caso de la Figura 1. Esta segunda señal se va a poder parametrizar en función de las características de la señal de referencia.

La primera consideración a tener en cuenta de la señal portadora es su amplitud. Como se ha comentado anteriormente es muy importante que la señal de referencia se encuentre en el rango de la portadora, para poder compararlas y calcular de esta forma un spike por cada paso de tiempo. Así, en un primer paso se debe normalizar la señal de referencia entre los valores 0 y 1 y definir la señal portadora en el mismo rango.

Por otra parte, en el método PWM-SNN presentado se emplean dos parámetros que permiten modelar la señal portadora para ajustar el nivel de precisión de la reconstrucción: 1) él número de ondas de la señal portadora (nc, *number of carrier waves*), el cual se encuentra directamente relacionado con el ancho del pulso; 2) el número

https://doi.org/10.17979/spudc.9788497498043.175

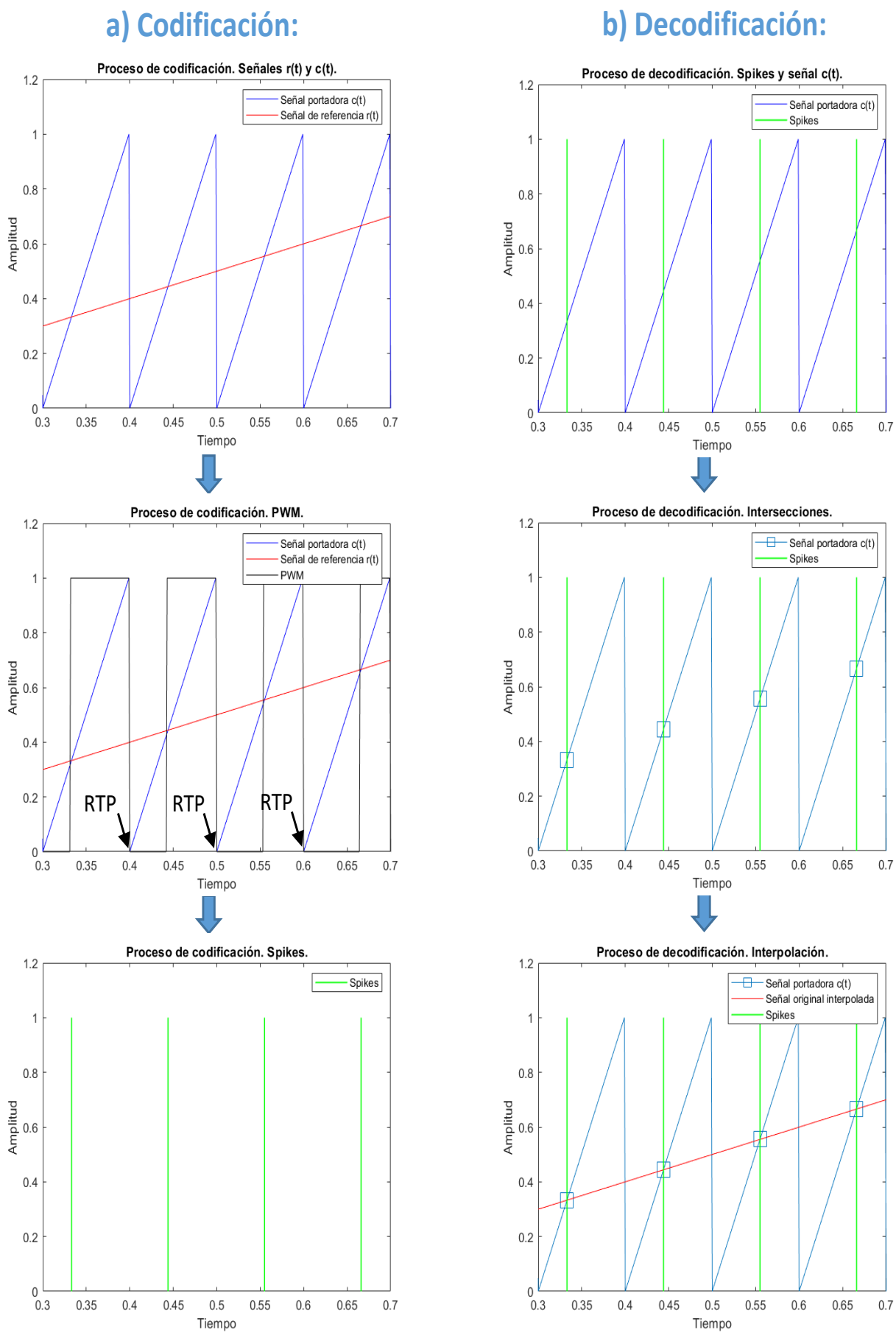

Figura 1: Resumen procesos codificación y decodificación

https://doi.org/10.17979/spudc.9788497498043.175

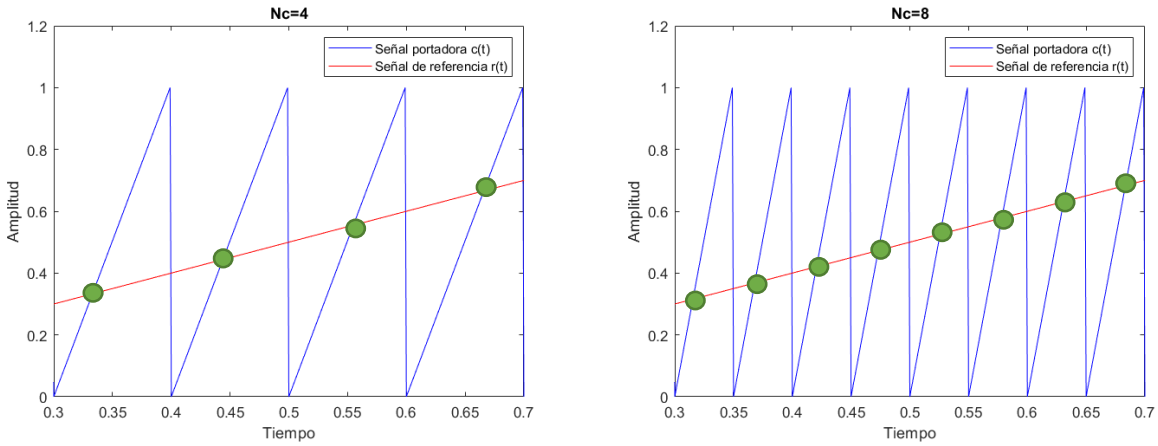

Figura 2: Comparación nc=4 y nc=8

de puntos por onda de la señal portadora (npc, *number of points per carrier wave*).

El número de ondas nc o ancho del pulso se encuentra directamente relacionado con la tasa de muestreo de la señal original r(t). De acuerdo con el teorema de muestreo de Nyquist-Shannon, una función x(t) con un ancho de banda limitado W puede ser reconstruida de forma exacta si se emplea una tasa de muestreo superior a 2W [17]. Así, en el muestreo tradicional un método empleado muy común consiste en multiplicar la señal analógica x(t) por una señal de muestreo s(t), basada en una serie de impulsos Delta de Dirac distribuidos a intervalos regulares, de manera que se captura un valor de x(t) por impulso.

En este sentido, el PWM se puede entender como un equivalente a un método clásico de muestreo [7]. Sin embargo, en vez de muestrear por cada impulso (Delta de Dirac), se muestrea utilizando la señal portadora. Un buen algoritmo de codificación se distingue porque devuelve una señal reconstruida prácticamente idéntica a la original. Esto en el algoritmo presentado se consigue seleccionando un correcto valor de nc, asegurando un correcto muestreo y, con ello, una correcta reconstrucción. En la Figura 2 se puede visualizar el proceso de muestreo de una señal de referencia con dos valores distintos de nc. Para la selección de este parámetro se decide seleccionar un valor de nc igual al número de puntos o muestras adquiridas (menos uno) disponibles en un fichero digitalizado. Lo que es lo mismo, se elegirá una anchura de pulso igual al periodo de muestreo seleccionado por el experto de dominio en el caso de aplicación específico.

Otro factor que se debe tener en cuenta es la resolución, la cual se define como el mínimo cambio que se es capaz de detectar e influye en la precisión de la señal reconstruida. Así, en el caso del PWM-SNN este mínimo cambio se determina por el número de puntos de la señal portadora, es decir, por el segundo parámetro objeto de estudio, el npc. De esta forma, el npc se encuentra directamente relacionado con la resolución de la señal y determina el número de instantes posibles en los que se puede emitir el spike. En la Figura 3 se puede visualizar cómo a mayor npc, mayor cantidad de instantes posibles y, por tanto, mayor precisión se consigue en la señal reconstruida. Como hipótesis de partida se debe buscar un compromiso entre el npc y la finalidad de la aplicación, debido a que al aumentar el npc, a parte de incrementarse la precisión de la salida, también se aumenta el número de puntos de la señal portadora y con ello el coste de memoria.

2.3 Ejemplo de codificación

Para validar los resultados obtenidos con este algoritmo de codificación se va a emplear una base de datos formado por la grabación de voz de una persona. La grabación se corresponde con el discurso de una persona en la *Carnegie Mellon University ARTIC* [9].

Tabla 1: **Listado de nc y npc**

Valores objeto de estudio					
nc	14120	28240	56480	112960	225920
npc	32	64	128	256	512

Se trata de una grabación adquirida con una frecuencia de muestreo de 32kHz durante 3,53 segundos, con lo que se dispone de un total de 112.960

https://doi.org/10.17979/spudc.9788497498043.175

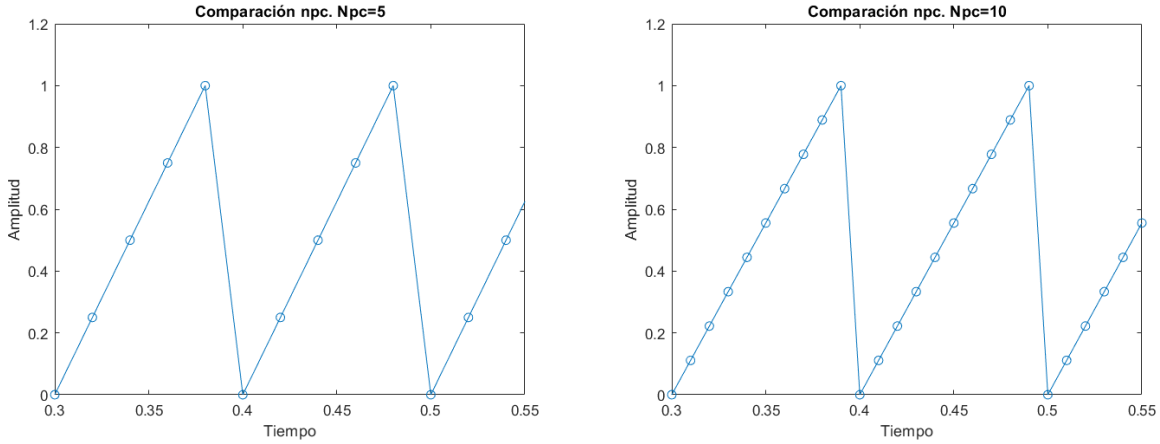

Figura 3: Comparación npc=5 y npc=10

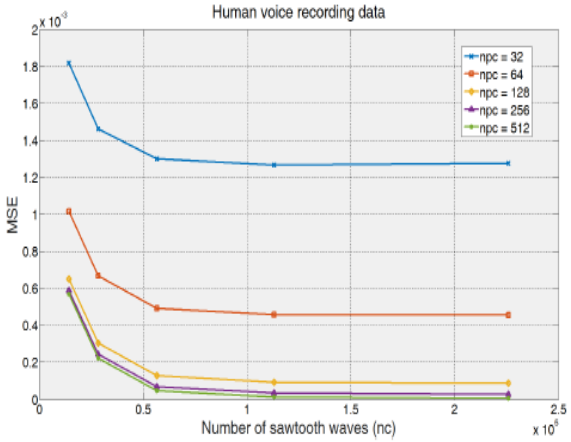

Figura 4: Error MSE con base de datos ARCTIC [2]

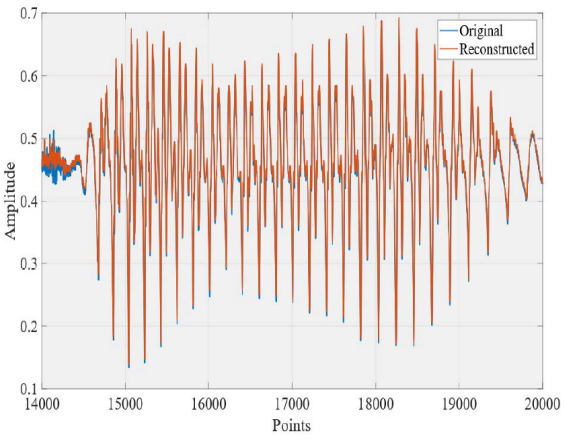

Figura 5: Comparación entre señal original y reconstruida para nc=56.480 y npc=256 [2]

muestras. El conjunto de valores de nc y npc que se van a estudiar son los que se representan en la Tabla 1.

Los errores medios cuadráticos cometidos (*Mean Square Error, MSE*) para la codificación y decodificación de la señal de la grabación de voz se visualizan en la Figura 4. Con este gráfico se verifica cómo con valores superiores a nc=112.960 el incremento en la precisión es mínimo, lo cual confirma la hipótesis planteada en el apartado anterior. Aumentar el número de ondas nc por encima del número de muestras que se emplean, incrementa el número de puntos codificados, pero sin una mejora notoria en la reconstrucción de la señal. Además, para visualizar el proceso de reconstrucción se adjunta la Figura 5, donde se puede comprobar la diferencia entre la señal original y reconstruida para un nc=56.480 y un npc=256.

Por tanto, una vez visualizados los resultados, exceptuando los casos donde se tienen unas fuertes restricciones de precisión y coste computacional, se recomienda el uso de un nc correspondiente con el periodo de muestreo utilizado para la adquisición de la señal.

3 Compresión de datos reales basada en PWM-SNN

Partiendo del nuevo algoritmo de codificación para Redes Neuronales de Impulsos PWM-SNN, esta sección presenta una aplicación basada en dicho algoritmo cuyo objetivo es comprimir datos de tipo real.

Así, esta propuesta presenta una nueva estrategia de compresión que ofrece la ventaja de que el ratio de compresión de datos C (ecuación 2) depende

https://doi.org/10.17979/spudc.9788497498043.175

únicamente del parámetro npc. En concreto, la estrategia de compresión se basa en almacenar la posición que ocupa cada spike dentro de la señal portadora, tal y como se ilustra en el ejemplo de la Figura 6. En este ejemplo, para un valor de npc igual a 8, el valor pasa a ocupar 3 bits tras aplicar esta estrategia.

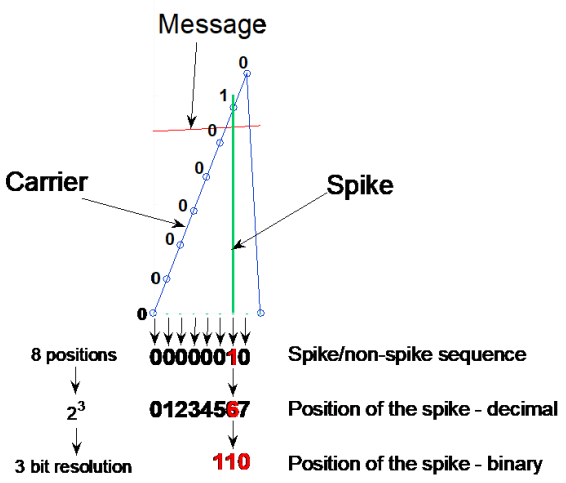

Figura 6: Compresión de los datos para la codificación PWM [2]

Los algoritmos de compresión de datos se dividen en dos tipos: sin pérdida *(lossless)* y con pérdida *(lossy)* [1]. Mientras los primeros permiten la recuperación exacta de los datos originales, los segundos pierden información original durante el proceso de compresión, con lo que no permiten la recuperación completa de los datos originales.

La mayoría de los trabajos sobre compresión de datos de doble precisión (double-precision floating-point format IEEE754) se basan en algoritmos lossless porque se suele tratar de aplicaciones de gran precisión, donde se requiere una exactitud elevada [11]. En la mayor parte de los métodos propuestos se emplean predicciones lineales y se codifican los residuos más pequeños, empleando variantes de códigos de longitud variable no estadísticos [4], o estadísticos [8]. Aunque su uso es importante en ciertas aplicaciones, con los métodos lossless raramente se consigue alcanzar ratios de compresión superiores a 1.5 en datos de doble precisión [11].

En este sentido, el algoritmo de compresión presentado se puede considerar lossless o lossy dependiendo de las preferencias del usuario. Para muestra de ello, se muestra en la Tabla 2 los ratios de compresión C, definiendo C como el ratio entre el tamaño original de los datos y el tamaño tras la compresión.

$$C = \frac{Tamaño\ sin\ comprimir}{Tamaño\ comprimido} \tag{2}$$

Para ello, como es habitual, se toma como referencia el estándar IEEE 754, en concreto, los conocidos *single-precision floating-point format (binary 32)* y *double-precision floating-point format (binary 64)*, con tamaño sin comprimir igual a 32 y 64 bits, respectivamente. El tamaño comprimido corresponde al número de bits requeridos para codificar la posición del spike de la señal portadora, lo que depende directa y únicamente del valor del parámetro npc seleccionado por el usuario. Por tanto, a mayor resolución o valor npc, menor será el valor de C.

Tabla 2: **Variación del ratio de compresión C en función del parámetro resolución (npc)**

Número de puntos por onda (npc)	Resolución (bit)	Ratio de compresión C (N bit/resolución)	
		Binary32	Binary64
16	4 Lossy	8	16
32	5	6.4	12.8
64	6	5.33	10.66
128	7	4.57	9.14
256	8	4	8
512	9	3.55	7.11
1024	10 Lossless	3.2	6.4

En términos de algoritmos de compresión, esta propuesta presenta una serie de ventajas como son la simplicidad del algoritmo y la escalabilidad SNR, permitiendo al usuario establecer el equilibrio deseado entre calidad y consumo de recursos de memoria. Así, esta propuesta supone incrementar la eficiencia de la implementación hardware en coste de memoria.

Por tanto, se ha diseñado el algoritmo de compresión basándose en la estrategia arriba expuesta, de manera que los datos comprimidos puedan ser almacenados en el disco duro en un computador de destino (u otro soporte de memoria secundaria).

La implementación persigue almacenar en memoria RAM de manera consecutiva las sucesivas secuencias comprimidas de n bits aprovechando la palabra de memoria al completo. El valor de n corresponde a la resolución elegida tal que $npc = 2^n$.

Una vez almacenados los datos de tal forma en memoria RAM, es posible almacenar de forma permanente dichos datos comprimidos mediante el almacenamiento en fichero binario.

Esta implementación se ha programado en Python para longitudes de palabra de 32 bits, si bien es fácilmente extensible/parametrizable para 64 bits.

https://doi.org/10.17979/spudc.9788497498043.175

En la implementación se ha hecho especial uso de la función bin() de Python, la cual devuelve el string binario equivalente al valor del tipo Integer de entrada. Así, dicho valor de tipo Integer representa la posición del spike en la señal portadora, de manera que esta posición, una vez convertida a tipo binario, se almacena tras el anterior valor comprimido en un array del mismo tipo, cuyo número de columnas es igual a la longitud de la palabra. Siguiendo con el ejemplo de la Figura 6, para npc=8, n=3, cada valor deberá ocupar dentro de la palabra de memoria 3 bits. Suponiendo que se desean almacenar los valores comprimidos de 5 señales portadoras en un plataforma de longitud de palabra 8 bit, la hipotética secuencia comprimida quedaría como se muestra en la Figura 7, donde $V_{k,c}$ hace referencia al vector que contiene los k valores comprimidos, siendo k=5 en el ejemplo.

V_{k_c} = [110, 010, 001, 100, 101];

1	1	0	0	1	0	0	0
1	1	0	0	1	0	1	

Figura 7: Ejemplo de implementación del almacenamiento de la extensión de compresión

4 Conclusiones

En este artículo se ha propone una nueva estrategia de compresión de datos de tipo real basada en un nuevo algoritmo de codificación y decodificación de datos analógicos en spikes, PWM-SNN. El algoritmo PWM-SNN permite codificar un spike por cada paso de tiempo, permitiendo así poder hacer reconstrucciones muy precisas de la señal original y afrontar problemas de regresión mediante Redes Neuronales de Impulsos.

Respecto a la propuesta de compresión de este artículo, como posibles trabajos futuros a realizar se deben destacar:

1. Desarrollar una interfaz sencilla que permita un correcto y fácil uso del algoritmo de compresión.
2. Analizar con detalle las capacidades de compresión del algoritmo, comparándolo con otros algoritmos reconocidos en la literatura, como pueden ser los compresores GZIP, FPZIP, ZFP, ZLIB, etc.
3. Realizar una búsqueda de bases de datos para seleccionar las más adecuadas para poder llevar a cabo dicho estudio comparativo (base de datos de imágenes, audio, etc.).

Agradecimientos

Este trabajo ha sido financiado por la Universidad del País Vasco UPV/EHU (GIU19/045) y el Departamento de Educación del Gobierno Vasco (PIBA_2020_1_0008).

English summary

REAL DATA COMPRESSION BASED ON A NOVEL ENCODING ALGORITHM FOR SPIKING NEURAL NETWORKS

Abstract

This article proposes a real data compression approach based on a novel encoding algorithm for Spiking Neural Network, which is inspired by the well-known pulsewidth modulation (PWM). This proposal offers a number of advantages, such as algorithm's simplicity. In this sense, it allows the user to reach a compromise between accuracy and memory consumption by selecting a simple parameter.

Keywords: Spiking Neural Network, encoding, decoding, PWM, compression.

Referencias

[1] H. Al-Bahadili. A novel lossless data compression scheme based on the error correcting Hamming codes. *Computers and Mathematics with Applications*, 56(1):143–150, jul 2008.

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

[3] A. Arriandiaga Laresgoiti. Recurrent Neural Network Based Approach for Estimating the Dynamic Evolution of Grinding Process Variables. Technical report, dec 2016.

[4] M. Burtscher and P. Ratanaworabhan. High throughput compression of double-precision floating-point data. In *Data Compression Conference Proceedings*, pages 293–302, 2007.

https://doi.org/10.17979/spudc.9788497498043.175

[5] W. M. Gerstner, Wulfram; Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity - Wulfram Gerstner, Werner M. Kistler - Google Libros*. Cambridge Edition Press, Cambridge.

[6] R. Guyonneau, R. VanRullen, and S. J. Thorpe. Temporal codes and sparse representations: A key to understanding rapid processing in the visual system. *Journal of Physiology Paris*, 98(4-6 SPEC. ISS.):487–497, jul 2004.

[7] J. Huang, K. Padmanabhan, and O. M. Collins. The sampling theorem with constant amplitude variable width pulses. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(6):1178–1190, 2011.

[8] M. Isenburg, P. Lindstrom, and J. Snoeyink. Lossless compression of predicted floating-point geometry. In *CAD Computer Aided Design*, volume 37, pages 869–877. Elsevier, jul 2005.

[9] J. Kominek, J. Kominek, A. W. Black, and V. Ver. CMU Arctic Databases for Speech Synthesis. 2003.

[10] S. H. Liao. Expert system methodologies and applications-a decade review from 1995 to 2004. *Expert Systems with Applications*, 28(1):93–103, jan 2005.

[11] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, dec 2014.

[12] 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.

[13] M. A. Montemurro, S. Panzeri, M. Maravall, A. Alenda, M. R. Bale, M. Brambilla, and R. S. Petersen. Role of precise spike timing in coding of dynamic vibrissa stimuli in somatosensory thalamus. *Journal of Neurophysiology*, 98(4):1871–1882, oct 2007.

[14] L. U. Perrinet, M. Samuelides, and L. Perrinet. Sparse Image Coding Using an Asynchronous Spiking Neural Network Long-range apparent motion processing in V1 View project Reinforcement and Eye movements View project Sparse Image Coding Using an Asynchronous Spiking Neural Network. Technical report, 2002.

[15] D. Reid, A. J. Hussain, and H. Tawfik. Financial time series prediction using spiking neural networks. *PLoS ONE*, 9(8):e103656, aug 2014.

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

[17] C. E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

https://doi.org/10.17979/spudc.9788497498043.175