ISSN 1870-4069

Implementación de modelos de red neuronal profundos en GPUs embebidas para la clasificación de secuencias de fibrilación auricular

Christian García-Aquino[1], Dante Mújica-Vargas[1], Juan Gabriel González-Serna[1], Manuel Matuz-Cruz[2]

[1] Tecnológico Nacional de México, Centro Nacional de Investigación y Desarrollo Tecnológico, México

[2] Tecnológico Nacional de México, México

m21ce012@cenidet.tecnm.mx

Resumen. En este articulo se implementaron arquitecturas basadas en aprendizaje profundo en tarjetas embebidas. La experimentación se realizó en una tarjeta embebida Nvidia Jetson Nano y se emplearon las señales ECG de la base de datos CinC Challenge 2017 de PhysioNet, de las cuales, se extrajeron características de Tiempo-Frecuencia para el entrenamiento de los modelos. Las modelos de aprendizaje profundo que se emplearon para las tareas de clasificación fue una CNN SqueezeNet y una RNN del tipo BiLSTM. Para cuantificar el rendimiento de clasificación de los modelos, se obtuvo la exactitud y el error promedio por cada clase. En donde se sugiere que el modelo convolucional tiene un menor costo computacional, sin embargo, en tareas de clasificación se sugiere que el modelo recurrente tuvo un mayor rendimiento y una mejor propagación del error frente al modelo convolucional.

Palabras clave: Jetson nano, physionet, squeezenet, BiLSTM, tiempo-frecuencia.

Implementation of Deep Neural Network Models on Embedded GPUs for Classification of Atrial Fibrillation Sequences

Abstract. In this article, architectures based on deep learning in embedded cards were implemented. The experimentation was carried out on an embedded Nvidia Jetson Nano card and ECG signals from the PhysioNet CinC Challenge 2017 database were used, from which Time-Frequency characteristics were extracted for model training.

The deep learning models that were used for the classification tasks were a SqueezeNet CNN and a BiLSTM type RNN. To quantify the classification performance of the models, the accuracy and average error for each class were obtained. Where it is suggested that the convolutional model has a lower computational cost, however, in classification tasks it is suggested that the recursive model had better performance and better error propagation compared to the convolutional model.

Keywords: Jetson nano, physioNet, squeezenet, biLSTM, time-frequency.

1. Introducción

Las Redes Neuronales Artificiales son sistemas de computación basados en un modelo simplificado del cerebro humano. Sus capacidades de clasificación y propiedades como la generalización, tolerancia a fallas y aprendizaje, los hacen atractivos para una basta cantidad de aplicaciónes que dificilmente se resuelven con equipos de computo convencionales [1], [2], [3] y [4]. Además, se utilizan en diversos campos del conocimiento [5], incurriendo a la necesidad de sistemas informáticos cada vez más potentes.

Como es bien sabido, obtener estos sistemas informáticos para ejecutar redes neuronales potentes no es barato, por lo que las alternativas económicas son esenciales para el desarrollo y la investigación. En la literatura, existen enfoques sobre tarjetas de bajo costo, mencionando algunos relevantes en relación al trabajo en cuestión. En [6] emplearon modelos previamente entrenados para clasificar pelotas en escenarios, demostrando tiempos de entrenamiento reducidos, mejor precisión y menor sobreajuste a diferencia de un modelo entrenado desde 0[3].

En [7] se realizó un sistema de reconocimiento de objetos basado en la red neuronal YOLO, implementado en una tarjeta Jetson TK1, consiguiendo detectar y reconocer objetos a 5 FPS. En [8] se propuso la red de detección de cruces peatonales (CDNet) basada en YOLOv5 para la detección rápida y precisa de cruces peatonales, implementada en el dispositivo Jetson Nano con una velocidad de detección de 33 FPS. En [9], se propuso un sistema de detección de carril basado en redes CNN Encoder-Decoder y LSTM, se implementó en una tarjeta NVIDIA Jetson Xavier NX obteniendo puntuaciónes mayores a 90 % en distintas métricas.

Como se puede ver a través de esta sección, se han generado desarrollos empleando tarjetas de desarrollo de bajo costo. Tales dispositivos, han sido de utilidad para la implementación de diferentes modelos neuronales a fin de tener un sistema portátil y de bajo consumo. Sin embargo, en los trabajos mencionados no se puntualiza un análisis del costo computacional en tareas de

[3] Desde 0 se refiere a un modelo que no ha sido entrenado en lo absoluto o que no ha recibido una transferencia de aprendizaje.

clasificación, aunado a que no suponen solución alguna orientada a la clasificación de fibrilación auricular (FA). Por lo tanto, el objetivo de esta investigación es implementar redes neuronales en tarjetas embebidas para realizar la clasificación de secuencias de FA, partiendo del empleo de la tarjeta embebida NVIDIA Jetson Nano. Enfocándose en realizar un análisis de rendimiento en los modelos neuronales, que con el uso de carácteristicas específicas lograr un costo computacional bajo y un alto rendimiento de clasificación.

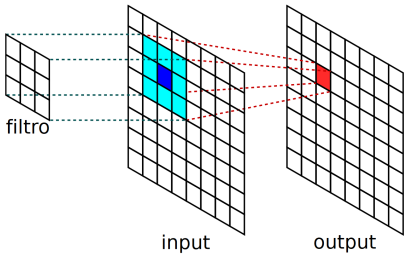

Fig. 1. Proceso de convolución [11].

El resto de este documento está organizado de la siguiente manera. En la sección II, se tiene una breve introducción sobre las Redes Neuronales y las características extraídas de los ECG´s utilizados durante la experimentación. El método de implementación de los modelos de red neuronal se indica en la sección III. Los resultados obtenidos de la experimentación y un análisis comparativo se presentan en la sección IV. Las conclusiones se mencionan en la sección final y se describe el trabajo futuro.

2. Marco teórico

2.1. Redes neuronales

Redes neuronales convolucionales. Las Redes Neuronales Convolucionales (CNN) son, en esencia, redes neuronales que emplean la operación de convolución como una de sus capas. Las CNN se han aplicado a problemas en los que los datos de entrada sobre los que se van a realizar las predicciones tienen una cuadrícula conocida como una serie temporal o una imagen [10]. En el caso de las imágenes, la operación de convolución se da intuitivamente por una entrada $I(m, n)$ y un Kernel $K(a, b)$ la operación de convolución se expresa como:

$$\text{Conv}(t) = \sum_{a} \sum_{b} I(a, b) * K(m - a, n - b), \tag{1}$$

donde m, n y a, b son las dimensiones de una imagen y del kernel de convolución respectivamente. Apreciándose gráficamente el proceso interno que realiza la capa de convolución de una CNN en la Figura 1.

Redes neuronales recurrentes. Las Redes Neuronales Recurrentes (RNN) se emplean para analizar datos que cambian con el tiempo [12]. Para permitir a la red memorizar y acceder a los historiales de entrada, se introducen las conexiones recurrentes para predecir el paso de tiempo actual y transfiriendo esa predicción al siguiente paso como una entrada.

De acuerdo con la Figura 2, un modelo RNN tiene la misma estructura que los modelos clásicos de RNA's, contando con una capa de entrada, n capas ocultas y una de salida, sin olvidar el parámetro t correspondiente al tiempo, siendo x_{t-1}, x_t y x_{t+1} las entradas del modelo RNN en diferentes instantes de tiempo.

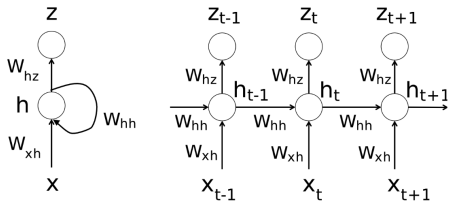

Fig. 2. Estructura básica de una RNN [12].

En la forma más básica de una RNN, se encuentra su función de aprendizaje en la ecuación 2 para las capas ocultas y en la ecuación 3 para la capa de salida:

$$h[t] = f(\, W_x^h(x[t] + b_h) + \, W_h^h(h[t-1] + b_h)), \tag{2}$$

$$y[t] = g(\, W_h^o(h[t] + b_o)). \tag{3}$$

2.2. Transformada de Fourier de tiempo corto

La Transformada de Fourier de Tiempo Corto (STFT) se encarga de analizar señales no estacionarias a través de la Transformada de Fourier (TF). En donde la STFT consiste en dividir la señal en pequeños segmentos de tiempo de tal manera que se pueda asumir que para cada segmento la señal es estacionaria, y así calcular la TF en cada porción de la señal, la cuál es tomada como una ventana que se desliza a lo largo del eje del tiempo, resultando en una representación de dos dimensiones de la señal [13]. Matemáticamente, se escribe como:

$$STFTx(t) = X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t-\tau)\; e^{-j\omega t} dt, \tag{4}$$

donde $w(t)$ es la función ventana de tipo Hann o colina Gaussiana inicializada en 0 y $x(t)$ es la señal de entrada a transfomar, $X(\tau, \omega)$ es en esencia la TF de $x(t)$, $w(t-\tau)$ es una función compleja que representa la fase y la magnitud de la señal sobre el tiempo y la frecuencia.

Concurrentemente la fase instantánea es empleada junto con el eje del tiempo τ y el eje de la frecuencia ω para suprimir cualquier discontinuidad por salto en

la fase resultante en la STFT. El índice de tiempo τ es normalmente considerado como un tiempo "lento" y usualmente no se expresa con tan alta resolución como con el tiempo t.

2.3. Transformada wavelet continua

En [14] se introdujo la Transformada Wavelet Continua (CWT) como una técnica alternativa a la STFT para suprimir el problema de resolución ocasionado por la yuxtaposición de los datos. En relación a eso, la CWT se realiza de manera comparable al análisis STFT, debido a que la señal es multiplicada por una función, en este contexto denominada "Wavelet", siendo análoga a la función de ventaneo de la STFT, y la transformada se calcula separadamente para distintos segmentos de la señal en el dominio del tiempo [15]. Escribiendo la CWT como se describe en la Fig. 3.

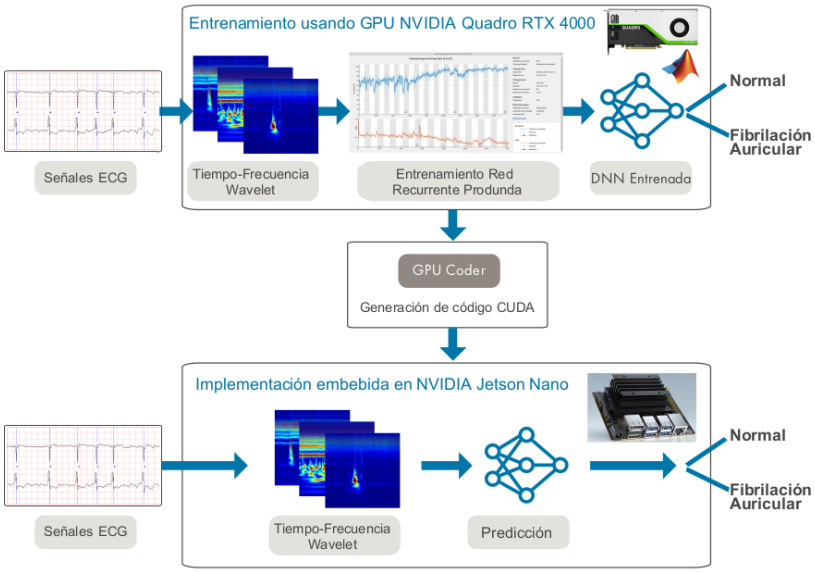

Fig. 3. Implementación.

En donde $f(t)$ es la señal a la cual se aplica la transformada, $\Psi^*_{u,s}(t)$ es la asociación compleja de la función Wavelet $\Psi_{u,s}(t)$ y la señal transformada W, es una función de dos variables, siendo u y s, los parámetros de tiempo y escala respectivamente, que irán cambiando su valor conforme a las iteraciones dentro del análisis:

$$W\ f(u,s) = \int_{-\infty}^{+\infty} f(t)\Psi^*_{u,s}(t)dt. \tag{5}$$

3. Implementación

Es conveniente analizar el proceso de implementación de los modelos de aprendizaje profundo para la clasificación de las secuencias de FA. Comenzando por la preparación de las señales ECG con las que posteriormente se extraerán las características de Tiempo-Frecuencia tanto para el modelo recurrente como para el modelo convolucional. Asímismo, se realiza el entrenamiento de los modelos con dichas carácteristicas y una vez entrenadas, embeber los modelos en una tarjeta electrónica de alto rendimiento para realizar las experimentaciónes. Una descripción de lo anterior puede apreciarse en la Figura 3.

3.1. Señales ECG

Para esta investigación se empleó la base de datos CinC 2017 de PhysioNet, el cuál contiene señales de ECG muestreadas a 300 Hz, organizadas en cuatro clases diferentes: Normal (N), AFib (A), Otro ritmo (O) y Grabación ruidosa, mismos que se consideraron las primeras dos clases para realizar experimentos de clasificación. Se pueden ver más detalles del conjunto de datos en [16].

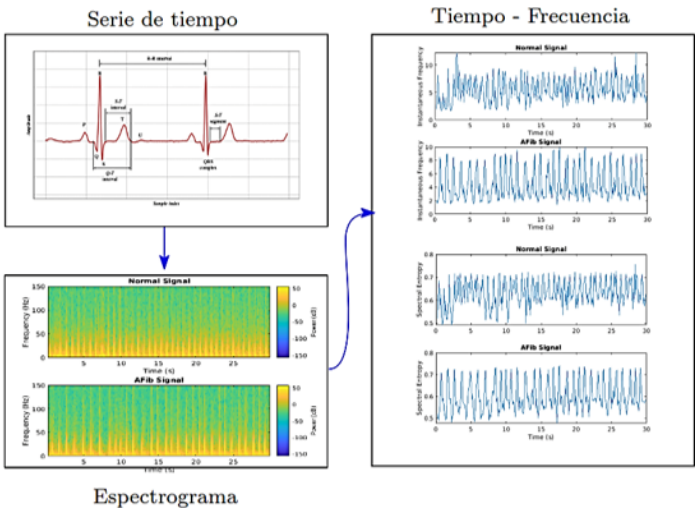

Fig. 4. Extracción de momentos tiempo-frecuencia.

3.2. Entrenamiento de los modelos basados en aprendizaje profundo

Momentos de tiempo-frecuencia. A fin de incrementar el rendimiento de la arquitectura neuronal, es conveniente utilizar características con mayor discriminación. En el área del Procesamiento Digital de Señales (DSP), es común el análisis de éstas en los dominios de frecuencia y tiempo-frecuencia; para ello se deben graficar los espectrogramas de los ECG de latído normal y con FA.

En la Figura 4, se muestran los espectros para señales de muestras en la parte inferior izquierda.

Para calcular el espectrograma, se considera la resolución espectral con la longitud total de la señal. Si es posible, se debe calcular un periodograma de toda la señal utilizando una ventana de Kaiser. De no ser posible, se debe calcular un solo periodograma modificado en un período de tiempo razonable, la función debe calcular un periodograma de Welch dividiendo la señal en segmentos superpuestos, tomando muestras de cada segmento mediante una ventana de Kaiser y promediando los periodogramas de los segmentos.

Una vez obtenidos los espectrogramas, se extraen los momentos de Tiempo-Frecuencia conocidos como la Frecuencia Instantánea y la Entropía Espectral. Empleando el primero para estimar la frecuencia dependiente del tiempo de una señal. La función calcula un espectrograma usando TF de corta duración a lo largo de 255 ventanas de tiempo y las salidas corresponden a los centros de las mismas.

El segundo momento es la Entropía Espectral encargada de medir qué tan plano es el espectro de una señal. Una señal con un espectro puntiagudo, como una suma de sinusoides, tiene una entropía espectral baja. Una señal con un espectro plano, como el ruido blanco, tiene una alta entropía espectral.

Modelo BiLSTM. Por otra parte, se empleó una Red Long Short-Term Memory (LSTM), adecuada para estudiar secuencias de datos y así aprender las dependencias a largo plazo.

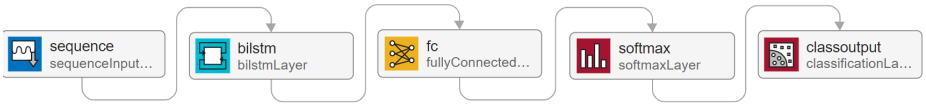

Fig. 5. Red BiLSTM.

La capa LSTM puede mirar la secuencia de tiempo en la dirección de avance, mientras que la capa de LSTM-Bidireccional puede mirar tanto en la dirección de avance como en la de retroceso. En este sentido, la arquitectura neuronal utiliza una capa LSTM-Bidireccional.

Esta arquitectura se puede reinterpretar como un modelo híbrido, ya que considera el procesamiento de una serie de tiempo a través de una red recurrente y una fase de clasificación dada por una red totalmente conectada del tipo Perceptrón Multicapa, con una capa final del tipo Softmax para clasificación multiclase, pudiendo observar la arquitectura completa en la Figura 5.

Representaciones de tiempo-frecuencia. Para las tareas de clasificación de señales ECG empleando CNN, se debe hacer una serie de adecuaciones. En contraste con la arquitectura BiLSTM, no es posible trabajar con series de tiempo.

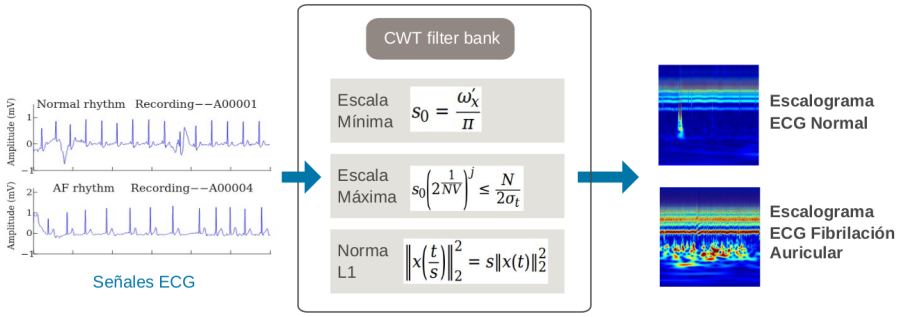

Fig. 6. Extracción de escalogramas.

Fig. 7. Modelo SqueezeNet.

Por lo tanto, las señales ECG deben ser transformadas en una representación 2D, utilizado Wavelets, en particular Bancos de Filtros (FB) de Transformación Wavelets Continuas (CWT).

El precálculo del (FB) es el método preferido cuando se obtiene el CWT de muchas señales usando los mismos parámetros, considerando varias series de tiempo ECG. Las escalas mínima y máxima se determinan automáticamente en función de la propagación de energía de la wavelet. Si x tiene un valor real, WT es una matriz 2D donde cada lado corresponde a una escala. El tamaño de la columna de WT es igual a la longitud de x.

Si x tiene un valor complejo, WT es una matriz 3-D, donde la primera página es el CWT para las escalas positivas y la segunda página es el CWT para las

escalas negativas. La función CWT usa la normalización L1, con lo cual, si se tienen componentes oscilatorios de igual amplitud en sus datos a diferentes escalas, se tendrán la misma magnitud en el CWT.

Para ejemplificar esta transformación se puede observar la Figura 6. La imagen permite ilustrar los procesos que se realizan con una serie de tiempo sana y otra con la presencia de fibrilación auricular.

Modelo SqueezeNet. SqueezeNet (Fig. 7) es una CNN profunda originalmente diseñada para clasificar imágenes en 1000 categorías. En el presente trabajo de investigación, se reutilizó esta arquitectura de red CNN para clasificar las señales de ECG en función de las imágenes del CWT de los datos de series de tiempo. Cada capa de la arquitectura de una CNN se considera un filtro.

Las capas iniciales identifican características más comunes de las imágenes, como manchas, bordes y colores. Las capas posteriores se centran en características más específicas para diferenciar categorías. SqueezNet está capacitada para clasificar imágenes en 1000 categorías de objetos. Para utilizar esta red en el problema de clasificación de señales ECG, resulta obligatorio volver a entrenar la red.

En las CNN se puede explotar el concepto de transferencia de conocimiento, lo cual sugiere que para reentrenar una red en otra tarea de clasificación basta con ajustar las últimas capas (fase de predicción), ya que las capas de convolución sirven para extraer características, y éstas son genericas para cualquier tipo de imagen. Por lo que se analizaron las últimas 6 capas y se realizó la modificación de la capa "conv10"de 1×1 con una nueva capa convolucional con el número de filtros proporcionalmente al número de clases con los que se trabajó.

Entrenamiento de los modelos. Dado que las arquitecturas basadas en Aprendizaje Profundo requieren grandes cantidades de información en la fase de entrenamiento, el tiempo requerido para esta tarea puede requerir una gran cantidad de horas. Fue necesario solventar esta problemática con tarjetas aceleradoras, para este proyecto se empleó una GPU Quadro RTX 4000 con 2304 núcleos CUDA. Con este dispositivose realizaron las fases de entrenamiento y validación de las redes.

En donde el modelo BiLSTM requirió un tiempo de entrenamiento de 2 minutos con un total de 30 épocas y 1770 iteraciones, mientras que el modelo convolucional requirió un tiempo de entrenamiento de 41 segundos con un total de 40 épocas y 320 iteraciones.

Implementación en la tarjeta embebida. Para comunicarse con el hardware embebido de NVIDIA, se creó un objeto de conexión de hardware. Posterior a la creación de la conexión de la tarjeta embebida con MATLAB, se hizo uso de GPU Coder para la generación de código CUDA. Finalmente, una vez que el procedimiento fue exitoso, todos los archivos generados son copiados en la tarjeta electrónica, resaltando el archivo con extensión elf, ya que es el ejecutable de todo el sistema de clasificación desarrollado.

Tabla 1. Exactitud obtenida por clase y modelo.

Muestra	ecgN RNN	ecgAF RNN	ecgN CNN	ecgAF CNN
1	0.910	0.851	0.893	0.848
2	0.893	0.997	0.869	0.979
3	0.943	0.959	0.906	0.952
4	0.920	0.935	0.901	0.934
5	0.975	0.961	0.907	0.923
6	0.923	0.912	0.908	0.895
7	0.889	0.918	0.871	0.880
8	0.899	0.994	0.886	0.963
9	0.905	0.854	0.887	0.854
10	0.942	0.895	0.907	0.868
11	0.897	0.904	0.872	0.876
12	0.957	0.971	0.884	0.945
13	0.932	0.872	0.897	0.850
14	0.904	0.837	0.889	0.828
15	0.967	0.869	0.895	0.838
16	0.989	0.845	0.911	0.835
17	0.938	0.878	0.903	0.863
18	0.917	0.920	0.909	0.885
19	0.926	0.968	0.905	0.933
20	0.935	0.913	0.903	0.897
Promedio	0.928	0.913	0.895	0.892

4. Experimentación y resultados

En este experimento se evaluó el rendimiento de ambos modelos neuronales en términos de la métrica de Exactitud, así como del Error en la predicción de las clases. Puesto que los modelos fueron adecuados para una clasificación binaria, solo se consideran las clases: a) Ritmo Normal (N) y b) Ritmo con FA (AF). Considerando 20 señales de muestra para cada clase. Para lo cuál, se asumen las siguientes nomenclaturas:

- **ecgN_RNN y ecgAF_RNN**: Representa la predicción de la Red Neuronal Recurrente entrenada con los momentos de Tiempo-Frecuencia para la clase ritmo cardíaco normal y fibrilación auricular.
- **ecgN_CNN y ecgAF_CNN**: Representa la predicción de la Red Neuronal Convolucional entrenada con los escalogramas de Tiempo-Frecuencia para la clase ritmo cardíaco normal y fibrilación auricular.

En la Tabla 1 se presenta de manera resumida la exactitud de predicción de ambos modelos neuronales. Se destaca una eficiencia promedio de la red neuronal recurrente del 92.8 % para la clase ritmo cardíaco normal y un 91.3 % para la clase ritmo cardíaco con fibrilación auricular. Mientras que para el modelo convolucional un 89.5 % y 89.2 %, respectivamente.

Gráficamente estos resultados pueden ser visualizados en la Figura 8. Es posible observar aspectos como los valores promedio, mínimo, máximo, así como la dispersión de los resultados. De manera rápida se puede visualizar que ambos modelos neuronales presentaron una menor varianza en los resultados para la clase ritmo cardíaco sano, en comparación del ritmo cardíaco con fibrilación auricular.

De manera complementaria, en la Tabla 2 se presenta el Error en la predicción de los modelos neuronales profundos. Los valores promedio sugieren una menor

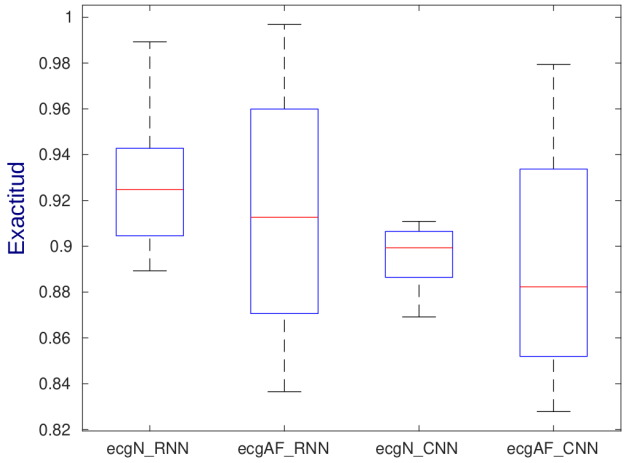

Fig. 8. Exactitud promedio por clase y modelo.

Tabla 2. Error obtenido por clase y modelo.

Muestra	ecgN RNN	ecgAF RNN	ecgN CNN	ecgAF CNN
1	0.090	0.149	0.107	0.152
2	0.107	0.003	0.131	0.021
3	0.057	0.041	0.094	0.048
4	0.080	0.065	0.099	0.066
5	0.025	0.039	0.093	0.077
6	0.077	0.088	0.092	0.105
7	0.111	0.082	0.129	0.120
8	0.101	0.006	0.114	0.037
9	0.095	0.146	0.113	0.146
10	0.058	0.105	0.093	0.132
11	0.103	0.096	0.128	0.124
12	0.043	0.029	0.116	0.055
13	0.068	0.128	0.103	0.150
14	0.096	0.163	0.111	0.172
15	0.033	0.131	0.105	0.162
16	0.011	0.155	0.089	0.165
17	0.062	0.122	0.097	0.137
18	0.083	0.080	0.091	0.115
19	0.074	0.032	0.095	0.067
20	0.065	0.087	0.097	0.103
Promedio	0.072	0.087	0.105	0.108

de predicción realizada por la RNN, en comparación con el modelo convolucional. De igual manera, las tendencias gráficas se pueden visualizar en la Figura 9.

Tiempo de procesamiento. El segundo experimento nos permitió conocer el tiempo de procesamiento promedio requerido en la fase de predicción de ambos modelos, a partir de las 20 instancias de muestra. Se consideraron 3 dispositivos distintos:

- **CPU** Intel Core i7-8750 con 12 núcleos a 4.1GHz.
- **GPU** Nvidia Quadro RTX 4000 con 2304 núcleos CUDA.
- **Tarjeta Embebida** Nvidia Jetson Nano con 128 núcleos CUDA.

Tabla 3. Costo computacional.

Dispositivo	Red Neuronal Recurrente	Red Neuronal Convolucional
CPU	0.769 s	0.128 s
GPU	0.061 s	0.010 s
Tarjeta Embebida	21.487 s	15.093 s

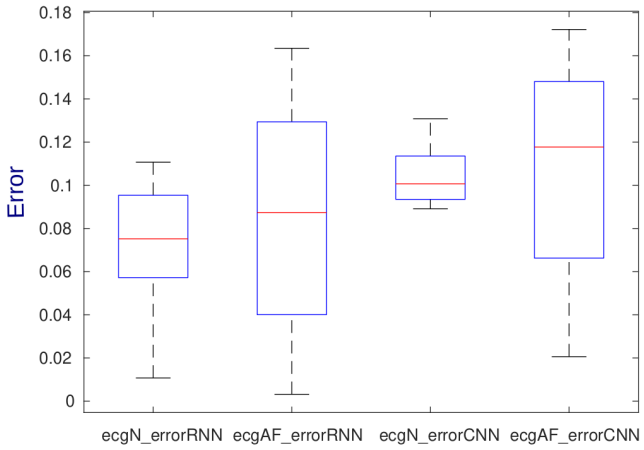

Fig. 9. Error promedio por clase y modelo.

En la Tabla 3 se presenta de manera resumida el costo computacional medido en segundos de los 3 dispositivos utilizados en la experimentación, en donde la información es indiferente a una clase en particular. El modelo recurrente requiere mayor tiempo de procesamiento, independientemente del dispositivo utilizado. Esto se debe a que este tipo de arquitecturas trabaja con información en el tiempo, lo cual es un aspecto importante, ya que permite garantizar un mejor rendimiento.

Por ende, fue de interés para esta investigación saber cuál era el tiempo de procesamiento de estos modelos neuronales en un dispositivo embebido de alto rendimiento. Pudiendo observar de la tabla que en ambos casos es superior a los 15 segundos para realizar una predicción, este tiempo se considera aceptable por la cantidad de procesamiento que se realiza en esta fase.

5. Conclusiones

Los resultados obtenidos en este experimento resaltan aspectos que influyen en el rendimiento, ya que emplear series de tiempo en crudo no garantiza un rendimiento de predicción aceptable, lo cuál no útil al realizar prototipos de manera física. Por ende, el uso del procesamiento digital de señales es esencial en las tareas del procesamiento de series de tiempo de las señales ECG.

Como se observó, se consideraron dos variantes de modelos neuronales. Para la arquitectura recurrente, se obtuvieron espectrogramas de frecuencia para ambos tipos de señales; sin embargo, estas eran imágenes las cuales no podían ser utilizadas por la red recurrente, así que a partir de los espectrogramas se obtuvieron los momentos Tiempo-Frecuencia (Frecuencia Instantánea y Entropía Espectral).

Estas características son series de tiempo que concentran la información más importantes de las señales ECG y permitieron acelerar el entrenamiento de la red recurrente, e incrementar su rendimiento a más del 92 %. Por otra parte, para la arquitectura convolucional bastó con transformar la señales ECG en Escalogramas de Frecuencia, esta transformación 1D en 2D permitió utilizar esta característica para hacer el entrenamiento de la red. Garantizando un rendimiento superior al 89 %.

Asumiendo que se podría obtener un mejor rendimiento si se recurrieran a técnicas como el aumento información, para mejorar su rendimiento, aunado a un mayor estudio de características significativas posibles que se puedan extraer de datos crudos de los ECG. Como trabajo futuro se tiene la tarea de mejorar los tiempos de procesamiento a fin de concebir un sistema que tenga la capacidad de trabajar los más cercano al tiempo real, de manera portátil y de manera remota.

Agradecimientos. Este trabajo fue apoyado por el Tecnológico Nacional de México/CENIDET a través del proyecto “Delimitación de masas sólidas malignas en mamografías mediante un algoritmo de nodos conectados con el menor angulo polar”, así como por el CONACYT.

Referencias

1. Curran, K., Li, X., McCaughley, N.: Neural network face detection. The Imaging Science Journal, vol. 53, no. 2, pp. 105–115 (2005)
2. Patil, V., Shimpi, S.: Handwritten english character recognition using neural network. Elixir Comput Sci Eng, vol. 41, pp. 5587–5591 (2011)
3. Bartlett, P. L.: The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE transactions on Information Theory, vol. 44, no. 2, pp. 525–536 (1998)
4. Åkesson, B. M., Toivonen, H. T.: A neural network model predictive controller. Journal of Process Control, vol. 16, no. 9, pp. 937–946 (2006)
5. Timotheou, S.: The random neural network: A survey. The computer journal, vol. 53, no. 3, pp. 251–267 (2010)
6. Holz, D., Genter, K., Saad, M.: RoboCup 2018: Robot world cup XXII. Lecture Notes in Computer Science (2019)
7. Reyes, E., Gómez, C., Norambuena, E., Ruiz-del-Solar, J.: Near real-time object recognition for pepper based on deep neural networks running on a backpack. In: Holz, D., Genter, K., Saad, M., von Stryk, RoboCup 2018: Robot World Cup XXII. RoboCup 2018. Lecture Notes in Computer Science, vol 11374. Springer, Cham (2019)

8. Zhang, Z. D., Tan, M. L., Lan, Z. C., et al.: CDNet: A real-time and robust crosswalk detection network on Jetson nano based on YOLOv5. Neural Comput and Applic (2022)
9. Kortli, Y., Gabsi, S., Lew, F. C., Voon, L. Y., Jridi, M., Merzougui, M., Atri, M.: Deep embedded hybrid CNN–LSTM network for lane detection on NVIDIA Jetson Xavier NX. Knowledge-Based Systems, vol. 240 (2022)
10. Ketkar, N.: Convolutional neural networks. Deep Learning with Python. pp. 63–78 (2017)
11. Redolfi, J.: Aplicación en agricultura de precisión de esquemas actuales de reconocimiento visual (2018)
12. Vogt, N.: CNNs, LSTMs, and attention networks for pathology detection in medical data (2019)
13. Smith, J. O.: Mathematics of the discrete Fourier transform (DFT). W3K Publishing (2007)
14. Morlet, J., Arens, G., Fourgeau, E., Giard, D.: Wave propagation and sampling theory—part II: sampling theory and complex waves. GEOPHYSICS, vol. 47, no. 2, pp. 222–236 (1982)
15. Bolós, V.J., Benítez, R.: The wavelet scalogram in the study of time series. In: Advances in Differential Equations and Applications. SEMA SIMAI Springer Series, vol. 4. Springer, Cham (2014)
16. Clifford, G. D., Liu, C., Moody, B., Li-wei, H. L., Silva, I., Li, Q., Johnson, A. E., Mark R. G.: AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017. In: 2017 Computing in Cardiology (CinC) pp. 1–4 (2017)