Redes Neuronales Artificiales en el Reconocimiento de Señales Electromiográficas.

1st Héctor Emmanuel Munguía Estrada
Centro Universitario de Los Lagos, Universidad de Guadalajara,
Lagos de Moreno, Jalisco, México
hector.munguia8200@alumnos.udg.mx

2nd Borislav Zaynullin Pacheco
Centro Universitario de Los Lagos, Universidad de Guadalajara,
Lagos de Moreno, Jalisco, México
borislav.zaynullin3747@alumnos.udg.mx

3rd **Gregorio Alejandro Oropeza-Gomez**
Centro Universitario de Los Lagos, Universidad de Guadalajara,
Lagos de Moreno, Jalisco, México
gregorio.oropeza@alumnos.udg.mx

4th J. Onofre Orozco-López
Centro Universitario de Los Lagos, Universidad de Guadalajara,
Lagos de Moreno, Jalisco, México
juan.onofre@academicos.udg.mx

5th **Miguel Mora-González**
Centro Universitario de Los Lagos, Universidad de Guadalajara,
Lagos de Moreno, Jalisco, México
miguel.mora@academicos.udg.mx

Resumen – **En el presente documento, se describe una forma de creación de redes neuronales convolucionales y redes neuronales artificiales en el reconocimiento de señales electromiografías para su posible aplicación en el desarrollo de prótesis.**

Palabras clave – Red Neuronal Convolucional (Convolutional Neural Network), Red Neuronal Artificial (Artificial Neural Network), Sensor Electromiográfico.

I. Introducción.

Las redes neuronales inspiradas en el funcionamiento del cerebro humano, desde su concepción hasta la fecha han recorrido un largo camino de modificaciones y mejoras Llegando a hacer herramientas tan complejas y útiles como lo son hoy en día, Por su parte los sensores electromiográficos también han evolucionado y pasado de ser grandes y costosos a compactos y accesibles, permitiendo mejoras en diversos campos de estudio. Gracias a la mejora continua de estas dos tecnologías hoy en día se trabaja en la creación de prótesis más precisas y funcionales.

II. Objetivo.

Se desea obtener un modelo que dadas las señales adquiridas mediante el sensor electromiográfico la tarjeta del microcontrolador Arduino UNO sea capas de detectar movimientos musculares.

III. Materiales y Metodos

A. *Montaje del circuito para el experimento*

Para el montaje del circuito [1] se utilizó el sensor electromiográfico AD 82226, dos pilas de 9 voltios y una placa Arduino UNO, como primer paso se conectaron ambas pilas en serie para poder obtener +9v y -9v, posteriormente se conectó el GND de la batería al GND del Arduino y al GND de la del sensor por último se conectó la salida del sensor a una de las entradas analógicas de Arduino.

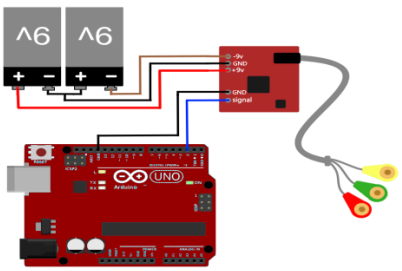

Fig. 1. Circuito del experimento.

B. *Creación de scripts en Arduino y Python para la toma de muestras.*

Para la obtención de muestras se crearon dos scripts uno en Python y otro en Arduino, siendo el código de Arduino el más simple de los dos, ya que este solo leía el valor del sensor y lo imprimía en el puerto serial. por su parte el código desarrollado en Python leía dicho valor del puerto serial y lo almacenaba en una lista, una vez que esa lista llegaba a un número *N* de elementos, se guardaba en un mismo CSV de manera asíncrona hasta que el CSV tuviera un total de *N* datos dispuestos en una sola columna.

C. Recopilación de muestras.

Para la obtención de las muestras los 3 electrodos de los que dispone el sensor (Rojo, Verde, Amarillo) se conectaron de la siguiente manera: el electrodo rojo se colocó en la mitad del bíceps extendido, en la parte interna del mismo, el electrodo verde se colocó en la parte final de la parte interna del bíceps, zona cercana al doblez del codo y por último el electrodo amarillo se colocó en el hueso del codo. Las muestras se obtuvieron de 2 sujetos diferentes, siendo un total de 20 muestras por sujeto 10 de cada brazo, en donde cada muestra se tomaba durante un periodo de aproximadamente 2 minutos, obteniendo un total de 200000 mediciones por cada muestra, el movimiento ejecutado fue una flexión de bíceps sin peso, variando la fuerza aplicada para hacer dicho movimiento.

D. Suavizado de la señal para su posterior clasificación.

A cada una de las señales obtenidas se les aplica un suavizado, que consiste en promediar *N* número de mediciones y tomarlas como un solo valor de una nueva señal, quedando la señal suavizada con un menor número de elementos que la original, en un factor de 1:*N*, pero con un menor ruido como se muestra en la siguiente porción de la gráfica.

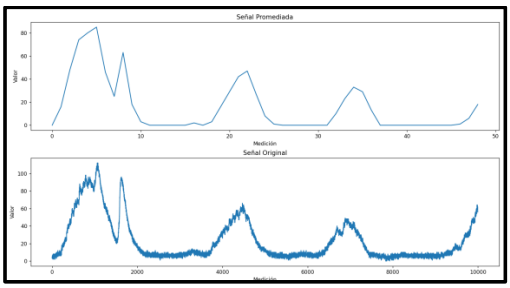

Fig. 2. Suavizado de la señal.

E. Creación de la señal de clasificación.

Para seleccionar cuales de las señales son picos de activación y cuales con solo ruido se optó por seleccionar un umbral el cual se seleccionó en base al ruido que el sensor genera cuando no se está haciendo ningún movimiento, y en base a este umbral y a la señal suavizada se sobrepuso una señal cuadrada, que en su valor alto engloba a todo el pico que se detecta como una activación del músculo, guardando este nuevo valor como una nueva columna de cada archivo de cada muestra, quedando la señal como se muestra en la siguiente porción de una muestra.

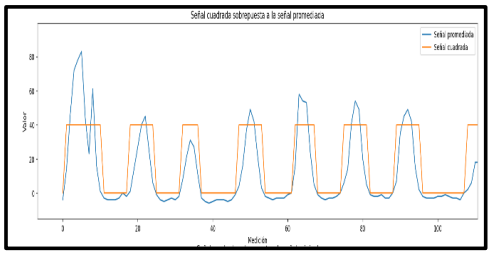

Fig. 3. Creación de la señal clasificada.

F. Sobreposición de la señal cuadrada a la señal original y supresión de ruido residual.

Una vez obtenida la señal cuadrada se procedió a expandirla, de tal manera que la señal cuadrada quedara correctamente sobrepuesta sobre la señal original, al hacerlo primero en la señal suavizada permite obtener una señal cuadrada que engloba correctamente los picos de activación que muestran las gráficas y eliminar de manera más rápida y sencilla casi todo el ruido que se generó en la señal de clasificación. Para eliminar el ruido residual se creó un programa en el cual se seleccionaba sobre la gráfica el intervalo que se quería fijar la señal cuadrada (señal de activación) en su valor alto, o en 0, recorriendo así toda la gráfica y eliminando todo el ruido residual.

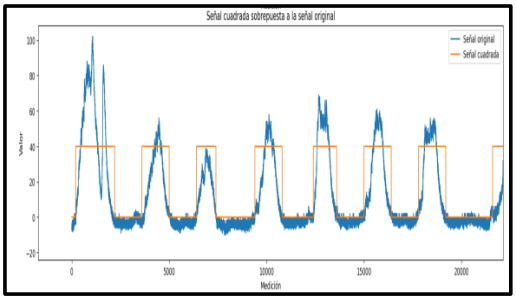

Fig. 4. Sobreposición y supresión de la señal.

IV. Resultados

A. *Creación de la señal de clasificación.*

Para unir todas las muestras del mismo sujeto en un solo archivo, discriminando de que brazo son ya que no hubo gran diferencia en las señales de un brazo a otro, se procedió a unir todas las columnas de todos los CSV de cada sujeto en una sola columna, posteriormente con la ayuda de un programa que se creó en Python, de manera similar a como se eliminó el ruido residual, se copiaban intervalos de la señal y se seleccionaba dónde colocarlos, esto con la intención de tener uniones definidas y no transiciones agresivas entre una muestra a otra.

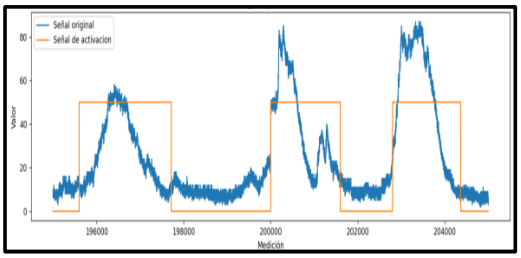

Fig. 5. Unión de las muestras.

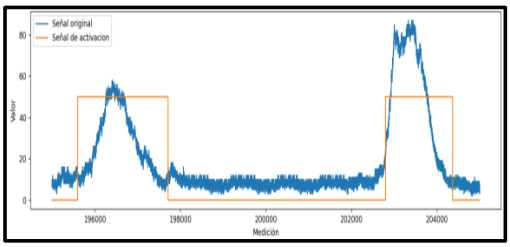

Fig. 6. Unión de las muestras.

B. *Creación del prototipo de Red Neuronal artificial.*

La estructura de la red neuronal consiste en un modelo secuencial con un total de 22 capas en total [2], [3], una capa para la salida, otra para la entrada y 20 capas ocultas, en donde el orden de neuronas fue el siguiente:

```
self.model = Sequential()
self.model.add(Input(shape=(input_size,)))
self.model.add(Dense(1900,activation='relu'))
self.model.add(Dense(1800,activation='relu'))
self.model.add(Dense(1700,activation='relu'))
self.model.add(Dense(1600,activation='relu'))
self.model.add(Dense(1500,activation='relu'))
self.model.add(Dense(1400,activation='relu'))
self.model.add(Dense(1200,activation='relu'))
self.model.add(Dense(1300,activation='relu'))
self.model.add(Dense(1100,activation='relu'))
self.model.add(Dense(1000,activation='relu'))
self.model.add(Dense(1000,activation='relu'))
self.model.add(Dense(1100,activation='relu'))
self.model.add(Dense(1200,activation='relu'))
self.model.add(Dense(1300,activation='relu'))
self.model.add(Dense(1400,activation='relu'))
self.model.add(Dense(1500,activation='relu'))
self.model.add(Dense(1600,activation='relu'))
self.model.add(Dense(1700,activation='relu'))
self.model.add(Dense(1800,activation='relu'))
self.model.add(Dense(1900,activation='relu'))
self.model.add(Dense(output_size,
                     activation='sigmoid'
                     ))
```

Además, cabe mencionar que en todas las capas excepto en la de entrada y salida se usó la función de actuación “relu” y en la capa de salida se utilizó la fusión “sigmoid”.

C. *Creación de imágenes para la CNN.*

Para la creación de la red neuronal convolucional es necesario crear imágenes las cuales serán de utilidad para mejorar el aprendizaje de nuestra IA[4], [5].

Las imágenes se elaboraron a través de un script de Python, en el cual clasificamos los intervalos de la señal por la ubicación de un pico, con un total de 16 clases y posteriormente clasificar en intervalos de 1000, 1500, 1800 y 2000 datos.

D. *Creación de imágenes para la CNN.*

La estructura de la red neuronal convolucional [6], [7]cuenta con un parámetro para identificar la ubicación de los picos de la señal, para el prototipo se utilizó la siguiente estructura:

```
self.model = Sequential([
            layers.Rescaling(1. / 255,
input_shape=(self.img_height, self.img_width,
3)),
            layers.Conv2D(16, 3, padding='same',
activation='relu'),
            layers.MaxPooling2D(),
            layers.Conv2D(32, 3, padding='same',
activation='relu'),
            layers.MaxPooling2D(),
            layers.Conv2D(64, 3, padding='same',
activation='relu'),
            layers.MaxPooling2D(),
            layers.Flatten(),
            layers.Dense(64, activation='relu'),
            layers.Dense(num_classes,
activation='softmax')
        ])
```

Solo se usaron las imágenes con un intervalo de 1000 datos obteniendo una precisión del 60%.

E. *Creación de imágenes para la CNN.*

Para el refinamiento [7] de la red neuronal se utilizó la siguiente estructura:

```
layers.Rescaling(
1./255,input_shape=(self.img_height,
                    self.img_width, 3)),
layers.Conv2D(16,3,padding='same',
                activation='relu'),
layers.MaxPooling2D(),
              layers.Conv2D(32,3,padding='same',
                activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(64,3,padding='same',
                activation='relu'),
layers.MaxPooling2D(),
layers.Flatten(),
layers.Dense(64, activation='relu'),
layers.Dense(num_classes, activation='softmax')
```

Este modelo es muy similar al anterior, lo que difiere del anterior modelo es el parámetro modificado en la compilación, el tipo de perdida:

```
Keras.loss = ‘categorical_crossentropy’
```

Además de que se utilizó una base de datos expandida, logrando una precisión del 96%.

V. Conclusiones.

Como resultado de la investigación, las redes neuronales artificiales (ANN) son inefectivas en comparación a las redes neuronales convolucionales (CNN), debido al extenso tamaño de nuestra muestra, no solo el tiempo de aprendizaje se extiende, sino que también el porcentaje de aprendizaje se mantiene bajo y no varía para valores altos. Y a su vez genera una mayor cantidad de pérdidas registradas. Por el contrario, la red neuronal convolucional mostró mejores resultados, mediante el uso de imágenes generadas a través de las gráficas de intervalos de la señal, lo cual redujo la muestra para ser mejor controlada por la IA. Por

lo que, aprendió con mayor rapidez y alcanzó una mayor precisión de aprendizaje. El alcance de la investigación está influenciado por el tamaño de la muestra aplicada, y debido a ello es preferible el uso de la red neuronal convolucional (CNN).

En conclusión, la buena comprensión de la teoría de las redes neuronales es necesaria para producir mejores resultados y evitar la pérdida de recursos y tiempo. Principalmente es necesario comprender cuál es el mejor aproche de aprendizaje dependiendo del tamaño de la muestra y su complejidad, la dificultad y el alcance de la tarea a realizar. Y aún con todo, el entorno donde la IA es generada no solo afecta a su aprendizaje, sino que también a su diseño. Por lo que, el camino a seguir es utilizar un sensor EMG avanzado para la recopilación de muestras más diversas y complejas, e aplicar una CNN para obtener los resultados esperados.

VI. Referencias.

[1] Ijcsis, J. o. C. S., Zamudio, V. M., Mora-Gonzalez, M., & Díaz, M. S. G. (2018). Detection of Physiological Changes in Women during Muscle Activity Using Electromyography and Other Techniques. www.academia.edu.

[2] Chakraverty, S., & Mall, S. (2017). Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations. CRC Press.

[3] Haykin, S. (1998). Neural Networks: a comprehensive foundation. Pearson Education.

[4] Priddy, K. L., & Keller, P. E. (2005). Artificial Neural Networks: An Introduction. SPIE PRESS.

[5] Sarangapani, J. (2018). Neural Network Control of Nonlinear Discrete-Time Systems. CRC Press.

[6] Y. Lu, Y. Chen, D. Zhao, B. Liu, Z. Lai and J. Chen, "CNN-G: Convolutional Neural Network Combined With Graph for Image Segmentation With Theoretical Analysis," in *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 3, pp. 631-644, Sept. 2021, doi: 10.1109/TCDS.2020.2998497

[7] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536.