

---

---

# Universidad Autónoma de Querétaro

Facultad de Ingeniería

Propuesta de red neuronal convolutiva para la clasificación de señales electromiográficas provenientes de movimientos de la mano.

Presentada por:

Axel Aarón Luna Quiñones

Que como parte de los requisitos para obtener el Grado de:  
Ingeniero Físico

Dr. Marco-Antonio Aceves Fernández

Presidente

Dr. Saúl Tovar

Secretario

Dr. Alberto Hernández Almada

Vocal

Centro Universitario, Querétaro, Qro.

2022



Dirección General de Bibliotecas y Servicios Digitales de  
Información



Propuesta de red neuronal convolutiva para la  
clasificación de señales electromiográficas provenientes  
de movimientos de la mano.

**por**

Axel Aarón Luna Quiñones

se distribuye bajo una [Licencia Creative Commons  
Atribución-NoComercial-SinDerivadas 4.0 Internacional](#).

**Clave RI:** IGLIN-272472-0323-222

# Agradecimientos

A mis padres por el apoyo brindado durante este proceso.

A mi tutor y profesores por el tiempo dedicado y los conocimientos brindados.

A todos los familiares y amigos que estuvieron conmigo durante esta etapa.

Dedicado a mi familia.

# Índice

<b>1</b>	<b>Introducción</b>	<b>11</b>
1.1	Planteamiento del objeto de estudio	11
1.2	Justificación	14
1.3	Hipótesis	15
1.4	Objetivos previstos	16
1.4.1	Objetivo general	16
1.4.2	Objetivos Particulares	16
<b>2</b>	<b>Antecedentes</b>	<b>17</b>
2.1	Señales electromiográficas	17
2.1.1	Unidades motoras	17
2.1.2	Generación y medición de las señales	18
2.1.3	Ruido en las señales electromiográficas	20
2.1.4	Método Wavelet para reducción de ruido	20
2.1.5	Banco de señales	24
2.2	Redes Neuronales Convolucionales	30
2.2.1	Concepto general de redes neuronales	30
2.2.2	Neuronas	31
2.2.3	Capas de la red	32
2.2.4	Funciones de activación	34
2.2.5	Función sigmoide	34
2.2.6	Función softmax	36
2.2.7	Función ReLU	37
2.3	Entrenamiento de la red neuronal	38

2.3.1	Función de error Cross-entropy	39
2.3.2	Gradiente descendente	40
2.3.3	Back propagation	43
2.3.4	Optimizador ADAM	45
2.4	Proceso de Regresión Gaussiano (GPR)	47
<b>3</b>	<b>Metodología utilizada</b>	<b>50</b>
3.1	Análisis del banco de señales	50
3.1.1	Análisis cualitativo de los canales de cada movimiento	50
3.1.2	Valores estadísticos de la base de datos	53
3.2	Pre-procesamiento del banco de señales	56
3.2.1	Segmentación de las señales	56
3.2.2	Reescalado de la base de datos	57
3.3	Propuesta de red neuronal convolucional	61
3.3.1	Arquitectura de la red neuronal convolucional	61
3.3.2	Entrenamiento de la red neuronal convolucional	64
3.3.3	Optimización de los hiperparámetros de la red	67
<b>4</b>	<b>Resultados y discusión</b>	<b>71</b>
4.1	Valores óptimos de los hiperparámetros	71
4.2	Comparación de los diferentes métodos de optimización de los valores de activación	72
4.3	Comparación de resultados	74
4.4	Métricas de calidad	74
4.5	Posibles aplicaciones	76
4.6	Uso del proyecto	76

**5 Conclusiones**

77

**6 Anexos**

83

## Índice de cuadros

1	Coefficientes usados en la normalización de la base de datos.	54
2	Hiperparámetros del modelo de red neuronal convolucional	63
3	Tamaño de los conjuntos de datos.	66
4	Valores constantes de los hiperparámetros	70
5	Valores óptimos de los hiperparámetros	71

# Índice de figuras

1	Representación esquemática de una unidad motora con n fibras musculares. Recuperado de [1]. . . . .	12
2	Ejemplo de electromiografía de superficie. Recuperado de [24]. . . . .	19
3	Esquema de la trasformada de wavelet. Recuperado de [4]. . . . .	21
4	Proceso de filtrado de una señal para la extracción de características, cada par de filtros en un determinado nivel, genera los coeficientes que serán pasados al siguiente nivel. . . . .	24
5	Lista de movimientos realizados por cada sujeto. (a) Posición inicial, (b) pronación, (c) supinación, (d) extensión de la muñeca, (e) flexión de la muñeca, (f) desviación cubital de la muñeca, (g) desviación radial de la muñeca, (h) recolección manual, (i) mano cerrada, (j) mano abierta. Tomado de [22]. . . . .	27
6	Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano en la posición inicial (7a). Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano realizando el movimiento de pronación (7b). Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano realizando el movimiento de extensión (7c). Señal correspondiente al canal 2 de la prueba realizada a un sujeto con la mano realizando el movimiento de extensión (7d). . . . .	29
7	Diagrama de los elementos de una neurona. Recuperado de [29]. . . . .	31
8	Representación gráfica de una red neuronal multicapa. Recuperado de [31]. . . . .	33
9	Diagrama del dropout de una capa a otra. Recuperado de [32]. . . . .	34
10	Diagrama de la función sigmoide. Recuperado de [33]. . . . .	35
11	Diagrama de la función softmax. Recuperado de [35]. . . . .	37

12	Diagrama de la función RELU. Recuperado de [36]	38
13	Representación gráfica del gradiente descendente. Recuperado de So-	
	taquirá, Miguel. (Julio de 2018). ¿Qué es el Gradiente Descendente?.	
	codificandobits.com	41
14	Diagrama de ejemplo para un proceso de regresión gaussiano. Recu-	
	perado de [41].	48
15	Señales de los ocho canales producidas por el movimiento de posición	
	inicial.	51
16	Señales de los ocho canales producidas por el movimiento de pronación.	52
17	Señales de los ocho canales producidas por el movimiento de pinza fina.	53
18	Media y desviación estándar por movimiento.	55
19	Media y desviación estándar por canal.	56
20	Ejemplo de la sección de la señal donde ocurre un evento. Recuperado	
	de [24]	57
21	Comparación entre la señal generada en el canal 3 del movimiento de	
	desviación radial antes y después de ser normalizada.	59
22	Comparación entre la señal generada en el canal 3 del movimiento de	
	desviación radial antes y después de ser reescalada.	60
23	Diagrama de capa de dropout. Recuperado de [29].	62
24	Diagrama de la red neuronal convolucional.	64
25	Diagrama de flujo del algoritmo de entrenamiento de la red neuronal	
	convolucional.	65
26	Proceso de gaussiano con la tasa de aprendizaje como variable.	68
27	Proceso de gaussiano con el tamaño del batch como variante.	69
28	Proceso de gaussiano con el valor de dropout como variable.	69
29	Proceso de gaussiano con el valor de división del conjunto de valida-	
	ción como variable.	70

30	Exactitud promedio del modelo.	72
31	Error promedio del modelo.	73
32	Error del modelo cuando se usa el método de optimización del gra-	
	diente descendiente.	74
33	Varianza en la probabilidad predicha por movimiento.	75

## Resumen

El reconocimiento de las señales electromiográficas (EMG) correspondientes a movimientos de la mano se realiza actualmente mediante dispositivos que detectan varias señales, la información de cada señal debe ser procesada, lo que puede ser lento y costoso en términos computacionales para procesar todos los datos de manera efectiva. Este documento presenta un estudio basado en el análisis de señales electromiográficas, buscando encontrar patrones a partir de un banco de señales que contiene la información correspondiente a pruebas realizadas consistentes en obtener información de ocho electrodos adheridos a los brazos de cincuenta sujetos, quienes realizan diez movimientos diferentes con sus manos en lapsos de tiempo definidos. Se diseñó una Red Neuronal Convolutiva (CNN) para ajustar un porcentaje de esa información con los movimientos correspondientes con el fin de formar un modelo con la finalidad de que clasifique esa información, logrando predecir con precisión la mayoría de los movimientos a excepción de un par de ellos, llegando a un modelo con optimización hiperparámetros y con métricas de calidad relativamente alta.

**Palabras clave:** Red neuronal convolutiva, Señales electromiográficas, Aprendizaje profundo, Clasificación de señales

## Abstract

Hand Movement recognition of electromyographic signals (EMG) is currently performed by devices with many channels where the information of each channel must be processed, which can be slow and costly to process all the data effectively. This document presents a study based on the analysis of electromyographic signals, looking to find patterns from a signal bank that contains the information corresponding to tests carried out consisting of obtaining information from eight electrodes attached to the arms of fifty subjects, making ten different movements with his hands. A Convolutional Neural Network (CNN) was designed to fit a percentage that information with the corresponding movements toward train a model expected to classify that information, managing to accurately predict most of the movements except

for a couple of them, reaching at a model with optimized hyperparameters and with relatively high quality metrics.

# 1 Introducción

## 1.1 Planteamiento del objeto de estudio

Las señales electromiográficas (EMG) son señales eléctricas producidas por un músculo durante el proceso de contracción y relajación, estos músculos cuentan con fibras musculares con dos membranas, las cuales al estar en reposo tienen una diferencia de potencial de 90 mV negativo con respecto al exterior. Los impulsos nerviosos emitidos a estas fibras musculares son los que provocan los movimientos voluntarios y le indican el grado de contracción al sistema nervioso [1]. El conjunto de estas fibras musculares con la neurona del sistema nervioso central que proyecta su axón al músculo se llama unidad motora. El registro de los cambios producidos por la descarga de las fibras musculares de la unidad motora son registrados por el electrodo, cada uno se conoce como PA, su amplitud media es de unos 0,5 mV y la duración varía entre 8 y 14 ms. Una señal electromiográfica es el conjunto de los PAs registrados por el electrodo en un intervalo de tiempo [2].

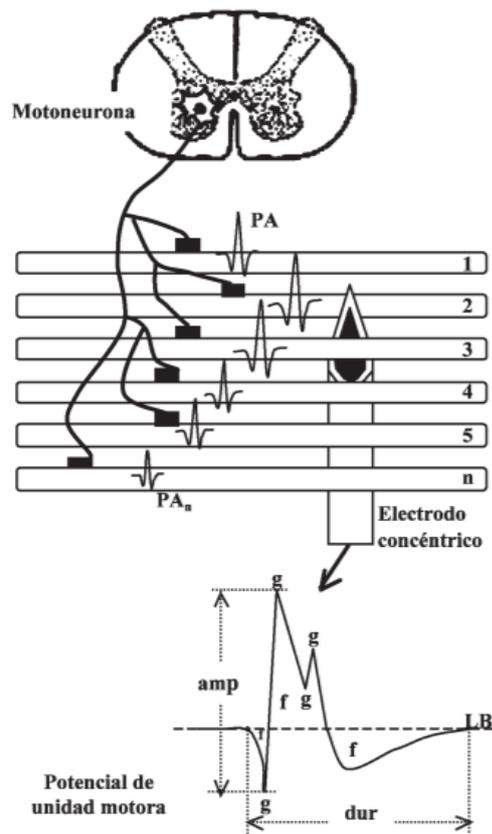


Figura 1: Representación esquemática de una unidad motora con n fibras musculares. Recuperado de [1].

La clasificación de señales electromiográficas se utiliza en diversos campos, como la ingeniería biomédica, la medicina, entre otros [5]. Tiene un papel importante en la ingeniería biomédica porque se usa comúnmente en el reconocimiento de movimientos para el control de prótesis [6]. La selección de músculos que son la fuente de estas señales EMG es muy importante en el desarrollo del método de clasificación ya que cada movimiento de la mano produce una activación muscular diferente que se puede distinguir por patrones específicos, clasificar que los patrones EMG consisten en algoritmos para la extracción y clasificación de las características más comunes entre la información correspondiente a un mismo electrodo y un mismo movimiento de la mano.

El problema radica en la dificultad de procesar la información de estas señales debido a su naturaleza ruidosa y es difícil encontrar el patrón debido a que las

señales EMG no son lineales y varían en el tiempo, lo que dificulta su clasificación [7]. Por lo tanto, la selección del algoritmo de clasificación es importante, por lo que se eligen las redes neuronales convolucionales debido a que son comúnmente utilizadas en problemas donde se deben encontrar patrones en sistemas de varias dimensiones como clasificación de imágenes. La mayoría de los métodos de aprendizaje automático requieren una base de datos de la cual extraer información para realizar pruebas y evaluar el modelo con los resultados obtenidos para ajustar sus parámetros. En este caso esa base de datos es un banco de señales que contiene información previamente clasificada [8].

Los datos del banco de señales se pueden dividir en objetos correspondientes a movimientos específicos que tienen características en el dominio del tiempo y características en el dominio de la frecuencia. Por tanto, es posible aprovechar la característica de las CNN de que pueden estructurarse en convoluciones de más de una dimensión. Siendo específicos, el banco de señales se divide primero en 50 grupo correspondientes cada uno a un sujeto de pruebas, cada grupo contiene a su vez otros dos subgrupos, cada uno corresponde a una de las dos manos, teniendo así uno para la izquierda y otro para la derecha, cada uno de estos subgrupos contiene 10 matrices correspondientes a los diez movimientos. Entonces tenemos 500 matrices que se de las cuales se pretende que el modelo de red neuronal pueda clasificar. De las 500 matrices podemos dividir las en grupos de 10 para tener 50 para cada movimiento, estas matrices son de dos dimensiones; una dimensión corresponde a los canales y otra a los PAs registrados por el electrodo al momento de obtener los datos; en total, cada matriz cuenta con 8 canales y 3200 datos por canal que varían en el dominio de las frecuencias, por lo que tenemos un sistema de tres dimensiones.

Una red neuronal común, trabaja con vectores lineales como entrada, al tener un sistema de 3 dimensiones no es posible ordenar la información de cada matriz en un vector, a menos que sea ordenando cada renglón de la matriz uno después del otro dentro de un mismo vector, lo que no tendría mucho sentido debido a que los elementos de los renglones tienen una dependencia temporal, la cual se perdería al momento de ordenar la matriz en un vector, por ese motivo es necesario un

modelo de red neuronal que admita matrices como entrada independientemente de la salida esperada. Ese modelo de red es la red neuronal convolucional, la cual tiene convoluciones que pueden ser de más de dos dimensiones.

Aunque las redes neuronales convolucionales han mostrado potencial en la clasificación de problemas, se requiere una buena selección de hiperparámetros. Estas redes cuentan con más hiperparámetros que las redes comunes debido a que también se puede variar el tamaño de las conexiones entre las neuronas de una capa a otra y el porcentaje de neuronas que pierden conexión de una capa a otra, así como los que tienen las redes neuronales normales, el número de capas, número de neuronas por capa y el learning rate los cuales se explicarán que es posteriormente en el documento. Todos esos hiperparámetros afectan el comportamiento de la red, por lo que la exactitud del modelo tiene una dependencia de estos factores, elegir una combinación de ellos de manera empírica es lo que usualmente se hace, aunque no siempre es lo más óptimo.

Seleccionar los hiperparámetros correctos no es sencillo debido a las muchas combinaciones posibles que pueden existir, para ello, se implementa un modelo de Gaussian Process Regression (GPR) con el fin de encontrar una función en términos de todos los parámetros que mostrará la variación de la precisión del modelo.

## **1.2 Justificación**

El uso de la EMG en las ciencias de la rehabilitación ha contribuido a comprender los patrones neuromusculares utilizados en la ejecución de distintos gestos motores. También ha ayudado al entendimiento de las posibles causas y consecuencias de una lesión y/o disfunción sobre el comportamiento electrofisiológico del músculo.

Poder clasificar señales electromiográficas puede ser de utilidad tanto para la medicina como para el entretenimiento, gracias a la clasificación de estas señales se pueden detectar enfermedades como esclerosis lateral amiotrófica, miastenia gravis, miopatías, polineuropatía diabética, polineuropatías, radiculopatía, neuropatías o síndrome del túnel del carpo; el uso de su clasificación para personas sanas y para

personas con estas enfermedades para que los especialistas puedan comparar resultados en la obtención de estas enfermedades para pacientes con posibilidades de tener alguna de ellas o para descartarlas en otros. También puede que en los siguientes años se encuentren maneras de aplicarlas en algún medio de entretenimiento debido a que hoy ya se puede visualizar una mano virtual mediante estas señales, con la clasificación de estas señales en varios movimientos de una mano.

Las redes neuronales han demostrado hacer predicciones y clasificación de patrones que no son estacionarios, mientras que las redes neuronales convolucionales pueden ser de mucha utilidad debido a que pueden trabajar con sistemas con dos o más dimensiones, aparte de que también han demostrado ser especialmente útiles para la clasificación. Es por este motivo que proponer una red neuronal convolucional puede ser de gran ayuda para clasificar los datos del banco de señales debido a que los datos no son estacionarios y contienen el ruido generado debido a la sensibilidad de los dispositivos que recolectan las señales [22].

### 1.3 Hipótesis

La hipótesis es que las redes neuronales convolucionales serían ideales para clasificar la información del banco de señales que contiene las señales electromiográficas de diferentes pruebas, crear un modelo y entrenarlo con la información correspondiente a las matrices que contienen las señales tomadas al momento de que el sujeto de pruebas estaba haciendo uno de los diez movimientos nos devolvería una red neuronal capaz de aceptar una de esas matrices como entrada y devolvernos un vector de diez elementos, cada uno con la probabilidad de que la matriz de entrada pertenezca a un movimiento en específico. La mayor de esas probabilidades es la predicción de la red de que esa matriz pertenezca a ese movimiento, se esperaba que las predicciones fueran acertadas en al menos un 75 % de las pruebas y alcanzando métricas de calidad como lo son la precisión alcanzarán valores superiores al 75 % también.

## 1.4 Objetivos previstos

### 1.4.1 Objetivo general

- El principal objetivo de la investigación es proponer una red neuronal convolucional que clasifique de manera precisa las señales electromiográficas obtenidas de electromiografos conectados a una mano en una posición específica. Alcanzando valores de precisión y exactitud de al menos el 75 %.

### 1.4.2 Objetivos Particulares

- Tener una propuesta de arquitectura de la red que sea compatible con el banco de señales.
- Tener un banco de señales filtrado y ordenado de manera que se generen entradas para la red neuronal con valores entre -1 y 1.
- Encontrar los parámetros de la red que mejor se adapten para que las entradas de la red se ajusten con las salidas de la red esperadas (los movimientos) mediante el algoritmo de aprendizaje ADAM.
- Encontrar los hiperparámetros óptimos de la red mediante un proceso de regresión gaussiano.
- Validar los resultados obtenidos para estimar las métricas de calidad del modelo.

Como productos del trabajo realizado se espera obtener un conjunto de señales reescaladas y ordenadas en tres conjuntos de matrices de dos dimensiones que sirvan de entrada a la red, estos cada una de las matrices de los conjuntos deben estar normalizadas y debe existir un vector que indique el movimiento que corresponde a cada matriz.

También se busca generar un modelo de red neuronal convolucional capaz de aprender a clasificar esas matrices entrenándolo mediante dos de los tres conjuntos, alcanzando una precisión y una exactitud relativamente altas.

Del modelo entrenado se esperan obtener buenas métricas de calidad al clasificar el tercer conjunto de matrices después de llegar a una convergencia en la exactitud de los otros dos conjuntos. Para contribuir con el aumento de las métricas de calidad se espera encontrar la combinación de hiper parámetros óptima de la red usando GPR y repitiendo el proceso. En general se obtendría un banco de señales normalizado y reordenado, del cual el modelo de red CNN debería ser capaz de clasificar un conjunto de señales electromiográficas correspondiente a uno de diez movimientos después de ser entrenado.

## 2 Antecedentes

### 2.1 Señales electromiográficas

#### 2.1.1 Unidades motoras

El movimiento a voluntad de los organismos vertebrados se debe al sistema muscular por el cual estamos constituidos. Este sistema está constituido por una variedad de músculos inervados, conocidos como los músculos esqueléticos o estriados, que produce actividad mioeléctrica relacionada con la electromiografía [9]. Este sistema está ampliamente estudiado por la medicina y así como las interacciones de sus componentes mediante impulsos nerviosos, el tejido muscular conduce la electricidad de manera similar a como lo hacen los nervios y el nombre dado a estas señales eléctricas es la acción muscular potencial.

El principal causante de las señales que queremos clasificar es llamado unidad motora, esta es la unidad funcional más pequeña para describir el control neuronal del proceso de contracción muscular. Cada unidad motora está compuesta por una motoneurona y todas las fibras musculares que inerva, las cuales pueden variar en número. Los músculos que usamos para controlar los movimientos más finos y precisos están formados varias unidades motoras pequeñas, mientras que los músculos que controlan movimientos gruesos o de potencia tienen una gran cantidad de fibras musculares por unidad motora. Para nuestro caso de estudio tenemos un sistema

compuesto por varias motoneuronas pequeñas [10].

Las fibras musculares se excitan a través del control neuronal. Un equilibrio iónico entre los espacios internos y externos de una célula muscular forma un potencial de reposo en la membrana de la fibra muscular de entre unos -80 y -90 mV dependiendo del tamaño de la membrana cuando no esta contraída. Esta diferencia de potencial se mantiene debido a los procesos fisiológicos que generan una carga intracelular negativa en comparación con la superficie externa, en una fibra muscular, la concentración de iones de potasio es mayor en el interior que en el exterior de las células; ocurre lo contrario con los iones de sodio. La entrada de más iones de sodio aumenta la carga positiva al interior de la fibra muscular, y ocurre un fenómeno denominado despolarización de membrana, que consiste en que ingreso de estos iones y la aceleración del flujo provoquen una pérdida de iones de potasio causando una variación del potencial de membrana de entre -80 mV y +30 mV aproximadamente [11]. La señal electromiográfica se basa en potenciales de acción en la membrana de la fibra muscular que resultan de los procesos de despolarización y repolarización descritos anteriormente.

### 2.1.2 Generación y medición de las señales

En pocas palabras, los músculos producen señales electromiográficas mediante los procesos de polarización de sus membranas. Esta característica muscular es la que se ha pretendido aprovechar para el control informático mediante la creación de interfaces de comunicación entre el usuario y una prótesis o una máquina o para diagnosticar varias enfermedades neuromusculares graves como lo es la esclerosis lateral amiotrófica. El hecho de poder crear una interfaz de este tipo permitiría a cualquier usuario controlar sistemas informáticos o electrónicos contrayendo ciertos músculos, pero más que para usuarios normales, un sistema de este tipo es especialmente interesante para individuos que padecen algún tipo de parálisis que dificulta sus actividades y su interacción con el mundo que les rodea, ofreciendo posibilidades antes inexistentes y mejorando su calidad de vida [12].

La electromiografía de superficie es un método para registrar la información

presente en estos potenciales de acción musculares de una manera no invasiva. Se ha utilizado un tipo de electrodo para adquirir estas señales, el electrodo no invasivo. Cuando EMG se adquiere de electrodos montados directamente en la piel, la señal es una combinación de todo el músculo potenciales de acción de las fibras que se producen en los músculos bajo la piel [13].



Figura 2: Ejemplo de electromiografía de superficie. Recuperado de [24].

Estos potenciales de acción ocurren en intervalos aleatorios. Entonces, en cualquier momento, la señal EMG puede dar un voltaje positivo o negativo. Los potenciales de acción de las fibras musculares a veces se adquieren mediante el uso de electrodos de alambre o aguja colocados directamente en el músculo. La combinación de la acción de las fibras musculares. potenciales de todas las fibras musculares de un solo motor unidad es el potencial de acción de la unidad motora que puede ser detectado por un electrodo de superficie de la piel (no invasivo) ubicado cerca de este campo. La señal se capta en el electrodo y se amplifica [15].

Las ventajas de la técnica intramuscular son que permite evaluar músculos profundos y pequeños que superficialmente son imposibles de medir debido a la interferencia de señales emitidas por músculos adyacentes [16]. También permite detectar la actividad eléctrica de zonas o fascículos específicos de músculos debido a que los

electrodos tienen un área de registro pequeño . Las desventajas son que la inserción de la aguja provoca molestias durante la contracción, lo cual podría afectar el gesto motor evaluado [17], debido a eso se eligió el método no invasivo para la toma de datos que está en el banco de señales.

### 2.1.3 Ruido en las señales electromiográficas

Al detectar y registrar la señal EMG, hay dos temas principales de preocupación que pueden llegar a influir en la señal, la primera es el ruido, es decir, la relación entre la energía en las señales EMG y la energía en las señales de ruido. En general, el ruido se define como señales eléctricas que no forman parte de las señales EMG y que provienen de la sensibilidad de los electrodos adheridos a los brazos de los sujetos. El otro problema es la distorsión de la señal, lo que significa que la contribución relativa de cualquier frecuencia el componente de la señal EMG no debe alterarse.

Por lo general, se utiliza un amplificador diferencial como primera etapa de amplificación. También pueden seguir pasos de amplificación adicionales. Antes de mostrarse o almacenarse, la señal se procesa para eliminar las señales de baja o alta frecuencia. Para ello se la hace pasar por una etapa de filtrado, en concreto se eliminan las componentes de frecuencia de 50 Hz. Las responsables de este ruido de 50 Hz aproximadamente son en general las líneas de tensión domésticas presentes en todas las instalaciones donde se realiza la toma de datos. Hay que tener en cuenta que existen otras fuentes de ruido que afectan a la señal y que no es posible su eliminación total [2].

### 2.1.4 Método Wavelet para reducción de ruido

El método Wavelet para la reducción de ruido se ha usado ampliamente para una gran cantidad de imágenes y señales, se ha usado en conjunto con algoritmos genéticos o de manera individual y ha demostrado mostrar buenos resultados en la reducción de ruido de las señales electromiográficas [3].

Las wavelets son señales, o formas de onda, las cuales tienen una duración limi-

tada y un valor promedio de cero. Las wavelets pueden ser irregulares y asimétricas, características que les otorgan una mejor adaptación en el análisis de señales en comparación con la transformada de Fourier [4].



Figura 3: Esquema de la transformada de wavelet. Recuperado de [4]

La wavelet elegida, para implementar la transformada wavelet a una señal, se le asigna el nombre de wavelet madre. Se le conoce como wavelet madre ya que será esta la que sufra algunas modificaciones para realizar el análisis: se expandirá o se comprimirá, y se trasladará a lo largo de la señal. Estas modificaciones están a cargo de los parámetros de escalamiento y desplazamiento. En el escalamiento se alarga o se comprime la wavelet, lo que nos permite ver tanto los detalles como los componentes de la señal de forma global. Mientras que el desplazamiento se refiere al recorrido de la wavelet a lo largo de la señal. Podemos definir a la wavelet madre  $\psi_{a,b}(t)$ , añadiéndole los parámetros de escalamiento y de desplazamiento, mediante la siguiente fórmula:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \cdot \psi\left(\frac{t-b}{a}\right) \quad (1)$$

donde  $\mathbf{a}$  es el escalamiento y  $\mathbf{b}$  es el desplazamiento. De forma general la transformada wavelet descompone una señal mediante el uso de las versiones escaladas y desplazadas de la wavelet madre. Podemos decir que la wavelet actúa como un filtro pasa banda el cual solo permite el paso de ciertos componentes de la señal a una determinada frecuencia [18].

Existen distintos tipos de transformada wavelet como lo son la transformada wavelet continua, la transformada wavelet discreta y la transformada wavelet packet. A continuación, se explicarán las dos primeras. La transformada wavelet (WT) da

una descomposición de la función en diferentes escalas, tendiendo a ajustarse a las escalas en lugares de tiempo donde la onda se asemeja mejor a la función. El proceso puede revertirse, dando así la reconstrucción de la función de vuelta. Es más conveniente definir el WT solo en escalas discretas  $a$  y tiempos discretos  $b$  [31].

Los parámetros de escalamiento y desplazamiento dan paso a la obtención de los coeficientes wavelet. Los coeficientes wavelet nos indican cuanta relación hay entre la señal y la wavelet madre. Esta relación nos permite conocer los componentes frecuenciales de la señal.

La transformada wavelet discreta (DWT) se obtiene al discretizar los parámetros de desplazamiento y escalamiento dentro de la transformada wavelet continua. Usualmente los valores que se implementan para realizar esto son:

$$\begin{aligned} a &= 2^{-j} \\ b &= k2^{-j} \end{aligned} \tag{2}$$

donde  $\mathbf{a}$  es el escalamiento,  $\mathbf{b}$  es el desplazamiento y  $\mathbf{j}, \mathbf{k}$  deben ser valores enteros. Definidos los parámetros de escalamiento y desplazamiento con valores discretos, la wavelet madre toma la forma:

$$\psi_{j,k}(t) = 2^{\frac{j}{2}} \psi(2^j t - k); j, k \in Z \tag{3}$$

Esto último permite definir la transformada wavelet discreta como:

$$DWT_{j,k} = 2^{\frac{j}{2}} \int_{-\infty}^{\infty} x(t) \psi(2^j t - k) dt \tag{4}$$

donde  $X(t)$  es una señal discreta.

La transformada wavelet discreta nos permite reconstruir la señal una vez que calculamos los coeficientes wavelet. Para realizar esta reconstrucción se necesita de dos funciones: la función wavelet  $\psi(t)$  y  $\phi(t)$  la función escala. La reconstrucción de la señal se obtiene mediante la siguiente ecuación:

$$x(t) = \sum_k \sum_j c_{j,k} \phi(t) + \sum_k \sum_j d_{j,k} \psi(t); j, k \in Z \quad (5)$$

Donde  $c_{j,k}$  son los coeficientes de escala o de aproximación y  $d_{j,k}$  representa los coeficientes wavelet o de detalle. Cabe mencionar que los coeficientes de aproximación están asociados con la función escala, mientras que los coeficientes de detalle lo están con la función wavelet.

Los coeficientes de aproximación y de detalle nos permiten obtener información sobre las características de la señal, además con su manipulación podemos obtener una nueva señal eliminando componentes no deseados de la señal original .

El desarrollo de algoritmos para evaluar el DWT condujo a la implementación de "bancos de filtros".<sup>Estos</sup> filtros corresponden a un filtro pasa bajo y un filtro pasa alto, cuando la señal original pasa a través de tales filtros, se obtienen los coeficientes de salida  $c_{j,k}$  y  $d_{j,k}$  respectivamente. Se puede obtener una descomposición de una señal a diferentes niveles pasando los coeficientes de escala obtenidos del filtrado anterior por un par de filtros idénticos, obteniendo así los coeficientes del siguiente nivel, como se muestra en la figura 4 [\[30\]](#).

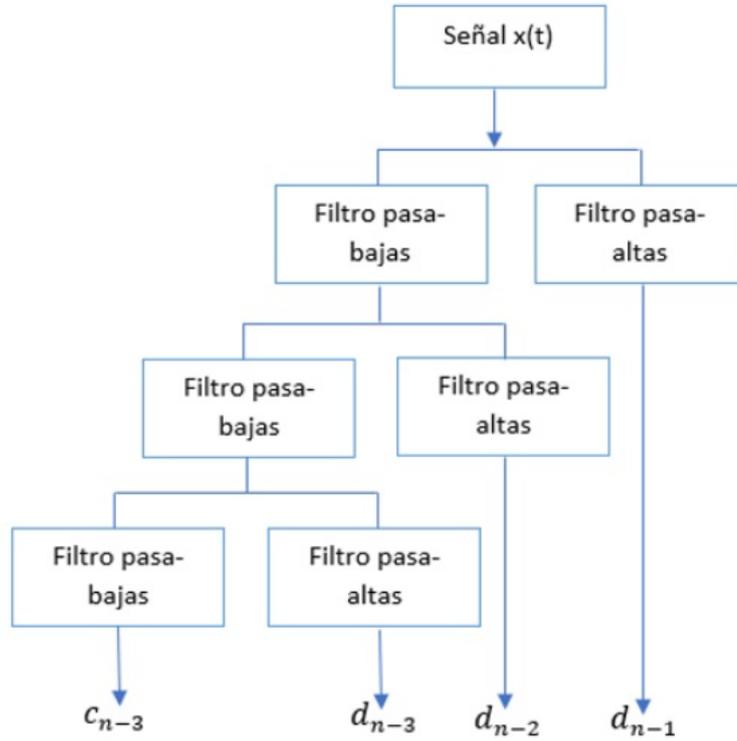


Figura 4: Proceso de filtrado de una señal para la extracción de características, cada par de filtros en un determinado nivel, genera los coeficientes que serán pasados al siguiente nivel.

Por esos motivos la base de datos usada en este documento fue tratada por el método Wavelet para reducir el ruido no deseado en las señales multicanal altamente no lineales, ya que la el método brinda posibilidades de estudio para el dominio de la frecuencia y del tiempo en una señal de manera simultanea.

### 2.1.5 Banco de señales

Llegados a este punto se ha obtenido una señal en unos márgenes de tensión adecuados y se ha eliminado ruido en la medida de lo posible. La señal original es de naturaleza analógica, es decir, entre dos instantes de tiempo cualesquiera de ésta existen infinitos valores de la señal. Al no poder trabajar los ordenadores con señales analógicas hay que convertirlas a señales digitales o discretas muestreando la señal analógica con una frecuencia adecuada para no perder ninguna información relevante. El teorema del muestreo afirma que para muestrear una señal sin per-

der información y por tanto poder reproducirla perfectamente partiendo de la señal discreta, se debe muestrear a una frecuencia de al menos el doble de la frecuencia fundamental de la señal analógica. Las señales EMG tienen una frecuencia que oscila entre 50 y 150 Hz, de aquí se deduce que la frecuencia de muestreo adecuada no debe ser menor de 300 Hz. Quizás se pueda pensar que una frecuencia de muestreo mayor, por ejemplo 600 Hz, sea mejor, pero con 300 Hz se obtiene toda la información fundamental de la señal y se minimiza el número de datos necesarios para manejar las señales EMG en ordenadores. A una frecuencia de 300 Hz y muestreando durante un segundo se representa una señal EMG como un vector de 3200 elementos, un tamaño aceptable que permite un tiempo de cómputo muy bueno en un ordenador actual de prestaciones medias [21].

Los movimientos que se muestran a continuación fueron diseñados por fisioterapeutas con la propósito de registrar la actividad generada por los músculos del brazo cuando la mano está en una posición específica, fueron recuperados de [22].

La lista de la descripción de cada movimiento es:

- Posición Inicial (Descanso), en esta posición el participante mantendrá la palma extendida con dedos cerrados, sin ejercer demasiada fuerza sobre ellos. Desde esta posición comenzarán movimientos posteriores (Figura 5 a).
- Pronación. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder para hacer el movimiento; gira la palma de la mano con la vista hacia abajo y vuelve a la posición inicial (figura 5 b).
- Supinación. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder para hacer el movimiento; la palma de la mano se gira hacia arriba y vuelve a la posición inicial (figura 5 c).
- Extensión. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder para hacer el movimiento; sin mover el codo, se saca la palma

de la mano y se vuelve a la posición inicial (figura 5 d).

- Flexión. Brazo pegado al tronco, codo en flexión de  $90^\circ$ , mano en posición inicial, proceder a hacer el movimiento; sin mover el codo, la palma de la mano se lleva hacia adentro, con la vista hacia el cuerpo y volver a la posición inicial (figura 5 e).
- Desviación cubital. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder a realizar el movimiento; sin mover el codo, la mano se inclina hacia abajo y a la posición inicial (figura 5 f).
- Desviación radial. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder a realizar el movimiento; sin mover el codo, la mano se inclina ligeramente hacia arriba y vuelve a la posición inicial (figura 5 g).
- Pinza fina. Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano en posición inicial, proceder para hacer el movimiento; toque las yemas de los dedos pulgar e índice, vuelva a la posición inicial (figura 5 h).
- Pinza gruesa (Puño cerrado). Brazo pegado al tronco, codo flexionado  $90^\circ$ , mano en posición inicial, proceder a realizar el movimiento; cierra el puño dejando el dedo pulgar afuera otros dedos y volver a la posición inicial (figura 5 i).
- Expansión (abducción de dedos). Brazo pegado al tronco, codo flexionado a  $90^\circ$ , mano dentro posición inicial, proceder a realizar el movimiento; todos los dedos se separan y giran hacia la posición inicial (figura 5 j).

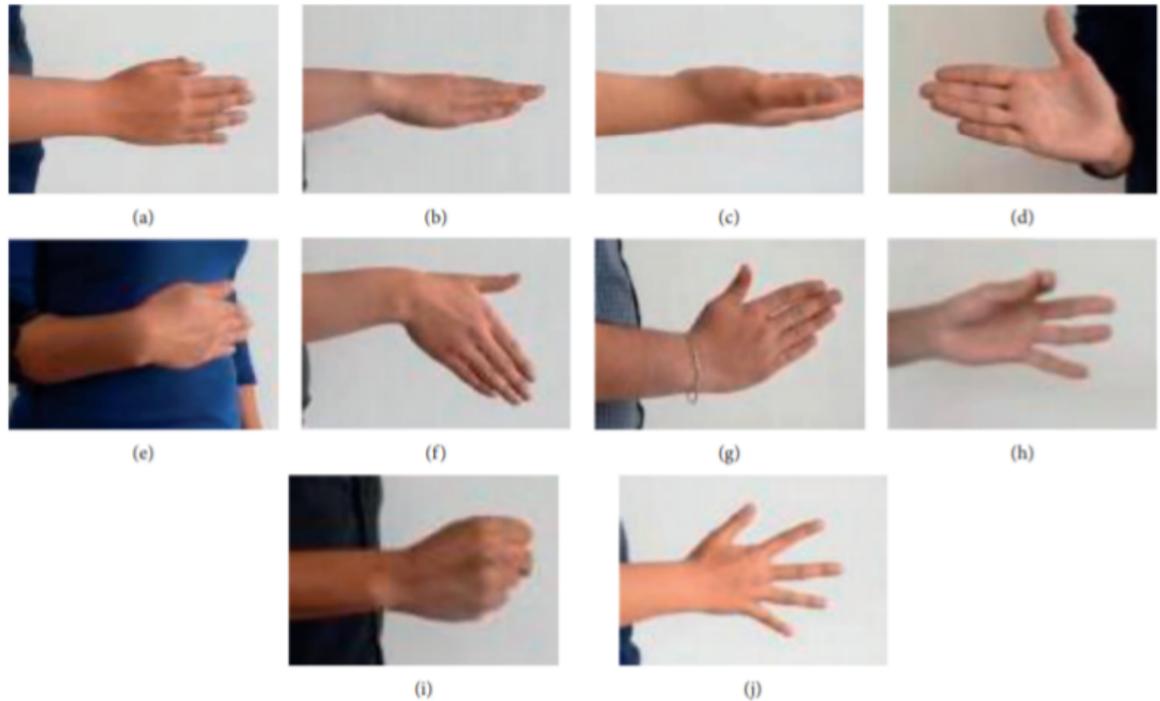


Figura 5: Lista de movimientos realizados por cada sujeto. (a) Posición inicial, (b) pronación, (c) supinación, (d) extensión de la muñeca, (e) flexión de la muñeca, (f) desviación cubital de la muñeca, (g) desviación radial de la muñeca, (h) recolección manual, (i) mano cerrada, (j) mano abierta. Tomado de [22].

Una prueba consistía en uno de los cincuenta sujetos de prueba realizó uno de los diez movimientos en un intervalo de dieciséis segundos con cada uno de sus manos mientras tenía los ocho electrodos conectados y tratando de espaciar los movimientos en intervalos de tiempo iguales, el banco de señales contiene una matriz correspondiente a cada prueba realizada. Cada matriz contiene ocho renglones cada uno con 3200 columnas, una matriz es la representación en computadora de cada prueba realizada y cada vector es la información recabada por uno de los ocho electrodos que el sujeto tenía en el brazo al hacerse la prueba.

El objetivo es encontrar la correlación que tienen las diferentes pruebas en los diferentes sujetos con cada uno de los movimientos que realizó al momento de la prueba, por ejemplo, en la posición inicial, al no contraer ningún musculo, las variaciones de la amplitud de la señal a lo largo del lapso de tiempo que dura la prueba en

teoría deberían ser mínimas, aunque la amplitud promedio de la señal pueda variar en la mano o en el sujeto. Lo observado al analizar las señales corresponde con la hipótesis que se tenía, en el siguiente ejemplo se muestra la señal observada para una prueba realizada al momento de tener la mano en posición inicial.

Sin embargo, al momento de hacer otros movimientos, es notorio el intervalo de la señal donde ocurre un evento. Eso se puede apreciar en la variación de la amplitud de la señal producida por la prueba en cualquier canal. Por ejemplo, para los movimientos de pronación y extensión se generan las variaciones en las amplitudes de un movimiento a otro también se aprecian aunque para algunos movimientos esa variación es más sutil como para el movimiento de extensión como se muestra en la figura 7b y 7c:

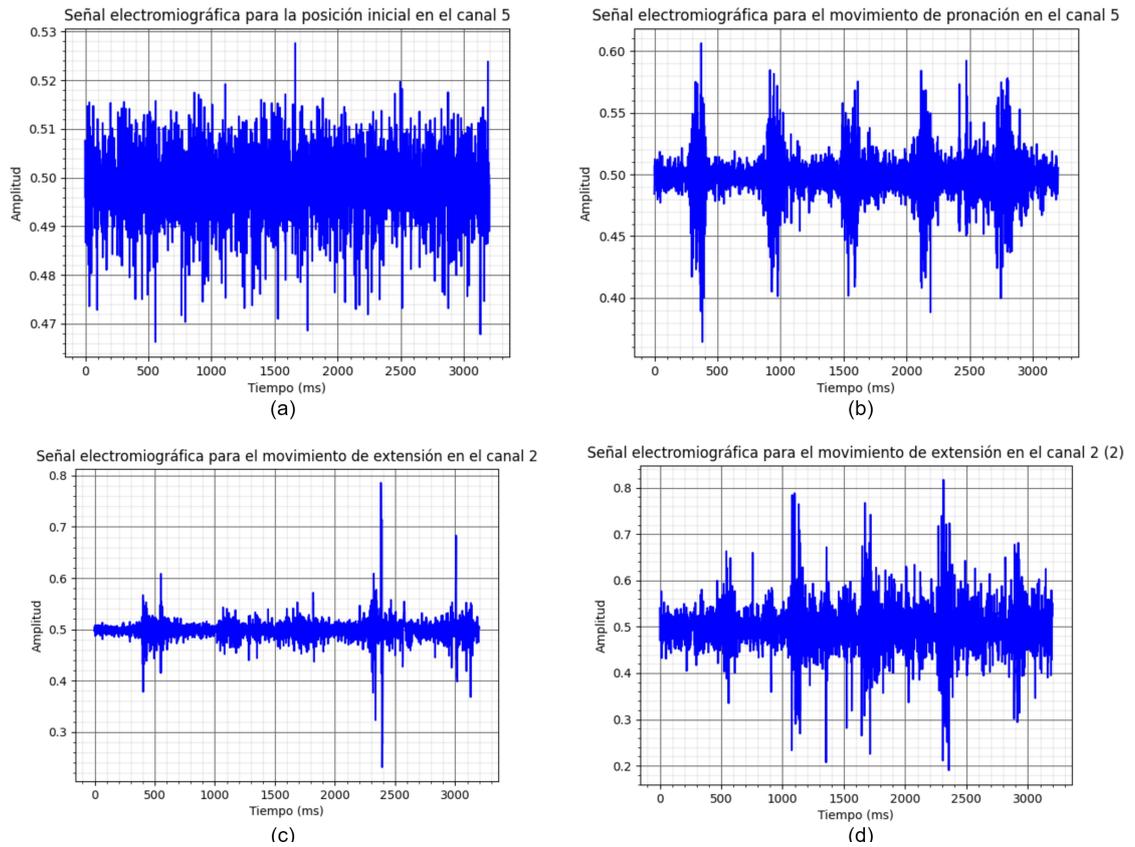


Figura 6: Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano en la posición inicial (7a). Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano realizando el movimiento de pronación (7b). Señal correspondiente al canal 5 de la prueba realizada a un sujeto con la mano realizando el movimiento de extensión (7c). Señal correspondiente al canal 2 de la prueba realizada a un sujeto con la mano realizando el movimiento de extensión (7d).

A diferencia de la posición inicial, hay cinco intervalos que resaltan más y cada uno corresponde al momento donde el sujeto pasa de la posición inicial a la de la prueba.

Las variaciones en las amplitudes no solo dependen del movimiento, también dependen del canal del que se aprecia la señal, usando el mismo movimiento de extensión como ejemplo, se puede apreciar que las variaciones de la amplitud de la señal del canal 2 no son idénticas a las del canal en la gráfica 7c.

En este caso se vuelve muy extrema la diferencia entre las amplitudes de los eventos que ocurren durante una misma prueba, el cuarto pico de la señal tiene una amplitud notablemente superior a los demás, esto puede darse por diversos factores, como por ejemplo una mala reducción de ruido para ese caso en específico o una diferencia en la fuerza con la que el sujeto cambia de la posición inicial a la de extensión durante el cuarto evento con respecto de los demás.

Por último, las variaciones también dependen del sujeto de pruebas, usando como ejemplo el movimiento de extensión y el canal 2, pero para un sujeto de pruebas distinto se obtuvieron los resultados de la gráfica 7d.

Los patrones varían con respecto del sujeto para cada movimiento, no es posible que dos personas generen las misma señales al hacer la misma acción, la variación en los picos de la señal se debe a su vez a que una persona no puede efectuar el mismo movimiento más de una vez exactamente de la misma manera ni generar la misma señal.

Teniendo la señal producida por la contracción muscular de una persona en el ordenador de forma manejable y a la que se le pueden aplicar determinadas operaciones. Nos centramos ahora en la descripción de las principales técnicas matemáticas para la manipulación de estas señales. Debido a la naturaleza estocástica de estas señales, hay que recurrir a técnicas específicas como las redes neuronales para clasificarlas exitosamente.

## **2.2 Redes Neuronales Convolucionales**

### **2.2.1 Concepto general de redes neuronales**

Uno de los algoritmos de machine learning más populares son las redes neuronales. Las redes neuronales existen desde hace muchos años y han pasado por varios períodos. Pero recientemente ganaron terreno sobre muchos otros algoritmos de aprendizaje automático de la competencia [26].

Eso es debido al desarrollo de computadoras que son rápidas, el uso de unidades

de procesamiento gráfico (GPU) contra el uso más tradicional de unidades de procesamiento informático (CPU), mejores algoritmos y el diseño de redes neuronales y conjuntos de datos cada vez más grandes. Se puede describir una red neuronal como un modelo matemático para el procesamiento de información. Una red neuronal no es un programa fijo, sino un modelo, un sistema que procesa información [26]. Una descripción más general de una red neuronal sería como un gráfico computacional de operaciones matemáticas, en el que podemos identificar dos características principales ([29]):

- La arquitectura de la red neuronal: describe el conjunto de conexiones, direcciones, feedforward, recurrencia, multicapas o de una sola capa, el número de capas y el número de neuronas en cada capa [29].
- El aprendizaje: describe lo que comúnmente se define como entrenamiento. El más común pero no exclusivo es con el gradiente descendiente o back propagation.

### 2.2.2 Neuronas

Una neurona es una función matemática que toma un valor de entrada y lo convierte en un valor de salida mediante una función de activación.

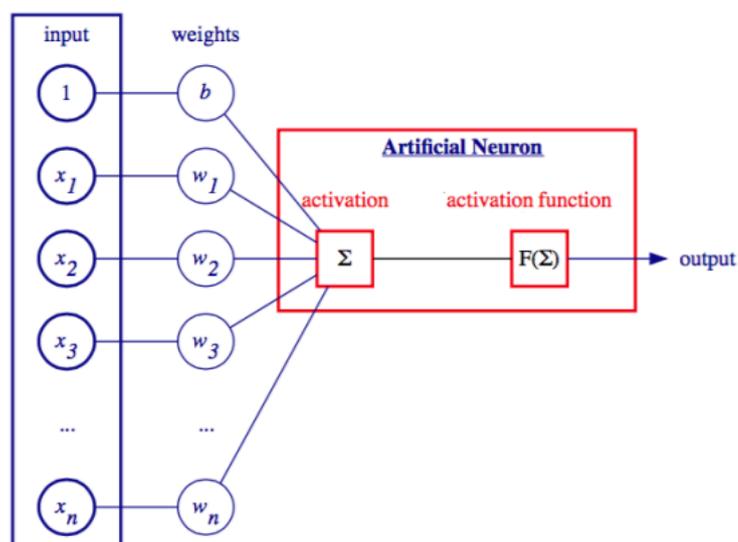


Figura 7: Diagrama de los elementos de una neurona. Recuperado de [29]

Esta salida se expresa de la siguiente manera:

$$y = f\left(\sum x_i w_i + b\right) \quad (6)$$

El proceso por el que una neurona procesa la información es el siguiente:

1.- Primero hace la sumatoria de

$$\sum x_i \omega_i$$

de cada una de las  $x_i$  entradas por su peso correspondiente  $\omega_i$ , también conocido como valor de activación.  $x$  representa el valor numérico de la entrada a la red neuronal dado al dato que le corresponde o a la salida de una neurona de una capa anterior; los valores de activación son representaciones de que tan afectado es la neurona por cada una de sus entradas; y los valores  $b$  son coeficientes llamados sesgo que cambian la forma en la que afecta todo el conjunto de entradas a la neurona.

2.- Después a al resultado obtenido de la sumatoria se le aplica una función de activación  $f$  la cual puede variar dependiendo de para que se haga la red y serán explicadas algunas más adelante.

El valor de activación de la neurona formalmente se define como el producto punto entre las entradas y los valores de activación, siendo ambos vectores perpendiculares entre si de manera que  $\vec{x} \cdot \vec{\omega} \neq 0$ , entonces todos los vectores  $\vec{x}$  que cumplan con la condición definirán un hiperplano en el espacio  $R^n$ , siendo  $n$  el número de elementos del vector  $\vec{x}$ .

Gráficamente, el trabajo del sesgo,  $b$ , es permitir que el hiperplano pueda alejarse del centro del sistema de coordenadas. Si no usamos el sesgo, el neurona tendrá un poder de representación limitado [30].

### 2.2.3 Capas de la red

Una red neuronal aparte de poder tener varias neuronas puede tener varias capas interconectadas, cada una con un número de neuronas independiente, la primer capa

de la red tiene tantas neuronas como entradas a la red se tengan, las capas ocultas, son las capas de la red que hay entre la capa de entrada y la capa de salida, el número de neuronas que tiene cada una de estas capas se define por estándares probados anteriormente, y la capa de salida tiene tantas neuronas como salidas de la capa se necesiten. Esos valores se mantienen constantes durante el proceso de entrenamiento de la red neuronal [31].

Las redes neuronales de una sola capa solo pueden clasificar clases linealmente separables y no existe ninguna restricción que impida el uso de más capas, no afecta de manera negativa el comportamiento de la red a menos que se agreguen demasiadas capas, en ese caso afecta por el coste computacional, exigiría mucho a un equipo el procesar tanto la información.

Las neuronas de una capa pueden estar conectadas en su totalidad con las neuronas de otra capa adyacente, las redes que cumplen con esta condición son llamadas "fully connected", como el siguiente ejemplo:

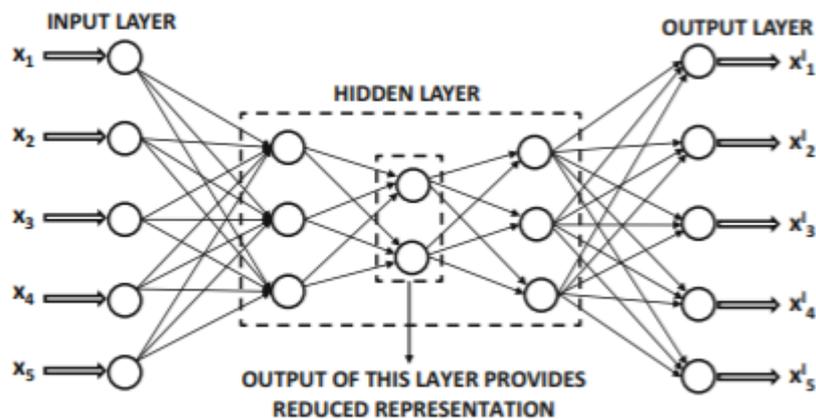


Figura 8: Representación gráfica de una red neuronal multicapa. Recuperado de [31]

Pero esa condición no es necesaria, se puede desconectar un porcentaje de esas neuronas para evitar el sobre aprendizaje de la red mediante algo llamado dropout. El dropout entre una capa y otra desconecta de manera aleatoria un cierto porcentaje de las neuronas, lo que es equivalente a hacer 0 un porcentaje de los elementos del vector  $\vec{w}$  [32].

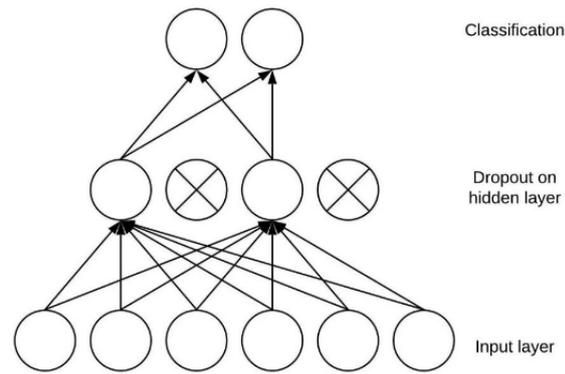


Figura 9: Diagrama del dropout de una capa a otra. Recuperado de [32]

#### 2.2.4 Funciones de activación

Ahora se sabe que las redes multicapa pueden clasificar clases linealmente independientes. Pero para hacer esto, necesitan satisfacer una condición más. Si las neuronas no tienen funciones de activación no lineales, su salida sería la suma ponderada de las entradas, que es una función lineal. Entonces toda la red neuronal, es decir, una composición de neuronas, se convierte en una composición de funciones lineales, que también es una función lineal a su vez. Esto significa que incluso si agregamos capas ocultas, la red seguirá siendo equivalente a un modelo de regresión lineal simple, con todas sus limitaciones. Para convertir la red en una función no lineal, usaremos funciones de activación no lineales para las neuronas. Por lo general, todas las neuronas de la misma capa tienen la misma función de activación, pero diferentes capas pueden tener diferentes funciones de activación [33].

#### 2.2.5 Función sigmoide

Muchos procesos naturales y curvas de aprendizaje de sistemas complejos muestran una progresión temporal desde unos niveles bajos al inicio, hasta acercarse a un clímax transcurrido un cierto tiempo; la transición se produce en una región caracterizada por una fuerte aceleración intermedia. La función sigmoide permite describir esta evolución. Su gráfica tiene una típica forma de "S" [33].

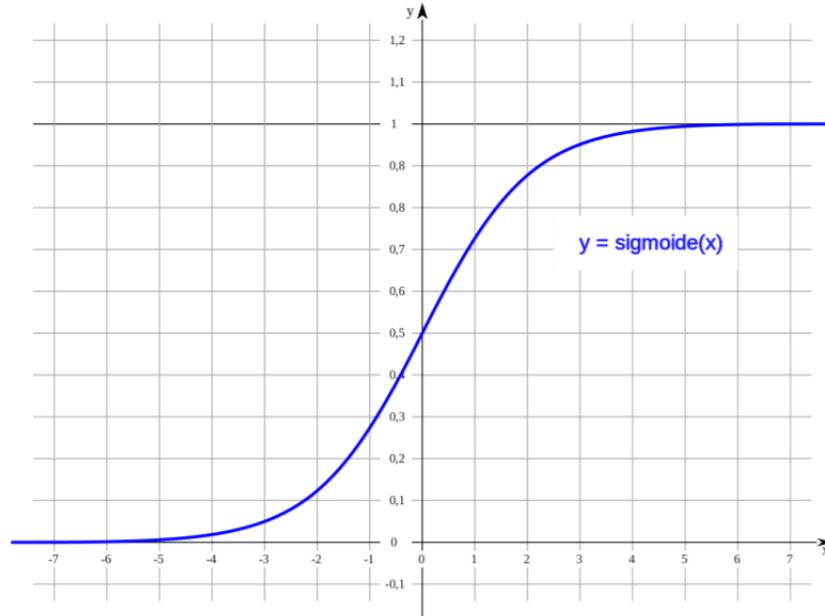


Figura 10: Diagrama de la función sigmoide. Recuperado de [33]

A menudo la función sigmoide se refiere al caso particular de la función logística, cuya gráfica se muestra a la derecha y que viene definida por la siguiente fórmula:

$$P(t) = 1 / (1 + e^{-t}) \quad (7)$$

Esta función tiene la propiedad de que su derivada se puede expresar en términos de la misma función:

$$P'(t) = P(t)[1 - P(t)] \quad (8)$$

Esta función de activación es especialmente útil al momento de clasificar, pues devuelve valores en el rango de cero a uno, lo que puede interpretarse como una probabilidad sin normalizar [34].

Algunos de los inconvenientes que puede presentar son que converge de una manera muy lenta, que no está centrada entre el cero y el uno y que puede saturar el método de entrenamiento del gradiente descendente.

### 2.2.6 Función softmax

La función softmax es comúnmente utilizada como capa de salida de los clasificadores basados en redes neuronales. Tales redes son comúnmente entrenadas usando un régimen de cross entropy, con lo que se obtiene una variante no lineal de la regresión logística multinomial. Es una generalización de la función logística. Se emplea para comprimir un vector de  $k$  dimensiones de valores reales arbitrarios en un vector con  $k$  elementos de valores reales en el rango  $[0, 1]$ . La función está dada por [31]:

$$\sigma(z)_j = e^{z_j} / \sum_k e^{z_k} \quad (9)$$

En teoría de la probabilidad, la salida de la función softmax puede ser utilizada para representar una distribución categórica- la distribución de probabilidad sobre  $K$  diferentes posibles salidas. La función softmax es empleada en varios métodos de clasificación multiclase tales como Regresión Logística Multinomial, análisis discriminante lineal multiclase, clasificadores Bayesianos, y Redes Neuronales Artificiales. Para la clasificación resulta especialmente útil, pues devuelve una salida normalizada que puede interpretarse como la probabilidad de que el vector de entrada corresponda a una clase u otra [35].

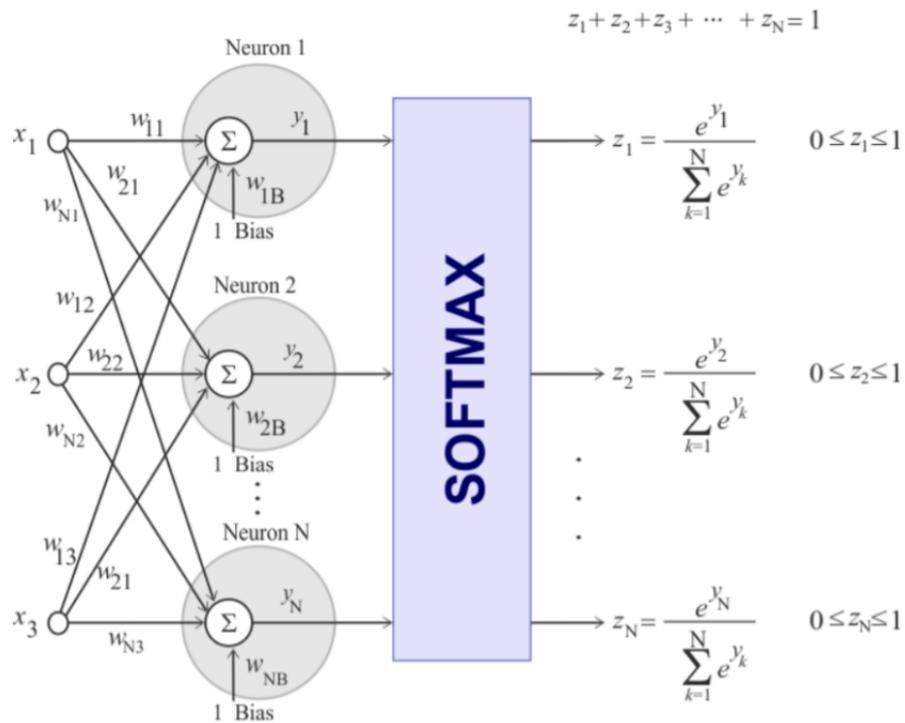


Figura 11: Diagrama de la función softmax. Recuperado de [35]

La función softmax es una generalización de la función logística que asigna un vector  $z$  de  $k$  dimensiones de valores reales arbitrarios a un vector  $\sigma(z)$  de  $k$  dimensiones de valores reales en el rango entre 0 y 1, de manera que la suma de sus valores es 1.

### 2.2.7 Función ReLU

La función ReLU (Rectified Lineal Unit) transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran. Está dada por [36]:

$$f(x) = \text{máx}(0, x) \quad (10)$$

Esta función transforma la entrada en cero si la entrada es menor que cero y la deja igual si es mayor a cero, su derivada es relativamente simple, pues es la misma pero en vez de dejar la entrada igual la hace uno [29]. Sus características son las

siguientes:

- Solo se activa si son positivos.
- No está acotada.
- Puede desconectar demasiadas neuronas en ciertos sistemas.
- Se comporta bien con imágenes.
- Buen desempeño en redes convolucionales.

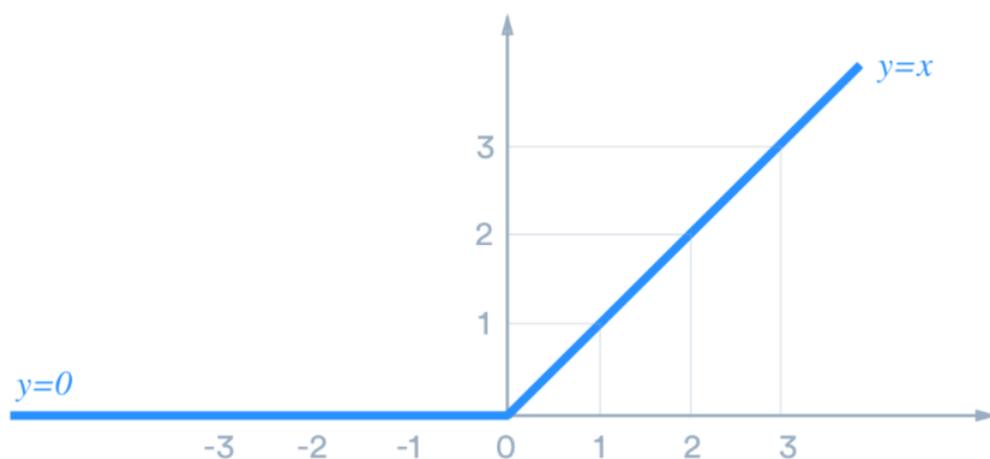


Figura 12: Diagrama de la función RELU. Recuperado de [36]

La función de activación lineal rectificadora se ha convertido rápidamente en la función de activación predeterminada al desarrollar la mayoría de los tipos de redes neuronales [36].

### 2.3 Entrenamiento de la red neuronal

Las redes neuronales, como se explicó anteriormente, generan vectores o valores de salida para determinados vectores o valores de entrada. Una vez que se define la arquitectura de la red neuronal se tienen que definir los valores de activación, los que a su vez definirán los estados de cada neurona dentro de la red [27].

Cada red neuronal es una aproximación de una función, por lo que cada red neuronal no será igual a la función deseada, pero en cambio diferirá por algún valor llamado error. Durante el entrenamiento, el objetivo es minimizar este error. Como el error es una función que depende de los valores de activación  $\omega$  de la red, el objetivo es minimizar el error con respecto a los valores de activación. La función de error es una función de muchas variables, teniendo  $n$  variables, siendo  $n$  el número de valores de activación de la red. Matemáticamente, el conjunto de puntos donde esta función es cero representa una hipersuperficie, y para encontrar un mínimo en esta superficie, queremos elegir un punto y luego seguir una curva que dirija la función de error al mínimo posible [29].

Existen varios métodos de optimización usados para minimizar la función de error, el más común es el método del gradiente descendente, el cual es explicado en la sección 2.3.2.

### 2.3.1 Función de error Cross-entropy

Al usarse una capa de salida con la función de activación softmax para la clasificación de un sistema con  $n$  clases, se obtiene una salida de la forma:

$$\vec{F}(x_i) = \frac{e^{x_i}}{\sum_{j=i}^n e^{x_j}} \quad (11)$$

Donde  $i, j = 0, 1, 2, \dots, n$  y  $x_i$  representa cada uno de los  $n$  valores reales correspondientes a las  $n$  clases mutuamente excluyentes [26]. La función colapsa cada salida en un valor entre 0 y 1, y la suma de todos los valores da 1, lo que puede ser interpretado como una distribución de probabilidad normalizada. La distribución de probabilidad real es, en la mayoría de los casos, un vector one-hot, lo que quiere decir que es un vector donde todos sus componentes son cero a excepción de uno, indicando que la probabilidad de que el vector corresponda a una clase en específico es del 100% y el 0% para las demás clases.

Entonces es óptimo usar una función de pérdida que compare la diferencia entre el valor estimado de la probabilidad correspondiente a cada clase y la distribución de

probabilidad real, esa diferencia es llamada cross-entropy [29]. La función de pérdida está definida de la siguiente manera:

$$H(p, q) = - \sum_{i=1}^n p_i \log(q_i(x)) \quad (12)$$

Aquí,  $q_i(x)$  es la probabilidad estimada de una de la salida  $i$ -ésima de la red de pertenecer a la clase  $i$  y  $p_i(x)$  es la probabilidad real de que pertenezca a esa clase. Cuando se usan los vectores one-hot, solo existe un valor de  $p_i(x)$  diferente de cero de los  $n$  elementos que tiene el vector, ese elemento es el correspondiente a la clase  $i$ . En este caso, la función solo calcula el error de la clase  $i$ -ésima y descarta las demás.

### 2.3.2 Gradiente descendente

El objetivo de los métodos de optimización, es encontrar los valores óptimos de los valores de activación que reduzcan el valor de la función de pérdida, de manera que algoritmo de la red neuronal pueda hacer mejores predicciones. Sabiendo que tanto difieren los resultados obtenidos de la red ( $y^i$ ) con los resultados esperados ( $t^i$ ) (mediante la función de pérdida  $H$ ), podemos suponer que cada valor  $y^i$  depende explícitamente de cada  $\omega_i$ , entonces  $H$  representa una hipersuperficie tantas dimensiones como valores de activación existen en el modelo. La proyección de esa hipersuperficie en uno de los valores de activación reflejaría una gráfica como la siguiente:

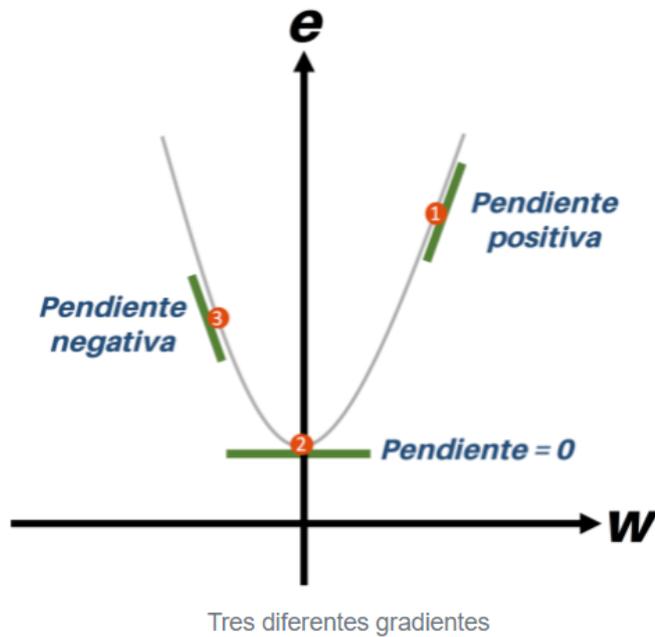


Figura 13: Representación gráfica del gradiente descendente. Recuperado de Sotaquirá, Miguel. (Julio de 2018). ¿Qué es el Gradiente Descendente?. codificandobits.com

Siendo  $e$ , una función de pérdida cualquiera. En el punto 1 se observa una línea con una inclinación correspondiente a un gradiente positivo, mientras que en el punto 3 la línea tiene una orientación opuesta y por tanto se dice que tiene una pendiente negativa. Por su parte, en el punto 2 la línea es totalmente horizontal y carece de inclinación, por lo que se dice que su pendiente es igual a cero, es en ese punto en donde se encuentra el valor óptimo de ese  $\omega$  en específico, el cual minimiza la función de pérdida.

Como el objetivo es minimizar esa función, en el caso de que fuera una variable, es posible derivarla para encontrar el valor de  $\omega$  donde la pendiente es cero, extrapolando a un sistema de muchas dimensiones, se usa el gradiente. El procedimiento mediante el cual el método encuentra ese valor óptimo es el siguiente:

- 1.- Se calcula la derivada parcial de la función con respecto de cada valor de activación.

$$\vec{d} = \frac{\partial e(\omega)}{\partial \omega_j} \quad (13)$$

2.- El resultado obtenido se multiplica por un coeficiente de aprendizaje (learning rate) definido por la letra griega  $\eta$ .

3.- Los valores de activación se actualizan de la siguiente manera:

$$\omega \rightarrow \eta \frac{\partial e(\omega)}{\partial \omega_j} \quad (14)$$

4.- Donde los valores de activación se actualizan y se generan nuevas salidas de la red neuronal.

5.- A partir de las nuevas salidas generadas se calcula de nuevo el error mediante la función de pérdida.

6.- El proceso se repite por iteraciones hasta alcanzar el valor de la función de pérdida dentro de los márgenes deseados.

Generalizando para un sistema de muchas dimensiones:

$$\omega \rightarrow \eta \nabla e \quad (15)$$

Es notorio que para actualizar los pesos, acumulamos el error en todas las muestras de entrenamiento. En realidad, hay grandes conjuntos de datos, y al iterarlos en una sola actualización haría que el entrenamiento fuera demasiado lento. Una solución a este problema es el algoritmo de descenso de gradiente lineal (SGD), que funciona de la misma manera como un descenso de gradiente regular, pero actualiza los pesos después de cada muestra de entrenamiento. Sin embargo, SGD es propenso al ruido en los datos. Si una muestra es un valor atípico, se corre el riesgo de aumentar el error en lugar de disminuirlo.

Además del mínimo global, la función de pérdida puede tener múltiples mínimos locales y minimizar su valor no es una tarea trivial.

El método descrito calcula el error y actualiza los pesos para una red neuronal de una sola capa, para redes multicapa es necesario aplicar un método adicional llamado back-propagation.

### 2.3.3 Back propagation

Los procesos por medio de los cuales el método del gradiente descendiente actualiza los valores de activación para redes neuronales multicapa cambia debido a que la función de pérdida no depende explícitamente de las salidas de las neuronas de las capas anteriores a la de salida. Para este tipo de redes el proceso comienza comparando la salida de la red neuronal con el valor esperado y actualizando los valores de activación esa capa, no existen valores esperados para las salidas de las neuronas de las demás capas, debido a esto, se debe propagar el error hacia atrás en la red neuronal, de ahí el nombre del método.

El método consiste en un algoritmo basado en la aplicación de la regla de la cadena [27]. Primero se definen los valores de activación entre la neurona  $i$ -ésima de la capa  $l$  con la neurona  $j$ -ésima de la capa  $l+1$  como  $w_{ij}$ . La salida de una neurona  $i$  de cualquier capa se define como  $y_i$  y la entrada a dicha neurona como  $a_i$ , siendo la salida de una neurona la entrada de otra. La entrada de la neurona es el producto:

$$\vec{a} = \vec{x} \cdot \vec{w}_{ij} \quad (16)$$

Estando en función de los valores de activación.

La función de pérdida, a cual puede ser cualquiera, se define como  $e$  y es una función que evalúa el error de cada neurona de la capa de salida en función de la salida de dicha neurona y de el valor esperado. Se aplica la regla de la cadena a la función de pérdida:

$$\frac{\partial e}{\partial \omega_{ij}} = \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial \omega_{ij}} \quad (17)$$

Por definición  $\frac{\partial a_j}{\partial \omega_{ij}} = y_i$ , lo cual se sustituye en la derivada de la función de pérdida:

$$\frac{\partial e}{\partial \omega_{ij}} = \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial a_j} y_i \quad (18)$$

Para capas ocultas de la red neuronal se usa la derivada de la función de activación de la capa donde se encuentra la neurona de la cual se quiere calcular la derivada del error, esta varía dependiendo de que sea elegida para cada capa. Solo se representará por  $\frac{\partial y_j}{\partial a_j}$ . Aplicando la regla de la cadena a cada función de activación es posible encontrar todas las derivadas, comenzando desde la última capa y moviéndose hacia atrás, debido solo podemos calcular esta derivada para la última capa en un principio pero tenemos una fórmula que nos permite calcular la derivada de una capa, asumiendo que podemos calcular la derivada para la siguiente. Generalizando la regla de la cadena para una capa oculta obtenemos:

$$\frac{\partial e}{\partial y_i} = \sum_j \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial y_i} = \sum_j \frac{\partial J}{\partial y_j} \frac{\partial y_j}{j} \frac{a_j}{\partial y_i} \quad (19)$$

La suma sobre  $j$  que se realiza la suma sobre todas las neuronas de la capa  $l+1$ . Eso es en el caso de redes con capas fully-connected, donde cada salida contribuye a  $y_i$  cuando se propaga el error.

Se puede calcular la derivada de cada función de activación con respecto de  $a_i$  y ya se conoce la derivada de  $a_j$  con respecto de la salida  $y_i$ , la cual es  $\omega_{ij}$ , entonces, conociendo la derivada de la función de pérdida con respecto de la última capa de la red, podemos calcular su derivada con respecto de una capa oculta, por lo tanto, también su derivada con respecto de cualquier valor de activación.

$$\frac{\partial e}{\partial \omega_{ij}} = \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial \omega_{ij}} \quad (20)$$

Se define:

$$\delta_j = \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial a_j} \quad (21)$$

Sustituyendo en la ecuación (18) se obtiene:

$$\frac{\partial e}{\partial y_i} = \sum_j \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial y_i} = \sum_j \frac{\partial e}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial y_i} = \sum_j \delta_j \omega_{ij} \quad (22)$$

Lo que implica que:

$$\delta_i = \left( \sum_j \delta_j \omega_{i,j} \right) \frac{\partial y_i}{\partial a_i} \quad (23)$$

Gracias a estas ecuaciones es posible calcular la variación para cada capa una vez que se calcule la variación para la capa posterior:

$$\delta_i = \left( \sum_j \delta_j \omega_{i,j} \right) \frac{\partial y_i}{\partial a_i} \quad (24)$$

Igualando lo anterior a la definición de  $\delta$  se obtiene:

$$\frac{\partial e}{\partial \omega_{i,j}} = \delta_j \frac{\partial a_j}{\partial \omega_{i,j}} = \delta_j y_i \quad (25)$$

La regla de actualización para los valores de activación de cada capa viene dada por la siguiente ecuación:

$$\omega_{i,j} \rightarrow \omega_{i,j} - \eta \delta_j y_i \quad (26)$$

Aplicando esa regla de actualización de manera iterada es como se encuentran los valores de activación que minimizan la función de pérdida.

### 2.3.4 Optimizador ADAM

La elección del algoritmo de optimización para el modelo de aprendizaje profundo puede significar la diferencia entre buenos resultados en minutos, horas y días [27].

El algoritmo de optimización de Adam es una extensión del descenso de gradiente que recientemente ha visto una adopción más amplia para aplicaciones de aprendizaje profundo en visión artificial y procesamiento de lenguaje natural [37].

Adam es un algoritmo de optimización que se puede utilizar en lugar del procedimiento de descenso de gradiente clásico para actualizar iterativamente los pesos de la red en función de los datos de entrenamiento. Adam fue presentado por Diederik

Kingma de OpenAI y Jimmy Ba de la Universidad de Toronto [37]. El nombre se deriva de su significado en inglés "adaptive moment estimation".

Las principales ventajas observadas al usar este método de optimización fueron:

- Fácil de implementar.
- Computacionalmente eficiente.
- Pocos requisitos de memoria.
- Invariante a la reescala diagonal de los gradientes.
- Muy adecuado para problemas que son grandes en términos de datos y/o parámetros.
- Apropiado para objetivos no estacionarios.
- Apropiado para problemas con gradientes muy ruidosos o escasos.
- Los hiperparámetros tienen una interpretación intuitiva y, por lo general, requieren pocos ajustes.

La forma en la que este optimizador funciona es diferente al descenso de gradiente. El descenso de gradiente mantiene una única tasa de aprendizaje denominada  $\alpha$  para todas las actualizaciones de peso y la tasa de aprendizaje no cambia durante el entrenamiento. Se mantiene una tasa de aprendizaje para cada valor de activación de la red y se adapta por separado a medida que se desarrolla el aprendizaje.

Adam es la combinación de las ventajas de otras dos extensiones del descenso de gradiente, las del algoritmo de gradiente adaptativo y la propagación cuadrática media, AdaGrad y RMSProp respectivamente por sus siglas en inglés.

Adagrad consiste en mantener una tasa de aprendizaje por parámetro que mejora el rendimiento en problemas con gradientes escasos (por ejemplo, problemas de lenguaje natural y visión por computadora). RMSProp mantiene tasas de aprendizaje por parámetro que se adaptan en función del promedio de las magnitudes recientes de los gradientes para el valor de activación. Esto significa que el algoritmo

funciona bien en problemas en línea y no estacionarios, como es el caso de las señales electromiográficas.

En lugar de adaptar las tasas de aprendizaje de parámetros en función del primer momento promedio como en RMSProp, Adam también utiliza el promedio de los segundos momentos de los gradientes. Específicamente, el algoritmo calcula un promedio móvil exponencial del gradiente y el gradiente cuadrático, y los parámetros  $\beta_1$  y  $\beta_2$  controlan las tasas de caída de estos promedios móviles.

El valor inicial de las medias móviles y los valores de  $\beta_1$  y  $\beta_2$  cercanos a 1,0 dan como resultado un sesgo de las estimaciones de momento hacia cero. Este sesgo se supera calculando primero las estimaciones sesgadas antes de calcular las estimaciones corregidas por sesgo [37].

## 2.4 Proceso de Regresión Gaussiano (GPR)

Los métodos tradicionales de regresión no lineal generalmente generan una función para ajustarse mejor al conjunto a un conjunto de datos. Sin embargo, puede haber más de una función que encaje los puntos de datos observados de una manera adecuada. Cuando la dimensión de un modelo de distribución gaussiana multivariable es infinito, es posible hacer predicciones en cualquier punto con estos números infinitos de funciones. La distribución previa de estas funciones infinitas es otra distribución gaussiana. La distribución previa representa las salidas esperadas de la función  $f$  sobre las entradas  $x$  sin haber observado ningún dato. Cuando se comienzan a tener cálculos, en lugar de números infinitos de funciones, solo mantenemos las funciones que se ajustan a los puntos de datos observados. Al generar nuevos cálculos o predicciones de la función  $f$ , los cálculos esperados serán ahora los nuevos cálculos y los esperados pasarán a ser los nuevos [38].

Un modelo de procesos gaussianos es una distribución de probabilidad sobre posibles funciones que se ajustan a un conjunto de datos. Porque tenemos la distribución de probabilidad sobre todas las funciones posibles, podemos calcular las medias como la función y las varianzas para indicar qué tan confiables son las predicciones. El

modelo consiste en que las actualizaciones de la función se generen a partir de nuevos cálculos. un modelo de proceso gaussiano es una distribución de probabilidad sobre posibles funciones, y cualquier muestra finita de funciones se distribuyen conjuntamente en una distribución gaussiana, de manera que la función media calculada por la distribución esperada de posibles predicciones para información desconocida de la función [39].

La función de regresión modelada para una distribución gaussiana multivariable está definida por:

$$P(f|X) = N(f|\mu, K) \quad (27)$$

Siendo  $X = [x_1, \dots, x_n]$ ,  $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]$ ,  $\mu = [m(\mathbf{x}_1), \dots, m(\mathbf{x}_n)]$  los datos observados,  $K_{ij}$  una función kernel y  $m$  la media, considerando que para un vector de datos observados sin elementos  $m(X) = 0$ . Entonces el proceso gaussiano es una distribución sobre funciones cuya forma o dimensión está definida por la función kernel. Si los elementos del vector  $X$  son considerados similares por el kernel, los cálculos de la función evaluada en esos elementos también será similar [41]. Lo que el modelo genera a partir de algunas observaciones y la función media  $f$ , es estimar predicciones para datos no observados con valores diferentes a los de las observaciones.

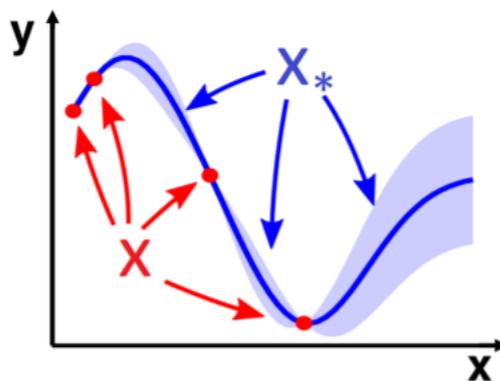


Figura 14: Diagrama de ejemplo para un proceso de regresión gaussiano. Recuperado de [41].

Los puntos rojos son datos observados, la línea azul representa la función media estimada por los puntos de datos observados, y las predicciones se realizarán en nuevos puntos azules.

La relación entre la distribución de las observaciones  $f$  y las predicciones  $f_*$  está dada por la siguiente ecuación:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right) \quad (28)$$

Siendo  $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$ ,  $\mathbf{K}_* = K(\mathbf{X}, \mathbf{X}_*)$ ,  $\mathbf{K}_{**} = K(\mathbf{X}_*, \mathbf{X}_*)$  y  $(m(\mathbf{X}), m(\mathbf{X}_*)) = 0$ .

La ecuación de la distribución de probabilidad de esta relación de distribuciones se define por  $P(f, f_* | X, X_*)$ , pero es necesaria una distribución condicional que solo dependa de  $f_*$ . Al derivar esta distribución condicional se obtiene una ecuación predicativa para el proceso de regresión gaussiano:

$$\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)) \quad (29)$$

donde

$$\begin{aligned} \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] \\ &= \mathbf{K}_*^\top [\mathbf{K} + \sigma_n^2 I]^{-1} \mathbf{y} \\ \text{cov}(\mathbf{f}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^\top [\mathbf{K} + \sigma_n^2 I]^{-1} \mathbf{K}_* \end{aligned}$$

Siendo  $\text{cov}(f_*)$  la función de varianza.

Una propiedad resaltable de la distribución gaussiana es que la función de varianza no depende de  $X$  ni de  $X_*$ .

## **3 Metodología utilizada**

### **3.1 Análisis del banco de señales**

Las señales son series de frecuencias igualmente espaciadas en un intervalo de tiempo de dieciséis segundos y de las cuales la información importante se distribuye en cinco intervalos de aproximadamente dos segundos durante los cuales se produce el evento y que no necesariamente están igualmente espaciados. Un evento representa la sección de la señal durante la cual el sujeto de pruebas realiza el movimiento indicado, es esta sección de la señal la que contiene los patrones que se busca clasificar.

De esta manera al realizar un experimento que consiste en la toma de las señales para las dos manos y para los diez movimientos en un sujeto de prueba se obtienen 3200 datos por cada uno de los ocho canales que corresponden a los ocho electrodos conectados al brazo del sujeto. El total de experimentos realizados fue de 1000, 20 por cada uno de los cincuenta sujetos de prueba.

Cada uno de los diez movimientos presenta patrones diferentes para cada uno de los canales de señal que contiene, estos canales presentan diferentes patrones entre sí aún tratándose de un mismo movimiento. Como se explicó en la sección de señales electromiográficas, esto se debe a que los impulsos nerviosos admitidos en diferentes músculos no necesariamente tienen que ser iguales o parecidos.

#### **3.1.1 Análisis cualitativo de los canales de cada movimiento**

Las diferencias entre las señales de cada canal varían dependiendo del movimiento, empezando por el de posición inicial, las variaciones en las amplitudes son prácticamente inexistentes como se puede apreciar en la siguiente figura:

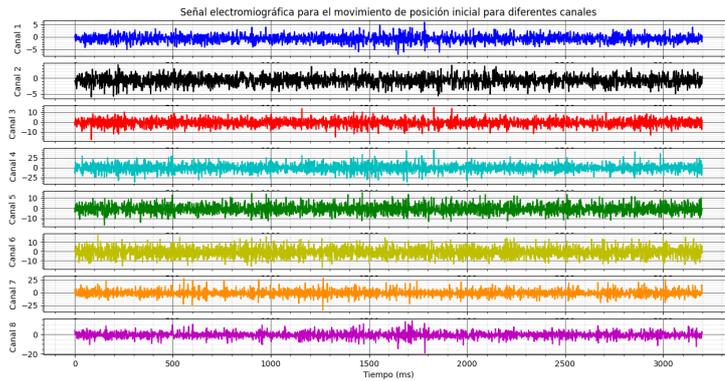


Figura 15: Señales de los ocho canales producidas por el movimiento de posición inicial.

El movimiento de posición inicial es un caso bastante particular, debido a que no existen eventos de interés en la señal, esto es debido a solo se midieron las señales para la mano en reposo para tener una referencia de como se debería de comportar la señal entre eventos para los demás movimientos. Debido a esa propiedad de este movimiento, la señal no muestra una gran variabilidad como lo hacen los demás movimientos.

Igual es muy importante destacar que los canales que presentan amplitudes mayores son el canal 4 y el canal 8, mientras que las amplitudes más bajas son para el canal 1 y canal 2. Esto fue observado en señales producidas para el mismo movimiento en diferentes sujetos de prueba.

En el movimiento de pronación se pueden apreciar los eventos a diferencia del canal 1, los canales con menores amplitudes entre eventos siguen siendo los primeros dos, sin embargo el canal con mayores amplitudes es ahora el canal 7.

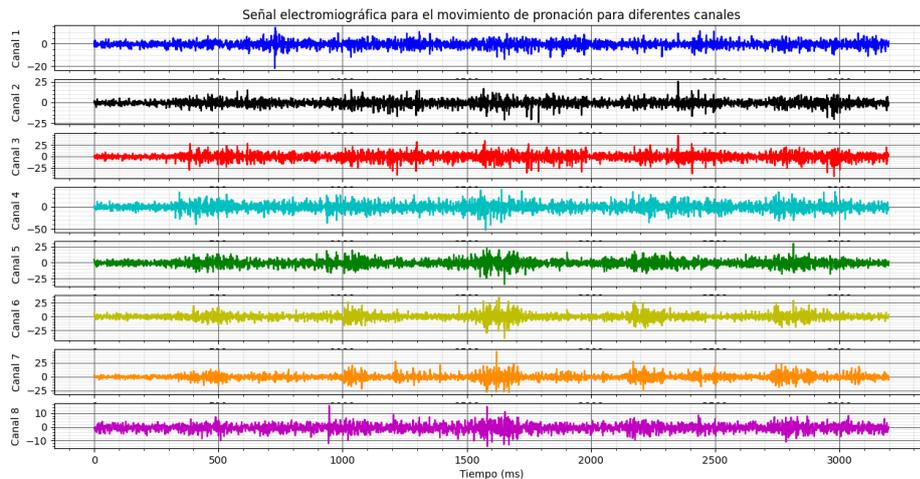


Figura 16: Señales de los ocho canales producidas por el movimiento de pronación.

Sin embargo la señal en los intervalos entre eventos se asemeja mucho a la del canal 1.

Para los canales 3 y 4, las amplitudes durante los eventos sobresale más, haciéndolos más notorios, las amplitudes entre eventos sigue asemejándose a las señales en la posición inicial y esto es algo que se repite para todos los movimientos.

Las menores amplitudes siguen siendo para los primeros dos canales, pero las más grandes son las del canal 4. Las variaciones durante los eventos para el canal 4 y 5 son muy pequeñas.

Para el movimiento de extensión los eventos parecen tener una réplica en los canales 1, 7 y 8. al momento de realizar este movimiento, es un patrón que no se repite para otros canales tratándose aún del mismo sujeto de pruebas. En otras pruebas se observa un desfase entre los eventos de interés de otros canales con el canal 8 y el canal 8.

Para ambos movimientos de desviación, en el canal 5 es donde menos se aprecian variaciones de la amplitud, los demás canales conservan una similitud en las secciones entre eventos con la señal de posición inicial.

La desviación estándar del canal 3 y el canal 4 también es menor en este movi-

miento de desviación cubital.

A diferencia de la desviación cubital, para la desviación radial hay un cambio relativamente grande de las amplitudes para la mayoría de los canales, sobre todo del canal 2 y 5.

Los movimientos de pinza son los que reflejan un mayor cambio en las amplitudes. Ambos movimientos llegan a presentar desfases, se observaron principalmente en los primeros dos canales.

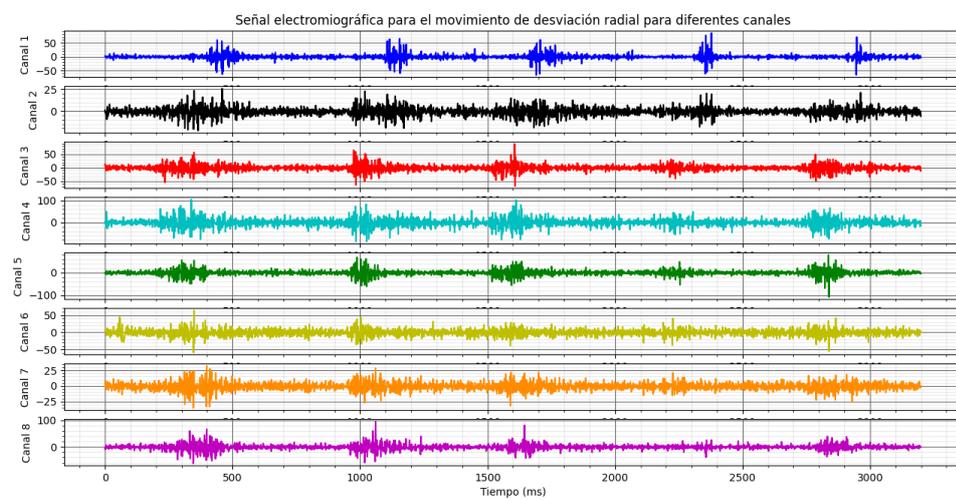


Figura 17: Señales de los ocho canales producidas por el movimiento de pinza fina.

El movimiento de pinza gruesa muestra replicas en el canal 3 aparte de los desfases de los primeros dos canales. El primer canal de estos movimiento de pinza también llega a presentar desfases en el evento de interés.

### 3.1.2 Valores estadísticos de la base de datos

Los valores de interés para cálculos necesarios para el reescalado de las señales son el valor máximo y el valor mínimo de el conjunto de datos en su totalidad, la media de los valores, esta se calcula mediante la fórmula:

$$\mu = \frac{\sum_i^n x_i}{n} \quad (30)$$

Siendo  $n$  el número total de datos y  $x_i$  el elemento  $i$ -ésimo del conjunto de datos.

También se calculó la desviación estándar promediando la desviación estándar de cada vector contenido en el banco de señales mediante la siguiente ecuación:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (31)$$

Dichos valores fueron encontrados mediante comandos de python de la librería numpy (véase Anexo 1). Los resultados fueron los siguientes:

Elemento del conjunto de datos	Amplitud medida
Valor máximo	213.6913
Valor mínimo	-214.2656
Media	-0.4407
Desviación estándar	9.2614

Cuadro 1: Coeficientes usados en la normalización de la base de datos.

Estos datos corresponden a la amplitud media, a su valor máximo y mínimo y la desviación estándar de todos los datos del banco de señales.

Calculando estos parámetros por movimiento se pueden apreciar características específicas de cada uno:

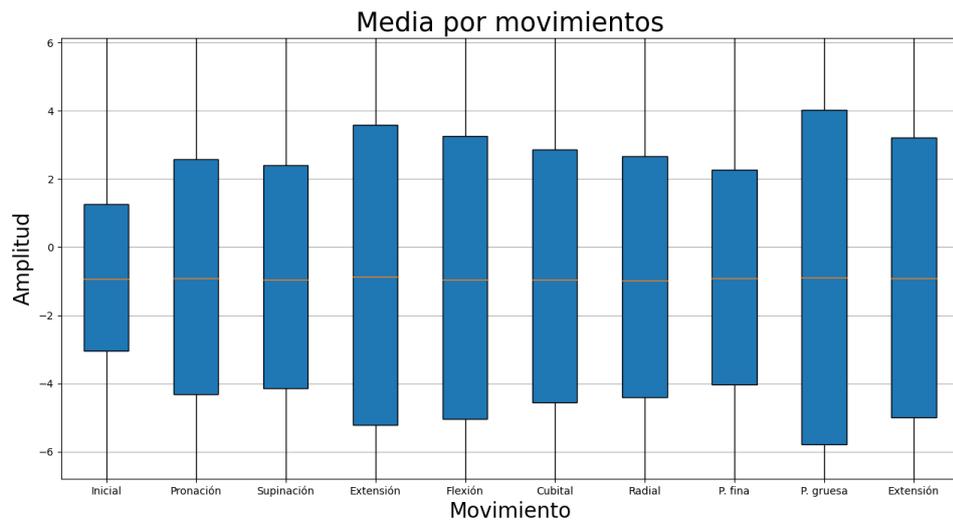


Figura 18: Media y desviación estándar por movimiento.

Como era de esperarse, el primer movimiento es en el que se muestran menos fluctuaciones de la amplitud al momento de hacer los experimentos, siendo os movimientos de pinza y el de extensión en los que se muestra una mayor desviación estándar de los datos. Las medias son relativamente similares, todas cercanas al cero y parecidas a la media de todo el banco de señales.

Calculando estos parámetros pero ahora por canal se obtiene lo siguiente:

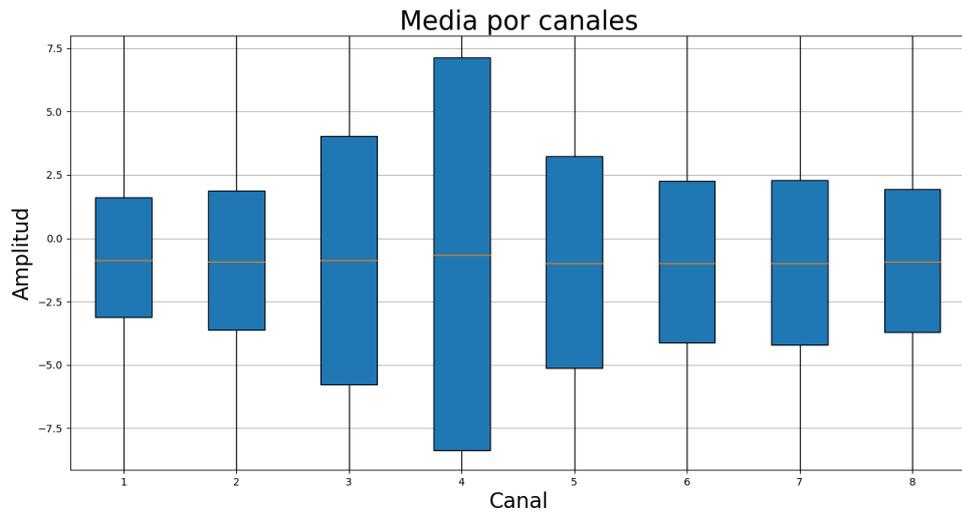


Figura 19: Media y desviación estándar por canal .

De igual manera que en la figura anterior, las medias son bastante similares, mientras que la desviación estándar depende de la posición en la que se coloca el electrodo al momento del experimento siendo el canal 4 el que presenta más fluctuaciones en la amplitud y el canal 1 el que menos presenta.

## 3.2 Pre-procesamiento del banco de señales

### 3.2.1 Segmentación de las señales

Debido a la naturaleza de las frecuencias en el dominio de su amplitud, para excluir información menos relevante, cada canal de señal se dividió en segmentos de dos segundos y luego se seleccionaron los cinco segmentos con la desviación estándar más alta para encontrar los segmentos que corresponden a cada evento. Gráficamente, el segmento de cada vector donde ocurre un evento luce de la siguiente manera:

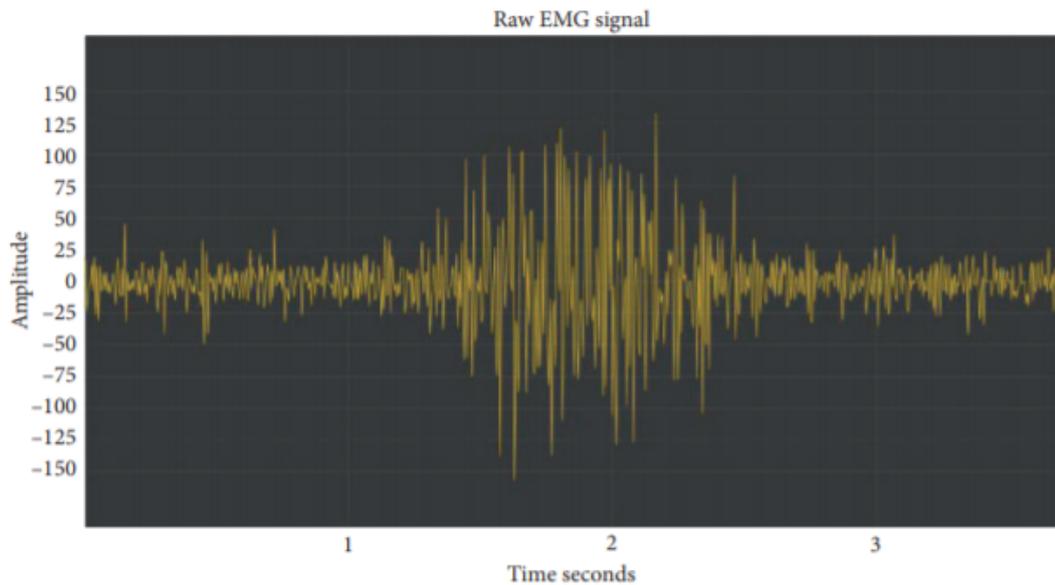


Figura 20: Ejemplo de la sección de la señal donde ocurre un evento. Recuperado de [24]

El centro de la señal pasa de ser un valor dependiente de la medición de los electrodos a ser el cero, con la finalidad de que la amplitud máxima y la mínima de cada señal tiendan a ser equidistantes del cero y hacer más práctico el sistema. A partir de los segmentos de dos segundos con más desviación estándar, se agrega medio segundo más en los extremos para ampliar el intervalo.

### 3.2.2 Reescalado de la base de datos

Se tomó la decisión de estandarizar los datos, lo que significa escalar los datos a un pequeño intervalo específico. El propósito general es eliminar el límite unitario de los datos y convertirlo en un valor puro de la dimensión, de modo que los indicadores de diferentes unidades u órdenes de magnitud puedan ponderarlo. En particular los datos fueron normalizados, lo que significa que cada uno de los elementos de las matrices que conforman el banco de señales se convirtieron en valores entre cero y uno de manera que no se pierda la correlación ya existente entre ellos.

La ventaja de la normalización es que en un sistema de evaluación de índices múltiples, debido a la naturaleza de cada índice de evaluación, generalmente tiene

una dimensión y un orden de magnitud diferentes. Cuando los niveles de los diversos indicadores son muy diferente, si los valores del indicador original se utilizan directamente para el análisis, resaltará el papel de los indicadores con valores más altos en el análisis integral y debilitará relativamente el papel de los indicadores con valor bajo. niveles. Por lo tanto, para garantizar la confiabilidad de los resultados, es necesario estandarizar los datos originales.

En el caso de las señales electromiográficas había partes de una misma señal con amplitudes prácticamente de cero y otras secciones con amplitudes demasiado altas.

Empíricamente, la normalización consiste en hacer que las características entre diferentes dimensiones tengan un cierto grado de comparación numérica, lo que puede mejorar en gran medida la precisión del modelo de clasificación. [26]

El método seleccionado para normalizar los datos es el de MinMax scaler, fue seleccionado debido a que fue observado en las señales que es muy común obtener amplitudes atípicas. En el enfoque del método MinMax scaler, los datos se escalan de tal manera que los valores generalmente oscilan entre 0 y 1. En contraste con la estandarización, el método da como resultado desviaciones estándar más pequeñas. Lo que esto significa esencialmente es que estaremos suprimiendo los efectos de los valores atípicos.

La normalización mediante este método es en realidad bastante sencilla, otra de las cualidades por las que fue seleccionado. Consiste en generar un conjunto de datos normalizado a partir del valor máximo y el valor mínimo del conjunto de datos, poniendo estos como cotas superior e inferior, respectivamente, del conjunto normalizado. La ecuación que representa el método fue la siguiente:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (32)$$

Siendo  $X_{norm}$  el conjunto de datos normalizado,  $X_{min}$  el valor mínimo del conjunto de datos original y  $X_{max}$  su valor máximo.

Recuperando los coeficientes correspondientes al valor máximo y mínimo de la Tabla 1, mediante un programa se le aplicó la fórmula a cada uno de los datos

para colapsar cada valor a uno en el intervalo entre 0 y 1. Estas series de datos correspondían a un canal de un movimiento de un sujeto usando una de sus manos y eran representados por vectores unidimensionales, la correlación entre un vector y otro o de una matriz a otra de información no se perdían debido a que los valores máximo y mínimo eran globales, es decir el valor máximo y el valor mínimo de todo el banco de señales.

De esta manera se reescaló la información, logrando obtener un conjunto una banco de señales normalizado, en la siguiente gráfica se muestra una comparación entre la señal antes y después de ser reescalada a los valores entre 0 y 1.

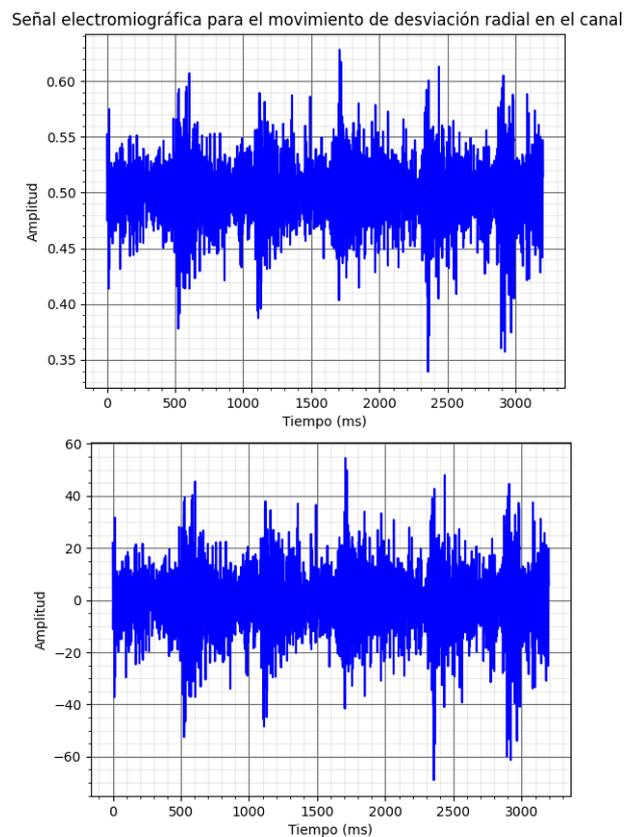


Figura 21: Comparación entre la señal generada en el canal 3 del movimiento de desviación radial antes y después de ser normalizada.

Como se puede observar los patrones de las gráficas son prácticamente iguales, solo cambia la escala, eso indica que se conserva la correlación entre la información

existente antes de normalizar los datos.

Otra alternativa probada para el reescalado del banco de señales fue mediante un método llamado z-score, al usar este método, las características se escalan de tal manera que terminan teniendo propiedades de una distribución normal estándar con una media igual a cero y una desviación estándar de uno. Simplemente calculamos el z-score de cada observación en el conjunto de datos para la característica. Un z-score se calcula utilizando la siguiente fórmula:

$$Z = \frac{x_i - \mu}{\sigma} \quad (33)$$

Siendo  $x_i$  uno de los valores del banco de señales,  $\mu$  el valor medio del banco de señales y  $\sigma$  su desviación estándar.

Se observó que el método también maneja valores atípicos como el MinMax scaler, pero no produce datos normalizados con exactamente la misma escala, lo que conlleva una pérdida en la correlación entre los patrones de amplitudes de las señales electromiográficas.

En la siguiente gráfica se puede apreciar una comparación entre una señal antes y después de ser reescalada:

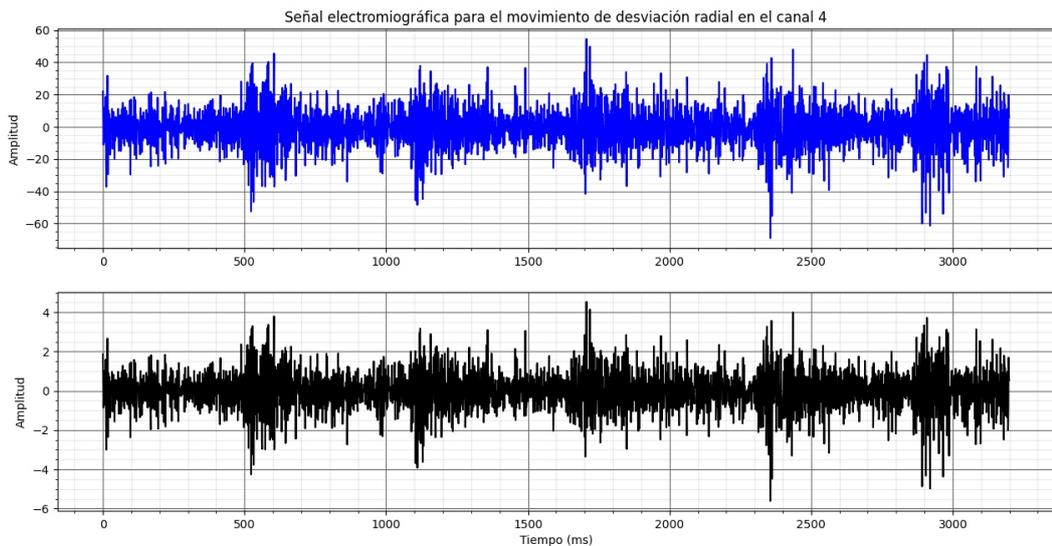


Figura 22: Comparación entre la señal generada en el canal 3 del movimiento de desviación radial antes y después de ser reescalada.

Aunque la diferencia pareciera ser insignificante, la diferencia en la desviación estándar de las señales presentó una variación de 0.0070, lo que implica que no se conserva la correlación de la información en su totalidad.

Aparte al reescalar mediante z-score no se obtiene un conjunto de datos en un rango de amplitudes entre 0 y 1, es por ese motivo que se optó por el primer método.

### 3.3 Propuesta de red neuronal convolucional

#### 3.3.1 Arquitectura de la red neuronal convolucional

Las diferentes capas que componen el modelo son:

- Capa de entrada: Esta capa consiste en la información que alimentará al modelo para entrenarlo, validarlo y probarlo. Esta información se almacena en el banco de señales. El banco de señales se divide en un conjunto de entrenamiento, un conjunto de validación y un conjunto de prueba, estos conjuntos se normalizan y escalan a valores entre -1 y 1 utilizando la fórmula de normalización MinMaxScaler, donde el nuevo conjunto escalado será.
- Capas ocultas: en cualquier red neuronal, las capas intermedias se denominan ocultas porque sus entradas y salidas están enmascaradas por la función de activación y la convolución final. En una red neuronal convolucional, las capas ocultas incluyen capas que realizan convoluciones. Estas capas son la parte del modelo que busca los patrones que muestran la relación entre la señal de cada canal y un movimiento específico, la información de esas señales proviene de la capa de entrada. Por lo general, esto incluye una capa que realiza un producto escalar del kernel de convolución con la matriz de entrada de la capa. Este producto suele ser el producto interno de Frobenius, y su función de activación es en este caso ReLU [29].
- Dropout: Para este modelo, es una capa de abandono entre la primera capa oculta y la segunda. La capa Dropout es un filtro que anula la contribución de

algunas neuronas hacia la siguiente capa y deja sin modificar todas las demás, esa contribución anulada es del 30

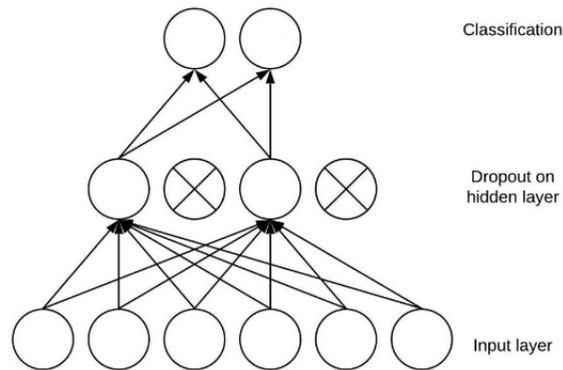


Figura 23: Diagrama de capa de dropout. Recuperado de [29].

- Capas de agrupación: Las capas de agrupación reducen las dimensiones de los datos al combinar las salidas de los grupos de neuronas en una capa en una sola neurona en la siguiente capa.
- Capa de salida: La capa de salida predice los datos según el modelo aprendido en las capas ocultas ya partir de su última capa se estima la pérdida.

Para la arquitectura de la red se inicia con hiperparámetros de la arquitectura que son comunes en el tipo de red, Las primeras capas cuenta con potencias de dos como número de filtros que va decreciendo conforme más profunda sea la capa, esto es lo más común al elegir los valores debido a que se busca llegar a un número de filtros cada vez menor antes de llegar a la capa de salida. La tercera capa tiene 100 neuronas para tener un una potencia del número de salidas que tiene el modelo. Hay 10 salidas cada una correspondiente a la probabilidad de que la entrada corresponda a uno de los diez movimientos. Estos se cambian a medida que se ve que hay una mejora en la red con respecto a los parámetros establecidos al inicio.

Capa	Filtros/neuronas	Función de activación
Primera capa	128	ReLU
Segunda capa	64	ReLU
Tercera capa	100	Sigmoid
salida	10	Softmax

Cuadro 2: Hiperparámetros del modelo de red neuronal convolucional

Es fundamental utilizar funciones de activación de unidades no lineales, la principal ventaja de utilizar la función ReLU frente a otras funciones de activación es que no activa todas las neuronas a la vez. Luego se cambia la función de activación en la capa densa debido a que la red neuronal está prediciendo una probabilidad, es necesario obtener valores entre 0 y 1, para ello se utiliza la función sigmoide. La capa de salida tiene que tener diez salidas correspondientes a las diez probabilidades de que la entrada pertenezca a una de las diez clases, como estas probabilidades tienen que ser normalizadas, se usa la función softmax [29].

Las dos primeras capas convolucionales tienen un tamaño de kernel de  $3 \times 3$ , porque para un filtro de tamaño impar, todos los filtros de capa anteriores estarían simétricamente alrededor del filtro de salida. Sin esta simetría, tendremos que tener en cuenta las distorsiones en el capas que suceden cuando se usa un kernel de tamaño uniforme, además de que CNN ha mostrado mejores resultados cuanto más pequeño es el tamaño del kernel. así que seleccionamos el tamaño de kernel más pequeño e impar diferente a  $1 \times 1$ . Entre esas capas se aplica un dropout para aumentar la precisión de la validación, los valores de salida se cambian y se seleccionan en la optimización de hiperparámetros.

Las siguientes capas cambian de dimensión a una capa de cien neuronas, cada neurona usa como función de activación la función sigmoide, esto es con la finalidad de que los valores se encuentren en el intervalo entre cero y uno al momento de entrar a la capa de salida. En la capa de salida se usa la función softmax para generar una distribución de probabilidad de la cual poder calcular la pérdida.

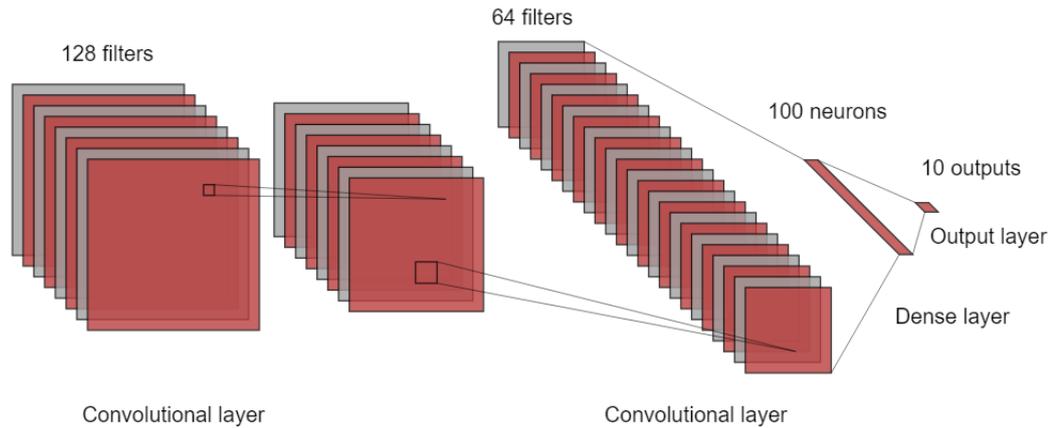


Figura 24: Diagrama de la red neuronal convolucional.

### 3.3.2 Entrenamiento de la red neuronal convolucional

Como la función de activación de la última capa nos da valores de entrada en el intervalo  $(0, 1)$  con la propiedad de que este vector con datos de salida está normalizado. Podemos interpretar las salidas softmax como una distribución de probabilidad de las diez clases normalizadas. Entonces, resulta útil usar una función de pérdida que compara la diferencia entre las probabilidades de clase estimadas y la distribución de clases real (la diferencia se conoce como cross-entropy).

La distribución de probabilidad esperada para cada entrada es un vector llamado one-hot, estos vectores sólo contienen información en el elemento correspondiente a la clase del vector (todos los elementos del vector son cero excepto el que corresponde a la clase del vector, en ese caso es uno), donde la clase real tiene una probabilidad de 1 y todos los demás tienen una probabilidad de 0. La función de pérdida que hace esto se llama Cross-Entropy Loss function (función de pérdida de entropía cruzada) y está dada por la ecuación (12).

Donde  $q(x)$  es la probabilidad estimada de que la salida pertenezca a la clase  $i$ -ésima (de 10 clases totales) y  $p(x)$  es la probabilidad real. Cuando usamos valores esperados one-hot para  $p(x)$ , solo la clase objetivo tiene un valor distinto de cero y todas las demás son ceros. En este caso, la pérdida solo capturará el error en la clase esperada y descarta todos los demás errores.

Si el valor de la precisión o la función de pérdida no son los esperados se usan buscan nuevos hiper parámetros mediante GPR y se vuelven a calcular las métricas de calidad.

El algoritmo mediante el que se puede expresar el modelo de red neuronal programado en python (véase anexo 2), se muestra en el siguiente diagrama:

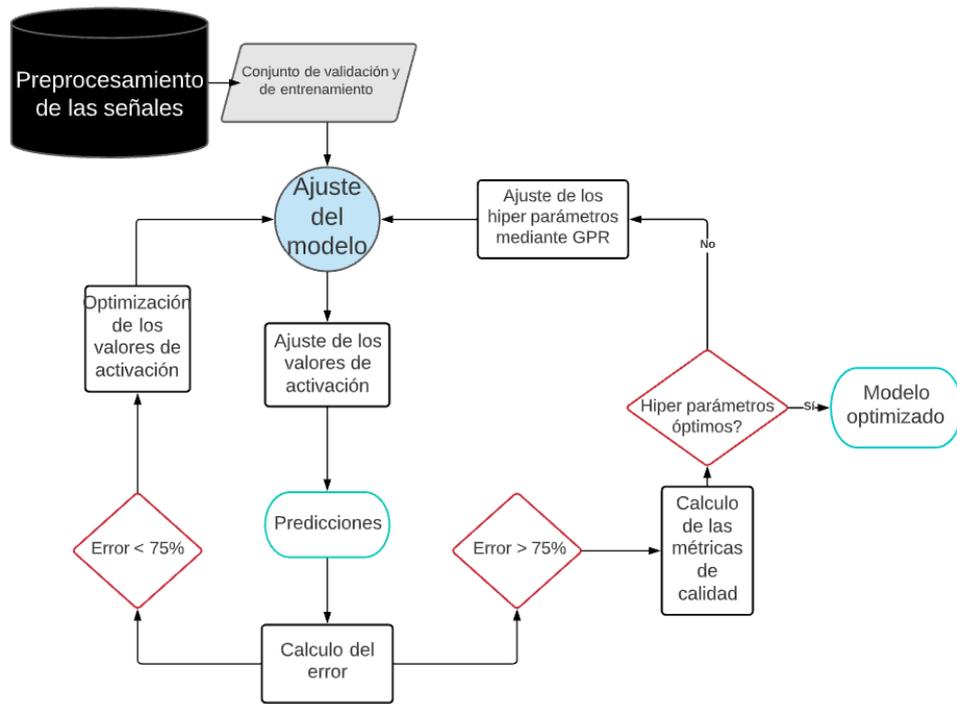


Figura 25: Diagrama de flujo del algoritmo de entrenamiento de la red neuronal convolucional.

El algoritmo consiste en generar diferentes conjuntos o sets de datos a partir del banco de señales previamente pre-procesado, para generar estos conjuntos primero se reordenan en las matrices que contienen las señales de un mismo canal y para un mismo movimiento, luego de la segmentación se obtiene una matriz donde cada vector contiene un evento de interés y una sección de la señal correspondiente a la mano en reposo, las dimensiones de estas matrices corresponden a la entrada a la red neuronal, entonces cada entrada a la red es una matriz de  $8 \times 640$ .

Cada conjunto de datos contiene el siguiente número de matrices o porcentaje del banco de señales:

Conjunto	Porcentaje del banco de señales	Número de datos que contiene
De entrenamiento	53.6 %	2573
De validación	26.4 %	1267
De prueba	20 %	960

Cuadro 3: Tamaño de los conjuntos de datos.

El primer conjunto de datos se usa para que la red ajuste los valores de activación mediante el método del gradiente descendiente o el método ADAM y que la salida de la red genere predicciones que se aproximen a los valores esperados, la medida de que tan próxima es la salida al valor esperado se define como error. El error se está dado por la función cross-entropy explicada en la sección de Antecedentes.

Si el error es superior al 75 % entonces los valores de activación se acuatizan de nuevo en otra iteración, una vez que se alcanzan valores menores al 75 % se calculan las métricas de calidad, se busca saber el número de veces que la red neuronal arroja verdaderos positivos (sus siglas en inglés son TP), falsos positivos (sus siglas en inglés son FP), verdaderos negativos (sus siglas en inglés son TN) y falsos negativos (sus siglas en inglés son FN) para cada uno de los movimientos, una vez con estos datos podemos comparar que tan específica, exacta y sensitiva es la red para comparar estos resultados con la precisión de la red. Los valores de estas métricas de calidad se calculan de la siguiente manera:

$$\text{sensitividad} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (34)$$

$$\text{especificidad} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (35)$$

$$\text{exactitud} = \frac{\text{TP} + \text{TN}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \quad (36)$$

Una vez alcanzadas unas métricas de calidad óptimas, con resultados con altos valores de exactitud y precisión se considerará que se tiene un modelo de red neuronal

terminado. de otra forma se buscarán hiperparámetros de la red que generen mejores resultados.

### 3.3.3 Optimización de los hiperparámetros de la red

En el caso de que sea necesario cambiar los hiperparámetros de la red se busca una combinación óptima de estos mediante el proceso de regresión gaussiano descrito en la sección de Antecedentes, donde las variables de la ecuación (25) se definen para el problema de estudio. Siendo  $X$  son los diferentes hiperparámetros ( $X = (x_1, \dots, x_n)$ ) cuyas variables son la tasa de aprendizaje, el valor de dropout, el tamaño del batch y el valor de división para el conjunto de validación y  $f$  es la precisión de las predicciones del modelo para los hiperparámetros ( $f = (f(x_1), \dots, f(x_n))$ ),  $\mu = m(x_1), \dots, m(x_n)$ , siendo  $m$  la función media,  $K_{ij} = k(10^{-3}, 10^3)$  representa una función kernel definida positivamente, el modelo de procesos gaussianos es una distribución sobre funciones cuya forma está definida por este valor. Como habíamos planteado anteriormente, si los puntos  $x_i$  y  $x_j$  que representan dos valores diferentes para un mismo hiperparámetro se consideran similares, se espera que la exactitud de los dos puntos,  $f(x_i)$  y  $f(x_j)$ , sean similares también.

Las regresiones obtenidas por el método se ilustran en las figuras 33-36. Dados los datos observados (puntos rojos) y una función media  $f$  (línea azul) estimada por estos puntos de datos observados, se generan nuevas predicciones (línea azul).

Las siguientes gráficas fueron generadas mediante un programa de python siguiendo el método descrito en la sección de Proceso de Regresión Gaussiano (véase Anexo 4).

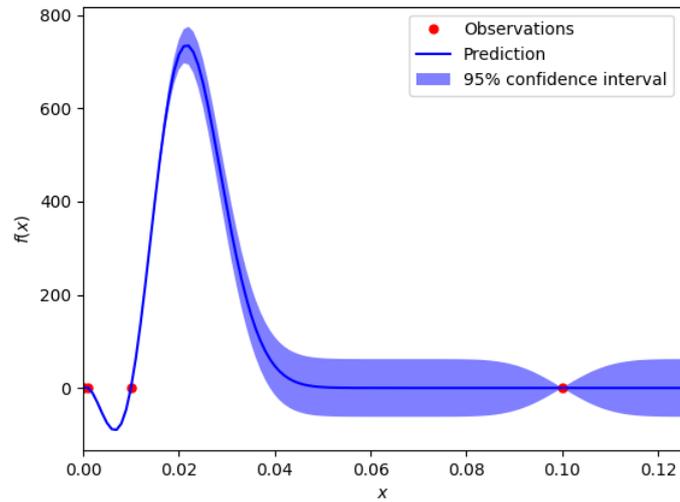


Figura 26: Proceso de gaussiano con la tasa de aprendizaje como variable.

Los valores de la tasa de aprendizaje tienen la restricción de ser potencias negativas de diez, mientras que el tamaño del batch tiene la restricción de ser potencias de dos.

La proyección de la función en el eje de la tasa de aprendizaje muestra valores superiores a 1, y no tiene sentido debido a que la máxima precisión es uno, pero debido a las restricciones de los posibles valores de la tasa de aprendizaje, la función evaluada en todos los valores permitidos es menor que uno.

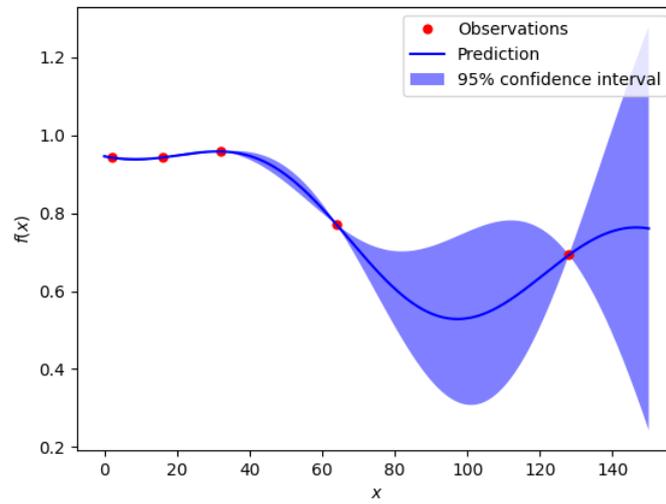


Figura 27: Proceso de gaussiano con el tamaño del batch como variable.

El valor del dropout y de la división de validación no tienen restricciones.

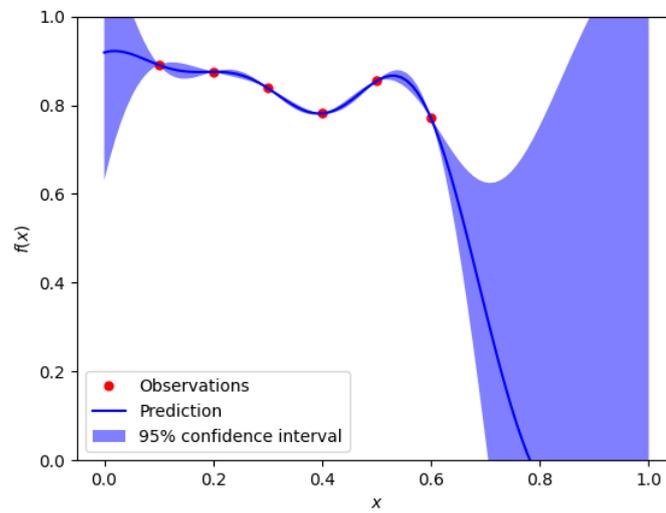


Figura 28: Proceso de gaussiano con el valor de dropout como variable.

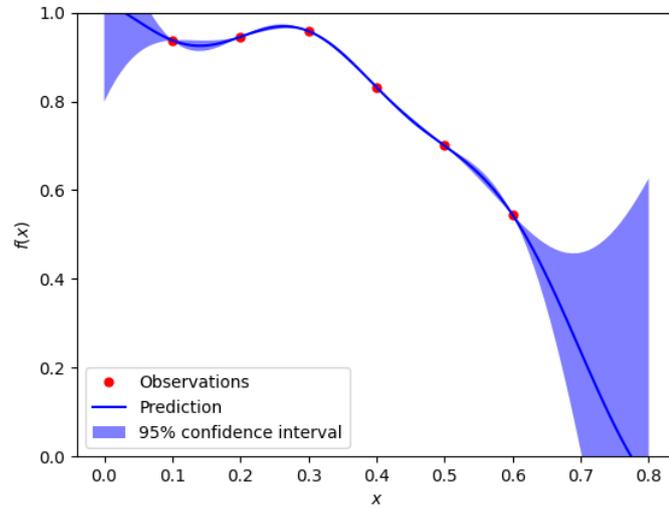


Figura 29: Proceso de gaussiano con el valor de división del conjunto de validación como variable.

Para la generación de las gráficas se tomaron como constantes las demás variables exceptuando la que se indica que se considera variable, los valores de las demás variables para cada caso son las siguientes:

Hiperparámetro	Valor constante
Tasa de aprendizaje	0.0001
Dropout	0.2
tamaño del batch	64
División de validación	0.3

Cuadro 4: Valores constantes de los hiperparámetros

## 4 Resultados y discusión

### 4.1 Valores óptimos de los hiperparámetros

Durante el proceso de entrenamiento se observó que el error y la el error de validación disminuyen al mismo tiempo, mientras que los valores de validación pueden diferir de los valores de entrenamiento si la división de validación es demasiado pequeña, y el punto de convergencia puede cambiar si cambia la tasa de aprendizaje, para valores altos podría ocurrir un sobre aprendizaje que podría hacer que la precisión converja antes y para valores bajos la exactitud y la pérdida converjan después. De manera similar, la precisión del modelo disminuye a medida que el valor de dropout se acerca a uno o para valores cercanos al cero. Y no alcanzará su máximo para valores diferentes de 32 para el tamaño del batch.

Luego de nuevo procesos iterativos para encontrar el valor de la optimización del modelo, se da un máximo de la función con los siguientes valores:

Hiperparámetro	Valor correspondiente a la máxima exactitud alcanzada
Tasa de aprendizaje	0.0001
Dropout	0.15
Tamaño del batch	32
División de validación	0.38

Cuadro 5: Valores óptimos de los hiperparámetros

Las curvas de las gráficas muestran valores más altos de la exactitud del modelo para esas proyecciones en específico, sin embargo, es la combinación de estos valores de los hiperparámetros la que genera una mayor exactitud y que cumple con las restricciones de cada variable.

## 4.2 Comparación de los diferentes métodos de optimización de los valores de activación

Como se mencionó, se usaron dos métodos para la optimización de los valores de activación de la red, estos fueron el método del gradiente descendiente y el optimizador ADAM. Tal como en la literatura, el optimizador ADAM resulto eficiente en términos computacionales y a pesar de la naturaleza no estacionaria de la base de datos, la convergencia tanto del error como de la exactitud de la red neuronal no se vio afectada.

Usando este método, la exactitud del entrenamiento y la exactitud de validación disminuyeron al mismo tiempo, se observó que esto no sucede si las entradas no están correctamente reescaladas o si tienen ruido. Durante 25 iteraciones, con el conjunto de hiperparámetros proporcionado por GPR, se obtuvo una precisión promedio de 94.95 % con una precisión de validación de 95.49 %.

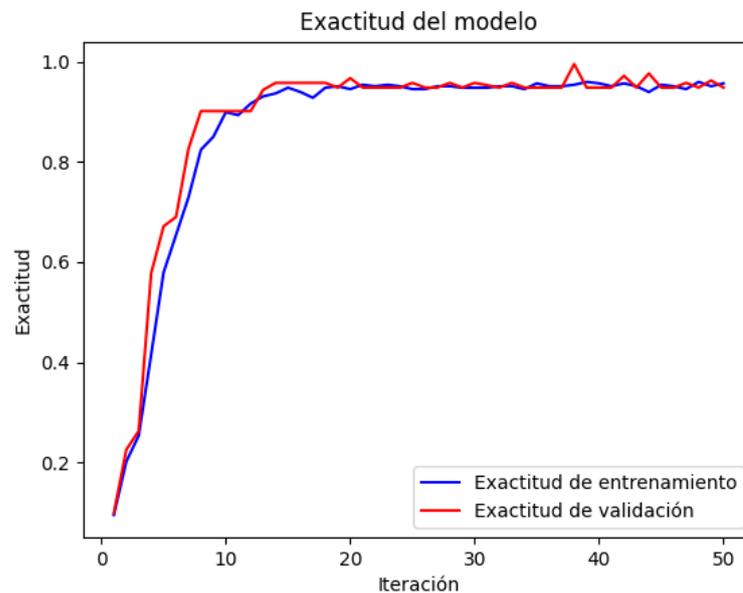


Figura 30: Exactitud promedio del modelo.

Como se puede apreciar, el valor de ambas líneas tiende a un valor constante cercano al 1 a partir de la iteración 12, ambas líneas tienen comportamientos bastante similares para las mismas etapas de entrenamiento.

Los valores alcanzados para la última iteración representa un ajuste adecuado de los datos de entrada con los valores esperados por parte de la red.

Mientras que el error medio para el mismo número de iteraciones se redujo al 19,43% con un error de validación del 18,54%.

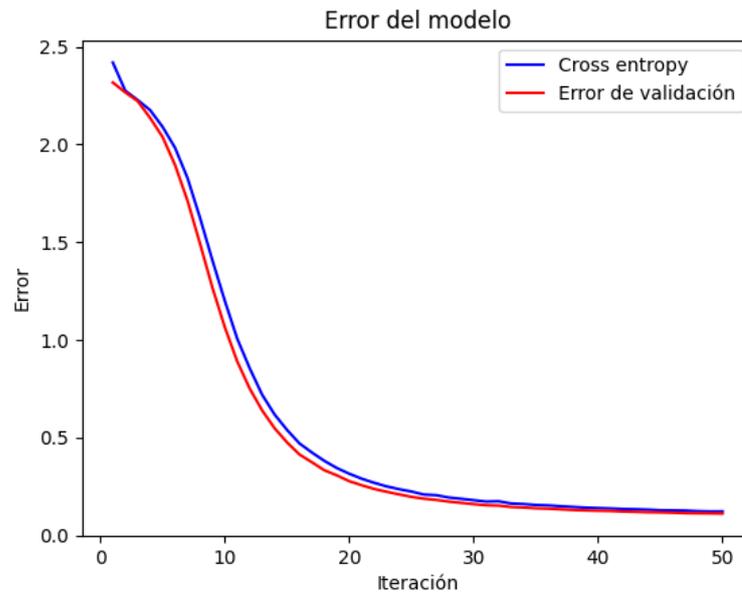


Figura 31: Error promedio del modelo.

Sin embargo, al usarse el método del gradiente descendiente, los resultados obtenidos difieren bastante. La diferencia más destacable es el coste computacional del método, debido a las propiedades de las señales, la exactitud del sistema tiende a generar un sobre aprendizaje obteniendo exactitudes menores al 0.1%.

No solo desciende la exactitud del sistema, si no que también alcanza su valor máximo después de más iteraciones, mientras que con el optimizador ADAM se alcanza en aproximadamente 12 iteraciones, con este optimizador se alcanza en aproximadamente 40, lo que implica más tiempo de computo.

El comportamiento de las líneas no es similar y la variación entre los valores de ambas es notablemente mayor que para el optimizador ADAM.

De manera similar, el error del modelo tarda más tiempo en bajar lo suficiente y no muestra tampoco una convergencia para un proceso de 50 iteraciones, generando

más tiempo de computo y errores más altos tanto para el conjunto de entrenamiento como para el de validación.

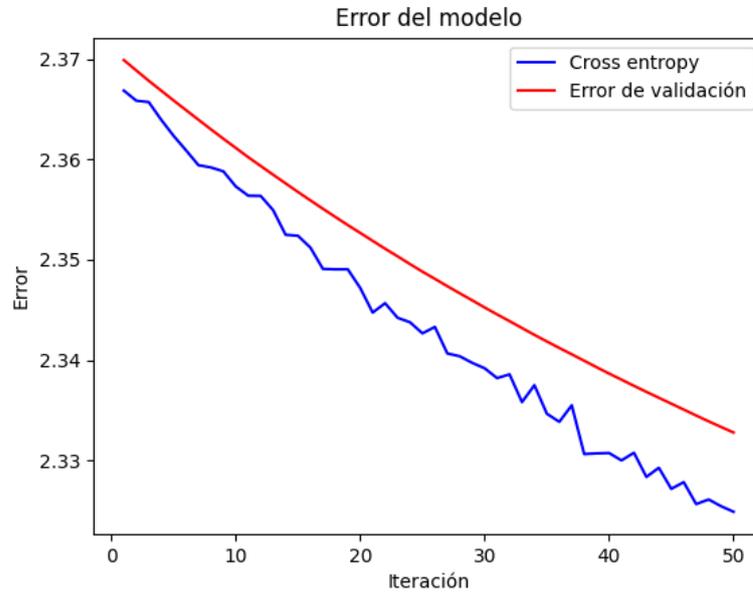


Figura 32: Error del modelo cuando se usa el método de optimización del gradiente descendiente.

### 4.3 Comparación de resultados

Fajardo [44] obtuvo una precisión entre 0.6 y 0.7 pero con 160 patrones observados y una mejor especificidad promedio, para un modelo de características combinadas. Hiraiwa [23] reconoció 20 de 30 patrones en 1000 iteraciones con una red neuronal simple. Aceves [24] alcanzó una precisión promedio entre 0.7964 y 0.9688 para cada movimiento usando cross-recurrent plots, también alcanzó una mayor sensibilidad y especificidad promedio para su modelo.

### 4.4 Métricas de calidad

La sensibilidad y especificidad promedio calculadas con la ecuación 32 son:

$$sensitividad = 95,26\% \tag{37}$$

$$\text{especificidad} = 99,37\% \quad (38)$$

Sus valores máximos son de 1 en la mayoría de los movimientos de los test realizados, sus valores mínimos son de 0.5263 y 0.9375 respectivamente para el movimiento de extensión. Entonces, en general, la probabilidad de que el modelo tenga una predicción correcta es relativamente alta.

Entonces, el modelo hizo un buen trabajo ajustando las entradas con las salidas esperadas, pero las predicciones del modelo para el movimiento inicial fueron considerablemente más pequeñas, y hay una diferencia significativa entre las probabilidades predichas en diferentes pruebas para movimientos iniciales y de extensión, así que la diferencia de varianza entre ese movimiento y los demás es notable:

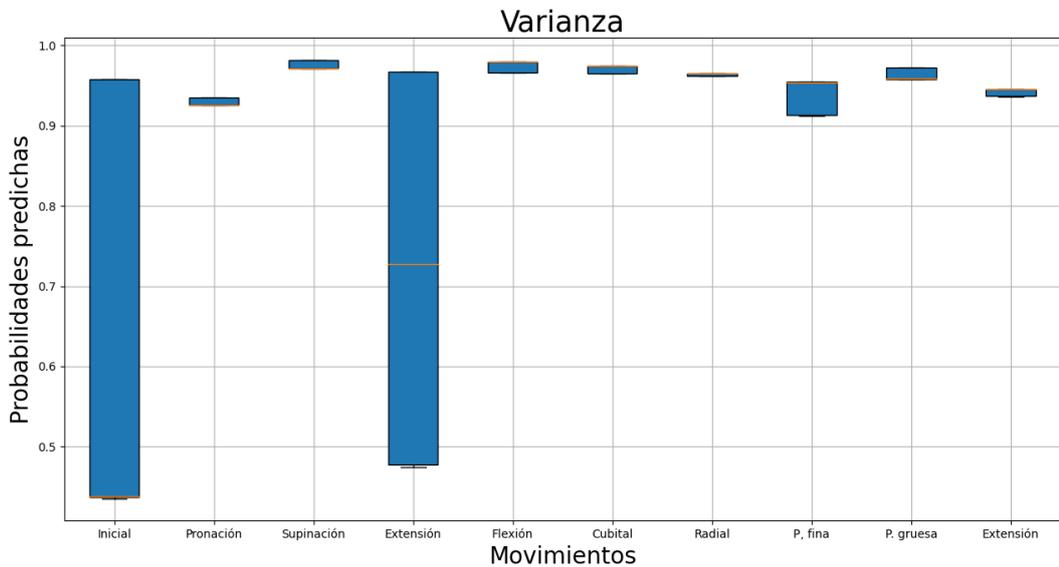


Figura 33: Varianza en la probabilidad predicha por movimiento.

La razón de esta gran diferencia en la varianza de estos dos movimientos se debe a que la red los confunde al clasificar uno en la clase del otro debido a la similitud en sus patrones en algunos canales como se discusión en la sección de Análisis del banco de señales.

## 4.5 Posibles aplicaciones

La clasificación con una exactitud elevada de los movimientos del banco de señales puede servir como punto de partida para una clasificación de movimientos más complejos de la mano mediante el mismo tipo de red neuronal y a largo plazo poder predecir con precisión la postura de la mano mediante una red neuronal que lea las señales EMG del brazo. Una de las aplicaciones que se puede llevar a cabo consiste en usar estas señales para que personas con un alto grado de discapacidad física en una de sus manos, puedan controlar una prótesis. Esto es posible siempre que el paciente disponga al menos de un músculo sano del cual poder recuperar estas señales, que será utilizado para controlar los movimientos de la prótesis.

También es posible el manejo de sistemas de software que reproduzcan palabras de un menú en la pantalla de un ordenador conectado a un sistema de adquisición de señales de este tipo. Para esto sólo hay que asociar determinadas señales EMG a los movimientos de un puntero en la pantalla del ordenador, mediante el que se escoge la palabra adecuada para ser reproducida. Otras aplicaciones son la determinación del tiempo de activación del músculo, la estimación de la fuerza producida por una contracción muscular y la obtención de un índice de la fatiga muscular

## 4.6 Uso del proyecto

La finalidad del proyecto es escribir un artículo científico para una revista sobre aplicaciones del Machine learning en la medicina, donde se detalle una propuesta de red neuronal convolucional como alternativa para los métodos de clasificación de señales electromiográficas ya existentes.

## 5 Conclusiones

Las redes neuronales convolucionales demostraron tener una gran capacidad como modelo de clasificación debido a que pueden aprender patrones de sistemas que cambian sus propiedades en función del tiempo, en específico la clasificación de estas señales fue adoptada con facilidad por el modelo debido a la capacidad que tienen las capas convolucionales de tener entradas que permitan matrices de dos o más dimensiones.

De igual manera la selección de hiperparámetros fue fundamental en el alcance de los valores obtenidos para el error y las métricas de calidad, el uso del proceso de regresión gaussiano fue una herramienta clave para encontrar la combinación ideal con el fin de llegar a valores más altos para la exactitud del modelo, el tener una distribución que dependa explícitamente de los hiperparámetros hizo posible apreciar la dependencia de la exactitud de dichas variables y encontrar su valor máximo.

A partir de la investigación con estas señales se han podido establecer los patrones neuromusculares de movimiento para sujetos de prueba de distintas edades y géneros. Esto puede ayudar a orientar la planificación de los programas de entrenamiento muscular para diferentes tipos de lesiones y patologías específicas durante el proceso de rehabilitación. También gracias al uso de su clasificación es posible el concepto de prótesis para personas afligidas por enfermedades musculares.

## Referencias

- [1] Sherwood, L., Human Physiology: From Cells to Systems (4 ed.). California: Brooks/Cole, (2001).
- [2] Kuoro, Samir & Musalev, Rodrigo. Tutorial introductorio a la teoría wavelet análisis de señales electromiográficas. Anales del Sistema Sanitario de Navarra, 32(Supl. 3), 27-43. Recuperado en 24 de enero de 2022, de [http://scielo.isciii.es/scielo.php?script=sci\\_arttextpid=S1137-66272009000600003lng=estlng=es](http://scielo.isciii.es/scielo.php?script=sci_arttextpid=S1137-66272009000600003lng=estlng=es).
- [3] J. Gao, H. Sultan, J. Hu, and W. W. Tung, “Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: a comparison,” IEEE Signal Processing Letters, vol. 17, no. 3, pp. 237–240, 2010.
- [4] Kuoro, Samir & Musalev, Rodrigo. Tutorial introductorio a la teoría wavelet, IEEE, Técnicas modernas en automática, 2022.
- [5] R. Merletti & P.P.A. Parker, Electromyography : Physiology, Engineering, and Noninvasive Applications, IEEE Press, 2004, pp. 494.
- [6] E. Scheme, K. Englehart, Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use, J. Rehabil. Res. Dev. 48 (6) (2011) 643–660, <http://dx.doi.org/10.1682/JRRD.2010.09.0177>.
- [7] Hyeon-min, S., Hongsub, An, Sanghyuk, L., Eung, L., Hong-ki, M., & Sangmin L., EMG Pattern Classification by Split and Merge Deep Belief Network, Symmetry, Volumen 8, pp. 1-6, (2016).
- [8] Yusuke Yamanoi, Yosuke Ogiri Ryu Kato, EMG-based posture classification using a convolutional neural network for a myoelectric hand, Biomedical Signal Processing and Control, Volume 55, 2020, 101574, pp. 1746-8094, <https://doi.org/10.1016/j.bspc.2019.101574>.

- [9] Costanzo, L. S., Gómez, J. P. (2000). Fisiología celular. McGraw-Hill Interamericana.
- [10] Tortora GJ, Derrickson BH. Principles of anatomy and physiology: Wiley Global Education; 2013.
- [11] Konrad P. The abc of EMG. A practical introduction to kinesiological electromyography. 2005;USA 1:30-5.
- [12] Young, A. J., Smith, L. H., Rouse, E. J., & Hargrove, L. J., Classification of simultaneous movements using surface EMG pattern recognition. IEEE Transactions on Biomedical Engineering, volumen 60, pp. 1250–1253, (2013).
- [13] Guzmán-Muñoz, Eduardo & Méndez-Rebolledo, Guillermo, Electromiografía en las Ciencias de la Rehabilitación, Salud Uninorte. Barranquilla (Col.) 2018; 34 (3): 753-765.
- [14] Liu, Danqing (30 November 2017). .<sup>A</sup> Practical Guide to ReLU”. Medium. (2021).
- [15] Al-Angari, H., Kanitz, G., Tarantino, S. & Cipriani, C., Distance and mutual information methods for EMG feature and channel subset selection for classification of hand movements, Biomed. Signal Process. Control, volumen 27, pp. 24-31, (2016).
- [16] Aparicio MV. Electromiografía cinesiología. Rehabilitación. 2005;39(6):255-64. doi: [http://dx.doi.org/10.1016/S0048-7120\(05\)74359-0](http://dx.doi.org/10.1016/S0048-7120(05)74359-0)
- [17] Basmajian J. Electrodes and electrode connectors. New Concepts of the Motor Unit, Neuromuscular Disorders, Electromyographic Kinesiology. 1. Karger Publishers; 1973:502-10.
- [18] Polikar, R. The Wavelet Tutorial. Durham Computation Center, Iowa State University, 1995.

- [19] Mamun, Md., Al-Kadi, Mahmoud, Marufuzzaman, Mohd.. (2013). Effectiveness of Wavelet Denoising on Electroencephalogram Signals. *Journal of applied research and technology*, 11(1), 156-160. Recuperado en 28 de febrero de 2022, de [http://www.scielo.org.mx/scielo.php?script=sci\\_arttextpid=S1665-64232013000100014lng=estlng=en](http://www.scielo.org.mx/scielo.php?script=sci_arttextpid=S1665-64232013000100014lng=estlng=en).
- [20] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE pattern Anal. and achine Intell.*, vol. 11, No. 7, pp. 674-693, 1989.
- [21] Alva, Carlos. PROCESAMIENTO DE SEÑALES DE ELECTROMIOGRAFÍA SUPERFICIAL PARA LA DETECCIÓN DE MOVIMIENTO DE DOS DEDOS DE LA MANO. Tesis para optar el título profesional de ingeniero electrónico. Universidad Ricardo Palma, Lima, Perú, 2012.
- [22] Ramírez I., Razo, N., Aceves, M. & Gorrostieta E., Metodología para la Adquisición de Señales Electromiográficas en el Brazo Utilizando un Lector de Señales Multicanal, *La Mecatrónica en México*, Volumen 8, pp. 24-34, (2019).
- [23] A. Hiraiwa, K. Shimohara & Y. Tokunaga, .<sup>EMG</sup> pattern analysis and classification by neural network, *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics*, pp. 1113-1115 vol.3, doi: 10.1109/ICSMC.1989.71472, (1989).
- [24] Aceves, M., Ramos, J., Gorrostieta, E. & Pedraza, J., Methodology Proposal of EMG Hand Movement Classification Based on Cross Recurrence Plots, *Computational and Mathematical Methods in Medicine*, Volume 2019, pp. 5-7, (2019).
- [25] Charniak, E., *Introduction to Deep Learning*, pp. 78-82, The MIT Press, Massachusetts (2018).
- [26] Vasilev, I., Slater, D., Spacagna, G., Roelants, P. & Zocca, V. *Python Deep Learning*, pp. 85-121, Packt, Birmingham (2019).

- [27] Aggarwal, C., Neural Networks and Deep Learning, pp. 134-140, Springer, New York (2018).
- [28] Anandh, S. & Clinton, D., HANDWRITTEN CHARACTER RECOGNITION USING CNN, IJRAR, Volumen 5, pp. 242-244 (2018).
- [29] Galushkin, A., Neural Networks Theory, [p43-p52], Springer, New York (2007).
- [30] Wang, S., Zhou, T. & Bilmes, J.. (2019). Bias Also Matters: Bias Attribution for Deep Neural Network Explanation. *Proceedings of the 36th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 97:6659-6667 Available from <https://proceedings.mlr.press/v97/wang19p.html>.
- [31] Charu, C, (2018). Neural Networks and Deep Learning, Nueva York, Springer.
- [32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.
- [33] Ramachandran, P., Zoph, B., Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.
- [34] Abad, D, (2008). Monitor EMG con conexión inalámbrica, (Trabajo de grado). Universidad Autónoma de Barcelona, Barcelona.
- [35] Sutton, R. S. Barto A. G. (1998). Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA. Softmax Action Selection.
- [36] Brownlee, Jason (8 January 2019). "A Gentle Introduction to the Rectified Linear Unit (ReLU)". *Machine Learning Mastery*. Retrieved 8 April 2021.
- [37] P. Kingma & Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, ICLR, 1412.6980, pp 1-15, 2017.
- [38] Rasmussen, C. E., & Williams, C. K. I., Gaussian processes for machine learning, The MIT Press. (2016).

- [39] Zhenwen Dai. Computationally efficient GPs. In Gaussian Process Summer School (GPSS), 2019.
- [40] Zoubin Ghahramani. A Tutorial on Gaussian Processes (or why I don't use SVMs). In Machine Learning Summer School (MLSS), 2011.
- [41] Wang, J. (2020). An intuitive tutorial to Gaussian processes regression. arXiv preprint arXiv:2009.10862.
- [42] Van Rijsbergen, C. Information Retrieval (2nd ed.). Butterworth-Heinemann, University of Glasgow. (1979).
- [43] Powers, David M. W. "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation". Journal of Machine Learning Technologies. 2 (1): 37–63. (2011).
- [44] Fajardo, J., Gomez, O. Prieto, F. EMG hand gesture classification using hand-crafted and deep features, Biomedical Signal Processing and Control, Volume 63, 102210, pp. 1746-8094, <https://doi.org/10.1016/j.bspc.2020.102210>, (2021).

## 6 Anexos

El primer código se usó para encontrar los valores máximo y mínimo del banco de señales.

```
1 mat=sio.loadmat('BancoSeales.mat')
2 print(sio.whosmat('BancoSeales.mat'))
3 lsen=mat['lSenR']
4 X=[]
5 Y=[]
6 vmax=0
7 vmin=0
8 for movimiento in range(0,20,2):
9     #Definimos que solo usaremos los datos de la mano izquierda de cada
10    sujeto
11    for sujeto in range(0,48):
12        #Vector de entrada
13        x=lsen[0][sujeto][0][movimiento]
14        for i in range(0,8):
15            for j in range(0,3200):
16                if vmax<x[i][j]:
17                    vmax=x[i][j]
18                if vmin>x[i][j]:
19                    vmin=x[i][j]
```

Mediante el siguiente programa se generaron las gráficas y de las señales electro-miográficas del banco de señales, antes y después de normalizar.

```
1 Y =[]
2 vmax=213.6912883660619 #max
3 vmin=-214.2656460876913 #min
4 for movimiento in range(0,20,2):
5     #Defnimos que solo usaremos los datos de la mano izquierda de cada
6     sujeto
7     for sujeto in range(0,48):
8         #Vector de entrada
9         x=lsen [0][sujeto][0][movimiento]
10        '''for i in range(0,8):
11            for j in range(0,3200):
12                x[i][j]=(x[i][j]-vmin)/(vmax-vmin)'''
13        X.append(np.array(x))
14        #Vector esperado
15        y=int(movimiento/2)
16        Y.append(y)
17 Y=np.array(Y)
18 plt.minorticks_on()
19 plt.grid(b=True, which='major', color='#666666', linestyle='-')
20 plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha
21         =0.2)
22 plt.show()
```

El siguiente segmento de código contiene una parte del programa donde se genera el modelo de red neuronal, se entrena y se generan predicciones para generar información y gráficas de las métricas de calidad de la red según el diagrama de flujo de la figura 25.

```
1 model = Sequential()
2 model.add(Conv1D(filters=128, kernel_size=3, activation='relu',
   input_shape=(48,640)))
3 '''model.add(Dropout(0.1))'''
4 model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
5 model.add(Dropout(0.1)) #[0.1,0.5]
6 model.add(MaxPooling1D(pool_size=3))
7 model.add(Flatten())
8 model.add(Dense(100, activation='sigmoid'))
9 model.add(Dense(10, activation='softmax'))
10 opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
11 model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=[
   'accuracy'])
12 # fit network
13 modelo1=model.fit(X_train, Y_train, epochs=25, batch_size=32,
   validation_split=0.38)
```

El siguiente programa fue usado para encontrar los hiperparámetros óptimos usados en el programa donde se modela la red neuronal convlucional mediante un proceso de regresión gaussiano.

```
1 param_grid = [{
2     "lr": [1e-8, 1e-2, 1e-9],
3     "filters": [2, 1024, 2]
4 }, {
5     "batch": [2, 1024, 2],
6     "kernel": [DotProduct(sigma_0) for sigma_0 in np.logspace(-1, 1, 2)
7 ]
8 }
9 # scores for regression
10 scores = ['explained_variance', 'r2']
11
12 gp = GaussianProcessRegressor()
13 for loss in losses:
14     print("# Tuning hyper-parameters for %s" % score)
15     print()
16
17     clf = GridSearchCV(estimator=gp, param_grid=param_grid, cv=4,
18                        scoring='%s' % score)
19     clf.fit(X_tr.reshape(-1, 1), y_tr)
```