CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE BIOELECTRÓNICA

Modelo virtual de prótesis de mano controlada mediante señales EEG

Tesis que presenta

Lorena Fernanda Morales Moreno

para obtener el Grado de

Maestra en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Directores de la Tesis: Dr. Juan Manuel Gutiérrez Salgado

Dra. Rebeca del Carmen Romo Vázquez

Ciudad de México Junio, 2022

Agradecimientos

A mis padres Emma y Héctor, por darme siempre su amor incondicional y enseñarme los valores y principios con los que me he formado hasta el día de hoy, por siempre preocuparse por mi educación y por recibirme amorosamente en su hogar cada que regreso a casa. A mis hermanos David y Paula por su apoyo, sus palabras de aliento y por el amor que me han dado durante toda la vida.

Alberto, gracias por acompañarme desde el inicio de esta aventura, por ayudarme a continuar todas las veces que pensé que ya no era posible y por el amor que me demuestras cada día.

A mis asesores, Dr. Juan Manuel Gutiérrez Salgado y Dra. Rebeca del Carmen Romo Vázquez, por su guía y paciencia durante todo este proceso. Gracias a ambos por aceptarme como su estudiante y preocuparse siempre por mi aprendizaje, resolviendo mis dudas y dedicando de su valioso tiempo a este proyecto de investigación. Así como a mis sinodales Dr. Lorenzo Leija, Dr. Arturo Vera y Dr. Carlos Alvarado, por tomarse el tiempo de leer mi trabajo y apoyarme con sus invaluables contribuciones.

Al CINVESTAV, en especial a la sección de Bioelectrónica y los investigadores que la conforman, por permitirme formar parte de esta sección y compartir conmigo sus conocimientos, mismos que me llevan a ser una mejor profesionista. De igual forma agradezco a los integrantes del laboratorio 11, con los que siempre pude contar durante mi estancia y a mis compañeros de generación con quienes a pesar de la distancia logré formar una gran amistad.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el financiamiento brindado con el cual pude desarrollar mis estudios de maestría.

Contenido

Índice de figuras

Índice de tablas

Resumen

En este trabajo se presenta el diseño y programación de un modelo virtual de ambas manos, de aspecto realista, que puede controlarse mediante señales electroencefalográficas registradas durante la ejecución de tareas de imaginación motora de ambas extremidades. Como fuente de información se utilizó la base de datos pública “BCI competition IV. Dataset IIa”, de la que se extrajeron los movimientos de las manos para cada uno de sus nueve participantes.

Para corroborar la existencia de la intencionalidad motora relacionada con los movimientos de las manos en los registros EEG disponibles, se calcularon los potenciales ERD utilizando el método de Transformada de Hilbert. Una vez corroborada la presencia de potenciales ERD, se optó por utilizar características temporales relacionadas con la tendencia, dispersión, deformación y la energía en las ondas cerebrales δ, θ, α y β. El clasificador encargado de identificar los movimientos imaginarios ejecutados por los participantes se construyó empleando redes neuronales artificiales y a partir de los resultados obtenidos de la clasificación, se consiguieron los comandos de control necesarios para manipular el modelo de manos virtualizado.

Los modelos neuronales entrenados lograron clasificar correctamente entre un 87.66% y 96.72% de los movimientos imaginarios realizados por los participantes con datos de prueba. Para el control del modelo virtual, se empleó el protocolo TCP/IP entre Matlab y la plataforma Unity de simulación, consiguiendo clasificar los ensayos y transferir la información en un tiempo de 0.015 s, sin pérdida de datos ni confusiones en la identificación de los comandos de control.

Abstract

This work presents the design and programming of both hands virtual model with realistic aspect, this can be controlled using electroencephalographic signals recorded during both extremities motor imagery tasks. As an information source, it was used the public dataset “BCI competition IV. Dataset IIa”, extracting only the movements of interest for each one of its nine subjects.

To bear out the existence of motor intention related to hands movements in the available EEG records, it was calculated the ERD potentials using the Hilbert Transform method. Once corroborated the existence of ERD potentials, it opted for temporal characteristics related to tending, dispersion, deformation, and energy in δ, θ, α and β brain waves. The classifier in charge to identify imaginary movements executed for participants was developed using artificial neural networks to obtain the necessary control commands to manipulate the virtualized hands model.

Trained neural models were capable of correctly classifying between 87.66% and 96.72% of the imagined movements executed from the subjects using test data. To virtual model control, TCP/IP protocol was used between Matlab and the simulation platform Unity, getting an information transfer in a time of 0.015 s, without data loss or confusion in control commands identification.

Capítulo 1. Introducción

Los sistemas de interfaces cerebro computadora (BCI, por sus siglas en inglés) han tomado un importarte auge en la actualidad en distintas ramas de la ciencia. El objetivo de estas interfaces es manipular sistemas de hardware o software mediante el uso de señales cerebrales (registradas comúnmente con electroencefalografía) que son procesadas y posteriormente transformadas en comandos de control [1].

Los ritmos sensoriomotores (SMR por sus siglas en inglés) son fenómenos cerebrales producidos en el área motora primaria durante la intención de movimiento y son transformadas en comandos de control por las “BCI basadas en imágenes motoras (MI-BCI por sus siglas en inglés)”. Entre las ventajas que ofrece este tipo de sistema se encuentra que para la generación de los SMR no se requiere de estímulos externos, ni mantener el control de la mirada en un punto en específico para generar los potenciales de interés, pues al funcionar de la misma manera en la que normalmente se controlan los músculos se convierte en un proceso que puede considerarse natural y ergonómico [2].

Como las MI-BCI no requieren de estímulos externos se basan en la autorregulación de la activación cerebral, proceso en el que a través de entrenamiento los usuarios aprenden a modificar de manera voluntaria sus ritmos y potenciales cerebrales, este proceso es conocido como “entrenamiento a través de neuro retroalimentación” [3]. Este tipo de entrenamiento se fundamenta en la neuroplasticidad (fenómeno que permite al cerebro adaptarse a diversas circunstancias), y contribuye a la activación de las vías motoras cerebrales, por lo que al ser utilizado en la rehabilitación puede reemplazar y restaurar funciones neurológicas que se han perdido a causa de algún desorden o accidente [4].

Los sistemas de tipo MI-BCI utilizan diversas técnicas y estrategias que facilitan la autorregulación cerebral, derivando en una mejor clasificación de movimientos y

favoreciendo al proceso de neurorrehabilitación. Ejemplo de estas técnicas incluye sugerir a los usuarios que imaginen cinestésicamente el movimiento solicitado, en lugar de solo visualizarlo [3, 5], agregar estímulos electrotáctiles durante la retroalimentación [6] o utilizar modelos virtuales realistas para inducir la actividad de las neuronas espejo causando la activación de las regiones correspondientes al movimiento a través de la imitación [7, 8]. Estas últimas se basan en técnicas conocidas como “sensación de propiedad” y “observación de acción” y proveen al usuario la sensación de estar controlando una parte de su propio cuerpo, lo que estimula áreas corticales relacionadas con la actividad motora [9].

Los trabajos reportados en la literatura que emplean modelos virtuales realistas hacen un extenso uso de los métodos de aprendizaje automático basados en modelos como las máquinas de soporte vectorial (SVM, por sus siglas en inglés) con kernel lineal y el análisis discriminante lineal (LDA, por sus siglas en inglés) para clasificar movimientos imaginarios [10, 11]. Por otra parte, los métodos de clasificación basados en modelos no lineales, apoyados del uso de modelos virtuales realistas, han demostrado mejoras en el rendimiento en la clasificación de movimientos imaginarios, sin embargo, en comparación con los sistemas que se basan en modelos lineales, la literatura no reporta un número amplio de sistemas MI-BCI que conjunten estas tecnologías [11–13].

1.1 Planteamiento del problema

En la etapa de procesamiento de las MI-BCI es importante considerar que la actividad cerebral registrada está asociada a imágenes motoras y puede presentar variaciones debido a los factores anatómicos o fisiológicos particulares de cada sujeto registrado [14, 15], así como a la vulnerabilidad al ruido y artefactos que se presentan en el EEG. Lo anterior, sugiere que se deben usar modelos computacionales robustos, distintos a los que mayormente se presentan en la literatura (métodos lineales), que permitan extraer patrones específicos relacionados con la intención de movimiento y logren su clasificación con un alto grado de certeza. En este contexto, una herramienta

que puede favorecer a la correcta extracción de patrones y con un menor tiempo de entrenamiento es la retroalimentación visual sustentada en las técnicas de sensación de propiedad y observación de acción.

Con base en lo anterior, en este trabajo se propone un método de clasificación no lineal basado en redes neuronales artificiales que, apoyado con características temporales extraídas de los registros cerebrales manipule un modelo virtual de ambas manos con un desempeño competitivo con respecto al reportado en el estado del arte. Esta propuesta representa un nicho de oportunidad en el que se favorece el proceso de neurorrehabilitación, reduciendo el tiempo de entrenamiento.

1.2 Objetivos

1.2.1 Objetivo general

Diseñar un modelo virtual de ambas manos que, a partir de la interpretación de los resultados de clasificación entre imágenes motoras de mano derecha e izquierda, sea capaz de emular la apertura y cierre de manera individual. Los movimientos por clasificar serán obtenidos de una base de datos pública, cuyos registros deben realizarse mediante la técnica de electroencefalografía durante la imaginación motora de la mano derecha e izquierda. La manipulación del modelo se realizará utilizando comandos de control que serán trasmitidos desde el software de procesamiento hacia la plataforma de modelado, integrando una plataforma unificada para el usuario final.

1.2.2 Objetivos específicos

- Identificar y seleccionar una base de datos pública que contenga registros electroencefalográficos que integren información de la intención de movimientos de mano derecha o izquierda.

- Evaluar la presencia de los potenciales relacionados a movimientos (ERD) en los registros de EEG que integran la base de datos empleando el método de Transformada de Hilbert.

- Establecer el tipo de herramienta de procesamiento y análisis de EEG multicanal que permitirá obtener conjuntos de características útiles en el domino del tiempo para crear modelos clasificadores robustos que identifiquen las imágenes motoras presentes en las señales EEG extraídas de la base de datos elegida.

- Validar la robustez y desempeño de los modelos clasificadores programados mediante la contaminación de los registros de EEG con 10%, 25%, 40% y 50% de ruido blanco proporcional a la energía contenida en ellos.

- Proporcionar los comandos correctos que permitan el control del modelo virtual de ambas manos para su operación en la ejecución de los movimientos a partir del modelo clasificador.

- Diseñar e implementar un modelo realista de ambas manos virtualizado en 3D empleando una plataforma de código abierto, que emule los movimientos naturales de las articulaciones mediante animaciones, para simular la apertura y cierre de las extremidades de manera individual, con el fin de tener retroalimentación visual de la capacidad de clasificación de las características extraídas de la EEG asociadas con la intención del movimiento.

- Integrar en una plataforma unificada el modelo virtualizado, las herramientas de procesamiento y los modelos clasificadores con la intención de evaluar la identificación de movimientos imaginarios presentes en los EEG y su correspondiente interpretación como entradas para manipular el modelo virtual.

1.3 Estructura de la tesis

Capítulo 1: En esta sección se proporciona una visión general del alcance de esta tesis. Justifica el problema a resolver junto con el objetivo general y específicos que se persiguieron.

Capítulo 2: En este capítulo se presenta de manera general las estructuras cerebrales involucradas en la generación de movimiento, así como la electroencefalografía, ritmos cerebrales, sincronización y desincronización relacionada con eventos y los sistemas BCI, así como las etapas que los conforman y medidas que evalúan el desempeño. Finalmente se describe el estado del arte relacionado con la aplicación de MI-BCI's para la manipulación de modelos virtuales.

Capítulo 3: En este apartado se describe la metodología implementada para dar solución al planteamiento del problema. En ella se describe la base de datos y el proceso bajo el que se obtienen los potenciales ERD, se llevan a cabo las etapas de procesamiento de una BCI, así como el diseño del modelo virtual en Blender y su integración en la plataforma de desarrollo Unity a través de la cual se transmiten los comandos de control, enviados desde Matlab utilizando el protocolo TCP/IP.

Capítulo 4: En este capítulo se presentan y discuten los resultados obtenidos por el desarrollo de la metodología. Entre ellos, se muestran las métricas obtenidas por los modelos clasificadores, así como la respuesta durante las etapas de validación. En esta sección también se muestra el resultado de desempeño del modelo virtual.

Capítulo 5: Finalmente en esta sección Se hace un análisis del desempeño del sistema implementado, mostrando conclusiones basadas en los resultados obtenidos y se plantean trabajos que podrían derivar de este trabajo de tesis.

Capítulo 2. Antecedentes

2.1 Áreas cerebrales relacionadas con el movimiento

El cerebro es el órgano más complejo del cuerpo humano. Se compone principalmente de 4 lóbulos nombrados como frontal, parietal, temporal y occipital, de acuerdo con su posición anatómica (ver figura 2.1).

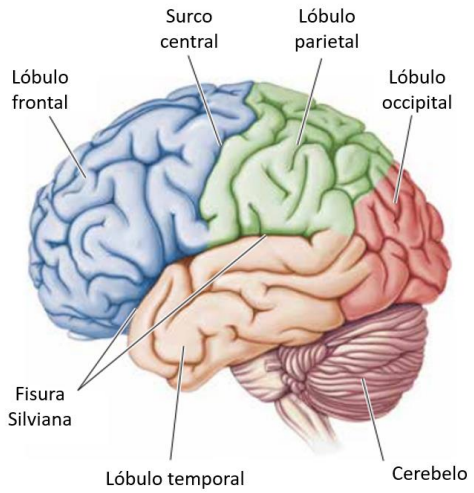

Fig. 2.1 Se muestran los lóbulos cerebrales. Los lóbulos frontal y parietal se encuentran divididos por el surco central, mientras que la fisura Silviana separa los lóbulos frontal y temporal [16].

De igual forma, el cerebro se encuentra dividido en áreas que se encargan de diversas tareas o funciones. Las áreas cerebrales relacionadas con tareas motoras son la corteza motora primaria (M1), el área motora suplementaria (SMA, por sus siglas en inglés) y el área premotora (PMA, por sus siglas en inglés), localizadas en el lóbulo frontal. La corteza motora primaria se encuentra en el giro precentral y es directamente responsable de la coordinación de movimientos voluntarios, esta correlaciona somatotópicamente áreas del M1 con los cambios electrofisiológicos generados durante el control de algunas partes del cuerpo [17], como se muestra en la figura 2.2.

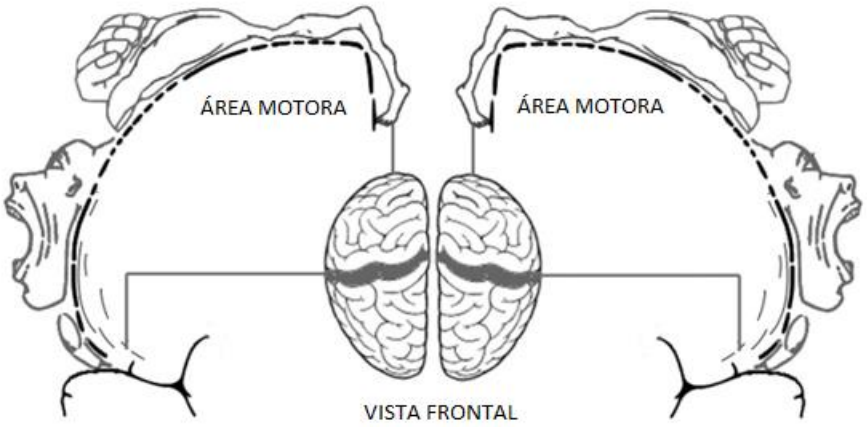

Fig. 2.2 Mapa somatotópico del giro precentral humano [18].

Los movimientos generados por M1 en ambos hemisferios son soportados por la PMA haciendo posible la ejecución de imágenes motoras, que pueden definirse como un ensayo imaginado de un acto motor sin resultar en un movimiento muscular [19], lo cual impulsa la activación del SMA por encima de un umbral iniciando la actividad motora [18].

El fenómeno fisiológico que permite que exista movimiento consiste en una proyección de axones desde los ganglios basales hacia la entrada del núcleo ventral lateral (VL), conocido como VLo, y hasta el SMA. Los ganglios basales, localizados alrededor del tálamo en ambos hemisferios, se conforman por el núcleo caudado, el putamen, globo pálido (que consta de un segmento interno, Gpi, y un segmento externo, Gpe), núcleo subtalámico y la sustancia negra (ver figura 2.3a) [16].

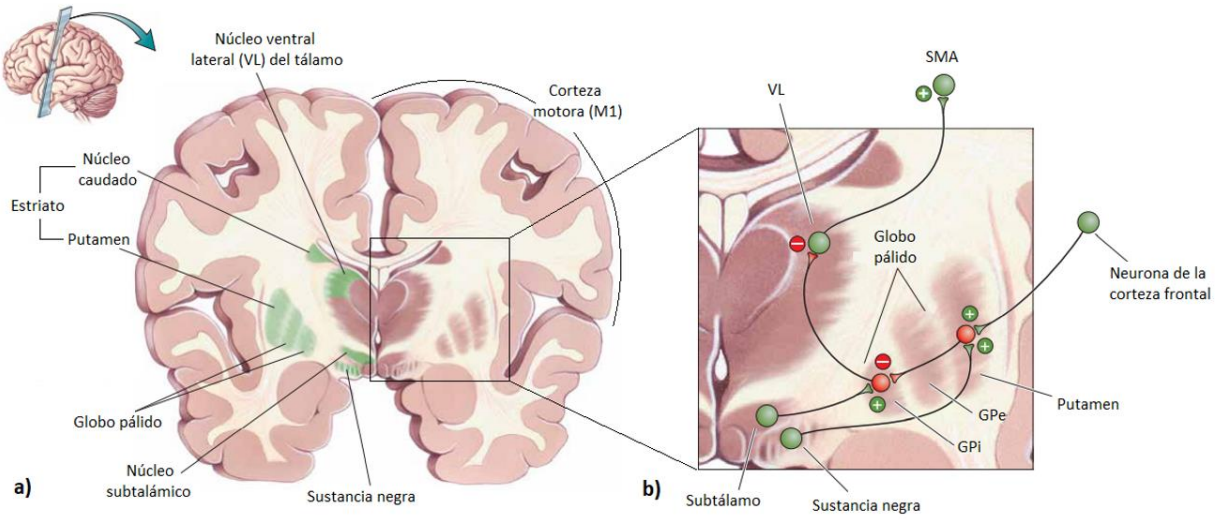

Fig. 2.3 a) Estructuras asociadas a los ganglios basales, estas se encuentran alrededor del tálamo, b) “Camino directo del movimiento”, las sinapsis marcadas con (+) son excitadoras y aquellas marcadas con (-) son inhibidoras [16].

Durante la implementación de tareas relacionadas con el movimiento, la actividad cognitiva comienza en la corteza frontal y se propaga hacia el putamen a través de conexiones excitadoras. La activación del putamen establece una sinapsis inhibitoria en las células del globo pálido, esta inactivación permite que el Vlo se active y logre retransmitir las señales hacia áreas específicas del SMA (ver figura 2.3b). Como resultado, las masas neuronales del SMA trabajan con diferentes tareas cognitivas complejas ocasionando que las neuronas se disparen rápidamente, pero no de manera sincronizada. Este proceso se conoce como “camino directo del movimiento” [16,18].

Por el contrario, cuando no se desempeña una tarea motora no existe una transmisión de información desde la corteza frontal, por lo que el putamen permanece inactivo y el globo pálido no logra inactivar el Vlo, permitiendo que el SMA reciba señales de células neuronales del tálamo conocidas como neuronas marcapasos, resultando en un comportamiento neuronal sincronizado [16].

El gran número de neuronas piramidales trabajando en sincronía en el SMA, PMA y M1 inducen una actividad eléctrica cerebral de gran amplitud, la cual puede ser medida de manera no invasiva a través de electrodos distribuidos sobre el cuero cabelludo [18]. A esta técnica se le conoce como electroencefalografía (EEG).

2.2 Electroencefalografía

El EEG registra la actividad eléctrica cerebral de grandes poblaciones neuronales de la corteza cerebral, medidas desde la superficie del cuero cabelludo a través de electrodos [20]. Esta es una técnica no invasiva que, en comparación con diversos métodos de adquisición, puede ser de bajo costo y como puede observarse en la figura 2.4, permite obtener información de la actividad eléctrica cerebral en el orden de milisegundos. Estas características hacen del EEG una técnica ampliamente utilizada para estudiar funciones cerebrales, monitoreo continuo por largos periodos de tiempo o el diagnóstico de desórdenes neurológicos como la epilepsia, desórdenes del sueño, demencia, entre otras [21].

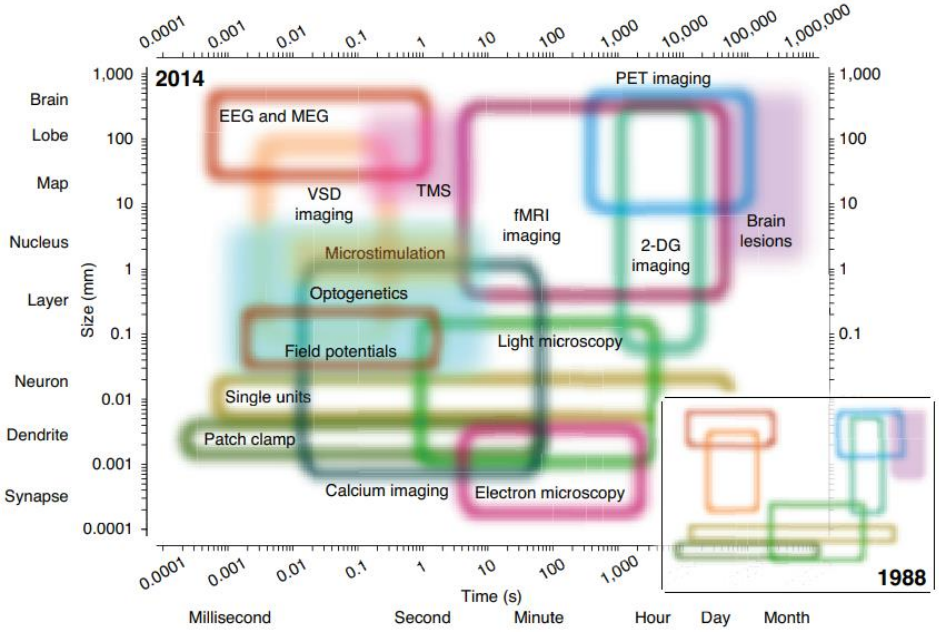

Fig. 2.4 Comparación entre los métodos disponibles para el estudio del sistema nervioso entre los años 1988 y 2014. El EEG se observa en la esquina superior izquierda, indicando que la información se registra en el orden de milisegundos [22].

2.2.1 Adquisición de las señales de EEG

La ubicación de los electrodos se hace conforme al “Sistema internacional 10 - 20” [23], el cuál distribuye los electrodos de forma que se conecta el nasión (punto de intersección entre los huesos frontal y nasal) con el inión (protuberancia pequeña situada en la parte posterior del cráneo) y los puntos preauriculares derecho e izquierdo, pasando a través del vértex. De esta manera se asegura que se obtendrá el registro de la actividad eléctrica en todos los lóbulos cerebrales (ver figura 2.1). Los electrodos se colocan a partir de la medición de la distancia existente entre los puntos de referencia, dejando un 10% de la distancia entre el punto de referencia y el electrodo más cercano a él y 20% de distancia entre el resto de los electrodos. Estos son nombrados con una letra mayúscula que indica la inicial del lóbulo donde se posicionan (P = parietal, T = temporal, F = frontal y O = occipital), seguido de la letra se coloca un número par, impar, o una z dependiendo si se encuentra en el hemisferio derecho, izquierdo o en la línea que une al nasión y al inión (ver figura 2.5). Los electrodos posicionados en el polo frontal son nombrados como Fp y los electrodos auriculares llevan la letra A [18].

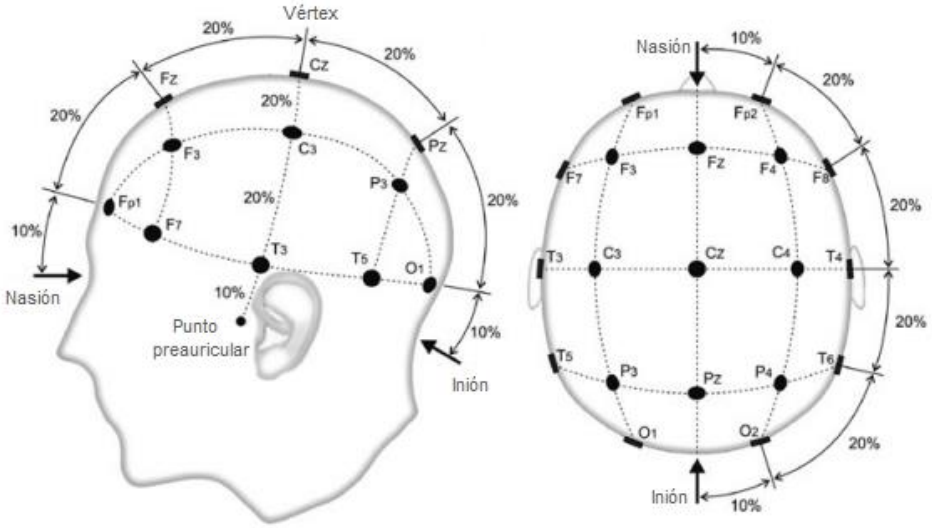

Fig. 2.5 Sistema internacional 10 - 20. Se muestra la distribución de los electrodos en su vista lateral (izquierda) y superior (derecha) [1].

2.2.2 Ritmos cerebrales

Las oscilaciones o ritmos cerebrales se encuentran categorizados en bandas de frecuencia, estas se relacionan con distintos estados cerebrales o funciones y se pueden observar en locaciones y circunstancias específicas. los ritmos que han sido más estudiados son delta (δ), theta (θ), alpha (α), beta (β) y gamma (γ) [18], mismos que se describen a continuación y pueden observarse en la figura 2.6.

Ritmo delta (δ): Se encuentra en bandas de frecuencia que van desde los 0.5 hasta los 4 Hz, y se detecta en el lóbulo frontal del cerebro, presenta una amplitud de hasta 100 μV. Se asocia principalmente al estado del sueño profundo, sin embargo, diversos estudios también lo han relacionado a la plasticidad cortical en sujetos despiertos, y a procesos cognitivos en estudios relacionados con eventos [24].

Ritmo theta (θ): Este ritmo cerebral puede alcanzar una amplitud de hasta 100 μV y tiene una frecuencia de entre los 4 y 8 Hz. Los cambios en la actividad de este ritmo se relacionan con la presencia de dislexia en niños [25], y la depresión en adultos [26].

Ritmo alpha (α): Localizado en el lóbulo occipital, se detecta principalmente en adultos normales durante el estado de reposo. Presenta una amplitud de entre 20 y 60 μV y su banda de frecuencias se encuentra entre los 8 y 13 Hz [24].

Ritmo beta (β): Es registrado en los lóbulos frontal y parietal del cerebro, este ritmo cerebral se presenta durante la activación, solución de problemas, concentración profunda y realización de tareas motoras. Su rango de frecuencias se encuentra entre los 13 y 30 Hz y tiene una amplitud de entre 20 y 30 μV [27].

Ritmo gamma (γ): Este ritmo involucra oscilaciones rápidas que se presentan durante la percepción consiente. Su banda de frecuencias se encuentra entre los 30 y

100 Hz y tienen una amplitud de menos de 50 μV. Suele asociarse a procesos de memoria o desordenes psiquiátricos [24].

Similar al ritmo α, Jasper y Andrews describieron el ritmo cerebral mu (μ), el cual tiene una banda de frecuencias entre los 8 y 13 Hz, pero a diferencia de α, su amplitud es mayor a los 50 μV y se localiza en el área precentral, en la corteza motora. Este ritmo cerebral se encuentra ampliamente relacionado con la imaginación motora [28].

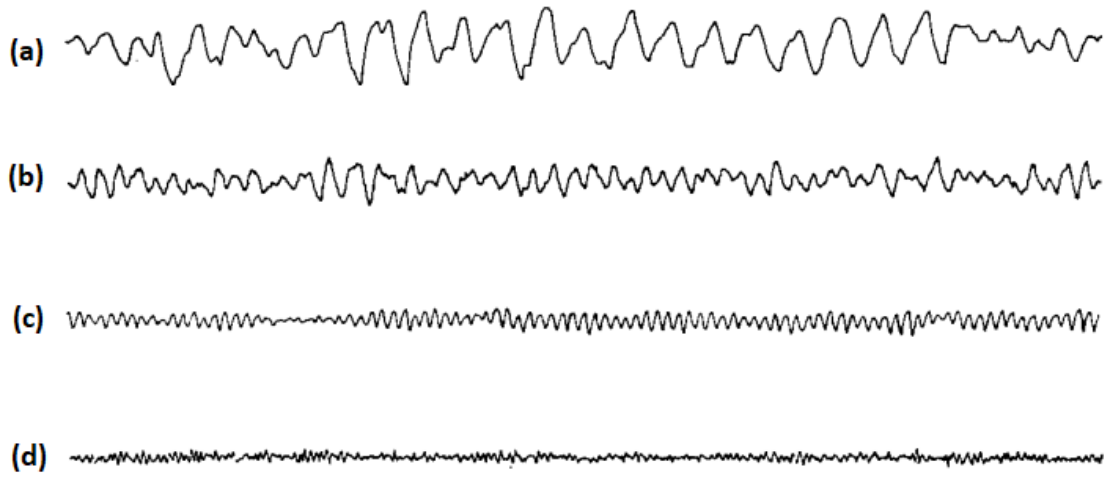

Fig. 2.6 Ejemplos de los ritmos (a) delta; (b) theta; (c) Alpha y (d) beta [29].

2.3 ERD /ERS

Como se describió en la subsección 2.1, las neuronas corticales se mantienen sincronizadas en periodos de inactividad. Sin embargo, al presentarse un evento particular o de referencia definible, las neuronas corticales reciben señales específicas ocasionando que pierdan su sincronía. Lo anterior ocasiona un cambio en el potencial de EEG conocido como potencial relacionado con eventos (ERP por sus siglas en inglés) [1, 18].

Los ERP que ocurren en ritmos cerebrales específicos se conocen como eventos de fase bloqueada, el decremento e incremento de energía que lo acompaña son la desincronización relacionada a eventos (ERD por sus siglas en inglés) y sincronización relacionada a eventos (ERS por sus siglas en inglés), respectivamente [18, 30, 31].

La ejecución de un movimiento y la imaginación motora, generan un cambio en los ritmos sensoriomotores (SMR por sus siglas en inglés), que son señales registradas desde el área motora y sensoriomotora del cerebro. Durante la preparación y ejecución de un movimiento se desencadena un ERD en los ritmos μ y β [32, 33], una vez concluido, los SMR vuelven a sincronizarse. El ERS, ocasionado por la finalización del movimiento, logra caracterizarse mejor en la banda de frecuencia β (ver figura 2.7) [34].

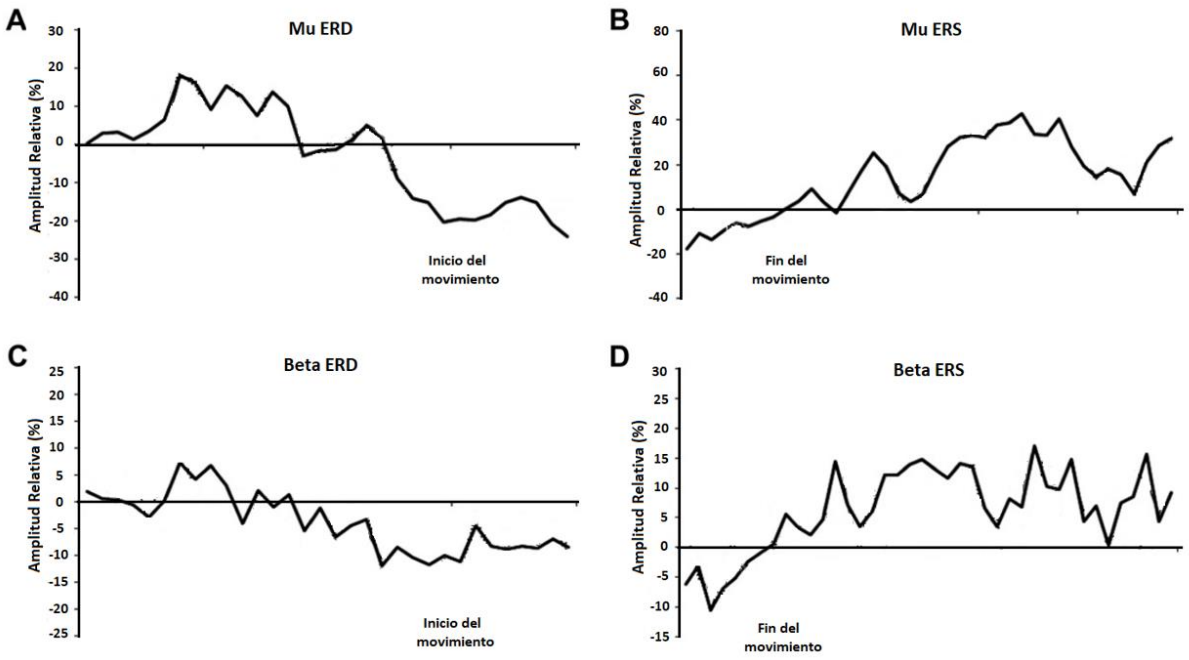

Fig. 2.7 Patrones ERD / ERS en la región central (C3 y C4) durante la imaginación motora en las bandas μ y β [35].

Los ERD / ERS se presentan de manera contralateral, y debido a la naturaleza de los SMR pueden medirse utilizando la técnica de EEG a través de los electrodos

colocados sobre la línea central del cerebro, o la corteza motora. De esta forma, durante la imaginación motora de la mano derecha se generará una desincronización neuronal en la corteza motora en el hemisferio izquierdo y se reflejará en el electrodo C3. Por otro lado, durante la imaginación motora de la mano izquierda será el hemisferio derecho quien sufra una desincronización neuronal, por lo que se registrará en el electrodo C4 [1, 18].

Con base en la característica de contralateralidad es posible determinar, por ejemplo, si se realiza un movimiento de mano derecha o izquierda, pues se observará un decremento mayor en el electrodo contralateral al de la mano involucrada en la tarea [31]. Sin embargo, los ERD / ERS generados durante la imaginación motora no pueden observarse directamente del registro EEG, por lo que se emplean diversos métodos de procesamiento para estimar los potenciales.

2.3.1 Cálculo de los potenciales ERD

El potencial ERD se define como la disminución de potencia en la señal, relativo a la potencia medida en un intervalo de referencia o línea base, que generalmente se selecciona varios segundos antes del inicio de este [36].

El cálculo de estos potenciales puede realizarse a través de diferentes métodos, entre los más utilizados se encuentran el método clásico [37] y el método de varianza entre ensayos [38], sin embargo, entre sus inconvenientes se encuentra la gran variabilidad generada en la señal, que impide obtener una buena resolución temporal. Para reducir este fenómeno es necesario promediar entre conjuntos de muestras correspondientes a intervalos de tiempo cuya dimensión depende de las repeticiones disponibles, entre más pequeño sea, más grande tendrá que elegirse el intervalo [36].

Alternativo a los métodos previos, si se pretende mantener un compromiso entre la resolución temporal y frecuencial puede ser utilizada la Transformada Wavelet con una

función madre Morlet [36, 39]. Con este método se obtiene una buena resolución temporal, sin embargo, su tiempo de procesamiento es elevado.

Para obtener una mejor resolución temporal, sin aumentar significativamente el tiempo de procesamiento, es posible caracterizar los potenciales ERD al aplicar la Transformada de Hilbert. Este método utiliza la frecuencia de muestreo original para generar una envolvente que conecta los picos de la señal de interés, generando una mejor visualización de los ERD [40].

El procedimiento utilizado para calcular el ERD con este método se describe en [41] y se enuncia a continuación:

1. La señal de EEG debe ser filtrada en la banda de frecuencias correspondientes al ritmo alpha.
2. Aplicar la transformada de Hilbert en tiempo discreto para obtener la señal analítica. Este cálculo de realiza en los canales C3 y C4 de manera independiente, a partir de este paso todos los cálculos deben realizarse por separado para cada uno de los canales.
3. Se elevan al cuadrado las muestras de las señales analíticas, de manera posterior se calcula la señal de potencia y se promedia entre el total de ensayos o repeticiones.
4. Calcular el porcentaje de cambio entre las señales de potencia y el promedio de potencia en un intervalo de referencia.

2.4 Interfaces cerebro computadora

Las interfaces cerebro computadora (BCI, por sus siglas en inglés) son sistemas de comunicación que traducen la modulación voluntaria de la actividad neuronal en información que puede ser utilizada en aplicaciones de comunicación y control [18, 42].

Estos sistemas se basan en el procesamiento de cambios medibles, conocidos como señales de control, que se generan a nivel cerebral durante tareas cognitivas, dependiendo del autor pueden ser exógenos y endógenos [43] o activos, reactivos y pasivos [18]. Los BCI exógenos o reactivos se caracterizan por utilizar señales de control que son respuestas naturales del cerebro a estímulos externos, y no requieren de periodos de entrenamiento para lograr un buen desempeño por el usuario [3, 42, 44]. Un ejemplo de BCI exógenos son las oscilaciones en las señales de EEG generadas en la misma frecuencia al excitar la retina con estímulos visuales repetitivos a una frecuencia específica [44, 45].

Por otro lado, los BCI endógenos o activos se basan en la autorregulación de la actividad cerebral, es decir que son controladas directa y concisamente por el usuario y no dependen de un estímulo externo [18, 44]. Un ejemplo de ellas son las basadas en imágenes motoras (MI-BCI), estas utilizan como señales de control los SMR, que surgen a partir de la imaginación de movimiento y desencadenan los potenciales ERD y ERS [43]. En este tipo de BCI es necesario realizar entrenamiento con retroalimentación basados en la imaginación motora, para que el usuario aprenda a modificar de manera voluntaria sus propios ritmos y potenciales cerebrales [3].

Durante el entrenamiento se estimulan las áreas corticales relacionadas con el movimiento utilizando técnicas como la observación de acción y la sensación de propiedad, en donde las tareas de imaginación motora son solicitadas a través de modelos virtuales mostrados de manera visual lo que estimula las neuronas espejo [46] y permite que el usuario experimente tener el control motor sobre sus propias acciones [47].

En la figura 2.8 se ilustra como los sistemas BCI integran los procesos de adquisición de la señal cerebral, procesamiento de la señal adquirida, extracción y selección de características y clasificación, seguido del desarrollo de una interfaz de control [48].

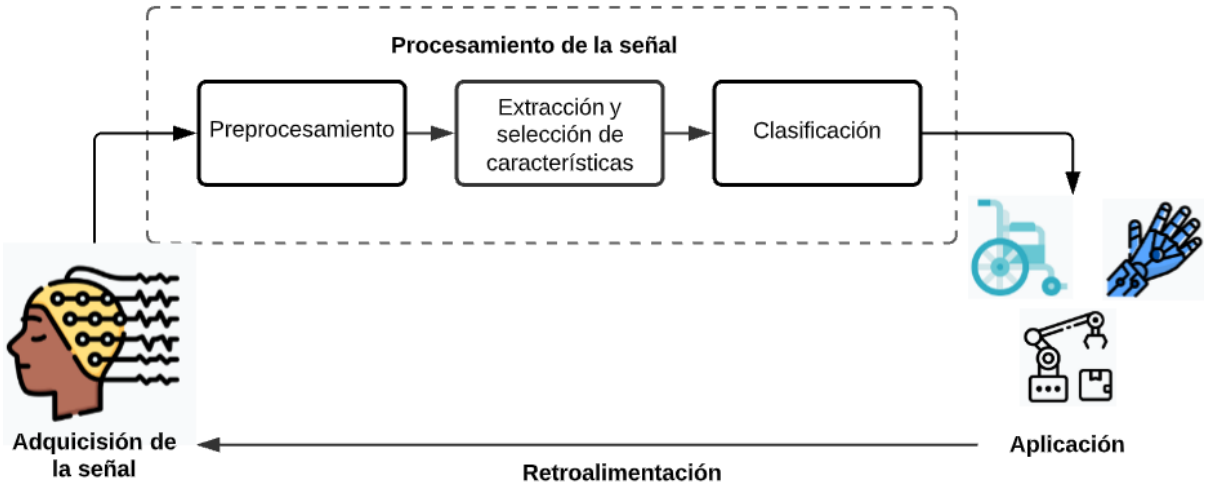

Fig. 2.8 Etapas de un BCI.

2.4.1 Adquisición de la señal

Una de las técnicas más utilizadas por las BCI para registrar la actividad cerebral es la de EEG debido a su carácter no invasivo, su costo de implementación y la buena resolución temporal de los registros obtenidos [42, 49, 50]. A través de esta técnica es posible registrar diversos fenómenos cerebrales que pueden ser utilizados como señales de control en un BCI, algunos de estos fenómenos son los potenciales visuales evocados, potencial P300, potenciales corticales cortos o los SMR, mencionados previamente [43].

2.4.2 Preprocesamiento de la señal

Durante la adquisición de la señal (registros de EEG) es posible registrar señales no deseadas asociadas al ruido y artefactos. El primero incluye fuentes de ruido externas y ambientales como el ruido de línea, la iluminación, etc. Mientras que los artefactos son señales fisiológicas relacionadas con potenciales bioeléctricos o fluctuaciones en la resistencia de la piel. Normalmente estos son involuntarios y no pueden evitarse durante la adquisición. Algunos ejemplos de artefactos son las

señales electrooculográficas (EOG), electromiográficas (EMG), electrocardiográficas (ECG) y la respiración [50].

Durante la etapa de preprocesamiento se pretende remover los componentes no deseados para mejorar el contenido obtenido con el registro cerebral y mejorar la relación señal a ruido (SNR, por sus siglas en inglés) [1]. Algunos de los algoritmos utilizados incluyen el submuestreo, filtros en frecuencia, filtros espaciales, selección de canales EEG, optimización en bandas de frecuencia, separación ciega de fuentes, entre otros. En la Tabla 2.1, se indican algunos de los filtros frecuenciales y espaciales más utilizados por los sistemas BCI durante el preprocesamiento de la señal [51].

Tabla 2.1 Filtros frecuenciales y espaciales comúnmente utilizados en el preprocesamiento de señales biomédicas.

Filtros frecuenciales	Filtros espaciales
Filtro pasa banda. Utilizada para eliminar artefactos distribuidos en un rango de frecuencia específico [52]. **Filtro notch.** Normalmente se implementa para eliminar el ruido ocasionado por la línea eléctrica [53].	**Derivación Laplaciana superficial (SLD, por sus siglas en inglés).** Calcula la diferencia entre el canal y el peso promediado de cuatro de sus canales vecinos [54]. **Referencia promedio común (CAR, por sus siglas en inglés).** Elimina la actividad promediada de EEG al sustraer el valor promedio de los otros canales [55].

2.4.3 Extracción y selección de características

La extracción de características se encarga de encontrar y organizar en un “vector de características” al conjunto de atributos más distintivo e informativo de la señal. Este vector representará a las señales originales y servirá para construir modelos que permitan estudiar los fenómenos de interés [56]. Comúnmente, las características se calculan utilizando métodos de procesamiento en dominio del tiempo, de la frecuencia

o del espacio, algunos ejemplos de características que pueden ser extraídas en una señal de EEG se muestran en la Tabla 2.2

En ocasiones, durante el proceso de extracción de características pueden generarse datos redundantes o que aportan poca información [57]. Por lo que suele establecerse un subconjunto del conjunto de características con la intención de mejorar el proceso clasificación [56]. Este proceso se conoce como selección de características y pretende retener la información más significativa de los registros originales [58].

Tabla 2.2. Se muestra algunos tipos de características que pueden ser extraídas en el dominio temporal, frecuencial, tiempo – frecuencia y espacial.

Características temporales	• Parámetros de Hjorth [59]. • Características estadísticas (promedio, mediana, desviación estándar, varianza, entre otros) [60].
Características frecuenciales	• Espectro de densidad de potencia (PSD, por sus siglas en inglés) [61]. • Energía [62]. • Transformada de Fourier [63].
Características en tiempo – frecuencia	• Transformada wavelet [64]. • Transformada de Hilbert [65].
Características espaciales	• Filtros comunes espaciales (CSP, por sus siglas en inglés) [66].

2.4.4 Clasificación

En la etapa de clasificación de un BCI se utilizan modelos de aprendizaje automático, estos utilizan programas informáticos y estadística para traducir el conjunto de características previamente extraídas, en comandos operativos [67].

Los algoritmos de clasificación son comúnmente categorizados como lineales y no lineales. Los modelos de clasificación lineal buscan establecer una relación lineal entre los datos de entrada y de salida, por su parte, los modelos no lineales permiten establecer una relación entre la entrada y la salida aun cuando no es posible construir una solución algorítmica entre ellas [48].

Los métodos de clasificación más utilizados en los sistemas BCI son el análisis discriminante lineal (LDA, por sus siglas en inglés) y las máquinas de vectores de soporte (SVM, por sus siglas en inglés), como modelos de clasificación lineal, y las redes neuronales artificiales (ANN, por sus siglas en inglés) y SVM con kernel de base radial, como modelos de clasificación no lineal [68].

2.4.4.1 Redes Neuronales Artificiales como clasificadores

Una red neuronal artificial modela la manera en la que el cerebro realiza tareas específicas o funciones de interés, realizando cálculos útiles a través un proceso de aprendizaje similar al de la plasticidad cerebral, en donde los pesos sinápticos se utilizan para almacenar los conocimientos adquiridos [69].

La unidad básica de la red neuronal es la neurona, esta se compone de tres elementos básicos que son los pesos sinápticos, el nodo de suma y la función de activación (ver figura 2.9) [70, 71]. Las entradas de la red neuronal son conocidas como instancias, y se refiere a cada uno de los datos de los que se dispone para hacer un análisis [71], es decir los vectores de características formados previamente (subsección 2.4.3).

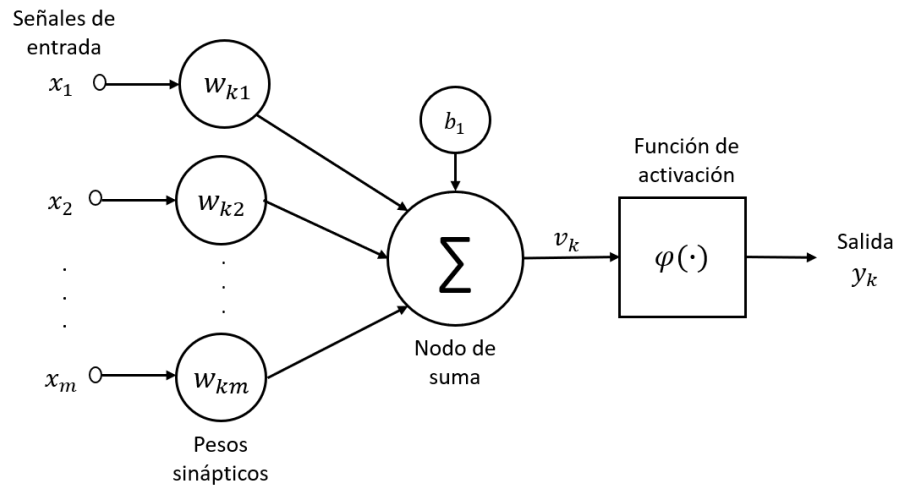

Fig. 2.9 Representación de una neurona artificial con sus elementos básicos.

De manera general, las instancias son ponderadas por su peso sináptico correspondiente, es decir, la señal x_j en la entrada de la sinapsis *j* conectada a una neurona *k* es multiplicada por un peso sináptico w_{kj}. La combinación lineal de estos elementos forma el potencial de activación v_k, que posteriormente pasa a través de una función de activación $\varphi(v)$, limitando la amplitud de la salida y_k de la neurona, descrita por la ecuación (2.1) [69].

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j\right) \tag{2.1}$$

Las funciones de activación $\varphi(v)$ definen la salida de la neurona en términos del campo local inducido v. Estas pueden ser lineales o no lineales y la selección de ellas depende de la aplicación de la red neuronal. Existen dos tipos básicos de funciones de activación, estos son la función de umbral y la función sigmoidal (ver figura 2.10).

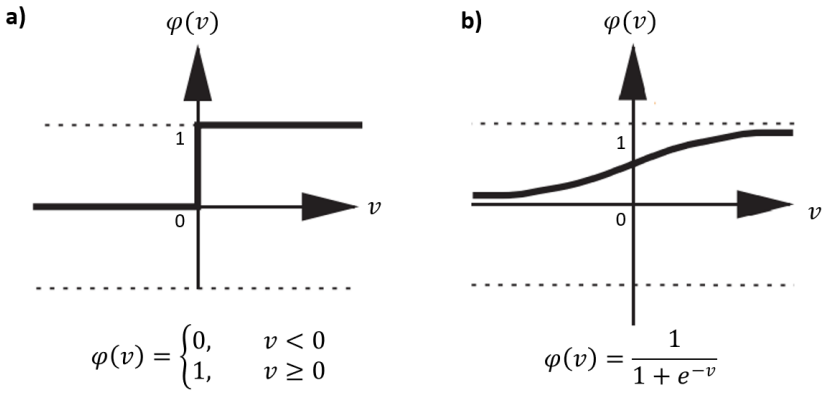

Fig. 2.10 a) Función de activación de umbral, b) Función de activación sigmoidal.

La función de umbral es una función lineal que toma el valor de 1 si el campo inducido de esa neurona no es negativo, de lo contrario tomará el valor de 0. En contraste con esta función, la función sigmoidal asume un rango continuo de valores entre 0 y 1 (sigmoidal unipolar), o en caso de ser necesario un rango entre -1 y 1 (sigmoidal bipolar).

Para los casos de clasificación es posible utilizar la función softmax en la capa de salida, definiendo al menos dos neuronas binarias. Esta función utiliza la ecuación (2.2) para calcular la probabilidad de que una de las neuronas de salida se active, de tal forma que la suma de los valores asignados a todas las neuronas de salida será igual a 1 [72].

$$y_k(x, w) = \frac{\exp\left(a_k(x, w)\right)}{\sum_j \exp\left(a_j(x, w)\right)} \tag{2.2}$$

Es posible contar con más de una neurona operando en paralelo, estas arquitecturas se conocen como redes neuronales artificiales multicapa [71]. Estas pueden estar distribuidas en una o más capas, en donde cada capa incluye los pesos sinápticos, nodos de suma, función de transferencia y el vector de salida, como se

muestra en la figura 2.11, las redes neuronales multicapa utilizan la salida de la capa anterior como entrada de la siguiente capa. El proceso de propagar de esta manera los atributos desde la entrada de la red hasta su salida se denomina como “retropropagación hacia adelante”.

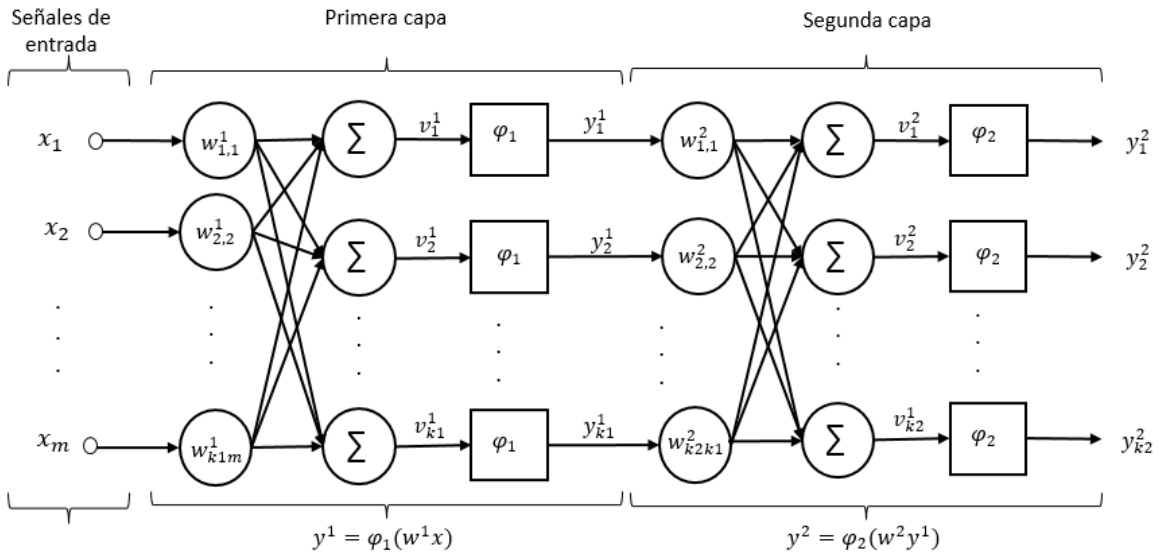

Fig. 2.11 Red neuronal artificial multicapa con dos capas ocultas. La salida de la capa anterior sirve como entrada para la siguiente capa.

El parámetro que afecta el comportamiento de la red es el conjunto de pesos, y la tarea del aprendizaje automático es encontrar el valor que deben tomar para sea posible la clasificación. Esto se logra a través del entrenamiento, cuyo objetivo es que el modelo aprenda a clasificar no solo las instancias con las que ha entrenado, si no que logre identificar datos que no fueron utilizados para entrenar la red, lo que se conoce como “capacidad de generalización de la red” [69].

Entrenamiento de la red neuronal

En un principio, los pesos son inicializados como números aleatorios pequeños (normalmente en el intervalo de -0.1 a 0.1) y se inicia la propagación hacia adelante

hasta la salida de la red. Una vez que se obtiene la salida se calcula la diferencia entre ella y el objetivo deseado.

Esta diferencia se conoce como error y a partir de él se determinará la manera en que se ajustarán los pesos. La función de error más común al utilizar redes neuronales artificiales es el error cuadrático medio (MSE, por sus siglas en inglés), este se define por la ecuación (2.3), donde el error es la media de la diferencia al cuadrado entre el objetivo t_n y la salida de la red y_n [70].

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(t_n - y_n)^2 \tag{2.3}$$

En problemas de clasificación en donde la salida se muestra como valores de probabilidad entre 0 y 1, se ha encontrado que utilizar la función de error de entropía cruzada en lugar del MSE conduce a entrenamientos más rápidos y una mejor generalización. La función de error de entropía cruzada se describe por la ecuación (2.4) Debido a su comportamiento logarítmico, el error incrementa rápidamente cuando la probabilidad obtenida por el clasificador diverge de su valor esperado [72].

$$E(w) = -\sum_{n=1}^{N}\{t_n ln\, y_n + (1 - t_n)\ln\,(1 - y_n)\} \tag{2.4}$$

Una vez calculado el error, la información es utilizada para ajustar los pesos a través de la retropropagación del error. Este algoritmo permite definir la influencia que tienen las neuronas en cada capa en el cálculo del error, evaluando la forma en la que varía el error de la función con respecto a los parámetros de la red.

Este algoritmo utiliza las derivadas parciales del error con respecto de los pesos asignados a las neuronas en la capa oculta y en las capas intermedias de la red neuronal. De esta manera, primero se calcula el error en la capa de salida y este es propagado hacía atrás para calcular el error en las capas ocultas como se muestra a continuación [69].

- Calculando la última capa:

Derivada parcial del error con respecto al peso, en donde la última capa se define con el índice superior L:

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w^L} \tag{2.5}$$

En donde $\frac{\partial C}{\partial a^L}$ calcula la forma en la que varía el error con respecto a la función de la activación (en la última capa la activación es la salida de la red), $\frac{\partial a^L}{\partial z^L}$ se refiere a la derivada de la activación con respecto a la suma ponderada de la neurona (derivada de la función de activación) y $\frac{\partial z^L}{\partial w^L}$ calcula la forma en la que varía la suma ponderada z^L con respecto a la variación de los parámetros. Por lo que $\frac{\partial z^L}{\partial w^L} = a_i^{L-1}$, donde a_i^{L-1} corresponde a la activación de la capa previa.

La influencia que tiene cada neurona se denomina “error imputado a la neurona” y se define por la ecuación (2.6) que define como varía el error de la función conforme se modifica la suma ponderada.

$$\delta^L = \frac{\partial C}{\partial z^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \tag{2.6}$$

Por lo tanto

$$\frac{\partial C}{\partial w^L} = \delta^L \cdot \frac{\partial z^L}{\partial w^L} = \delta^L \cdot a_i^{L-1} \tag{2.7}$$

- Calculando las capas ocultas:

Para calcular la derivada parcial del error con respecto al peso en las capas ocultas se utiliza la siguiente ecuación:

$$\frac{\partial C}{\partial w^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \cdot \frac{\partial z^{L-1}}{\partial w^{L-1}} \tag{2.8}$$

Al tratarse de retropropagación la mayoría de los términos han sido definidos por la capa de salida, por lo que únicamente se requiere calcular el término $\frac{\partial z^L}{\partial a^{L-1}}$ que indica la manera en la que la matriz de parámetros w que conecta ambas capas, mueve el error de una capa a la capa anterior por lo que $\frac{\partial z^L}{\partial a^{L-1}} = W^L$.

De esta manera se calcula el error imputado para cada neurona en las capas ocultas de la red neuronal.

$$\delta^{L-1} = \frac{\partial C}{\partial z^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}} \tag{2.9}$$

Por lo tanto:

$$\frac{\partial C}{\partial w^{L-1}} = \delta^{L-1} \cdot \frac{\partial z^{L-1}}{\partial w^{L-1}} = \delta^{L-1} \cdot a_i^{L-2} \tag{2.10}$$

Una vez que se conoce la derivada parcial de los pesos, se utiliza esta información para minimizar la función de error. El algoritmo más utilizado es la minimización por gradiente descendente, y se describe por la ecuación (2.11), en done ϵ es la tasa de aprendizaje.

$$w_{ji}^{(L)} = w_{ji}^{(L)} - \epsilon\delta^{(L)} \tag{2.11}$$

La tasa de aprendizaje es un parámetro de ajuste que determina el tamaño del paso en cada iteración mientras la función se aproxima al mínimo. Sin embargo, si este parámetro es muy pequeño se necesitará un gran número de pasos para llegar al mínimo, y si el valor es muy alto la función tenderá a oscilar evitando la convergencia [73].

Existen diversos algoritmos que buscan mejorar el proceso de aprendizaje minimizando el efecto de la tasa de aprendizaje, uno de ellos es el de retropropagación resiliente [74]. Este algoritmo no toma en cuenta la magnitud de la derivada parcial de los pesos, si no que se enfoca únicamente en su signo, fundamentándose en que un cambio de signo en la derivada parcial indica que se ha brincado un mínimo local. Como consecuencia de no considerar la magnitud de la derivada parcial, la tasa de aprendizaje no se ve afectada por la misma, acelerando el proceso de aprendizaje y reduciendo el costo computacional.

Durante el desarrollo de este algoritmo se introduce el valor de actualización Δ_{ij} que determina el tamaño de actualización del peso. Si existe un cambio de signo en la derivada parcial, Δ_{ij} decrecerá por un factor de η^{-}, por el contrario, cuando la derivada parcial conserva su signo Δ_{ij} incrementará para acelerar la convergencia. De esta manera, la minimización del error se realizará a partir de la función mostrada en la ecuación (2.12).

$$w_{ji}^{(L)} = w_{ji}^{(L)} + \Delta w_{ji}^{(L)} \tag{2.12}$$

Otro algoritmo de entrenamiento que busca mejorar el proceso de aprendizaje de las redes neuronales es el de retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos [71]. Este algoritmo agrega a la ecuación de gradiente descendente un parámetro conocido como momentum (μ), el cuál vuelve el proceso más estable y acelera la convergencia en la región menos profunda de la función de error.

El momentum actúa como un filtro pasa bajas permitiendo que la red ignore pequeñas características en la superficie de error y logre deslizarse a través de un mínimo, obteniendo una función para minimizar el erro como la que se muestra en la ecuación (2.13). Posteriormente la actualización de los pesos se basa en que el error disminuya o incremente dentro de los límites establecidos (comúnmente entre 1% y 5%), actualizando también la tasa de aprendizaje y momentum en cada iteración.

$$\Delta w_{ji}^{(L)} = -\eta \delta^{(L)} + \mu \Delta w_{ji}^{(L)} \tag{2.13}$$

Un inconveniente que puede presentarse durante el entrenamiento de las redes neuronales artificiales es el sobre ajuste, esto es observable cuando las redes memorizan fenómenos presentes solo en los datos con los que han sido entrenadas, por lo que al presentarles este conjunto de datos son perfectamente capaces de clasificarlos, pero al recibir información nueva no logra generalizar lo aprendido [69].

Una de las estrategias adoptadas para validar que el modelo de red neuronal no se encuentra sobre ajustado es separar el conjunto total de los datos disponibles en grupos de entrenamiento y validación. De esta manera los modelos se entrenarán únicamente con los datos del grupo correspondiente, posteriormente se utilizarán los

datos de validación como nuevas entradas a los modelos, con lo que se podrá observar la capacidad de generalización de la red neuronal diseñada.

2.4.4.2 Métricas de evaluación de desempeño de modelos clasificadores

Una etapa esencial en el desarrollo de modelos clasificadores es la evaluación de desempeño, en ella se determina la calidad del modelo a partir de su rendimiento y capacidad de generalización [75, 76].

Una forma de evaluar los modelos clasificadores es a partir de la conocida matriz de confusión (ver figura 2.12). Esta matriz integra los resultados de predicción de las clases involucradas en el problema de clasificación. A través de ellos se puede observar la manera en que el modelo se comporta, contabilizando los datos correctamente clasificados, junto con aquellos que son confundidos entre clases o no clasificados en ninguna clase [77].

Los resultados obtenidos pueden encontrarse en alguno de los siguientes casos:

- Verdadero positivo (VP): El objetivo y la predicción son positivas.
- Falso positivo (FP): El objetivo es negativo, pero se predice como positivo. Error de tipo 1 o sobreestimación.
- Verdadero negativo (VN): El objetivo y la predicción son negativas.
- Falso negativo (FN): El objetivo es positivo, pero se predice como falso. Error de tipo 2 o subestimación.
- No clasificados (NC): Son datos que no lograron ser clasificados como positivos ni negativos.

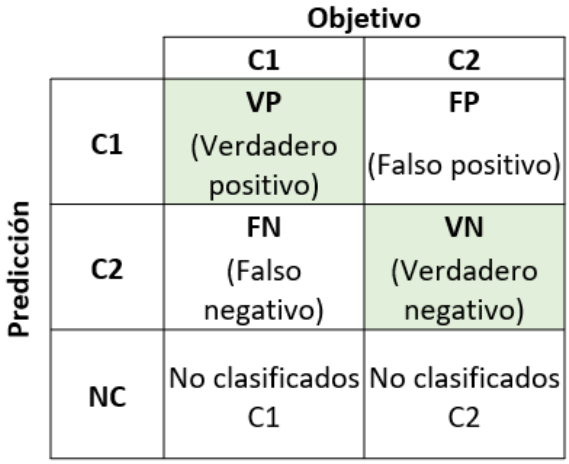

Fig. 2.12 Matriz de confusión con resultados de predicción.

A partir de las predicciones mostradas en la matriz de confusión pueden calcularse un gran número de medidas del desempeño en la clasificación [68, 76]. Algunas de las más utilizadas para evaluar modelos de clasificación basados en *machine learning* son:

Precisión. Esta métrica indica la calidad del modelo calculando la probabilidad de que los datos clasificados como positivos han sido etiquetados correctamente. Para ello se divide el total de datos clasificados como verdaderos positivos, entre el total de positivos predichos por el modelo.

$$Precision = \frac{VP}{VP + FP} \tag{2.14}$$

Sensibilidad. Proporciona información acerca de la cantidad de positivos que fueron identificados como tal. Por lo que se divide el total de datos verdaderos positivos, entre el total de datos que deberían ser clasificados como positivos. Un valor alto en sensibilidad indica un valor bajo de falsos negativos.

$$Sensibilidad = \frac{VP}{VP + FN} \tag{2.15}$$

Exactitud. Indica la frecuencia de clasificaciones correctas sobre un conjunto de datos proporcionados. Este se calcula sumando los datos clasificados correctamente y dividiéndolos entre el total de datos del conjunto.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.16}$$

Especificidad. Se define como la proporción de negativos reales, que fueron predichos como negativos. Se define dividiendo los verdaderos negativos entre el total de datos negativos dentro del conjunto. Un alto valor en especificidad indica una tasa baja de falsos positivos.

$$Especificidad = \frac{VN}{VN + FP} \tag{2.17}$$

F1-score. Permite comparar el rendimiento combinado de la sensibilidad y la precisión, asumiendo que ambas tienen la misma importancia.

$$F1 - score = \frac{VP}{VP + \frac{1}{2}(FN + FP)} \tag{2.18}$$

Kappa Cohen. Esta medida indica que tan bien se desempeñó el clasificador en comparación de que tan bien lo hubiera hecho simplemente por casualidad. Este se describe por la ecuación (2.19), en donde P_o corresponde a la exactitud observada y P_e es la exactitud esperada con base en los totales de la matriz de confusión. Utilizando

está métrica de evaluación se obtiene como resultado un coeficiente entre 0 y 1 que indica la probabilidad de que el resultado obtenido no se deba al azar. De esta forma en los coeficientes entre 0.1 y 0.2 esta probabilidad es mínima, entre 0.21 y 0.4 la probabilidad es débil, de 0.41 a 0.6 muestran una probabilidad moderada, entre 0.61 y 0.8 presentan una probabilidad fuerte y entre 0.81 y 1 indican que el resultado no se debe para nada al azar [78].

$$K = \frac{P_o - P_e}{1 + P_e} \tag{2.19}$$

En donde:

$$P_o = \frac{VP + VN}{VP + VN + FP + FN}$$

$$P_e = P_{corr} + P_{inc}$$

$$P_{corr} = \left[\frac{VP + FN}{VP + FN + VN + FP}\right] * \left[\frac{VP + FP}{VP + FN + VN + FP}\right]$$

$$P_{inc} = \left[\frac{FP + VN}{VP + FN + VN + FP}\right] * \left[\frac{FN + VN}{VP + FN + VN + FP}\right]$$

2.5 Estado del arte

La discriminación entre movimientos imaginarios se apoya en la correcta identificación de características que describen a las señales EEG relacionados con diversos movimientos [79]. Estudios recientes se apoyan del uso de interfaces cerebro – computadora basados en imágenes motoras (MI-BCI) debido a que utilizan herramientas y técnicas de retroalimentación que promueven la generación de potenciales ERD durante la ejecución de movimientos imaginarios, y son capaces de

discriminar entre tareas que involucran diversas extremidades utilizando métodos de clasificación tradicionales [80].

Algunas de las técnicas de retroalimentación utilizadas recientemente consisten en el uso de modelos anatómicos o ambientes virtuales que sirvan como guía visual para indicar la instrucción del movimiento imaginario que se solicita [81]. Así mismo, diversos trabajos han desarrollado estrategias como la sensación de propiedad y la inmersión para mejorar la experiencia de los participantes durante el entrenamiento [82]. También se integra la retroalimentación a través de modelos virtuales, pinzas robóticas o electroestimulación, como una herramienta que permita conocer al participante su desempeño al realizar tareas de imaginación motora.

En contraste, los métodos de procesamiento de las interfaces MI-BCI implementadas recientemente utilizan, en su mayoría, filtros de patrones comunes espaciales (CSP, por sus siglas en inglés) y algunos de sus derivados como herramienta para la extracción de características de las señales de EEG. Mientras que, en el caso de los métodos de clasificación, la mayoría utiliza modelos basados en LDA o SVM.

A continuación, se presentan algunos de los trabajos de investigación reportados recientemente, en los que se han diseñado diversas estrategias para mejorar la clasificación del movimiento imaginario entre ambas manos, considerando que estas extremidades son las comúnmente estudiadas en estudio de la intención del movimiento.

En el año 2016, Liang [83] estudia el efecto generado en los potenciales ERD y ERS al utilizar guías visuales en forma de movimientos dirigidos, para ello emplea la Transformada de Hilbert – Huang para extraer características en tiempo – frecuencia con las que logra clasificar correctamente el 76.8% de los movimientos imaginarios correspondientes a apertura y cierre, utilizando el método de LDA. Concluyendo que

el instruir a los participantes con este tipo de representación visual podría inducir potenciales que logran ser caracterizados con mayor facilidad.

De forma similar, en 2018 Skola [84] compara los potenciales generados por un grupo de participantes a quienes se les indicó el movimiento imaginario a realizar a través del paradigma de Graz [15], y un segundo grupo que observó el movimiento solicitado a través de un modelo virtual. Los resultados obtenidos mostraron que los participantes que recibieron guías visuales basadas en modelos virtuales generaban potenciales más intensos en menos sesiones de entrenamiento.

Con base a los resultados obtenidos, Skola [85] diseña en 2019 un videojuego de realidad virtual en el que se solicita a los participantes realizar movimientos de sujeción a través de modelos virtuales de ambas manos, logrando clasificar correctamente el 75.84% de las tareas solicitadas. En ambos trabajos, las características de las señales de EEG son extraídas con filtros CSP y clasificadas con LDA. Los modelos virtuales desarrollados por Skola se muestran a los participantes a través de la plataforma de desarrollo Unity, utilizando un socket local TCP (del inglés *Transmission Control Protocol*) para establecer comunicación entre la plataforma y el software de procesamiento.

Durante el año 2019, Korik [86] se apoya de modelos virtuales para clasificar movimientos entre ambas manos hacia un objetivo mostrado en un espacio 3D. Para ello extrae características utilizando bancos de filtros CSP (FBCSP, por sus siglas en inglés), con lo cual logra clasificar correctamente el 70% de los movimientos utilizando el modelo LDA. En este trabajo de investigación, la retroalimentación también se presenta en tiempo real utilizando el protocolo de datagrama de usuario (UDP, por sus siglas en inglés) para conectar la plataforma Unity con un módulo de Simulink de Matlab® dedicado al procesamiento de señales en tiempo real.

También en 2019, podemos encontrar trabajos como el realizado por Chowdhury [87], en el que la retroalimentación es mostrada con el movimiento de una pinza robótica conectada a un microcontrolador a través del software Matlab®. La pinza realiza los movimientos de apertura y cierre dependiendo del movimiento imaginario identificado, utilizando como clasificador un modelo de SVM que empleando un kernel de tipo lineal obtuvo hasta un 78.75% de exactitud. Un trabajo similar es el desarrollado por Xu [88] en el mismo año, en el que la retroalimentación consiste en el movimiento de un brazo robótico que se mueve a la derecha o izquierda dependiendo del movimiento imaginario clasificado por un modelo de LDA con el que se lograron identificar correctamente cerca del 70% de los movimientos imaginarios registrados.

En el año 2020, Choi [89] compara el efecto de entrenar participantes utilizando el mismo modelo virtual pero mostrado a un grupo en un ambiente de realidad virtual y a otro grupo en una pantalla colocada frente al participante. En los resultados experimentales se puede observar que al utilizar realidad virtual los participantes requieren un número menor de sesiones de entrenamiento para mejorar el desempeño de sus potenciales ERD generados. En este trabajo de investigación también se compara el desempeño de un modelo LDA, utilizado para clasificar los movimientos de ambas manos, obteniendo menos del 60% de movimientos clasificados correctamente.

Durante el año 2020, Choi [90] desarrolla un videojuego basado en realidad virtual en el que las tareas son presentadas a los participantes utilizando modelos virtuales de ambas manos. En este trabajo se estudia el efecto en los potenciales al comparar dos tipos de retroalimentación durante el entrenamiento. El primero consiste en girar ligeramente la cámara hacía el lado indicado, y en el segundo se muestra el movimiento de la mano virtual correspondiente, siendo este grupo quien presentó potenciales más fáciles de discriminar. En ambos trabajos la extracción de las características se realizó utilizando filtros CSP y la clasificación se llevó a cabo con modelos LDA obteniendo hasta un 75.04% de exactitud.

Achanccaray [91] en el año 2021, propone un modelo de entrenamiento basado en un ambiente de realidad virtual, en donde se realizan movimientos imaginarios de sujeción. En este trabajo propone acompañar la retroalimentación visual con estimulación electrotáctil, logrando clasificar correctamente hasta el 90% de las imágenes motoras, utilizando filtros CSP como método de extracción de características y SVM con un kernel RBF (del inglés, radial basis function) como modelo clasificador, estos fueron programados en el lenguaje **C**#, utilizando la librería LibSVM.

Capítulo 3. Metodología propuesta

En este capítulo se presenta la metodología propuesta. Esta se compone por dos etapas que pueden ser desarrolladas en paralelo y su conjunción converge en la manipulación del modelo virtual a través de comandos de control (ver figura 3.1).

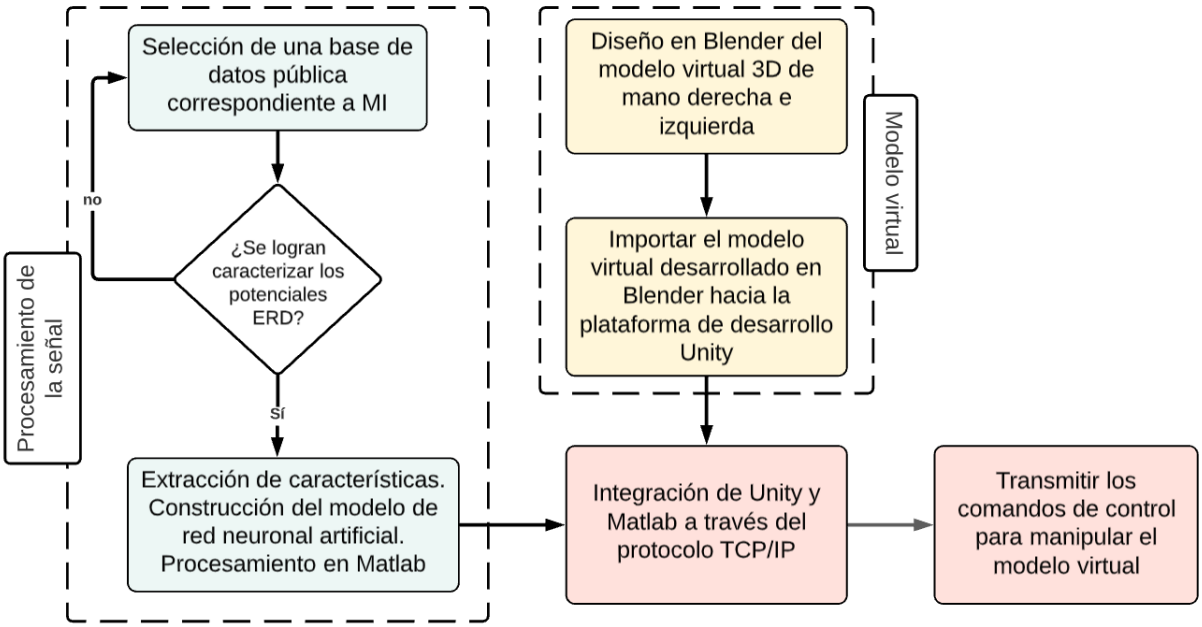

Fig. 3.1 Metodología propuesta

En la primera etapa se realiza el procesamiento de los registros de EEG obtenidos durante el desempeño de tareas de imaginación motoras de ambas manos, se estiman los ERD de cada participante y se desarrollan las etapas que componen el procesamiento de la BCI desarrollada (extracción de características y construcción del modelo clasificador basado en *machine learning*). Este procesamiento se desarrolla en Matlab® R2020b.

La segunda etapa corresponde al diseño del modelo virtual y las animaciones que integrarán el movimiento, que se desarrollan en la plataforma de diseño y animación Blender v2.93. Durante esta etapa, también se aborda la exportación de este modelo

a la plataforma de desarrollo Unity, en donde a partir de diversas estrategias propias de Unity se establecerán las condicionantes para ejecutar la simulación del modelo virtual.

Los resultados de las etapas previas se integran a través del protocolo TCP/IP para manipular el modelo virtual en Unity utilizando comandos de control enviados desde Matlab®, que responden a la clasificación de los modelos de clasificación basados en *machine learning*.

3.1 Etapa de procesamiento de la señal

3.1.1 Base de datos

La base de datos usada en este trabajo es “BCI Competition IV, Dataset IIa” (Institute of Neural Engineering Graz Brain-Computer Interface Lab) [92], integrada por 18 registros EEG de 9 participantes (2 registros por participante) nombrados como A01, A02, A03, A04, A05, A06, A07, A08 y A09. Estos registros fueron adquiridos durante tareas de imaginación motora correspondientes a movimiento de mano derecha o izquierda, movimiento de la lengua y movimiento de los pies.

La base de datos integra 288 ensayos por sesión para cada uno de los participantes. Estos ensayos fueron registrados en dos sesiones en donde cada una se compone de 6 corridas separadas por descansos cortos en donde se realizaron 48 ensayos (12 por cada clase de movimiento).

Al inicio de cada ensayo se presenta un estímulo acústico cuya duración es descrita como “corta”, acompañado de una pantalla negra en donde se muestra una cruz de fijación. Después de dos segundos, se muestra una flecha apuntando hacia la derecha, izquierda, arriba o abajo que indica el movimiento imaginario que debe realizarse (mano derecha, mano izquierda, lengua o pies respectivamente). La flecha permanece en la pantalla durante 1.25 s, sin embargo, se solicita a los participantes continuar con

el movimiento imaginario hasta el segundo 6, en donde la pantalla se muestra nuevamente negra. Este paradigma experimental puede observarse en la figura 3.2.

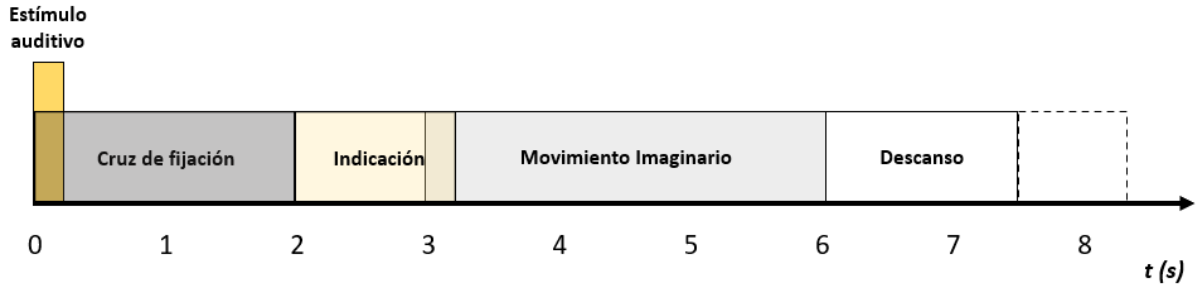

Fig. 3.2 Paradigma experimental utilizado en la base de datos “bci competition IV. Dataset IIa”

Para el registro se emplearon 22 electrodos de Ag/AgCl colocados sobre el cuero cabelludo a una distancia de 3.5 cm entre ellos, utilizando el mastoides izquierdo como referencia y el mastoides derecho como tierra. Adicional a los 22 electrodos mencionados, se utilizaron 3 electrodos para registrar la actividad ocular EOG (ver figura 3.3). La señal fue registrada con una frecuencia de muestreo de 250 Hz y filtrada entre 0.5 Hz y 100 Hz, incluyendo un filtro notch a 50Hz para suprimir el ruido de línea. La sensibilidad del amplificador fue configurada en 100 µV para el registro de EEG y en 1 mV para el registro de EOG.

La información de la base de datos se integra por dos variables para cada sujeto, una de ellas contiene las señales adquiridas, mientras que la otra es una estructura que contiene la información relevante de la señal como la frecuencia de muestreo y vectores que contienen la referencia de la posición en muestras de cada inicio de tarea, el tipo de movimiento imaginario realizado, así como el tiempo total de cada ensayo en particular.

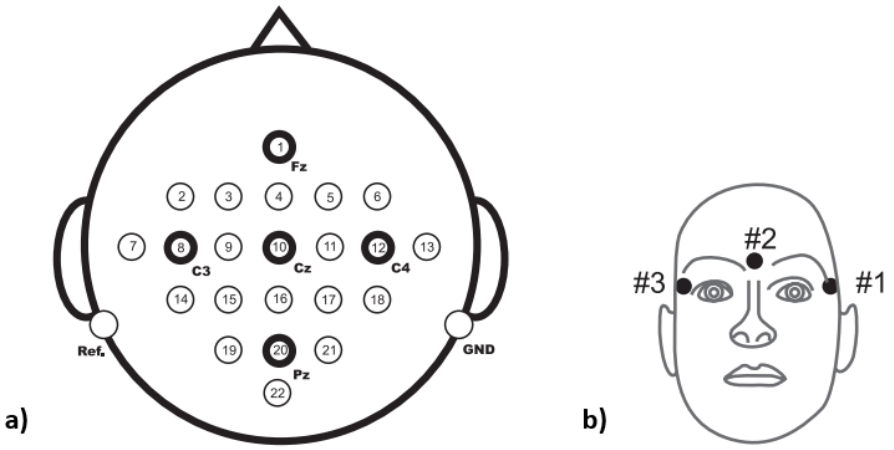

Fig. 3.3 (a) Distribución de los electrodos utilizados durante el registro de EEG, (b) Electrodos utilizados en el registro EOG [92].

Las señales adquiridas se encuentran guardadas en una matriz de dimensión Nx25, en donde N representa las muestras de todos los ensayos. En las columnas se encuentran los electros utilizados, las primeras 22 pertenecen al registro de EEG y se distribuyen como se muestra en la figura 3.3a, las siguientes 3 corresponden el registro EOG.

Debido a que los potenciales relacionados con el movimiento imaginario de las manos se generan en la corteza motora primaria, se seleccionó únicamente la información registrada por los electrodos C3 y C4 pues se localizan justo sobre el área descrita, de forma que se integra una matriz de dimensión Nx2.

El inicio de cada ensayo, así como la clase del movimiento imaginario solicitado se describen en la variable de tipo estructura de cada uno de los participantes. Esta información es utilizada durante el acondicionamiento de las señales, en donde la señal concatenada se separa en sus 288 ensayos utilizando como referencia las etiquetas que indican el inicio de cada uno de ellos.

Una vez separados los ensayos, se utilizaron las etiquetas correspondientes al tipo de movimiento para seleccionar únicamente los correspondientes al movimiento imaginario de mano derecha e izquierda, con lo que finamente se obtiene un total de 144 ensayos por participante, organizados en matrices de 144x2N, donde N se encuentra duplicada debido a que la matriz contiene la información de los electrodos C3 y C4, como se observa en la figura 3.4.

	muestra 1		muestra 2		...		muestra N	
Ensayos	C3	C4	C3	C4			C3	C4
1								
...								
144								

Fig. 3.4. Dimensionalidad de las matrices resultantes, las filas representan los ensayos disponibles (72 para cada clase) y las columnas representan las muestras presentes en cada ensayo, para los electrodos C3 y C4.

3.1.2 Estimación del ERD

La información contenida en cada una de las matrices resultantes fue separada por tipo de movimiento (clases) y canal de EEG (electrodo C3 y C4). Como resultado, por cada uno de los participantes se obtuvieron cuatro matrices de dimensión Nx72, en donde N es el total de muestras por ensayo, integradas de la siguiente manera:

- Señales registradas por el electrodo C3 durante la imaginación motora de la mano derecha.
- Señales registradas por el electrodo C4 durante la imaginación motora de la mano derecha.
- Señales registradas por el electrodo C3 durante la imaginación motora de la mano izquierda.
- Señales registradas por el electrodo C4 durante la imaginación motora de la mano izquierda.

Para estimar el ERD correspondiente a los movimientos imaginarios efectuados por los participantes, se utilizó un filtro Butterworth pasa banda entre los 8 Hz y 14 Hz asociado con el ritmo alpha, y se seleccionó una ventana de tiempo de 2 s que comprende desde los 0.5 s hasta los 2.5 s después de la indicación del movimiento imaginario en la tarea experimental (ver figura 3.5) [93–95].

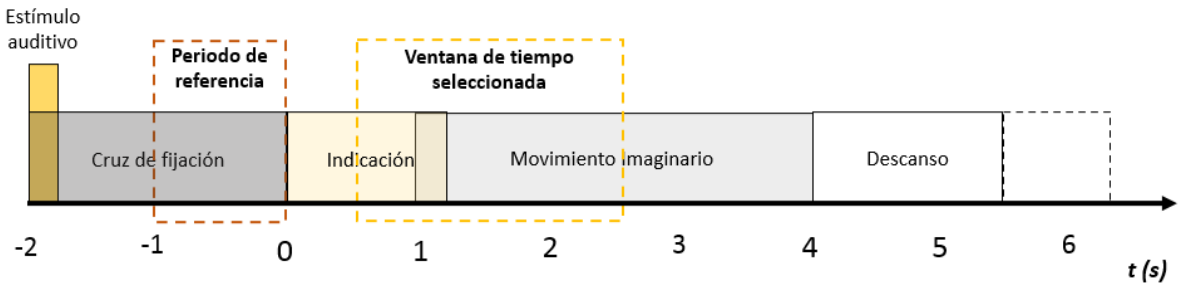

Fig. 3.5 Ventana de tiempo y periodo de referencia seleccionados para el cálculo de los ERD.

Se empleo el método de Transformada de Hilbert para calcular los potenciales ERD de cada uno de los participantes, debido a que ofrece una mejor resolución temporal comparada con los métodos de varianza entre ensayos y clásico, y requiere menor tiempo de procesamiento que el método de Transformada Wavelet con función madre Morlet.

El procesamiento se realizó siguiendo el método descrito por [41], que consiste en los siguientes pasos:

1. Filtrar la señal en la banda de frecuencia alpha y calcular la Transformada de Hilbert discreta en los electrodos C3 y C4 para obtener las señales analíticas en tiempo discreto del movimiento imaginario de mano izquierda ($IC_{3\alpha}[n]$) y derecha ($DC_{3\alpha}[n]$), como se muestra en las ecuaciones (3.1) y (3.2).

$$IC_{3\alpha}[n] = Ic_{3\alpha}[n] + jIc_{3\alpha H}[n] \tag{3.1}$$

$$DC_{3\alpha}[n] = Dc_{3\alpha}[n] + jDc_{3\alpha H}[n] \tag{3.2}$$

2. Para obtener señales de potencia, se eleva al cuadrado cada una de las muestras de las señales analíticas de tiempo discreto y se promedian a través de todos los ensayos. Esto se describe en las ecuaciones (3.3) y (3.4) en donde $IC_{3\alpha}^2[m,n]$ y $DC_{3\alpha}^2[m,n]$ representan la n^{th} muestra de potencia del m^{th} ensayo, y N_I, N_D representan el total de ensayos correspondientes a movimientos imaginarios izquierdos y derechos, respectivamente.

$$P_{IC3\alpha}[n] = \frac{1}{N_I}\sum_{m=1}^{N_I} IC_{3\alpha}^2\,[m,n] \tag{3.3}$$

$$P_{DC3\alpha}[n] = \frac{1}{N_D}\sum_{m=1}^{N_D} DC_{3\alpha}^2\,[m,n] \tag{3.4}$$

3. Finalmente, se calcula el porcentaje de cambio entre las señales de potencia y el promedio de potencia en un intervalo de referencia. Las ecuaciones (3.5) y (3.6) definen el porcentaje de cambio, en donde $P_{IC_{3\alpha Evento}}$, $P_{DC_{3\alpha Evento}}$ son señales de potencia durante el periodo relacionado al evento, y $P_{IC_{3\alpha Ref}}$, $P_{DC_{3\alpha Ref}}$ representan la señal promediada en el periodo de referencia.

$$\%EP_{IC_{3\mu}}[n] = \frac{P_{IC_{3\mu Evento}}[n]}{P_{IC_{3\mu Ref}}} x100\% \tag{3.5}$$

$$\%EP_{DC_{3\mu}}[n] = \frac{P_{DC_{3\mu Evento}}[n]}{P_{DC_{3\mu Ref}}} x100\% \tag{3.6}$$

Este procedimiento se realizó de manera individual con las cuatro matrices obtenidas para cada participante, con las que se muestra el comportamiento del potencial en los electrodos C3 y C4 durante los movimientos imaginarios de la mano derecha o izquierda.

3.1.3 Extracción de características temporales

Para la extracción de características temporales se utilizaron ventanas de 2s, relacionadas con el tiempo de ejecución motora [37].

Las características temporales son: la media, mediana, desviación estándar, varianza, asimetría y curtosis, para representar la distribución de las señales correspondientes a MI [21].

La media (3.7) y la mediana (3.8) son medidas de tendencia central, es decir que indican el centro o punto sobre el que gravita el conjunto de una serie de números a partir de los cuales se calcula. La media define el valor promedio de todos los datos y la mediana divide a la serie de datos en dos partes iguales, en donde la mitad de los valores son menores y la otra mitad son mayores [96].

$$\bar{x} = \frac{\sum_i^n x_i}{n} \tag{3.7}$$

$$Me = x_{\frac{n+1}{2}} \tag{3.8}$$

La varianza (3.9) y la desviación estándar (3.10) son medidas de dispersión e indican que tan separados se encuentran los valores entre ellos. La varianza mide la mayor o menor dispersión de los valores de la variable respecto a la media, un valor de varianza elevado representa una mayor dispersión y por tanto menor

representatividad de la media. La desviación estándar es la desviación en promedio de las diferencias de los valores con respecto a su media [97].

$$\sigma^2 = \frac{\sum_i^n (\bar{x} - x_i)^2}{n} \tag{3.9}$$

$$\sigma = \frac{\sqrt{\sum_i^n (\bar{x} - x_i)^2}}{n} \tag{3.10}$$

Por su parte, la asimetría (3.11) y curtosis (3.12) son medidas de forma que informan la deformación horizontal y vertical de la distribución. La primera de ellas describe la forma de una distribución alrededor de su media, mientras que la curtosis mide si la distribución de los datos, en relación con un valor normal es plana o tiene un pico.

$$skew = \frac{\frac{\sum_i^n (\bar{x} - x_i)^3}{n}}{\left[\frac{\sum_i^n (\bar{x} - x_i)^2}{n-1}\right]^{3/2}} \tag{3.11}$$

$$kurt = \frac{\frac{\sum_i^n (\bar{x} - x_i)^4}{n}}{\left[\frac{\sum_i^n (\bar{x} - x_i)^2}{n-1}\right]^2} - 3 \tag{3.12}$$

De igual forma, se calculó la energía de las señales en dominio temporal, para los ritmos cerebrales δ (0.5 – 4 Hz), θ (4 – 8 Hz), α (8 13 Hz) y β (13 – 30 Hz), esta característica permitirá diferenciar irregularidades en la amplitud de las señales en las diferentes frecuencias, y al ser calculada en dominio temporal requiere un tiempo de procesamiento menor [98]. Para delimitar la frecuencia de la señal en cada uno de estos ritmos, se utilizaron filtros Butterworth de tercer orden, posteriormente la energía

se calculó para cada una de las señales filtradas como se describe en la ecuación (3.13), en donde *N* el número total de muestras y $x(n)$ representa a la señal de EEG filtrada en el ritmo de interés, por lo que esta ecuación se repitió para cada banda de frecuencias.

$$E = \frac{1}{N}\sum_{n=1}^{N}|x(n)|^2 \tag{3.13}$$

De esta manera se calcularon 10 características para cada uno de los ensayos, están se integraron por la media, mediana, desviación estándar, varianza, asimetría, curtosis, energía en δ, energía en θ, energía en α y energía en β. Las características fueron calculadas para cada uno de los electrodos y normalizadas entre 0 y 1 de manera independiente, posteriormente se concatenaron para formar vectores de características de dimensión 1x20, que representan a la señal en cada uno de los ensayos (ver figura 3.6).

Vector de características																			
1		2		3		4		5		6		7		8		9		10	
C3	C4	C3	C4	C3	C4	C3	C4	C3	C4	C3	C4	C3	C4	C3	C4	C3	C4	C3	C4
Media $\bar{x}$		Desviación estándar σ		Asimetría $skew$		Curtosis $kurt$		Mediana Me		Varianza σ^2		Energía en delta E_δ		Energía en theta E_θ		Energía en alpha E_α		Energía en beta E_β	

Fig. 3.6 Vector de características

3.1.4 Clasificación

Para la construcción del clasificador se optó por uso de redes neuronales artificiales, considerando que son particularmente útiles para tareas complejas de clasificación y reconocimiento de patrones, por lo que son ampliamente utilizadas para la clasificación de biopotenciales como el EEG [99]. Estos se encargan de diferenciar entre los movimientos de mano derecha e izquierda a partir del vector de características generado y de proporcionar los comandos de control que están

destinados a operar el modelo virtual 3D. Los modelos de red neuronal artificial se programaron en el sistema de cómputo numérico Matlab R2020b, utilizando la librería *Deep Learning* v.14.1.

Para garantizar que los modelos neuronales no sean susceptibles a posibles tendencias presentes en los datos, los ensayos fueron aleatorizados para cada una de las clases y posteriormente mezclados. Este proceso de aleatorización se muestra en la figura 3.7, en donde N representa el número total de ensayos disponibles para cada clase.

En la figura 3.7a se observa que las características correspondientes a cada clase se colocan inicialmente en una sola matriz en donde las primeras columnas representan a la clase 1 y posteriormente se representa la clase 2. De esta manera al aleatorizar las filas (que corresponden a los ensayos) ambas clases se distribuirán de la misma manera obteniendo la aleatorización que se muestra en la figura 3.7b. Por último, los ensayos aleatorizados se alternan entre clases resultando en una matriz como la que se muestra en la figura 3.7c, en donde las filas representarán los 144 ensayos disponibles.

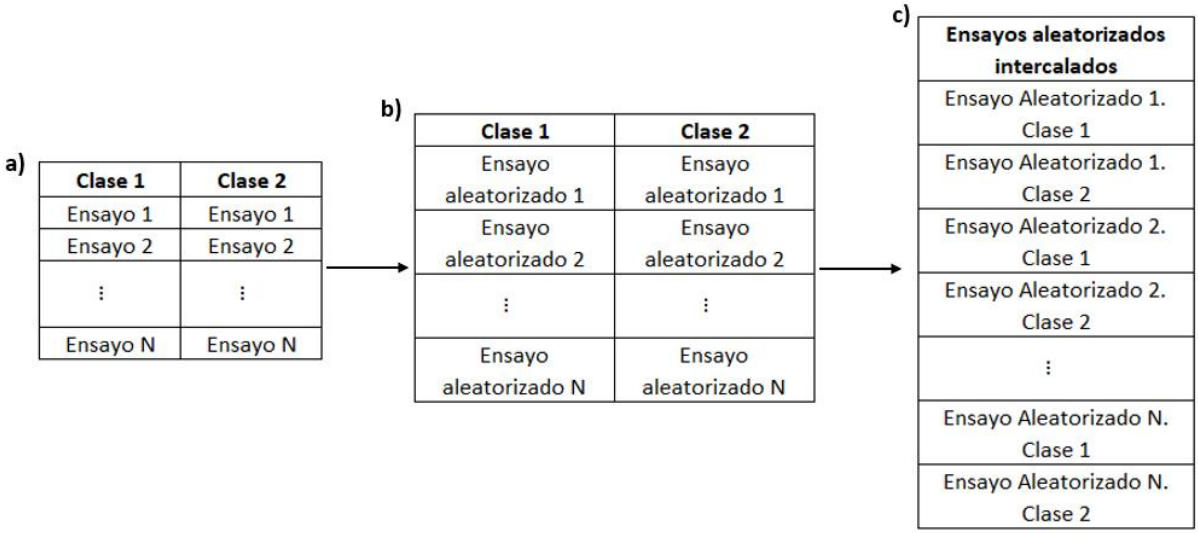

Fig. 3.7 a) Ensayos previos a la aleatorización, separados por clases. b) Ensayos aleatorizados, separados por clases. c) Ensayos aleatorizados con las clases intercaladas.

El conjunto de datos disponibles se separó seleccionando los primeros 80 ensayos aleatorizados para el entrenamiento (55% de los datos disponibles) y los 64 ensayos aleatorizados restantes para validar el desempeño de las redes neuronales artificiales (45% de los datos disponibles). Debido a la manera en la que se han aleatorizado los datos, implica que se seleccionaron 40 ensayos por clase para la etapa de entrenamiento y 32 ensayos por clase para la validación de los modelos.

Durante el entrenamiento se utilizó el método de validación cruzada *k-fold.* Este método es una estrategia de partición de la información que permite utilizarla de forma efectiva para analizar la capacidad de generalización de los clasificadores [69]. Para ello, el total de datos de prueba son separados en *k*-grupos independientes, de ellos *k-1* grupos son utilizados para entrenar un modelo, mientras que las pruebas se realizan con el grupo restante. Este procedimiento se repite *k* veces obteniendo el desempeño de cada grupo, mismo que posteriormente es promediado entre todos los grupos. En este trabajo se seleccionó un valor de k=10, por lo que en cada grupo se utilizaron 36 ensayos por clase para entrenamiento y 4 ensayos por clase para probar los modelos.

3.1.4.1 Diseño de los modelos de red neuronal

Los vectores de características se definieron como la entrada de la red neuronal obteniendo un total de 20 neuronas de entrada, mientras que a la salida se definieron dos neuronas que forman una salida binaria (ver figura 3.8) en donde la combinación (0 1) representa a la clasificación de un movimiento derecha y la combinación (1 0) a la clasificación de un movimiento izquierdo.

Se seleccionó el logaritmo sigmoidal como función de activación en las capas ocultas, mientras que en las capas de salida se utilizó la función softmax, definida así

debido a que devuelve el porcentaje de probabilidad de que una de las dos neuronas en la salida binaria se active. Para ello se consideró una incertidumbre del 30%, de esta manera se vuelve necesario que la probabilidad de que una de las neuronas se active sea de al menos 70%, de lo contrario el movimiento no será clasificado.

Para calcular el error durante el algoritmo de retropropagación se utilizó el método de entropía cruzada, recomendado para problemas de clasificación en donde la salida se indique por un valor de probabilidad entre 0 y 1, debido a su comportamiento logarítmico esta función incrementa rápidamente el error cuando este se aleja del valor esperado [72].

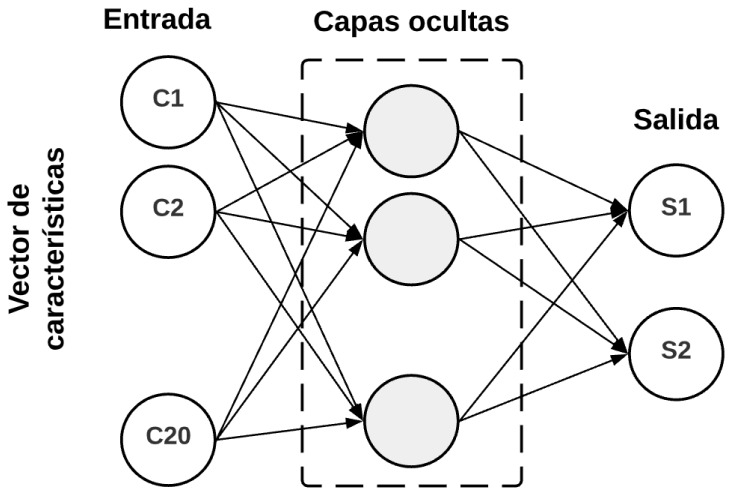

Fig. 3.8 Arquitectura general de la red neuronal diseñada

Se realizaron pruebas con tres algoritmos de entrenamiento distintos con el objetivo de encontrar aquel que, en conjunto con las características temporales definidas, obtenga el mejor desempeño durante la clasificación de los movimientos imaginarios. Los tipos de entrenamiento implementados fueron retropropagación resiliente, retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos y retropropagación por secante de un paso.

Los algoritmos de retropropagación resiliente y de retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos se encuentran dentro de las técnicas adaptativas locales, pues utilizan la información de la derivada parcial, para adaptar parámetros específicos como los pesos y la tasa de aprendizaje [73]. El primero de ellos realiza el ajuste de los pesos de cada neurona dependiendo únicamente del signo de la derivada parcial calculada en la retropropagación del error, al existir un cambio en el signo indica que la función ha pasado un mínimo local [74]. Por su parte, el segundo algoritmo toma en cuenta el valor de la derivada parcial obtenida durante la retropropagación para ajustar la tasa de aprendizaje y determinar el tamaño del paso que se realizará en la función para localizar el mínimo local [73].

Por otro lado, el algoritmo de retropropagación por secante de un paso es una técnica adaptativa global, pues utiliza la información de cada peso en la red para modificar la dirección de la búsqueda del gradiente descendente y definir el tamaño de los pasos de manera eficiente [100].

Durante las pruebas realizadas con los distintos algoritmos de entrenamiento, la topología de la red se mantuvo constante para cada sujeto, ajustando únicamente los parámetros de entrenamiento de tal forma que favorecieran a la clasificación.

La Tabla 3.1 muestra la topología de la red de cada uno de los participantes, en ella se describe el número de capas ocultas y la cantidad de neuronas que las componen, mientras que las Tabla 3.2 y Tabla 3.4 muestran los parámetros utilizados en cada algoritmo de entrenamiento probado.

Tabla 3.1 Topología seleccionada por participante para los modelos de red neuronal diseñados.

	Participantes								
	A01	A02	A03	A04	A05	A06	A07	A08	A09
Número de capas ocultas	2	1	1	2	2	1	2	2	1

Número de neuronas en las capas ocultas	19, 2	15	19	19, 6	19, 2	21	21, 2	21, 3	16
Función de activación en las capas ocultas	logsig, logsig	logsig	logsig	logsig, logsig	logsig, logsig	logsig	logsig, logsig	logsig, logsig	logsig
Función de activación en la capa de salida	softmax								

Tabla 3.2. Parámetros de entrenamiento seleccionados para los modelos neuronales utilizando el algoritmo de retropropagación resiliente

Retropropagación resiliente	Participantes								
	A01	A02	A03	A04	A05	A06	A07	A08	A09
Límite de épocas	1000								
Error de entrenamiento	0.001	1e-4	0.014	0.001	0.001	1e-4	1e-4	0.001	1e-4
Tasa de aprendizaje	0.01	0.03	0.03	0.03	0.01	0.03	0.01	0.05	0.03
Gradiente de rendimiento mínimo	1e-5								
Incremento de cambio en el peso	1.2								
Decremento de cambio en el peso	0.5								
Cambio inicial en los pesos	0.07								

Tabla 3.3. Parámetros de entrenamiento seleccionados para los modelos neuronales utilizando el algoritmo de retropropagación por secante de un paso

Retropropagación por secante de un paso	Participantes								
	A01	A02	A03	A04	A05	A06	A07	A08	A09
Límite de épocas	1000								
Gradiente de rendimiento mínimo	1e-10	1e-10	1e-10	1e-10	1e-10	1e-9	1e-12	1e-10	1e-11
Error de entrenamiento	0.001	1e-4	0.014	0.001	0.001	1e-4	0.001	0.001	1e-4

Tabla 3.4. Parámetros de entrenamiento seleccionados para los modelos neuronales utilizando el algoritmo de retropropagación de gradiente descendiente con tasa de aprendizaje y momentum adaptativos

Retropropagación con tasa de aprendizaje y momentum adaptativos.	Participantes								
	A01	A02	A03	A04	A05	A06	A07	A08	A09
Límite de épocas	2000								

Tasa de aprendizaje	0.03	0.03	0.03	0.04	0.02	0.03	0.03	0.03	0.03
Incremento en la tasa de aprendizaje	1.04	1.03	1.04	1.04	1.02	1.01	1.04	1.03	1.04
Decremento en la tasa de aprendizaje	0.7								
Momentum	0.7	0.7	0.6	0.8	0.6	0.7	0.7	0.7	0.7
Error de entrenamiento	0.001	1e-4	0.014	0.001	0.001	1e-4	1e-4	0.001	1e-4
Gradiente de rendimiento mínimo	1e-5	1e-5	1e-5	1e-5	1e-5	1e-7	1e-5	1e-5	1e-5

3.1.4.2 Validación de los modelos neuronales

El desempeño de los modelos neuronales se calculó a partir de la exactitud, precisión, sensibilidad, especificidad, f1-score y el coeficiente de Kappa. Para ello se utilizaron los ensayos aleatorizados separados previamente como conjunto de validación. Estos corresponden al 45% del total del conjunto de datos disponibles, y representan instancias nuevas para los modelos neuronales diseñados.

Adicional a ello, se validó la robustez de las redes neuronales artificiales contaminando con ruido blanco los registros de EEG correspondientes al conjunto de datos para validación, siguiendo la metodología propuesta en [101]. La contaminación de las señales se realizó previo a su procesamiento, añadiendo 10%, 25%, 40% y 50% de ruido blanco, equivalente al nivel de potencia absoluto del registro original, asegurando que la contaminación de los registros de EEG sea proporcional a la energía contenida en ellos. El cálculo de la potencia absoluta y la energía de la señal se realizó con las ecuaciones (3.14) y (3.15), en donde $x(n)$ es el elemento n-ésimo del registro y N se refiere al total de muestras en el registro.

$$P_{dBW} = 10log_{10}P \tag{3.14}$$

$$E = \frac{1}{N}\sum_{n=1}^{N}|x(n)|^2 \tag{3.15}$$

Una vez contaminados los registros de EEG, se extrajeron características para cada nivel de ruido añadido, siguiendo el procedimiento mencionado en la sección 3.1.3. Posteriormente estos vectores de características fueron utilizados como entradas para los modelos neuronales diseñados en la etapa de entrenamiento.

La contaminación con ruido blanco de las señales de EEG tiene como propósito generar cambios significativos en los registros, de tal forma que estos resulten en una señal distinta a la que fue mostrada con los datos de prueba originales. De esta manera se esperaría ver que conforme se incremente el nivel de ruido blanco a las señales, el modelo clasificador presente mayor dificultad al identificar los movimientos imaginarios.

3.2 Etapa de Diseño del modelo Virtual

Se diseñó y desarrolló el modelo virtual de ambas manos, incluyendo una sección del antebrazo mediante la plataforma de modelado y animación Blender v.2.93. El modelo virtual es capaz de mover cada uno de sus dedos de manera independiente con el objetivo de generar animaciones que emulen de forma natural la apertura y cierre de la mano.

Para simular estos movimientos, fue necesario proveer de un esqueleto al modelo virtual, en él se consideraron las articulaciones interfalángicas distales, interfalángicas proximales, metacarpofalángicas, carpometacarpiana del pulgar y radiocarpiana. mismas que se ilustran en la figura 3.9, otorgando de esta manera tres grados de libertad a cada dedo y un grado de libertad en la muñeca.

El esqueleto articulado se vinculó al modelo virtual para proveerle de movimiento, de manera posterior se diseñaron las animaciones para la apertura y cierre de las manos. Para realizar estas animaciones se utilizaron animaciones no lineales o *NLA-Tracks* [102], esta herramienta disponible en Blender permite generar y ejecutar por

separado varias secuencias de movimiento destinadas al mismo objeto, de otra forma las animaciones se reproducirían sobre una misma línea de tiempo, imposibilitando asignar cada animación a una acción específica. El procedimiento realizado en Blender se detalla en el Apéndice A.

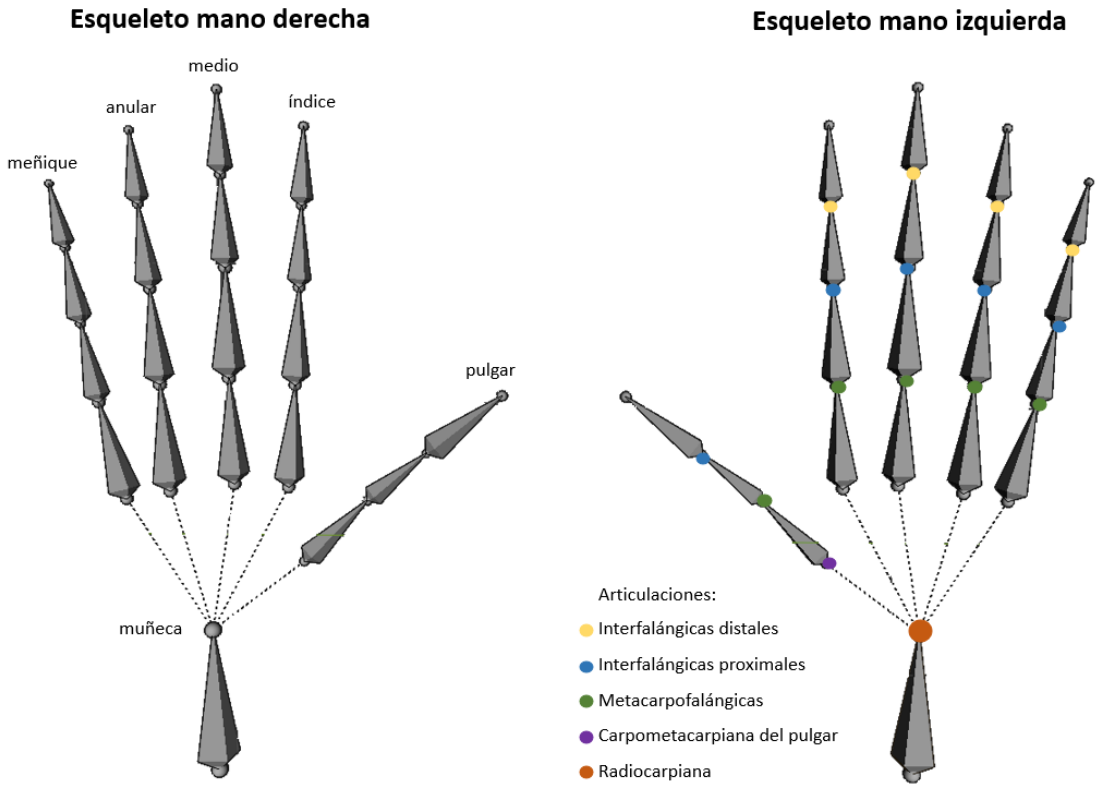

Fig. 3.9 En el esqueleto de mano derecha se muestra la parte de la mano a la que corresponde cada conjunto de huesos, mientras que en el esqueleto de la mano izquierda se representan las articulaciones consideradas en el modelo virtual.

Se diseñaron un total de cuatro animaciones correspondientes a: el estado de reposo, cierre y apertura de mano izquierda, cierre y apertura de mano derecha y cierre y apertura de ambas manos. Cada una de las animaciones tiene una duración de 4 s (100 fotogramas a una tasa de 25 FPS).

3.2.1 Importando el modelo virtual a Unity

El modelo virtual diseñado, así como sus animaciones, fue exportado hacia la plataforma de desarrollo Unity, a través de la cual se realizará la comunicación con Matlab. La exportación se realiza en formato FBX, utilizado principalmente para intercambiar animaciones de personajes entre aplicaciones manteniendo un resultado final que luce igual al diseñado en Blender.

Una vez en Unity, el modelo virtual se encontrará como un objeto y las animaciones como fragmentos de video, para comenzar a manipular el modelo es necesario asignarle un avatar y animación genéricos desde la herramienta de inspector, esto permite la asignación de las animaciones importadas al modelo virtual. Realizado este procedimiento se puede observar el modelo virtual tal y como se encontraba en Blender antes de su exportación.

Se utilizó la herramienta “controlador de animador” (ver Apéndice A) disponible en Unity [103], para ejecutar las animaciones en el modelo virtual, esta herramienta permite programar animaciones en secuencia lineal o condicionadas a algún evento. Para fines del presente trabajo de tesis se utilizaron animaciones condicionadas a los comandos de control obtenidos del modelo neuronal y que son recibidos desde Matlab, este proceso se detallará en la sección 3.3.

En la herramienta animador se utilizan diagramas a bloques que se componen de estados de animación. Estos son bloques de construcción básicos que a los que se le asigna una secuencia de animación individual, es decir, cada bloque contendrá una animación no lineal previamente exportada desde Blender. Esta animación será reproducida mientras la simulación se encuentre en ese estado. Las transiciones entre estados ocurren al activarse o desactivarse un evento durante la simulación. Dichas transiciones, permiten cambiar entre los estados que podría tomar la animación y pueden ser configuradas para que solo ocurran cuando una condición sea cierta. Solo

puede existir una transición activa en cualquier momento, sin embargo, esta puede ser configurada para interrumpirse por otra transición.

El diagrama a bloques que compone el controlador de animador debe contener al menos dos estados definidos, estos son el desencadenante (elemento que activará las transiciones) y la animación que se realizará por default. Estos estados serán ejecutados de acuerdo con las transiciones asignadas.

Como se muestra en la figura 3.10, en este trabajo se definen los comandos de control como desencadenantes de las animaciones que realizará el modelo virtual, la animación por default se define por el estado de reposo y se le asigna la animación correspondiente. Adicional a los dos bloques principales se agregan tres estados más que corresponden al cierre y apertura de mano derecha, cierre y apertura de mano izquierda y cierre y apertura de ambas manos, asignando la animación no lineal correspondiente a cada uno de estos movimientos, y cada transición (mostrados en forma de flechas que salen o regresan al estado de reposo) se define por el inicio y fin de cada secuencia de animación. De esta forma el modelo virtual permanecerá en reposo hasta recibir un comando de control, momento en que generará una transición hacia el estado de animación correspondiente, una vez concluida la animación asignada el modelo virtual volverá a permanecer en reposo hasta recibir un nuevo comando de control.

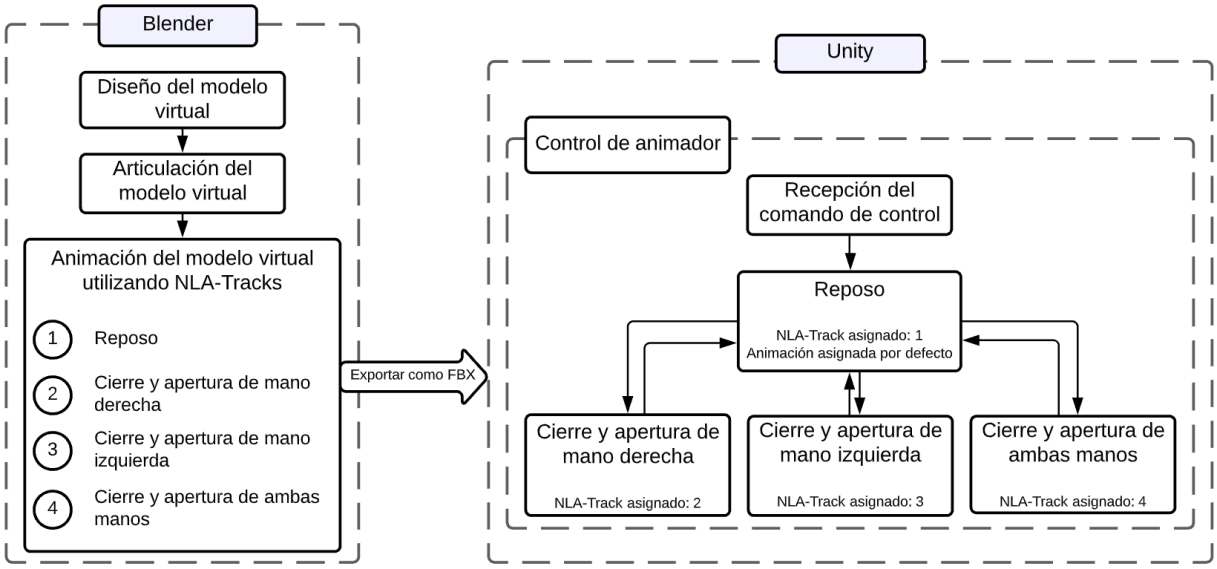

Fig. 3.10 Exportación del modelo virtual desde la plataforma Blender hacia Unity. Del lado izquierdo se muestran los elementos que deben ser exportados hacia Unity, mientras que en el lado derecho de indica el orden en el que se asignaron las animaciones no lineales en los bloques de estados que se forman en el control de animador.

La comunicación entre el diagrama diseñado en el controlador de animador y el modelo virtual se definirá a través de condiciones descritas en un *script* programado en Lenguaje **C**#, este deberá crearse directamente desde Unity y asignarse tanto al animador como al modelo virtual. Los scripts cuentan con dos funciones definidas que son la función *start* y *update*.

En la función *update* se coloca el código que se actualiza una vez por *frame*, dentro de esta función se colocan los objetos que deben ser manipulados durante el tiempo como movimientos, *triggers* o respuestas a *inputs* del usuario. Por su parte, la función *start* será llamada por Unity antes de que la función *update* sea llamada por primera vez, en ella se realizan las inicializaciones.

3.3 Comunicación Matlab-Unity y manipulación del modelo virtual

El uso en Unity de modelos y animaciones diseñadas en Blender es ampliamente utilizado entre diseñadores de personajes virtuales o videojuegos, debido a que permite crear ambientes virtuales con mayor facilidad. En la presente tesis, la integración entre las plataformas busca crear comunicación directa entre el modelo virtual y Matlab, posibilitando la transmisión de los comandos de control entregados por los resultados de la red neuronal hacia Unity.

Para lograr la transmisión de datos se utilizó el protocolo TCP/IP en el que se definió a Matlab como cliente y a Unity como servidor. La comunicación en Unity se estableció utilizando la librería *System.Net.Sockets* disponible en el lenguaje de programación **C**#, mismo que se encuentra integrado en la plataforma y a través del cual se definieron las animaciones que se desencadenarían con cada comando de control. En el caso de Matlab se utilizó la librería Instrument Control Toolbox v.4.3.

El *script* a través del cual se realizó la conexión entre Matlab y Unity fue el mismo en el que se establecieron las condiciones de control para el controlador de animador, de esta forma el *script* se encuentra asociado a ambos elementos. La lectura de la dirección IP y la inicialización del protocolo TCP/IP se definen en la función *start*, por su parte la lectura de los datos transmitidos, así como la identificación de estos y la asignación a una condición para el controlador de animador se realizan dentro de la función *update*.

Se asignaron las animaciones correspondientes a los movimientos clasificados por el modelo de red neuronal artificial, de esta forma, al clasificar un movimiento de mano derecha se ejecutará la animación de cierre y apertura de mano derecha, de igual manera al clasificar un movimiento izquierdo la animación ejecutada será la de cierre y apertura de la mano izquierda.

En el caso de que el modelo neuronal no logre clasificar un movimiento, el modelo virtual ejecutará la animación de apertura y cierre de ambas manos. Al igual que en los casos anteriores las animaciones del modelo virtual permiten identificar visualmente el resultado de clasificación obtenido por la red neuronal artificial, por lo que al asignar la animación mencionada es más fácil diferenciar esta clase del estado de reposo.

En el lado izquierdo de la figura 3.11 se describe la relación entre la salida de la red neuronal, el comando de control asignado, el estado de animación al que se asigna el comando de control y la animación que se ejecutará. En el lado derecho se ilustra el diagrama a bloques diseñado con la herramienta controlador de animación (mencionado en la sección 3.2) en donde se muestra la manera en la que se han asignado los comandos de control mostrados en la tabla.

Salida de la red neuronal	Comando de control	Estado de animación	Animación
(1 0)	1	Mov. mano derecha	Cierre y apertura de mano derecha
(0 1)	2	Mov. mano izquierda	Cierre y apertura de mano izquierda
No identificado	3	Mov. no identificado	Cierre y apertura de ambas manos

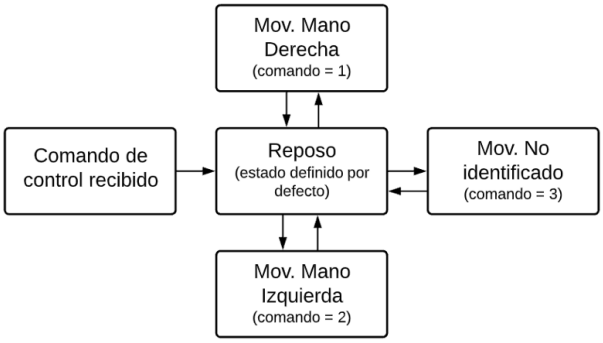

Fig. 3.11 Relación “salida de la red neuronal – comando de control” y su interpretación por el controlador de animaciones en Unity.

De manera adicional, se agregó un contador de los movimientos clasificados por los modelos de red neuronal artificial, estos son “movimiento de mano izquierda”, “movimiento de mano derecha” y “movimiento no identificado”. Este se muestra en forma de tabla durante la ejecución de las animaciones del modelo virtual y se actualiza cada vez que se recibe información.

Capítulo 4. Resultados

4.1 Análisis ERD

En la figura 4.1 se muestra el resultado de la estimación del ERD para el participante A06 durante la ejecución de movimientos imaginarios de la mano derecha e izquierda, este se observa como un decremento porcentual en la potencia mostrada por el electrodo contralateral al movimiento imaginado. De esta forma, durante la imaginación motora de la mano derecha se observa un mayor decremento en C3, mientras que durante la imaginación motora de la mano izquierda el mayor decremento se observa en C4. En ambas gráficas, la línea azul representa el comportamiento de la información del electrodo C3 y en color naranja el del electrodo C4.

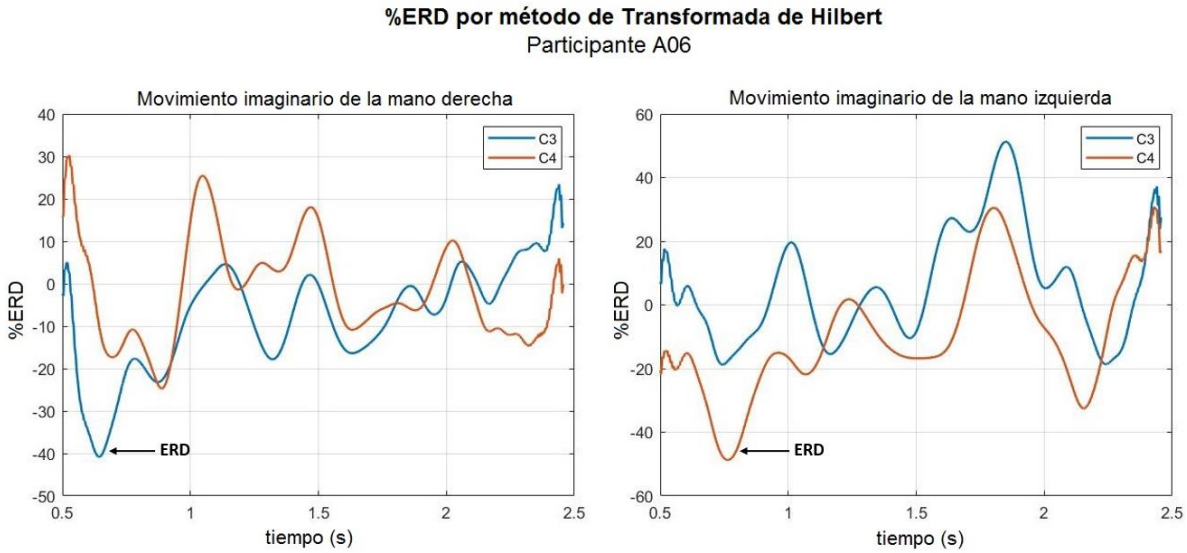

Fig. 4.1 %ERD estimado durante el movimiento imaginario de las manos, utilizando el método de Transformada de Hilbert. Ejemplo mostrado a través del participante A06.

Con el propósito de observar la variación en el decremento del potencial entre C3 y C4, se calculó el decremento de la potencia ocasionada por el ERD de todos los participantes durante la imaginación motora de cada mano.

Estos resultados se muestran en la figura 4.2, en donde se observa que los movimientos imaginarios de la mano derecha presentan un decremento mayor en la potencia que los movimientos ejecutados por la mano izquierda. En el primer caso, el mayor decremento en el electrodo contralateral es cercano al 70%, mientras que el menor se aproxima al 25%, de manera contraria durante la imaginación motora de la mano izquierda el mayor decremento en el electrodo contralateral se acerca al 60%, mientras que el menor permanece cerca del 10%.

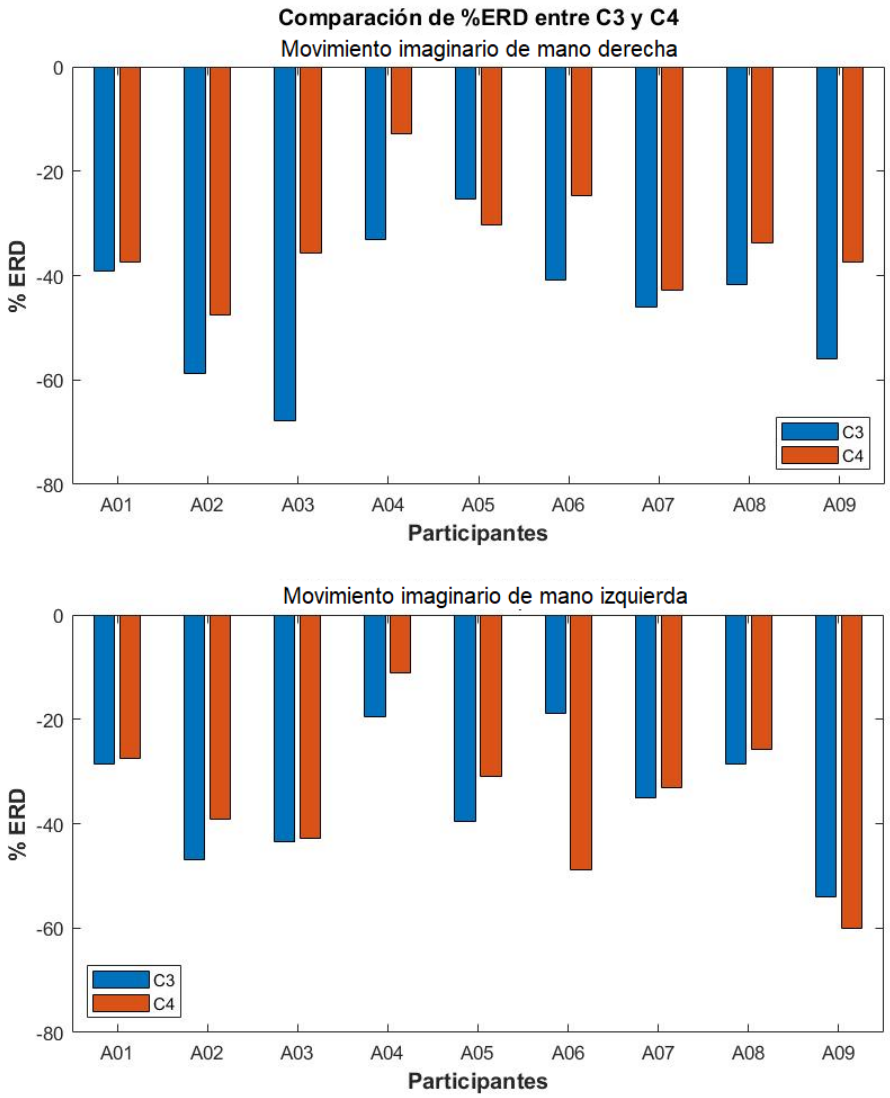

Fig. 4.2 Comparación del ERD entre los electrodos C3 y C4 para cada uno de los participantes. Las barras muestran el decremento en la potencia de la señal.

Con excepción del participante A05, los participantes presentan contralateralidad al ejecutar movimientos imaginarios de la mano derecha, mientras que al ejecutar imágenes motoras de la mano izquierda, solo los participantes A06 y A09 muestran esta característica.

De igual manera, la diferencia entre el decremento de C3 y C4 es mayor durante las tareas de imaginación motora derecha, y se reduce significativamente al compararse en los movimientos imaginarios de la mano izquierda. Un ejemplo de ello se observa en los participantes A01, A03 y A07 quienes presentan decrementos similares en ambos electrodos.

4.2 Resultados de los modelos de clasificación

El entrenamiento de los modelos neuronales se realizó con el conjunto de datos seleccionado para este fin, mismo que representa al 55% del conjunto de datos disponibles. Estos ensayos fueron utilizados para entrenar los modelos de red neuronal con algoritmos de entrenamiento de retropropagación resiliente, retropropagación por secante de un paso y retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos para cada uno de los participantes. Estos fueron diseñados con los parámetros de red y de entrenamiento descritos en la subsección 3.1.4.1.

Durante el entrenamiento se utilizó la técnica de validación cruzada k-fold (con k=10) para obtener una mejor representación estadística de los resultados de clasificación. Al evaluar el desempeño de los modelos se tomaron en cuenta métricas que, aparte de mostrar el porcentaje de datos clasificados correctamente (calculado a partir de la exactitud), permitieran responder ¿cuántos ensayos identificados como clase 1 o clase 2 pertenecen realmente a ellas? y ¿qué tan probable es que estos resultados fueran causados por el azar? Para atender el primer cuestionamiento se calcularon la precisión, sensibilidad, especificidad y el F1-score, en donde la precisión

también permite evaluar la reproducibilidad que presentan los modelos. Por su parte, la segunda pregunta puede ser respondida calculando el coeficiente de Kappa.

El modelo de red neuronal que, en conjunto con el vector de características propuesto en este trabajo, presentó mejores resultados fue el que empleó el algoritmo de entrenamiento de retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos, por lo que los procesos de validación se realizaron únicamente para los modelos que utilizan este algoritmo de entrenamiento.

4.2.1 Validación de los modelos de red neuronal artificial

La capacidad de generalización se evaluó al utilizar el conjunto de datos de prueba como entradas de los modelos neuronales entrenados. Los ensayos utilizados en esta etapa constituyen el 45% del total de datos disponibles y no habían sido utilizados previamente, por lo que todos representan instancias nuevas para los modelos neuronales.

Las instancias nuevas fueron presentadas como entradas a los diez grupos generados en el proceso de validación cruzada y se calcularon las métricas de desempeño de exactitud, precisión, sensibilidad, especificidad, f1-score y coeficiente de Kappa. De igual manera, se calculó el promedio y la desviación estándar del total de grupos para cada uno de los participantes, obteniendo los resultados se muestran en la figura 4.3, representados por un círculo y una línea de intervalo respectivamente.

El desempeño de clasificación obtenido por los participantes se muestra en la Tabla 4.1, en donde se observa que el promedio de datos clasificados correctamente por todos los participantes es de 93%. Este es un desempeño bastante competitivo con respecto a trabajos reportados en la literatura en donde se utiliza la misma base de datos para clasificar movimientos imaginarios de mano derecha e izquierda, en los que se obtuvieron porcentajes de clasificación promedio de entre 70% y 82.5% al utilizar modelos de clasificación lineal (LDA y SVM con kernel lineal) en conjunto con

características calculadas en el dominio temporal, espacial y frecuencial [95, 104] y de hasta un 91.3% al utilizar un clasificador de distancia mínima de Mahalonobis con características espaciales [105].

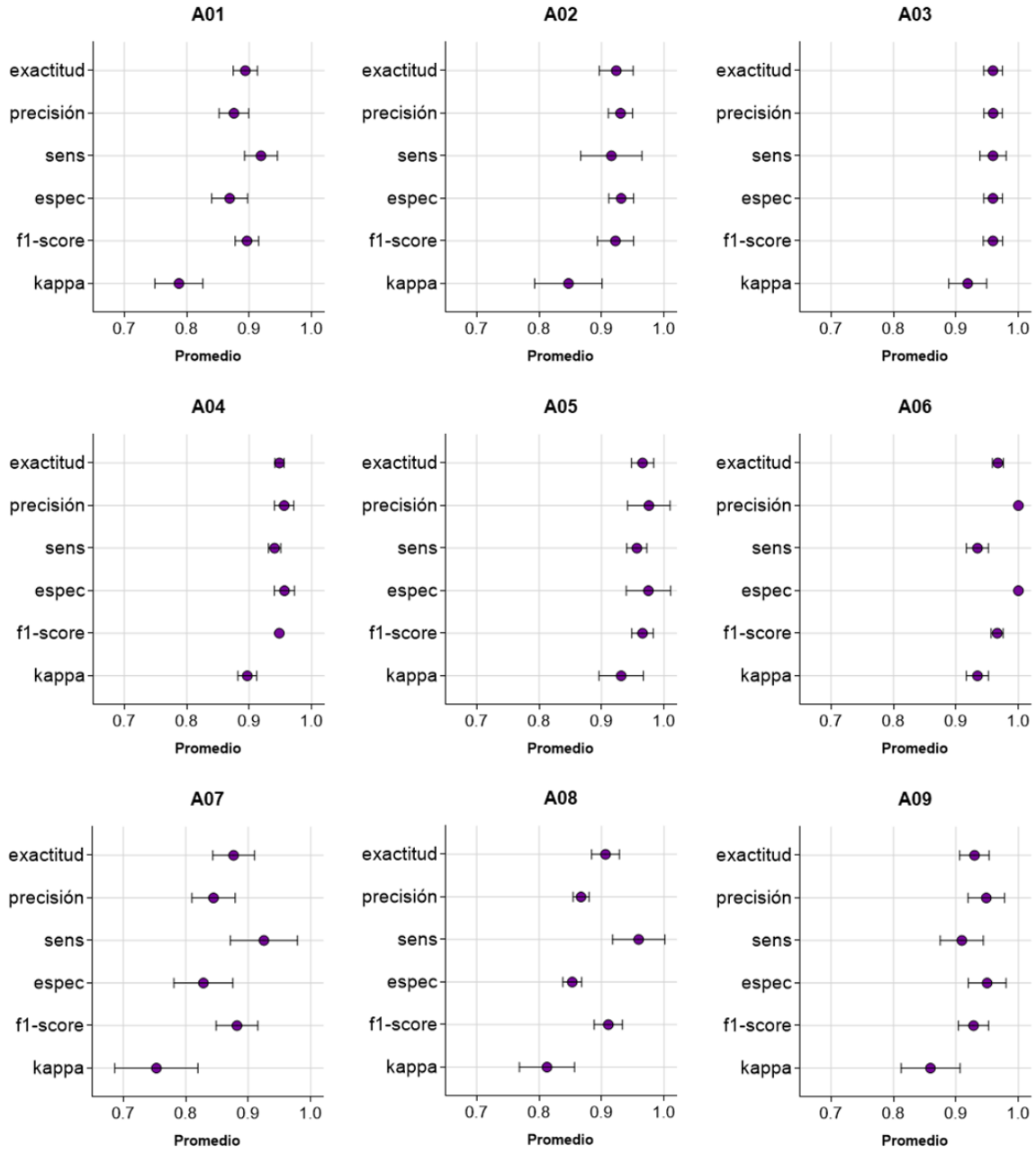

Fig. 4.3 Resultados de la validación de los modelos de redes neuronales, entrenados con algoritmo de retropropagación de gradiente descendiente con tasa de aprendizaje y momentum adaptativos

Tabla 4.1 Porcentaje de movimientos imaginarios clasificados correctamente de cada uno de los participantes.

Participante	Movimientos clasificados correctamente (%)
A01	89.38
A02	92.34
A03	95.94
A04	94.84
A05	96.56
A06	96.72
A07	87.66
A08	90.63
A09	92.97
Promedio	**93.00±3.28**

Correlacionando los resultados obtenidos en la validación de los modelos con los decrementos ocasionados por el ERD de cada uno de los participantes (ver figura 4.2), es posible apreciar que los participantes que presentaron menor desempeño también mostraron la menor diferencia en el ERD entre los electrodos C3 y C4 tanto en los movimientos de mano derecha como de mano izquierda.

En este trabajo se reporta además un desempeño óptimo en el coeficiente de Kappa, indicando que el resultado de clasificación de los modelos neuronales no se debe al azar, si no, al correcto desempeño de los modelos neuronales diseñados.

Las métricas de sensibilidad y especificidad permiten identificar el tipo de error que comete el modelo durante la clasificación de los ensayos, recordando que un desempeño bajo en la sensibilidad indica mayor número de falsos negativos (error de tipo II), mientras que un desempeño bajo en la especificidad representa un mayor número de falsos positivos (error de tipo I). En los resultados mostrados en la figura 4.3, la mayoría de los participantes presentan un desempeño mejor de sensibilidad que de especificidad, indicando que los modelos tienden a confundir los movimientos imaginarios de mano izquierda con movimientos imaginarios de mano derecha.

De igual manera, se analiza el porcentaje de movimientos imaginarios de mano derecha y mano izquierda no clasificados (C1 y C2, respectivamente). Estos porcentajes se muestran en la figura 4.4, en donde el porcentaje de movimientos no clasificados más alto corresponde al participante con tan solo un 4.38%.

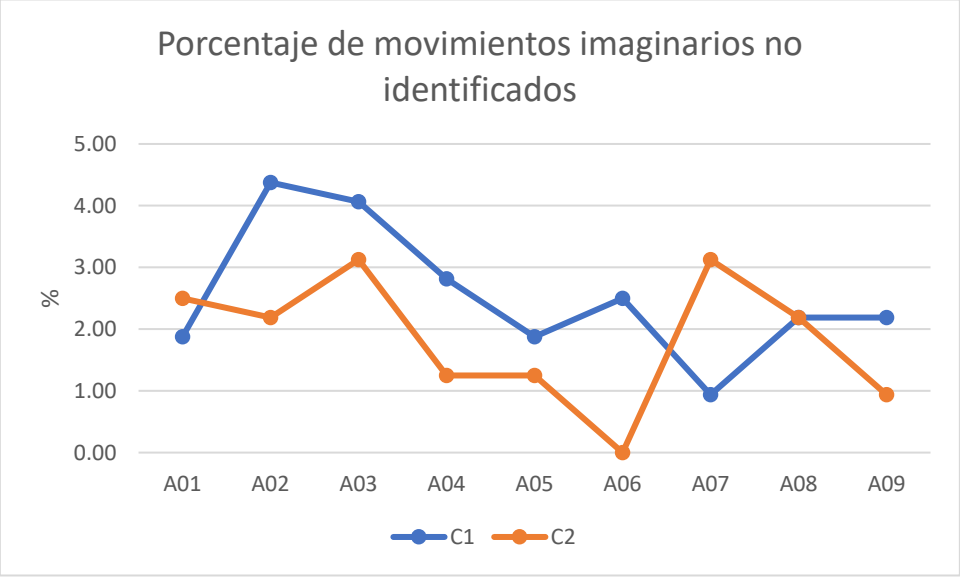

Fig. 4.4 Porcentaje de movimientos no identificados por sujeto, la línea azul (C1) representa los movimientos imaginarios de mano derecha, la línea naranja (C2) los de mano izquierda.

Lo anterior en conjunto con el resultado obtenido en las métricas de exactitud y precisión, son un indicativo del buen desempeño que presentan los modelos de red neuronal artificial diseñados para cada uno de los participantes.

4.2.2 Validación con registros de EEG contaminados con ruido blanco

Al contaminar las señales de EEG con porcentajes de ruido blanco equivalentes al nivel de potencia absoluto de la señal, se espera que entre mayor sea el porcentaje la señal se comprometa al grado de representar una señal muy distinta a la original. Para probar este argumento se calculó el coeficiente de correlación entre la señal de EEG original y las señales contaminadas, obteniendo los coeficientes mostrados en la Tabla 4.2. De esta forma es posible observar como la similitud entre ambas señales se deteriora al aumentar el porcentaje de ruido blanco.

Tabla 4.2 Coeficiente de correlación calculado entre las señales de EEG originales y las señales contaminadas con distintos niveles de ruido blanco.

	Coeficiente de correlación			
	10%	**25%**	**40%**	**50%**
A01	0.9533	0.8942	0.8444	0.8147
A02	0.9532	0.8945	0.8450	0.8159
A03	0.9534	0.8941	0.8459	0.8154
A04	0.9533	0.8941	0.8448	0.8160
A05	0.9534	0.8955	0.8448	0.8168
A06	0.9535	0.8942	0.8459	0.8170
A07	0.9531	0.8934	0.8450	0.8156
A08	0.9533	0.8944	0.8442	0.8169
A09	0.9535	0.8948	0.8451	0.8149

Los vectores de características extraídos de las señales contaminadas se utilizaron para probar los modelos de red neuronal diseñados. Las gráficas mostradas en la figura 4.5 indican la manera en la que estos modelos clasificaron los datos contaminados con ruido blanco, en comparación con los resultados obtenidos con los datos sin contaminar. En estas gráficas el eje vertical indica el valor obtenido luego de promediar el resultado de cada modelo entrenado con la técnica de validación k-fold.

Mientras que en el eje horizontal se agrupan los resultados obtenidos por nivel de ruido para cada una de las métricas de desempeño calculadas.

La distribución de estas gráficas se realizó con el objetivo de distinguir con mayor facilidad la manera en la que adicionar ruido blanco a la señal afecta el desempeño de los modelos neuronales. La barra naranja indica el resultado obtenido por el conjunto de datos de pruebas sin contaminar, este es el mismo que el reportado en la figura 4.3, seguido de ella se muestran las barras correspondientes a los datos de prueba con 10%, 25%, 40% y 50% de ruido blanco añadido.

Las barras de color amarillo representan a los datos de prueba contaminados con 10% de ruido blanco, al compararlas con los datos de prueba sin contaminar se aprecia que la diferencia entre los resultados de clasificación es mínima. Este resultado es esperado pues, como se demostró al calcular el coeficiente de correlación (Tabla 4.2) existe bastante similitud entre ambas señales.

En la figura 4.5 se puede observar que conforme incrementa el nivel de ruido blanco añadido, el desempeño durante la clasificación de estos disminuye. Los cambios son más notables al alcanzar el 40% de ruido blanco añadido (barra de color rosa), a partir de este todos los participantes sufren un decremento significativo en las métricas de exactitud, precisión y coeficiente de Kappa. Esto indica un menor número de ensayos clasificados correctamente, así como una mayor posibilidad de que los datos clasificados solo hayan sido identificados a causa del azar.

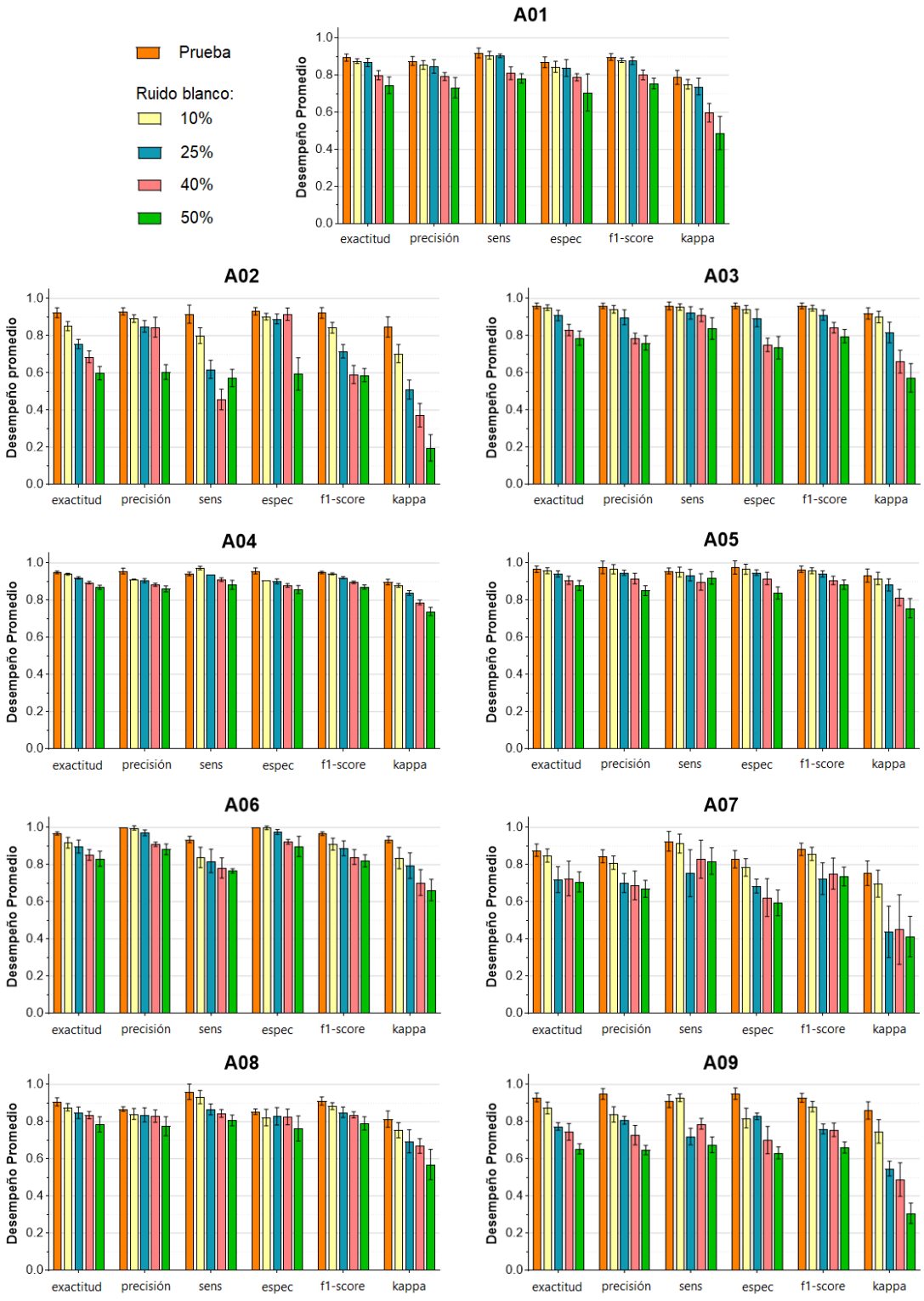

Fig. 4.5 Desempeño obtenido con los ensayos de prueba sin contaminar (primera barra de cada grupo), y los ensayos de prueba contaminados con 10%, 25%, 40% y 50% de ruido blanco (barras de la 2 a la 5 de cada grupo).

4.3 Diseño del modelo virtual Blender - Unity

En este trabajose presenta, el diseño en Blender v2.93 el modelo virtual de ambas manos como se muestra en la figura 4.6, estas manos están articuladas de tal forma que permiten el movimiento de cada dedo de manera independiente para simular los movimientos de apertura y cierre.

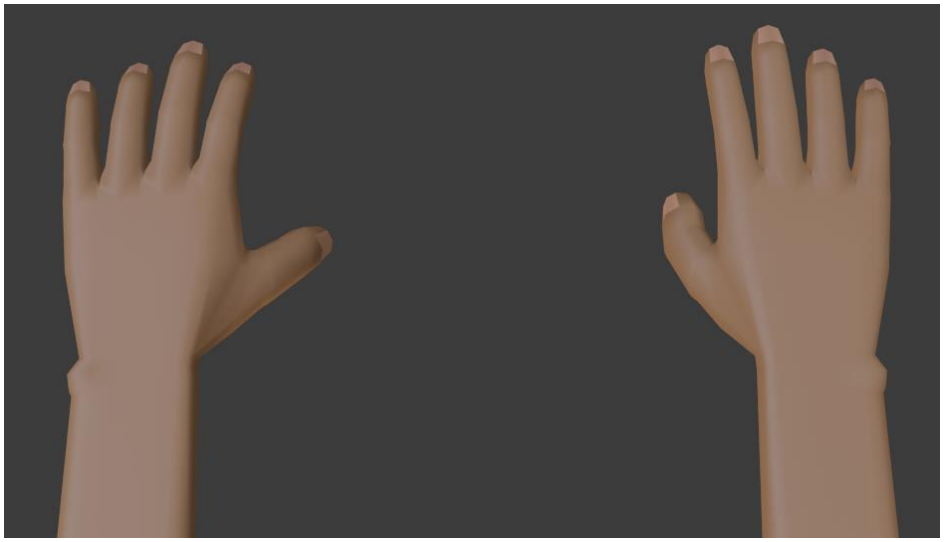

Fig. 4.6 Modelo virtual diseñado en la plataforma de modelado y animación Blender v.2.93

Para simular estos movimientos se utilizaron animaciones no lineales [102], lo que permitió crear el conjunto de cuatro animaciones o estados asignados al modelo virtual, que son capaces de ejecutarse de manera independiente. La importancia de esta independencia recae en que las animaciones tendrán que ser desencadenadas a partir de comandos de control provenientes de una fuente externa al software de simulación, por lo que si estas animaciones se encontraran en la misma línea temporal sería imposible ejecutarlas por separado.

En la figura 4.7 se muestra el modelo virtual ejecutando cada una de las animaciones diseñadas, estas consisten en el cierre y apertura de la mano izquierda

o derecha mientras la mano contraria permanece en reposo, cierre y apertura de ambas manos y reposo de ambas manos.

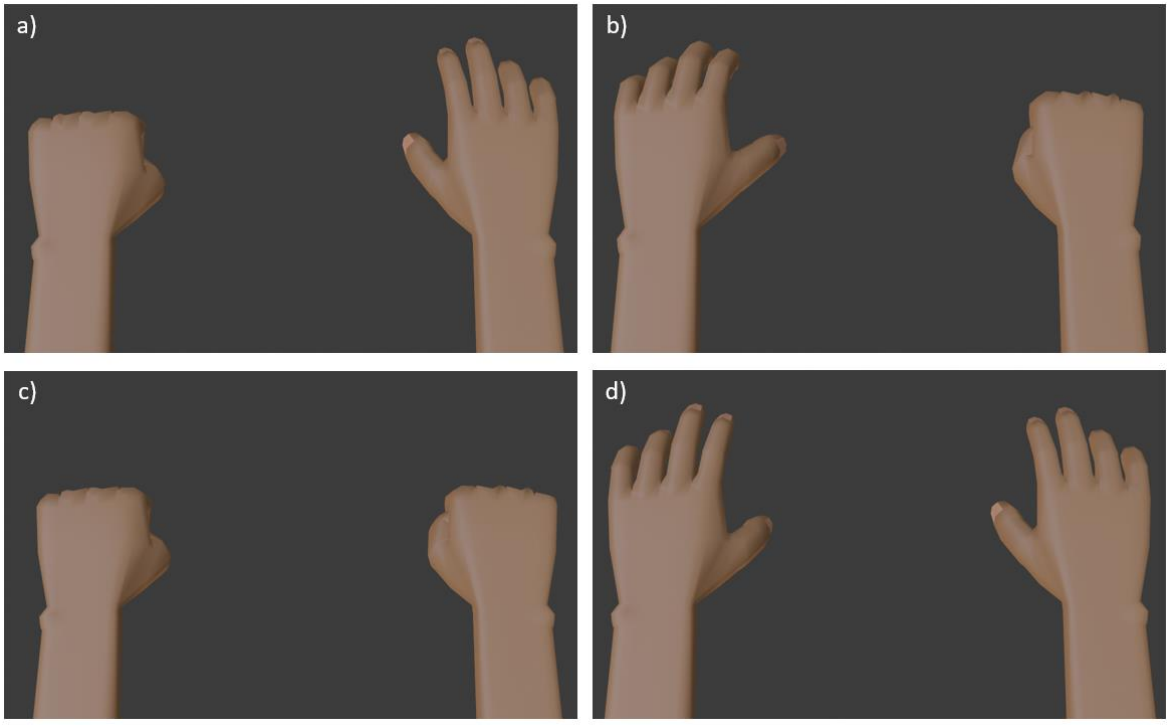

Fig. 4.7 Animaciones diseñadas (a) Cierre y apertura de mano izquierda, (b) Cierre y apertura de mano derecha, (c) Cierre y apertura de ambas manos, (d) Manos en reposo.

El modelo virtual y las animaciones diseñadas fueron exportadas hacia la plataforma de Unity utilizando el formato FBX, que permite que el modelo no pierda las características diseñadas en Blender.

El modelo virtual se mostró en perspectiva de primera persona y se le asignaron las animaciones diseñadas en Blender. Estas también fueron asociadas con su estado correspondiente en el controlador de animación, traduciendo el diagrama a bloques de la figura 3.11 en un ejecutor de movimientos del modelo virtual.

Por último, se añadió una caja de texto en donde se mostrará el número de veces que se ejecute cada uno de los movimientos, permitiendo observar el resultado de la clasificación dentro de la plataforma Unity. Estos contadores se muestran en la pantalla durante toda la simulación. El resultado final del modelo virtual implementado puede observarse en la figura 4.8.

Fig. 4.8 Modelo virtual de ambas manos importado en Unity.

4.4 Comunicación Unity – Matlab

La comunicación entre Unity y Matlab se estableció utilizando el protocolo TCP/IP, a través del cual fue posible transmitir datos entre ambas plataformas estableciendo a Matlab como cliente y a Unity como servidor.

Los resultados obtenidos de la validación de los modelos de redes neuronales fueron codificados a comandos de control que pudieran transmitirse desde Matlab en

forma de cadena. Estos son recibidos por Unity a través de un *script* desarrollado en lenguaje **C**# en el que se asocian el modelo virtual, los estados del diagrama a bloques y las animaciones

El envío de datos por parte del cliente (Matlab) se muestra en el diagrama de flujo de la figura 4.9, en donde se observa que cada uno de los ensayos será evaluado para definir el tipo de movimiento que se ha clasificado. La escritura por medio de TCP/IP se activa y desactiva cada vez que el cliente realiza la evaluación de un ensayo, y es hasta que se concluye la evaluación de todos los ensayos cuando se termina la comunicación por parte del cliente.

Como se ilustró en la figura 3.11, los comandos escritos a partir de cada evaluación pueden ser ‘1’ en caso de que se clasifique un movimiento derecho, ‘2’ al clasificar un movimiento izquierdo y ‘3’ en caso de que no lograr clasificar un ensayo. Para concluir la comunicación con el servidor se escribe un comando ‘0’, indicando que se ha concluido la evaluación de todos los ensayos.

Por su parte del servidor (Unity) se prepara para recibir los comandos de control provenientes del cliente, por lo que permanece activo verificando constantemente si ha recibido algún dato, de ser así este comando es identificado para desencadenar un movimiento en el modelo virtual. Este desarrollo se muestra en el diagrama de flujo de la figura 4.10.

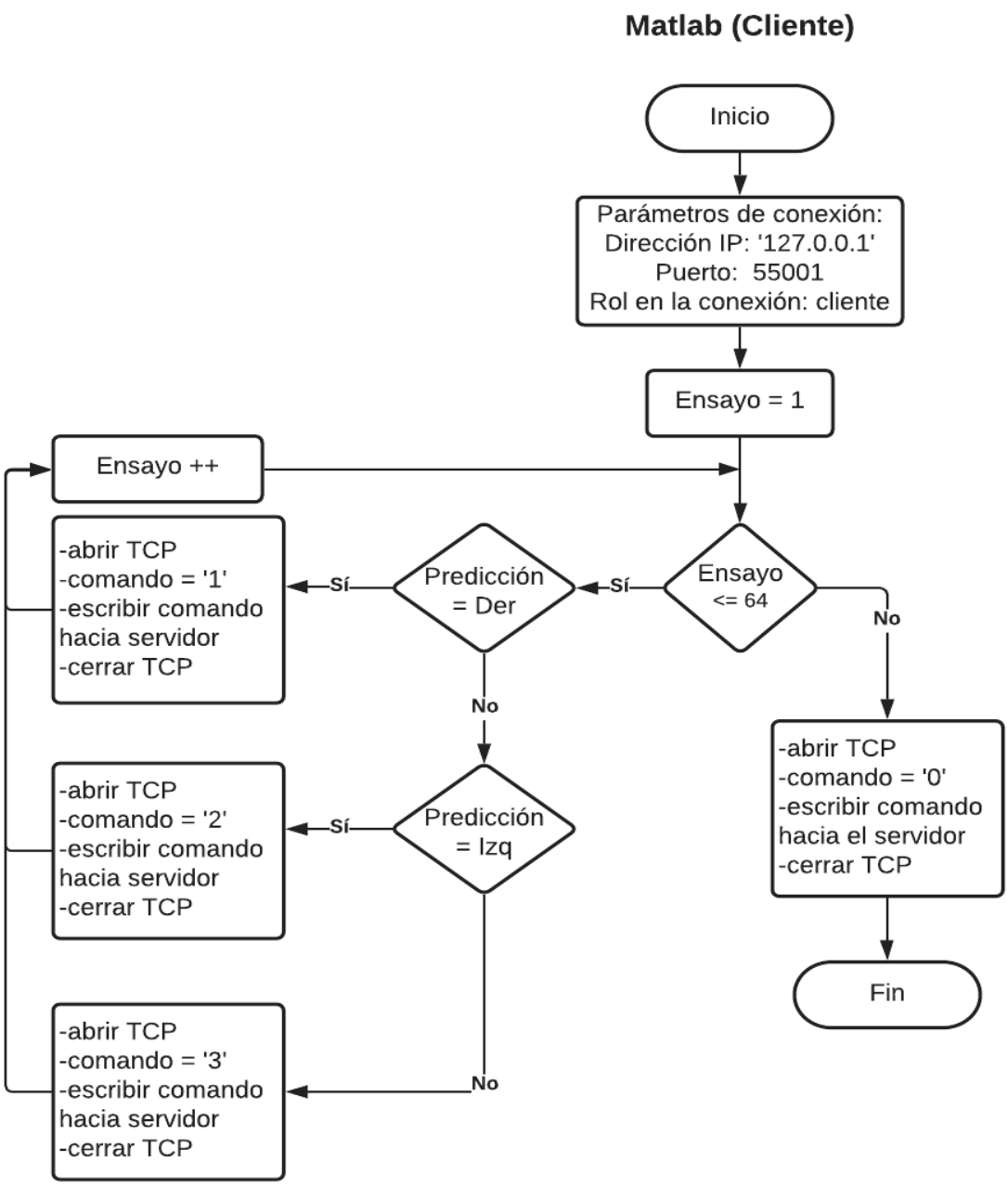

Fig. 4.9 Diagrama de flujo del envío de comandos a partir del protocolo TCP-IP desde el cliente.

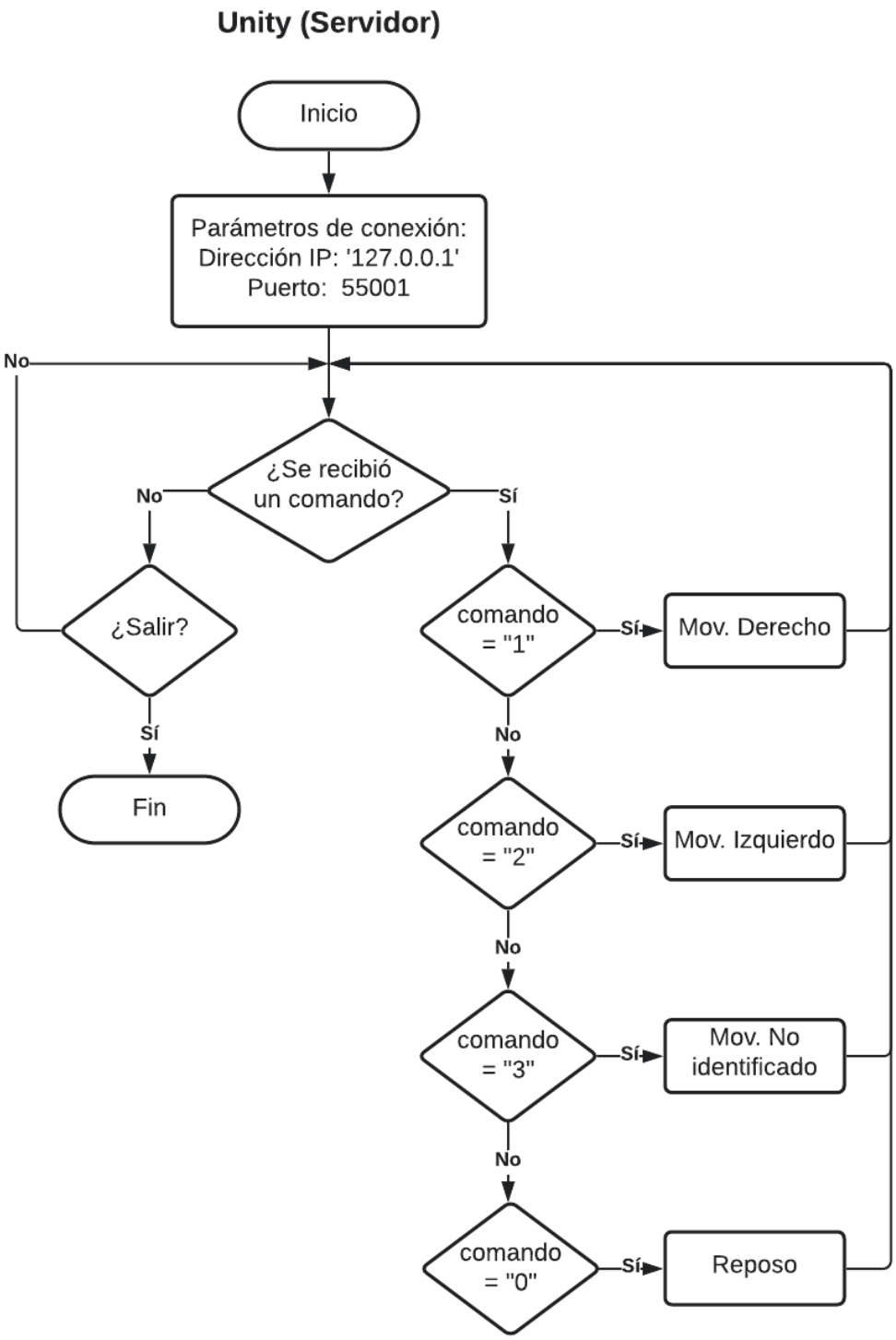

Fig. 4.10 Diagrama de flujo de la recepción de comandos a partir del protocolo TCP-IP desde el servidor.

El tiempo total de transmisión se calcula desde que comienza la clasificación del movimiento imaginario, hasta que concluye la recepción de información en Unity a través del protocolo TCP/IP, sin tomar en cuenta el tiempo asignado a la animación del modelo virtual. En promedio se obtuvo un tiempo de ejecución de 0.015 s, calculado a partir del tiempo obtenido para cada uno de los 64 ensayos disponibles en el conjunto de datos de validación.

Para que la comunicación se realice de manera correcta el servidor debe encontrarse activo antes de que el cliente comience a enviar comandos, es por esto que primero debe inicializarse la simulación del modelo virtual y seguido de ello la evaluación de los resultados obtenidos por las redes neuronales. De igual manera, la dirección IP y el puerto a utilizar deben ser los mismos para el cliente y el servidor, de lo contrario no existirá comunicación entre ellos.

Como resultado se observa el modelo virtual de ambas manos que ejecuta las animaciones diseñadas y las contabiliza mostrando estos resultados durante toda la simulación. La figura 4.11 muestra el modelo virtual durante la simulación realizada en Unity, como puede observarse, el modelo mantiene las características diseñadas en Blender, se muestra en perspectiva de primera persona y ejecuta las animaciones al mismo tiempo que se contabilizan los movimientos clasificados.

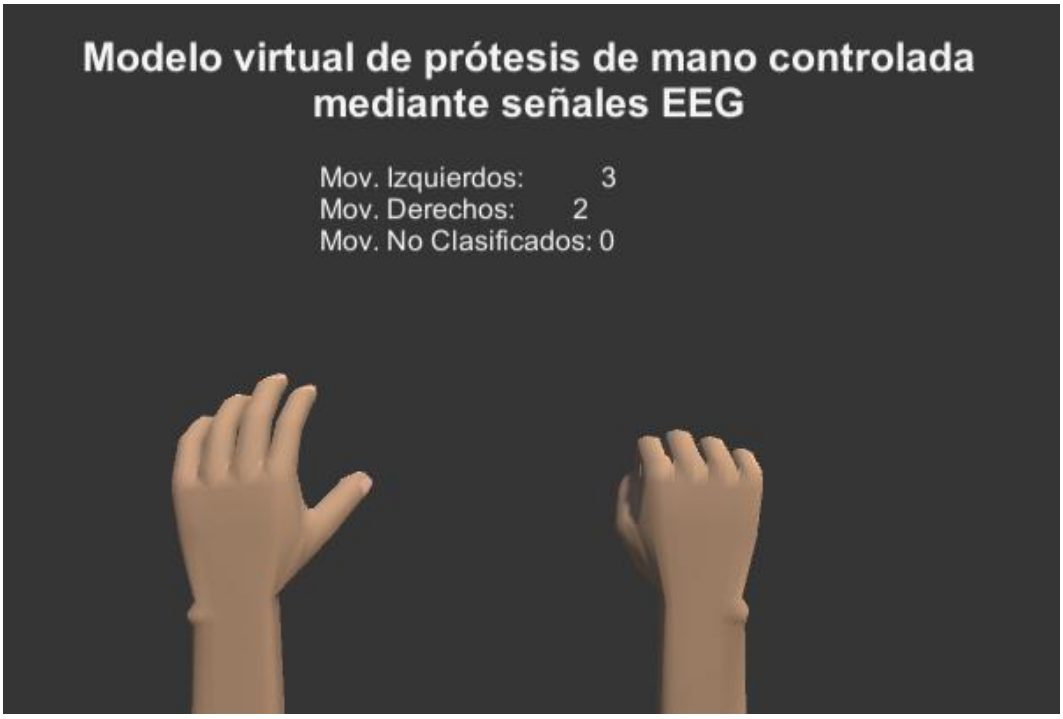

Fig. 4.11 Modelo virtual durante la simulación. En la parte superior de la pantalla se observa el contador de movimientos.

Para corroborar que el envío de datos se realiza de manera correcta, una vez concluida la comunicación por TCP/IP se compararon los movimientos contabilizados durante la simulación y los resultados de la matriz de confusión obtenida en Matlab®.

Para ello se tomaron los datos de prueba del participante A06 y se utilizaron para validar el modelo de red y la comunicación TCP/IP. Los resultados de este procedimiento pueden observarse en la figura 4.12, del lado derecho se presentan la matriz de confusión de la validación del modelo de red neuronal y del lado izquierdo se muestran el número de movimientos realizados por el modelo virtual durante la simulación.

Como puede observarse, el total de movimientos de mano derecha corresponde a la suma de los datos clasificados como “clase 1” (movimiento imaginario derecho), los movimientos de mano izquierda corresponden al total de datos clasificados como

“clase 2” (movimiento imaginario izquierdo) y los movimientos no clasificados corresponden al total de ensayos que la red no logró clasificar.

Matriz de confusión entregada por Matlab

Predicción \ Objetivo	C1	C2
C1	VP 31	FP 2
C2	FN 0	VN 29
NC	1	1

Movimientos realizados por el modelo virtual

Movimiento	Repeticiones
Mano Derecha	33
Mano Izquierda	29
Ambas manos	2

Fig. 4.12 Validación del envío de comandos de control entre Matlab® y Unity.

Capítulo 5. Conclusiones y trabajo futuro.

El objetivo de este trabajo consistió en implementar un sistema BCI que utilice como aplicación un modelo virtual de ambas manos capaces de realizar movimientos de apertura y cierre de manera independiente, y cuyo movimiento es controlado a partir de la clasificación de movimientos imaginarios de las manos derecha e izquierda. Para conseguirlo, se propuso y desarrolló una metodología en la que se abordaron etapas referentes al procesamiento y clasificación de señales EEG correspondientes a movimientos imaginarios, el diseño y articulación de un modelo virtual y la comunicación entre ambos. A través de estas etapas se alcanzaron de manera satisfactoria los objetivos propuestos, obteniendo los resultados mostrados y discutidos en el capítulo 4, a partir de los cuales se presentan las siguientes conclusiones.

Un paso esencial durante la primera etapa fue la selección de la base de datos, para ello se estimaron los potenciales ERD de cada participante utilizando el método de Transformada de Hilbert, a través del cual se logró observar que este potencial se encontraba presente en cada participante que conformaba la base de datos, haciendo posible su selección. La presencia de este potencial funcionó como un indicativo de que las tareas de imaginación motora fueron asimiladas de manera correcta por los participantes, logrando la activación de la corteza motora primaria.

Durante el procesamiento se extrajeron características temporales a las señales de EEG, a través de las cuales se describió su comportamiento con respecto a sus medidas de tendencia central, variabilidad, deformación y energía. De manera posterior se realizó la clasificación de los movimientos imaginarios a través del método de redes neuronales artificiales, utilizando un algoritmo de entrenamiento de retropropagación de gradiente descendente con tasa de aprendizaje y momentum adaptativos. Siguiendo esta metodología se obtuvieron porcentajes de clasificación superiores a los reportados en la literatura en trabajos en donde se utiliza la misma

base de datos y trabajos donde se manipulan modelos virtuales de aspecto realista. Estos resultados permiten concluir que el método de clasificación implementado en conjunto con las características seleccionadas conforma una metodología viable en el procesamiento de las señales de imaginación motora correspondiente a movimientos de las manos.

Con respecto al desarrollo de la segunda etapa de este trabajo, se diseñó en Blender el modelo virtual de manos derecha e izquierda, obteniendo un modelo anatómicamente realista y con la capacidad de realizar animaciones de apertura y cierre de mano de manera independiente. La asignación de estas animaciones al modelo virtual se realizó dentro de la plataforma de Unity, en donde además de preparar visualmente la simulación, se asignaron los eventos a desencadenar una vez recibidos los comandos de control.

La comunicación completa y precisa de los comandos de control se realizó a través del protocolo TCP/IP, a partir del cual se logró comunicar los resultados de la clasificación para desencadenar los movimientos asignados al modelo virtual. La clasificación y transferencia de información se realizó en tan solo 0.014 s, por lo que cada vez que se clasificaba un movimiento el modelo virtual se movía de acuerdo con el comando de control desencadenado por esta, por lo que se concluye que la comunicación a través del protocolo TCP/IP es una forma viable de transmitir información entre los lenguajes de programación utilizados durante las diferentes etapas de este trabajo.

Trabajo futuro

El buen desempeño obtenido tanto por el modelo clasificador como en la transmisión de los comandos de control permite plantear que la metodología implementada en este trabajo puede ser considerada para proponerse en sistemas BCI enfocados a proporcionar retroalimentación a través de modelos virtuales, ya sea con el objetivo de beneficiar a la neuroplasticidad con fines de rehabilitación motora, o

de implementar estrategias que permitan el control de dispositivos externos o prótesis. En cualquier caso, sería necesario considerar el ajuste de los parámetros de entrenamiento y la arquitectura de las redes neuronales artificiales para que logren un desempeño satisfactorio.

De igual forma, al utilizar modelos virtuales se cuenta con la ventaja de que pueden modificarse para representar cualquier otra extremidad, ya sean inferiores, superiores o incluso los dedos de la mano. Por lo que podría plantearse como trabajo futuro el desarrollo de una base de datos propia en donde el paradigma experimental involucre la imaginación de movimientos indicados a través del modelo virtual. De esta manera, los modelos virtuales serían utilizados como guía visual para los participantes y para mostrar la retroalimentación del desempeño durante la ejecución motora.

Referencias

[1] B. Graimann, B. Allison, and G. Pfurtscheller, “Brain–Computer Interfaces: A Gentle Introduction BT - Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction,” B. Graimann, G. Pfurtscheller, and B. Allison, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–27.

[2] P. Wierzgała, D. Zapała, G. M. Wojcik, and J. Masiak, “Most Popular Signal Processing Methods in Motor-Imagery BCI: A Review and Meta-Analysis,” *Front. Neuroinform.*, vol. 12, p. 78, Nov. 2018, doi: 10.3389/fninf.2018.00078.

[3] L. F. Nicolas-Alonso and J. Gomez-Gil, “Brain computer interfaces, a review,” *Sensors (Basel).*, vol. 12, no. 2, pp. 1211–1279, 2012, doi: 10.3390/s120201211.

[4] T. Karácsony, J. Hansen, H. Iversen, and S. Puthusserypady, *Brain Computer Interface for Neuro-rehabilitation With Deep Learning Classification and Virtual Reality Feedback*. 2019.

[5] C. Neuper, R. Scherer, M. Reiner, and G. Pfurtscheller, “Imagery of motor actions: differential effects of kinesthetic and visual-motor mode of imagery in single-trial EEG.,” *Brain Res. Cogn. Brain Res.*, vol. 25, no. 3, pp. 668–677, Dec. 2005, doi: 10.1016/j.cogbrainres.2005.08.014.

[6] K. Li, P. Boyd, Y. Zhou, Z. Ju, and H. Liu, “Electrotactile Feedback in a Virtual Hand Rehabilitation Platform: Evaluation and Implementation,” *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1556–1565, 2019, doi: 10.1109/TASE.2018.2882465.

[7] G. Rizzolatti, “The mirror neuron system and its function in humans.,” *Anat. Embryol. (Berl).*, vol. 210, no. 5–6, pp. 419–421, Dec. 2005, doi: 10.1007/s00429-005-0039-z.

[8] G. Rizzolatti and L. Craighero, “The mirror-neuron system.,” *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004, doi: 10.1146/annurev.neuro.27.070203.144230.

[9] M. Tani *et al.*, “Action observation facilitates motor cortical activity in patients with

stroke and hemiplegia.," *Neurosci. Res.*, vol. 133, pp. 7–14, Aug. 2018, doi: 10.1016/j.neures.2017.10.002.

[10] D. Wen *et al.*, "Combining brain–computer interface and virtual reality for rehabilitation in neurological diseases: A narrative review," *Ann. Phys. Rehabil. Med.*, vol. 64, no. 1, p. 101404, Jan. 2021, doi: 10.1016/J.REHAB.2020.03.015.

[11] D. Camargo-Vargas, M. Callejas-Cuervo, and S. Mazzoleni, "Brain-Computer Interfaces Systems for Upper and Lower Limb Rehabilitation: A Systematic Review," *Sensors 2021, Vol. 21, Page 4312*, vol. 21, no. 13, p. 4312, Jun. 2021, doi: 10.3390/S21134312.

[12] N. Padfield, J. Zabalza, H. Zhao, V. Masero, and J. Ren, "EEG-Based Brain-Computer Interfaces Using Motor-Imagery: Techniques and Challenges," *Sensors (Basel).*, vol. 19, no. 6, Mar. 2019, doi: 10.3390/S19061423.

[13] K. Värbu, N. Muhammad, and Y. Muhammad, "Past, Present, and Future of EEG-Based BCI Applications," *Sensors (Basel).*, vol. 22, no. 9, p. 3331, Apr. 2022, doi: 10.3390/S22093331.

[14] M. Ahn and S. C. Jun, "Performance variation in motor imagery brain–computer interface: A brief review," *J. Neurosci. Methods*, vol. 243, pp. 103–110, 2015, doi: https://doi.org/10.1016/j.jneumeth.2015.01.033.

[15] C. Jeunet, E. Jahanpour, and F. Lotte, "Why standard brain-computer interface (BCI) training protocols should be changed: an experimental study.," *J. Neural Eng.*, vol. 13, no. 3, p. 36024, Jun. 2016, doi: 10.1088/1741-2560/13/3/036024.

[16] M. F. Bear, B. W. Connors, and M. A. Paradiso, *Neuroscience: Exploring the brain*, 4th ed. Wolters Kluwer, 2015.

[17] J. E. Hall and M. E. Hall, *Guyton and Hall Textbook of medical physiology*, 14th ed. Philadelphia: Elsevier Saunders, 2006.

[18] A. B. Benevides, A. S. da Paz Floriano, M. Sarcinelli-Filho, and T. F. Bastos-Filho, "Review of the human brain and EEG signals," in *Introduction to non-invasive EEG-based brain-computer interfaces for assistive technologies*, First., T. F. Bastos-Filho, Ed. New York: Taylor & Francis Group, 2021, pp. 1–50.

[19] G. Pfurtscheller and C. Neuper, “Movement and ERD/ERS BT - The Bereitschaftspotential: Movement-Related Cortical Potentials,” M. Jahanshahi and M. Hallett, Eds. Boston, MA: Springer US, 2003, pp. 191–206.

[20] R. J. Davidson, D. C. Jackson, and C. L. Larson, “Human electroencephalography,” in *Handbook of psychophysiology*, 2nd ed., J. T. Cacioppo, L. G. Tassinary, and G. G. Berntson, Eds. New York: Cambridge University Press, 2000, pp. 27–52.

[21] S. Siuly, Y. Li, and Y. Zhang, *EEG Signal Analysis and Classification. Techniques and Applications*, 1st ed. Switzerland: Springer, Cham, 2016.

[22] T. J. Sejnowski, P. S. Churchland, and J. A. Movshon, “Putting big data to good use in neuroscience,” *Nat. Neurosci.*, vol. 17, no. 11, pp. 1440–1441, 2014, doi: 10.1038/nn.3839.

[23] R. COOPER, J. W. OSSELTON, and J. C. SHAW, “5 - Operational Techniques,” R. COOPER, J. W. OSSELTON, and J. C. B. T.-E. E. G. T. (Second E. SHAW, Eds. Butterworth-Heinemann, 1974, pp. 79–107.

[24] A. S. Malik and H. U. Amin, “Chapter 1 - Designing an EEG Experiment,” in *Designing EEG Experiments for Studying the Brain*, A. S. Malik and H. U. Amin, Eds. Academic Press, 2017, pp. 1–30.

[25] W. Klimesch *et al.*, “Theta band power change in normal and dyslexic children,” *Clin. Neurophysiol.*, vol. 112, pp. 1174–1185, Aug. 2001, doi: 10.1016/S1388-2457(01)00545-4.

[26] D. Pizzagalli *et al.*, “Anterior cingulate activity as a predictor of degree of treatment response in major depression: evidence from brain electrical tomography analysis.,” *Am. J. Psychiatry*, vol. 158, no. 3, pp. 405–415, Mar. 2001, doi: 10.1176/appi.ajp.158.3.405.

[27] A. K. Engel and P. Fries, “Beta-band oscillations--signalling the status quo?,” *Curr. Opin. Neurobiol.*, vol. 20, no. 2, pp. 156–165, Apr. 2010, doi: 10.1016/j.conb.2010.02.015.

[28] J. M. Estébanez, R. J. Campos, J. R. Millán, and U. P. de C. C. de R. en E. Biomèdica, *EEG-based Analysis for the Design of Adaptive Brain Interfaces*.

2003.

[29] R. COOPER, J. W. OSSELTON, and J. C. SHAW, “6 - Visual Analysis of the EEG,” R. COOPER, J. W. OSSELTON, and J. C. B. T.-E. E. G. T. (Second E. SHAW, Eds. Butterworth-Heinemann, 1974, pp. 108–132.

[30] G. Pfurtscheller, “Graphical display and statistical evaluation of event-related desynchronization (ERD),” *Electroencephalogr. Clin. Neurophysiol.*, vol. 43, no. 5, pp. 757–760, 1977, doi: https://doi.org/10.1016/0013-4694(77)90092-X.

[31] G. Pfurtscheller, “Event-related synchronization (ERS): an electrophysiological correlate of cortical areas at rest.,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 83, no. 1, pp. 62–69, Jul. 1992, doi: 10.1016/0013-4694(92)90133-3.

[32] G. Pfurtscheller and A. Berghold, “Patterns of cortical activation during planning of voluntary movement,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 72, no. 3, pp. 250–258, 1989, doi: https://doi.org/10.1016/0013-4694(89)90250-2.

[33] J. Kaiser, R. Ulrich, and W. Lutzenberger, “Dynamics of sensorimotor cortex activation to spatial sounds precueing ipsi- versus contralateral manual responses.,” *Brain Res. Cogn. Brain Res.*, vol. 17, no. 3, pp. 573–583, Oct. 2003, doi: 10.1016/s0926-6410(03)00171-x.

[34] W. Gaetz and D. Cheyne, “Localization of sensorimotor cortical rhythms induced by tactile stimulation using spatially filtered MEG.,” *Neuroimage*, vol. 30, no. 3, pp. 899–908, Apr. 2006, doi: 10.1016/j.neuroimage.2005.10.009.

[35] Y. Jeon, C. S. Nam, Y.-J. Kim, and M. C. Whang, “Event-related (De)synchronization (ERD/ERS) during motor imagery tasks: Implications for brain–computer interfaces,” *Int. J. Ind. Ergon.*, vol. 41, no. 5, pp. 428–436, 2011, doi: https://doi.org/10.1016/j.ergon.2011.03.005.

[36] B. Graimann and G. Pfurtscheller, “Quantification and visualization of event-related changes in oscillatory brain activity in the time–frequency domain,” in *Event-Related Dynamics of Brain Oscillations*, vol. 159, C. Neuper and W. B. T.-P. in B. R. Klimesch, Eds. Elsevier, 2006, pp. 79–97.

[37] G. Pfurtscheller and F. H. Lopes da Silva, “Event-related EEG/MEG synchronization and desynchronization: basic principles,” *Clin. Neurophysiol.*,

vol. 110, no. 11, pp. 1842–1857, 1999, doi: https://doi.org/10.1016/S1388-2457(99)00141-8.

[38] J. Kalcher and G. Pfurtscheller, “Discrimination between phase-locked and non-phase-locked event-related EEG activity,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 94, no. 5, pp. 381–384, 1995, doi: https://doi.org/10.1016/0013-4694(95)00040-6.

[39] S. Kiebel, J. Kilner, and K. Friston, “Hierarchical models for EEG and MEG,” in *Statistical Parametric Mapping: The Analysis of Functional Brain Images*, 2007, pp. 211–220.

[40] P. Clochon, J.-M. Fontbonne, N. Lebrun, and P. Etévenon, “A new method for quantifying EEG event-related desynchronization: amplitude evvelope analysis,” *Electroencephalogr. Clin. Neurophysiol.*, vol. 98, no. 2, pp. 126–129, 1996, doi: https://doi.org/10.1016/0013-4694(95)00192-1.

[41] N. Bagh and M. R. Reddy, “Hilbert transform-based event-related patterns for motor imagery brain computer interface,” *Biomed. Signal Process. Control*, vol. 62, p. 102020, 2020, doi: https://doi.org/10.1016/j.bspc.2020.102020.

[42] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, “Brain–computer interfaces for communication and control,” *Clin. Neurophysiol.*, vol. 113, no. 6, pp. 767–791, 2002, doi: https://doi.org/10.1016/S1388-2457(02)00057-3.

[43] V. Martínez-Cagigal, E. Santamaría-Vázquez, J. Gomez-Pilar, and R. Hornero, “A brain–computer interface web browser for multiple sclerosis patients,” *Neurological Disorders and Imaging Physics, Volume 2*. IOP Publishing, pp. 12–31, 2019, doi: 10.1088/978-0-7503-1762-7ch12.

[44] S. Mueller, W. Cardoso, T. Freire, and M. Sarcinelli-Filho, “Brain-computer Interface Based on Visual Evoked Potentials to Command Autonomous Robotic Wheelchair,” *J. Med. Biol. Eng*, vol. 30, Jan. 2010, doi: 10.5405/jmbe.765.

[45] M. A. Pastor, J. Artieda, J. Arbizu, M. Valencia, and J. C. Masdeu, “Human Cerebral Activation during Steady-State Visual-Evoked Responses,” *J. Neurosci.*, vol. 23, no. 37, pp. 11621 LP – 11627, Dec. 2003, doi:

10.1523/JNEUROSCI.23-37-11621.2003.

[46] J. Kim, B. Lee, H. S. Lee, K. H. Shin, M. J. Kim, and E. Son, “Differences in Brain Waves of Normal Persons and Stroke Patients during Action Observation and Motor Imagery.,” *J. Phys. Ther. Sci.*, vol. 26, no. 2, pp. 215–218, Feb. 2014, doi: 10.1589/jpts.26.215.

[47] O. Blanke and T. Metzinger, “Full-body illusions and minimal phenomenal selfhood.,” *Trends Cogn. Sci.*, vol. 13, no. 1, pp. 7–13, Jan. 2009, doi: 10.1016/j.tics.2008.10.003.

[48] D. Bansal and R. Mahajan, “Chapter 2 - EEG-Based Brain-Computer Interfacing (BCI),” in *EEG-Based Brain-Computer Interfaces*, D. Bansal and R. Mahajan, Eds. Academic Press, 2019, pp. 21–71.

[49] J. R. Wolpaw *et al.*, “Brain-computer interface technology: a review of the first international meeting,” *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 2, pp. 164–173, 2000, doi: 10.1109/TRE.2000.847807.

[50] M. M. Fouad, K. M. Amin, N. El-Bendary, and A. E. Hassanien, “Brain Computer Interface: A Review BT - Brain-Computer Interfaces: Current Trends and Applications,” A. E. Hassanien and A. T. Azar, Eds. Cham: Springer International Publishing, 2015, pp. 3–30.

[51] J. Hong, X. Qin, J. Li, J. Niu, and W. Wang, “Signal processing algorithms for motor imagery brain-computer interface: State of the art,” *J. Intell. Fuzzy Syst.*, vol. 35, pp. 6405–6419, 2018, doi: 10.3233/JIFS-181309.

[52] D. Jarchi, V. Abolghasemi, and S. Sanei, “Source localization of brain rhythms by empirical mode decomposition and spatial notch filtering,” in *2009 17th European Signal Processing Conference*, 2009, pp. 627–631.

[53] L. Yao, X. Shu, J. Meng, D. Zhang, X. Sheng, and X. Zhu, “Enhanced motor imagery based brain-computer interface via unilateral wrist vibrotactile stimulation,” in *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2013, pp. 29–32, doi: 10.1109/NER.2013.6695863.

[54] P. L. Nunez, R. B. Silberstein, P. J. Cadusch, R. S. Wijesinghe, A. F. Westdorp, and R. Srinivasan, “A theoretical and experimental study of high resolution EEG

based on surface Laplacians and cortical imaging," *Electroencephalogr. Clin. Neurophysiol.*, vol. 90, no. 1, pp. 40–57, 1994, doi: https://doi.org/10.1016/0013-4694(94)90112-0.

[55] O. Bertrand, F. Perrin, and J. Pernier, "A theoretical justification of the average reference in topographic evoked potential studies," *Electroencephalogr. Clin. Neurophysiol. Potentials Sect.*, vol. 62, no. 6, pp. 462–464, 1985, doi: https://doi.org/10.1016/0168-5597(85)90058-9.

[56] R. Begg, D. T. H. Lai, and M. Palaniswami, *Computational Intelligence in Biomedical Engineering*. CRC Press, 2007.

[57] A. Abraham, R. Falcón, and R. Bello, *Rough Set Theory: A True Landmark in Data Analysis*, vol. 174. Berlin, Heidelberg: Springer, 2009.

[58] A. Phinyomark, F. Quaine, S. Charbonnier, C. Serviere, F. Tarpin-Bernard, and Y. Laurillau, "EMG feature evaluation for improving myoelectric pattern recognition robustness," *Expert Syst. Appl.*, vol. 40, no. 12, pp. 4832–4840, 2013, doi: https://doi.org/10.1016/j.eswa.2013.02.023.

[59] B. Hjorth, "EEG analysis based on time domain properties," *Electroencephalogr. Clin. Neurophysiol.*, vol. 29, no. 3, pp. 306–310, 1970, doi: https://doi.org/10.1016/0013-4694(70)90143-4.

[60] J. Kevric and A. Subasi, "Comparison of signal decomposition methods in classification of EEG signals for motor-imagery BCI system," *Biomed. Signal Process. Control*, vol. 31, pp. 398–406, 2017, doi: https://doi.org/10.1016/j.bspc.2016.09.007.

[61] S. A. Unde and R. Shriram, "Coherence Analysis of EEG Signal Using Power Spectral Density," in *2014 Fourth International Conference on Communication Systems and Network Technologies*, 2014, pp. 871–874, doi: 10.1109/CSNT.2014.181.

[62] R. Romo-Vázquez, H. Velez Perez, R. Ranta, V. Louis-Dorr, D. Maquin, and L. Maillard, "Blind source separation, wavelet denoising and discriminant analysis for EEG artefacts and noise cancelling," *Biomed. Signal Process. Control*, vol. 7, pp. 389–400, Jul. 2012, doi: 10.1016/j.bspc.2011.06.005.

[63] G. Pfurtscheller and C. Neuper, “Motor imagery and direct brain-computer communication,” *Proc. IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001, doi: 10.1109/5.939829.

[64] E. Mohamed, “Enhancing EEG Signals in Brain Computer Interface Using Wavelet Transform,” *Int. J. Inf. Electron. Eng.*, vol. 4, Jan. 2014, doi: 10.7763/IJIEE.2014.V4.440.

[65] A. B. Jerbic, P. Horki, S. Sovilj, V. Isgum, and M. Cifrek, “Hilbert-Huang Time-Frequency Analysis of Motor Imagery EEG Data for Brain-Computer Interfaces BT - 6th European Conference of the International Federation for Medical and Biological Engineering,” 2015, pp. 62–65.

[66] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg, “Designing optimal spatial filters for single-trial EEG classification in a movement task,” *Clin. Neurophysiol.*, vol. 110, no. 5, pp. 787–798, 1999, doi: https://doi.org/10.1016/S1388-2457(98)00038-8.

[67] E. Alpaydin, *Introduction to Machine Learning*, Second. MIT Press, 2014.

[68] P. Cichoz, *Data mining algorithms: Explained using R*. United Kingdom: John Wiley & Sons, 2014.

[69] S. O. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Ontario Canada: Pearson, 2009.

[70] M. Kubat, “Artificial Neural Networks,” in *An Introduction to Machine Learning*, M. Kubat, Ed. Cham: Springer International Publishing, 2015, pp. 91–111.

[71] M. Hagan, M. H. Bale, and O. De Jesus, *Neural Network Design*, 2nd ed. 2014.

[72] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. New York, NY: Springer New York, NY, 2006.

[73] R. Battiti, “First- and Second-Order Methods for Learning: Between Steepest Descent and Newton’s Method,” *Neural Comput.*, vol. 4, no. 2, pp. 141–166, 1992, doi: 10.1162/neco.1992.4.2.141.

[74] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: the RPROP algorithm,” in *IEEE International Conference on Neural*

Networks, 1993, pp. 586–591 vol.1, doi: 10.1109/ICNN.1993.298623.

[75] A. Subasi, *Practical Guide for Biomedical Signals Analysis Using Machine Learning Techniques*. United Kingdom: Academic Press, 2019.

[76] G. Zeng, “On the confusion matrix in credit scoring and its analytical properties,” *Commun. Stat. - Theory Methods*, vol. 49, no. 9, pp. 2080–2093, May 2020, doi: 10.1080/03610926.2019.1568485.

[77] M. Kuhn and K. Johnson, “Measuring Performance in Classification Models,” in *Applied Predictive Modeling*, M. Kuhn and K. Johnson, Eds. New York, NY: Springer New York, 2013, pp. 247–273.

[78] J. Carletta, “Assessing Agreement on Classification Tasks: The Kappa Statistic,” *Comput. Linguist.*, vol. 22, no. 2, pp. 249–254, 1996, [Online]. Available: https://aclanthology.org/J96-2004.

[79] D. Bansal and R. Mahajan, “Chapter 1 - Introduction,” D. Bansal and R. B. T.-E.-B. B.-C. I. Mahajan, Eds. Academic Press, 2019, pp. 1–19.

[80] F. Pichiorri *et al.*, “Brain-computer interface boosts motor imagery practice during stroke recovery.,” *Ann. Neurol.*, vol. 77, no. 5, pp. 851–865, May 2015, doi: 10.1002/ana.24390.

[81] S. Vogt, F. Di Rienzo, C. Collet, A. Collins, and A. Guillot, “Multiple roles of motor imagery during action observation,” *Front. Hum. Neurosci.*, vol. 7, p. 807, Nov. 2013, doi: 10.3389/fnhum.2013.00807.

[82] M. Alimardani, S. Nishio, and H. Ishiguro, “The Importance of Visual Feedback Design in BCIs; from Embodiment to Motor Imagery Learning,” *PLoS One*, vol. 11, no. 9, pp. e0161945-, Sep. 2016, [Online]. Available: https://doi.org/10.1371/journal.pone.0161945.

[83] S. Liang, K.-S. Choi, J. Qin, W.-M. Pang, Q. Wang, and P.-A. Heng, “Improving the discrimination of hand motor imagery via virtual reality based visual guidance,” *Comput. Methods Programs Biomed.*, vol. 132, pp. 63–74, 2016, doi: https://doi.org/10.1016/j.cmpb.2016.04.023.

[84] F. Škola and F. Liarokapis, “Embodied VR environment facilitates motor imagery

brain–computer interface training," *Comput. Graph.*, vol. 75, pp. 59–71, 2018, doi: https://doi.org/10.1016/j.cag.2018.05.024.

[85] F. Škola, S. Tinková, and F. Liarokapis, "Progressive Training for Motor Imagery Brain-Computer Interfaces Using Gamification and Virtual Reality Embodiment.," *Front. Hum. Neurosci.*, vol. 13, p. 329, 2019, doi: 10.3389/fnhum.2019.00329.

[86] A. Korik, R. Sosnik, N. Siddique, and D. Coyle, "Decoding Imagined 3D Arm Movement Trajectories From EEG to Control Two Virtual Arms—A Pilot Study ," *Frontiers in Neurorobotics* , vol. 13. 2019, [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00094.

[87] M. A. S. Chowdhury and D. K. Saha, "Processing of Motor Imagery EEG Signals for Controlling the Opening and the Closing of Artificial Hand," in *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, 2019, pp. 1–5, doi: 10.1109/EICT48899.2019.9068828.

[88] Y. Xu *et al.*, "Shared control of a robotic arm using non-invasive brain–computer interface and computer vision guidance," *Rob. Auton. Syst.*, vol. 115, pp. 121–129, 2019, doi: https://doi.org/10.1016/j.robot.2019.02.014.

[89] J. W. Choi, B. H. Kim, S. Huh, and S. Jo, "Observing Actions Through Immersive Virtual Reality Enhances Motor Imagery Training," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 7, pp. 1614–1622, 2020, doi: 10.1109/TNSRE.2020.2998123.

[90] J. W. Choi, S. Huh, and S. Jo, "Improving performance in motor imagery BCI-based control applications via virtually embodied feedback," *Comput. Biol. Med.*, vol. 127, p. 104079, 2020, doi: https://doi.org/10.1016/j.compbiomed.2020.104079.

[91] D. Achanccaray, S.-I. Izumi, and M. Hayashibe, "Visual-Electrotactile Stimulation Feedback to Improve Immersive Brain-Computer Interface Based on Hand Motor Imagery," *Comput. Intell. Neurosci.*, vol. 2021, p. 8832686, 2021, doi: 10.1155/2021/8832686.

[92] C. Brunner, R. Leeb, G. R. Müller-Putz, A. Schlögl, and G. Pfurtscheller, "BCI Competition 2008 – Graz data set A," 2008.

https://bbci.de/competition/iv/desc_2a.pdf.

[93] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang, “Filter Bank Common Spatial Pattern Algorithm on BCI Competition IV Datasets 2a and 2b ,” *Frontiers in Neuroscience* , vol. 6. 2012, [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00039.

[94] T. Nguyen, I. Hettiarachchi, A. Khatami, L. Gordon-Brown, C. P. Lim, and S. Nahavandi, “Classification of Multi-Class BCI Data by Common Spatial Pattern and Fuzzy System,” *IEEE Access*, vol. 6, pp. 27873–27884, May 2018, doi: 10.1109/ACCESS.2018.2841051.

[95] S. Selim, M. Tantawi, H. Shedeed, and A. Badr, “A Comparative Analysis of Different Feature Extraction Techniques for Motor Imagery Based BCI System BT - Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020),” 2020, pp. 740–749.

[96] J. M. Montero Lorenzo, *Estadística Descriptiva*. Thomson Ediciones Paraninfo, 2007.

[97] S. Monroy Saldívar, *Estadística descriptiva*. México D.F.: Instituto Politécnico Nacional, 2008.

[98] F. O.K. and R. R., “Time-domain exponential energy for epileptic EEG signal classification,” *Neurosci. Lett.*, vol. 694, pp. 1–8, 2019, doi: https://doi.org/10.1016/j.neulet.2018.10.062.

[99] M. K. Kiymik, M. Akin, and A. Subasi, “Automatic recognition of alertness level by using wavelet transform and artificial neural network.,” *J. Neurosci. Methods*, vol. 139, no. 2, pp. 231–240, Oct. 2004, doi: 10.1016/j.jneumeth.2004.04.027.

[100] R. Battiti, “Accelerated backpropagation learning: Two optimization methods,” *Complex Syst*, vol. 3, Jan. 1989.

[101] J. M. Gutiérrez Salgado, “Desarrollo de una red neuronal Wavelet para el procesamiento de señales no estacionarias,” CINVESTAV, México D.F., 2004.

[102] “Blender Documentation. Nonlinear Animations.” https://docs.blender.org/manual/en/latest/editors/nla/introduction.html

(accessed Jan. 13, 2022).

[103] “Unity Documentation. Animator Controller.” https://docs.unity3d.com/es/530/Manual/class-AnimatorController.html (accessed Jan. 13, 2022).

[104] Y. Zhang, C. S. Nam, G. Zhou, J. Jin, X. Wang, and A. Cichocki, “Temporally Constrained Sparse Group Spatial Patterns for Motor Imagery BCI,” *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3322–3332, 2019, doi: 10.1109/TCYB.2018.2841847.

[105] V. Mishuhina and X. Jiang, “Complex common spatial patterns on time-frequency decomposed EEG for brain-computer interface,” *Pattern Recognit.*, vol. 115, p. 107918, Jul. 2021, doi: 10.1016/j.patcog.2021.107918.

Apéndice A. Diseño e integración del modelo virtual

El diseño y animación del modelo virtual se desarrolló en la plataforma de diseño y animación Blender v2.93. Para proveer de movimiento a las manos virtuales se diseñó un esqueleto al que a través de articulaciones se le otorgaron dos grados de libertad al dedo pulgar y tres grados de libertad a los dedos índice, medio, anular y meñique.

Para manipular los dedos se colocaron esqueletos extras que actúan como botones a través de los cuales se manipuló el movimiento (ver figura A-1a), estos fueron vinculados a la falange proximal de cada uno de los dedos, que funcionan como indicadores del movimiento para las falanges medias y distales (mostradas en verde). Los botones se encuentran separados del modelo de esqueleto, por lo que se utilizaron *drivers* para vincular el movimiento de los dedos a partir de la manipulación de los botones.

Para comenzar la manipulación del modelo es necesario identificar el eje en el que se efectuará el movimiento de los dedos para que este luzca similar a un movimiento natural. El eje seleccionado en este trabajo fue el eje de rotación X, este se observa resaltado en color morado en la figura A-1b, y sobre él se añadieron los drivers correspondientes a partir de la opción "*Add Driver*". En la figura A-1c se especifica la manera en la que se establece el *driver*. En este caso, el objeto que desea manipularse es el esqueleto (*armature*) y desea manipularse a partir de la localización en el eje Z (*Type*) del botón seleccionado (Bone), es decir, que al desplazar el botón por el eje de las Z el esqueleto rotará en el eje X, generando un movimiento que luzca natural.

Este procedimiento se realizó para cada uno de los dedos de ambas manos, de esta manera los botones pueden alejarse del modelo virtual y los esqueletos pueden ser manipulados con mayor control otorgando un funcionamiento realista.

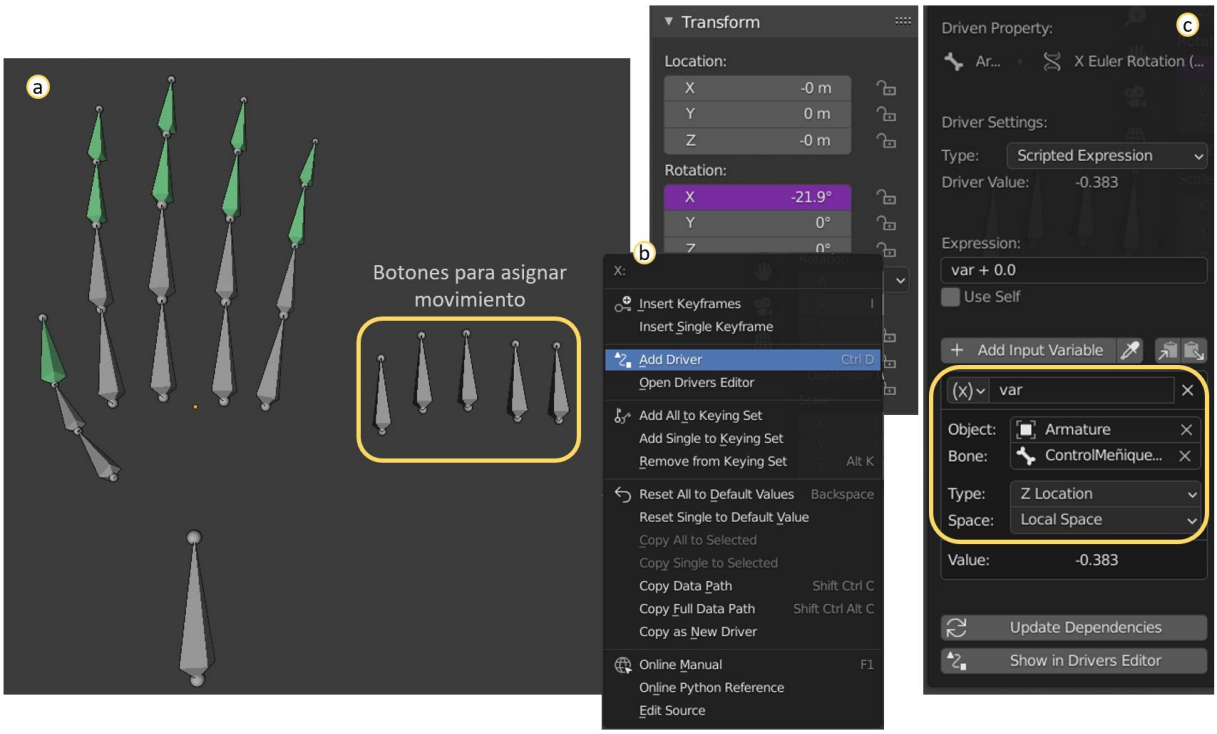

Fig A-1 Asignación de botones para manipular el modelo virtual. En (a) se muestran el aspecto de estos botones en Blender, mientras que en (b) y (c) se presentan los menús a través d ellos cuales se vincula el movimiento.

La programación de las animaciones se realizó a través la línea de tiempo, en donde la localización en el eje Z de los botones fue guardada en el *frame* seleccionado, esto se muestra con una flecha verde en la figura A-2, en donde cada punto en la franja naranja corresponde a una posición guardada. Como los botones están vinculados a la rotación de las articulaciones de los dedos en el eje X, al mover un botón con dirección -Z la articulación del dedo simula el movimiento de flexión, en caso contrario, si la dirección es hacia Z, se simula el movimiento de extensión.

Estos movimientos se programaron de manera independiente utilizando animaciones no lineales (*NLA Tracks*, por sus siglas en inglés), que permiten al modelo efectuar movimientos en líneas de tiempo independientes. Para ello, los movimientos se indicaron en el menú de editor de acción, que se observa dentro de un rectángulo verde en la figura A-2, en donde después de definir los movimientos que componen a

cada animación, fueron guardados cuidando que estas se encontraran dentro del elemento *armature*.

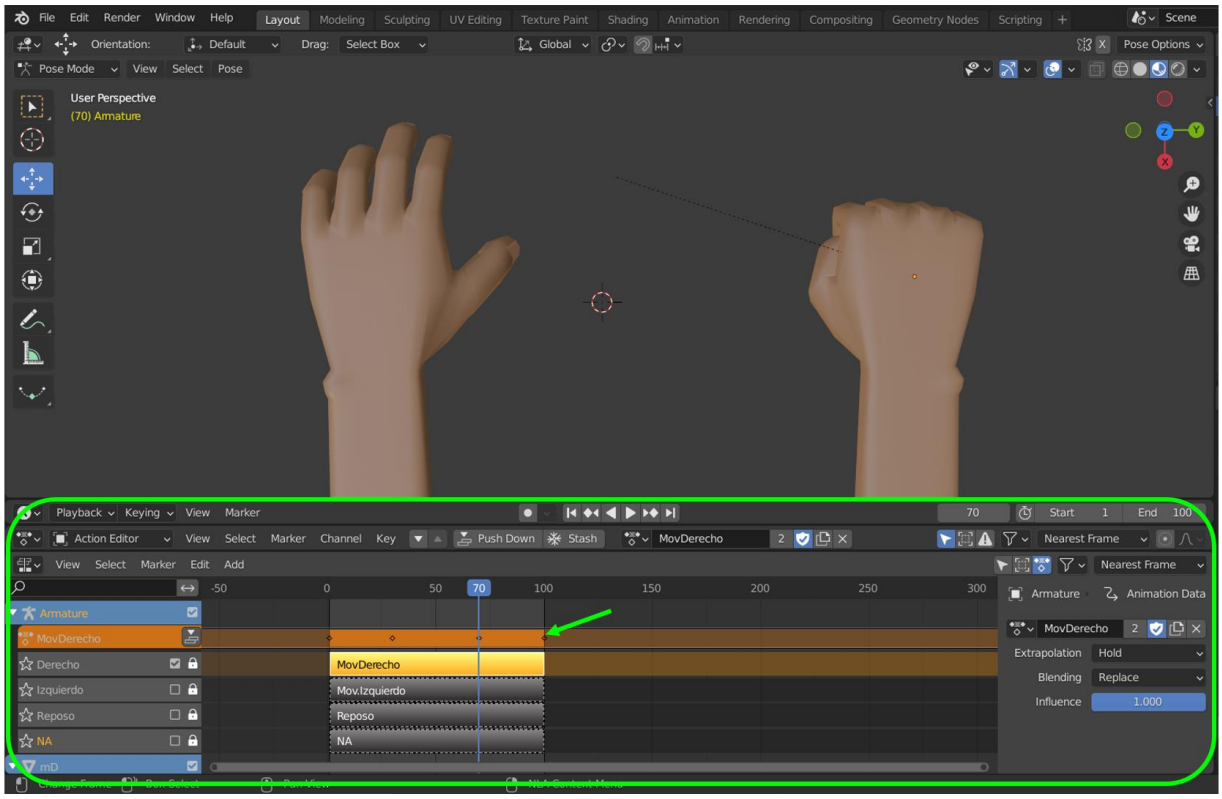

Fig A-2 Menú “Editor de acción” y presentación de las animaciones no lineales en la línea de tiempo.

De esta manera, se diseñaron cuatro movimientos correspondientes al cierre de la mano derecha, cierre de la mano izquierda, cierre de ambas manos y reposo, identificados como derecho, izquierdo, NA (en abreviación a “no aplica”) y reposo, respectivamente.

Ya diseñado el modelo virtual con sus animaciones, se realizó la exportación del modelo hacía la plataforma de desarrollo Unity. La figura A-3 ilustra los pasos a seguir para exportar correctamente el modelo virtual y sus animaciones, siguiendo la numeración mostrada en esta figura, se seleccionó la opción de exportar desde el

menú *file* y posteriormente el formato FBX (.fbx), y como se muestra en el paso 3, solo se seleccionaron los elementos que se desea exportar.

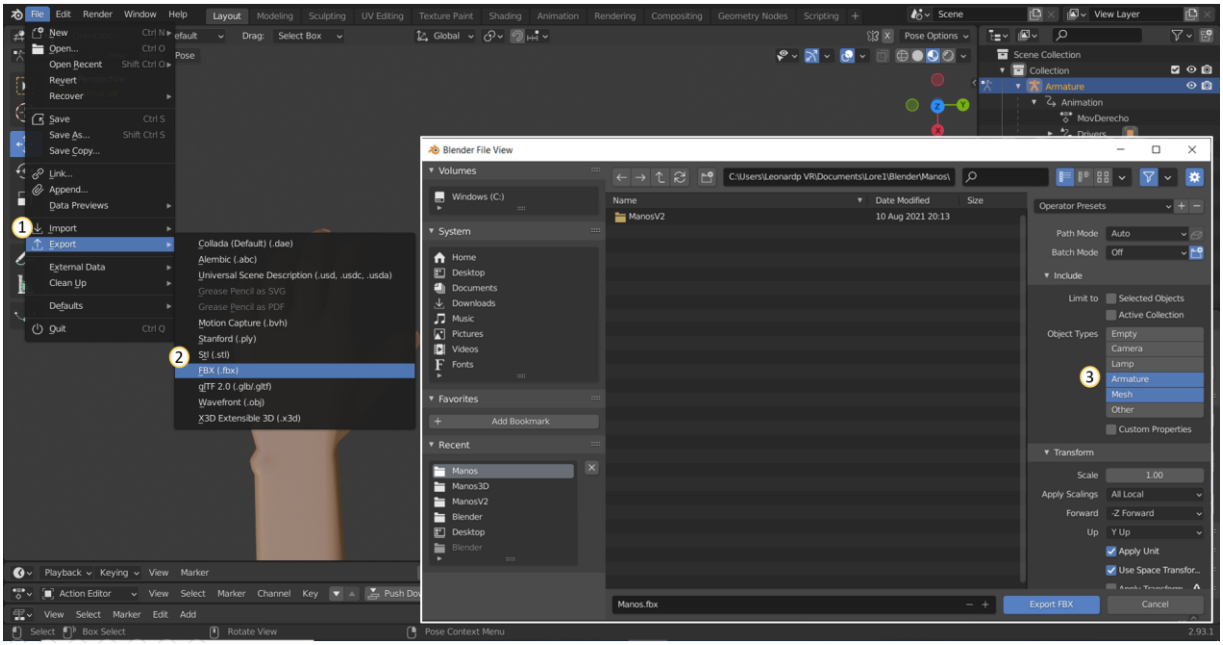

Fig A-3 Exportación del modelo en formato FBX

Para cargar el modelo virtual y las animaciones dentro de Unity, el archivo .fbx generado se insertó dentro del menú *Project* de esta plataforma. Los elementos que conforman al archivo .fbx se observan en el menú *inspector*, en donde es necesario verificar que en el submenú *rig* el tipo de animación sea genérica y se asigne un avatar para este modelo, y en el submenú *animation* se hayan guardado las animaciones importadas. Estos atributos se observan en la figura A-4.

Para lograr que las animaciones se asignaran a un evento específico se utilizó la herramienta “controlador de acción”, esta fue creado en el menú Project con la misma jerarquía del modelo importado para que se posible generar un vínculo entre ellos.

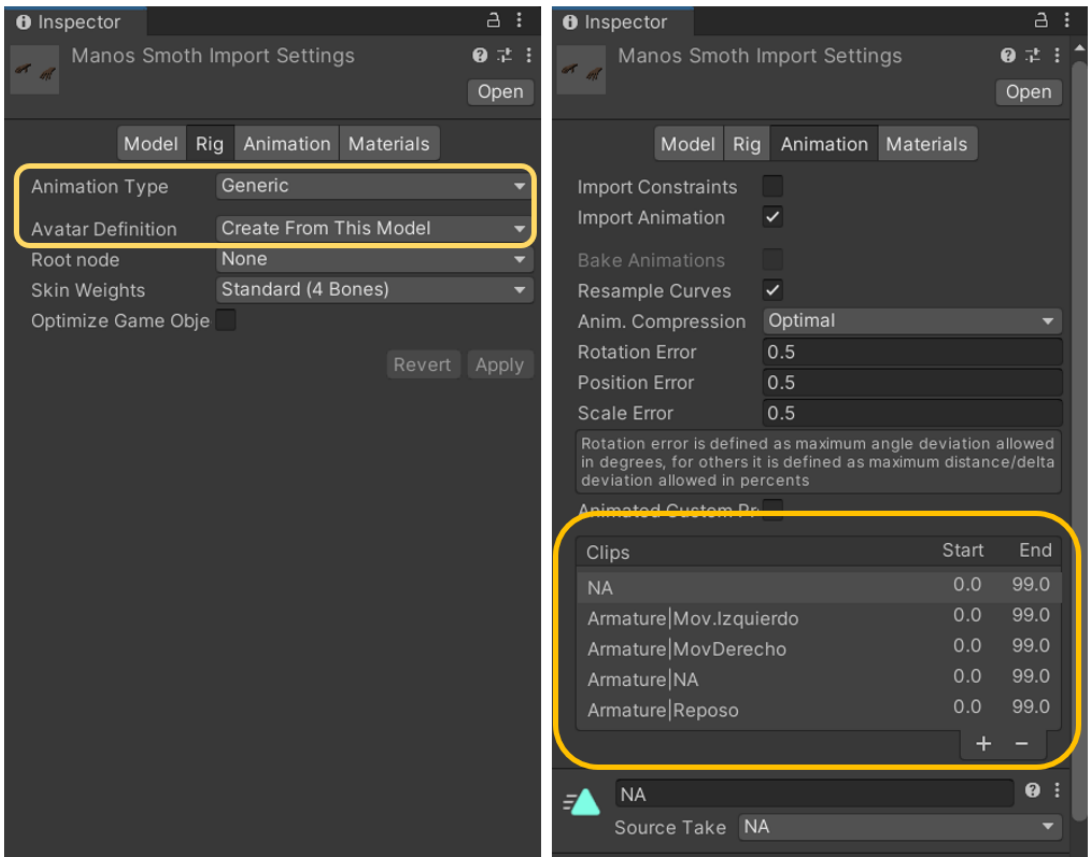

Fig A-4 Ajustes necesarios para importar correctamente en Unity el modelo en formato FBX

El controlador de animador utiliza diagramas a bloques para definir y establecer los condicionantes de la secuencia de animación que se reproducirá durante la simulación. En la figura A-5 se muestran las transiciones diseñadas para manipular el modelo virtual, en el centro se encuentra el diagrama a bloques, en él se indican los estados en los que podría encontrarse el modelo durante la simulación.

En la sección *Parameters* (ver figura A-5a) se observa marcado el estado de reposo, esto indica que fue definido como el estado por defecto, es decir que durante el inicio y el final de la simulación se reproducirá la animación asignada a él. Esta animación es asignada en el parámetro *motion* localizado en el menú *Inspector*, en la figura A-5b se muestra cómo se asigna la animación definida como *None* al estado de reposo.

Las transiciones entre una animación y otra se definen en diagrama a bloques a través de flechas que van hacía un estado u otro. Para que el cambio entre ellas no luzca de forma abrupta, en la sección *Transitions* se estableció una transición hacia el reposo un poco antes de que el movimiento termine (ver figura A-5c).

Por lo tanto, el diagrama a bloques diseñado se interpreta de la siguiente manera. Al iniciar la simulación el modelo se encontrará en estado de reposo, y conforme reciba los comandos de control desde Matlab® hará una transición hacia uno de los estados disponibles (bloques grises), una vez concluida la animación la simulación volverá al estado de reposo y esperará el siguiente comando de control. El proceso se repetirá hasta que se concluya el envío de datos, y el modelo permanecerá en estado de reposo hasta que la simulación sea cerrada.

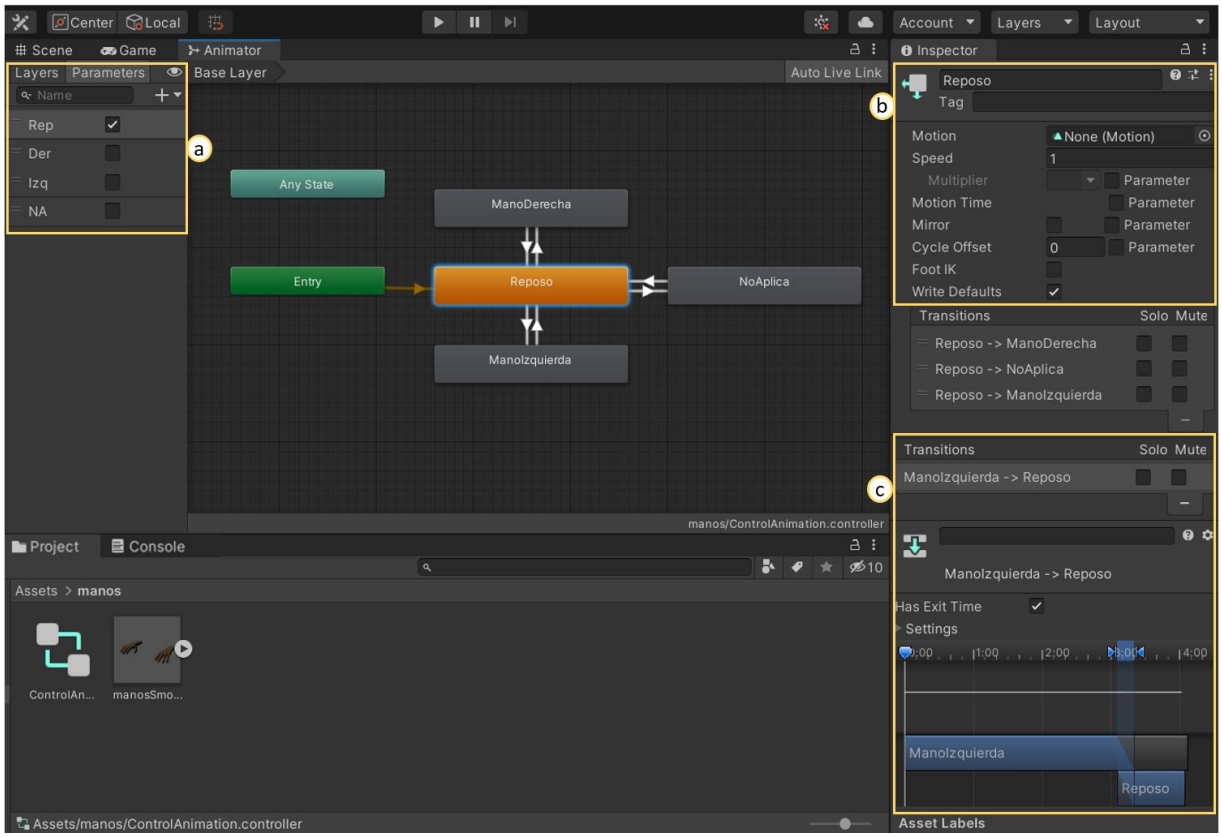

Fig A-5 Diseño de la transición de animaciones a través del Controlador de animador.

Los comandos de control fueron enviados desde la plataforma de programación y cálculo numérico Matlab® que fue definido como cliente para la comunicación a través del protocolo TCP/IP. Estos comandos de control fueron enviados durante la validación de los resultados obtenidos por el modelo clasificador. Cada que se valida el tipo de movimiento clasificado se abre el protocolo de comunicación y se envía el comando utilizando la función *fwrite(tcpipClient,comando)*, una vez enviado el comando de control la comunicación se cierra, se evalúa el siguiente ensayo clasificado y se repite el proceso. El código implementado en Matlab® se muestra en la figura A-6 figura N, en donde los comandos de control se muestran en la variable "*a*".

```
% Iniciar protocolo TCP-IP
tcpipClient = tcpip('127.0.0.1',55001,'NetworkRole','Client');
set(tcpipClient,'Timeout',30);

for i=1:k % k=Total de ensayos clasificados
   if pred(i,1) == 1 && pred(i,2) == 0
      fopen(tcpipClient);
      a = '2'; % movimiento izquierdo
      fwrite(tcpipClient,a);
      fclose(tcpipClient);
   elseif pred(i,1) == 0 && pred(i,2) == 1
      fopen(tcpipClient);
      a = '1'; % movimiento derecho
      fwrite(tcpipClient,a);
      fclose(tcpipClient);
   else
      fopen(tcpipClient);
      a = '3'; % movimiento no identificado
      fwrite(tcpipClient,a);
      fclose(tcpipClient);
   end
end
    fopen(tcpipClient);
    a = '0';
    fwrite(tcpipClient,a);
    fclose(tcpipClient);
```

Fig A-6. *Script* desarrollado en Matlab® para el envío de comandos de control utilizando el protocolo de comunicación TCP/IP.

La recepción de los comandos de control se realizó a partir del protocolo TCP/IP, para establecer esta comunicación fue necesario diseñar un *script* que se vinculó al modelo virtual y a la herramienta de control de animación. En la figura A-7 se muestra el *script* desarrollado en lenguaje de programación **C**#, este se integra de dos funciones principales definidas como *start* y *update*.

```
void Start() // Función Start
{
   // Se configura Unity para escuchar matlab
   listener = new TcpListener(IPAddress.Parse("127.0.0.1"), 55001);
   listener.Start();
   anim = GetComponent <Animator>();
}

void Update() // Función Update
{
   TcpClient client = listener.AcceptTcpClient();
   NetworkStream ns = client.GetStream();
   StreamReader reader = new StreamReader(ns);
   msg = reader.ReadToEnd();

   // Asignación de las animaciones a partir de comandos de control
   switch (msg.ToLower()) {
        case "1":
             anim.Play("ManoDerecha");
            break;
        case "2":
            anim.Play("ManoIzquierda");
            break;
        case "3":
            anim.Play("NoAplica");
            break;
        case "0":
            anim.Play("Reposo");
            break;
        default:
            anim.Play("Reposo");
            break;
      }
}
```

Fig A-7. *Script* desarrollado en Unity para la recepción de comandos de control utilizando el protocolo de comunicación TCP/IP.

La configuración de inicialización para la comunicación TCP/IP se establece en la función *start*, esta función es llamada al inicio de la simulación y no vuelve a ser llamada a lo largo de esta. Una vez establecida la comunicación, se utiliza la función *update* para recibir los datos enviados, durante esta función el código se actualiza en cada *frame*, lo que permite asignar los comandos de control hacia un estado del controlador de animador a través de una estructura *switch*.