UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

FACULTAD DE INGENIERÍA

PROYECTO DE INVESTIGACIÓN

EVALUACIÓN DE ALGORITMOS BASADOS EN LSTM PARA LA PREDICCIÓN DE PARÁMETROS DE MARCHA EN PERSONAS CON DISCAPACIDAD VISUAL

PREVIO A LA OBTENCIÓN DEL TÍTULO:

INGENIERÍA EN BIOMÉDICA

PRESENTADO POR:

21941023 PILAR ANDREA RUIZ ESTRADA

21941195 JENNIFER ELIZABETH ORELLANA MARTÍNEZ

ASESOR: KARLA REYES

CAMPUS SAN PEDRO SULA; OCTUBRE, 2023

DEDICATORIA

Dedico esta tesis, en primer lugar, a Dios, quien ha escuchado mis oraciones y me ha acompañado en todo momento. También, me dedico esta tesis a mí misma, por mi fortaleza y determinación para no rendirme. Y, de manera especial, dedico esta tesis a la memoria de mi abuelo Pilo Estrada Vásquez, quien ha sido una constante fuente de inspiración y aliento tanto en mi vida personal como académica, enseñándome que alcanzar nuestros sueños y metas requiere trabajo duro y perseverancia.

- **Pilar Ruiz**

-

Dedico esta tesis a mi hermana menor Isis Orellana, gracias por ser mi compañera de vida y de aventuras, por estar a mi lado siempre, ser mi apoyo constante y mi confidente que me anima en mis peores momentos. Mi más grande deseo es que cumplas tus metas y ser para ti una fuente de inspiración para que busques alcanzar todos tus sueños.

- **Jennifer Orellana**

AGRADECIMIENTOS

Quiero expresar mi sincera gratitud a Dios por guiarme y darme la sabiduría necesaria para completar esta tesis. Su gracia y dirección me han acompañado durante este viaje académico y ha demostrado estar para mi tanto en los momentos malos como en los buenos.

A mis padres, Jetty Estrada y Carlos Ruiz, su fe inquebrantable en mí y su negativa a permitir que me rindiera me han impulsado a superar obstáculos y a creer que soy capaz de enfrentar cualquier desafío, incluso cuando las circunstancias se tornan difíciles.

A Jenny Estrada, quien siempre ha creído en mí, brindándome su apoyo incondicional y amor. No tengo palabras suficientes para expresar cuánta gratitud siento hacia ti. Tu habilidad para inspirarme a superar mis propios límites, con la certeza de que puedo alcanzar mi mejor versión y que nunca es tarde para aprender algo nuevo, ha sido un regalo invaluable en mi vida.

A mis queridos abuelos, les agradezco de corazón por brindarme su amor incondicional y su valioso conocimiento.

A mis amigos, quiero agradecerles sinceramente por la alegría y el apoyo que me brindaron Siempre estuvieron ahí para mí y haciendo que esos momentos de presión fueran mucho más llevaderos. A Jennifer Orellana, mi compañera de tesis que hemos compartido este trayecto académico trabajando en equipo, brindándonos apoyo mutuo en cada paso.

A Potty Lu y Vicky, su compañía durante mis momentos de trabajo fue reconfortante y su presencia me llenó de tranquilidad.

- **PILAR RUIZ**

Agradezco a mis padres Julio Orellana y Gennie Martinez por haber sido un gran apoyo durante toda mi carrera universitaria, por confiar en mi e impulsarme a seguir mis sueños. A mi hermano mayor Cesar Orellana por ser una fuente de inspiración constante para mí. A mi tía Irma Martínez por estar pendiente de mí todos los días con sus mensajes y llamadas de apoyo.

También quiero agradecer a mi compañera constante y más grande amiga en el transcurso de mis años universitarios María José Alvarado por todas las noches de desvelo, las salidas que nunca podían faltar los fines de semana y por ser mi más grande apoyo dentro de toda mi carrera universitaria. También agradecer a mi amiga Alicia García por todo el apoyo brindado estos últimos años, por todas las risas, llantos y peleas que nos han llevado a donde estamos el día de hoy. De igual forma agradecer a mi compañera de tesis Pilar Ruiz por acompañarme en esta etapa tan importante de nuestras vidas.

Quisiera agradecer a nuestra asesora la Ing. Karla Reyes por acompañarnos durante todo el proceso de realización de esta tesis. De igual forma agradecer a cada uno de los docentes que nos acompañaron en nuestro proceso de formación académica. Especialmente al Ing. Manuel Gamero por enseñarnos a cuestionar todo lo que nos rodea.

También agradecer a mis amigas Sary Noguera, Andrea Alvarado, María Aguilar y Andrea Aguilar por estar ahí para escucharme cada vez que lo necesito y por ayudarme a recuperar la confianza en mí misma cada vez que la perdía. Ustedes son una de las razones por las que nunca me rendí.

Por último, pero no menos importante agradecer a mis queridas mascotas Toby y Doki, por ser una de mis mayores fuentes de inspiración y alegría. - **JENNIFER ORELLANA**

EPÍGRAFE

Definir es limitar

- *Oscar Wilde*

RESUMEN EJECUTIVO

La prevalencia creciente de personas con discapacidad visual en Honduras subraya la necesidad de tecnologías de asistencia para mejorar la movilidad de personas con discapacidad visual. Esta investigación se centra en el potencial de los algoritmos de aprendizaje automático, específicamente modelos híbridos LSTM+CNN y LSTM+FCN, para predecir parámetros de marcha esenciales, como la velocidad, la longitud de zancada, distancia y número de pasos. Usando una base de datos proporcionada por una investigación previa la cual contiene datos obtenidos por sensores inerciales que abarca a nueve participantes en diversas condiciones de caminata, estos modelos identifican patrones temporales y características espaciales. Se compararon las predicciones entre los modelos híbridos y el modelo biomecánico que utilizó configuraciones de TWS y THS.

En relación con el tamaño de zancada, el modelo LSTM+FCN obtuvo un error de 0.015 ± 0.014 metros, superando al modelo biomecánico. En cuanto a la velocidad, los errores absolutos medios fueron de 0.018 ± 0.013 m/s y 0.044 ± 0.017 m/s para los dos modelos, respectivamente, siendo significativamente más bajos que los del modelo biomecánico. En la predicción del contador de pasos con el modelo Seq2One mostró una mayor variabilidad con un valor de 16.06 ± 32.11 en las métricas de los experimentos. Sin embargo, en comparación con el modelo biomecánico, presentó un margen de error más amplio, lo que destaca la necesidad de optimización futura. En cuanto al parámetro de distancia, el error absoluto medio global fue de 13.13 ± 7.93 m para el modelo LSTM+4FCN y de 21.37 ± 9.25 m para el modelo LSTM+CNN, El modelo Seq2One obtuvo un error absoluto medio global de 2.96 ± 3.3. En comparación con el TWS, este logra superarlo en términos de predicción, sin embargo, obtuvo mejores resultados en comparación con el modelo LSTM+FCN. Los algoritmos basados en LSTM pueden proporcionar predicciones más precisas de los parámetros de la marcha en relación con el método tradicional, por ende, se deberían de realizar pruebas adicionales para los parámetros de distancia y número de pasos.

Palabras clave: Análisis de la marcha; Discapacidad visual; O&M; Redes neuronales

ABSTRACT

The increasing prevalence of people with visual impairment in Honduras highlights the need for assistive technologies to enhance the mobility of individuals with visual disabilities. This research focuses on the potential of deep learning algorithms, specifically hybrid models LSTM+CNN and LSTM+FCN, to predict essential gait parameters such as velocity, step length, distance, and step count. Using a database provided by a previous study, which includes data obtained from inertial sensors and covers nine participants in various walking conditions, these models identify temporal patterns and spatial characteristics. Predictions were compared between the hybrid models and the biomechanical model that used TWS and THS configurations.

Regarding step length, the LSTM+FCN model achieved an error of 0.015 ± 0.014 meters, surpassing the biomechanical model. As for speed, the mean absolute errors were 0.018 ± 0.013 m/s and 0.044 ± 0.017 m/s for the two models, respectively, which were significantly lower than those of the biomechanical model. In predicting step count with the Seq2One model, a higher variability was observed with a value of 16.06 ± 32.11 in the experiment metrics. However, compared to the biomechanical model, it exhibited a wider margin of error, emphasizing the need for future optimization. Concerning the distance parameter, the overall mean absolute error was 13.13 ± 7.93 m for the LSTM+4FCN model and 21.37 ± 9.25 m for the LSTM+CNN model, while the Seq2One model obtained an overall mean absolute error of 2.96 ± 3.3. Compared to TWS, it managed to outperform it in terms of prediction; although it achieved better results compared to the LSTM+FCN model. LSTM-based algorithms can provide more precise predictions of gait parameters compared to the traditional method; however, additional tests should be conducted for parameters like distance and step count.

Key words: Gait analysis; neuronal networks; O&M; visual impairment

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ECUACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

LISTA DE SIGLAS

ASHONCI	Asociación Hondureña de Ciegos
CAIC	Centro de Atención Integral para Ciegos
CNN	Convolutional Neural Network
COM	Center of mass
CSV	Valores Separados por Comas
DL	Deep learning
DNN	Red neuronal profunda
ETAS	Electronic Travel Aid Systems
FCN	Fully Convolutional Neural Network
FOAL	Fundación ONCE para América Latina
GPRS	General Packet Radio Service
GPS	Global Positioning System
GRF	Ground Reaction Forces
GSM	Global System for Mobile Communications
GPU	Graphics Processing Unit
HAR	Human Activity Recognition
HH	Altura de la Mano
ICEVI	Comité Internacional para la Educación de las Personas Ciegas a nivel de América Latina

IA	Inteligencia Artificial
IMUs	Inertial Measurement Units
LSTM	Long-Short Term Memory
LOSO	Leave-one-subject-out validation
MAE	Mean absolute error
ML	Machine Learning
MLP	Multi layer perceptron
MOCAP	Sistemas de captación de movimiento
MSE	Mean Squared Error.
O&M	Orientación y Movilidad
OMS	Organización Mundial de la Salud
OPS	Organización Panamericana de la Salud
PDV	Personas con discapacidad visual
PLN	Procesamiento del lenguaje natural
PRID	Rehabilitación e Inclusión de Personas con Discapacidad
RFID	Radio-Frequency Identification
RNA	Red Neuronal Artificial
RNN	Red Neuronal recurrente

RMSE	Root mean square error
SKDH	SciKit Digital Health
SL	Stride Length
SZ	Zona de Seguridad
THS	Configuración de un sensor de muslo
TWS	Configuración de dos sensores
ULAC	Unión Latinoamericana de Ciegos
UNCIH	Unión nacional de ciegos hondureños

GLOSARIO

Autocodificador: "Es un tipo de algoritmo cuyo objetivo de aprender una representación "informativa" de los datos que puede utilizarse para diferentes aplicaciones a aprendiendo a reconstruir un conjunto de observaciones de entrada lo suficientemente bien"(Michelucci, 2022).

Características: "El conjunto de atributos, a menudo representado como un vector, asociados a un ejemplo"(Mohri et al., 2018).

Épocas: "Cada iteración sobre todos los datos de entrenamiento"(Chollet, 2021).

Etiquetas: "Valores o categorías asignadas a ejemplos. En la regresión, los elementos se asignan etiquetas de valores reales"(Mohri et al., 2018).

Función de pérdida: "Una función que mide la diferencia o pérdida entre una etiqueta predicha y una etiqueta verdadera". (Mohri et al., 2018)

Muestra de entrenamiento: "Ejemplos utilizados para entrenar un algoritmo de aprendizaje" (Mohri et al., 2018).

Muestra de validación: "Ejemplos utilizados para ajustar los parámetros de un algoritmo de aprendizaje cuando se trabaja con datos etiquetados" (Mohri et al., 2018).

Red LSTM: "Son un tipo de RNN que se caracterizan por su capacidad de recordar un dato relevante en la secuencia y preservarlo durante varios instantes de tiempo. Por lo tanto, tienen memoria a corto y a largo plazo"(Buisson Garcia, 2023).

Red neuronal artificial: "Son modelos matemáticos que se inspiran en las estructuras neuronales biológicas del cerebro humano. Estas redes están compuestas por capas organizadas de unidades interconectadas o nodos"(Arana, 2021).

Redes neuronales convolucionales: "Son un caso especial de redes neuronales feedforward que consiste en una secuencia de capas, donde cada capa transforma las activaciones o salidas de la capa anterior a través de otra función diferenciable" (Teuwen & Moriakov, 2020)

Redes neuronales recurrentes: "Son un tipo de redes neuronales que capturan activamente sus dependencias secuenciales y temporales" (Arana, 2021).

Redes totalmente convolucionales: "Son una amplia clase de CNNs que se basan únicamente en operaciones espacialmente invariantes para producir sus salidas, y por lo tanto son naturalmente espacialmente generalizables." (Nibali et al., 2018)

Regresión escalar: "Una tarea donde el objetivo es un valor escalar continuo"(Chollet, 2021).

Regresión vectorial: "Una tarea donde el objetivo es un conjunto de valores continuos"(Chollet, 2021).

Tamaño de lote: Es la cantidad de ejemplos de datos de entrenamiento que se procesan en paralelo durante una iteración de entrenamiento (Chollet, 2021).

Tamaño de ventana: Los pasos de tiempo que contiene la cantidad de datos o elementos que se incluyen en una ventana deslizante al analizar una secuencia de información (Jaén-Vargas et al., 2022).

I. INTRODUCCIÓN

La marcha, siendo un aspecto integral de la vida humana, permite a las personas movilizarse, posicionarse en su entorno, relacionarse con él y realizar diversas actividades de forma autónoma. Sin embargo, aquellos con discapacidad visual enfrentan dificultades para llevar a cabo estas tareas debido a limitaciones visuales causadas por enfermedades oculares, lesiones o condiciones congénitas. Esto conlleva a una complejidad adicional en su movilidad, dependiendo en gran medida de asistencia técnica, bastones, perros guía y capacitación en orientación y movilidad (O&M) para utilizar sus otros sentidos.

El proceso de orientación y movilidad (O&M) es crucial para permitir a las personas ciegas o con visión reducida desplazarse de manera segura e independiente en su entorno. No obstante, las nuevas tecnologías están abriendo oportunidades para mejorar la autonomía de estas personas con discapacidad visual.

En Honduras, la falta de recursos económicos y de transporte dificulta el acceso de las personas con discapacidad visual a entrenamientos y tecnologías de asistencia adecuadas. Para abordar este problema, se ha buscado la implementación de soluciones tecnológicas menos complejas y costosas, como el uso de sensores inerciales, que capturan y analizan información sobre el movimiento y posición del cuerpo a un costo más accesible.

Además, la aplicación de redes neuronales en el aprendizaje automático ha mejorado la precisión y estimación de parámetros de la marcha, lo que las convierte en una herramienta eficaz para procesar datos y predecir dichos parámetros al utilizar sensores inerciales. Esta investigación propone la aplicación de dos modelos basados en redes neuronales, el modelo LSTM + CNN y LSTM+FCN, para predecir datos de movimiento como el tamaño de zancada, la cantidad de pasos y la velocidad, particularmente considerando las dependencias a largo plazo en el tiempo.

La combinación de redes neuronales y sensores inerciales ha mostrado ser prometedora en la búsqueda de soluciones efectivas para la movilidad de personas con discapacidad visual. Estas tecnologías podrían resultar útiles para calcular con precisión los parámetros de la marcha, convirtiéndose en herramientas de apoyo crucial para el desarrollo de tecnología asistencial que permita a estas personas desplazarse de manera independiente y segura en su entorno. Estos parámetros, como la longitud de la zancada, la distancia, el contador de pasos y la velocidad, son

esenciales para la orientación y movilidad independientes. Las redes neuronales pueden aprender a reconocer patrones complejos en los datos de los sensores inerciales, lo que permite una estimación más precisa de estos parámetros. No obstante, se requiere una serie de pruebas para validar su eficacia, además de realizar una comparativa con métodos tradicionales para determinar cuál es más efectivo en la predicción de los parámetros de la marcha.

El presente informe se estructura de la siguiente manera: el primer capítulo presenta el planteamiento del problema, estableciendo los fundamentos de la investigación y su abordaje. El segundo capítulo desarrolla el marco teórico respaldado por una revisión bibliográfica. El tercero detalla los métodos y procedimientos utilizados para recopilar los datos necesarios, mientras que el cuarto capítulo presenta los resultados y análisis. El quinto capítulo concluye la eficacia de los modelos basados en redes neuronales en la predicción de parámetros de marcha en personas con discapacidad visual.

II. PLANTEAMIENTO DEL PROBLEMA

En el presente capítulo del planteamiento del problema se detallan los aspectos clave que se abordan en esta investigación. En primer lugar, se presentan los precedentes relevantes, incluyendo estudios previos que resultan pertinentes para la investigación. A continuación, se define de manera precisa el problema investigado. Posteriormente, se justifica la necesidad de abordar este problema, considerando su impacto en la calidad de vida y la independencia de las personas ciegas. Además, se presentan las preguntas de investigación que guían este estudio y los objetivos que se esperan alcanzar.

2.1 PRECEDENTES DEL PROBLEMA

La marcha es una actividad crucial para la independencia y la participación social de las personas ciegas. Sin embargo, las limitaciones en la capacidad de medir y analizar de manera precisa los parámetros de marcha en esta población han dificultado la implementación de estrategias efectivas de rehabilitación y apoyo. (Reyes Leiva, Jaén-Vargas, Codina, et al., 2021). Según (OMS, 2023), en el mundo hay al menos 2200 millones de personas con deterioro de la visión cercana o distante. En al menos 1000 millones de esos casos, es decir, casi la mitad, la discapacidad visual podría haberse evitado o todavía no se ha aplicado un tratamiento.

La pérdida total o parcial de la visión conlleva cambios que impactan en las relaciones familiares y sociales. Las personas con discapacidad visual pueden enfrentar limitaciones en la realización de sus actividades diarias debido a obstáculos presentes en el entorno y la falta de accesibilidad de espacios públicos o privados. Mejorar la eficiencia y fluidez de la marcha en personas ciegas conlleva beneficios físicos y emocionales, como reducción de la fatiga, un menor riesgo de caídas y una mayor confianza en sí mismas.

Cuando una persona nace con discapacidad visual o experimenta una lesión o enfermedad que resulta en esta condición, es necesario proporcionarle apoyo a través de rehabilitación. Algunas herramientas utilizadas en el aprendizaje de personas con discapacidad visual incluyen el sistema braille, el uso de bastones largos para la movilidad, técnicas de alimentación adaptada a la falta de visión, así como el aprovechamiento óptimo de la visión residual y la enseñanza de habilidades para mejorar el funcionamiento visual en las actividades diarias. Además, los especialistas brindan capacitación en O&M para mejorar la independencia en la vida diaria (Reyes

Leiva, Jaén-Vargas, Cuba, et al., 2021). Sin embargo, es fundamental priorizar la adquisición de habilidades motoras para mejorar la eficiencia en el desplazamiento y la funcionalidad general antes de utilizar ayudas o dispositivos que promuevan la movilidad independiente (Parreira et al., 2017).

Existe un interés fundamental en mejorar la marcha de las personas ciegas y proporcionarles herramientas de apoyo efectivas en el campo de la rehabilitación y la calidad de vida. La utilización de sensores inerciales y algoritmos de aprendizaje automático, como los algoritmos a base de LSTM, ha surgido como una posible solución para predecir con precisión los parámetros de marcha en personas ciegas.

A lo largo de los últimos años, los avances en tecnologías de sensores inerciales y el crecimiento del campo del aprendizaje automático han permitido abordar de manera más efectiva los desafíos relacionados con la predicción de parámetros de marcha en personas ciegas (Flores & Manduchi, 2018). La combinación de estos avances ha llevado al surgimiento de algoritmos a base de LSTM como una herramienta prometedora en la predicción de parámetros de marcha. Investigaciones clave en este campo han contribuido a mejorar la comprensión y la aplicación de estos enfoques. Sin embargo, es necesario profundizar en la aplicación de algoritmos de predicción para mejorar el rendimiento y precisión en la estimación de los parámetros de marcha en personas ciegas ya que, aunque los avances recientes en estas tecnologías han permitido abordar de manera más efectiva los desafíos relacionados con la predicción de parámetros de marcha, aún existen áreas en las que se pueden mejorar. Específicamente en la predicción de parámetros de marcha en personas con discapacidad visual es un tema que debe abordarse a profundidad debido a la complejidad de la marcha y la variabilidad entre estos individuos.

2.2 DEFINICIÓN DEL PROBLEMA

El problema radica en la precisión de los métodos utilizados para calcular los parámetros de marcha en personas ciegas. Los métodos que utilizan sensores inerciales colocados en los segmentos de las piernas se concentran en dos enfoques para determinar los parámetros de la marcha. "Un enfoque implica la identificación de variables relacionadas que se correlacionan con parámetros desconocidos. Otro enfoque implica la estimación directa utilizando modelos biomecánicos o cadenas cinemáticas" (Reyes Leiva, Jaén-Vargas, Cuba, et al., 2021). Estos

enfoques tradicionales pueden presentar limitaciones al capturar patrones complejos en los datos de marcha de estas personas. Esto se debe a que la forma de caminar y los patrones de marcha pueden diferir en personas con discapacidad visual. Por otro lado, los métodos basados en algoritmos LSTM ofrecen una mayor precisión y capacidad de adaptación en la predicción de los parámetros de marcha ya que evitan el problema de la señal de error decreciente entre capas haciendo que las redes LSTM "recuerden" la información durante más tiempo (Emmert-Streib et al., 2020).

2.3 JUSTIFICACIÓN

La marcha es un aspecto fundamental en la vida cotidiana de las personas, y su análisis puede proporcionar información valiosa para el diagnóstico y seguimiento de diversas condiciones de salud. En el caso de las personas ciegas, la evaluación de la marcha se vuelve aún más relevante, ya que su habilidad para moverse de manera segura y eficiente depende en gran medida de una buena percepción y control de los parámetros de marcha (Alemán-Ramírez, 2020). Realizar la predicción de la marcha en personas ciegas podría promover la independencia y mejorar la funcionalidad de cada individuo de manera personalizada.

Las personas con discapacidad visual utilizan distintos dispositivos de apoyo para poder ejecutar la marcha de forma segura, como lentes y bastones. Además, se están estudiando distintas tecnologías de asistencia portátiles como los sistemas electrónicos para viaje. Sin embargo, muchas de ellas se basan en arquitecturas complejas y muy costosas para adquirirlas. Por ejemplo, dispositivos de detección de objetos que emplean tecnologías de láser o cámaras avanzadas para identificar obstáculos y proporcionar retroalimentación al usuario o sistemas de navegación GPS especiales diseñados específicamente para personas que ciegas incluyen características avanzadas, como indicaciones detalladas y alertas de puntos de interés.

Según (Organización mundial de la salud, 2023) La discapacidad visual representa una significativa carga económica a nivel global, estimándose que los costos anuales relacionados con la pérdida de productividad debido a deficiencias visuales alcanzan la suma de US$ 411 000 millones a nivel mundial. “El uso de sensores inerciales, como los acelerómetros, giroscopios y magnetómetros, junto con algoritmos de aprendizaje automático, como los algoritmos LSTM, han

demostrado ser prometedores en la predicción de parámetros de marcha" (Reyes Leiva, Jaén-Vargas, Cuba, et al., 2021).

Los sensores inerciales son dispositivos de bajo costo y accesibles utilizados para recopilar datos precisos sobre la orientación, velocidad, aceleración y posición de un objeto o una persona. Estos datos resultan especialmente beneficiosos para la predicción de los parámetros de la marcha, ya que permiten obtener información detallada sobre los parámetros de movimiento involucrados y en el caso de personas con discapacidad visual las cuales tienen una forma de marcha distinta a la de una persona normal llegan a ser muy útiles para su estudio.

El uso de modelos basados en LSTM puede mejorar la precisión de los datos obtenidos, ya que su capacidad para capturar patrones complejos en datos secuenciales los convierte en una herramienta altamente efectiva. Al realizar la evaluación de los algoritmos basados en LSTM los cuales son el LSTM+ CNN y LSTM+ 4FCN utilizados en la predicción de la marcha se podrá verificar la eficacia para obtener los datos correspondientes y si estos concuerdan con los del método tradicional o si existe una variación entre los dos métodos. Al momento de hacer la comparativa se podrá demostrar cuál de los dos métodos es más beneficioso para el estudio en donde podremos identificar las fortalezas y debilidades de cada uno, cuál es el nivel de rendimiento que brindan, una mejor comprensión de los datos obtenidos y si la utilización de algoritmos basados en LSTM es mejor que el método tradicional, esto respaldaría su viabilidad y relevancia en el campo de investigación de la predicción de parámetros de marcha en personas ciegas.

2.4 PREGUNTAS DE INVESTIGACIÓN

2.4.1 PREGUNTA GENERAL

- ¿Qué tan precisa es la predicción de parámetros de marcha en personas ciegas utilizando algoritmos basados en LSTM y cómo se compara con el método tradicional de cálculo?

2.4.2 PREGUNTAS ESPECÍFICAS

- ¿Cuál es la importancia del etiquetado preciso de la base de datos de experimentos de marcha de personas ciegas utilizando sensores inerciales?
- ¿Cómo calcular los parámetros de tamaño de zancada, velocidad de marcha y contador de pasos?

- ¿Cómo se puede optimizar el proceso de entrenamiento de los algoritmos basados en LSTM utilizando la base de datos etiquetada para mejorar la precisión en la predicción de los parámetros de marcha en personas ciegas?
- ¿Cómo afectan las modificaciones en los parámetros de los modelos CNN-LSTM y LSTM-4FCN, en la mejora de la precisión del algoritmo y en la predicción de los parámetros de marcha?
- ¿Cómo se compara el rendimiento y la precisión del algoritmo entrenado en la predicción de los parámetros de marcha en personas ciegas con el método tradicional?

2.5 OBJETIVOS

2.5.1 OBJETIVO GENERAL

- Evaluar la precisión de los algoritmos basados en LSTM en la predicción de parámetros de marcha en personas ciegas y compararlo con el método tradicional de cálculo.

2.5.2 OBJETIVOS ESPECÍFICOS

- Etiquetar la base de datos existente de experimentos de marcha de personas ciegas utilizando sensores inerciales.
- Calcular los parámetros de tamaño de zancada, velocidad de la marcha, distancia y contador de pasos.
- Entrenar los algoritmos basados en LSTM utilizando la base de datos etiquetada.
- Realizar modificaciones en los parámetros de los modelos basados en LSTM para mejorar su precisión en la predicción de los parámetros de marcha.
- Generar una discusión en torno a la efectividad del método tradicional en comparación con el método de Deep Learning, considerando la influencia de los parámetros de la marcha y el método de obtención y preparación de los datos etiquetados.

III. MARCO TEÓRICO

El siguiente capítulo tiene como objetivo establecer el contexto y los fundamentos conceptuales que sustentan esta investigación. En este sentido, se aborda la situación actual del problema en el ámbito de la marcha en individuos con discapacidad visual, considerando tanto el macroentorno social y sanitario que afecta a esta población y los hallazgos que ha habido a nivel mundial, así como el microentorno específico de las necesidades que tiene esta población en Honduras. Se conceptualizan los temas más importantes relacionados con la marcha en personas ciegas, destacando la relevancia y las limitaciones actuales en cuanto a la evaluación de sus parámetros de marcha. Asimismo, se presentan las teorías de sustento que fundamentan el uso de algoritmos basados en LSTM para la predicción en este contexto, junto con las bases teóricas que sustentan el uso de sensores inerciales como herramienta de recolección de datos. Además, se revisan las metodologías utilizadas por otros investigadores en estudios similares, y se describen los instrumentos y técnicas de análisis empleados en investigaciones previas.

3.1 ANÁLISIS DE LA SITUACIÓN ACTUAL

A nivel mundial, la discapacidad visual representa un desafío significativo para la salud pública, con una creciente prevalencia de personas ciegas que enfrentan limitaciones en su movilidad y calidad de vida. A través de una revisión exhaustiva de la literatura existente, se han identificado diversos estudios previos que abordan la predicción de parámetros de marcha en personas ciegas. Sin embargo, la mayoría de estos enfoques han utilizado métodos tradicionales de aprendizaje automático y han demostrado desafíos en términos de precisión y generalización. Adicionalmente, es importante destacar la escasa investigación específica que ha explorado el potencial de algoritmos basados en LSTM en este contexto. En el caso particular de Honduras, se enfrentan desafíos y limitaciones particulares que deben ser cuidadosamente considerados y abordados en el desarrollo de esta investigación.

3.1.1 MACRO ENTORNO

A nivel global, hay una creciente conciencia sobre la importancia de abordar los desafíos que enfrenta la población con discapacidad visual. La necesidad de soluciones efectivas y precisas para evaluar y mejorar la marcha en personas ciegas se ha convertido en una prioridad para la sociedad en general. En respuesta a esta demanda, las investigaciones recientes se han enfocado

en la aplicación de tecnologías de asistencia y análisis de la marcha, con el objetivo de proporcionar soluciones innovadoras y basadas en evidencia que contribuyan a mejorar la independencia y la calidad de vida de esta población.

3.1.1.1 Latinoamérica

La ceguera y la deficiencia visual constituyen importantes problemas de salud en América Latina. "En muchos países es estimado que por cada millón de habitantes hay 5.000 ciegos y 20.000 personas con discapacidad visual, al menos 2/3 partes es debido a causas tratables como la catarata, defectos refractivos, retinopatía diabética, ceguera infantil, glaucoma, oncocercosis y tracoma" (*Salud visual - OPS/OMS | Organización Panamericana de la Salud*, 2018). "En América Latina, la prevalencia de glaucoma varía entre 1% y 3,4% en personas mayores de 50 años y alcanza a representar entre 15% y 20% de las causas de ceguera en los países con más ascendencia africana" (*Salud visual - OPS/OMS | Organización Panamericana de la Salud*, 2018)

En muchos países de Latinoamérica, realizan iniciativas para mejorar la calidad de vida de las personas con discapacidad visual, especialmente mediante la implementación de programas de rehabilitación. Dentro de estos esquemas, se ofrecen diversos programas destinados a brindar apoyo y oportunidades a las personas ciegas las cuales son programas de Rehabilitación a Domicilio, de residencial y basados en la comunidad.

La ULAC es la principal referencia a nivel de organizaciones de personas ciegas en América Latina. Esta asociación representa y aboga por los derechos de las personas ciegas en toda la región. En colaboración con FOAL, han logrado implementar centros en diez países de América Latina, incluyendo Guatemala, Nicaragua, El Salvador, Honduras, Panamá, Costa Rica, República Dominicana, Perú, Paraguay y Chile. Estos centros ofrecen programas innovadores de rehabilitación y capacitación, mejorando significativamente la calidad de vida de las personas con discapacidad visual. Además, su labor ha generado una mayor conciencia pública sobre los derechos y necesidades de este grupo, promoviendo así la inclusión y la igualdad de oportunidades en la sociedad latinoamericana.

- ✓ Tecnologías de asistencia para personas con discapacidad visual

Las personas con discapacidad visual dependen mucho de herramientas de apoyo para movilizarse en el entorno y evitar los obstáculos presentes, por lo tanto, su calidad de vida es

difícil a comparación a de una persona que tiene una visión normal, como por ejemplo para desarrollar relaciones personales o al momento de trabajar para un lugar. Algunas de las herramientas comunes son el uso de bastón o los lazarillos, no obstante, estas herramientas contienen limitaciones como que el bastón solo tiene un alcance de 1.5 metros y en cuanto al uso de los lazarillos, se demora mucho ya que se debe de entrenar al perro y al mismo tiempo el usuario se debe de acostumbrar a él.

En la actualidad se han creado innovaciones para poder mejorar la calidad de vida de las personas que sufren de discapacidad visual, estas herramientas han sido beneficiosas ya que han brindado buenos resultados proporcionando dispositivos electrónicos diseñados para adecuarse a las actividades de movilidad y desplazamiento con el propósito de hacerlas más fáciles y convenientes. A continuación, se presentan algunas de estas tecnologías que se usan en la actualidad:

- ✓ Herramientas electrónicas integradas en bastones

Se han enfocado en mejorar el desarrollo de estos dispositivos con la implementación de sensores de proximidad, infrarrojos, GPS, GSM, GPRS que permite posicionar a las personas, facilitando su ubicación precisa en un lugar determinado o guiándolas de manera efectiva hacia una dirección específica y RFID que mejoran la navegación en áreas con diversas etiquetas o señales ubicadas estratégicamente, y brindar una experiencia de orientación y guía más efectiva en dichos entornos. Unos de los ejemplos destacados con este tipo de tecnología es el Smart Cane."El Smart Cane es un dispositivo portable que cuenta con un sistema de sensores ultrasónicos para detectar los obstáculos en frente del usuario y poder proporcionar instrucciones a través de mensajes de audio o alertas vibratorias en la mano" (Salazar & Mesa, 2019).

- ✓ Herramienta a partir de algoritmos de Visión Artificial

"La visión artificial es una disciplina que pretende simular los diferentes procesos y elementos que otorgan visión a una máquina. Estas incluyen las propiedades geométricas, como el color, iluminación, textura y composición" (Salazar & Mesa, 2019).

Cuenta con dos etapas, una es la generación y la otra es la interpretación de la imagen que se obtienen por medio de una cámara. La visión artificial será un remplazo a la visión humana en los dispositivos de apoyo en donde se centra en la asistencia de orientación, movilidad,

reconocimiento de objetos, acceso a información e interacción social, no obstante, contiene limitaciones como la adaptación a las necesidades particulares de las personas, esto ocasiona que no se puedan cumplir con los requisitos que solicitan los usuarios.

- ✓ Herramientas integrando Visión por Computador y Redes Neuronales Artificiales

"Las redes neuronales fueron creadas dentro del movimiento de investigación en Inteligencia Artificial. Surgiendo como la posibilidad de poder realizar tareas cotidianas para un ser humano con la ayuda de máquinas" (Salazar & Mesa, 2019).

Los dispositivos que hacen uso de sistemas entrenados con RNA y la adquisición de imágenes con cámaras para el reconocimiento de patrones tienen la capacidad de detectar diversos tipos de objetos como sillas, muebles, puertas, esquinas y hacen de manera más fácil el reconocimiento de personas rostros y expresiones faciales. Se emplean de manera conjunta diversos algoritmos de visión por computadora para el procesamiento de imágenes, junto con técnicas de inteligencia artificial para analizar los datos y obtener los resultados finales.

- ✓ Herramientas con procesos de clasificación de imágenes usando Deep Learning

"El Deep Learning o aprendizaje profundo, es una rama del Machine Learning con una familia de algoritmos que simulan el proceso realizado por las neuronas cerebrales, para llevar a cabo el reconocimiento de voz, imágenes, palabras, entre otras aplicaciones"(Salazar & Mesa, 2019).

Implementar algoritmos DL a las herramientas de apoyo para personas con discapacidad visual permiten realizar el reconocimiento de expresiones faciales y una clasificación de imágenes que proviene de grandes bases de datos, uso de robot que tiene la capacidad de evadir objetos y categorizarlos, así mismo en la asistencia en la movilidad del usuario.

3.1.2 MICRO ENTORNO

Honduras, como país de América Central, presenta un entorno único con características socioeconómicas, culturales y de salud que influyen en la disponibilidad de recursos y el acceso a tecnologías de asistencia. La discapacidad visual en Honduras es un tema relevante para el sistema de salud y la sociedad, y la mejora de la movilidad y calidad de vida de las personas ciegas representa un objetivo importante. Sin embargo, se deben abordar desafíos específicos, como la disponibilidad de infraestructura para la investigación, la capacitación de personal especializado

y la consideración de aspectos culturales y éticos. En este sentido, el presente estudio se enmarca en el contexto del micro entorno hondureño, buscando desarrollar soluciones adaptadas a esta realidad y contribuir al avance de la tecnología de evaluación de la marcha en beneficio de la comunidad de personas ciegas en Honduras.

3.1.2.1 Honduras

"En Honduras, en el 2013 se realizó un estudio de prevalencia de la ceguera en donde se estimó una prevalencia de la ceguera del 1,9%" (Eunice Amador Rosa et al., 2022).

Entre 2018 y 2019 se realizó de la misma forma a diferencia que "se evaluaron 7992 pacientes en donde se obtuvieron resultados de porcentaje del 2,5% con deficiencia visual grave y del 18,5% con deficiencia visual moderada. El 79% presentaba visión normal o deficiencia visual leve"(Eunice Amador Rosa et al., 2022)."La prevalencia de la ceguera ha incrementado de manera secuencial del 1.9% al 4.5%" (Eunice Amador Rosa et al., 2022). Desgraciadamente, esta situación sigue siendo la misma debido a la falta de recursos suficientes para apoyar a las personas con discapacidad visual.

En comparación con otros países, el número de organizaciones dedicadas a la rehabilitación de personas ciegas en Honduras es relativamente limitado, no obstante, esta la ASHONCI, CAIC, PRID y UNCIH, y Programa Nacional para la Prevención.

Para esta investigación, se solicitó la ayuda de UNCIH con el objetivo de obtener información relevante sobre la situación en Honduras en relación con las personas ciegas, en los anexos 151 y 152 se observan las actividades que realizamos junto a ellos. UNCIH es una organización sin fines de lucro tiene como misión ayudar a las personas con discapacidad visual a que se les cumplan sus derechos. A través de procesos de incidencia política, buscan mejorar las condiciones de vida y lograr una inclusión real y significativa en la sociedad.

La UNCIH lleva 33 años operando como una organización legalmente constituida compuesta por personas ciegas. A nivel nacional, UNCIH es la principal representante de las personas ciegas y ha establecido convenios a nivel internacional. Actualmente, UNCIH opera en 12 departamentos de Honduras, incluyendo Gracias Lempira, que es la filial más reciente, además de Esperanza Intibucá, Santa Bárbara, El Paraíso, Olancho, Cortés, Progreso, La Ceiba, La Paz, y

Tegucigalpa, donde se encuentra la sede central. La organización cuenta con aproximadamente 1200 afiliados a nivel nacional, y esta cifra se actualiza constantemente.

En el departamento de Cortés, UNCIH cuenta con 184 afiliados, cubriendo los 12 municipios de ese departamento, incluyendo niños, jóvenes y adultos. Los afiliados varían en edad, con un porcentaje entre 18 y 50 años y un mínimo de 70 años. También incluyen un 5% de niños de entre 5 y 14 años, así como adolescentes de entre 15 y 17 años.

Las necesidades de los afiliados en el departamento de Cortés incluyen la falta de instalaciones, la necesidad de personal de apoyo y voluntarios, así como la necesidad de equipo. La movilización de los afiliados y sus necesidades médicas también son un desafío. La falta de recursos económicos es el problema principal ya que las herramientas utilizadas por las personas ciegas son costosas y deben comprarse en el extranjero.

Dentro de las posibilidades de sus recursos para la impartición de programas de orientación y movilidad, UNCIH cuenta con la ayuda gratuita de profesionales capacitados por ICEVI. Los programas incluyen rehabilitación y movilidad, actividades de la vida diaria, técnicas de rastreo, técnicas de clasificación de ropa, adiestramiento manual, braille, rehabilitación en tecnología, caligrafía y comunicación.

UNCIH trabaja tanto con niños como con adultos, con enfoques diferentes. En niños la estimulación temprana es crucial, y la participación de la familia es esencial en el proceso de rehabilitación. En cuanto a los adultos al ya tener una idea de cómo es un entorno se les es más fácil, sin embargo, dependerá de sus habilidades, por lo tanto, los programas son personalizados y se adaptan a las necesidades individuales de cada persona ciega.

Además de UNCIH, otras organizaciones como Infracnovi se centran en la estimulación temprana y la educación inclusiva en niños. Sin embargo, UNCIH se enfoca particularmente en jóvenes y adultos, ya que han sido históricamente descuidados en términos de rehabilitación en Honduras y busca abordar esta brecha a nivel nacional.

3.2 CONCEPTUALIZACIÓN

En esta sección se establecen los fundamentos conceptuales que sustentan este estudio. Se exploran y describen las principales teorías y conceptos relacionados con la marcha humana, la discapacidad visual y el análisis biomecánico. De igual forma se aborda la relevancia de los

sensores inerciales en la recopilación de datos de movimiento, y se examinan las distintas redes neuronales.

3.2.1 DISCAPACIDAD VISUAL

La discapacidad visual es "ausencia o disminución de la capacidad para ver que dificulta o impide la realización normal de las tareas visuales, provocando dificultades de interacción entre el sujeto afectado y su entorno" (Ortiz, 2011).

Las personas que sufren de discapacidad visual llegan a tener dificultades al movilizarse en el entorno como la detección de obstáculos, orientarse en nuevos espacios, identificar espacios o personas y dificultad para leer un texto ya sea de cerca o de lejos. La discapacidad visual puede manifestarse en distintas etapas de la vida y su progresión varía según la edad en la que se origine. Esto puede deberse a un desarrollo inadecuado de los órganos visuales o a padecimientos y accidentes que afecten los ojos, las vías visuales o el cerebro. La evolución y características específicas de la discapacidad visual dependerán de cuándo se inició en la vida del individuo.

3.2.1.1 Niveles de discapacidad visual

Se encuentran dos tipos diferentes de discapacidad visual las cuales se clasifican en diferentes niveles, estos tipos son la deficiencia visual y la ceguera, estos tipos abarcan los niveles de la perdida de la visión hasta la ceguera total. Según (Saucedo et al., 2016) Los niveles son:

- Baja visión: Este nivel abarca a aquellas personas que logran tener una distancia de lectura de moderada de 20/61 y 20/41, después de la corrección de un ojo en donde puede realizar actividades con el apoyo de lentes y ver como una persona normal.
- Legalmente ciego y funcionalmente ciego: En cuanto a estos dos niveles se obtiene una distancia de lectura de entre 20/76 y 20/122 severa en donde en la clasificación se establece como severa ya que la persona tiene más dificultad para realizar una tarea con precisión. Lo que quiere decir que al individuo le tomara más tiempo en ejecutarla.
- Ceguera total: El último nivel se obtiene una distancia de lectura de 20/305 la cual se clasifica como profunda, en esta el individuo tiene la discapacidad de realizar las actividades visuales cotidianas de una persona normal.

3.2.1.2 Rehabilitación y terapia visual

La rehabilitación en las personas con discapacidad visual es fundamental para que estas obtengan una mejor calidad de vida de acuerdo con sus necesidades, esta abarca desde personas con baja visión hasta la ceguera total. Cuando se va a iniciar una rehabilitación se debe tener en cuenta la evaluación realizada al paciente para determinar qué tipo de rehabilitación necesita según sus necesidades, ya que se debe de identificar en qué nivel de pérdida visual esta y las habilidades que puede hacer con ella. Cuando ya se obtiene esta información, se desarrolla un plan de acción en donde procede a enseñarle al paciente técnicas de movilidad y orientación para que pueda llegar a movilizarse fácilmente en el entorno donde puede hacer uso de tecnologías de apoyo como el bastón, dispositivos especiales, perros guías, etc. Se enfoca en desarrollarle al paciente habilidades que puedan ayudarle a adaptarse a la vida independiente, en la participación en la sociedad y mejorar su autonomía.

En la rehabilitación se realiza diversos procedimientos en donde se deben tomar en cuenta ciertos aspectos como el reconocimiento de patrones, localización y evitación de obstáculos, identificación y captación del espacio cercano, identificación y alcance en el espacio locomotor, localización de puntos de referencia, todo este aprendizaje es proporcionando por un profesional especializado en el área.

3.2.2 LA MARCHA HUMANA

El ciclo de marcha representa una secuencia fundamental de movimientos físicos que se llevan a cabo durante el acto de caminar, siendo de suma importancia para la movilidad humana. Este ciclo se compone de varias etapas claramente definidas (Ilustración 1), cada una de las cuales desempeña un papel específico en la coordinación del movimiento de las extremidades inferiores. En primer lugar, encontramos el "contacto inicial" o "heel strike," que se produce cuando el talón de un pie entra en contacto con el suelo, preparando tanto el tobillo como la cadera para soportar la carga del cuerpo (Buisson Garcia, 2023). A continuación, tenemos la "respuesta de carga" o "Toe strike," que marca el momento en que todo el peso del cuerpo se transfiere a la pierna en apoyo y el pie se prepara para despegarse (Buisson Garcia, 2023).

La fase denominada "apoyo plano" comprende el período desde que los dedos de los pies hacen contacto con el suelo hasta el inicio de la elevación del talón, durante el cual el peso del

cuerpo se desplaza hacia adelante sobre el pie que está en apoyo (Buisson Garcia, 2023). Después, viene la "pre-oscilación," que señala el instante en que la pierna que está en el aire alcanza la posición vertical y se dispone para iniciar el movimiento de oscilación. Esta fase culmina con el "Toe Off," el momento en que el talón deja de estar en contacto con el suelo, dando paso a la fase de oscilación (Buisson Garcia, 2023).

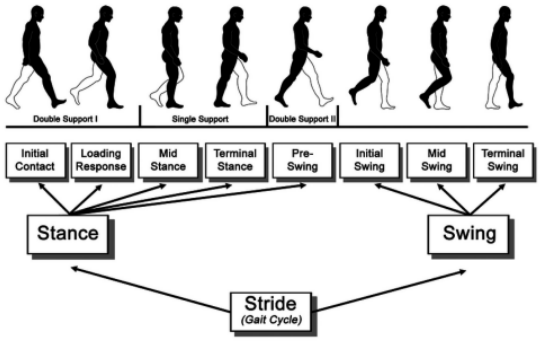


Ilustración 1: Fases del ciclo de marcha

Fuente: (Cicirelli et al., 2022)

3.2.2.1 Características de la marcha en PDV

Las personas con discapacidad visual suelen tener pasos más cautelosos, cortos y controlados por el hecho de que quieren evitar caídas en entornos desconocidos. Esto genera una reducción en los rangos de movimiento, afectando negativamente su funcionalidad y generando estrés muscular. Por esa razón hacen uso de dispositivos de apoyo y de la adaptación del equilibrio dinámico reducido.

"Es importante la valoración de la caminata en las PDV para que se desarrollen patrones adecuados y no presenten desbalances o acortamientos musculares. Logrando una mejora de su locomoción"(alemán-Ramírez, 2020).

Así mismo, estas personas utilizan más su demás sentido como el sentido del tacto o el de la audición para obtener información de su ubicación como los obstáculos, recorridos, reconocimiento de paredes y muebles cercanos.

Utilizan más fuerzas de frenado y conducción que las personas sin discapacidad. La longitud de la zancada se acorta. Realizan control predictivo en respuesta a cambios en el entorno. Existen diferencias en la velocidad de la marcha, la longitud de la zancada y el ritmo, y los costos

metabólicos son mayores. También hay una mayor flexión del tronco y la cabeza, un movimiento mínimo de la cabeza y una inclinación pélvica anterior. (Alemán-Ramírez, 2020).

3.2.3 PARÁMETROS DE MARCHA

En el estudio de la marcha, se consideran parámetros tanto espaciales como temporales para comprender en profundidad este proceso. Las medidas espaciales abarcan la "longitud de zancada" y la "longitud de paso," las cuales nos indican cuánto avanza una persona con cada paso individual y la distancia total recorrida durante un ciclo de marcha, respectivamente (Sanchis, 2019). Por otro lado, las medidas temporales incluyen la "duración de las fases," que representa el tiempo necesario para completar cada etapa del ciclo de marcha, la "cadencia," que refleja el número de pasos por unidad de tiempo, y la "velocidad de la marcha," que cuantifica la distancia recorrida en la misma unidad de tiempo (Sanchis, 2019). Además, se puede analizar la "velocidad de los segmentos de los pies," que combina aspectos espaciales y temporales al relacionar la longitud de los pasos con el tiempo que requieren en cada fase del ciclo de marcha, proporcionando una visión detallada de la eficacia de los movimientos de los pies durante la caminata. Estas mediciones son esenciales en la evaluación de la marcha y pueden ser fundamentales en el contexto de la rehabilitación y la investigación biomecánica.

3.2.3.1 Factores que afectan los parámetros de marcha en PDV

La autonomía se puede ver afectada por limitadas habilidades en sus desplazamientos, en la orientación, en la capacidad de subir y bajar gradas, y en la forma y la velocidad de caminar. Ahora bien, centrándonos en este patrón de movimiento, caminar es un elemento esencial para la capacidad de independencia y movilidad, su ausencia o no ejecución influye directamente en los bajos niveles de actividad física (Alemán-Ramírez, 2020).

También el grado de discapacidad visual puede influir en la marcha ya que las personas con baja visión pueden tener menos dificultades para desplazarse. Otro factor que afecta la marcha de los PDV es de que al no caminar de manera ideal esto causa que una baja economía de movimiento resultante de la reducción de la velocidad generada.

Los movimientos articulares generados son de mayor rango que los de una persona sin discapacidad visual como en el tobillo y en cuanto a la pelvis y la cabeza se generan movimientos

más pequeños, dando a indicar que no se genera una adecuada técnica en los movimientos causando así que se realicen de forma forzada, provocando mayor estrés muscular y óseo.

3.2.3.2 Métodos de evaluación de la marcha

Existe una amplia variedad de métodos para analizar la marcha, que posibilitan obtener de manera objetiva los parámetros cuantitativos característicos. Es importante considerar diversos factores durante el análisis para realizar una evaluación adecuada. Algunos de estos factores deben ser tenidos en cuenta desde la etapa de configuración y planificación del laboratorio, mientras que otros son relevantes en el proceso de adquisición de los parámetros o en el análisis de los resultados obtenidos.

La obtención de los parámetros espaciales se encuentra técnicas como el uso de plataformas de fuerza para registrar las fuerzas aplicadas, sistemas de captura de movimiento que hace uso de cámaras infrarrojas para capturar los movimientos, cintas métricas y marcadores anatómicos para medir diferentes distancias entre distintos puntos de referencias y las unidades de medición inercial (IMUs) que permiten evaluar de manera objetiva los problemas de movimiento y marcha, así como evaluar un gran número de pasos. La fotogrametría es otra técnica que "se utiliza para rastrear la posición tridimensional de un conjunto de puntos fiduciales constituidos por marcadores retro reflectantes (pasivos) o emisores de luz (activos)"(Baqersad et al., 2017)

En la obtención de los parámetros espacio temporales se puede hacer uso de la medición óptica que permitirá la obtención de frecuencia de zancada, tiempo de contacto, longitud de zancada, entre otros. El uso de sistema de fotogrametría también puede ser útil para la toma de imágenes con el objetivo de adquirir el posicionamiento de puntos anatómicos y realizar un análisis tridimensional. Sistemas de acelerometría, presurometría y dinamografía son otro tipo de técnicas para la obtención de parámetros espacio temporales.

Las técnicas para obtener los parámetros temporales para el análisis de la marcha humana son similares a las técnicas que se utilizan en la obtención de los demás parámetros mencionados como la utilización de unidades de medición inercial, plataformas de fuerza, acelerómetros, análisis de video a través de grabaciones de videos que seguidamente de enviaran a un software

para realizar la evaluación y analizar el ciclo de la marcha. Los softwares que se pueden utilizar para esta técnica son el kinovea, Dartfish, Vicon Nexus, entre otros.

3.2.4 SENSORES INERCIALES

Los sensores inerciales son dispositivos versátiles que tienen la capacidad de medir múltiples aspectos relacionados con el movimiento de un objeto en movimiento, incluyendo la aceleración lineal, velocidad angular y orientación espacial (Martinez Méndez & Romero-Huertas, 2018).

Dentro de esta categoría, el "acelerómetro" desempeña un papel fundamental al medir la aceleración lineal en tres ejes (X, Y y Z) al detectar la fuerza que actúa sobre una pequeña masa dentro del sensor. Esto permite determinar la aceleración experimentada por el objeto y se relaciona con los principios de la segunda ley del movimiento de Newton (Collin et al., 2018).

Por otro lado, el "giroscopio" se utiliza para medir la velocidad angular y la tasa de cambio del ángulo de rotación en los tres ejes. Su funcionamiento se basa en detectar la fuerza de Coriolis generada cuando un objeto en rotación se desplaza perpendicularmente a su eje de rotación. Esto se convierte en una señal eléctrica que refleja la velocidad angular del objeto, siendo crucial para medir la orientación y estabilidad (Collin et al., 2018).

El "magnetómetro," por último, mide la orientación del objeto en relación con el campo magnético terrestre utilizando principios como la magneto resistencia o el efecto Hall. Estos fenómenos aprovechan las propiedades de materiales magneto sensibles para detectar campos magnéticos y tienen aplicaciones en la navegación y geolocalización (Paci et al., 2022). Estos sensores inerciales son esenciales en diversas aplicaciones, desde sistemas de navegación hasta dispositivos de seguimiento de actividad física.

3.2.5 INTELIGENCIA ARTIFICIAL

La inteligencia artificial "es la "disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico"(Buisson Garcia, 2023).La ha experimentado un crecimiento significativo en las últimas décadas, permitiéndole emular procesos de aprendizaje humanos a través de algoritmos y sistemas flexibles. Este avance tecnológico ha tenido un profundo impacto en nuestra vida diaria, evidenciado por la creciente presencia de programas inteligentes en

dispositivos móviles, motores de búsqueda en Internet y diversas aplicaciones de software que analizan el comportamiento humano. Además, la IA se ha convertido en una herramienta crucial en una amplia variedad de campos, incluyendo física, química, economía, ciencias ambientales, automatización, ciencias sociales, biología y medicina, como lo demuestra la abundante literatura científica publicada en estas disciplinas (Vorobioff et al., 2022).

La IA se puede subdividir en dos diferentes ramas, el aprendizaje automático y el aprendizaje profundo (Ilustración 2). El aprendizaje automático es una disciplina en donde se emplean algoritmos basados en reglas matemáticas que posibilitan que las computadoras adquieran conocimiento de manera similar al proceso de aprendizaje humano mientras que en el aprendizaje profundo se enfoca en la creación de representaciones abstractas de muestras u objetos a un nivel de abstracción superior. Este campo pertenece a la categoría del aprendizaje automático y, en su mayoría, implica la utilización de redes neuronales altamente complejas (Vorobioff et al., 2022).

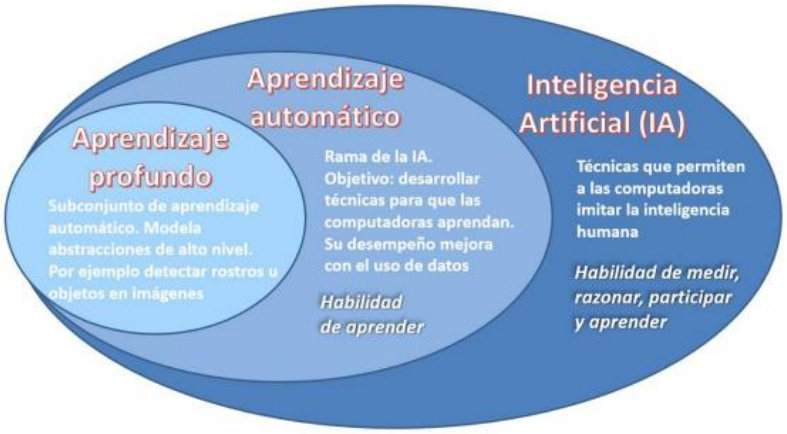


Ilustración 2 - Ramas de IA

Fuente: (Vorobioff et al., 2022)

3.2.5.1 Machine Learning

El aprendizaje automático se define ampliamente como un conjunto de métodos computacionales que hacen uso de experiencias previas para mejorar su desempeño o realizar pronósticos precisos. En este contexto, la experiencia se refiere a información pasada disponible para el aprendiz, la cual generalmente toma la forma de datos electrónicos recopilados y puestos a disposición para su análisis. Estos datos pueden ser conjuntos de entrenamiento digitalizados

con etiquetas creadas por humanos o diferentes tipos de información obtenida a través de interacciones con el entorno. En todos los casos, la calidad y cantidad de estos datos juegan un papel fundamental en el éxito de las predicciones realizadas por el aprendizaje automático(Mohri et al., 2018).

3.2.5.2 Deep learning

En términos generales, las redes neuronales profundas son aquellas que tienen muchas capas y una gran cantidad de neuronas, a menudo organizadas de una manera que no está específicamente diseñada para un dominio en particular. Gracias a la disponibilidad de capacidad de cálculo y grandes conjuntos de datos, estas estructuras de gran envergadura han demostrado ser altamente eficaces en el aprendizaje de características ocultas y patrones de datos. (Rebala, G et al., 2019)

3.3 TEORÍAS DE SUSTENTO

En esta sección, se exploran diversas teorías y conceptos clave relacionados con la marcha, los sensores inerciales y el aprendizaje profundo, específicamente en el contexto de las redes neuronales LSTM. Se examina cómo las propiedades de memoria a largo plazo y la capacidad de capturar dependencias temporales hacen que los algoritmos LSTM sean idóneos para modelar secuencias de datos complejas, como los patrones de movimiento en la marcha humana. Además, se analiza la relevancia de las redes neuronales LSTM en la predicción de secuencias de tiempo, lo que proporciona una base sólida para su aplicación en la estimación precisa de parámetros de marcha en personas ciegas. Mediante el análisis crítico de estas teorías y su relación con el objetivo de la investigación, esta sección establecerá una justificación teórica fundamentada para el uso de algoritmos LSTM y proporcionará una comprensión profunda de cómo esta metodología contribuye significativamente a la evaluación de la marcha en individuos con discapacidad visual.

3.3.1 BASES TEÓRICAS

3.3.1.1 Cálculo de marcha de personas con discapacidad visual

El cálculo de los parámetros de marcha en personas ciegas es un proceso fundamental para comprender y mejorar la movilidad de esta población. La discapacidad visual impone desafíos únicos en el ciclo de marcha debido a la falta de información visual sobre el entorno circundante. Para medir y analizar con precisión estos parámetros, se utilizan diversas

herramientas y técnicas, como sistemas de captura de movimiento y sensores de movimiento (Ilustración 3), que permiten registrar y cuantificar los movimientos del cuerpo durante la marcha. Además, en el contexto de la discapacidad visual, el bastón se convierte en una extensión esencial del individuo, proporcionando información táctil y kinestésica sobre el terreno y los obstáculos cercanos. El uso adecuado del bastón, junto con el análisis detallado de los parámetros de marcha, juega un papel crucial en el desarrollo de estrategias de entrenamiento y adaptación que promuevan una movilidad segura y una mayor independencia en la vida diaria de las personas ciegas.

La HH y SZ son medidas utilizadas para evaluar la distancia de reacción en la O&M de personas con discapacidad visual. La distancia de reacción se refiere al espacio de advertencia que el bastón proporciona al detectar un objeto o peligro en el camino. La HH se refiere a la altura a la que una persona sostiene su bastón mientras camina, y la SZ es el espacio alrededor de la persona que es cubierto por el bastón al moverse. Estos parámetros son importantes para medir el tiempo que una persona tiene para reaccionar y evitar posibles obstáculos o peligros mientras se desplaza utilizando el bastón como una herramienta esencial para obtener información táctil del entorno.

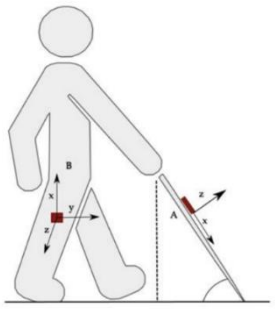


Ilustración 3 - Sistema local de coordenadas del sensor colocado en el bastón (A) y sistema local de coordenadas del sensor colocado en la pierna (B)

Fuente: (Reyes Leiva, Jaén-Vargas, Cuba, et al., 2021)

En términos de cobertura, una marcha adecuada es crucial para el desarrollo de las habilidades de O&M, por lo tanto, durante el entrenamiento de O&M, la velocidad de la marcha y la longitud de la zancada están siendo constantemente evaluadas visualmente por el rehabilitador.

3.3.1.2 Principios de los Sensores Inerciales

Los sensores inerciales se basan en los principios de la física inercial. Estos sensores suelen estar integrados en unidades de medición inercial (IMU), que incluyen varios sensores diferentes, como giroscopios, acelerómetros y magnetómetros. Estos sensores recopilan datos que pueden combinarse para obtener mediciones precisas del movimiento (Andrenacci et al., 2021).

La posición de la masa cambia y esto genera una señal eléctrica proporcional a la aceleración del objeto. Cuando el instrumento experimenta aceleración a lo largo de su eje sensible, la masa de prueba se desplaza con respecto a la caja del instrumento. En situaciones estables, la fuerza que actúa sobre la masa se equilibra con la tensión del muelle. La extensión o contracción del muelle genera una fuerza proporcional al desplazamiento. Cuando no hay fuerza de arrastre que dificulte el movimiento de la masa de prueba, su desplazamiento es directamente proporcional a la aceleración. Por lo tanto, midiendo el desplazamiento de la masa de prueba se puede medir la aceleración aplicada.

El giroscopio es capaz de medir la velocidad de rotación inducida por cambios en la actitud del objeto con respecto al espacio inercial. Este sensor aprovecha las propiedades inerciales de una rueda que gira a alta velocidad y tiende a mantener la dirección de su eje de rotación debido a los principios de conservación del momento angular. Aunque la orientación del eje no permanece fija, cambia en respuesta a un par externo mucho menor y en una dirección diferente a la que lo haría sin el gran momento angular asociado a la alta velocidad de giro y al momento de inercia del disco. Para minimizar el efecto de este par externo, los dispositivos giroscópicos se montan en cardanes, lo que permite que su orientación permanezca prácticamente invariable, independientemente de cualquier movimiento de la plataforma en la que estén instalados. (Collin et al., 2018)

En el caso del magneto resistencia, se utiliza un material que experimenta cambios en su resistencia eléctrica cuando se expone a un campo magnético. Estos cambios se deben a la alteración de la estructura cristalina del material bajo la influencia del campo magnético. Al medir los cambios en la resistencia eléctrica, es posible determinar la intensidad del campo magnético. Por otro lado, el efecto Hall se basa en el principio de que cuando un conductor por el que circula una corriente eléctrica se expone a un campo magnético perpendicular a la dirección de la

corriente, se genera una tensión eléctrica perpendicular tanto a la corriente como al campo magnético. Esta tensión eléctrica, conocida como voltaje Hall, es proporcional a la intensidad del campo magnético (Paci et al., 2022). El magnetómetro se utiliza para compensar las lecturas del acelerómetro y el giroscopio, mejorando la precisión de la medición de la orientación.

Al tener estos componentes integrados en la IMU, este se puede conectar a un microcontrolador o a otros dispositivos para procesar y utilizar los datos recopilados. Para obtener mediciones precisas, es común utilizar algoritmos de fusión sensorial, como el filtro de Kalman, que combinan los datos de los diferentes sensores y compensan sus limitaciones individuales, como errores y derivas, para proporcionar estimaciones más precisas de la orientación, la aceleración lineal y la velocidad angular del objeto.

3.3.1.3 Aplicaciones de sensores inerciales en la medición de la marcha

Los sensores inerciales son ampliamente reconocidos por su capacidad para suministrar información precisa sobre la dirección y la distancia recorrida en diversos contextos. En el ámbito de la estimación de la longitud del paso, existen dos enfoques principales. La primera categoría se fundamenta en modelos biomecánicos que se valen de principios biomecánicos para realizar dicha estimación. Por otro lado, la segunda categoría se basa en establecer relaciones empíricas entre el patrón de la señal de aceleración y la longitud del paso.

En los modelos biomecánicos, además de los parámetros de escala determinados empíricamente, se requiere considerar ciertos parámetros relacionados con el usuario, como la longitud de las piernas. La detección de pasos puede realizarse de forma sencilla mediante el análisis del patrón de señal del componente de aceleración vertical. Sin embargo, este enfoque resulta sensible a los errores de orientación del sensor, ya que se asume que un eje está alineado con la vertical o que se conoce la transformación hacia la vertical. Otra opción consiste en calcular la magnitud del vector de aceleración medido, es decir, la norma de aceleración. El patrón de la señal varía según la ubicación en la que el usuario coloque el sensor (Collin et al., 2018). En la detección de pasos, es común basarse en la identificación de picos de señal o cruces de la señal con su media o algún otro nivel de referencia. Frecuentemente, los algoritmos de detección combinan la detección de picos y cruces de nivel de referencia.

Los sensores inerciales son ampliamente utilizados en la medición de la marcha humana. En el contexto de la marcha, los sensores inerciales se colocan en diferentes partes del cuerpo, como los pies, las piernas o la cintura, y se utilizan para capturar datos biomecánicos durante el proceso de caminar.

"Los métodos tradicionales de modelización biomecánica desarrollados a partir de datos de marcadores de captura de movimiento tridimensionales (3D) de referencia requieren que el usuario establezca un modelo anatómico subyacente." (Mundt et al., 2021)

Para ello, utilizando sensores IMU, es necesario determinar la orientación del sensor con respecto a la orientación del segmento en un sistema de referencia global (alineación sensor-segmento). Para ello pueden adoptarse diversos métodos, todos ellos dependientes de algoritmos de fusión de sensores y de la determinación de la alineación inicial sensor-segmento mediante la aplicación de posturas de calibración o movimientos funcionales que el participante debe ejecutar.

3.3.1.4 Redes neuronales artificiales

En el aprendizaje automático e inteligencia artificial, los modelos de redes neuronales, como las redes LSTM y las redes neuronales convolucionales utilizan estos sistemas para predecir el procesamiento del lenguaje natural y la visión por computadora.

Para entender de mejor manera cuales son los elementos básicos que componen una red neuronal es esencial saber el funcionamiento de una neurona. Las neuronas tienen tres componentes principales, las cuales son denominadas dendritas, el cuerpo de la célula o soma y el axón. El punto de conexión entre el axón de una célula y una dendrita de otra célula se llama sinapsis. En términos computacionales, las dendritas, son las receptoras de la red, que cargan de señales eléctricas el cuerpo de la célula. El cuerpo de la célula realiza la suma de esas señales de entrada. El axón es una fibra larga que lleva la señal desde el cuerpo de la célula hasta otras neuronas (Ilustración 4). (Acevedo, E. et al., 2017)

Ilustración 4 - Comparación entre una neurona biológica (izquierda) y una artificial (derecha)

Fuente: (Vorobioff et al., 2022)

Las redes neuronales ofrecen una serie de ventajas significativas en el ámbito de la inteligencia artificial y el procesamiento de datos.

- Tolerancia a fallos: Al experimentar daños parciales la red tiene la habilidad de mantenerse, lo que no altera significativamente la respuesta total del sistema.
- Operación en tiempo real: Los cómputos neuronales pueden realizarse en paralelo utilizando máquinas diseñadas y construidas con hardware especializado.
- Son sistemas distribuidos no lineales: Debido a que una neurona en sí misma es no lineal, la interconexión entre ellas resultará en que el dispositivo también será no lineal.
- Adaptabilidad: Tiene la capacidad de modificar los parámetros y adquirir destrezas en la ejecución de tareas a partir de un proceso de formación o de una experiencia inicial.

Las funciones de activación desempeñan un papel crucial en el aprendizaje y la comprensión de una red neuronal artificial cuando se trata de tareas complejas y mapeos funcionales no lineales entre las entradas y la variable de respuesta. Estas funciones agregan propiedades no lineales a nuestra red, y su función principal es transformar una señal de entrada de un nodo en una RNA en una señal de salida. Esta señal de salida se emplea luego como entrada en la siguiente capa de la estructura.

- Función de Activación

  Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado. La función activación calcula el estado de actividad de una neurona; transformando la entrada global (menos el

umbral, Θi) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1). Esto es así, porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1). (Matich, 2001)

- Función de entrada

Al tratar múltiples valores de entrada como una sola entidad, la neurona se conoce como entrada global. Por lo tanto, el desafío actual es combinar estas simples entradas (in_{i1}, in_{i2},...) dentro de la entrada capa de entrada global, $g_i n_i$. Esto se logra a través de la función de entrada, la cual se calcula a partir del vector entrada. La función de entrada puede describirse como sigue (Ecuación 1):

$$input_i = (in_{i1} \bullet w_{i1}) * (in_{i2} \bullet w_{i2}) * \ldots (in_{in} \bullet w_{in})$$

Ecuación 1 - Función de entrada

Fuente: (Matich, 2001)

Dónde:

- ➢ representa al operador apropiado (por ejemplo: máximo, sumatoria, productora, etc.)
- ➢ n al número de entradas a la neurona N_i y W_i al peso.

Los valores de entrada se multiplican por los pesos que se le dieron previamente a la neurona. Las ponderaciones, que normalmente no tienen restricciones, determinan en qué medida se ven afectados los valores de entrada. Estos pesos, cuando son pequeños, pueden tener un impacto significativo en la influencia de implementar valores arbitrarios para diferentes valores de entrada (Matich, 2001).

- Función de salida

La función de salida es el último factor importante de una neurona. La salida de la neurona i está determinada por esta función, que a su vez determina el valor enviado a sus neuronas asociadas. En caso de que la función de activación no alcance un nivel predeterminado, la salida se transmite a la neurona siguiente. Es común que la salida de las neuronas esté restringida a [0, 1] o [1, 1], ya que los datos de entrada no siempre están disponibles. Por ejemplo, pueden tomar valores binarios como 0, 1 o incluso menos de un decimal (Matich, 2001).

Las funciones de activación se pueden dividir en 2 tipos:

- Función de Activación Lineal

Una función de activación lineal es una función matemática utilizada en redes neuronales que produce una salida proporcional a la entrada (Ilustración 5). En otras palabras, si la entrada se multiplica por un valor específico, la salida también se multiplica por ese mismo valor (Ali & Muhammad, 2019).

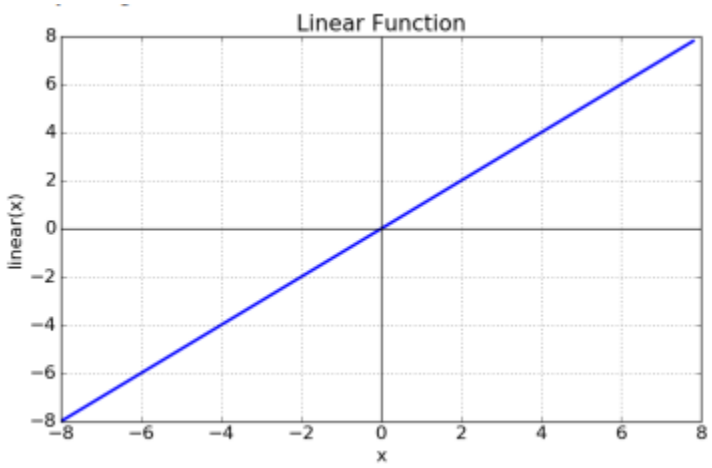


Ilustración 5 - Función de Activación Lineal

Fuente: (Ali & Muhammad, 2019)

- Función de Activación No Lineales

Las funciones de activación no lineales son funciones matemáticas utilizadas en redes neuronales que introducen no linealidad en el modelo (Ilustración 6). Esto significa que la salida de la función no es simplemente una combinación lineal de las entradas, lo que permite a la red modelar relaciones más complejas en los datos valor (Ali & Muhammad, 2019).

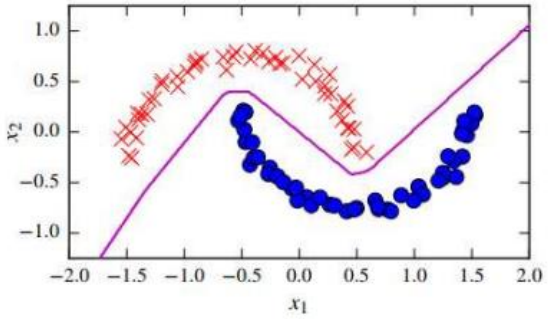


Ilustración 6 - Función de Activación No Lineales

Fuente: (Ali & Muhammad, 2019)

- Diferentes tipos de Funciones de Activación No Lineales

La función de activación sigmoidea es una función matemática utilizada en redes neuronales y otros modelos de aprendizaje automático (Ilustración 7). Esta función toma un valor de entrada y produce una salida en el rango de 0 a 1 valor (Matich, 2001).

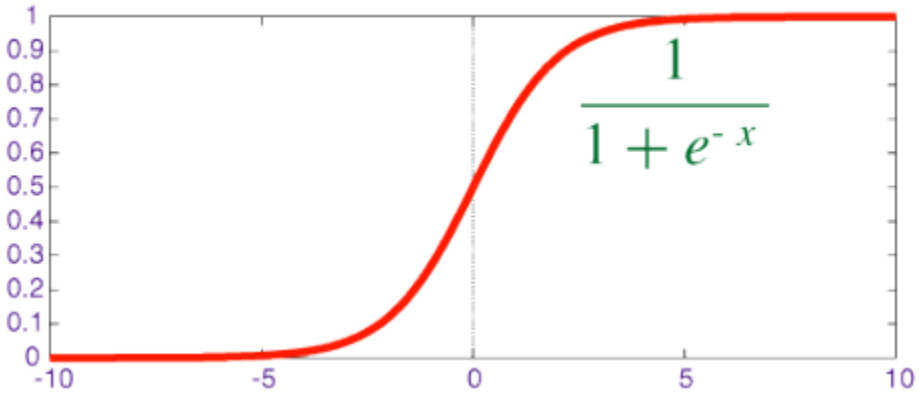


Ilustración 7 - Función de activación sigmoidea

Fuente: (Ali & Muhammad, 2019)

La función de activación tanh es una función matemática utilizada en redes neuronales y otros modelos de aprendizaje automático (Ilustración 8). Esta función toma un valor de entrada y produce una salida en el rango de -1 a 1 valor (Ali & Muhammad, 2019).

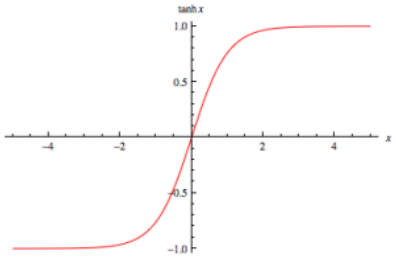


Ilustración 8 - Función de activación tangente hiperbólica

Fuente: (Ali & Muhammad, 2019)

La función de activación ReLU es una función matemática utilizada en redes neuronales y otros modelos de aprendizaje automático (Ilustración 9). Su característica principal es que produce una salida igual a la entrada si la entrada es mayor que cero, y produce una salida igual a cero si la entrada es menor o igual a cero valor (Ali & Muhammad, 2019).

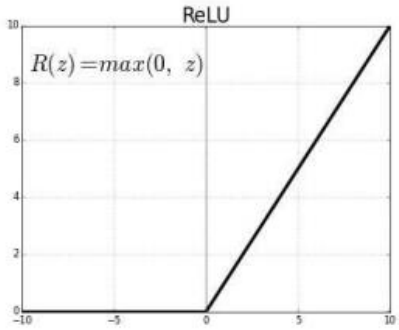


Ilustración 9 - Función de activación ReLU

Fuente: (Ali & Muhammad, 2019)

- ➢ Clasificación de redes neuronales
- ✓ Según Arquitectura:

Las neuronas están organizadas en redes en las que generalmente se distribuyen en varias capas (Ilustración 10). En esta estructura, las neuronas de una capa están conectadas a las neuronas de la capa siguiente, permitiéndoles transmitir información entre sí. Cada neurona dentro de esta red funciona como una unidad de procesamiento de información que recibe datos a través de las conexiones que tiene con las neuronas de la capa anterior.

- Capa de Entrada: Es quien recibe información del exterior. En las redes biológicas, esta sería tarea de las dendritas.
- Capas ocultas: La cuáles están encargadas de realizar el trabajo de la red. En las redes biológicas, está sería el soma.
- Capa de Salida: Proporciona el resultado del trabajo de la red al exterior y envía información hacia otras neuronas. En las redes biológicas, esta sería una actividad realizada por el axón.

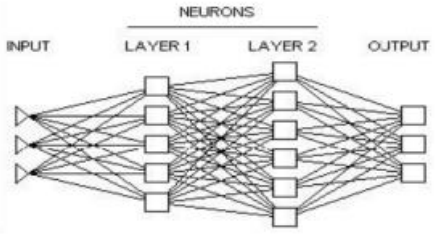


Ilustración 10 - Capas de una red neuronal

Fuente: (Acevedo, E. et al., 2017)

- Red neuronal mono capa

En las redes neuronales de una sola capa, las conexiones se establecen exclusivamente entre las neuronas dentro de esa única capa (Ilustración 11). Por lo general, las redes de una sola capa se emplean en contextos relacionados con la auto asociación, que implica la capacidad de regenerar información de entrada que se presenta de manera incompleta o distorsionada (Matich, 2001).

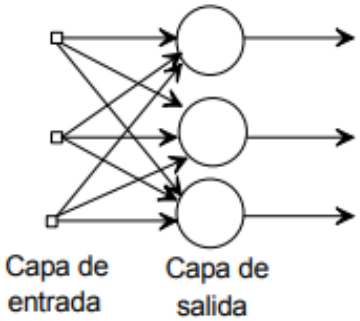


Ilustración 11 - Esquema de una red mono capa

Fuente: (Serrano et al., 2009)

- Red neuronal Multicapa

Las redes neuronales multicapa son aquellas que incorporan un conjunto de neuronas organizadas en múltiples niveles o capas, que pueden ser dos, tres o más (Ilustración 12). En estas situaciones, una forma de distinguir a qué capa pertenece una neurona es observar de dónde provienen las señales de entrada que recibe y hacia dónde se dirigen las señales de salida que emite (Matich, 2001).

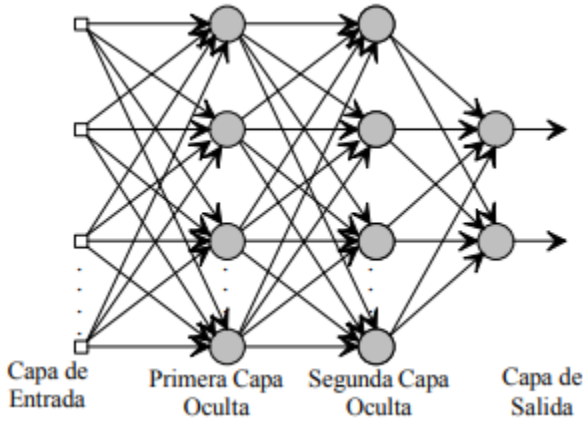


Ilustración 12 - Esquema de red neuronal multicapa

Fuente: (Serrano et al., 2009)

- Conexión entre neuronas

Las redes no recurrentes se refieren a la propagación de las señales que se producen en un solo sentido, donde no existe la posibilidad de la retroalimentación por lo tanto no tienen memoria (Serrano et al., 2009). Dentro de estas conexiones se encuentra la conexión hacia delante o feedforward la cual indica que las neuronas reciben señales de entrada procedentes de una capa anterior (que se encuentra más cerca de la entrada de la red) y transmiten señales de salida hacia una capa posterior (que está más próxima a la salida de la red). También está la conexión hacia atrás o feedback la cual conecta la salida de las neuronas de capas posteriores a la entrada de capas anteriores (Matich, 2001).

Las redes recurrentes sin embargo estas si obtienen una retroalimentación (Ilustración 13) Estos lazos de retroalimentación provienen de neuronas entre diferentes capas o entre la misma neurona por ende la hace ideal para el estudio de la dinámica de sistemas no lineales(Serrano et al., 2009). Estas redes disponen de conexiones tanto hacia adelante como hacia atrás o redes feedforward/feedback (Matich, 2001). En las redes multicapa se pueden encontrar conexiones como la feedforward o feedforward/feedback.

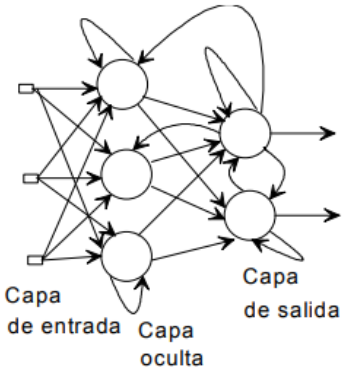


Ilustración 13 - Red recurrente

Fuente: (Serrano et al., 2009)

- ✓ Según su aplicación:
- ➢ Clasificación

"Se refiere al problema de identificar la categoría a la que pertenece una entrada entre un conjunto posible de categorías. El conjunto posible de categorías está etiquetado, y generalmente se aprenden modelos a partir de datos de entrenamiento" (Rebala, G et al., 2019).

La clasificación es un algoritmo de aprendizaje supervisado en el que se dispone de un conjunto de entrenamiento de datos correctamente identificados o etiquetados. El modelo aprendido a partir de los datos de entrenamiento para identificar la categoría o clase de la característica o datos de entrada se llama clasificador. El clasificador puede ser un clasificador binario o un clasificador multiclase. Un clasificador binario identifica la entrada como perteneciente a una de las dos categorías de salida. Por ejemplo, determina si el correo recibido es spam o no spam (Rebala, G et al., 2019).

- Regresión

Es una técnica estadística utilizada para analizar y modelar la relación entre una variable dependiente que es una variable predicha y una o más variables independientes que son las variables predictivas.

El tipo de análisis de regresión que debe utilizarse en una situación determinada depende principalmente de los tres parámetros siguientes:

- Número y naturaleza de la(s) variable(s) independiente(s)
- Número y naturaleza de la/s variable/s dependiente/s
- Forma de la línea de regresión

"La regresión lineal es una técnica común para relacionar dos variables continuas en un modelo lineal, que puede ser simple (una variable independiente) o múltiple (varias variables independientes). Es útil para analizar relaciones lineales entre datos cuantitativos"(NJ Gogtay et al., 2017).

"La regresión lineal múltiple se utiliza una variable dependiente continua y dos o más variables independientes. Las variables independientes pueden ser cuantitativas o cualitativas. Las dos variables independientes podrían expresarse como datos continuos o cualitativos. Deberá existir una relación lineal entre las variables dependientes e independientes"(NJ Gogtay et al., 2017). El proceso de análisis de regresión implica tres etapas distintas:

- Examina la correlación, que revela la fuerza y dirección de las relaciones entre los datos.
- Realiza el ajuste de la línea de regresión utilizando el método de los mínimos cuadrados.
- Evaluar la validez y utilidad del modelo resultante.

Según (C.M. Saliba et al., 2018):

Un enfoque de pronóstico híbrido utilizando la regresión lineal múltiple en combinación con el análisis de componentes principales. Este método permitió prever las fuerzas de contacto de la articulación de la rodilla durante la marcha, utilizando medidas cinemáticas y cinéticas de cada individuo. El análisis del movimiento se integró con un modelo estadístico de regresión, que podría ejecutarse casi en tiempo real. La regresión lineal múltiple se aplicó para calcular la puntuación de cada componente principal de las fuerzas de contacto medial y lateral, teniendo en cuenta el peso corporal del sujeto y las características principales de las formas de onda cinemáticas y cinéticas, que incluyeron un total de 34 componentes. (Caldas R. et al., 2020)

De acuerdo con lo mencionado los métodos de regresión llegan a ser beneficiosos al momento de predecir los parámetros de la marcha humana ya que brinda un modelo matemático que describe la relación entre variables independientes como dependientes y al obtener un modelo ajustado se pueden obtener predicciones precisas con una generalización de resultados donde se puede aplicar en diferentes situaciones al momento de predecir la marcha.

- ✓ Según el método de aprendizaje:

El proceso de aprendizaje en las redes neuronales artificiales es un proceso secuencial en el que la red adquiere conocimiento a medida que experimenta diferentes situaciones. El conjunto de datos utilizado para entrenar una red neuronal artificial se caracteriza por tener dos cualidades esenciales: debe ser significativo y representativo (Ilustración 14). Para que el aprendizaje sea efectivo, es necesario contar con un conjunto de datos que contenga un número suficiente de ejemplos, y estos ejemplos deben ser variados y equilibrados en términos de su diversidad.

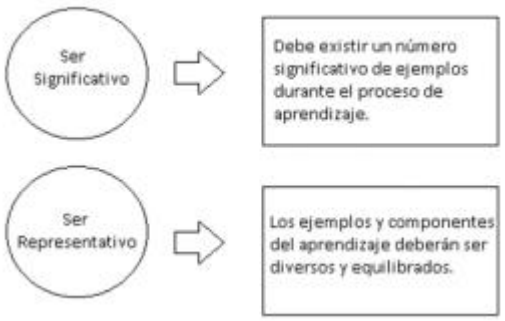


Ilustración 14 - Características del conjunto de aprendizaje de una red neuronal artificial

Fuente: (Acevedo, E. et al., 2017)

Existen diferentes categorías de aprendizaje en las redes neuronales, y su proceso de aprendizaje se fundamenta en un algoritmo específico que varía según el tipo de aprendizaje que se aplique. Entre ellas se encuentra:

- Aprendizaje supervisado: "Se realiza un entrenamiento de la red neuronal que estará supervisado y controlado por el diseñador de esta, para determinar que la respuesta de la red sea una específica dependiendo de la entrada"(Acevedo, E. et al., 2017).
- Aprendizaje no-supervisado: "Este tipo de aprendizaje muestra un proceso auto organización hasta cierto grado. La red neuronal descubre con los datos de entrada las características, regularidades, correlaciones y categorías, y lo hace de una forma autónoma"(Acevedo, E. et al., 2017).

Existen varios tipos de redes neuronales artificiales, cada una diseñada para abordar problemas específicos y aplicaciones en el campo del aprendizaje profundo y la inteligencia artificial. Entre ellas se enfocará en dos tipos:

- Redes neuronales recurrentes

"Las Redes Neuronales Recurrentes son modelos de redes neuronales diseñados especialmente para procesar datos secuenciales. Los datos secuenciales pueden encontrarse en varios dominios de problemas, como datos de series temporales o simplemente una secuencia de datos que presenta propiedades repetitivas" (Rebala, G et al., 2019). Sin embargo, una RNN simple no es capaz de capturar las dependencias a largo plazo y conducirá a fallos en el modelado.

La arquitectura de las redes neuronales recurrentes simplifica y optimiza la construcción de estas redes neuronales. Como su nombre sugiere, la estructura de la tarea se repite varias veces durante el cálculo, y la definición de la tarea se realiza solo una vez. De manera intuitiva, esto implica que las entradas anteriores se almacenan en la memoria con valores ponderados y se combinan con la entrada actual para realizar predicciones (Rebala, G et al., 2019).

Las LSTM provenientes de las RNN surgieron para abordar uno de los desafíos más críticos de las RNN: el modelado de dependencias a largo plazo. A diferencia de las RNN tradicionales, las LSTM incorporan una estructura de memoria sofisticada y compuertas que les permiten capturar y retener información relevante a lo largo del tiempo en secuencias de datos, superando así el problema de los gradientes que desaparecen (Ilustración 15).

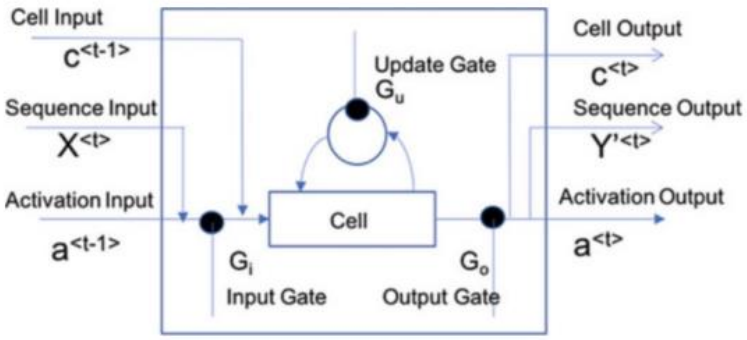


Ilustración 15 - Esquema de una red LSTM

Fuente: (Rebala, G et al., 2019)

Las redes neuronales recurrentes, como las redes de memoria a largo plazo (LSTM), se diseñaron para problemas de predicción de secuencias. Aprovechan las dependencias temporales de los datos, lo que explica su éxito en el procesamiento del lenguaje natural. Por tanto, también son adecuadas para tareas de predicción de secuencias de movimiento.

Desgraciadamente, las redes LSTM son intensivas de entrenar y requieren grandes conjuntos de datos, algo raramente disponible en biomecánica. Las redes LSTM también se han utilizado anteriormente para predecir ángulos y momentos articulares a partir de datos de sensores inerciales (Mudt et al., 2021).

Como una LSTM aprende dependencias temporales, es necesario mantener esta dimensión en los datos de entrada y salida, lo que da como resultado una matriz tridimensional de entrada y salida de tamaño [número de muestras × número de características de entrada/salida × pasos temporales]. Si se desea una predicción de ángulos y momentos articulares en tiempo real es mejor utilizar una red LSTM (Mundt et al., 2021).

En particular, las redes LSTM han demostrado ser especialmente eficaces para modelar secuencias de datos, como el ciclo de marcha en personas con discapacidad visual. A través del análisis de secuencias temporales, las redes LSTM pueden capturar relaciones complejas y dependencias a largo plazo, lo que las convierte en una elección ideal para predecir parámetros como el rendimiento en la marcha y la detección de obstáculos en entornos cambiantes. Además, CNN y DL también juegan un papel significativo en esta área, al permitir la extracción automática de características relevantes de datos visuales y la comprensión de información compleja para mejorar la movilidad y la autonomía de las personas con discapacidad visual.

- Redes Convolucionales

Las CNN se diseñaron originalmente para asignar datos de imágenes a una única variable de salida, una tarea de clasificación (Ilustración 16). Aprenden de los datos brutos de la imagen explotando las correlaciones entre píxeles locales. Funcionan especialmente bien en datos con una relación espacial, y debido al hecho de que también se puede encontrar una relación ordenada en los datos de series temporales, esto hace que las CNN sean adecuadas para la predicción de series temporales del movimiento humano. Las CNN se han utilizado con entradas de datos de sensores inerciales para predecir la cinemática y la cinética de las articulaciones. Anteriormente se han entrenado diferentes modelos de libre acceso en grandes conjuntos de datos para la clasificación de imágenes, lo que permite utilizar el aprendizaje por transferencia o el ajuste fino de un modelo en lugar de entrenar una CNN desde cero. Para poder utilizar modelos entrenados en grandes bases de datos de imágenes y aplicar el aprendizaje por transferencia en lugar de entrenar desde cero, Johnson et al. transformaron secuencias de tiempo de movimiento en imágenes para la predicción de momentos tridimensionales de la articulación de la rodilla y secuencias de fuerza de reacción del suelo basadas en entradas de datos de captura de movimiento.

Las CNN se utilizan principalmente para clasificar datos de imágenes. Para adaptarlas a nuestro propósito, las secuencias de movimiento se transformaron en imágenes RGB en las que cada canal mostraba una dirección de las señales del sensor. La matriz resultante se interpola para ajustarla al tamaño de las imágenes utilizadas anteriormente.

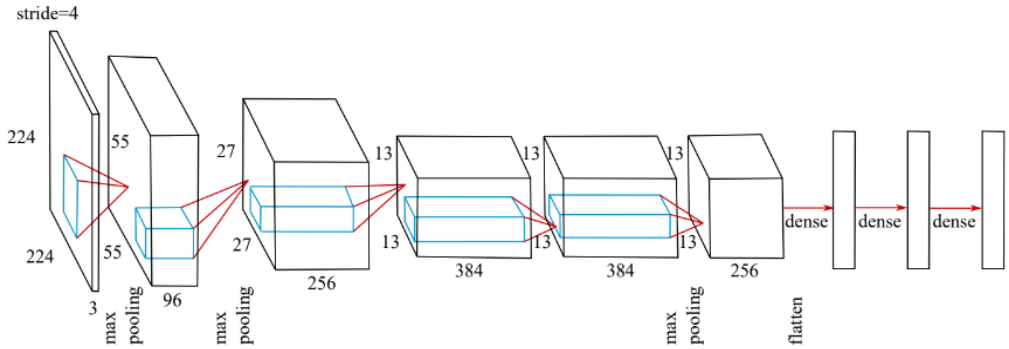


Ilustración 16 - Ejemplo arquitectura CNN

Fuente: (Mundt et al., 2021)

- Método Kernel en CNN

En el contexto de regresión, las funciones kernel permiten modelar relaciones no lineales entre variables y realizar predicciones más precisas en situaciones en las que los modelos lineales tradicionales pueden no ser adecuados. Al igual que en la clasificación, las funciones kernel elevan el espacio de características original a dimensiones superiores, lo que facilita la captura de patrones y relaciones complejas en los datos de regresión. Esto los convierte en una herramienta valiosa para resolver una amplia gama de problemas de regresión en el aprendizaje automático (Amor Perea, J., 2018).

Los problemas de clasificación implican la tarea de entrenar un algoritmo utilizando muestras previas que se conocen por su clase de pertenencia (Ilustración 17). A partir de esta información y las características de las muestras anteriores, el algoritmo debe ser capaz de clasificar las muestras objetivo en una de las categorías predefinidas. Esto implica que la salida del algoritmo es discreta. El objetivo principal en estos casos es encontrar un hiperplano que tenga la capacidad de separar los datos en dos clases distintas (Amor Perea, J., 2018).

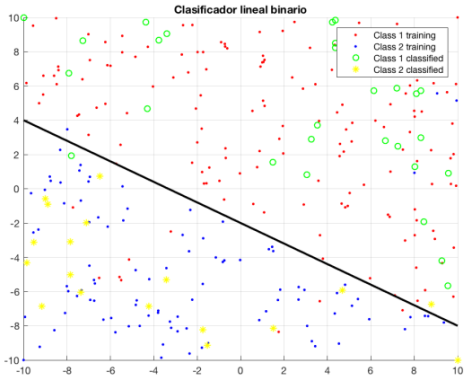


Ilustración 17 - Ejemplo de problema de clasificación con el método kernel

Fuente: (Amor Perea, J., 2018)

A diferencia de los problemas de clasificación, los problemas de regresión la salida no es discreta, y el objetivo es pronosticar un valor numérico en lugar de asignar una categoría (Ilustración 18). Un ejemplo podría ser predecir la temperatura que alcanzará un sistema en un momento específico en lugar de clasificarlo en una categoría particular. Este tipo de problemas caen dentro de la categoría del aprendizaje automático supervisado (Amor Perea, J., 2018).

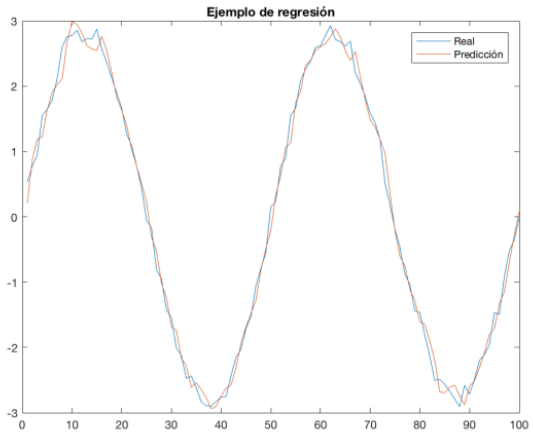


Ilustración 18 Ejemplo de problema de regresión con el método kernel

Fuente: (Amor Perea, J., 2018)

3.3.1.5 Deep learning con Python

Son relativamente escasas las herramientas de software de código abierto y uso general para el procesamiento de diversas actividades de la vida diaria. Además, pocos estudios incluyen código para replicación o paquetes de software listos para usar. En su trabajo, (Adamowicz et al., 2022) presenta SKDH, un paquete de software Python que contiene varios algoritmos para derivar características clínicas de la marcha, la bipedestación, la actividad física y el sueño, envueltos en un marco fácilmente extensible.

Python cuenta con bibliotecas y frameworks especializados en el desarrollo de redes neuronales y aprendizaje profundo, como TensorFlow, Keras, PyTorch y scikit-learn. Estas herramientas hacen que sea más fácil y rápido construir y entrenar modelos de redes neuronales para el análisis del ciclo de marcha.

- Implementación práctica de redes neuronales en Python

La forma en que una red neuronal calcula una salida es un proceso que implica codificación de datos. En otras palabras, se requiere encontrar valores adecuados para representar las características simbólicas, como "alto", "bajo" o "adecuado". Este proceso de codificación es fundamental para que la red pueda comprender y procesar la información de manera efectiva. Se explora cómo esta codificación desempeña un papel esencial en el funcionamiento de las redes neuronales y su capacidad para realizar tareas de aprendizaje automático. Se distinguen dos tipos de variables a ser codificadas:

- Variables o atributos numéricos (frecuentemente llamadas continuas).

Los datos son codificados dentro de un intervalo, [0.0 + buffer de baja, 1.0 – buffer de alta], por medio de una función lineal. Los buffers (amortiguadores) son necesarios, especialmente cuando se trabaja con series de tiempo, porque a menudo puede observarse que una variable numérica cae por debajo del valor mínimo presenciado hasta el momento, o por encima del máximo. Por medio de esta manera de codificación se conduce a un conjunto de valores por encima de 0.0 y por debajo de 1.0, cuando se utiliza un salto de 0.0 a 1.0 (Matich, 2001).

- Variables o atributos simbólicos (frecuentemente llamados discretos).

Cada atributo simbólico que se codifica se adjunta a una neurona en la capa de entrada. Si hay n valores simbólicos, n neuronas serán necesarias, cada una de ellas con un conjunto de entradas permitido: {0, 1} (o {-1, 1}). Por este motivo, se utilizan neuronas binarias (Matich, 2001).

Luego de completar el proceso de entrenamiento, los pesos de las conexiones en la red neuronal se mantienen invariables. El siguiente paso implica verificar si la red neuronal puede resolver nuevos problemas de naturaleza general, para los cuales ha sido previamente entrenada. Para lograr esto, es necesario utilizar otro conjunto de datos conocido como conjunto de validación o conjunto de pruebas (Matich, 2001).

Cada ejemplo presente en el conjunto de validación incluye los valores de las variables de entrada, pero no se proporciona la solución correspondiente a la red neuronal en este caso. En su lugar, la red neuronal debe calcular la solución por sí misma para cada ejemplo de validación. Posteriormente, se compara esta solución calculada con la solución conocida para determinar la eficacia y precisión de la red en la resolución de nuevos problemas (Matich, 2001).

a) Preprocesamiento de Datos

1. Importar bibliotecas: Entre ellas se encuentra la biblioteca numpy, que se utiliza para arreglos multidimensionales. Luego, importamos la biblioteca panda, que se utiliza para importar el conjunto de datos, y finalmente, importamos la biblioteca matplotlib, que se utiliza para graficar los gráficos.
2. Carga de Datos: El proceso comienza con la carga de los datos brutos desde una fuente, como un archivo CSV o una base de datos. Esto implica importar los datos en una estructura que sea manejable para su procesamiento posterior.

3. Escalado de Características: Algunos modelos son sensibles a la escala de las características. En esta etapa, se puede aplicar el escalado de características para asegurarse de que todas las características tengan un rango similar.
4. División del Conjunto de Datos en Conjuntos de Entrenamiento y Prueba: Este paso implica dividir el conjunto de datos en dos subconjuntos: uno destinado al entrenamiento del modelo y otro para probar su rendimiento. Esta división es esencial para evaluar qué tan bien se desempeña el modelo en datos que no ha visto durante el entrenamiento.

b) Construcción de la red

1. Definición de la Arquitectura de la Red: Se decide la arquitectura de la red, incluyendo el número de capas ocultas, el número de neuronas en cada capa, la función de activación y la arquitectura de entrada y salida.
2. Creación del Modelo: Se utiliza una biblioteca de aprendizaje profundo como TensorFlow, Keras o PyTorch para crear el modelo. Se puede crear una instancia de un modelo secuencial o personalizar la arquitectura según las necesidades.
3. Compilación del Modelo: Se configura el modelo especificando la función de pérdida, el optimizador y las métricas que se utilizarán para evaluar el rendimiento del modelo durante el entrenamiento.
4. Entrenamiento del Modelo: Se alimenta el modelo con datos de entrenamiento y ajusta los pesos de las conexiones entre neuronas mediante el proceso de retro propagación. Esto se hace durante un número fijo de épocas o hasta que se alcance un criterio de detención.

- Entrenamiento de la red neuronal

Normalmente, el protocolo de entrenamiento de una RNA se basa en minimizar una función de pérdida definida sobre la salida deseada de los datos y la salida real de la RNA mediante la actualización de los parámetros. Los enfoques clásicos suelen ajustar los parámetros basándose en las derivadas de la función de pérdida. Sin embargo, gran parte de la potencia de las RNA procede de la función no lineal en las unidades ocultas utilizada para modelar el mapeo no lineal entre la entrada y la salida. Por desgracia, este tipo de arquitectura pierde la elegancia

de encontrar la solución mínima global con respecto a todos los parámetros de la red, ya que la función de pérdida depende de la salida de las neuronas no lineales (Ilustración 19).

"Se ha demostrado que los métodos de entrenamiento basados en la aleatorización pueden aumentar significativamente el rendimiento o la eficiencia de las redes neuronales. Entre estos métodos, la mayoría utilizan la aleatorización para cambiar la distribución de los datos y/o para fijar una parte de los parámetros o configuraciones de la red." (Zhang & Suganthan, 2016).

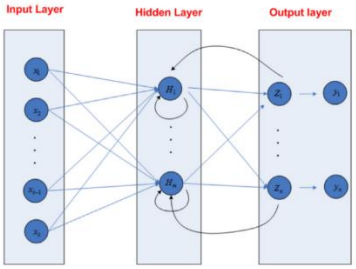


Ilustración 19 - Estructura de RNN

Fuente: (Zhang & Suganthan, 2016)

Una RNN aleatoria es un conjunto de N neuronas totalmente conectadas como se muestra en la Ilustración 19. Los pesos que conectan las neuronas se inicializan aleatoriamente. Los estados de las neuronas son X(t) = xi(t), i = 1, . . ., N, donde cada xi se establece proporcional a la frecuencia de disparo de la neurona ith.

En el aprendizaje profundo, las estructuras profundas con múltiples capas de operaciones no lineales son capaces de aprender abstracciones de alto nivel. Esta abstracción de alto nivel es el factor clave del éxito de muchos sistemas de última generación en visión, lenguaje y otras tareas de IA. Los algoritmos de entrenamiento complejos combinados con parámetros cuidadosamente elegidos (es decir, velocidad de aprendizaje, tamaño del mini lote, número de épocas) pueden dar lugar a una red neuronal profunda (DNN) de alto rendimiento.

El autocodificador puede descomponerse en dos partes: codificador y decodificador. Un codificador es un mapeo determinista que mapea la entrada x a una representación oculta y mediante: f(x) = s (Wx + b) donde = {W, b}, y s es una función de activación no lineal como la sigmoidea. En el decodificador, la representación oculta se vuelve a mapear para reconstruir la entrada x. Este mapeo se consigue mediante g(y) = s (W y + b). La Fig. 4 muestra la estructura de un autocodificador de eliminación de ruido. En un autocodificador de eliminación de ruido, en

primer lugar, la entrada se corrompe por algún ruido aleatorio y el autocodificador intenta reconstruir la entrada "limpia".

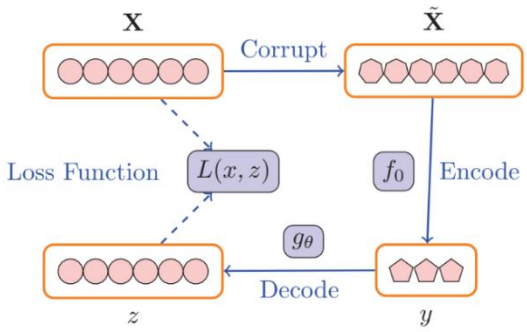


Ilustración 20 - Estructura básica de un Autocodificador de eliminación de ruido

Fuente: (Zhang & Suganthan, 2016)

Las CNN combinan tres ideas arquitectónicas para garantizar cierto grado de invariabilidad de desplazamiento, escala y distorsión: pesos compartidos, submuestreo y campos receptivos locales.

La CNN se compone de capas convolucionales y de pooling alternadas. Las ideas de "pesos compartidos" y "campos receptivos locales" están implicadas en las capas convolucionales. Un filtro de un tamaño predefinido (por ejemplo, 5 × 5 o 7 × 7) se convoluciona con la entrada para obtener el mapa de características. Para aumentar el rendimiento de una CNN, se suele dejar que la red aprenda un conjunto "sobrecompleto" de mapas de características. Podemos apilar fácilmente esta arquitectura en arquitecturas profundas estableciendo que la salida de una capa de agrupación sea la entrada de otra capa convolucional. En este caso, las entradas de cada filtro de la capa superior pueden conectarse aleatoriamente a la salida de la capa de agrupación inferior.

5. Evaluación del Modelo: Se evalúa el rendimiento del modelo utilizando un conjunto de pruebas separado. Calcula métricas como la precisión, la pérdida o el error según la naturaleza del problema.

Luego de completar el proceso de entrenamiento, los pesos de las conexiones en la red neuronal se mantienen invariables. El siguiente paso implica verificar si la red neuronal puede resolver nuevos problemas de naturaleza general, para los cuales ha sido previamente entrenada. Para lograr esto, es necesario utilizar otro conjunto de datos conocido como conjunto de validación o conjunto de pruebas (Matich, 2001).

Cada ejemplo presente en el conjunto de validación incluye los valores de las variables de entrada, pero no se proporciona la solución correspondiente a la red neuronal en este caso. En su lugar, la red neuronal debe calcular la solución por sí misma para cada ejemplo de validación. Posteriormente, se compara esta solución calculada con la solución conocida para determinar la eficacia y precisión de la red en la resolución de nuevos problemas (Matich, 2001).

3.3.1.6 Métricas de evaluación

Las métricas de error cuadrático medio y error cuadrático absoluto son dos medidas comúnmente utilizadas para evaluar el rendimiento de modelos de aprendizaje automático en tareas de predicción. Estas métricas son particularmente útiles cuando se trata de problemas de regresión, donde el objetivo es predecir un valor numérico, en lugar de una clasificación.

En el contexto específico de predecir el análisis de marcha de personas con discapacidad visual, el objetivo es obtener predicciones numéricas (por ejemplo, mediciones relacionadas con la marcha) y compararlas con los valores reales observados en el conjunto de datos de prueba.

- ✓ Error cuadrático medio

El ECM es una métrica que calcula el promedio de los errores al cuadrado entre las predicciones y los valores reales. La fórmula para el ECM es (Ecuación 2):

$$ECM = \Sigma (y\_pred - y\_real)^2 / n$$

Ecuación 2 - Error cuadrático medio

Fuente: (Su & Gutierrez-Farewik, 2020)

Donde:

- ➢ y_pred es el valor predicho por el modelo.
- ➢ y_real es el valor real en el conjunto de datos de prueba.
- ➢ n es el número de muestras en el conjunto de datos de prueba.

El ECM penaliza más los errores más grandes, ya que los errores se elevan al cuadrado. En este caso, un ECM más bajo indica que el modelo tiene una mejor precisión en las predicciones, es decir, se acerca más a los valores reales.

- ✓ Error cuadrático absoluto

El ECA es otra métrica de evaluación que calcula el promedio de los errores absolutos entre las predicciones y los valores reales. La fórmula para el ECA es (Ecuación 3):

$$ECA = \Sigma |y\_pred - y\_real| / n$$

Ecuación 3 - Error cuadrático absoluto

Fuente: (Ribeiro et al., 2020)

Donde:

- |x| representa el valor absoluto de x.

A diferencia del MSE, el MAE no eleva los errores al cuadrado, lo que significa que no penaliza tanto los errores más grandes. Al igual que el MSE, un MAE más bajo indica una mejor precisión del modelo en las predicciones.

Ambas métricas son útiles para evaluar el rendimiento de un algoritmo LSTM+CNN en la predicción del análisis de marcha de personas con discapacidad visual. Es importante considerar que la elección de la métrica puede depender del contexto específico del problema y las preferencias del equipo de investigación o desarrolladores del modelo. Generalmente, se recomienda utilizar varias métricas de evaluación para tener una comprensión completa del rendimiento del modelo.

3.3.1.7 Problema de la generalización del modelo y el sobreajuste

El propósito de una red neuronal es obtener un modelo final que demuestre un rendimiento satisfactorio tanto en los datos que se utilizaron para su entrenamiento (como el conjunto de entrenamiento) como en los datos nuevos en los que se empleará el modelo para realizar predicciones.

Es crucial que el modelo aprenda el conocimiento de ejemplos pasados y luego lo aplique a ejemplos futuros. Para evaluar la capacidad del modelo para generalizar a datos previamente inexplorados, se emplean métodos como dividir los datos en conjuntos de entrenamiento y prueba. Puede resultar difícil lograr un equilibrio entre el aprendizaje y la generalización durante este proceso. Un ajuste insuficiente o un rendimiento deficiente del modelo tanto en el conjunto de entrenamiento como en los datos nuevos es una señal de aprendizaje insuficiente. Cuando se aprende demasiado, un fenómeno conocido como sobreajuste, el modelo tiene dificultades con los datos nuevos a pesar de tener un buen desempeño en el conjunto de entrenamiento. El modelo no se habrá generalizado efectivamente en ninguna de las situaciones (Brownlee, J., 2018).

- Modelo con Subajuste. Un modelo que no logra aprender adecuadamente el problema y tiene un rendimiento deficiente en un conjunto de entrenamiento, así como un mal rendimiento en un conjunto de validación.
- Modelo con Sobreajuste. Un modelo que aprende el conjunto de entrenamiento en exceso tiene un buen rendimiento en el conjunto de entrenamiento, pero no tiene un buen rendimiento en un conjunto de validación.
- Modelo con Buen Ajuste. Un modelo que aprende de manera adecuada el conjunto de entrenamiento y generaliza de manera efectiva en el conjunto de validación.

El ajuste de un modelo se puede entender en el contexto del equilibrio entre sesgo y varianza. Un modelo con subajuste muestra un alto sesgo y una baja varianza. En otras palabras, no importa cuáles sean las muestras específicas en los datos de entrenamiento, no puede aprender adecuadamente el problema en cuestión. Por otro lado, un modelo con sobreajuste tiene un bajo sesgo y una alta varianza. Esto significa que el modelo aprende de manera exhaustiva los datos de entrenamiento, pero su rendimiento puede variar ampliamente cuando se enfrenta a nuevos ejemplos que no ha visto antes o incluso cuando se introduce ruido estadístico en los ejemplos del conjunto de entrenamiento (Brownlee, J., 2018).

3.3.1.8 Integración de sensores inerciales con redes neuronales en la predicción de parámetros de marcha

Las RNA aprenden las conexiones entre los datos de entrada (datos de la IMU) y los datos de destino (por ejemplo, cinemática/cinética articular). Para el entrenamiento, los conjuntos de datos deben contener datos de la IMU (entradas) y datos de captura de movimiento de marcadores 3D (objetivos), de modo que los datos objetivos para el modelo de RNA estén anatómicamente estandarizados, un requisito para los modelos biomecánicos de referencia. Posteriormente, el modelo entrenado puede utilizarse para predecir los parámetros del objetivo, por lo que sólo se necesitan sensores inerciales. Por otro lado, los modelos de RNA sólo pueden predecir las relaciones que han aprendido durante el proceso de entrenamiento. Esto significa que, si una RNA se ha entrenado con datos de la IMU recogidos con una alineación sensor-segmento específica, sólo predecirá la cinemática/cinética articular exacta cuando se replique la alineación en los nuevos datos. Para superar este problema, es necesaria una gran variación en la

alineación sensor-segmento de la IMU en el conjunto de datos de entrenamiento. Además del requisito de un conjunto de datos adecuado, la elección del modelo de RNA es importante. (Mundt et al., 2021)

Los modelos de DL son muy útiles para el reconocimiento de la actividad humana, estos métodos presentan una mayor precisión para el HAR en comparación con los tradicionales, entre otras ventajas. DL aprende a partir de datos no etiquetados y extrae características de datos sin procesar, como en el caso de la aceleración de series temporales. Las ventanas deslizantes son una técnica de extracción de características.

Cuando se utiliza para pre procesar datos de series temporales, mejora la precisión, la latencia y el coste del procesamiento. El tiempo y el coste del pre procesamiento pueden ser beneficiosos especialmente si el tamaño de la ventana es pequeño, pero ¿cómo de pequeña puede ser esta ventana para mantener una buena precisión? El objetivo de esta investigación era analizar el rendimiento de cuatro modelos de DL: DNN, CNN; una red de memoria a corto plazo larga (LSTM); y un modelo híbrido (CNN-LSTM), al variar el tamaño de la ventana deslizante utilizando ventanas superpuestas fijas para identificar un tamaño de ventana óptimo para HAR. Comparamos los efectos en dos fuentes de aceleración: sensores portátiles de IMU y MOCAP.

En cuanto al tiempo de inferencia, los datos con una ventana deslizante de 20 fotogramas pueden pre procesarse unas 4 veces (LSTM) y 2 veces (CNN-LSTM) más rápido que los datos que utilizan 100 fotogramas.

Una técnica común para abordar el reconocimiento de actividad humana con datos de sensores es el uso de ventanas deslizantes. Esto implica dividir las secuencias de datos de aceleración en ventanas más pequeñas y luego clasificar cada ventana individualmente. El tamaño de la ventana y el grado de traslape entre ventanas pueden afectar el rendimiento del modelo.

Las redes neuronales LSTM sobresalen en el tratamiento de datos que comprenden longitudes de secuencia arbitrarias. Con el fin de garantizar que las RNA probadas recibieran exactamente los mismos datos en este estudio, se requirió que los datos de entrada se normalizaran en el tiempo, a pesar de que se ha demostrado previamente que esto conduce a una subestimación de las capacidades de predicción de una red LSTM.

Para comparar las predicciones de las distintas RNA (Mundt et al., 2021), analizaron el error cuadrático medio normalizado al rango de los datos RMSE y el coeficiente de correlación. La evaluación del RMSE en lugar del error cuadrático medio permite la comparación entre predicciones cinemáticas y cinéticas, además de comparaciones directas entre diferentes planos de movimiento. Dado que la marcha comprende pequeños rangos de movimiento en los planos de movimiento no sagitales, pequeñas desviaciones en la predicción del ángulo articular pueden dar lugar a grandes valores RMSE.

3.3.1.9 Implementación de ventanas deslizantes

El método más comúnmente empleado en la etapa de segmentación del Reconocimiento Automático de Patrones (RAP) es conocido como el enfoque de la ventana deslizante. En este método, las señales del sensor se dividen en segmentos de tamaño fijo. Si estos segmentos se superponen entre sí, se denomina ventana deslizante con superposición; de lo contrario, se le llama técnica de ventanas no superpuestas (Dehghani et al., 2019).

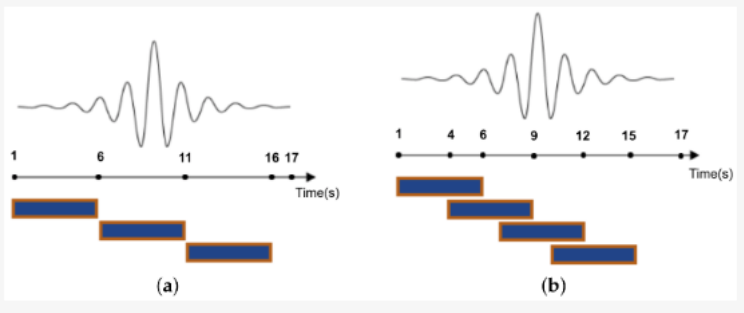


Ilustración 21 - Representación gráfica de ambas técnicas, la de ventanas no superpuestas y la de ventanas deslizantes con superposición

Fuente: (Dehghani et al., 2019)

Las ventanas deslizantes se pueden crear utilizando una de dos técnicas principales: ya sea trabajando con las muestras de la secuencia o dividiéndola en segmentos con un intervalo de tiempo de segundos. Además, las ventanas se pueden dividir en dos grupos: fijas y adaptables; y con o sin superposición (Ilustración 21). Las ventanas se clasifican en fijas si mantienen el mismo tamaño durante toda la secuencia, y en adaptativas si, cuando se produce un movimiento, su tamaño cambia de acuerdo con unos criterios predeterminados. Las ventanas también se superponen cuando la ventana anterior contiene una parte de la secuencia anterior, es

decir, cuando hay una superposición entre dos ventanas sucesivas. Por el contrario, se las conoce como ventanas no superpuestas (Jaén-Vargas et al., 2022).

Una ventana deslizante se compone de tres elementos esenciales: muestras, el tamaño de ventana (también conocido como número de pasos de tiempo) y las características (entradas) (Ilustración 22). En este contexto, una secuencia representa una muestra más amplia, que puede incluir una o varias muestras individuales. El tamaño de ventana, que corresponde a los pasos de tiempo, representa una observación dentro de la secuencia y puede abarcar uno o varios fotogramas. Cabe destacar que, cuando la ventana se desplaza a lo largo de las muestras, se inicia una nueva fase (Jaén-Vargas et al., 2022).

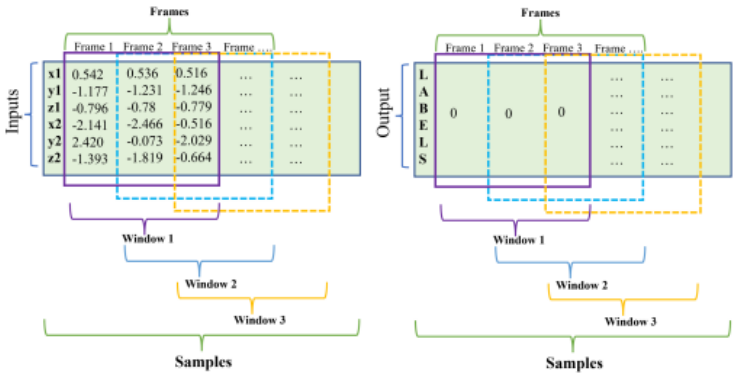


Ilustración 22 - Esquema de ventana deslizante

Fuente: (Jaén-Vargas et al., 2022)

El tamaño adecuado de la ventana deslizante y la aplicación de modelos de aprendizaje profundo afectan los costos de procesamiento de manera significativa, así como la capacidad de reconocer la actividad humana. Para conseguir métricas de alto rendimiento, es decir, buena capacidad de predicción es importante tener en cuenta que no es necesario utilizar grandes ventanas deslizantes. Esto se debe al hecho de que aumentaría la capacidad de respuesta y el costo de procesamiento de cualquier aplicación creada utilizando el modelo entrenado. Por este motivo, es fundamental elegir un tamaño de ventana que sea apropiado y utilizar algoritmos de aprendizaje profundo para examinar las secuencias que contienen cada sistema en uso para producir una predicción final de alta calidad (Jaén-Vargas et al., 2022).

3.3.2 METODOLOGÍAS DESARROLLADAS POR OTROS INVESTIGADORES

Se realizó una revisión de literatura donde se exploraron las diversas técnicas y enfoques utilizados en estudios previos sobre el análisis de la marcha y como estos fueron implementados haciendo uso de diversas tecnologías como el uso de sensores inerciales y redes neuronales que han permitido obtener una comprensión más profunda de los patrones de movimiento involucrados en la locomoción humana.

3.3.2.1 Desarrollo de un método para evaluar la longitud del paso y parámetros de la marcha mediante sensores inerciales y ángulos de orientación absolutos

(Reyes Leiva, Jaén-Vargas, Cuba, et al., 2021) propusieron tres métodos basados en IMU para mejorar la medición de parámetros clave de la marcha, como la longitud del paso, la velocidad, el recuento de pasos y el desplazamiento total. Estos parámetros fueron calculados utilizando ángulos de orientación absolutos, y se emplearon tres algoritmos diferentes según el modelo biomecánico seleccionado. Los métodos incluyeron la colocación de un sensor en el muslo o en la pantorrilla, y un tercer método que combinó ambos sensores para obtener mediciones precisas de las rotaciones experimentadas por los segmentos de la pierna en el plano sagital durante el ciclo de la marcha.

Con el sensor inercial colocado en la cara externa de la pierna, con las mismas coordenadas locales que el sensor colocado en el bastón largo, se utilizó el ángulo de cabeceo para calcular la longitud del paso en un ciclo de marcha y dos de las técnicas de desplazamiento. La longitud del paso se calculó en un algoritmo que promediaba la estimación del desplazamiento de la pierna durante el ciclo de marcha siguiendo la diferencia de cada representación pico a pico de los movimientos oscilatorios del ángulo de cabeceo, donde cada pico representa el valor más alto de cada fase en el ciclo de marcha. Por lo tanto, conociendo la longitud de la pierna de cada usuario, y estableciendo constantemente los valores de θlegmax y θlegmin, se podría calcular la longitud del paso mediante la siguiente (Ecuación 4):

$$\mathrm{SL} = 2 \times \sin\left(\frac{\theta_{\mathrm{legmax}} - \theta_{\mathrm{legmin}}}{2}\right) \times \mathrm{LL}$$

Ecuación 4 - Longitud de paso

Fuente: (Reyes Leiva, Jaén-Vargas, Cuba, et al., 2021)

Donde:

- SL es la longitud del paso y LL es la longitud de la pierna del usuario. El algoritmo es capaz de detectar si se está ejecutando un paso con los umbrales θlegmax y θlegmin.

En esta investigación se concluyó que la TWS puede ser más preciso para la evaluación de la marcha de ancianos con discapacidad visual que tienen una velocidad de marcha más lenta y una longitud de paso menor. Como próxima etapa en esta investigación, la validación de los dos métodos seleccionados en diferentes condiciones experimentales con la participación de VIP se debe hacer para determinar cuál es el método más adecuado para la tecnología de asistencia como una herramienta de O&M.

No existe una estandarización en los métodos de formación, y que estos métodos pueden variar en función de la experiencia de cada especialista. Por ello, la investigación en O&M también puede beneficiarse de esta herramienta. El desarrollo de la herramienta presentada por el estudio anteriormente presentado permite evaluar estos parámetros de movilidad independientemente de la complejidad del entorno al que se someta gradualmente el entrenamiento, ya que se trata de un dispositivo portátil de bajo coste.

3.3.2.2 Aplicación de algoritmos de aprendizaje automático para detectores de velocidad cero en la marcha humana y el diagnóstico en la marcha patológica

Entre las distintas investigaciones se puede encontrar el modelo que integra 3 tipos de algoritmos uno de bosque aleatorio, refuerzo de gradiente basado en histograma y red de memoria a largo plazo con el propósito de encontrar detectores de velocidad cero basados en aprendizaje automático (Kone et al., 2020). Otra investigación que mencionar es la creación de un modelo basado en meta clasificadores donde hacen uso de registros cinéticos y técnicas de minería de datos con el objetivo de obtener el diagnóstico en la marcha patológica. (Muñoz & Landínez, 2020)

El sistema puede proporcionar información sobre la rotación de la empuñadura, la zona de seguridad, la altura de la mano durante las técnicas de desplazamiento, la amplitud y los patrones del barrido, y los parámetros de la marcha con una gran precisión utilizando sólo dos sensores inerciales.

Las tecnologías basadas en sensores de IMU se utilizan en un número amplio y creciente de aplicaciones, como el guiado de inteligencia, los robots autos conducidos, el seguimiento del movimiento de todo el cuerpo y la navegación. Un acelerómetro mide la fuerza específica externa que actúa sobre el sensor. Un giroscopio mide la velocidad angular: la tasa de cambio de la orientación del sensor. Así, la integración de las mediciones del giroscopio proporciona información sobre la orientación en el sensor. Los magnetómetros complementan a los acelerómetros proporcionando el rumbo del sensor (orientación alrededor del vector de gravedad), que es una información que los acelerómetros o giroscopios no pueden proporcionar. Con la fusión de acelerómetro, giroscopio y magnetómetros, la orientación se estima a partir de la dirección del campo magnético. En el sistema presentado en este artículo, los parámetros de O&M se calculan utilizando los valores absolutos de orientación de la fusión de sensores que proporciona el módulo IMU BNO055.

Con el uso del ángulo de balanceo de Euler θ pierna y la interpretación de la detección de pasos según los valores de la orientación absoluta filtrada, se desarrolló un algoritmo para calcular la longitud de los pasos utilizando las coordenadas locales del sensor colocado en la pierna. Además, para obtener la métrica de barrido con las coordenadas locales del sensor colocado en el bastón, se utilizaron los ángulos de balanceo de Euler φcane, cabeceo θcane y guiñada γcane para proporcionar consecutivamente la rotación de agarre, la métrica de la zona de seguridad y las características de barrido.

3.3.2.3 Comparación de métodos tradicionales y basados en sensores IMU en la modelización biomecánica de la marcha humana

Según (Mundt et al., 2021), los métodos tradicionales de modelización biomecánica se han desarrollado a partir de datos de captura de movimiento tridimensionales con marcadores de referencia. Estos métodos requieren que el usuario establezca un modelo anatómico subyacente. Al utilizar sensores IMU, es necesario determinar la orientación del sensor en relación con la orientación del segmento en un sistema de referencia global (alineación sensor-segmento). Para lograr esto, se pueden aplicar diversos métodos que dependen de algoritmos de fusión de sensores y de la determinación de la alineación inicial sensor-segmento mediante la ejecución de posturas de calibración o movimientos funcionales por parte del participante.

IV. METODOLOGÍA

El presente capítulo detalla el enfoque de la investigación utilizada para abordar las preguntas y objetivos planteados. Esta sección proporciona una descripción de los materiales, instrumentos, métodos y procedimientos empleados para recopilar, analizar y evaluar los datos pertinentes a la investigación.

4.1 ENFOQUE

Esta investigación se basó en un enfoque cuantitativo con un alcance exploratorio que permitió la medición y análisis numérico de datos aplicando métricas de evaluación de regresión para evaluar dos algoritmos basados en LSTM para la predicción de parámetros de marcha en personas ciegas, por lo tanto, proporciono una visión detallada y precisa del comportamiento del algoritmo.

El tipo de diseño de esta investigación es experimental dado que se buscó determinar el rendimiento y efectividad del algoritmo CNN+LSTM y LSTM+4FCN y llevar a cabo una comparativa con los resultados obtenidos de parte de estos algoritmos basados en LSTM con los valores reales estimados con el método tradicional.

4.2 VARIABLES DE INVESTIGACIÓN

4.2.1 VARIABLE DEPENDIENTE

En esta investigación, tenemos cuatro variables dependientes: tamaño de la zancada, número de pasos, velocidad de marcha y distancia de la marcha (Ilustración 23). Así que es un diseño multicéntrico. Estas cuatro variables están estrechamente relacionadas con los parámetros de la marcha humana en personas con discapacidad visual.

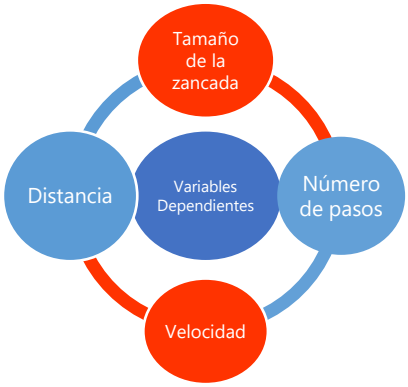


Ilustración 23 - Variables Dependientes

Fuente: Elaboración propia

4.2.2 VARIABLE INDEPENDIENTE

- Modelos: Se refiere a los dos modelos que se utilizaran para el entrenamiento los cuales son LSTM+4FCN y LSTM+CNN.
- Características de los modelos: Se refiere a los parámetros que se estarán variando en los modelos con el fin de mejorar su precisión. Estos son el número de épocas, tamaño de la ventana y tamaño del lote (Ilustración 24).

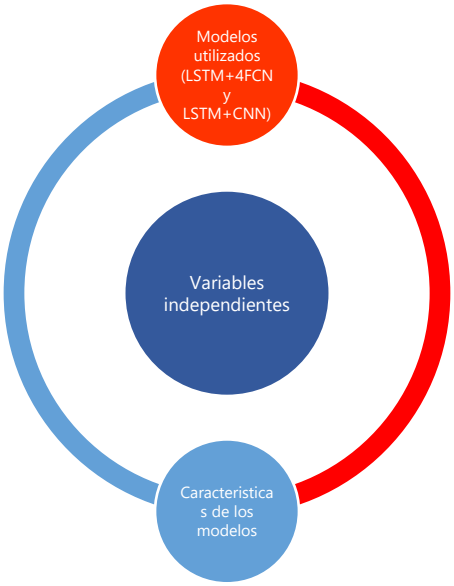


Ilustración 24 - Variable Independiente

Fuente: Elaboración propia

4.3 TÉCNICAS E INSTRUMENTOS APLICADOS

Para la evaluación de los algoritmos basados en LSTM en la predicción de los parámetros de marcha en personas ciegas se utilizaron los siguientes instrumentos con el fin de obtener los resultados a base de entrenamiento que se le realizara al algoritmo y poder realizar la evaluación final.

4.3.1 PYTHON

Python es un lenguaje de programación semi interpretado que compila el código para generar pseudo código máquina que es el propiamente ejecuta el intérprete. Este lenguaje de alto nivel es gratuito, su sintaxis es clara, versátil, y cuenta con características para ser un lenguaje de paradigma funcional.

El uso de Inteligencia artificial, aprendizaje profundo, entre otras implica muchos algoritmos complejos con muchos lenguajes de programación, no obstante, Python requiere poca codificación en términos de líneas de código y puede usarse para desarrollar muchas de estas aplicaciones. Esta funcionalidad se puede probar fácilmente, lo que permite a los desarrolladores centrarse más en la programación real.

"En comparación con otros lenguajes de programación orientados a objetos, Python usa hasta 1/5 del código para implementar la misma lógica, por ende para la investigación será una herramienta de apoyo útil al momento de realizar el entrenamiento del algoritmo LSTM" (Nagpal & Gabrani, 2019).

Las bibliotecas que se usaron en la codificación en Python son las siguientes:

- "TensorFlow se utiliza para entrenar y ejecutar redes neuronales profundas para el PLN, el reconocimiento de imágenes, modelos secuencia a secuencia para traducción automática, redes neuronales recurrentes, etc."(Nagpal & Gabrani, 2019).
- "Keras es una biblioteca a nivel de modelo que proporciona bloques de construcción de alto nivel para desarrollar modelos de aprendizaje profundo. No se encarga de operaciones de bajo nivel como la manipulación de tensores y la diferenciación"(Chollet, 2021).
- "La biblioteca Panda está diseñada para el análisis de datos. Proporciona un mayor rendimiento y estructuras de datos con herramientas accesibles y de fácil uso. Habitualmente se utiliza para cargar datos, preparar datos, manipular datos, modelar y analizar datos" (Nagpal & Gabrani, 2019).
- "La biblioteca Matplotlib es a base de gráficos en 2D con el fin de incrustar diferentes tipos de gráficos en distintas plataformas. Se utiliza en shells IPython, servidores de aplicaciones web, scripts python y Jupyter notebook, y produce cifras de calidad de publicación" (Nagpal & Gabrani, 2019).
- "La biblioteca Numpy consiste en cálculos numéricos y científicos en matrices multidimensionales debido a que mantiene un bien rendimiento del mismo al conformarse de varias funciones y operaciones" (Nagpal & Gabrani, 2019).

4.3.2 GOOGLE COLLAB

"Colaboratory es un producto de Google Research. Colab permite a cualquiera escribir y ejecutar código python arbitrario a través del navegador, y es especialmente adecuado para aprendizaje automático, análisis de datos y educación"(Naik et al., 2022).

Colab nos permitió el fácil acceso y flexibilidad al momento de configurar, almacenar en la unidad la base de datos y desarrollar el código, en donde ya cuenta con librerías preinstaladas de Python como TensorFlow, Scikit-learn, Matplotlib, además que cuenta con aceleración de GPU gratuita que nos ayudó al entrenar los modelos y que estos se ejecutaran de manera más rápida y eficiente.

4.4 MATERIALES

4.4.1 BASE DE DATOS

La base de datos que se utilizó en la investigación fue brindada por Reyes Leiva et al. (2023) la cual proviene de un estudio titulado "Estimation of Spatio-Temporal Parameters of Gait and Posture of Visually Impaired People Using Wearable Sensors" esta nos proporcionó información sobre el análisis de la postura y la marcha en voluntarios ciegos los cuales realizaron una serie de tareas caminando con el bastón en las propias casas o apartamentos (interior) y en las calles (exterior).

> Para la creación de esta base de datos se utilizó 4 sensores inerciales MetaMotionR de MbientLab; uno para valores de orientación-euler de 3 ejes y otro para giroscopio de 3 ejes + valores brutos de aceleración de 3 ejes y 2 sensores inerciales BNO055 de Bosch, montados en la pcb personalizada de donde se tomaron los valores de Aceleración, Giroscopio, Magnetómetro y Orientación-Euler de 3 ejes. (Reyes Leiva et al.,2023)

Se cuenta con la información de 9 sujetos, 5 hombres y 4 mujeres en donde al realizar las diversas tareas se registraron el caminado en diferentes velocidades como lento, regular y Rápido incluido con pausas, cambios de dirección y en diferentes distancias como se puede visualizar en la Tabla 1. Se cuenta con la información obtenida de cada experimento en archivos CSV. Estos archivos contienen parámetros de los sensores inerciales utilizados en cada una de las pruebas en las que se pueden observar los valores del eje 'x', 'y' y 'z' de rodilla y cadera del acelerómetro, giroscopio, magnetómetro y orientación-Euler.

Tabla 1: Información de los experimentos

Experimento	Metros	Descripción	Nombre
Experimento 1	7-10 metros	Velocidad normal desde un punto A hacia un punto B	Regular
Experimento 2	7-10 metros	Velocidad rápida desde un punto A hacia un punto B	Rápido
Experimento 3	7-10 metros	Velocidad lenta desde un punto A hacia un punto B	Lento
Experimento 4	14-20 metros	Velocidad normal con dos paradas desde un punto A hacia un punto B	Regreso
Experimento 5	60-100 metros	Velocidad normal en el exterior desde un punto A hacia un punto B	Externo

Fuente: Elaboración propia

4.5 POBLACIÓN

Para llevar a cabo el presente estudio, se utilizó una base de datos previamente recopilada en una investigación brindada por (Reyes Leiva et al.,2023).

La selección de la muestra se llevó a cabo mediante un enfoque de muestreo por conveniencia. Esta elección se basó en la naturaleza específica y limitada de la población de interés, que comprende personas con discapacidad visual que decidieron participar en el estudio. Dado este contexto, resultaba difícil aplicar una metodología aleatoria tradicional. A pesar de que la muestra por conveniencia puede presentar ciertas limitaciones en términos de generalización de los resultados, ya que la selección está influenciada por la disponibilidad y accesibilidad de los participantes, se consideró adecuada para el propósito de este estudio.

4.6 METODOLOGÍA DE ESTUDIO

Se definió que la metodología de estudio fuera de alcance exploratorio. En este estudio metodológico, la atención se centró en explorar el potencial del uso de dos algoritmos con base en LSTM para predecir los parámetros de la marcha en individuos ciegos. Se utilizarán dos modelos, un 4FCN+LSTM y un CNN+LSTM los cuales se pueden visualizar en los anexos 150 y

142. El proceso de evaluación y mejora del algoritmo consta de varias etapas fundamentales, que se describen a continuación:

4.6.1 Revisión de base de datos

Se analizó la información proveniente de la investigación previa (Reyes Leiva et al., 2023). La base de datos contiene datos recopilados de nueve voluntarios con diferentes niveles de discapacidad visual mediante sensores inerciales. En este proyecto, se compararon los datos obtenidos de la investigación, que se encontraban en archivos CSV, con los videos grabados por los propios investigadores. Esta metodología permitió seleccionar los sujetos y ensayos relevantes para nuestro propio proyecto. En el anexo 14 se pueden visualizar las señales de los dos sensores utilizados en la investigación base.

4.6.2 Etiquetado de la base de datos existente

Se realizó el etiquetado de la base de datos existente que contiene los experimentos de marcha de personas con discapacidad visual que participaron en el estudio. Esta tarea implico asignar etiquetas o marcas precisas a los datos de la base de datos existente. Se utilizaron los datos reales de tamaño de zancada, velocidad de la marcha, distancia y contador de pasos obtenidos en la investigación previa (Reyes Leiva et al., 2023). Este proceso fue fundamental para garantizar la calidad y utilidad de los datos utilizados en el entrenamiento y evaluación de los dos algoritmos con base LSTM que se evaluaron. Esta tarea requirió una minuciosa revisión y análisis de los datos recolectados mediante sensores inerciales, con el propósito de asignar etiquetas o marcas precisas a cada uno de los eventos y parámetros de interés.

4.6.3 Creación de las bases de datos

Se crearon las bases de datos que contienen los cálculos de la media de los parámetros basados en los datos reales (videos y anotaciones). Las bases de datos fueron necesarias para poder realizar la evaluación de los algoritmos basados en LSTM. Con el conjunto de datos reales como punto de referencia, se pudo evaluar la capacidad de los algoritmos basados en LSTM para generalizar y adaptarse a una amplia variedad de situaciones y condiciones de marcha. Asimismo, la comparación con el método tradicional permitió identificar las mejoras significativas que pueden aportar los algoritmos basados en LSTM en la predicción de parámetros de marcha en personas ciegas.

La base de datos que tiene los cálculos de la media de los parámetros basados en los videos y anotaciones existentes de la base de datos es de suma importancia para establecer un punto de referencia confiable para la investigación. En esta base de datos, se registraron los parámetros de tamaño de zancada, velocidad de la marcha y contador de pasos a partir de la recopilación de los datos reales de personas ciegas caminando bajo diferentes condiciones y escenarios que se obtuvieron en la investigación previamente consultada. La obtención de la media de estos parámetros permitió reducir el efecto de las variaciones individuales y proporcionar valores promedio que reflejan el comportamiento típico de la marcha en personas ciegas.

4.6.4 ENTRENAMIENTO DE LOS ALGORITMOS

Se utilizó la base de datos etiquetada para entrenar ambos algoritmos. Durante esta fase, el algoritmo aprendió a reconocer y capturar los patrones subyacentes presentes en los datos etiquetados, lo que le permitió realizar predicciones precisas de los parámetros de marcha en personas ciegas. El proceso de entrenamiento requirió múltiples iteraciones para encontrar la configuración óptima de los parámetros del modelo. Se evaluó regularmente la precisión del algoritmo en un conjunto de datos de validación y se ajustaron los parámetros en función de los resultados obtenidos. La capacidad de iterar y mejorar continuamente el modelo es una característica clave de los algoritmos de aprendizaje automático como los que tienen una base en LSTM.

4.6.4.1 MODELO FCN+LSTM

Para el modelo 4FCN+LSTM, los datos se sometieron a pre procesamiento, incluida la normalización mediante escalado Min-Max. El conjunto de datos se dividió en conjuntos de entrenamiento y de prueba. El modelo 4FCN+LSTM se construyó con una capa de entrada que aceptaba secuencias de puntos de datos con ventanas. La arquitectura del modelo incluía una serie de capas densas y LSTM. El modelo se compiló utilizando el optimizador RMSprop y la función de pérdida de error cuadrático medio (MSE). El proceso de entrenamiento se llevó a cabo utilizando el conjunto de entrenamiento, y el rendimiento del modelo se evaluó en el conjunto de prueba. La duración del entrenamiento del modelo se registró para su análisis. Las pérdidas de entrenamiento y validación se visualizaron mediante gráficos para observar la convergencia.

Los datos de entrada para el modelo 4FCN+LSTM son una secuencia de ventanas de mediciones de sensores recogidas de sensores portátiles conectados a voluntarios con discapacidad visual. Cada secuencia contiene un número fijo de pasos temporales (tamaño de la ventana) e incluye múltiples características, como lecturas de acelerómetros y giroscopios de diferentes ubicaciones corporales. La forma de entrada es (tamaño del lote, tamaño de la ventana, número de características), donde el tamaño del lote representa el número de secuencias en un lote, el tamaño de la ventana es el número de pasos temporales en cada secuencia y el número de características es el número de mediciones del sensor para cada paso temporal.

La salida del modelo 4FCN+LSTM es un único valor continuo que representa el parámetro estimado de la marcha, concretamente la longitud del paso. El modelo se entrena para predecir este valor continúo basándose en las secuencias de entrada de las mediciones de los sensores. La forma de salida es (tamaño del lote, 1).

4.6.4.2 MODELO CNN+LSTM

Se aplicaron pasos de pre procesamiento similares al modelo CNN+LSTM. Los datos se dividieron en conjuntos de entrenamiento y de prueba, y se construyó la arquitectura CNN+LSTM. Este modelo incluía una combinación de capas Conv1D y LSTM para la extracción de características y el modelado de secuencias. El modelo se compiló utilizando el optimizador Adam y la pérdida de error cuadrático medio (MSE). Se llevaron a cabo el entrenamiento y la validación, y se analizó la convergencia del modelo mediante gráficos de pérdidas.

De forma similar al modelo 4FCN+LSTM, los datos de entrada para el modelo CNN+LSTM también son una secuencia de ventanas de mediciones de sensores obtenidas de sensores portátiles. Cada secuencia contiene un número fijo de pasos temporales (tamaño de ventana) con múltiples características. La forma de entrada es (tamaño de lote, tamaño de ventana, número de características).

La salida del modelo CNN+LSTM es también un único valor continuo que representa el parámetro estimado de la marcha, como la longitud del paso. Al igual que el modelo 4FCN+LSTM, el modelo CNN+LSTM predice este valor continúo basándose en las secuencias de entrada de las mediciones de los sensores. La forma de salida es (tamaño del lote, 1).

En ambos modelos, el objetivo es aprender las relaciones entre los datos de los sensores de entrada y el parámetro de la marcha (longitud del paso) a través del entrenamiento. Los modelos utilizan varias capas, como LSTM, Conv1D y capas densas, para capturar patrones temporales y características espaciales en los datos de entrada. La salida resultante proporciona una estimación del parámetro de la marcha para personas con discapacidad visual basada en sus datos de movimiento recogidos por sensores portátiles. La Tabla 2 muestra las diferentes características de ambos modelos.

Tabla 2: Características de los modelos

Parameters		LSTM+4FCN	CNN+LSTM
Layers with neurons		3 Dense Layers: (32 neurons, 16 neurons, 8 neurons)	2 Dense layers (100 neurons, 1 neuron)
		Regressor Layer: 1 neuron	
Dropout rate		0.5	0.5
Activation function		LSTM Layer (First): default (tanh)	2 TimeDistributed Conv1D Layers: ReLU
		3 Dense layers: ReLU (Rectified Linear Unit)	LSTM Layer: Default (tanh)
		Regressor Layer: ReLU (output is non-negative)	2 Dense layers: ReLU
			Output layer: Linear
Optimizer		RMSprop	Adam
Loss function		Mean Squared Error (MSE)	Mean Squared Error (MSE)
Batch Size	**Test 1**	128	128
	Test 2	256	256
	Test 3	128	128
Epochs	**Test 1**	25	25
	Test 2	25	25
	Test 3	30	30

Fuente: Elaboración Propia

Ambos modelos utilizan diferentes arquitecturas de capas para procesar las secuencias de entrada y extraer las características relevantes (Ilustración 25). Para evitar el sobreajuste, durante el entrenamiento se desactivan aleatoriamente algunas neuronas. La función de activación ReLU

introduce no linealidad, mientras que la función de activación lineal en la capa de salida produce predicciones continuas. El optimizador RMSprop adapta las tasas de aprendizaje para actualizar los pesos de forma eficaz. El número de épocas determina el número de veces que el modelo itera sobre los datos de entrenamiento, y el tamaño del lote controla el número de muestras utilizadas en cada paso de actualización de pesos. En los Anexos 149 y 150 se puede visualizar el código propuesto para ambos modelos utilizando el enfoque con ventanas deslizantes propuesto por (Jaén-Vargas et al., 2022). (Khan & Abedi, 2022)

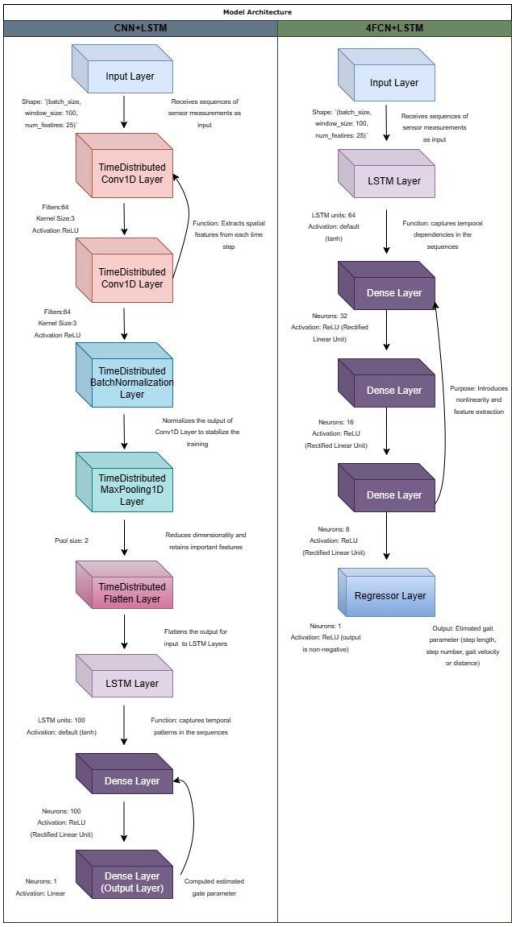


Ilustración 25 - Arquitectura del modelo CNN+LSTM y 4FCN+LSTM

Fuente: Elaboración propia

4.6.4.3 ALTERNATIVA DE ALGORITMO LSTM+FCN – SECUENCIA A UNO

El modelo de predicción Secuencia-a-Uno fue adoptado debido a su capacidad de condensar una secuencia de datos de entrada en una sola predicción. Este enfoque es especialmente útil cuando se quiere resumir información temporal en una única salida, que puede ser esencial para aplicaciones donde se busca una rápida toma de decisiones o cuando el interés

se centra en prever un punto específico en el tiempo. En este caso, se utilizó para la predicción de los parámetros de contador de pasos y distancia.

Este enfoque tiene algunas diferencias con el anteriormente mencionado, empezando por el pre procesamiento de datos que se centra en formatear los datos de entrada de cada experimento realizado como una sola secuencia y la salida se configura como un solo punto de datos. Esto lleva a una simplificación en la estructura del dataset. El pre procesamiento de datos para el modelo Secuencia-a-Uno es de vital importancia. Aquí, la serie temporal se transforma en múltiples muestras donde cada muestra contiene una secuencia de datos de entrada que se correlaciona con una única salida. Esta transformación permite al modelo aprender a partir de una amplia variedad de secuencias temporales y prever un resultado específico basado en ellas. La correcta estructuración de estos datos es crucial para el éxito del modelo, ya que determina cómo el modelo interpretará y utilizará la información temporal.

Mientras que el enfoque anterior toma una secuencia de datos y predice otra secuencia, el Secuencia-a-Uno toma una secuencia y predice un único dato. Esta diferencia fundamental altera el diseño y arquitectura del modelo, así como su aplicación.

El modelo Secuencia-a-Secuencia puede ser más útil cuando se desean predecir series temporales completas, como patrones de comportamiento en el tiempo. El Secuencia-a-Uno, en cambio, es más apto para situaciones donde el interés se centra en prever un evento específico o un dato concreto basado en una secuencia de información previa como es el caso de distancia recorrida y número total de pasos. En el Anexo 153 se puede visualizar el código creado a partir de los pasos propuestos en (Khan & Abedi, 2022)

4.6.5 EVALUACIÓN DE LOS ALGORITMOS BASADOS EN LSTM

Se evaluó la precisión de los algoritmos basados en LSTM en la predicción de los parámetros de marcha utilizando métricas de evaluación de regresión. Luego se comparó los resultados obtenidos con los valores reales estimados con el método tradicional. Estas métricas fueron especialmente adecuadas para medir el grado de ajuste y la capacidad predictiva del modelo en problemas de predicción numérica, como es el caso de la estimación de parámetros de marcha en personas ciegas.

Para esta evaluación se utilizaron métricas como el error cuadrático medio, que calcula la diferencia cuadrada promedio entre las predicciones de los algoritmos basados en LSTM y los valores reales estimados, y el error absoluto medio que mide la diferencia promedio entre las predicciones y los valores reales en términos absolutos. En ambas métricas, un valor más bajo indica una mayor precisión en la predicción del modelo.

Además de la evaluación mediante métricas de regresión, los resultados obtenidos por los algoritmos basados en LSTM se compararon con los valores reales estimados utilizando el método tradicional. Esta comparación permitió identificar las ventajas y desventajas del enfoque de aprendizaje automático en relación con el método tradicional y determino si los algoritmos con base LSTM ofrecen una mejora significativa en la precisión de la predicción de parámetros de marcha en personas ciegas.

Es importante destacar que la evaluación y comparación se llevó a cabo en un conjunto de datos de prueba independiente, que no ha sido utilizado en el entrenamiento del algoritmo. Esto asegura una evaluación imparcial y precisa del rendimiento del modelo en situaciones no vistas previamente, lo que fue crucial para determinar su capacidad de generalización y aplicabilidad en el mundo real.

4.6.6 MODIFICACIÓN DE PARÁMETROS DEL MODELO

Se realizan modificaciones en los parámetros de los modelos LSTM+CNN y LSTM+FCN con el objetivo de mejorar su precisión en la predicción de los parámetros de marcha. Estas modificaciones pueden incluir ajustes en el tamaño de la ventana, el número de épocas y el tamaño de lote. Además, se pueden explorar diferentes funciones de pérdida y métricas de evaluación que se adapten mejor a la naturaleza del problema de predicción de parámetros de marcha.

4.7 METODOLOGÍA DE VALIDACIÓN

La metodología de validación de esta investigación se basa en una comparativa de los resultados obtenidos con resultados previos establecidos mediante trabajos previos. Esta comparación tiene como objetivo principal verificar la eficacia y el avance logrado por los algoritmos con base LSTM propuestos en relación con los métodos existentes y establecidos en

la literatura científica. Se recopilaron sus resultados y métricas de evaluación, como el MSE, el MAE y el RMSE, u otras medidas relevantes utilizadas en dichos estudios.

1. Error cuadrático medio:

- Utilizado para ambos modelos.

- Mide la diferencia cuadrática media entre los valores previstos y los reales establecidos en la base de datos.

- Proporciona información sobre la magnitud de los errores.

- Los valores de MSE más pequeños indican un mejor rendimiento del modelo.

2. Raíz del error cuadrático medio y error absoluto medio. Estos errores se utilizan comúnmente en tareas de regresión:

- RMSE: Raíz cuadrada del MSE, proporciona el error en unidades originales.

- MAE: Diferencia absoluta media entre los valores predichos y los reales.

3. Curvas de pérdidas de validación y de entrenamiento:

- Generadas durante el entrenamiento del modelo.

- Visualiza cómo cambia la función de pérdida durante el entrenamiento y la validación.

- Ayuda a identificar el sobreajuste (si la pérdida de validación empieza a aumentar).

4. Tiempo de entrenamiento:

- Medido en segundos.

- Indica la duración del entrenamiento del modelo.

- Permite comparar la eficacia del entrenamiento entre modelos.

5. Curva de aprendizaje:

- Traza las métricas de entrenamiento y validación (por ejemplo, MSE) frente al número de muestras de entrenamiento.

- Ayuda a comprender si los modelos se ajustan poco o demasiado.

En conjunto, estas métricas proporcionan una visión global del rendimiento de los modelos. Los valores más bajos de MSE, RMSE y MAE indican que los modelos están produciendo predicciones que se acercan más a los valores reales. El seguimiento de la pérdida de validación a lo largo de las épocas ayuda a identificar el sobreajuste y a determinar cuándo detener el entrenamiento. La curva de aprendizaje ayuda a diagnosticar cualquier problema de sesgo o

varianza en el rendimiento del modelo. El tiempo de ejecución proporciona información sobre la eficiencia computacional. Todas estas métricas de evaluación ofrecen una imagen clara de la capacidad de los modelos para estimar con precisión los parámetros de la marcha.

Se evaluó cómo los algoritmos con base en LSTM se desempeñan en la predicción de los parámetros de marcha en personas ciegas en relación con el método tradicional calculado utilizando la misma base de datos. Se analizará si el algoritmo logra superar o igualar las métricas de evaluación de los métodos anteriores, y si presenta mejoras significativas en términos de precisión.

4.8 CRONOGRAMA DE ACTIVIDAD

Tabla 3 - Cronograma de Actividades

Actividades	Julio- Septiembre									
	Semanas									
	1	2	3	4	5	6	7	8	9	10
Primera interacción con UNCIH para recolección de datos										
Revisión de la literatura sobre redes neuronales.										
Segunda interacción con UNCIH para recolección de datos										
Etiquetado de la base existente y cálculo de parámetros										
Creación de la base de datos										
Adaptación de los modelos CNN + LSTM Y FCN + LSTM para realizar tareas de regresión.										
Modificación de los hiper parámetros de los modelos para realizar diversas pruebas en cada uno de los parámetros.										
Entrenamiento de los modelos CNN + LSTM Y FCN + LSTM										
Pruebas de cálculo de tamaño de zancada en los modelos CNN + LSTM Y FCN + LSTM										
Pruebas de cálculo de velocidad en los modelos CNN + LSTM Y FCN + LSTM										
Tercera interacción con UNCIH para recolección de datos										
Pruebas de cálculo de distancia en los modelos CNN + LSTM Y FCN + LSTM										
Desarrollo del modelo Seq2One										

Actividades	Julio- Septiembre									
	Semanas									
	1	2	3	4	5	6	7	8	9	10
Modificación de los hiper parámetros del modelo Seq2One para realizar las pruebas en el parámetro de numero de pasos.										
Pruebas de cálculo de numero de pasos en el modelo Seq2One										
Pruebas de cálculo de distancia en el modelo Seq2One										
Análisis de resultados										
Comparativa entre de los algoritmos basados en LSTM y el método biomecánico tradicional										
Finalización del proyecto de investigación										

Fuente: Elaboración propia

4.9 OPERALIZACIÓN DE LAS VARIABLES

Tabla 4 - Operalización De Las Variables

Objetivo General	Variable Dependiente	Definición Conceptual	Dimensiones	Indicadores
Evaluar la precisión de los algoritmos basados en LSTM en la predicción de parámetros de marcha en personas ciegas y compararlo con el método tradicional de cálculo.	Tamaño de la zancada	La distancia lineal desde el punto de inicio de un paso hasta el punto de finalización del siguiente paso en una secuencia de movimientos al caminar.	Unidades de distancia	Longitud promedio de la zancada Variabilidad en el tamaño de la zancada Tamaño de la zancada en situaciones específicas. Número de pasos totales
	Número de pasos	Cantidad total de pasos individuales que una persona da durante un período de tiempo específico o a lo largo de una distancia determinada al caminar.	Adimensionales	Número de pasos por unidad de tiempo Variabilidad en el número de pasos Número de pasos por distancia recorrida Número de pasos en diferentes contextos
	Velocidad de marcha	La tasa de cambio de posición en el tiempo durante el acto de caminar.	Unidades de longitud por unidad de tiempo.	Velocidad de marcha promedio Variabilidad en la Velocidad de marcha Velocidad de marcha en diferentes situaciones.
	Distancia de la marcha	Es una medida que cuantifica el espacio entre dos puntos en el espacio.	Unidades en metros	Distancia total Distancia recorrida Distancia promedio

Objetivo Específicos	Variable Independiente	Definición Conceptual	Dimensiones	Indicadores
Etiquetar la base de datos existente de experimentos de marcha de personas ciegas utilizando sensores inerciales. Calcular los parámetros de tamaño de zancada, velocidad de la marcha y contador de pasos. Entrenar los algoritmos basados en LSTM utilizando la base de datos etiquetada. Realizar modificaciones en los parámetros de los modelos basados en LSTM para mejorar su precisión en la predicción de los parámetros de marcha. Analizar los resultados obtenidos y generar una discusión en torno a la efectividad del método tradicional en comparación con el método de Deep Learning, considerando la influencia del tamaño de segmento de la pierna.	Modelos Características de los modelos	Los dos modelos que se utilizaran para el entrenamiento los cuales son LSTM+4FCN y LSTM+CNN. Los parámetros para modificar son las épocas, tamaño de la ventana y tamaño del lote en donde se estará realizando modificaciones en ellos con el fin de mejorar su precisión.	Arquitectura del Modelo Híper parámetros Tamaño del Modelo Funciones de Activación Tamaño de la ventana Numero de épocas Tamaño del lote	Error Cuadrático Medio Error Absoluto Medio Raíz del cuadrático absoluto medio Conjunto de métricas de entrenamiento Conjunto de métricas de evaluación

Fuente: Elaboración propia

V. RESULTADOS Y ANÁLISIS

En este capítulo, se presentan los resultados obtenidos de la investigación en la que se evaluaron las predicciones del tamaño de la zancada, la velocidad y la distancia de la marcha utilizando dos algoritmos basados en LSTM: 4FCN+LSTM y CNN+LSTM. Los valores predichos se muestran en comparación con los valores reales, junto con una exhaustiva evaluación de las métricas de entrenamiento y predicción, que incluyen el error medio cuadrático, el error medio absoluto y la raíz cuadrática absoluta media. Se ajustaron los valores del tamaño de lote, épocas y tamaño de la ventana para evaluar la precisión en cada una de las pruebas realizadas. Con el fin de realizar una comparativa más precisa entre los modelos, se establecieron los mismos parámetros modificados en cada prueba.

Tras una exhaustiva revisión de la base de datos, se emplearon ocho sujetos, lo que totalizó 35 experimentos que contenían la información necesaria para llevar a cabo la investigación. Cada tabla presenta las predicciones obtenidas de las pruebas realizadas en los modelos entrenados, junto con sus métricas de evaluación para cada sujeto. En total, se obtuvieron 210 predicciones por parámetro, 105 para cada modelo utilizado. En relación con el modelo Seq2One, se llevaron a cabo 70 predicciones en total, 35 para cada parámetro, es decir, distancia y número de pasos. Se realizaron tres pruebas para cada uno de los tres parámetros en ambos modelos, como se detalla en la tabla presentada en este capítulo.

Tabla 5 - Pruebas realizadas por cada modelo

Pruebas	Tamaño de lote	Épocas	Tamaño de ventana
Prueba 1	128	25	100
Prueba 2	256	25	100
Prueba 3	128	30	100

Fuente: Elaboración propia

5.1 RESULTADOS

En esta sección, se presentarán los resultados obtenidos a partir del estudio realizado. Los resultados se expondrán de manera clara y concisa, utilizando tablas y gráficos. Esta sección proporcionará una visión detallada de los datos recopilados y servirá como base para la posterior discusión e interpretación de estos en la sección correspondiente.

5.1.1 TAMAÑO DE ZANCADA

5.1.1.1 Modelo LSTM+ CNN

Tabla 6 - Prueba 1 Modelo LSTM+CNN

Modelo LSTM + CNN					
Prueba 1 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.63400	0.00012	0.01033	0.04233
Rápido	0.71	0.63570	0.00012	0.01033	0.02303
Lento	0.53	0.60370	0.00620	0.03696	0.04896
Regreso	0.61	0.63400	0.00766	0.06055	0.06156
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.52890	0.00067	0.02293	0.02233
Rápido	0.67	0.62600	0.00085	0.02659	0.03659
Lento	0.40	0.28890	0.00104	0.02395	0.02495
Regreso	0.52	0.52890	0.01182	0.04614	0.05614
Externo	0.57	0.54340	0.00750	0.03961	0.04961
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.4946	0.0071	0.0489	0.084
Lento	0.23	0.4913	0.0489	0.0325	0.0465
Regreso	0.40	0.4147	0.0101	0.0753	0.1
Externo	0.41	0.4532	0.0118	0.0928	0.108
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.54600	0.00083	0.02639	0.06239
Rápido	0.41	0.51570	0.00092	0.02814	0.08214
Lento	0.41	0.51570	0.00076	0.02494	0.04894
Regreso	0.41	0.52650	0.01023	0.04360	0.08160

Modelo LSTM + CNN					
Prueba 1 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.47	0.02	0.15	0.149
Lento	0.47	0.57	0.02	0.11	0.141
Regreso	0.51	0.52	0.03	0.14	0.159
Externo	0.70	0.46	0.28	0.13	0.1195
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.5410	0.0006	0.0223	0.0223
Rápido	0.50	0.5301	0.0054	0.0331	0.0331
Lento	0.45	0.4428	0.0240	0.0659	0.0659
Regreso	0.48	0.5299	0.0094	0.0599	0.0599
Externo	0.57	0.5873	0.0072	0.0380	0.0380
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.537	0.019	0.126	0.138
Rápido	0.43	0.614	0.007	0.046	0.0819
Lento	0.34	0.608	0.013	0.088	0.116
Externo	0.45	0.555	0.012	0.093	0.108
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.5175	0.0006	0.0223	0.0235
Rápido	0.60	0.6132	0.0007	0.0235	0.0645
Lento	0.56	0.5430	0.0006	0.0219	0.0219
Regreso	0.55	0.6131	0.0140	0.0489	0.0486
Externo	0.48	0.5295	0.0071	0.0401	0.0201
		PROMEDIO	0.016919427	0.054912857	0.068262086

Fuente: Elaboración propia

Tabla 7- Prueba 2 Modelo LSTM+CNN

Modelo LSTM + CNN					
Prueba 2 de Tamaño de zancada (metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.5123	0.0154	0.1241	0.124
Rápido	0.71	0.5360	0.0085	0.0917	0.0919
Lento	0.53	0.5540	0.0275	0.1460	0.166
Regreso	0.61	0.5042	0.0274	0.1251	0.166
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.528876	0.000666	0.022927	0.022927
Rápido	0.67	0.626041	0.000851	0.026588	0.026588
Lento	0.40	0.310155	0.001040	0.023952	0.023952
Regreso	0.52	0.528876	0.011821	0.046137	0.046137
Externo	0.57	0.543856	0.007499	0.039608	0.039608
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.46616	0.01904	0.10205	0.138
Lento	0.23	0.42984	0.01047	0.07471	0.102
Regreso	0.40	0.43761	0.01116	0.08219	0.106
Externo	0.41	0.51530	0.00394	0.04456	0.0627
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.515747	0.000833	0.026394	0.029394
Rápido	0.41	0.526492	0.000922	0.028145	0.056145
Lento	0.41	0.545391	0.000755	0.024935	0.042735
Regreso	0.41	0.526492	0.010226	0.043658	0.043658
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.47592	0.01944	0.13910	0.139
Lento	0.47	0.57570	0.02766	0.15486	0.166
Regreso	0.51	0.52145	0.03398	0.16963	0.184
Externo	0.70	0.49330	0.01601	0.10666	0.126

Modelo LSTM + CNN					
Prueba 2 de Tamaño de zancada (metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.51790	0.00061	0.02227	0.02527
Rápido	0.50	0.52989	0.00537	0.03306	0.03406
Lento	0.45	0.53788	0.02398	0.06590	0.07590
Regreso	0.48	0.50756	0.00940	0.05994	0.05594
Externo	0.57	0.58731	0.00716	0.03796	0.03764
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.5079	0.0282	0.1366	0.167
Rápido	0.43	0.4238	0.0467	0.2080	0.216
Lento	0.34	0.5204	0.0205	0.1165	0.143
Externo	0.45	0.4802	0.0322	0.1643	0.179
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.517531	0.000621	0.022314	0.023138
Rápido	0.60	0.613162	0.000676	0.023515	0.025152
Lento	0.56	0.543000	0.000585	0.021852	0.028519
Regreso	0.55	0.539516	0.013958	0.048883	0.048827
Externo	0.48	0.529565	0.007122	0.040102	0.041025
		PROMEDIO	0.012918356	0.075546288	0.085806194

Fuente: Elaboración propia

Tabla 8- Prueba 3 Modelo LSTM+CNN

Modelo LSTM + CNN					
Prueba 3 de Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.5632	0.0055	0.0743	0.0744
Rápido	0.71	0.5717	0.0025	0.0497	0.0499
Lento	0.53	0.6143	0.0045	0.0428	0.0668
Regreso	0.61	0.5758	0.0097	0.0686	0.0985
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.528876	0.000666	0.022927	0.029267
Rápido	0.67	0.626041	0.000851	0.026588	0.025877
Lento	0.40	0.545181	0.001040	0.023952	0.029524
Regreso	0.52	0.528876	0.011821	0.046137	0.041371
Externo	0.57	0.543856	0.007499	0.039608	0.036078
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.4613	0.0071	0.0680	0.0841
Lento	0.23	0.4444	0.0060	0.0572	0.0772
Regreso	0.40	0.5140	0.0053	0.0462	0.0728
Externo	0.41	0.4448	0.0227	0.1334	0.15
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.54601	0.00083	0.02639	0.04859
Rápido	0.41	0.51575	0.00092	0.02814	0.04034
Lento	0.41	0.51575	0.00076	0.02494	0.04615
Regreso	0.41	0.52649	0.01023	0.04366	0.06586
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.5276	0.0099	0.0992	0.0992
Lento	0.47	0.5154	0.0259	0.1444	0.16
Regreso	0.51	0.5983	0.0138	0.0878	0.117
Externo	0.70	0.5488	0.0098	0.0846	0.0988

Modelo LSTM + CNN					
Prueba 3 de Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.517904	0.000609	0.022275	0.022749
Rápido	0.50	0.537883	0.005369	0.033063	0.030628
Lento	0.45	0.529891	0.023981	0.065899	0.068987
Regreso	0.48	0.507564	0.009404	0.059938	0.059385
Externo	0.57	0.587309	0.007157	0.037964	0.039637
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.5079	0.0282	0.1366	0.133
Rápido	0.43	0.5821	0.0092	0.0701	0.0958
Lento	0.34	0.5274	0.0186	0.1176	0.136
Externo	0.45	0.6061	0.0078	0.0697	0.0881
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.517531	0.000621	0.022314	0.023138
Rápido	0.60	0.613162	0.000676	0.023515	0.025152
Lento	0.56	0.543000	0.000585	0.021852	0.028519
Regreso	0.55	0.539516	0.013958	0.048883	0.048827
Externo	0.48	0.529565	0.007122	0.040102	0.041025
		PROMEDIO	0.0083	0.0574	0.067220178

Fuente: Elaboración propia

5.1.1.2 Modelo LSTM+ FCN

Tabla 9 - Prueba 1 Modelo LSTM+FCN

Modelo LSTM + FCN					
Prueba 1 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.6138	0.000116	0.0077	0.0106
Rápido	0.71	0.6099	0.00053	0.0217	0.0232
Lento	0.53	0.6287	0.00164	0.0170	0.0405
Regreso	0.61	0.6287	0.00146	0.0150	0.0382
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.5493	0.0006	0.0203	0.0205
Rápido	0.67	0.5852	0.0002	0.0122	0.0322
Lento	0.40	0.4874	0.0007	0.0214	0.0214
Regreso	0.52	0.5433	0.0004	0.0061	0.0061
Externo	0.57	0.5526	0.0003	0.0039	0.0049
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.4911	0.00067	0.02194	0.026
Lento	0.23	0.4915	0.00095	0.2208	0.0309
Regreso	0.40	0.4582	0.001588	0.03145	0.0398
Externo	0.41	0.5008	0.00026	0.01276	0.0164
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.5291	0.0003128	0.0165	0.0195
Rápido	0.41	0.5151	0.001421	0.03748	0.03148
Lento	0.41	0.505	0.00116	0.03347	0.03547
Regreso	0.41	0.4748	0.0007234	0.004513	0.005513

Modelo LSTM + FCN					
Prueba 1 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.6119	0.00005117	0.0058	0.0071
Lento	0.47	0.6612	0.002119	0.03429	0.046
Regreso	0.51	0.4864	0.014	0.09349	0.1183
Externo	0.70	0.5818	0.002169	0.04085	0.0465
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.5475	0.000034	0.004622	0.004628
Rápido	0.50	0.5328	0.000070	0.006725	0.014725
Lento	0.45	0.4976	0.002378	0.02125	0.01125
Regreso	0.48	0.5458	0.0004064	0.004885	0.004882
Externo	0.57	0.5639	0.0002621	0.003105	0.003205
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.595	0.001	0.188	0.0332
Rápido	0.43	0.600	0.001	0.024	0.0279
Lento	0.34	0.642	0.001	0.024	0.0328
Externo	0.45	0.631	0.000	0.010	0.0162
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.57	0.00064	0.02	0.024
Rápido	0.60	0.60	0.00138	0.04	0.037
Lento	0.56	0.57	0.00027	0.01	0.025
Regreso	0.55	0.56	0.00081	0.01	0.005
Externo	0.48	0.55	0.00051	0.01	0.006
		PROMEDIO	0.0011806	0.030006086	0.024726771

Fuente: Elaboración propia

Tabla 10 - Prueba 2 Modelo LSTM+FCN

Modelo LSTM + FCN					
Prueba 2 Tamaño de zancada (metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.70594	0.00379	0.06080	0.0615
Rápido	0.71	0.69486	0.00073	0.02109	0.027
Lento	0.53	0.60461	0.00151	0.02341	0.0388
Regreso	0.61	0.61873	0.00157	0.02107	0.0395
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.53340	0.00087	0.02914	0.03914
Rápido	0.67	0.56050	0.00023	0.01366	0.01466
Lento	0.40	0.49940	0.00055	0.01977	0.02978
Regreso	0.52	0.52832	0.00275	0.03038	0.03038
Externo	0.57	0.57008	0.00216	0.00907	0.00907
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.56906	0.00625	0.04550	0.079
Lento	0.23	0.56258	0.00301	0.04301	0.0548
Regreso	0.40	0.49109	0.00115	0.02251	0.0339
Externo	0.41	0.47290	0.00220	0.04148	0.0469
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.4842517	0.0031242	0.0552803	0.0652803
Rápido	0.41	0.5779665	0.0017465	0.0401959	0.0411959
Lento	0.41	0.4763429	0.0030668	0.0543783	0.0544783
Regreso	0.41	0.4670050	0.0008440	0.0100373	0.0100473

Modelo LSTM + FCN					
Prueba 2 Tamaño de zancada (metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.64096	0.00120	0.03277	0.0346
Lento	0.47	0.64960	0.00237	0.03056	0.0487
Regreso	0.51	0.63232	0.00152	0.01962	0.039
Externo	0.70	0.61295	0.00199	0.02237	0.0446
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.5507	0.0117	0.1046	0.1236
Rápido	0.50	0.5211	0.0006	0.0211	0.0298
Lento	0.45	0.4954	0.0015	0.0126	0.0326
Regreso	0.48	0.5077	0.0023	0.0211	0.0411
Externo	0.57	0.5982	0.0008	0.0132	0.0093
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.61030	0.00299	0.02902	0.0547
Rápido	0.43	0.58700	0.00256	0.03009	0.0506
Lento	0.34	0.61760	0.00217	0.02137	0.0465
Externo	0.45	0.60243	0.00181	0.03248	0.0426
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.526728	0.000120	0.008389	0.008389
Rápido	0.60	0.593183	0.000248	0.013133	0.013133
Lento	0.56	0.556868	0.000094	0.007926	0.009926
Regreso	0.55	0.560861	0.002191	0.020262	0.020262
Externo	0.48		0.000836	0.006266	0.006266
		PROMEDIO	0.0020749	0.028216643	0.038031729

Fuente: Elaboración propia

Tabla 11- Prueba 3 Modelo LSTM+FCN

Modelo LSTM + FCN					
Prueba 3 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.63	0.63803	0.0000561	0.005532	0.00749
Rápido	0.71	0.63244	0.0000236	0.004093498	0.00486
Lento	0.53	0.610533	0.0002982	0.01330133	0.0172
Regreso	0.61	0.64194	0.0004626	0.0104737	0.0215
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.53	0.56270	0.00142	0.03623	0.01623
Rápido	0.67	0.62250	0.00132	0.03373	0.03373
Lento	0.40	0.52690	0.00015	0.01049	0.01029
Regreso	0.52	0.52267	0.00091	0.01430	0.01460
Externo	0.57	0.57507	0.00072	0.00664	0.00164
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.50	0.4861	0.0005	0.0194	0.0229
Lento	0.23	0.5101	0.0003	0.0072	0.0178
Regreso	0.40	0.4840	0.0009	0.0250	0.0296
Externo	0.41	0.5158	0.0001	0.0059	0.012
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.44	0.57017	0.00036	0.01819	0.01119
Rápido	0.41	0.54118	0.00014	0.00998	0.01498
Lento	0.41	0.52750	0.00022	0.01230	0.02230
Regreso	0.41	0.30212	0.00033	0.00379	0.00374

Modelo LSTM + FCN					
Prueba 3 Tamaño de zancada (metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0.69	0.617700	0.000029	0.004607	0.00537
Lento	0.47	0.588200	0.001711	0.029265	0.0413
Regreso	0.51	0.616900	0.000641	0.013180	0.0253
Externo	0.70	0.610430	0.000609	0.011421	0.0246
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.533301	0.000125	0.009188	0.010188
Rápido	0.50	0.541474	0.000188	0.010570	0.010570
Lento	0.45	0.459531	0.009357	0.077475	0.088575
Regreso	0.48	0.517999	0.000605	0.010351	0.021351
Externo	0.57	0.571112	0.000320	0.003669	0.004769
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.36	0.6289	0.0007	0.0094	0.0262
Rápido	0.43	0.6444	0.0009	0.0245	0.0293
Lento	0.34	0.5946	0.0015	0.0288	0.0385
Externo	0.45	0.6013	0.0006	0.0122	0.0236
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0.50	0.51977	0.00015	0.01007	0.03007
Rápido	0.60	0.55102	0.00059	0.02390	0.02890
Lento	0.56	0.56126	0.00041	0.01710	0.00710
Regreso	0.55	0.57979	0.00090	0.01120	0.02120
Externo	0.48	0.51896	0.00042	0.00600	0.01600
		PROMEDIO	0.0007983	0.015702121	0.020426829

Fuente: Elaboración propia

Tabla 12 - Promedio de las métricas de regresión de los modelos del tamaño de zancada

Promedio de las métricas de regresión de los modelos del tamaño de zancada (Metros)						
Modelo	Tamaño de lote	Épocas	Tamaño de ventana	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
LSTM + CNN	128	25	100	0.0169194	0.0549129	0.0682621
	256	25	100	0.0129184	0.0755463	0.0858062
	128	30	100	0.0082929	0.0573781	0.0672202
LSTM + FCN	128	25	100	0.0011806	0.0300061	0.0247268
	256	25	100	0.0020749	0.0282166	0.0380317
	128	30	100	0.0007983	0.0157021	0.0204268

Fuente: Elaboración propia

5.1.2 VELOCIDAD

5.1.2.1 Modelo LSTM+CNN

Tabla 13 - Prueba 1 de velocidad de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba de Velocidad (metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	0.76158	0.00687	0.05853	0.082908528
Rápido	1,43	0.83511	0.00225	0.02377	0.0474073
Lento	0,63	0.68940	0.00765	0.07180	0.0874443
Regreso	0,65	0.83568	0.00325	0.04199	0.056984
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	0.82537	0.00346	0.03260	0.0444672
Rápido	1,33	0.78158	0.00218	0.02328	0.0466554
Lento	0,57	0.79610	0.00141	0.02059	0.0375228
Regreso	0,59	0.81614	0.00323	0.03107	0.05680207
Externo	1,05	0.79860	0.00262	0.03669	0.051179792

Modelo LSTM + CNN					
Prueba de Velocidad (metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	0.8142	0.0047	0.0382	0.0687923
Lento	0,21	0.2636	0.0039	0.0383	0.062816
Regreso	0,38	0.6963	0.0055	0.05236304	0.07422604
Externo	0,49	0.1236	0.0053	0.0520	0.07276783
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	0.82685	0.00610	0.04719	0.078087
Rápido	0,88	0.81495	0.00255	0.02808	0.05052357
Lento	0,54	0.38862	0.00458	0.04006	0.06765747
Regreso	0,39	0.42596	0.00635	0.04897	0.07967353
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	0.77	0.0023	0.0302	0.0477806
Lento	0,60	0.75	0.0048	0.0409	0.0695772
Regreso	0,50	0.80	0.0033	0.032	0.057437042
Externo	1,11	0.78	0.01	0.0616	0.085868
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	0.7605	0.0061	0.0512	0.07786659
Rápido	0,83	0.7441	0.0050	0.0429	0.07080385
Lento	0,53	0.3984	0.0020	0.0213	0.04469603
Regreso	0,49	0.2129	0.0125	0.0944	0.111997
Externo	0,95	0.8255	0.0040	0.0482	0.06341748
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	0.871	0.002	0.026	0.039658
Rápido	0,71	0.870	0.006	0.056	0.079607
Lento	0,42	0.405	0.014	0.082	0.1173785
Externo	0,72	0.731	0.008	0.059	0.090242

Modelo LSTM + CNN					
Prueba de Velocidad (metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	0.7581	0.0036	0.0341	0.0600466
Rápido	0,90	0.7129	0.0054	0.0406	0.073282
Lento	0,56	0.7979	0.0055	0.0496	0.0739332
Regreso	0,56	0.3403	0.0061	0.0568	0.0783494
Externo	0,88	0.7738	0.0030	0.0359	0.05521794
		PROMEDIO	0.004938755	0.044230575	0.067516387

Fuente: Elaboración propia

Tabla 14- Prueba 2 de velocidad de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba 2 de Velocidad (Metros/segundo)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	0.74382	0.00584	0.04976	0.07641556
Rápido	1,43	0.85551	0.00503	0.04483	0.0709275
Lento	0,63	0.88913	0.00548	0.04735	0.07401352
Regreso	0,65	0.65960	0.01079	0.07677	0.1038642
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	0.36184	0.01959	0.13971	0.1399564
Rápido	1,33	0.43950	0.00431	0.06366	0.065651862
Lento	0,57	0.41610	0.00721	0.08465	0.08493811
Regreso	0,59	0.45530	0.00949	0.09614	0.097415749
Externo	1,05	0.43806	0.00635	0.07665	0.0797
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	0.7428	0.0152	0.0928	0.123135
Lento	0,21	0.3692	0.0168	0.0809	0.1295261
Regreso	0,38	0.3881	0.0116	0.06148827	0.10792
Externo	0,49	0.3803	0.0123	0.0792	0.110894664

Modelo LSTM + CNN					
Prueba 2 de Velocidad (Metros/segundo)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	0.39480	0.00902	0.09493	0.094998
Rápido	0,88	0.47390	0.00031	0.01720	0.017746293
Lento	0,54	0.40010	0.00880	0.09372	0.09381
Regreso	0,39	36.69330	0.00890	0.09179	0.0943564
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	0.78	0.0094	0.0682	0.0970833
Lento	0,60	0.40	0.0083	0.0756571	0.0910721
Regreso	0,50	0.75	0.0101	0.065	0.1003906
Externo	1,11	0.77	0.01	0.0647	0.0952397
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	0.3690	0.0096	0.0977	0.09778426
Rápido	0,83	0.4207	0.0093	0.0958	0.09646373
Lento	0,53	0.5385	0.0072	0.0813	0.08465969
Regreso	0,49	0.5130	0.0053	0.0663	0.07263832
Externo	0,95	0.5344	0.0034	0.0471	0.0582619
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	0.304	0.011	0.073	0.10464108
Rápido	0,71	0.801	0.008	0.062	0.09116235
Lento	0,42	0.421	0.011	0.071	0.106112
Externo	0,72	1.000	0.007	0.055	0.08146711
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	0.3788	0.0127	0.1128	0.11287854
Rápido	0,90	0.4663	0.0033	0.0559	0.0574056
Lento	0,56	0.4637	0.0006	0.0242	0.024398959
Regreso	0,56	0.5126	0.0015	0.0346	0.039075
Externo	0,88	0.4917	0.0311	0.1631	0.1762234
		PROMEDIO	0.008998832	0.074446164	0.090063628

Fuente: Elaboración propia

Tabla 15- Prueba 3 de velocidad de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba 3 de Velocidad (Metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	0.32249	0.00905	0.09507	0.09515339
Rápido	1,43	0.43558	0.00040	0.01945	0.02005312
Lento	0,63	0.65931	0.05851	0.24067	0.24189701
Regreso	0,65	0.61291	0.08825	0.28010	0.2970685
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	0.42887	0.00594	0.07659	0.077066
Rápido	1,33	0.47536	0.00016	0.01160	0.01269739
Lento	0,57	0.42347	0.00613	0.07787	0.078299459
Regreso	0,59	0.55520	0.00355	0.05657	0.05956763
Externo	1,05	0.39250	0.00539	0.05998	0.073446159
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	0.6555	0.0147	0.0938	0.12144638
Lento	0,21	0.4013	0.0070	0.0545	0.083746905
Regreso	0,38	0.4202	0.0049	0.03797281	0.0702134
Externo	0,49	0.4270	0.0153	0.1074	0.123682537
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	0.48126	0.000067	0.00791	0.008182
Rápido	0,88	0.23468	0.08329	0.28811	0.288592565
Lento	0,54	0.42010	0.00720	0.08474	0.08484923
Regreso	0,39	0.40879	0.00853	0.09120	0.0923456
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	0.43	0.0001	0.0071	0.007339436
Lento	0,60	0.68	0.0303	0.113931757	0.174199796
Regreso	0,50	0.51	0.0203	0.094	0.14249129
Externo	1,11	0.76	0.01	0.0511	0.085789784

Modelo LSTM + CNN					
Prueba 3 de Velocidad (Metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	0.4413	0.0053	0.0724	0.07305463
Rápido	0,83	0.4189	0.0049	0.0700	0.070189362
Lento	0,53	0.5443	0.0025	0.0434	0.049883073
Regreso	0,49	0.5365	0.0030	0.0536	0.05467102
Externo	0,95	0.5154	0.0031	0.0511	0.05564339
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	0.442	0.009	0.066	0.096287347
Rápido	0,71	0.673	0.011	0.070	0.1038969
Lento	0,42	38.713	0.015	0.074	0.122852246
Externo	0,72	0.694	0.013	0.086	0.111804415
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	0.4055	0.0065	0.0806	0.0806321
Rápido	0,90	0.3650	0.0152	0.1232	0.123226721
Lento	0,56	0.4080	0.0098	0.0988	0.09890193
Regreso	0,56	0.4168	0.0169	0.1286	0.12990865
Externo	0,88	0.4606	0.0033	0.0563	0.0575523
		PROMEDIO	0.014133529	0.086389293	0.099046619

Fuente: Elaboración propia

5.1.2.2 Modelo LSTM +FCN

Tabla 16 – Prueba 1 de velocidad de modelo LSTM+FCN

Modelo LSTM + FCN					
Prueba de Velocidad (metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	0.4202	0.000179	0.01272	0.01338
Rápido	1,43	0.40332	0.00022036	0.010459	0.014844
Lento	0,63	0.36314	0.0031701	0.0446	0.056303
Regreso	0,65	0.43813	0.001532	0.0349	0.03914
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	0.4571	0.0002	0.0108	0.01225573
Rápido	1,33	0.4414	0.0010	0.0302	0.0315556
Lento	0,57	0.4770	0.0000255	0.0037	0.005052
Regreso	0,59	0.5011	0.0006	0.0119	0.02376598
Externo	1,05	0.4706	0.0003	0.0031	0.0164631
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	0.5093	0.0004272	0.01096	0.206
Lento	0,21	0.5177	0.000376	0.0111	0.0194
Regreso	0,38	0.5203	0.001789	0.02144	0.0423
Externo	0,49	0.5376	0.0006328	0.02131	0.02515
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	0.4852	0.00013228	0.01038	0.011501671
Rápido	0,88	0.49434	0.000391	0.0194898	0.0197775
Lento	0,54	0.4625	0.000180972	0.0120597	0.01345258
Regreso	0,39	0.48363	0.000788	0.008479	0.028072218

Modelo LSTM + FCN					
Prueba de Velocidad (metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	0.416279	0.00018234	0.011915	0.01350359
Lento	0,60	0.43032	0.002770615	0.029538	0.052636
Regreso	0,50	0.41156	0.00144694	0.0159394	0.038038
Externo	1,11	0.404971	0.00061388	0.01911	0.024776749
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	50.76158	0.000092	0.0080363	0.0095887
Rápido	0,83	0.49037	0.000139	0.010125	0.0117784
Lento	0,53	0.43883	0.005878	0.047	0.0766686
Regreso	0,49	0.47014	0.000429356	0.0048834	0.0207209
Externo	0,95	53.47534	0.0004639	0.00716735	0.021538
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	0.423	0.003	0.0329	0.0512
Rápido	0,71	0.385	0.002	0.025	0.03998
Lento	0,42	0.455	0.003	0.055	0.058552
Externo	0,72	0.401	0.001	0.015	0.0295705
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	0.52	0.001538	0.04	0.039219
Rápido	0,90	0.4730	0.0000576	0.01	0.00758647
Lento	0,56	0.4738	0.00026	0.01	0.0160298
Regreso	0,56	0.49	0.000553	0.01	0.0235088
Externo	0,88	0.45	0.00067	0.01	0.02588157
		PROMEDIO	0.001012765	0.018084515	0.032548299

Fuente: Elaboración propia

Tabla 17- Prueba 2 de velocidad de modelo LSTM+ FCN

Modelo LSTM + FCN					
Prueba 2 de Velocidad (Metros/segundo)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	0.42308	0.00032	0.01583	0.01787893
Rápido	1,43	0.39127	0.00025	0.01532	0.015695
Lento	0,63	0.40032	0.00198	0.01590	0.04448933
Regreso	0,65	0.41508	0.00107	0.01762	0.0326549
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	0.45155	0.00095	0.02855	0.0307502
Rápido	1,33	0.42230	0.00337	0.05795	0.05807
Lento	0,57	0.47479	0.00007	0.00545	0.0083829
Regreso	0,59	0.51938	0.00119	0.01587	0.034434
Externo	1,05	0.49912	0.00089	0.00505	0.029764
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	0.48331	0.00232	0.04082	0.0482023
Lento	0,21	0.50672	0.00144	0.01647	0.0379457
Regreso	0,38	0.52731	0.00083	0.01536	0.02873817
Externo	0,49	0.52029	0.00134	0.01977	0.0365804
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	0.5039500	0.0005907	0.0239010	0.02430373
Rápido	0,88	0.4936000	0.0001678	0.0104930	0.0129528
Lento	0,54	0.4491560	0.0005719	0.0232902	0.02391458
Regreso	0,39	0.4820030	0.0005364	0.0047919	0.02315933
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	0.36503	0.00289	0.05194	0.053766
Lento	0,60	0.44248	0.02693	0.09475	0.164093223
Regreso	0,50	0.51223	0.02677	0.11088	0.163616
Externo	1,11	0.39102	0.00111	0.02342	0.0332974

Modelo LSTM + FCN					
Prueba 2 de Velocidad (Metros/segundo)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	0.4960	0.0004	0.0195	0.0209035
Rápido	0,83	0.4961	0.0007	0.0195	0.026026
Lento	0,53	0.4698	0.0019	0.0127	0.0436476
Regreso	0,49	0.4742	0.0006	0.0070	0.02365673
Externo	0,95	0.4146	0.0011	0.0178	0.032624143
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	0.41746	0.00311	0.02864121	0.055797265
Rápido	0,71	0.34788	0.00309	0.04180	0.055553538
Lento	0,42	0.37069	0.00279	0.03533	0.052806
Externo	0,72	0.34144	0.00588	0.07091	0.07667983
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	0.435606	0.001882	0.042441	0.04337673
Rápido	0,90	0.512470	0.000693	0.025305	0.026322816
Lento	0,56	0.483971	0.000060	0.006566	0.00775976
Regreso	0,56	0.518080	0.001521	0.024221	0.038997
Externo	0,88	0.397186	0.001841	0.023241	0.04291237
		PROMEDIO	0.00289	0.02824	0.041992919

Fuente: Elaboración propia

Tabla 18- Prueba 3 de velocidad de modelo LSTM+ FCN

Modelo LSTM + FCN					
Prueba 3 de Velocidad (Metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	1,00	41.53180	0.00068	0.02588	0.0260519
Rápido	1,43	38.87500	0.00028	0.01628	0.016759
Lento	0,63	79.43559	0.00449	0.04789	0.067034
Regreso	0,65	41.01610	0.00157	0.01727	0.039672

Modelo LSTM + FCN					
Prueba 3 de Velocidad (Metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,73	49.91445	0.00003	0.00479	0.005625
Rápido	1,33	49.63714	0.00004	0.00489	0.006322
Lento	0,57	46.50083	0.00175	0.03885	0.041877
Regreso	0,59	49.89340	0.00055	0.00662	0.0234713
Externo	1,05	44.96330	0.00482	0.03214	0.06945651
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	0,83	88.51440	0.00210	0.04291	0.045870912
Lento	0,21	20.01590	0.00039	0.01063	0.01978649
Regreso	0,38	53.72240	0.00058	0.01802	0.0241815
Externo	0,49	52.84040	0.00054	0.01847	0.0232972
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,88	49.9860760	0.0000517	0.0060238	0.00718792
Rápido	0,88	48.0826280	0.0001997	0.0132660	0.0141315
Lento	0,54	48.9178500	0.0001718	0.0110086	0.0131072
Regreso	0,39	44.7584530	0.0006096	0.0123804	0.0246902
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	1,50	42.80900	0.00001	0.00245	0.00297066
Lento	0,60	41.52870	0.00126	0.01305	0.0354822
Regreso	0,50	42.79620	0.00678	0.02918	0.082337
Externo	1,11	95.99040	0.00023	0.00669	0.01519251
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,63	51.5044	0.00003	0.0044	0.0057605
Rápido	0,83	51.3616	0.0001	0.0057	0.00820211
Lento	0,53	51.6531	0.0018	0.0383	0.042741
Regreso	0,49	46.3410	0.0002	0.0042	0.014892303
Externo	0,95	52.6070	0.0005	0.0074	0.02185011

Modelo LSTM + FCN					
Prueba 3 de Velocidad (Metros/segundo)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,48	40.65200	0.00067	0.00962	0.02595
Rápido	0,71	43.70560	0.00260	0.03695	0.0510013
Lento	0,42	39.70370	0.00055	0.01122	0.02346
Externo	0,72	89.05960	0.00028	0.00796	0.016731471
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	0,69	46.302590	0.002186	0.045655	0.04675694
Rápido	0,90	44.4047361	0.003343	0.057361	0.05781736
Lento	0,56	52.764243	0.000381	0.017390	0.01950752
Regreso	0,56	50.700340	0.000228	0.003150	0.015103
Externo	0,88	50.814634	0.000667	0.005970	0.025826326
		PROMEDIO	0.00116	0.01811	0.02800297

Fuente: Elaboración propia

Tabla 19 Promedio de las métricas de regresión de los modelos de velocidad

Promedio de las métricas de regresión de los modelos de velocidad						
Modelo	Tamaño de lote	Épocas	Tamaño de ventana	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
LSTM + CNN	128	25	100	0.0049388	0.0442306	0.0675164
	256	25	100	0.0089988	0.0744462	0.0900636
	128	30	100	0.0141335	0.0863893	0.0990466
LSTM + FCN	128	25	100	0.0010128	0.0180845	0.0325483
	256	25	100	0.0028877	0.0282416	0.0419929
	128	30	100	0.0011628	0.0181138	0.0280030

Fuente: Elaboración propia

5.1.3 DISTANCIA

5.1.3.1 Modelo LSTM + CNN

Tabla 20 - Prueba 1 de distancia de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana= 100 Épocas= 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	13.61622	55.217842	7.08268	7.43087
Rápido	10	13.61622	67.787059	7.92867	8.23329
Lento	10	13.61622	88.333389	8.80949	9.39858
Regreso	20	20.83547	136.827863	11.15955	11.69734
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	11.52155	737.45783	26.71670	27.15617
Rápido	8	11.52155	709.60580	26.15536	26.63843
Lento	8	11.52155	2708.77402	41.42623	52.04588
Regreso	16	15.68845	580.87864	22.57878	24.10142
Externo	60	40.28928	789.13961	26.82477	28.09163
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	9.0437	44.04236	5.3993	6.6364
Lento	10	9.0437	1034.8513	20.0012	32.1691
Regreso	20	20.8834	1200.0017	23.6205	34.6410
Externo	60	65.0855	2222.0502	34.5666	47.1386
			Sujeto 4		
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	21.90642	529.70229	22.72631	23.01526
Rápido	7	21.90642	503.28376	22.12290	22.43399
Lento	7	21.90642	564.32761	23.48378	23.75558
Regreso	14	24.20370	552.30522	23.26208	23.50117

Modelo LSTM + CNN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana= 100 Épocas= 25					
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	9.34	10.815854	2.6460	3.2887
Lento	9	12.01	36.618874	5.0722	6.0514
Regreso	18	13.34	29.080571	4.4600	5.3926
Externo	60	27.00	1859.8483	28.600	43.126
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	23.2224	539.1149	22.9348	23.2188
Rápido	10	23.0948	541.0120	22.9812	23.2597
Lento	10	22.1014	520.5748	22.4960	22.8161
Regreso	20	24.1181	912.5466	28.6773	30.2084
Externo	60	41.0966	506.8964	26.2587	22.5144
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	9.8196	2442.3971	30.325	49.421
Rápido	10	9.809	3408.9171	25.485	58.386
Lento	10	9.820	2662.0315	34.287	51.595
Externo	106	98.196	1444.2972	24.534	38.004
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	22.1230	560.9112	23.4256	23.6836
Rápido	9	22.1230	557.3345	23.3575	23.6079
Lento	9	22.1230	564.7211	23.5193	23.7639
Regreso	18	24.7372	524.8278	22.6458	22.9091
Externo	60	63.6622	547.4351	22.4982	23.3973
		PROMEDIO	862.6847916	21.37338232	25.79223

Fuente: Elaboración propia

Tabla 21 - Prueba 2 de distancia de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba 2 de Distancia (Metros)					
Tamaño de lote = 256 Tamaño de ventana= 100 Épocas= 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	39.13625	1205.699646	34.65962	34.72319
Rápido	10	53.19946	2744.714196	52.26330	52.39002
Lento	10	30.60735	5356.483344	52.44967	73.18800
Regreso	20	27.08063	109.28132	8.69375	10.45377
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	18.51061	795.39011	27.98861	28.20266
Rápido	8	26.76365	8405.01154	77.21950	91.67885
Lento	8	6.41374	339.91191	18.09037	18.43670
Regreso	16	22.86047	5297.33595	62.97623	72.78280
Externo	60	57.49100	7686.92176	56.98682	87.67509
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	8.8979	27.01750	4.2685	5.1978
Lento	10	4.5516	9505.80736	57.7160	97.4977
Regreso	20	28.4262	256.70504	13.8590	16.0220
Externo	60	34.5818	200.74951	12.6493	14.1686
			Sujeto 4		
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	7.66206	134.91025	10.18151	11.61509
Rápido	7	1.84260	7069.72656	60.68900	84.08167
Lento	7	9.88492	42.81774	5.88274	6.54353
Regreso	14	5.07033	7287.37856	56.77586	85.36614
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	20.31	501.541306	21.9953	22.3951
Lento	9	17.12	257.851434	15.6016	16.0578
Regreso	18	19.53	5.431510	1.7612	2.3306
Externo	60	41.01	170.516587	11.057	13.0582

Modelo LSTM + CNN					
Prueba 2 de Distancia (Metros)					
Tamaño de lote = 256 Tamaño de ventana= 100 Épocas= 25					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	11.1617	99.7994	9.4711	9.9900
Rápido	10	41.8278	1683.1566	40.9460	41.0263
Lento	10	6.8606	536.7294	22.0836	23.1674
Regreso	20	56.7290	2730.0464	44.9410	52.2498
Externo	60	58.5560	854.6154	26.3694	29.2338
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	18.5070	9370.9567	82.819	96.804
Rápido	10	29.7970	8746.9060	83.955	93.525
Lento	10	33.513	5449.5915	67.081	73.821
Externo	106	97.122	22296.3010	139.714	149.319
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	62.5987	3,475.6590	58.9192	58.9547
Rápido	9	26.1484	567.1623	23.6801	23.8152
Lento	9	5.3910	4,931.0591	63.9160	70.2215
Regreso	18	24.3933	1,146.4423	33.3460	33.8592
Externo	60	29.2370	619.7585	22.5627	24.8950
		PROMEDIO	3425.982482	39.53051973	46.42136

Fuente: Elaboración propia

Tabla 22 - Prueba 3 de distancia de modelo LSTM+ CNN

Modelo LSTM + CNN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana= 100 Épocas= 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	25.91576	454.006893	21.19556	21.30744
Rápido	10	23.62159	698.476184	25.84127	26.42870
Lento	10	36.63810	31411.564327	126.84286	177.23308
Regreso	20	28.93064	695.11479	26.26635	26.36503

Modelo LSTM + CNN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana= 100 Épocas= 30					
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	44.92858	2931.64146	54.03452	54.14463
Rápido	8	51.55924	2983.77755	54.59989	54.62397
Lento	8	8.40974	5484.40985	43.79527	74.05680
Regreso	16	45.99916	2426.92839	48.22154	49.26386
Externo	60	70.00553	99.91002	8.08206	9.99550
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	13.9074	414.07203	18.9306	20.3488
Lento	10	29.8164	700.1104	23.3486	26.4596
Regreso	20	25.3830	699.7718	24.1800	26.4532
Externo	60	35.4796	277.5611	12.5675	16.6602
			Sujeto 4		
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	44.73557	1589.28679	39.73130	39.86586
Rápido	7	58.48041	3467.29837	58.53958	58.88377
Lento	7	3.49484	1953.84952	29.39276	44.20237
Regreso	14	28.74657	542.23774	23.21360	23.28600
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	2.97	268.980242	15.9075	16.4006
Lento	9	13.91	268.980238	15.9075	16.4006
Regreso	18	35.75	6493.521578	56.3843	80.5824
Externo	60	48.25	185.5646	10.963	13.622
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	11.9944	1141.4819	31.8578	33.7858
Rápido	10	22.0771	6146.0300	56.2196	78.3966
Lento	10	8.5604	81.7570	8.0535	9.0420
Regreso	20	41.9243	1726.0037	36.0613	41.5452
Externo	60	55.7673	798.8661	25.3291	28.2642

Modelo LSTM + CNN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana= 100 Épocas= 30					
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	14.2522	12808.3226	76.787	113.174
Rápido	10	5.939	10642.7799	86.796	103.164
Lento	10	11.847	21822.7237	107.688	147.725
Externo	106	79.725	14818.4901	85.599	121.731
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	25.5630	2,737.8648	49.3853	52.3246
Rápido	9	57.7594	3,186.8389	56.2960	56.4521
Lento	9	57.8356	3,205.0850	56.4691	56.6135
Regreso	18	14.5594	86.8568	7.6575	9.3197
Externo	60	37.7491	789.9730	23.1790	28.1065
		PROMEDIO	4115.432503	41.29498711	50.17797

Fuente: Elaboración propia

5.1.3.2 Modelo LSTM + FCN

Tabla 23 - Prueba 1 de distancia de modelo LSTM+ FCN

Modelo LSTM + FCN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	10.03829	25.169528	4.23707	5.01692
Rápido	10	11.85291	9.262356	2.28048	3.04341
Lento	10	21.44498	256.287208	15.85176	16.00897
Regreso	20	18.74068	14.019496	3.58494	3.74426

Modelo LSTM + FCN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	9.08232	108.156100	9.08232	10.39981
Rápido	8	7.03696	332.209402	18.21190	18.22661
Lento	8	7.74959	192.089185	13.67439	13.85962
Regreso	16	16.39956	896.423707	21.56063	29.94034
Externo	60	66.50192	246.196481	14.00235	15.69065
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	12.6599	85.418672	8.9867	9.2422
Lento	10	11.9379	425.686547	16.9619	20.6322
Regreso	20	19.9543	143.518135	9.9748	11.9799
Externo	60	61.8116	448.882643	22.2047	21.1869
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	8.53105	32.122907	5.10788	5.66771
Rápido	7	5.52357	893.393802	29.83934	29.88969
Lento	7	11.98951	21.518061	4.17611	4.63876
Regreso	14	14.82520	37.736232	5.87469	6.14298
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	5.45	61.441026	6.7212	7.8384
Lento	9	8.98	10.486940	2.7414	3.2384
Regreso	18	18.30	124.219329	8.5838	11.1454
Externo	60	67.71	856.822358	20.328	29.272

Modelo LSTM + FCN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	13.4526	146.61637	11.2659	12.1085
Rápido	10	16.2311	77.54987	7.1929	8.8062
Lento	10	7.1888	992.52351	31.4952	31.5043
Regreso	20	18.4497	192.77399	3.6522	13.8843
Externo	60	100.1029	791.43019	14.7575	28.1324
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	9.3700	870.3096	20.701	29.501
Rápido	10	9.370	449.5869	16.138	21.203
Lento	10	9.752	935.3126	17.117	30.583
Externo	106	92.940	902.5309	24.978	30.042
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	10.7249	173.30751	12.6876	13.1646
Rápido	9	8.9018	21.45543	4.2876	4.6320
Lento	9	21.5874	620.85093	24.0332	24.91688
Regreso	18	19.8334	273.86385	16.4509	16.5488
Externo	60	86.9019	390.23689	10.7529	19.7544
		PROMEDIO	344.554533	13.12848918	16.04533

Fuente: Elaboración propia

Tabla 24- Prueba 2 de distancia de modelo LSTM+ FCN

Modelo LSTM + FCN					
Prueba 2 de Distancia (Metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	24.63521	330.699026	18.11500	18.18513
Rápido	10	24.70712	339.594870	18.33122	18.4281
Lento	10	65.73551	3095.197590	55.6345	55.63450
Regreso	20	26.69589	590.854663	24.11387	24.30750
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	29.87645	2000.077409	44.07763	44.72223
Rápido	8	29.87645	1917.781305	42.96462	43.79248
Lento	8	27.37583	1876.128584	42.69019	43.31430
Regreso	16	14.97100	362.881926	17.73494	19.04946
Externo	60	49.90330	305.394498	16.22923	17.47554
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	65.7355	3095.197590	55.6340	55.6345
Lento	10	21.6189	567.331899	22.1120	23.8187
Regreso	20	22.3748	713.668250	24.8820	26.7146
Externo	60	21.1050	2075.248622	20.9844	45.5549
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	14.27054	109.482740	10.08021	10.46340
Rápido	7	53.59308	1882.60584	43.38901	43.38901
Lento	7	8.71290	79.416793	8.26000	8.91161
Regreso	14	13.40340	18.695506	3.75726	4.32383
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	18.06	149.734624	12.1238	12.2366
Lento	9	26.34	266.551337	16.3198	16.3264
Regreso	18	17.65	61.649962	7.5460	7.8517
Externo	60	50.66	273.5253	14.594	16.539

Modelo LSTM + FCN					
Prueba 2 de Distancia (Metros)					
Tamaño de lote = 256 Tamaño de ventana = 100 Épocas = 25					
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	18.5885	188.10699	13.5683	13.7152
Rápido	10	7.4962	320.299030	13.8983	17.8969
Lento	10	7.4962	317.7556	14.4504	17.8257
Regreso	20	23.6889	1494.13945	29.8933	38.6541
Externo	60	74.4579	3288.93372	39.5791	57.3492
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	24.8548	985.0442	22.072	31.385
Rápido	10	43.9214	690.29740	19.465	26.274
Lento	10	24.575	954.34158	22.109	30.892
Externo	106	96.929	902.0679	26.025	30.034
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	14.8265	104.30131	10.0727	10.2128
Rápido	9	15.6850	108.679166	10.3081	10.4249
Lento	9	10.4295	20.59070	4.2820	4.53770
Regreso	18	18.8074	22.44588	4.4130	4.7377
Externo	60	30.4717	1,724.83227	29.3634	41.5311
		PROMEDIO	892.387243	22.25922416	25.48984

Fuente: Elaboración propia

Tabla 25- Prueba 3 de distancia de modelo LSTM+ FCN

Modelo LSTM + FCN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	13.15710	13.341502	3.22882	3.65260
Rápido	10	13.62359	17.091431	3.76417	4.13418
Lento	10	13.51235	16.879394	3.60560	4.10845
Regreso	20	23.69936	193.505095	13.87217	13.91061

Modelo LSTM + FCN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	10.19252	270.688991	16.32520	16.45263
Rápido	8	44.20277	2709.56374	51.95280	52.05347
Lento	8	20.91460	766.40275	27.57989	27.68398
Regreso	16	43.53663	2067.72853	41.69413	45.47228
Externo	60	53.66552	397.08533	18.62796	19.92700
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	23.6994	185.281676	13.6112	13.6118
Lento	10	12.5295	120.97074	10.3060	10.9987
Regreso	20	10.7972	58.35046	6.1409	7.6387
Externo	60	60.9604	856.6698	19.0609	29.2689
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	9.16660	47.016776	6.60719	6.85688
Rápido	7	9.16662	42.23050	6.24100	6.49850
Lento	7	45.55210	1749.225228	41.75210	41.82374
Regreso	14	19.64043	268.450356	16.21453	16.38445
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	46.20	1471.006718	38.2914	38.3537
Lento	9	24.77	392.142967	19.5516	19.8026
Regreso	18	20.10	25.002900	4.3245	5.0003
Externo	60	35.25	52.0396	5.453	7.214
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	9.4245	8.63595	2.5541	2.9387
Rápido	10	9.4245	7.867464	2.3748	2.8049
Lento	10	29.6300	656.8251	21.9404	25.6286
Regreso	20	45.1100	1032.13128	25.3962	32.1268
Externo	60	41.9840	1428.51796	34.8081	37.7957

Modelo LSTM + FCN					
Prueba 3 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 30					
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	24.8548	985.0442	22.072	31.385
Rápido	10	9.780	690.29740	19.465	26.274
Lento	10	24.575	954.34158	22.109	30.892
Externo	106	96.930	902.0679	26.025	30.034
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	50.6686	2,442.33343	49.3938	49.4200
Rápido	9	22.1772	233.964863	15.2196	15.2959
Lento	9	9.5029	9.37764	2.6901	3.06229
Regreso	18	13.7760	301.821129	16.8053	17.3730
Externo	60	60.5670	2,106.95688	42.4745	45.9016
		PROMEDIO	670.881637	19.18664911	21.19373

Fuente: Elaboración propia

Tabla 26 - Promedio de las métricas de regresión de los modelos de distancia

Promedio de las métricas de regresión de los modelos de distancia						
Modelo	Tamaño de lote	Épocas	Tamaño de ventana	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
LSTM + CNN	128	25	100	862.6847916	21.3733823	25.7922304
	256	25	100	3425.9824818	39.5305197	46.4213588
	128	30	100	4115.4325030	41.2949871	50.1779663
LSTM + FCN	128	25	100	344.5545330	13.1284892	16.0453324
	256	25	100	892.3872429	22.2592242	25.4898373
	128	30	100	670.8816368	19.1866491	21.1937339

5.1.3.3 Modelo LSTM+FCN (Secuencia a uno)

Tabla 27 - Distancia de modelo LSTM+FCN (Secuencia a uno)

Modelo LSTM + FCN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	7.45500	6.477025	2.54500	2.54500
Rápido	10	9.80151	0.039398	0.19849	0.19849
Lento	10	9.17343	0.683218	0.82657	0.82657
Regreso	20	9.76355	104.784909	10.23645	10.23645
Sujeto 2					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	8	7.26707	0.537186	0.73293	0.73293
Rápido	8	8.21217	0.045016	0.21217	0.21217
Lento	8	7.98306	0.000287	0.01694	0.01694
Regreso	16	10.71756	27.904172	5.28244	5.28244
Externo	60	62.26911	5.148860	2.26911	2.26911
Sujeto 3					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	10	9.6958	0.092510	0.3042	0.3042
Lento	10	7.6482	5.531102	2.3518	2.3518
Regreso	20	21.5060	2.267946	1.5060	1.5060
Externo	60	20.6292	1550.057530	39.3708	39.3708
Sujeto 4					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	7	8.00000	1.000000	1.00000	1.00000
Rápido	7	9.52234	6.362202	2.52234	2.52234
Lento	7	10.24791	10.548887	3.24791	3.24791
Regreso	14	13.60398	0.156832	0.39602	0.39602

Modelo LSTM + FCN					
Prueba 1 de Distancia (Metros)					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 5					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	9	9.21	0.043044	0.2075	0.2075
Lento	9	8.02	0.966245	0.9830	0.9830
Regreso	18	15.02	8.869973	2.9783	2.9783
Externo	60	56.83	10.020643	3.1655	3.1655
Sujeto 6					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	12.4428	5.96739	2.4428	2.4428
Rápido	10	10.4873	0.237435	0.4873	0.4873
Lento	10	8.2571	3.0375	1.7429	1.7429
Regreso	20	16.3348	13.43347	3.6652	3.6652
Externo	60	52.5644	55.28770	7.4356	7.4356
Sujeto 7					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	10	9.1100	0.7921	0.890	0.890
Rápido	10	8.840	1.34560	1.160	1.160
Lento	10	10.154	0.02372	0.154	0.154
Externo	106	92.830	173.4494	13.170	13.170
Sujeto 8					
Velocidades	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	9	12.2777	10.74360	3.2777	3.2777
Rápido	9	10.0560	1.115044	1.0560	1.0560
Lento	9	10.8100	3.27602	1.8100	1.80998
Regreso	18	24.7553	45.63448	6.7553	6.7553
Externo	60	50.8082	84.48992	9.1918	9.1918
		PROMEDIO	19.374978	2.959768066	2.95977

Fuente: Elaboración propia

Tabla 28 - Comparativa de los promedios de las métricas de regresión en distancia entre el modelo LSTM +FCN y LSTM +FCN (Secuencia a uno)

Comparativa de los promedios de las métricas de regresión en distancia entre el modelo LSTM +FCN y LSTM +FCN (Secuencia a uno)						
Modelo	Tamaño de lote	Épocas	Tamaño de ventana	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
LSTM + FCN	128	25	100	344.5545330	13.1284892	16.0453324
LSTM + FCN (Secuencia a uno)	128	25	100	19.37497794	2.959768066	2.959768066

Fuente: Elaboración propia

5.1.4 Contador de pasos Modelo LSTM+FCN (Secuencia a uno)

Tabla 29 - Contador de pasos de modelo LSTM+FCN (Secuencia a uno)

Modelo LSTM + FCN					
Prueba de Numero de Pasos					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 9					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	16	14.55027	2.101719	1.44973	1.44973
Rápido	14	14.15693	0.0246268	0.15693	0.15693
Lento	19	16.76173	5.009831	2.23827	2.23827
Regreso	33	26.92497	36.906016	6.07503	6.07503
Sujeto 2					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	15	13.81585	1.402197	1.18414	1.18414
Rápido	12	13.00000	1.00000	1.00000	1.00000
Lento	20	16.24057	14.13331	3.75943	3.75943
Regreso	31	28.06142	8.63520	2.93858	2.93858
Externo	106	50.90804	4236.96320	65.09196	65.09196
Sujeto 3					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	20	19.6239	0.141467	0.3761	0.3761
Lento	43	34.2235	77.027535	8.7765	8.7765
Regreso	50	47.1836	7.932109	2.8164	2.8164
Externo	146	61.8112	7087.754045	84.1888	84.1888

Modelo LSTM + FCN					
Prueba de Numero de Pasos					
Tamaño de lote = 128 Tamaño de ventana = 100 Épocas = 25					
Sujeto 4					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	16	23.53220	56.734018	7.53220	7.53220
Rápido	17	23.51751	42.477886	6.51751	6.51751
Lento	17	20.76406	14.168180	3.76406	3.76406
Regreso	34	25.57420	70.994168	8.42580	8.42580
Sujeto 5					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Rápido	13	18.95	35.358507	5.9463	5.9463
Lento	19	24.38	28.957792	5.3812	5.3812
Regreso	35	28.56	41.529947	6.4444	6.4444
Externo	86	57.40	817.809157	28.5974	28.5974
Sujeto 6					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	20	24.0867	16.70072	4.0867	4.0867
Rápido	20	20.6708	0.45003	0.6708	0.6708
Lento	22	22.2553	0.06519	0.2553	0.2553
Regreso	42	33.3224	75.30076	8.6776	8.6776
Externo	105	52.6485	2740.67587	52.3515	52.3515
Sujeto 7					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	28	26.0398	3.8425	1.960	1.960
Rápido	23	23.747	0.5582	0.747	0.747
Lento	29	24.943	16.4586	4.057	4.057
Externo	237	82.853	23761.4402	154.147	154.147
Sujeto 8					
Experimentos	Valores reales	Valores Calculados	Error cuadrático medio	Error absoluto medio	Raíz del Error cuadrático medio
Regular	18	19.9264	3.71094	1.9264	1.9264
Rápido	15	17.1167	4.48021	2.1167	2.1167
Lento	16	14.2988	2.89421	1.7012	1.7012
Regreso	33	25.8053	51.76338	7.1947	7.1947
Externo	124	54.5737	4,820.00918	69.4263	69.4263
		PROMEDIO	1259.583170	16.05656	16.05656

Fuente: Elaboración Propia

Se obtuvieron las métricas de entrenamiento de cada sujeto de los dos modelos para cada uno de los parámetros. En el parámetro en el modelo LSTM+ FCN en la prueba 1 se muestran en anexos 17-24, en la prueba 2 en los anexos 25-32 y en la prueba 3 en los anexos 33-40. En cuanto al modelo LSTM+CNN en la prueba 1 se muestran en los anexos 41-50, en la prueba 2 en los anexos 51-58 y en la prueba 3 en los anexos 59-66.

En el parámetro de distancia recorrida, en el modelo LSTM+ FCN en la prueba 1 se muestran en anexos 91-98, en la prueba 2 en los anexos 99-106 y en la prueba 3 en los anexos 107-114. En el modelo LSTM+CNN en la prueba 1 se muestran en los anexos 67-74, en la prueba 2 en los anexos 75-82 y en la prueba 3 en los anexos 83-90.

En el parámetro de tamaño de zancada, en el modelo LSTM+ FCN en la prueba 1 se muestran en los anexos 115-122, en la prueba 2 en los anexos 130-133 y en la prueba 3 en los anexos 134-137. En el modelo LSTM+CNN en la prueba 1 se muestran en los anexos 123-125, en la prueba 2 en los anexos 126-129 y en la prueba 3 en los anexos 138- 141. En el parámetro de numero de pasos, en el modelo LSTM+FCN, las métricas de entrenamiento se muestran en los anexos 143 – 148.

Durante el proceso de entrenamiento de cada sujeto, se evaluó la precisión de los dos modelos LSTM+CNN y LSTM+FCN de los diferentes parámetros. Los resultados revelaron una notable similitud en la precisión entre estos dos modelos en todos los sujetos analizados en los diferentes parámetros. En los promedios de precisión por modelos, en el parámetro de velocidad se muestran en los anexos 1 – 3, para el parámetro de distancia recorrida en los anexos 4 – 6 y en el parámetro de tamaño de zancada en los anexos 7 – 9.

En los promedios de precisión por pruebas, en el parámetro de velocidad se muestran en los anexos 10 - 11, en el de distancia recorrida en los anexos 12 – 13 y en el parámetro de tamaño de zancada en los anexos 15 -16.

Se observó que el modelo LSTM+FCN alcanzó la casi la misma precisión que el modelo LSTM+CNN en todas las pruebas. Estos hallazgos sugieren que ambos modelos son igualmente efectivos.

En el siguiente gráfico, se muestran un ejemplo de cómo se observan los resultados obtenidos del entrenamiento de los dos modelos, capturando la precisión que obtuvieron cada uno a la hora de realizar predicciones en el parámetro de velocidad.

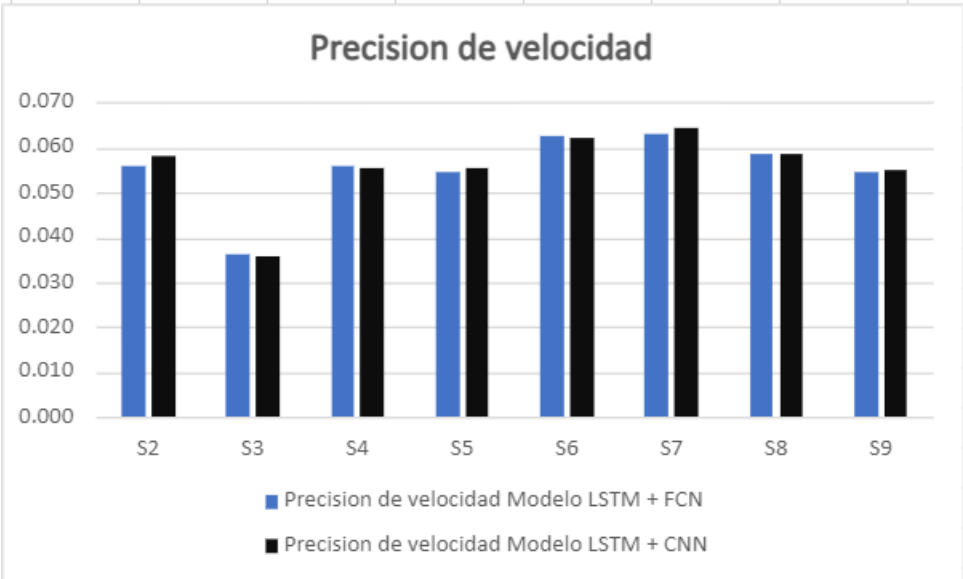


Ilustración 26 - Precisión del entrenamiento de los datos de velocidad del Modelo LSTM+FCN y Modelo LSTM+CNN

Fuente: Elaboración propia

5.2 DISCUSIÓN DE RESULTADOS

En esta sección, se analizarán y contextualizarán los resultados obtenidos en el estudio. Se explorarán las implicaciones, interpretaciones y conclusiones derivadas de los datos presentados, además de comparar los hallazgos con investigaciones previas. También se abordarán posibles limitaciones del estudio.

5.2.1 TAMAÑO DE ZANCADA

En el parámetro del tamaño de zancada se realizaron diversas pruebas las cuales se pueden observar en la Tabla 5.

5.2.1.1 Modelo LSTM+CNN

Los resultados predichos en el modelo LSTM+CNN de la prueba 1 en la Tabla 5 revelan que el sujeto 8 logró una mayor precisión en comparación con los demás sujetos, ya que presentó solo una diferencia mínima de 3 dígitos en relación con los valores reales.

En cuanto a las métricas de regresión en cada uno de los experimentos del sujeto 8 mostraron resultados muy cercanos a cero lo que indica que las predicciones del modelo son

prácticamente idénticas a los valores reales. Como se muestran en la Tabla 6, los experimentos que obtuvieron las mejores métricas de regresión están entre la caminata regular y lenta, en la caminata regular el MAE obtuvo un valor de 0.02231m, el MSE un valor de 0.0006 y el RMSE un valor de 0.02351m y en la caminata lenta, el MSE un valor de 0.0006m, el MAE un valor de 0.02185m.

No obstante, se observaron desempeños menos acertados en algunos de los sujetos, específicamente en los sujetos 7 y 5, quienes presentaron diferencias de aproximadamente 10 dígitos entre las predicciones y los valores reales. En el caso del sujeto 5, la caminata de regreso fue la más precisa con una predicción de 0.52 m frente al valor real de 0.51m. A pesar de que los sujetos 7 y 5 registraron valores bajos en las métricas de regresión, no se compararon con los resultados de los demás sujetos que se aproximaban más a cero.

En la segunda prueba que se muestra en la Tabla 7 y tercera prueba en la Tabla 8 del modelo LSTM+CNN, de igual forma el sujeto 8 fue el de mejor predicción a comparación con los demás sujetos con buenos resultados en las métricas de regresión al acercarse a cero. El menos acertado en la prueba 2 fue el sujeto 5 con solo una predicción cercana en el experimento de caminata en el exterior. En la prueba 3 fue el sujeto 7 que fue muy variado en las predicciones y ninguno tuvo una diferencia de mínimo 5 dígitos.

5.2.1.2 Modelo LSTM + 4FCN

En el modelo LSTM + FCN en la prueba 1, los sujetos que obtuvieron las mejores predicciones fue el sujeto 8 y 6 como se muestra en la Tabla 9, donde también obtuvieron las mejores métricas de regresión, en el sujeto 8 los únicos que tuvieron una diferencia de más de 5 dígitos fueron la caminata regular y la de exterior y en el sujeto 6 solo hubo un experimento que obtuvo una diferencia de más de 5 dígitos la cual fue la caminata de regreso en las predicciones. El sujeto con menor precisión en este caso fue el sujeto 7 ya que al momento de predecir surgieron varias variaciones en los resultados.

En la prueba 2, los sujetos que obtuvieron las mejores predicciones fueron el sujeto 8 y 6 como se muestra en la Tabla 10, no obstante, a diferencia de la prueba 2, el sujeto 6 en ninguna de las predicciones se sobrepasó de los 5 dígitos. Las métricas de estos dos sujetos salieron

menores por lo tanto brindan resultados precisos. En esta prueba 2, también el sujeto 7 es el que se encuentra más variado en cuestión de las predicciones.

En la prueba 3, el sujeto que obtuvo la mejor precisión fue el sujeto 6 con las mejores métricas entre todos los sujetos, su precisión fue mejorando en cada prueba, aun así, se debe mencionar que el sujeto 8 también cuenta con predicciones precisas como se muestra en la Tabla 11.

En cada una de las pruebas llevadas a cabo, destacó la precisión de los sujetos 8 y 6. Estos participantes presentaron diferencias mínimas entre los valores reales en los diferentes experimentos. En algunos casos, incluso lograron predicciones exactamente iguales a los valores reales, y sus métricas de precisión fueron consistentemente más bajas, lo que sugiere que sus estimaciones estuvieron notablemente cerca de los valores reales.

Tanto en el modelo LSTM+ FCN como en el modelo LSTM + CNN se calculó el promedio total de error medio cuadrático y el error medio absoluto de las predicciones de cada sujeto. Al calcular los promedios totales de las métricas de cada prueba mostrada en la Tabla 11, se observa en el modelo LSTM+ FCN que la prueba que obtuvo las mejores métricas de regresión es la prueba 3.

El cálculo de MAE obtuvo un valor de 0.01570m, esto se debe a que estuvieron más cerca de los valores reales en comparación con las otras pruebas como también en el MSE con un valor de 0.000798 m esto indica que, en promedio, las predicciones en la prueba 3 tuvieron errores cuadráticos más pequeños y en el RMSE con un valor de 0.020426 m, pero considerando la Raíz cuadrada. En el modelo LSTM+CNN que obtuvo mejor MAE fue la prueba 1, con un valor de 0.0549 m. En el MSE, el mejor promedio la obtuvo la prueba 3 con un valor de 0.008292 m, asimismo también fue el mejor promedio total en el RMSE con un valor de 0.06722m.

Entre los dos modelos, destaca que el modelo LSTM+FCN obtuvo los mejores resultados y métricas más sólidas. En cuanto a las pruebas específicas, tanto el modelo LSTM+FCN como el LSTM+CNN demostraron un rendimiento excepcional en la prueba 3. Esto puede sugerir que la prueba 3 podría haber tenido características de datos que se adaptan especialmente bien a ambos modelos o que los parámetros utilizados para la prueba 3 eran particularmente adecuados para estos modelos en comparación con las otras pruebas.

5.2.1.3 Modelo tradicional vs modelo basado en LSTM

Al haber comparado los resultados de los modelos LSTM+FCN y LSTM+CNN con los valores reales, también podemos contextualizar estos hallazgos con el estudio previo realizado por (Reyes Leiva et al., 2023). Según este estudio, el tamaño de zancada, el promedio del error absoluto medio para el método tradicional fue de 0.0459 ± 0.0369 metros para el método TWS y de 0.0737 ± 0.0468 metros para el método THS. Esta comparación proporciona una referencia importante para evaluar el desempeño de los modelos LSTM+FCN y LSTM+CNN en relación con los enfoques convencionales.

Utilizando el modelo de LSTM+FCN se encontró que el error absoluto medio de ocho sujetos es de 0.015 ± 0.014 metros y de 0.054 ± 0.039 metros utilizando el modelo LSTM+CNN. Demostrando que, para el tamaño de zancada, el error más pequeño se consigue utilizando el modelo LSTM+FCN.

5.2.2 VELOCIDAD

En la evaluación del parámetro de velocidad, se llevaron a cabo múltiples pruebas registradas en la Tabla 5, con el objetivo de realizar una comparativa entre los modelos LSTM+CNN y LSTM+FCN.

5.2.2.1 Modelo LSTM+CNN

En el caso del modelo LSTM+CNN presentado en la Tabla 13 se observaron resultados notables. En la prueba 1, el sujeto que demostró la mejor precisión fue el sujeto 4, aunque se registró una diferencia significativa en el experimento de caminata lenta. A pesar de esta diferencia, el sujeto 4 obtuvo resultados superiores en comparación con los otros sujetos. En contraste, el sujeto 2 presentó la menor precisión, ya que los valores obtenidos se encontraban en un rango mayor de 0 a 1, lo que dificulta las predicciones debido a la normalización del modelo. El sujeto 4, por otro lado, logró resultados con métricas de regresión más bajas, lo que indica un rendimiento destacado. Además, los demás sujetos también obtuvieron métricas de regresión que reflejan un buen desempeño en general.

En la segunda prueba, se observó un cambio en el rendimiento de los sujetos en la tabla 14. El sujeto 3 destacó al lograr la mejor precisión en esta ocasión. En contraste con la prueba anterior, el sujeto menos preciso fue el sujeto 2, lo que sugiere que posiblemente el aumento en

el tamaño del lote no fue adecuado para la base de datos de este sujeto en particular. Los sujetos 6, 7 y 8 siguieron al sujeto 3 en términos de precisión, obteniendo resultados sólidos en las métricas de regresión. Estos hallazgos muestran cómo los cambios en los parámetros pueden afectar significativamente el desempeño de los sujetos en diferentes pruebas.

En la tercera prueba la cual se muestra en la Tabla 15, el sujeto que destacó por su mayor precisión fue el sujeto 7, lo cual es notable ya que había obtenido la menor precisión en cuanto al parámetro de tamaño de zancada. Sin embargo, al aumentar el número de épocas y trabajar con la base de datos de velocidad, demostró una notable mejora en sus capacidades predictivas. Por otro lado, el sujeto 8 que había obtenido la mayor precisión en relación al tamaño de zancada, obtuvo la menor precisión en la prueba 3 en cuanto a velocidad. Estos resultados resaltan la influencia significativa que pueden tener los cambios en los parámetros y las bases de datos en el rendimiento de los sujetos en distintas pruebas.

5.2.2.2 Modelo LSTM+4FCN

En el modelo LSTM+FCN, se destacan los excelentes resultados en las métricas a lo largo de todas las pruebas presentadas en las Tablas 16, 17 y 18. Los sujetos que sobresalen por su desempeño son el 4 y el 6, quienes lograron las mejores métricas en las tres pruebas realizadas. Esto se refleja en los valores extremadamente bajos y cercanos a cero del MSE lo que indica una precisión sobresaliente en las predicciones y una sólida adaptación del modelo a los datos. Además, los sujetos 8 y 7 también obtuvieron resultados destacados en las tres pruebas, demostrando una eficaz capacidad predictiva. En el modelo LSTM+FCN, se observó un empate en los resultados de cada sujeto, lo que implica que varios sujetos lograron la mejor precisión en diferentes experimentos.

En el cálculo de promedio total de las métricas en las pruebas que se muestran en la Tabla 19, el modelo LSTM + FCN, el MAE con un valor de 0.001012 m y el MSE con un valor de 0.01808 m obtuvieron los mejores resultados en la prueba 1, mientras que en RMSE con un valor de 0.02800 m, la prueba que obtuvo mejor resultado fue la de prueba 3, lo que indica que la prueba 1 fue la más precisa entre las tres pruebas.

Así mismo en la Tabla 19, el modelo LSTM + CNN, el MAE que obtuvo un mejor valor de 0.0442306 m fue la prueba 1 y el MSE con un valor de 0.0141335 m obtuvo un mejor resultado

en la prueba 3, mientras que en RMSE con un valor de 0.0675164 m también fue en la prueba 1, lo que indica que la prueba 1 fue la más precisa entre las tres pruebas.

Ambos modelos mostraron resultados de alta precisión a pesar de que algunos valores reales se encontraban por encima del rango de 0 a 1 en diferentes experimentos. En el modelo LSTM+FCN, las métricas de evaluación fueron consistentemente bajas, lo que indica un alto nivel de precisión. En el caso del modelo LSTM+CNN, en general, también ofreció resultados notables en términos de precisión.

5.2.2.3 Método tradicional vs modelo basado en LSTM

En la investigación (Reyes Leiva et al., 2023) se encontró que para el parámetro de velocidad el error absoluto medio fue de 0,104 ± 0,0688 m/s para el TWS y de 0,0703 ± 0,0713 m/s para el THS. En nuestra investigación se obtuvo un error absoluto medio de 0.018 ± 0.013 m/s con el modelo LSTM+FCN y un error absoluto medio de 0.044 ± 0.017 m/s con el modelo LSTM+CNN siendo estos errores más pequeños que los valores obtenidos en la investigación previa utilizando el método tradicional.

5.2.3 DISTANCIA RECORRIDA

5.2.3.1 Modelo LSTM+CNN

Para poder optimizar el rendimiento del modelo de distancia se realizaron las distintas pruebas mencionadas en la Tabla 5 y de esta forma encontrar la mejor configuración para el modelo CNN+LSTM para el parámetro de distancia comparado con los valores reales establecidos en la base de datos.

Como se puede observar en la Tabla 20 la prueba número uno es en la que se obtuvieron mejores resultados de predicción de distancia, siendo esta prueba la que obtuvo en promedio los mejores resultados para predicciones y los errores más pequeños en comparación con los demás.

Dentro de esta prueba, el sujeto que obtuvo las mejores predicciones fue el sujeto 3. Sin embargo, el sujeto 9 también cuenta con muy buenas predicciones y es este sujeto el que tiene mejores métricas de regresión como se puede observar en la Tabla 21. Se puede observar que las métricas más pequeñas se encuentran en los experimentos de velocidad regular y velocidad rápida, siendo el error cuadrático medio de estas de 0.055 metros para el sujeto 9 en velocidad regular y 0.044 metros para el sujeto 3 en velocidad rápida, mientras que estas se vuelven más

grandes entre mayor es la distancia que recorre el sujeto, ya que este error aumentó a 0.1368 metros en el experimento de retorno del sujeto 9 en el que se recorrió una distancia de 20 metros y a 4.5 en el experimento llevado a cabo en una distancia de 60 metros en el exterior del sujeto 3.

El sujeto que tuvo una mayor variación en las predicciones dentro de esta prueba fue el sujeto 8. Sin embargo, fue el sujeto 7 el que cuenta con un error más alto teniendo una métrica de error cuadrático medio de 7.99 m en velocidad regular.

Por otro lado, la prueba 2 mostrada en la Tabla 21 fue la que obtuvo los resultados más variados en el modelo LSTM+CNN. En esta prueba se puede observar que la diferencia entre los valores calculados y los valores predichos por el modelo varían bastante, esto muestra señales de una adaptación excesiva del modelo, ya que el rendimiento del modelo es superior durante la etapa de entrenamiento que en la etapa de evaluación. Lo que significa que el modelo está aprendiendo los valores de entrenamiento tan bien que está impactando de una forma negativa a su habilidad de generalizar información que no ha visto anteriormente.

5.2.3.2 Modelo LSTM+4FCN

El modelo LSTM+4FCN obtuvo resultados significativos mostrados en las Tabla 23 correspondiente a la prueba 1, la Tabla 24 correspondiente a la prueba 2 y la Tabla 25 correspondiente a la prueba 3 en donde mostró un porcentaje de error más pequeño que el modelo LSTM+CNN en las predicciones de distancia de cada sujeto. En ambos modelos se mantiene que la primera prueba es la que muestra un menor porcentaje de error, mientras que la segunda prueba es donde se presenta un mayor error mostrada en la Tabla 23. El Sujeto 3 sigue siendo el sujeto que tiene las predicciones más cercanas al valor real y el Sujeto 9 también es el sujeto que tiene los errores más pequeños en este modelo. Sin embargo, en este caso, el error cuadrático es menor en el experimento de velocidad rápida tanto en el sujeto 9 como en el sujeto 3 siendo este de 0.0093 metros para el sujeto 9 y de 0.085 para el sujeto 3. De igual forma, en este caso el sujeto 7 es el que tiene el error más grande, llegando a tener un error cuadrático medio de 2.45 en la velocidad regular.

5.2.3.3 Método tradicional vs modelo basado en LSTM

En el estudio realizado previamente (Reyes Leiva et al., 2023) se encontró que el error absoluto medio global para el parámetro de distancia fue de 1,78 ± 1,80 m para el método TWS

y de 3,20 ± 3,38 m para el método THS. Señalando que existe una variación en la precisión de los valores medios al cambiar las condiciones interiores/exteriores; se produce un aumento del error absoluto para el THS de 2,03 ± 1,92 a 7,9 ± 2,74 m, mientras que el error del TWS se mantiene prácticamente sin cambios.

Siguiendo la misma metodología se encontró que el valor del error absoluto medio global de los ocho sujetos utilizados en los 732 m recorridos fue de 13.13 ± 7.93 m para el modelo LSTM+4FCN y de 21.37 ± 9.25 para el modelo LSTM+CNN. Al cambiar las condiciones interiores/exteriores se produce un aumento del error absoluto de 20.17 ± 9.58 m a 27.21 ± 4.16 m en LSTM+CNN y de 12.15 ± 9.87 m a 17.84 ± 5.01 m en LSTM+4FCN. Siendo estos valores mayores que los obtenidos previamente con el método tradicional.

En la tabla 26 se puede observar que los errores más pequeños se alcanzan al utilizar la primera prueba con el modelo LSTM+FCN. Con este modelo específicamente se encuentran valores más precisos que con el modelo LSTM+CNN con el cual el error aumenta significativamente. El MSE más grande se obtiene con la segunda prueba del modelo LSTM+CNN siendo esta la prueba menos precisa al momento de la evaluación de la distancia recorrida por los sujetos.

5.2.3.4 Distancia utilizando modelo Alternativo LSTM+FCN (Secuencia a Uno)

Debido a que se observaron áreas con errores sustanciales, especialmente en predicciones de movimientos lentos se decidió probar con un enfoque diferente para el parámetro de distancia. Por eso, se utilizó el modelo Seq2One (Secuencia a uno), a pesar de su estructura simplificada, mostró resultados competitivos e incluso superiores en ciertos aspectos. Sin embargo, su capacidad para predecir distancias precisas varió considerablemente entre los sujetos.

La Raíz del Error Cuadrático Medio (RMSE) es una métrica comúnmente utilizada para evaluar la precisión de un modelo de predicción. Un RMSE más bajo indica que el modelo tiene un error menor, y por lo tanto es mejor. Como se puede visualizar en la Tabla 27 y 28, el Modelo Alternativo de secuencia a uno tiene un rendimiento mucho mejor que el Modelo original, con un RMSE de 2.96 en comparación con 16.05656 del Modelo original.

El modelo analizado en esta investigación es menos preciso ya que este tiene un error de 2.96 ± 3.3, esta baja a 2.03 ± 2.25 tomando en consideración solo los experimentos internos y

sube a 7.43 ± 4.11 en condiciones de exterior. Al comparar con el estudio de Reyes Leiva et al., 2023, es evidente que aún hay margen de mejora en los modelos propuestos. El método TWS, con un error absoluto medio (EAM) global de 1,78 m, superó a ambos modelos en términos de precisión.

5.2.4 Contador de Pasos

5.2.4.1 Modelo LSTM+4FCN (Secuencia a uno)

Para la predicción del parámetro del contador de pasos se ha utilizado una versión diferente del modelo LSTM+4FCN con un enfoque distinto. Se generó un código con enfoque secuencia a uno que genera un único valor para todo el experimento. Este enfoque es más estable y menos sensible al ruido de los datos, pero si no está bien ajustado, puede llevar a errores, especialmente en escenarios que no están bien representados en los datos de entrenamiento. De igual forma, si hay características importantes en la secuencia que son cruciales para la predicción, este modelo podría pasarlas por alto.

Se puede observar en la Tabla 29 que la evaluación del rendimiento del algoritmo de conteo de pasos presenta resultados variados según las condiciones del experimento. Para experimentos llevados a cabo en condiciones interiores y a diferentes velocidades, como son los llamados: "Regular", "Rápido", "Lento" y "Retorno", los errores absolutos medios y los errores cuadráticos medios obtenidos fueron generalmente bajos, lo que sugiere que el algoritmo tiene una capacidad robusta de predicción en estas condiciones. En contraste, para experimentos en condiciones exteriores, como el denominado "Externo", se observó un marcado incremento en los errores, indicando una posible limitación en la capacidad de generalización del modelo en tales escenarios.

La discrepancia en el experimento "Externo" indica que puede haber una falta de representatividad de estos datos en el conjunto de datos de entrenamiento. Los experimentos "externos" para varios sujetos muestran errores significativamente más altos en comparación con otros experimentos. Estos experimentos externos son representativos de "nuevos datos", el alto error sugiere que el modelo podría estar sobreajustando a los experimentos realizados en el interior y no generalizando bien a los experimentos en el exterior.

Una posible causa de los resultados menos precisos en los experimentos exteriores es la limitación en la cantidad y diversidad de datos de entrenamiento relacionados con escenarios exteriores. Ya que el modelo se entrenó con un conjunto limitado de datos exteriores, podría no haber aprendido adecuadamente las características y variabilidades específicas de los escenarios exteriores, llevando a predicciones inexactas.

5.2.4.2 Modelo Tradicional vs Modelo basado-LSTM

En el estudio de referencia de (Reyes Leiva et al., 2023) se culminó que "El algoritmo de recuento de pasos demostró que los algoritmos de recuento de pasos funcionaban con precisión, con un error medio de 1,48 ± 1,56.". En comparación con los resultados obtenidos en el modelo LSTM+4FCN con un enfoque de secuencia a uno, este cuenta con una mayor variabilidad habiendo obtenido un valor 16.06 ± 32.11 haber obtenido las métricas de cada uno de los experimentos. Sin embargo, como se puede observar en la Tabla 29, el experimento que presentó el error más grande fueron los experimentos realizados en el exterior, mientras que los demás experimentos tenían un error aceptable. Solo tomando en cuenta los experimentos realizados en el interior se llegó a un error absoluto medio de 4.2095 ± 2.8149. Estos resultados siguen sin ser tan precisos como los que se adquirieron en el estudio de referencia.

Dada la alta variabilidad en los resultados del experimento "Externo" y los errores significativamente mayores en comparación con los otros experimentos, es plausible que el modelo no esté bien entrenado para manejar los datos del experimento "Externo". Una posible razón es que el conjunto de datos no tenía suficientes ejemplos similares al experimento "Externo". Dado que este experimento fue realizado en condiciones muy distintas a los otros cuatro experimentos y que la base de datos cuenta solamente con cinco muestras de este experimento, añadiendo que estas muestras son variables entre sí, esta puede ser la razón por la que el modelo está teniendo problemas en predecir el parámetro de contador de pasos en dicho experimento.

VI. CONCLUSIONES

Por medio de este estudio se logró comprobar que los algoritmos basados en LSTM pueden predecir con precisión los parámetros de la marcha en personas con discapacidad visual. Siendo el modelo LSTM+FCN el que obtuvo métricas de evaluación más precisas. La precisión de los algoritmos basados en LSTM fue comparable a la del método tradicional de cálculo, habiendo analizado los errores absolutos de investigaciones anteriores con los hallazgos obtenidos en esta investigación se concluyó que este nuevo método cuenta con un error más pequeño al momento de predecir la velocidad de la marcha y el tamaño de zancada en comparación con el método tradicional utilizado por otros investigadores. Por lo tanto, los algoritmos con base LSTM pueden proporcionar predicciones más precisas de los parámetros de la marcha en relación con el método tradicional, sin embargo, se concluyó que se debían realizar más pruebas para parámetros como distancia que obtuvo un error mayor con este método. Es por esto por lo que se realizó un enfoque distinto de secuencia a uno para poder evaluar este parámetro junto con el contador de pasos. Por lo tanto, se concluyó que el enfoque de secuencia a uno es adecuado para poder realizar la predicción de los parámetros de distancia y contador de pasos. Sin embargo, este enfoque a pesar de ser estable tiene una menor capacidad de adaptabilidad por lo que se necesita de una base de datos mayor para poder tener un mayor alcance y mejorar su capacidad de predicción. Estos resultados sugieren que los algoritmos basados en LSTM podrían utilizarse para desarrollar nuevas intervenciones de rehabilitación de la marcha para personas ciegas en un futuro.

El proceso de etiquetado de la base de datos fue exhaustivo, ya que implicó la consolidación de los datos recopilados de todos los sujetos con sus respectivas etiquetas, con la única excepción de los datos destinados a evaluación. Se realizaron, en total, 32 bases de datos consolidadas, 8 bases de datos para cada sujeto en relación con tres parámetros clave: tamaño de zancada, distancia y velocidad de marcha, Sin embargo, antes de consolidar la base de datos, se debieron hacer ciertas modificaciones ya que algunos los archivos CSV otorgados por los estudios previos de los diferentes experimentos no concordaban con la evidencia audiovisual por lo tanto se debió corregir para obtener una base de datos más exacta. Esta base de datos

desempeñó un papel fundamental en la predicción y análisis de los resultados, permitiéndonos obtener información valiosa sobre el rendimiento de los sujetos en cada uno de estos parámetros.

Para lograr una predicción precisa de los parámetros de marcha, se desarrolló un algoritmo que ha demostrado ofrecer un eficiente nivel de precisión en la estimación de estos parámetros. Este algoritmo nos ha proporcionado tanto los valores máximos como los mínimos, así como los valores promedio y la primera predicción, brindando así un espectro completo de información que enriqueció la comprensión de la marcha humana. Además de las predicciones precisas, este proceso también nos permitió calcular métricas de regresión esenciales, como el error absoluto medio, el error cuadrático medio y la raíz del error cuadrático medio. Estas métricas han servido como indicadores cruciales de la calidad y el rendimiento de nuestros modelos.

Gracias al aprovechamiento de la base de datos configurada y etiquetada, que ha sido procesada a través de modelos de aprendizaje profundo, específicamente los modelos LSTM + FCN y LSTM + CNN. Estos modelos han demostrado ser herramientas eficientes en la predicción de parámetros de marcha, éstos al momento de entrenarlos obtuvieron métricas de entrenamientos significativas como los valores de precisión y pérdida de validación y el error cuadrático medio los cuales obtuvieron valores menores lo que indica un buen porcentaje de entrenamiento. El entrenamiento con datos etiquetados nos ha permitido validar y ajustar nuestros modelos con base LSTM de manera efectiva, lo que ha llevado a una mayor confiabilidad en las predicciones.

Se comprobó la flexibilidad de los modelos LSTM+CNN y LSTM+FCN ya que permiten ajustar y afinar sus híper parámetros de manera efectiva. Esto ha resultado en mejoras significativas en la precisión de las predicciones de parámetros de marcha, como el tamaño de zancada, la velocidad y la distancia. Además, al realizar las pruebas con diferentes configuraciones en tamaño de ventana, tamaño de lote y época, hemos identificado cuáles de ellos tienen un impacto crítico en la calidad de las predicciones. Tanto el modelo LSTM+ CNN como en el modelo LSTM + FCN en el tamaño de zancada se notó que el que más le beneficio fue al momento de incrementar las épocas y al aumentar las épocas permitió que el modelo se ajuste a los datos de entrenamiento y mejore su capacidad de converger hacia una solución óptima y obtener resultados más precisos, mientras que en el velocidad y distancia se les beneficioso más los valores

ampliamente mente utilizados en previos estudios los cuales fueron valores razonables para comenzar el entrenamiento de los modelos.

Tras analizar los resultados obtenidos, se ha podido evidenciar diferencias significativas entre el método tradicional y el método de Deep Learning en el contexto de la predicción de parámetros de marcha de personas con discapacidad visual. El Deep Learning, con su capacidad para manejar grandes volúmenes de datos y aprender patrones complejos, ha demostrado ser más eficiente en la predicción de parámetros como el tamaño de zancada y la velocidad de marcha en comparación con el método tradicional. Sin embargo, con otros parámetros de marcha como la distancia y el conteo de pasos se deben realizar pruebas adicionales con una base de datos más extensa. De esta forma, se podrá concluir si el enfoque utilizado puede llegar a ser eficiente para estos parámetros. Al realizar los dos enfoques para estos últimos parámetros resultó evidente la importancia del preprocesamiento de los datos de entrada para la eficiencia de las predicciones realizadas. Es imperativo, por lo tanto, no solo centrarse en la técnica en sí, sino también en cómo se obtienen y se preparan los datos para el análisis, ya que esto puede afectar en gran medida los resultados finales.

VII. RECOMENDACIONES Y LIMITACIONES

- El estudio se vio limitado por el pequeño tamaño de la muestra obtenido debido a que se centró en un grupo de personas muy específico como lo son las personas con discapacidad visual. Deberían realizarse investigaciones futuras con un tamaño de muestra mayor y en un entorno más grande para poder confirmar los hallazgos de este estudio.
- Desarrollar un código de prueba utilizando métodos de simulación para evaluar el desempeño del algoritmo entrenado en diferentes escenarios.
- Realizar más investigaciones con otras bases de datos para confirmar los hallazgos de esta investigación e investigar otras configuraciones de los modelos que puedan beneficiar a otros parámetros de la marcha como la distancia.
- Investigar sobre otros métodos de etiquetado de datos y si estos pueden mejorar las predicciones de otros parámetros de la marcha.

VIII. APLICABILIDAD/IMPLEMENTACIÓN

En esta sección, se abordarán las aplicaciones prácticas y posibles implementaciones de los resultados de esta investigación. La evaluación de algoritmos basados en LSTM para la predicción de parámetros de la marcha en personas con discapacidad visual representa una importante contribución científica con el potencial de influir en diversas áreas de estudio para el desarrollo de tecnologías asistenciales para personas con discapacidad visual.

Tecnologías de apoyo para personas con discapacidad visual: Los algoritmos basados en LSTM entrenados pueden integrarse en tecnologías de asistencia diseñadas para ayudar a los individuos con discapacidad visual a navegar por su entorno de forma segura e independiente.

Rehabilitación y asistencia sanitaria: Estos algoritmos pueden contribuir en aplicaciones en programas de rehabilitación para personas con problemas de movilidad y pueden ayudar a los profesionales sanitarios a evaluar los patrones de la marcha y hacer un seguimiento de los progresos.

Detección y prevención de caídas: Mediante el análisis de los parámetros de la marcha, los algoritmos basados en LSTM entrenados pueden contribuir a los sistemas de detección de caídas que alertan automáticamente a los cuidadores o a los servicios de emergencia cuando se detecta una caída, previniendo potencialmente lesiones en la población anciana o con discapacidad visual.

Investigación del análisis de la marcha: Los investigadores y científicos que trabajan en los campos de la biomecánica, las ciencias del movimiento humano y la rehabilitación pueden revisar los hallazgos obtenidos en esta investigación con estos algoritmos y poder realizar estudios complementarios sobre estos para poder realizar un análisis de la marcha y mejorar las metodologías científicas relacionadas con la movilidad.

Programas de rehabilitación personalizados: Los profesionales de rehabilitación pueden crear programas personalizados de rehabilitación de la marcha adaptados a las necesidades específicas de cada paciente donde estos no van a requerir de movilizarse de un lugar a otro para poder recibir su rehabilitación en O&M utilizando los modelos basados en LSTM entrenados.

IX. EVOLUCIÓN DE TRABAJO FUTURO

En nuestra investigación, hemos realizado una evaluación exhaustiva de los parámetros de marcha utilizando modelos LSTM+CNN Y LSTM + FCN. Si bien hemos logrado avances significativos en la predicción de parámetros como la velocidad y el tamaño de zancada, reconocemos que aún existen desafíos por abordar, especialmente en lo que respecta a la predicción del número de pasos y la distancia recorrida con el enfoque de secuencia a uno por lo tanto planteamos dos líneas de acción para el futuro. En primer lugar, se propone llevar a cabo estudios adicionales con una mayor cantidad de datos para perfeccionar la precisión en la predicción de la distancia recorrida y evitar que los modelos se sobreajuste. En segundo lugar, agregar a la base de datos una mayor cantidad de pruebas conducidas en situaciones en el exterior para poder disminuir el error en la predicción del número de pasos durante la marcha.

X. ANEXOS

Anexo 1: Promedio de precisión de velocidad en prueba 1 por modelos

Precisión de velocidad de prueba 1		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.056	0.058
S3	0.036	0.036
S4	0.056	0.055
S5	0.054	0.055
S6	0.063	0.062
S7	0.063	0.064
S8	0.059	0.059
S9	0.055	0.055

Fuente: Elaboración propia

Anexo 2: Promedio de precisión de velocidad en prueba 2 por modelos

Precisión de velocidad en Prueba 2		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.058	0.059
S3	0.036	0.035
S4	0.055	0.055
S5	0.054	0.055
S6	0.061	0.062
S7	0.064	0.064
S8	0.058	0.059
S9	0.054	0.055

Fuente: Elaboración propia

Anexo 3: Promedio de precisión de velocidad en prueba 3 por modelos

Precisión de velocidad en Prueba 3		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.059	0.058
S3	0.035	0.036
S4	0.055	0.055
S5	0.055	0.054
S6	0.062	0.063
S7	0.064	0.065
S8	0.058	0.059
S9	0.056	0.055

Fuente: Elaboración propia

Anexo 4: Promedio de precisión de distancia prueba 1 por modelos

Precisión de distancia		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.215	0.214
S3	0.194	0.184
S4	0.208	0.207
S5	0.170	0.160
S6	0.281	0.271
S7	0.061	0.060
S8	0.248	0.247
S9	0.162	0.152

Fuente: Elaboración propia

Anexo 5: Promedio de precisión de distancia prueba 2 por modelos

Precisión de distancia en Prueba 2		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.213	0.215
S3	0.183	0.184
S4	0.207	0.207
S5	0.158	0.159
S6	0.269	0.271
S7	0.060	0.060
S8	0.246	0.247
S9	0.149	0.151

Fuente: Elaboración propia

Anexo 6: Promedio de precisión de distancia prueba 3 por modelos

Precisión de distancia en Prueba 3		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.214	0.213
S3	0.183	0.185
S4	0.206	0.207
S5	0.159	0.159
S6	0.272	0.272
S7	0.060	0.060
S8	0.248	0.248
S9	0.150	0.152

Fuente: Elaboración propia

Anexo 7: Promedio de precisión en tamaño de zancada de prueba 1 por modelos

Precisión de tamaño de zancada		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.055	0.056
S3	0.032	0.032
S4	0.047	0.052
S5	0.055	0.054
S6	0.059	0.060
S7	0.060	0.060
S8	0.069	0.068
S9	0.093	0.093

Fuente: Elaboración propia

Anexo 8: Promedio de precisión en tamaño de zancada de prueba 2 por modelos

Precisión de tamaño de zancada en Prueba 2		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.050	0.055
S3	0.032	0.032
S4	0.052	0.052
S5	0.054	0.055
S6	0.059	0.059
S7	0.060	0.060
S8	0.068	0.068
S9	0.093	0.093

Fuente: Elaboración propia

Anexo 9: Promedio de precisión en tamaño de zancada de prueba 3 por modelos

Precisión de tamaño de zancada en Prueba 3		
Sujetos	Modelo LSTM + FCN	Modelo LSTM + CNN
S2	0.049	0.055
S3	0.032	0.031
S4	0.052	0.053
S5	0.055	0.055
S6	0.058	0.060
S7	0.061	0.061
S8	0.069	0.069
S9	0.094	0.093

Fuente: Elaboración propia

Anexo 10 Precisión de velocidad (metros /segundo) en modelo LSTM+FCN por prueba

Modelo LSTM + FCN			
Precisión de velocidad velocidad (metros /segundo)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.056	0.058	0.059
S3	0.036	0.036	0.035
S4	0.056	0.055	0.055
S5	0.054	0.054	0.055
S6	0.063	0.061	0.062
S7	0.063	0.064	0.064
S8	0.059	0.058	0.058
S9	0.055	0.054	0.056

Fuente: Elaboración propia

Anexo 11: Precisión de velocidad (metros /segundo) en modelo LSTM+CNN por prueba

Modelo LSTM + CNN			
Precisión de velocidad (metros /segundo)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.058	0.059	0.058
S3	0.036	0.035	0.036
S4	0.055	0.055	0.055
S5	0.055	0.055	0.054
S6	0.062	0.062	0.063
S7	0.064	0.064	0.065
S8	0.059	0.059	0.059
S9	0.055	0.055	0.055

Fuente: Elaboración propia

Anexo 12: Precisión de distancia (metros) en modelo LSTM+FCN por prueba

Modelo LSTM + FCN			
Precisión de distancia (metros)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.215	0.213	0.214
S3	0.194	0.183	0.183
S4	0.208	0.207	0.206
S5	0.170	0.158	0.159
S6	0.281	0.269	0.272
S7	0.061	0.060	0.060
S8	0.248	0.246	0.248
S9	0.162	0.149	0.150

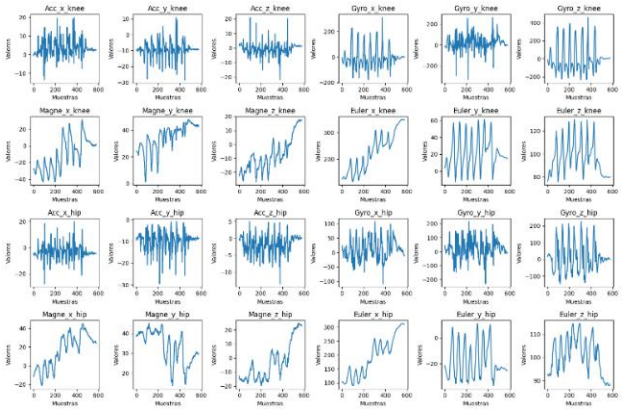
Fuente: Elaboración propia

Anexo 13: Precisión de distancia (metros) en modelo LSTM+CNN

Modelo LSTM + CNN			
Precisión de distancia (metros)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.214	0.215	0.213
S3	0.184	0.184	0.185
S4	0.207	0.207	0.207
S5	0.160	0.159	0.159
S6	0.271	0.271	0.272
S7	0.060	0.060	0.060
S8	0.247	0.247	0.248
S9	0.152	0.151	0.152

Fuente: Elaboración propia

Anexo 14: Señales adquiridas del sujeto 8



Fuente: Elaboración propia utilizando Octave

Anexo 15: Precisión de tamaño de zancada (metros) en modelo LSTM+CNN por prueba

Modelo LSTM + CNN			
Precisión de tamaño de zancada (metros)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.056	0.055	0.055
S3	0.032	0.032	0.031
S4	0.052	0.052	0.053
S5	0.054	0.055	0.055
S6	0.060	0.059	0.060
S7	0.060	0.060	0.061
S8	0.068	0.068	0.069
S9	0.093	0.093	0.093

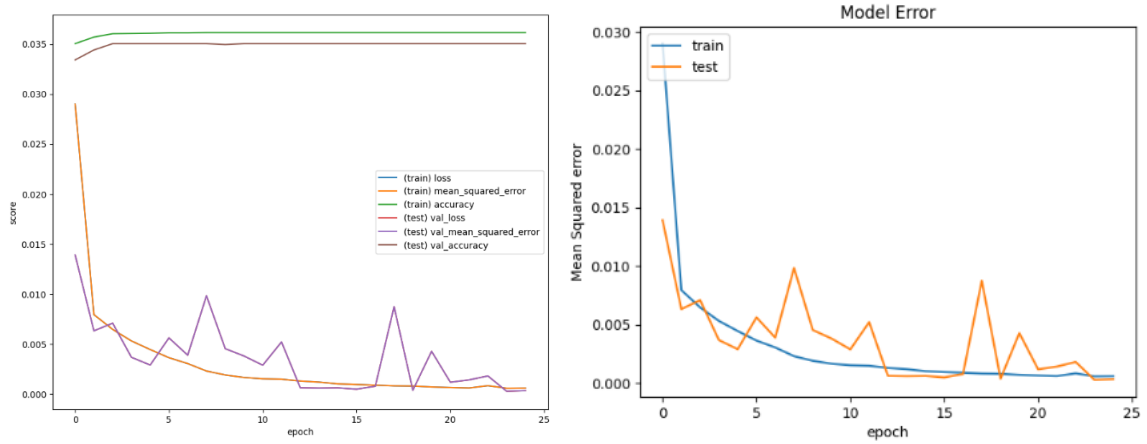
Fuente: Elaboración propia

Anexo 16: Precisión de tamaño de zancada (metros) en modelo LSTM+FCN por prueba

Modelo LSTM + FCN			
Precisión de tamaño de zancada (metros)			
Sujetos	Prueba 1	Prueba 2	Prueba 3
S2	0.055	0.050	0.049
S3	0.032	0.032	0.032
S4	0.047	0.052	0.052
S5	0.055	0.054	0.055
S6	0.059	0.059	0.058
S7	0.060	0.060	0.061
S8	0.069	0.068	0.069
S9	0.093	0.093	0.094

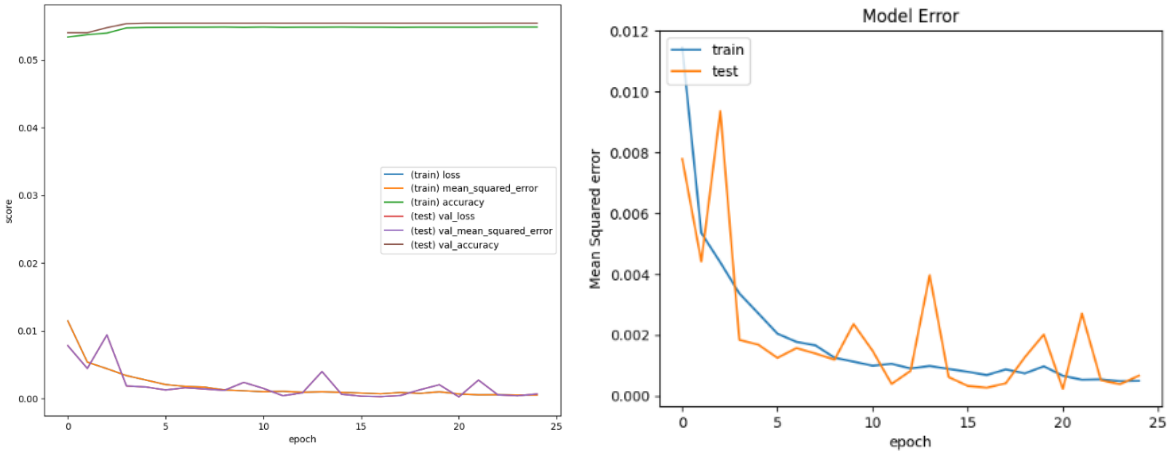
Fuente: Elaboración propia

Anexo 17: Métricas de entrenamiento de Sujeto 3 Prueba 1 Modelo LSTM+FCN para velocidad



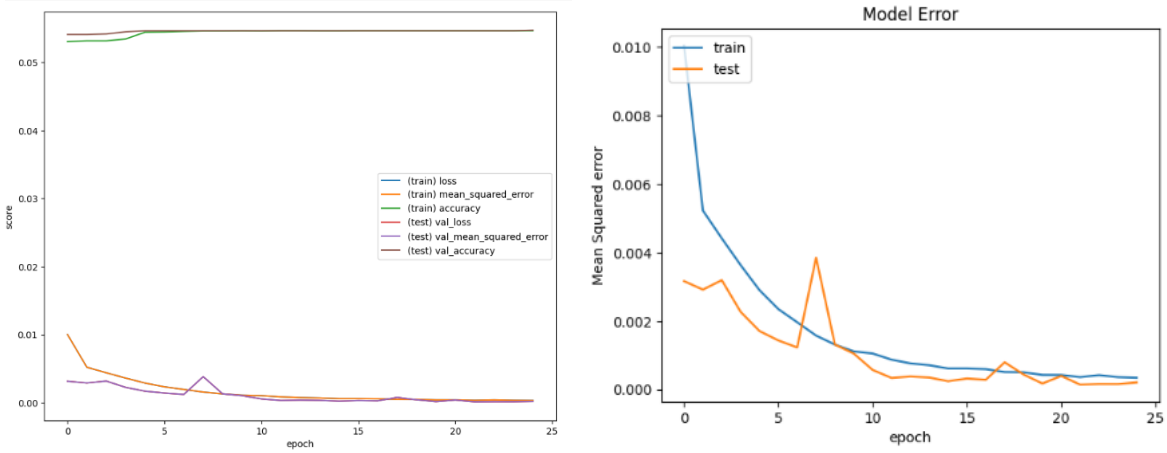
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 18: Métricas de entrenamiento Sujeto 9 Prueba 1 Modelo LSTM+ FCN para velocidad



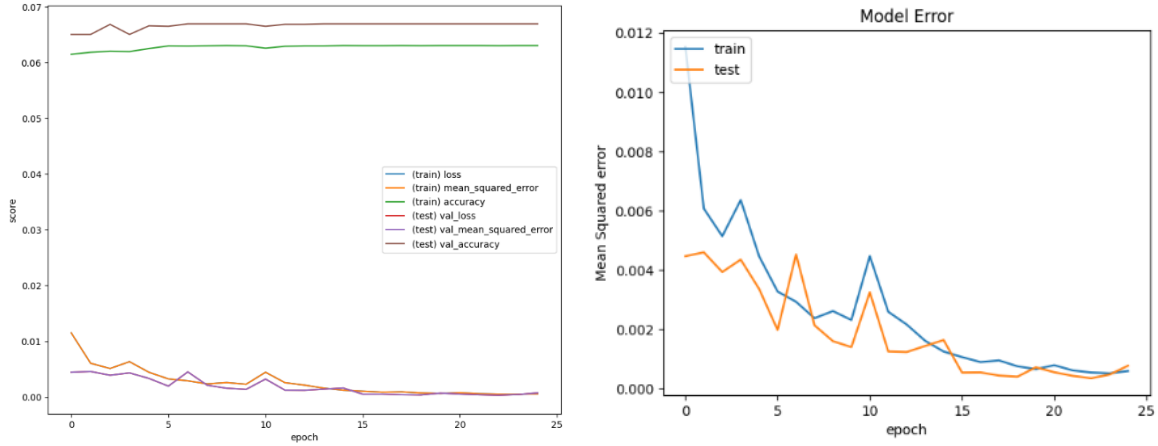
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 19: Métricas de entrenamiento de sujeto 5 Prueba 1 Modelo LSTM+FCN para velocidad



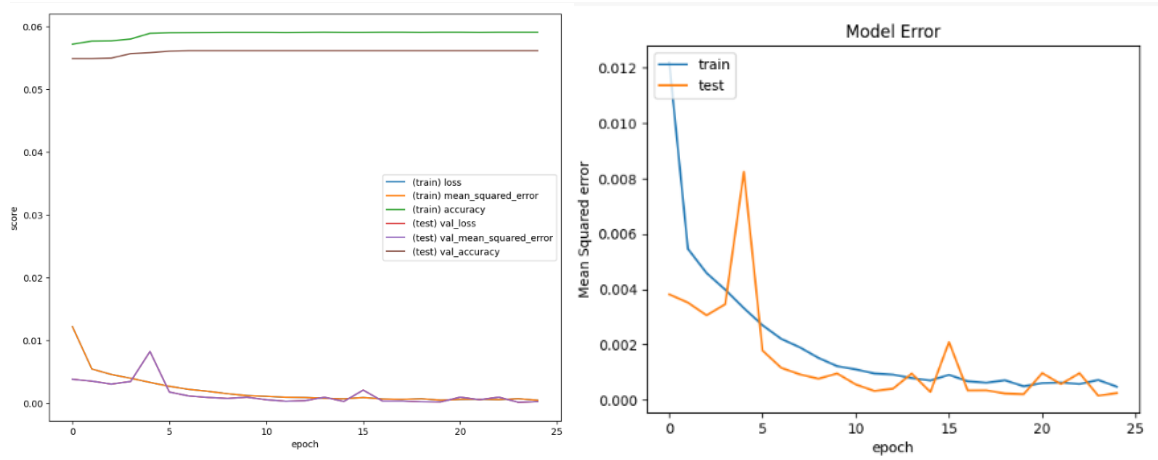
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 20: Métricas de entrenamiento del sujeto 7 Prueba 1 Modelo LSTM+FCN para velocidad



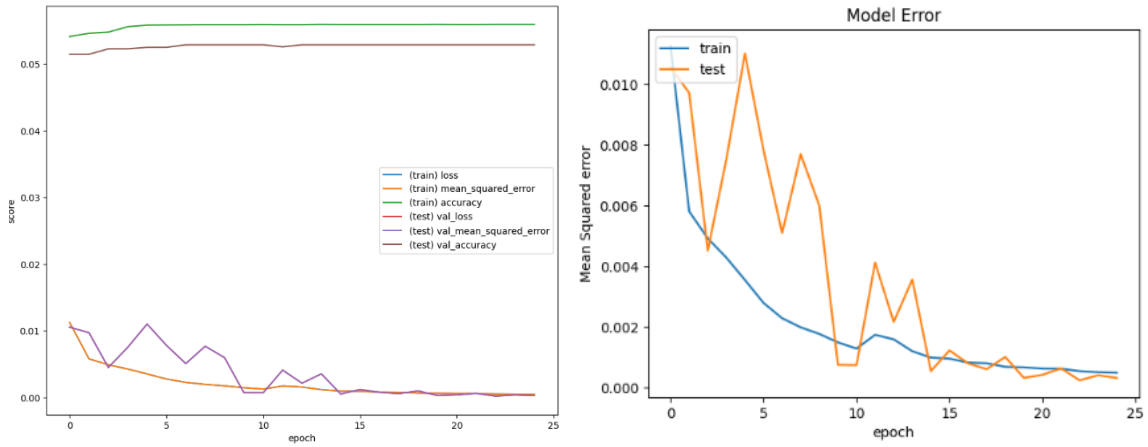
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 21: Métricas de entrenamiento del sujeto 2 Prueba 1 Modelo LSTM+FCN para velocidad



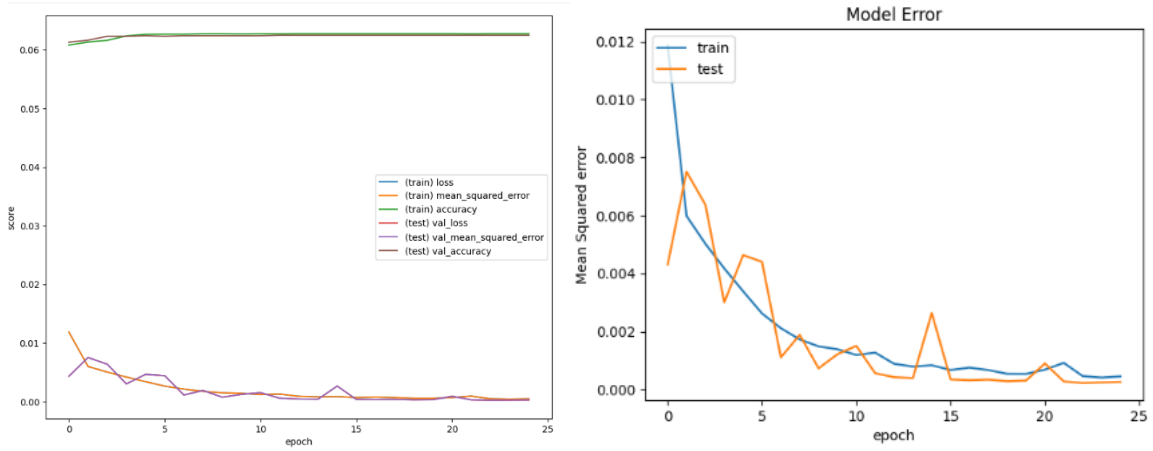
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 22: Métricas de entrenamiento del sujeto 4 Prueba 1 Modelo LSTM+FCN para velocidad



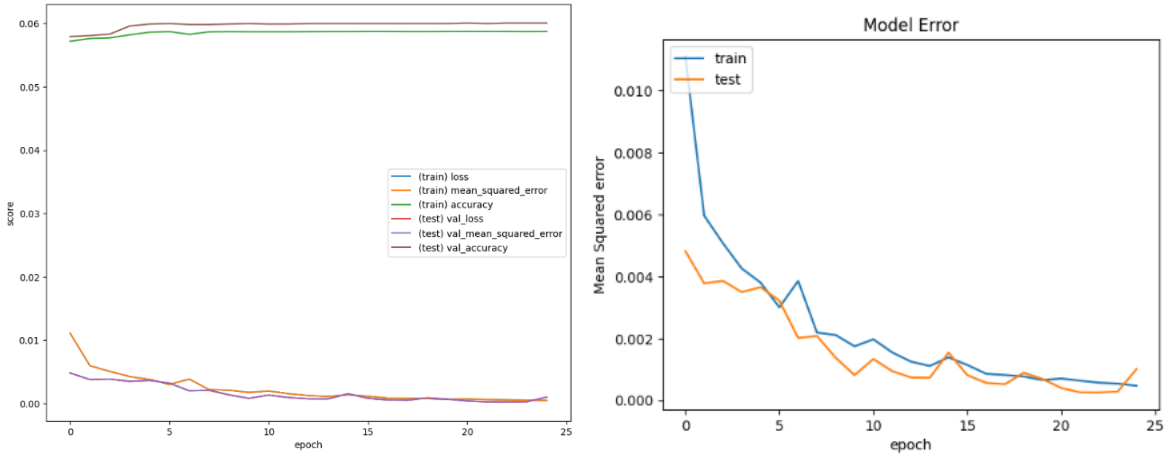
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 23: Métricas de entrenamiento del sujeto 6 Prueba 1 Modelo LSTM+FCN para velocidad



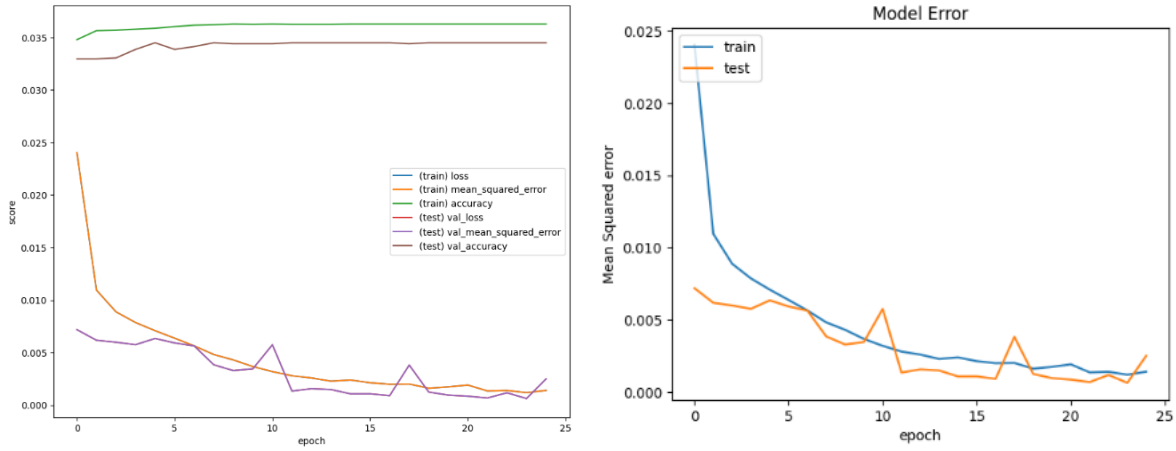
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 24: Métricas de entrenamiento del sujeto 8 Prueba 1 Modelo LSTM+FCN para velocidad



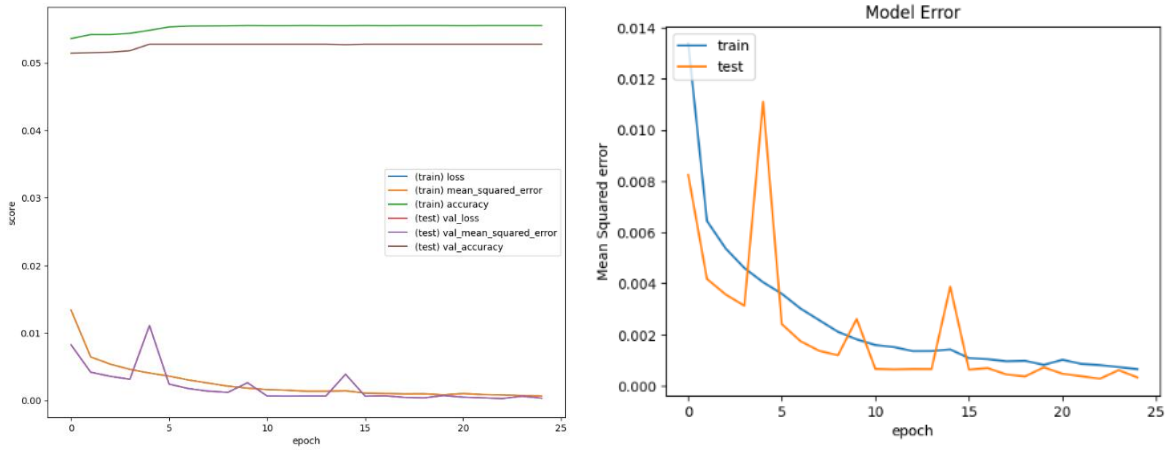
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 25: Métricas de entrenamiento del sujeto 3 Prueba 2 Modelo LSTM+FCN para velocidad



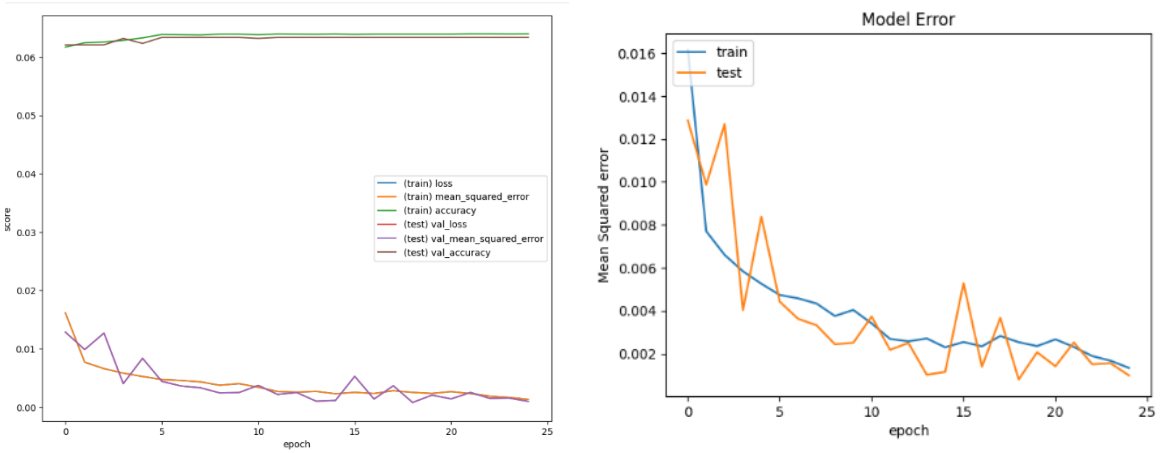
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 26: Métricas de entrenamiento del Sujeto 9 Prueba 2 Modelo LSTM+FCN para velocidad



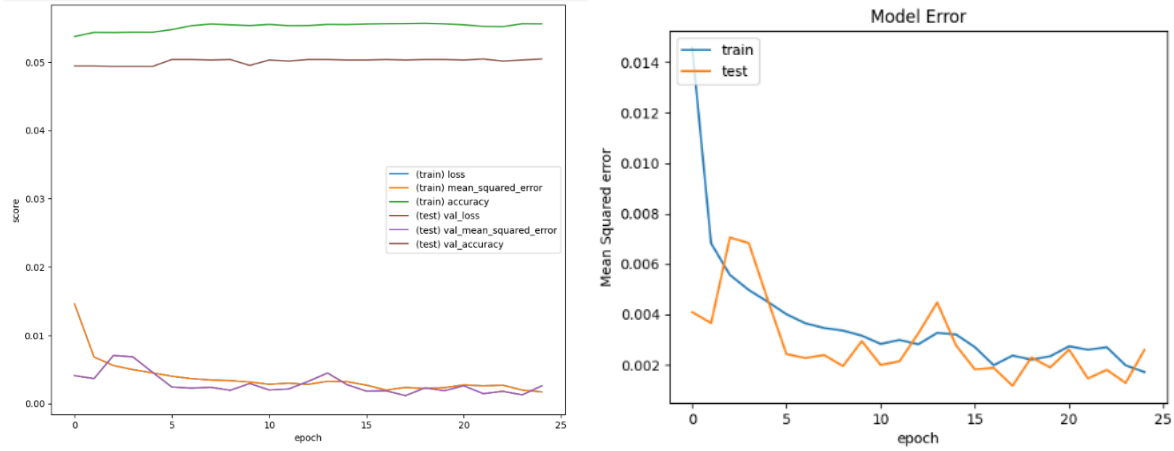
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 27: Métricas de entrenamiento del Sujeto 7 Prueba 2 Modelo LSTM+FCN para velocidad



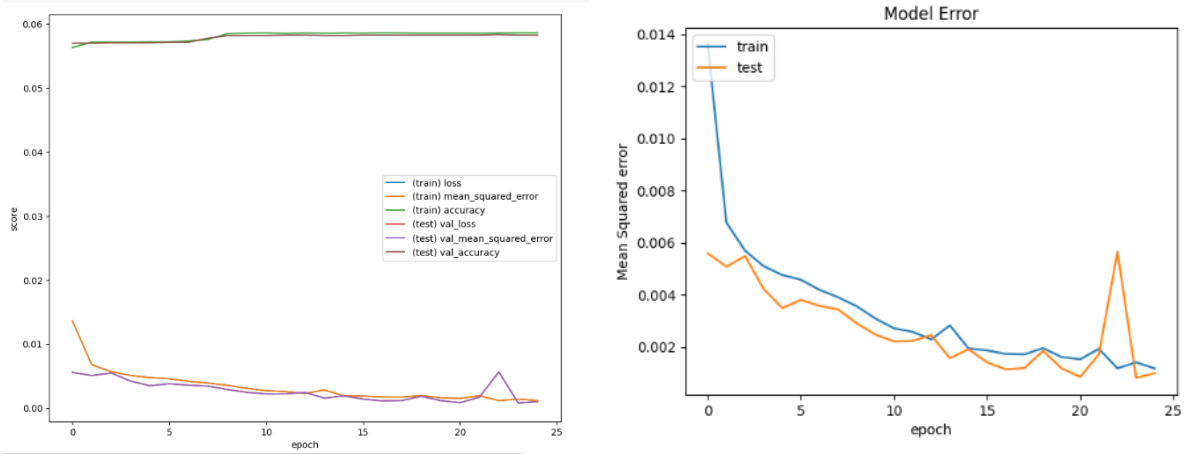
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 28: Métricas de entrenamiento del Sujeto 5 Prueba 2 Modelo LSTM+FCN para velocidad



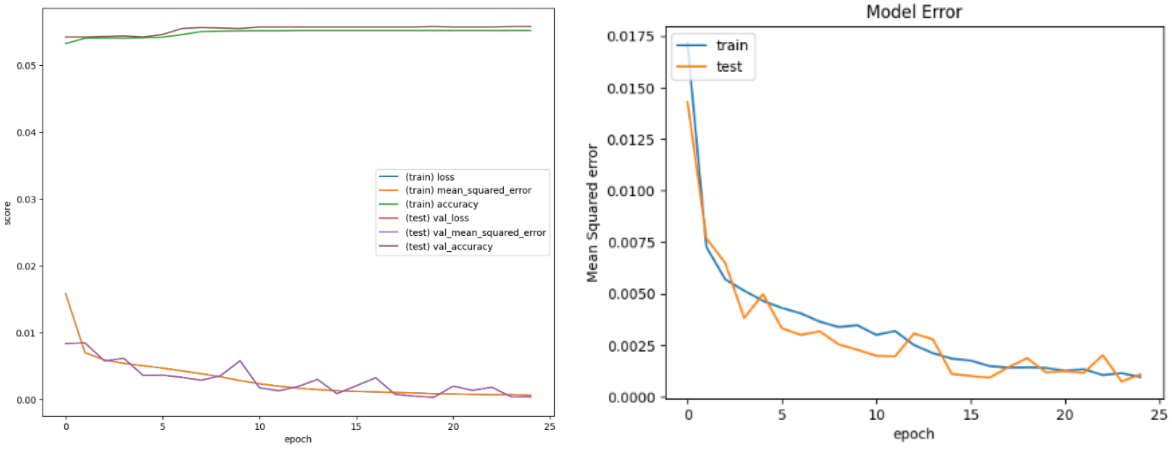
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 29:Métricas de entrenamiento del Sujeto 2 Prueba 2 Modelo LSTM+FCN para velocidad



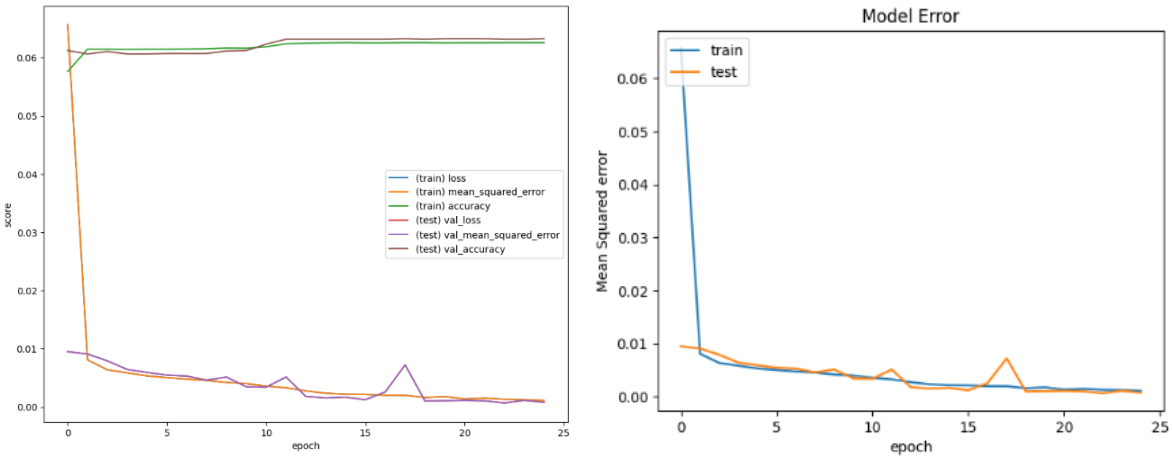
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 30: Métricas de entrenamiento del Sujeto 4 Prueba 2 Modelo LSTM+FCN para velocidad



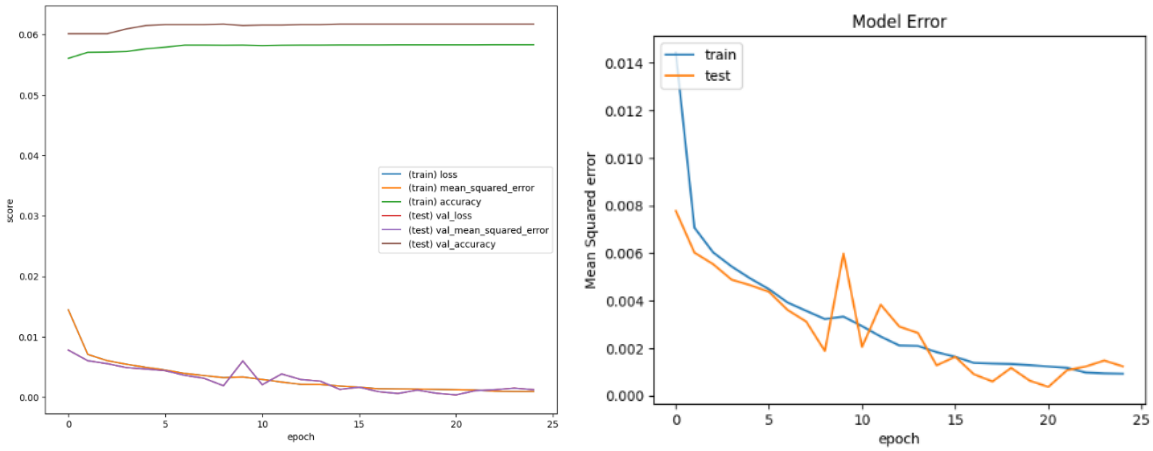
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 31: Métricas de entrenamiento del Sujeto 6 Prueba 2 Modelo LSTM+FCN para velocidad



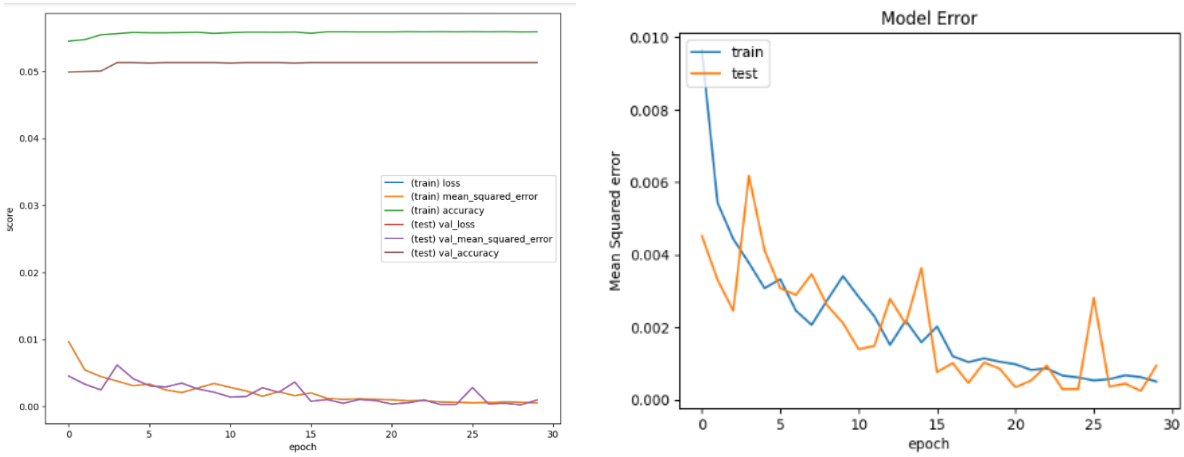
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 32: Métricas de entrenamiento del Sujeto 8 Prueba 2 Modelo LSTM+FCN para velocidad



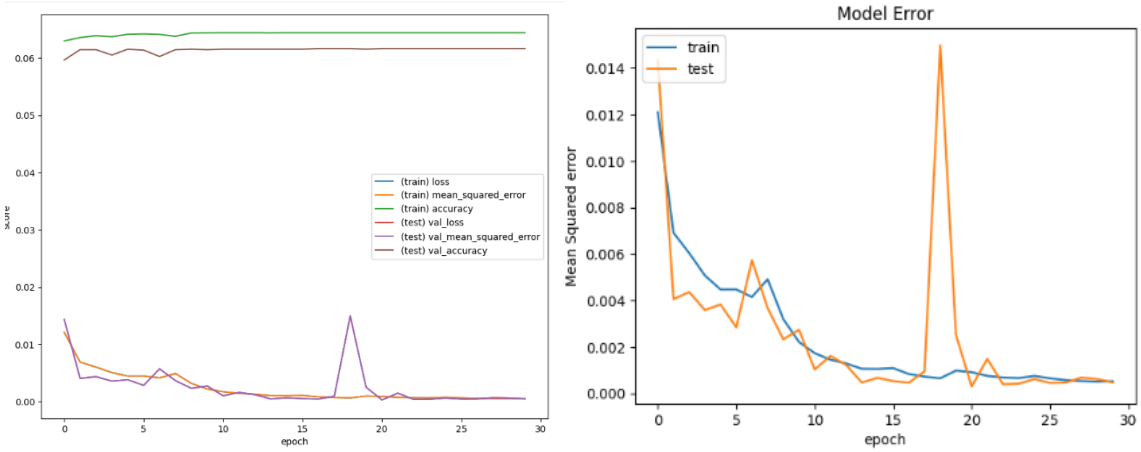
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 33: Métricas de entrenamiento del Sujeto 9 Prueba 3 Modelo LSTM+FCN para velocidad



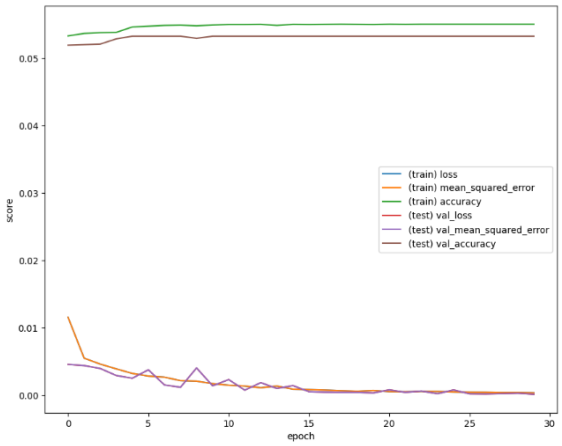
Fuente: Elaboración propia obtenida por código de Google Colab

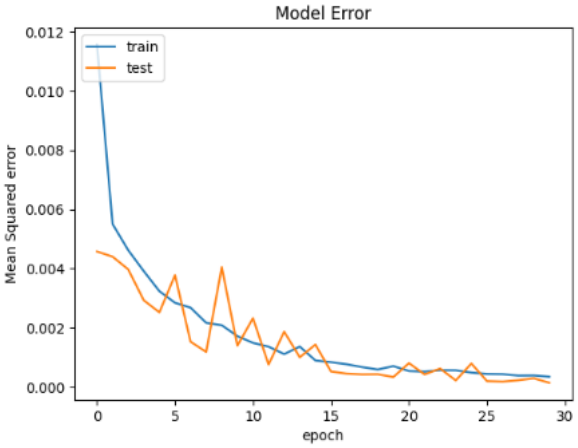
Anexo 34: Métricas de entrenamiento del Sujeto 7 Prueba 3 Modelo LSTM+FCN para velocidad



Fuente: Elaboración propia obtenida por código de Google Colab

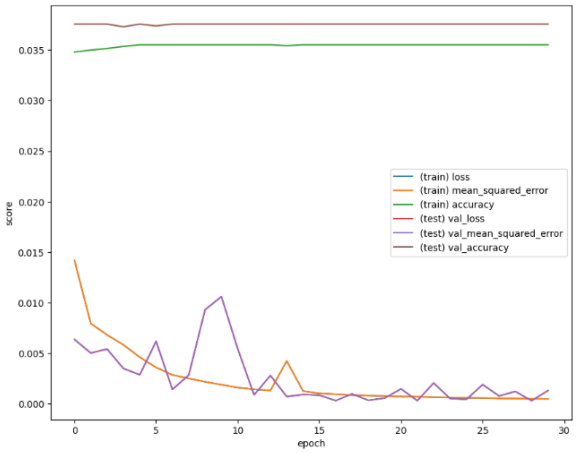
Anexo 35: Métricas de entrenamiento del Sujeto 5 Prueba 3 Modelo LSTM+FCN para velocidad

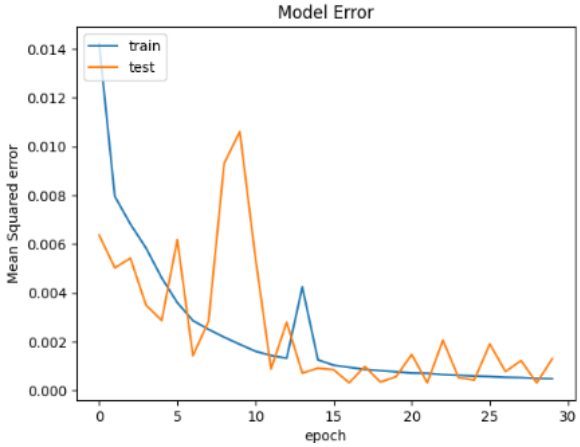




Fuente: Elaboración propia obtenida por código de Google Colab

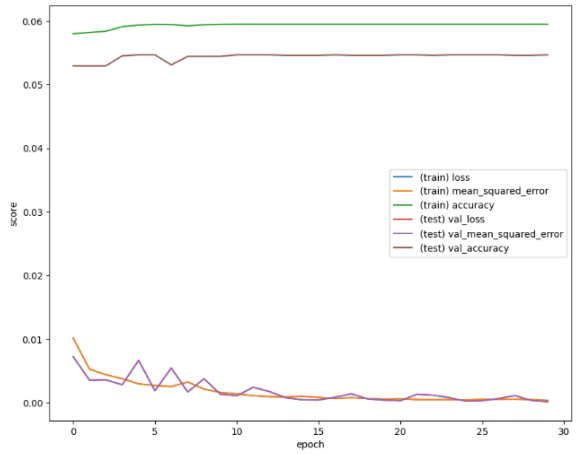
Anexo 36: Métricas de entrenamiento del Sujeto 3 Prueba 3 Modelo LSTM+FCN para velocidad

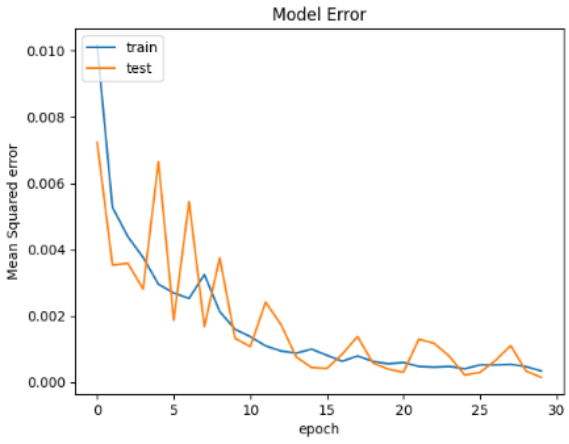




Fuente: Elaboración propia obtenida por código de Google Colab

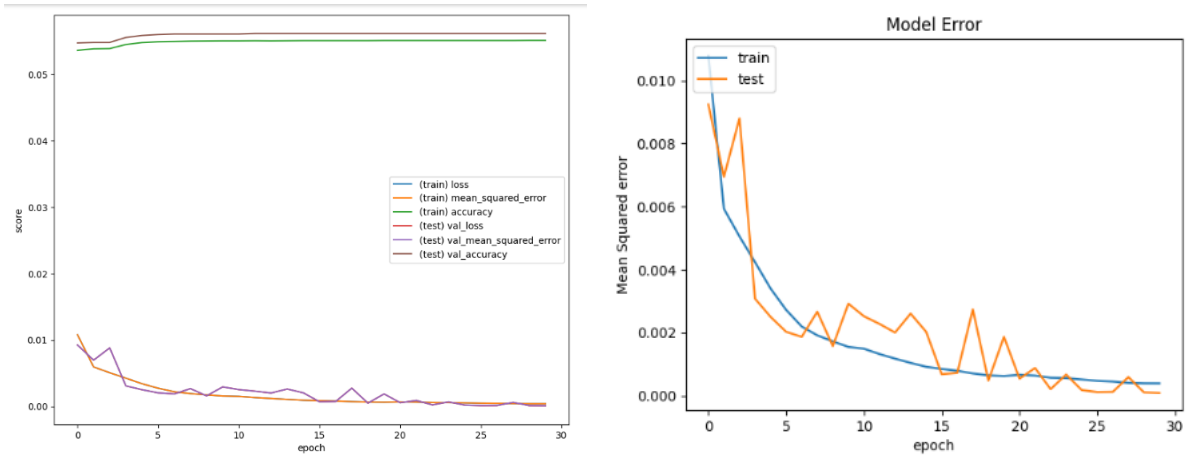
Anexo 37: Métricas de entrenamiento del Sujeto 2 Prueba 3 Modelo LSTM+FCN para velocidad





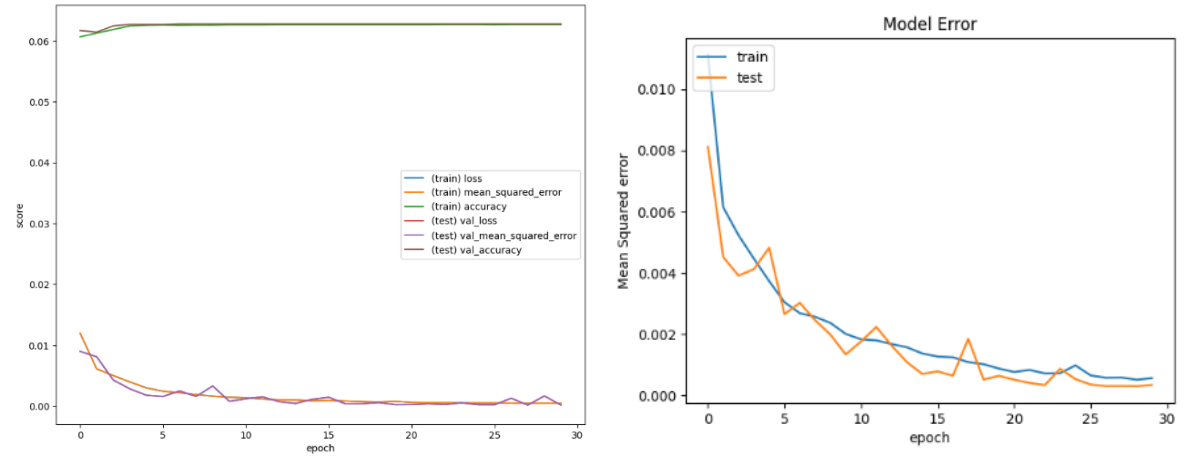
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 38: Métricas de entrenamiento del Sujeto 4 Prueba 3 Modelo LSTM+FCN para velocidad



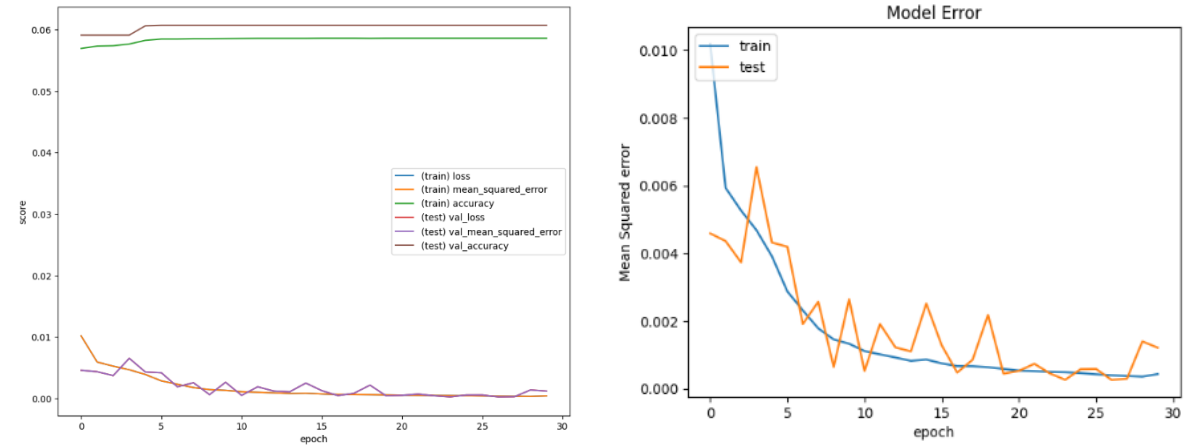
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 39: Métricas de entrenamiento del Sujeto 6 Prueba 3 Modelo LSTM+FCN para velocidad



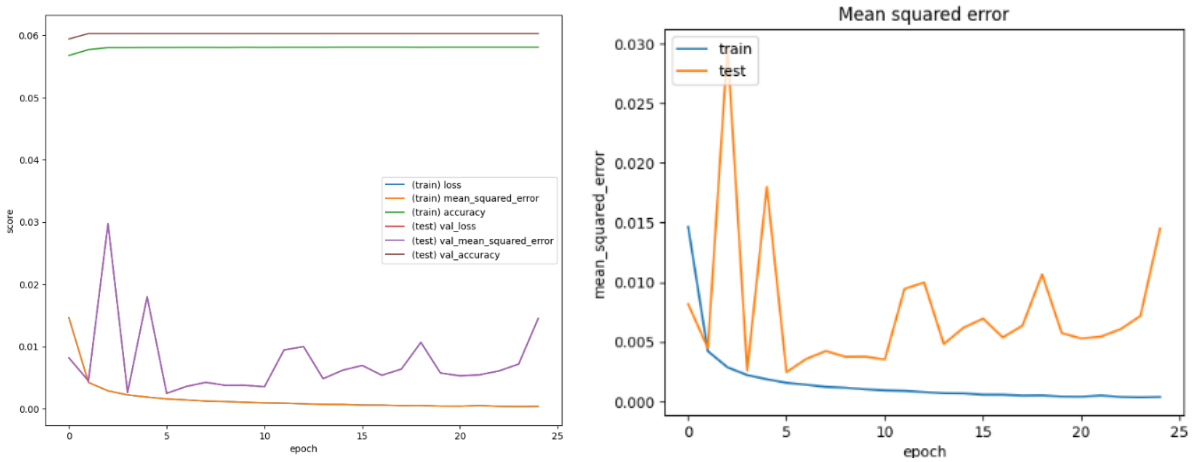
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 40: Métricas de entrenamiento del Sujeto 8 Prueba 3 Modelo LSTM+FCN para velocidad



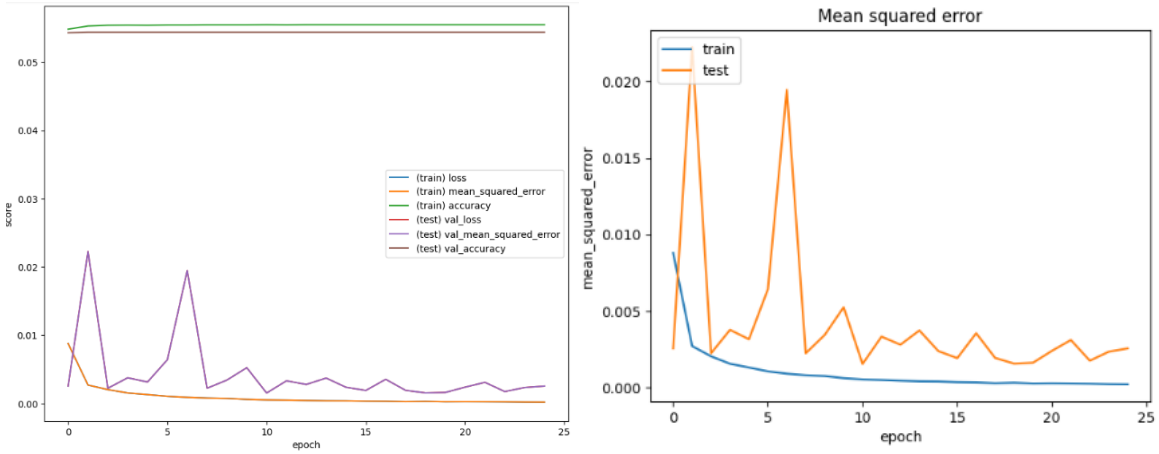
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 41: Métricas de entrenamiento del Sujeto 2 Prueba 1 Modelo LSTM+CNN para velocidad



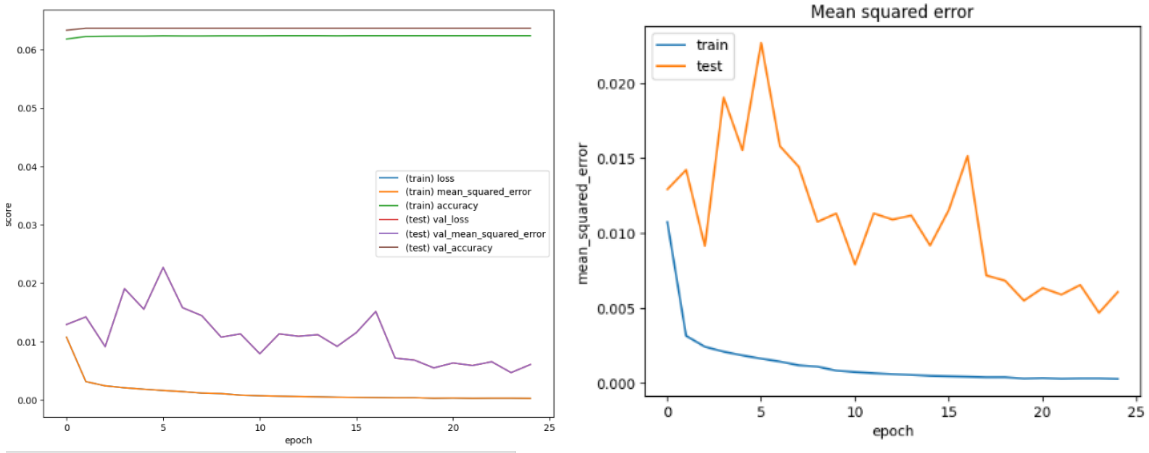
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 42: Métricas de entrenamiento del Sujeto 4 Prueba 1 Modelo LSTM+CNN para velocidad



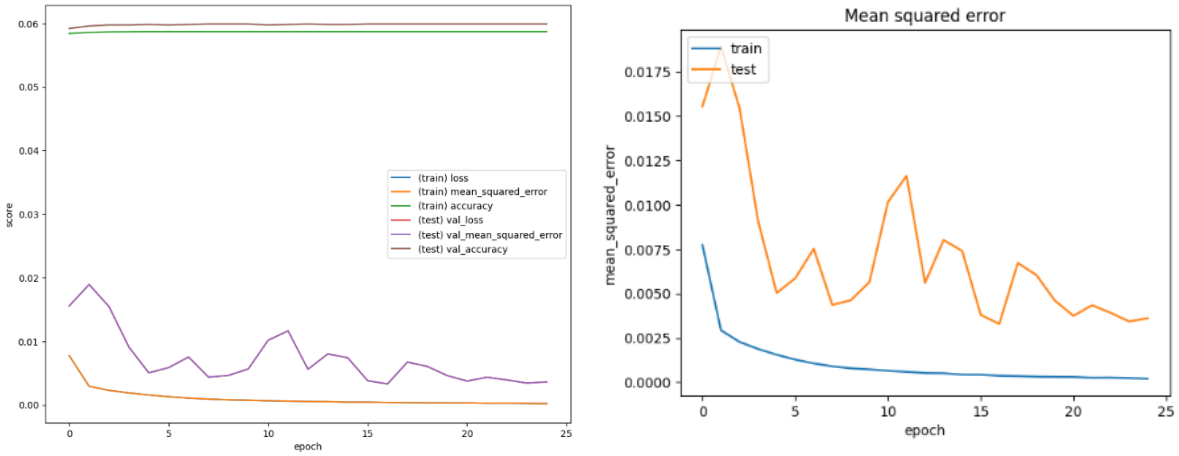
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 43: Métricas de entrenamiento del Sujeto 6 Prueba 1 Modelo LSTM+CNN para velocidad



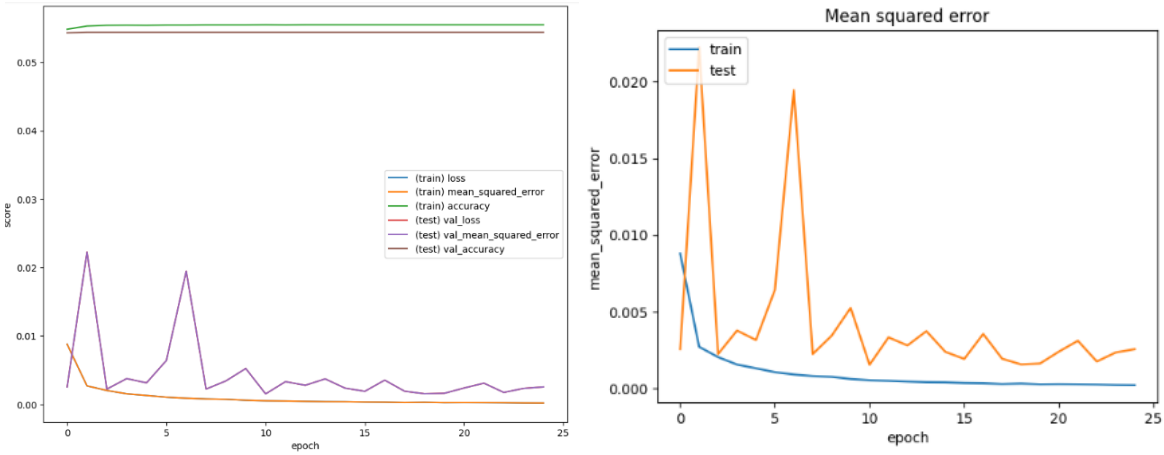
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 44: Métricas de entrenamiento del Sujeto 8 Prueba 1 Modelo LSTM+CNN para velocidad



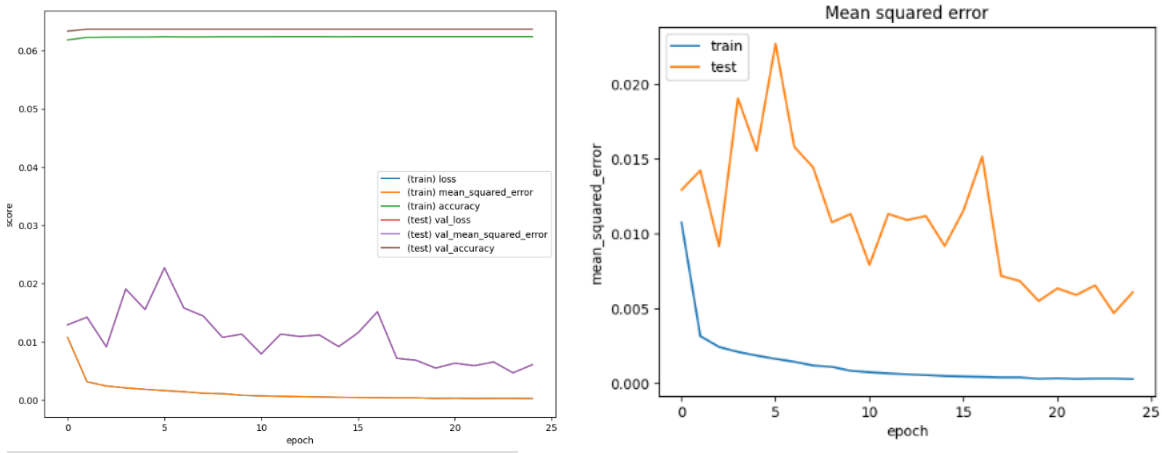
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 45: Métricas de entrenamiento del Sujeto 4 Prueba 1 Modelo LSTM+CNN para velocidad



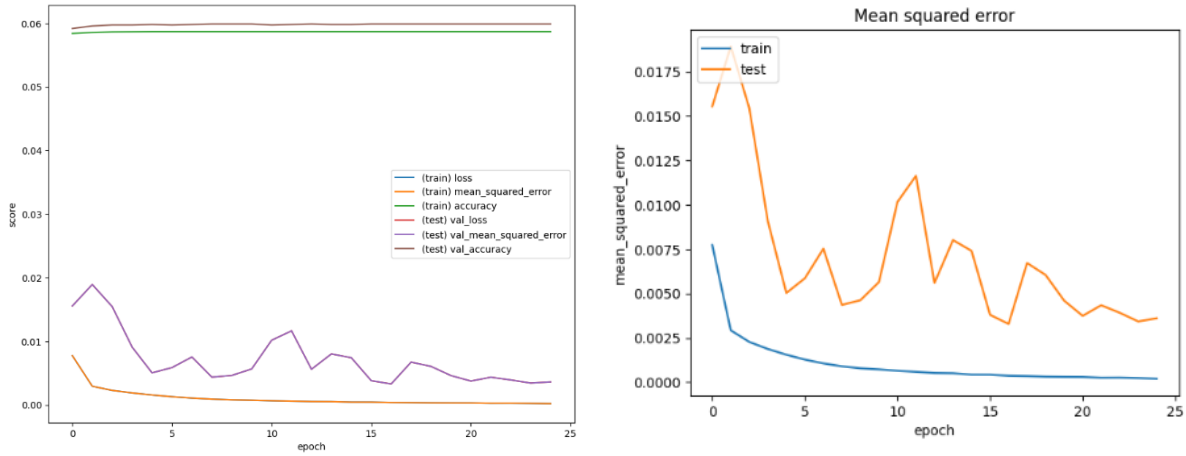
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 46: Métricas de entrenamiento del Sujeto 6 Prueba 1 Modelo LSTM+CNN para velocidad



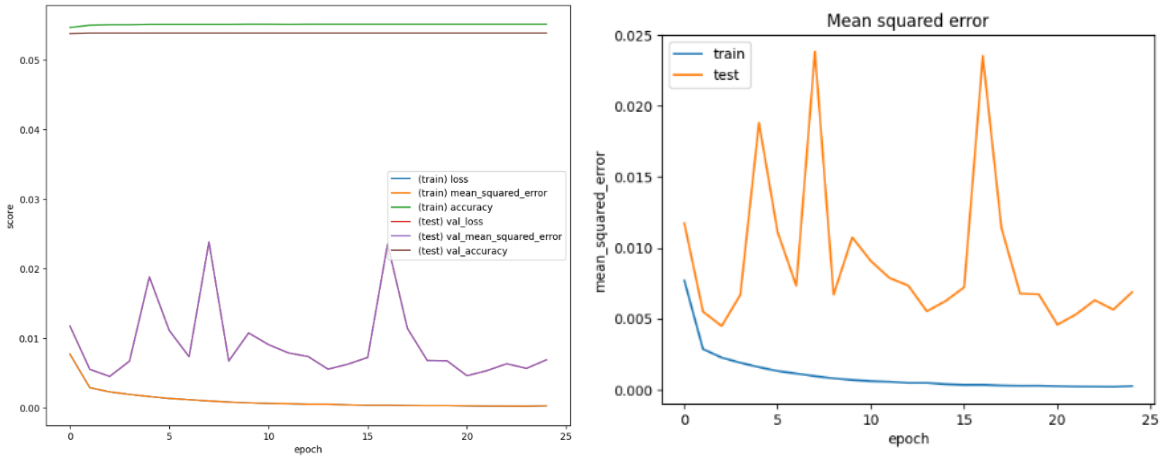
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 47: Métricas de entrenamiento del Sujeto 8 Prueba 1 Modelo LSTM+CNN para velocidad



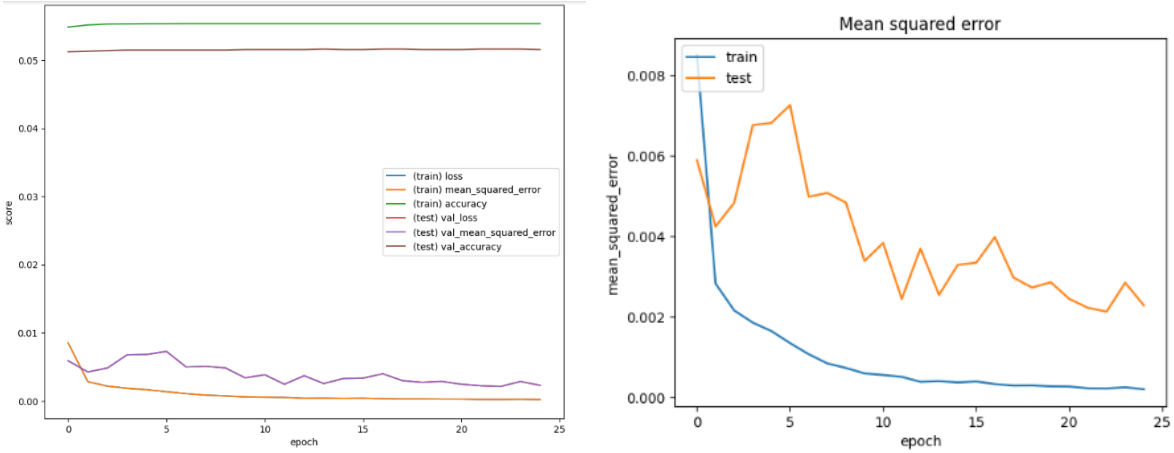
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 48: Métricas de entrenamiento del Sujeto 9 Prueba 1 Modelo LSTM+CNN para velocidad



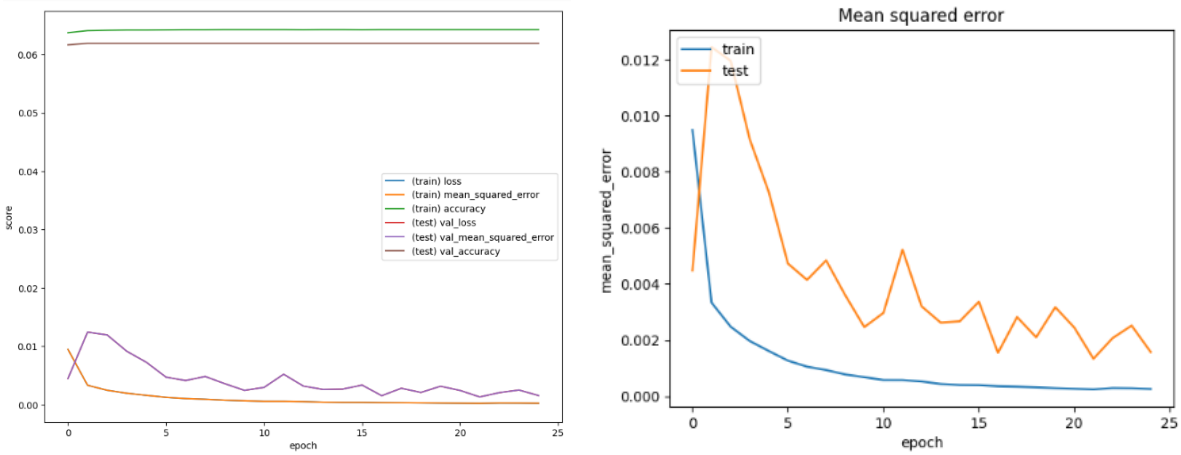
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 49: Métricas de entrenamiento del Sujeto 5 Prueba 1 Modelo LSTM+CNN para velocidad



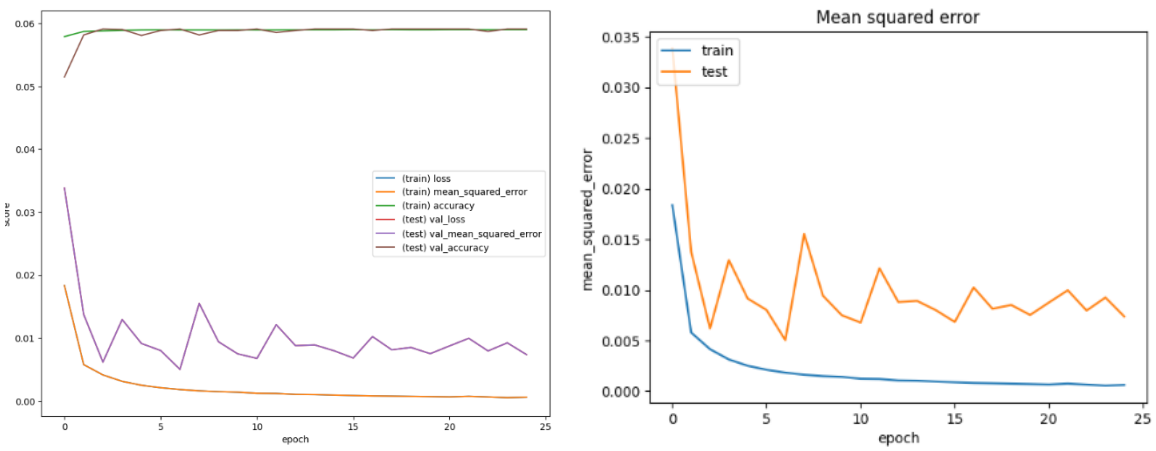
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 50: Métricas de entrenamiento del Sujeto 7 Prueba 1 Modelo LSTM+CNN para velocidad



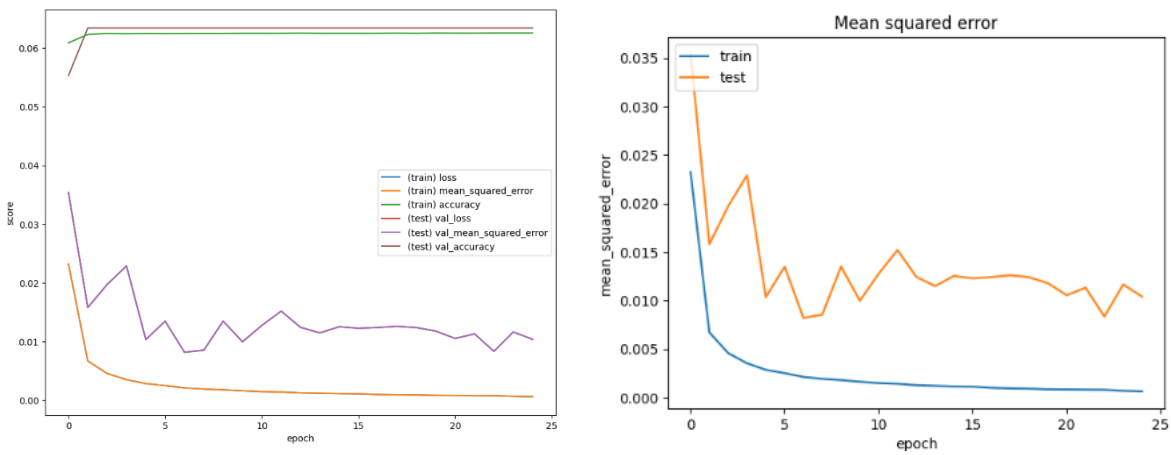
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 51: Métricas de entrenamiento del Sujeto 8 Prueba 2 Modelo LSTM+CNN para velocidad



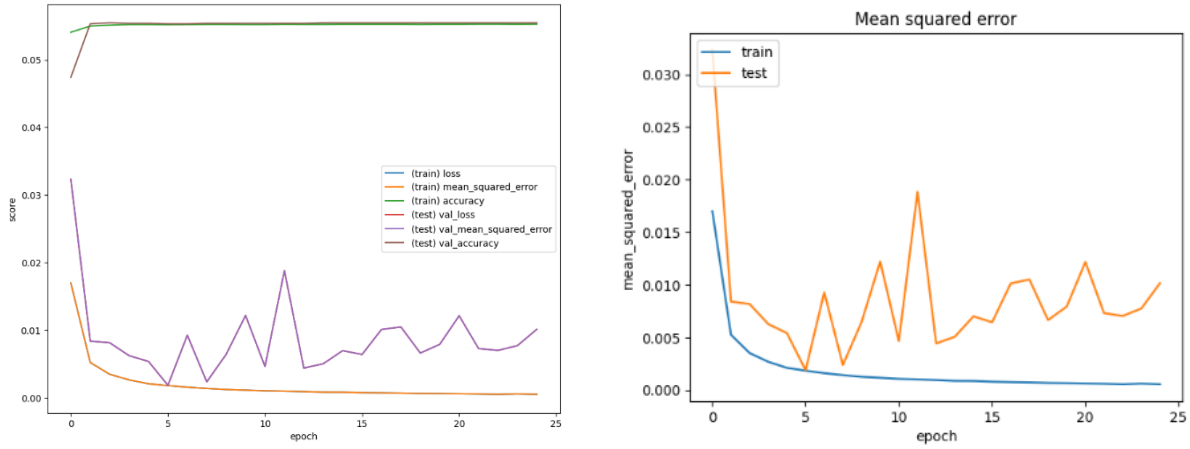
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 52: Métricas de entrenamiento del Sujeto 6 Prueba 2 Modelo LSTM+CNN para velocidad



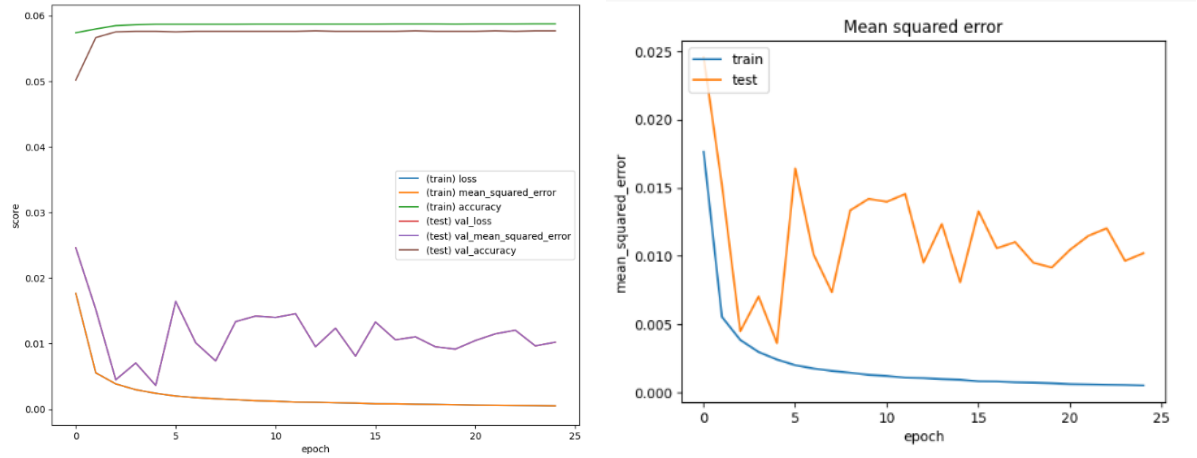
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 53: Métricas de entrenamiento del Sujeto 4 Prueba 2 Modelo LSTM+CNN para velocidad

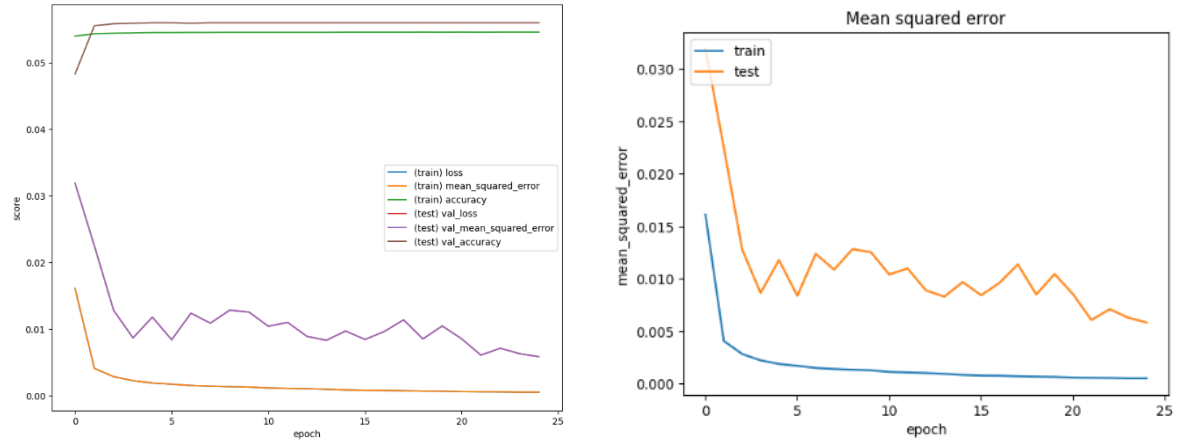


Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 54: Métricas de entrenamiento del Sujeto 2 Prueba 2 Modelo LSTM+CNN para velocidad

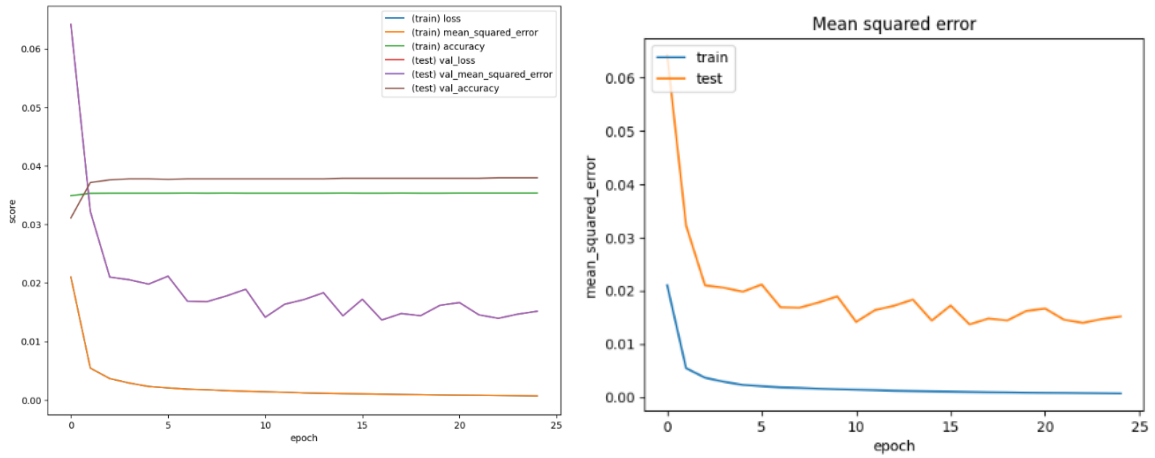


Anexo 55: Métricas de entrenamiento del Sujeto 9 Prueba 2 Modelo LSTM+CNN para velocidad



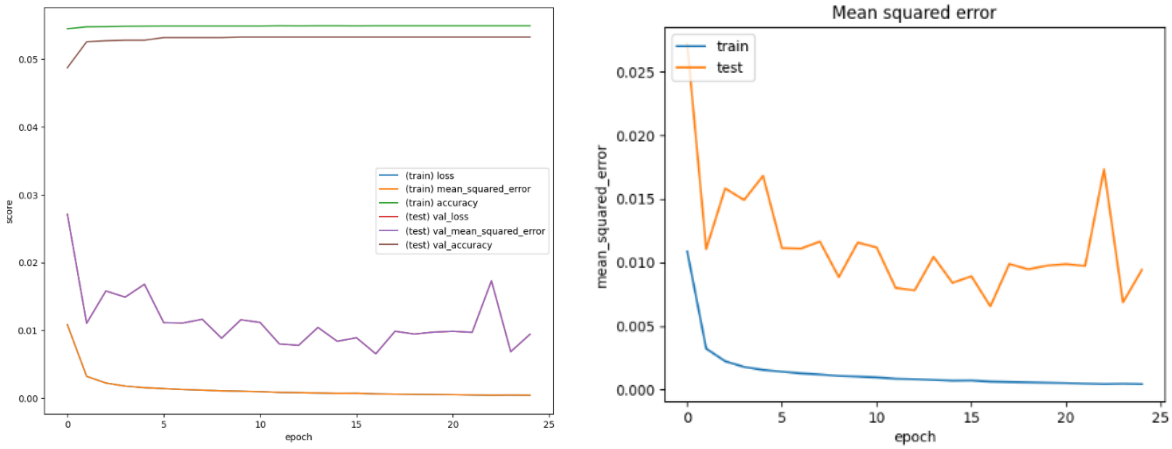
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 56: Métricas de entrenamiento del Sujeto 3 Prueba 2 Modelo LSTM+CNN para velocidad



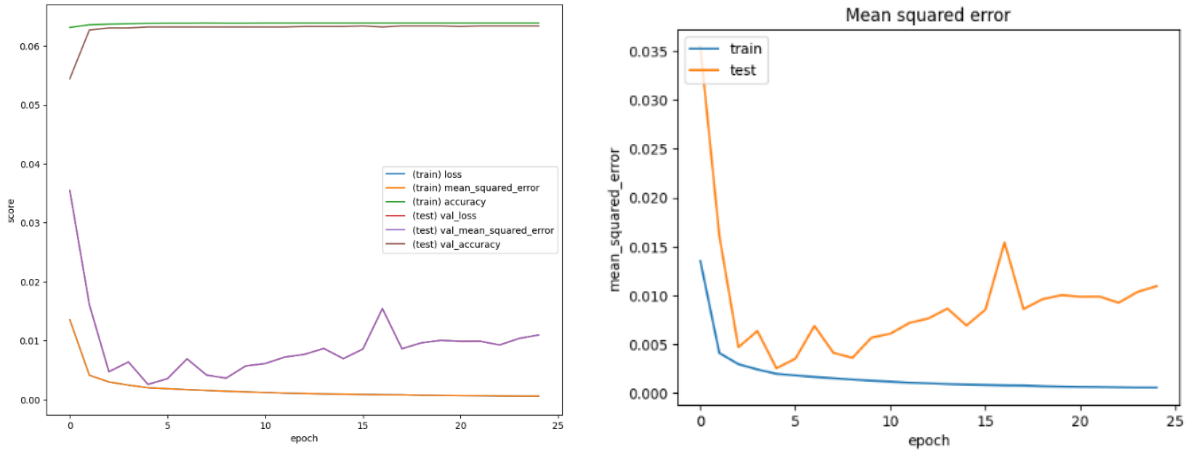
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 57: Métricas de entrenamiento del Sujeto 5 Prueba 2 Modelo LSTM+CNN para velocidad



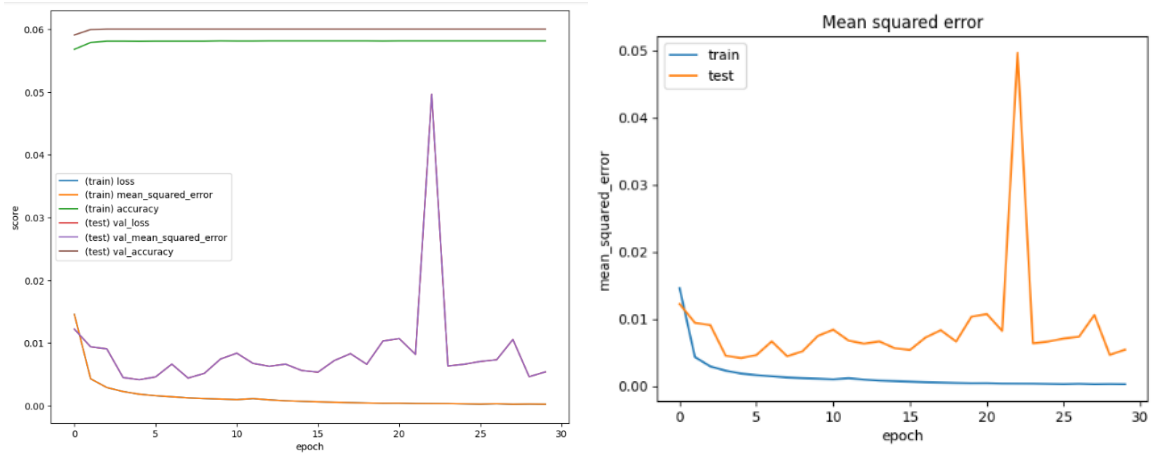
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 58: Métricas de entrenamiento del Sujeto 7 Prueba 2 Modelo LSTM+CNN para velocidad



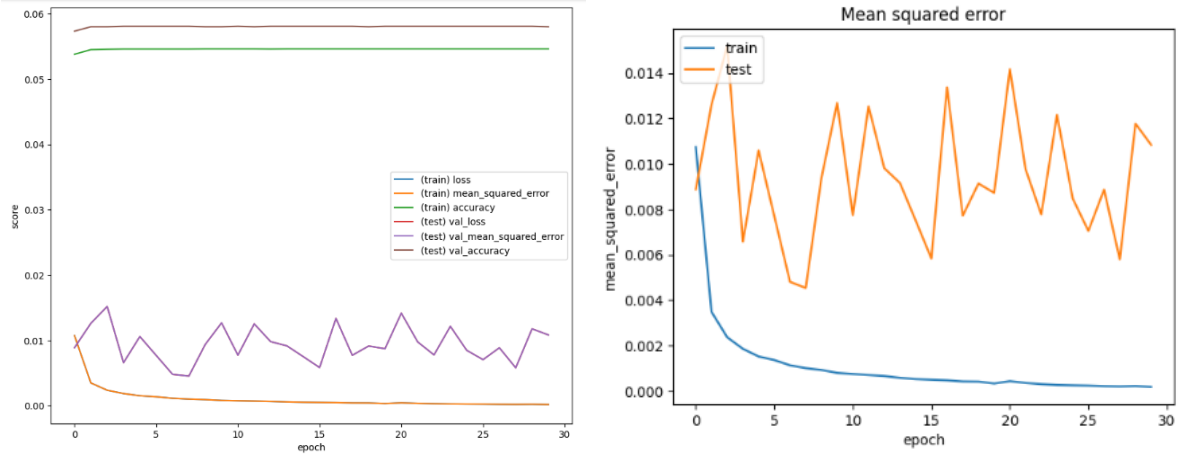
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 59: Métricas de entrenamiento del Sujeto 2 Prueba 3 Modelo LSTM+CNN para velocidad



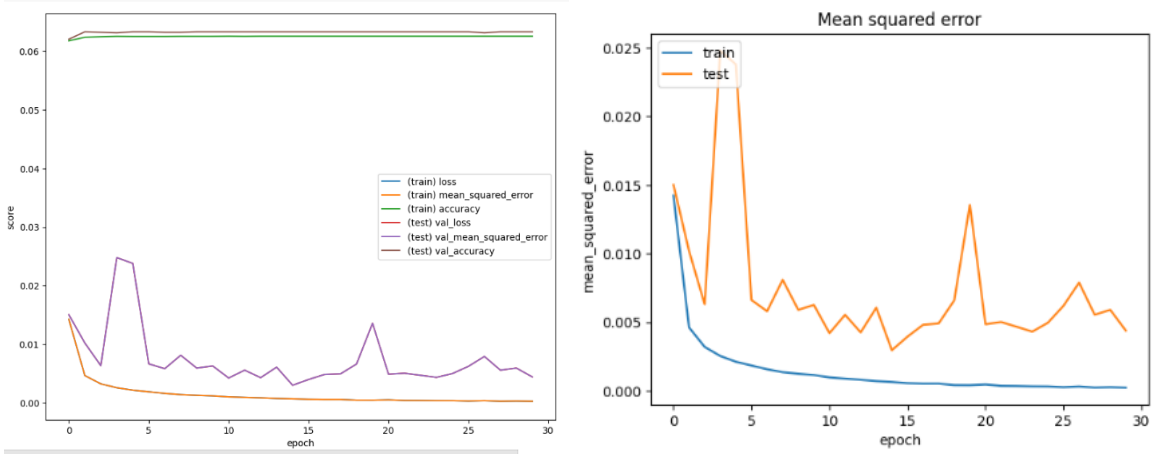
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 60: Métricas de entrenamiento del Sujeto 4 Prueba 3 Modelo LSTM+CNN para velocidad



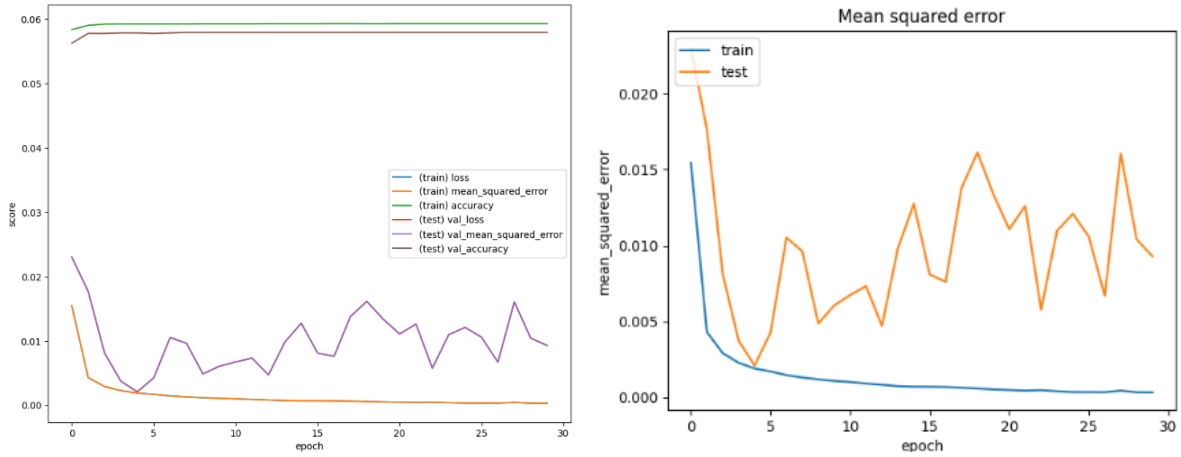
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 61: Métricas de entrenamiento del Sujeto 6 Prueba 3 Modelo LSTM+CNN para velocidad



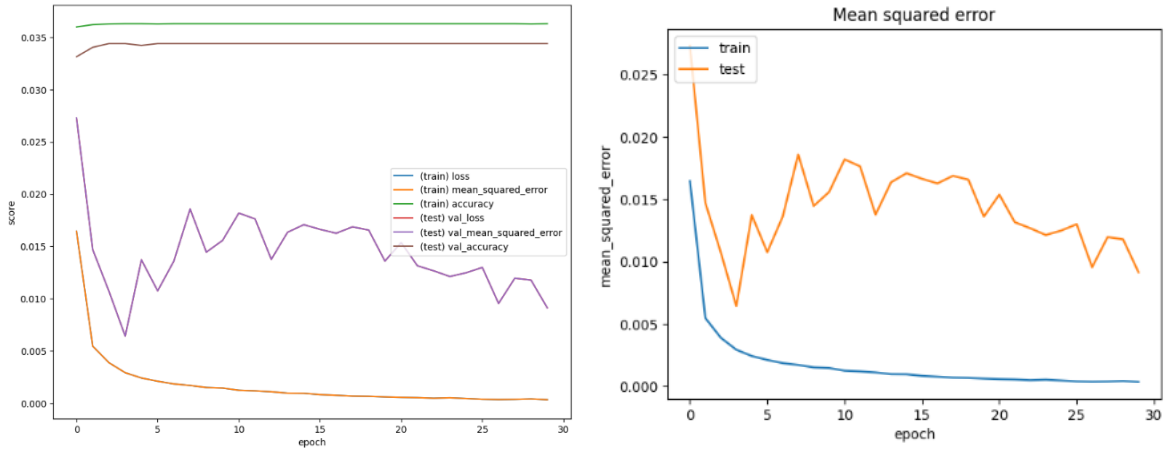
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 62: Métricas de entrenamiento del Sujeto 8 Prueba 3 Modelo LSTM+CNN para velocidad



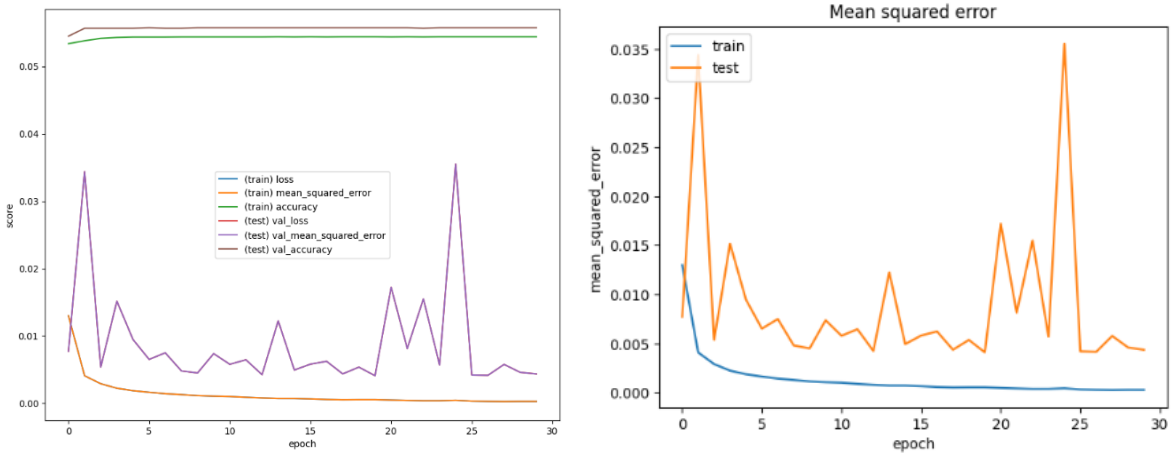
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 63: Métricas de entrenamiento del Sujeto 3 Prueba 3 Modelo LSTM+CNN para velocidad



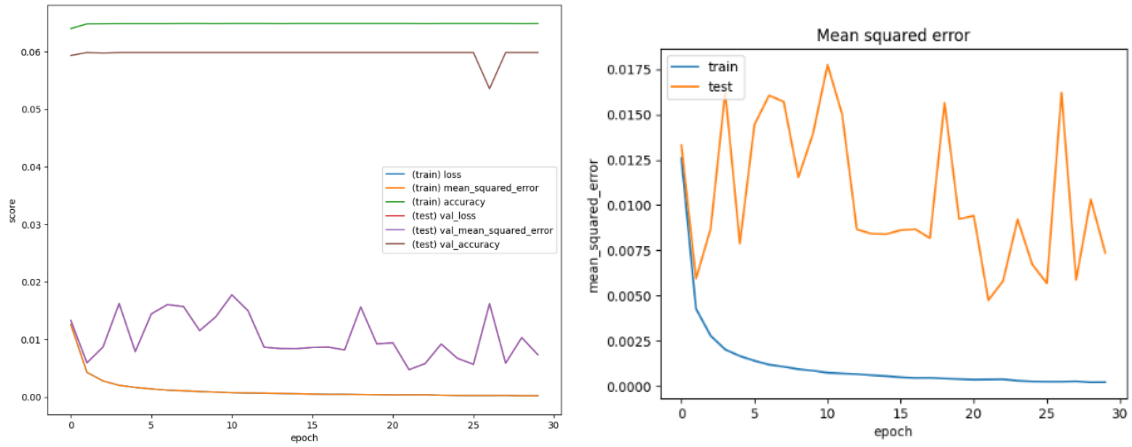
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 64: Métricas de entrenamiento del Sujeto 5 Prueba 3 Modelo LSTM+CNN para velocidad



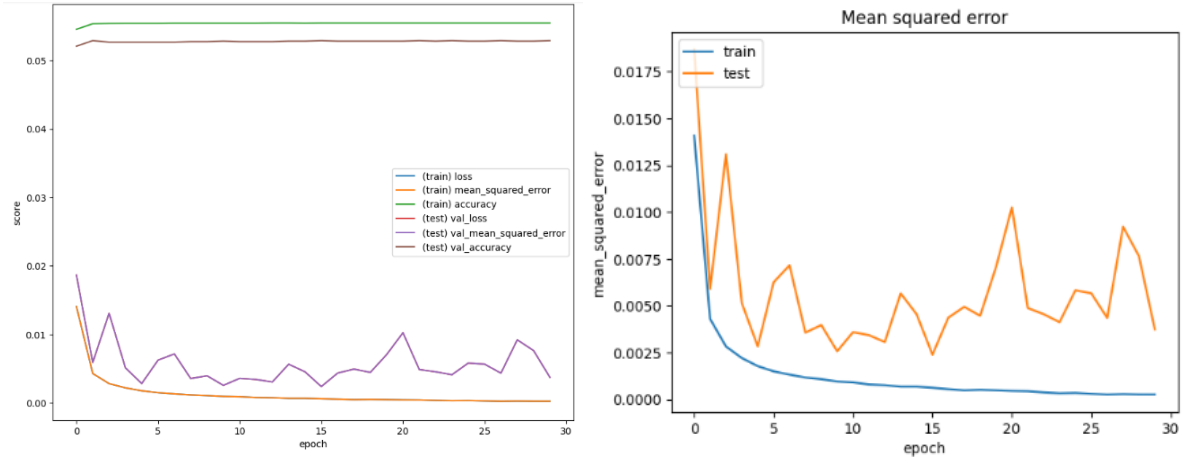
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 65: Métricas de entrenamiento del Sujeto 7 Prueba 3 Modelo LSTM+CNN para velocidad



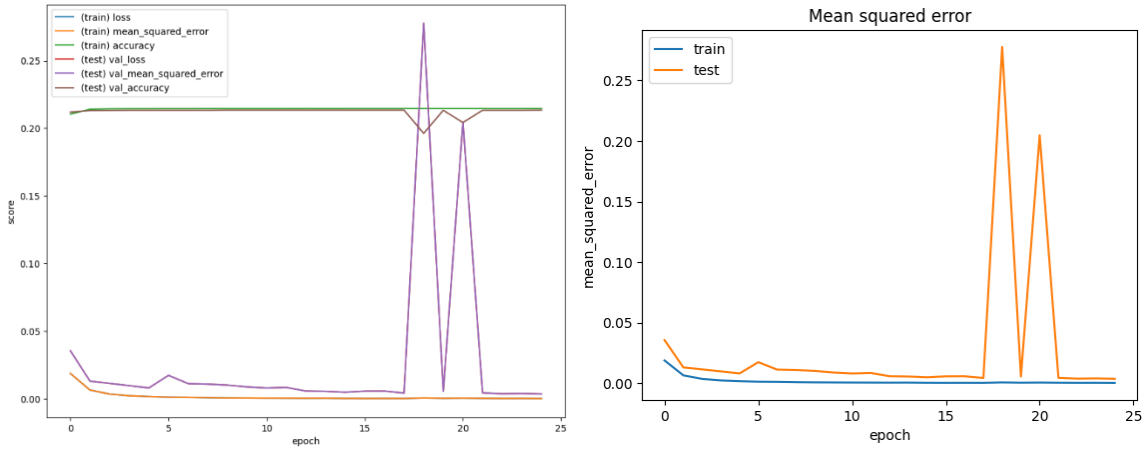
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 66: Métricas de entrenamiento del Sujeto 9 Prueba 3 Modelo LSTM+CNN para velocidad



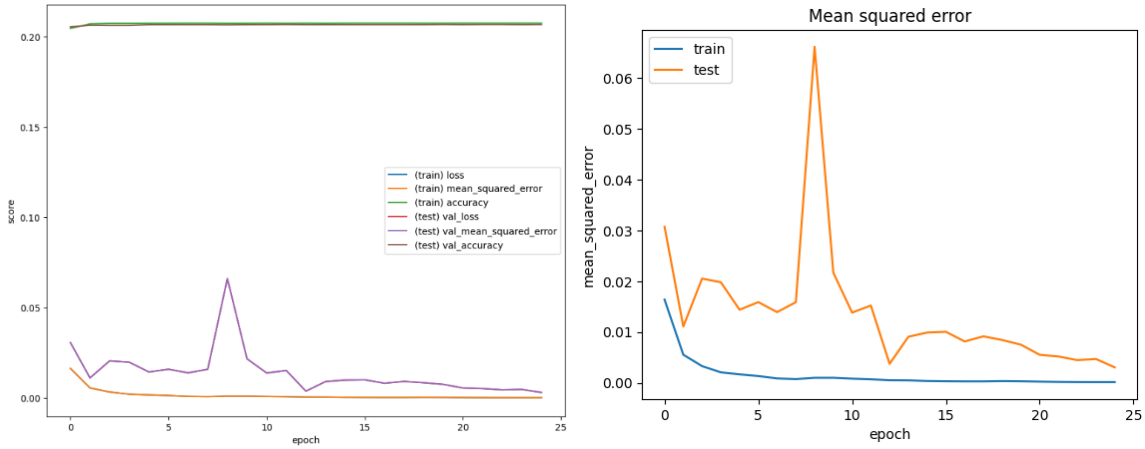
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 67: Métricas de entrenamiento Sujeto 2 Prueba 1 Modelo LSTM+CNN para distancia



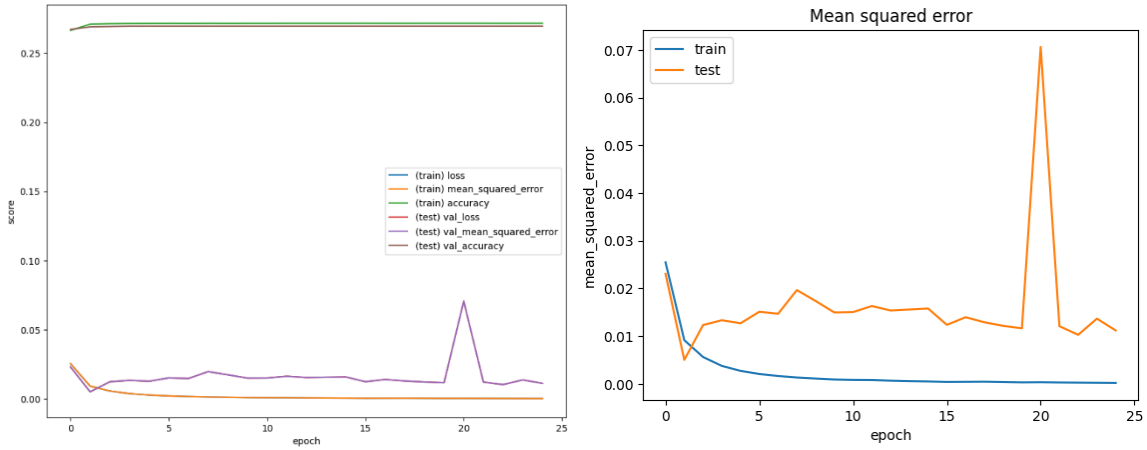
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 68: Métricas de entrenamiento Sujeto 4 Prueba 1 Modelo LSTM+CNN para distancia



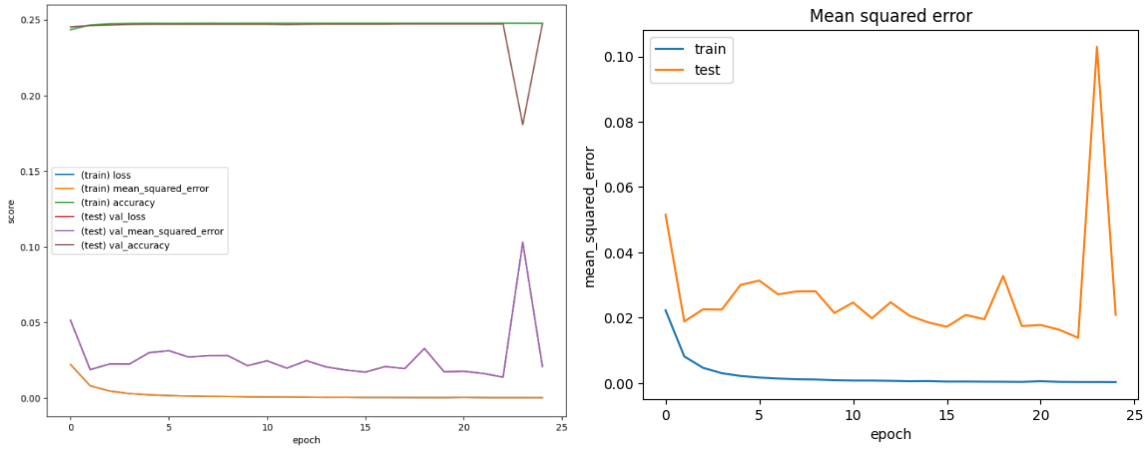
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 69: Métricas de entrenamiento Sujeto 6 Prueba 1 Modelo LSTM+CNN para distancia



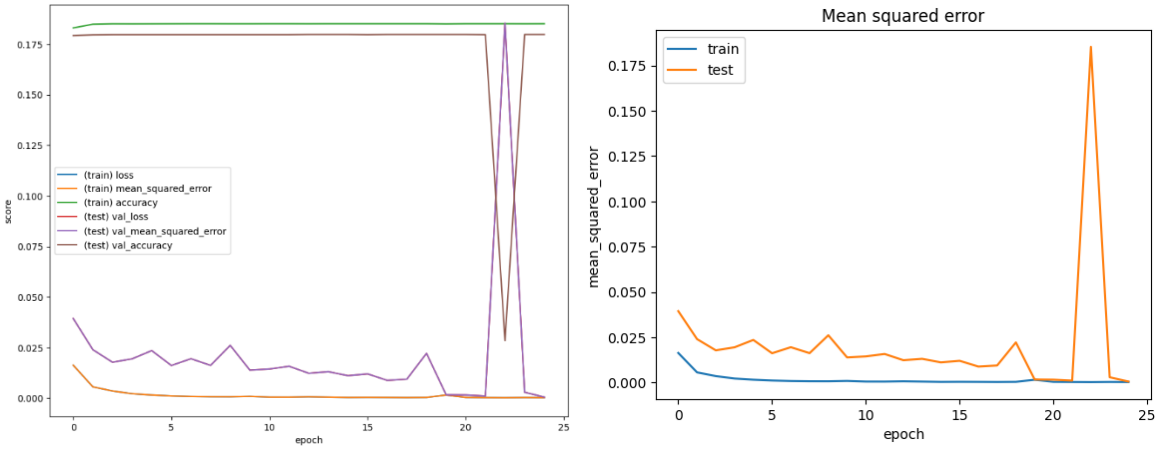
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 70: Métricas de entrenamiento Sujeto 8 Prueba 1 Modelo LSTM+CNN para distancia



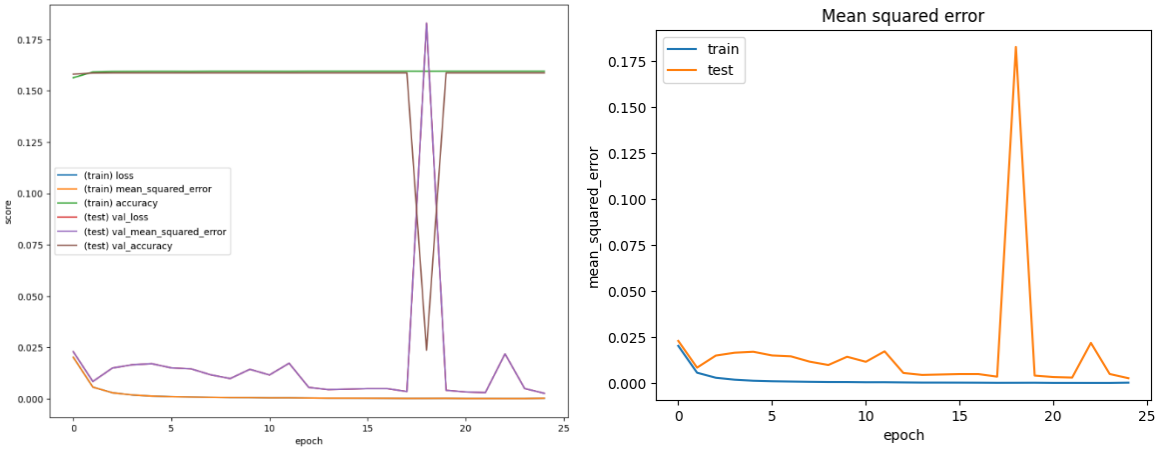
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 71: Métricas de entrenamiento Sujeto 3 Prueba 1 Modelo LSTM+CNN para distancia



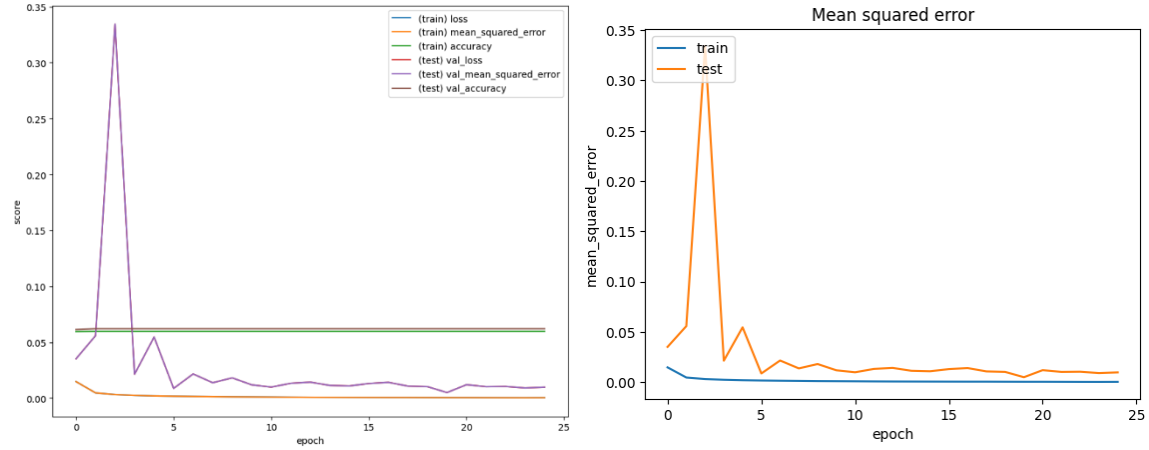
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 72: Métricas de entrenamiento Sujeto 5 Prueba 1 Modelo LSTM+CNN para distancia



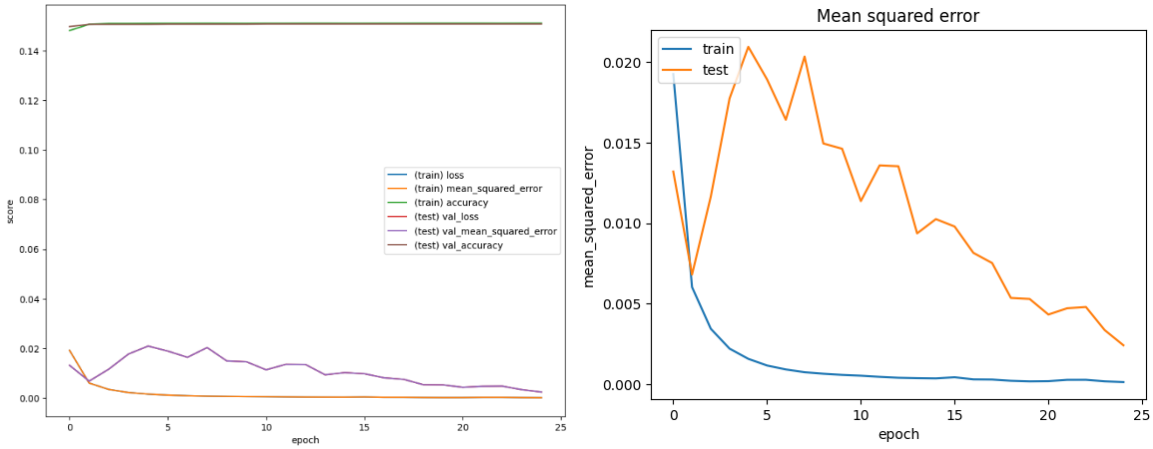
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 73: Métricas de entrenamiento Sujeto 7 Prueba 1 Modelo LSTM+CNN para distancia



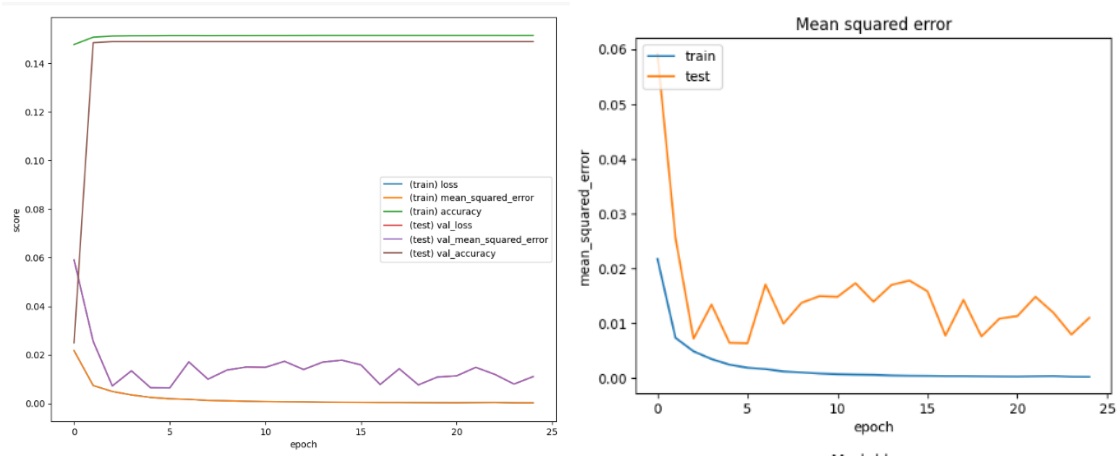
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 74: Métricas de entrenamiento Sujeto 9 Prueba 1 Modelo LSTM+CNN para distancia



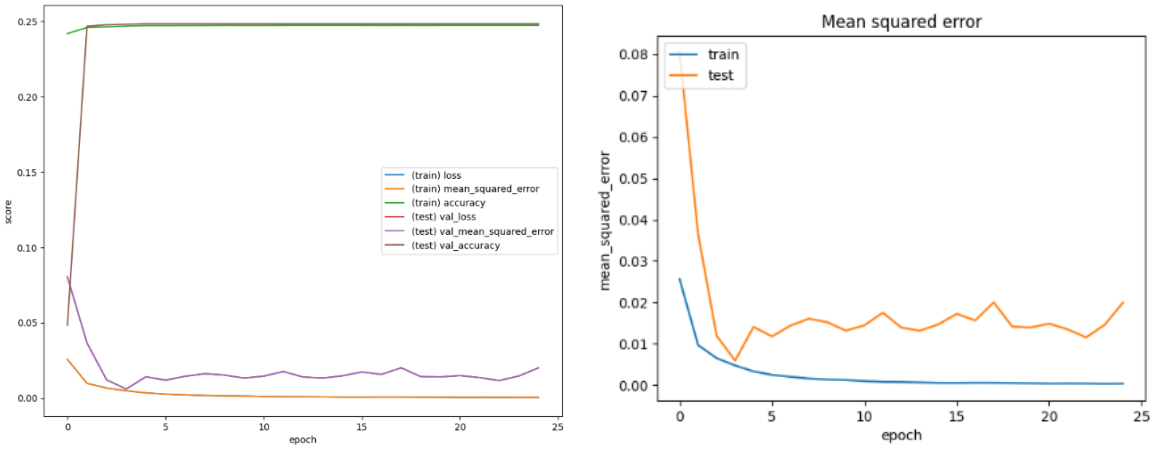
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 75: Métricas de entrenamiento Sujeto 9 Prueba 2 Modelo LSTM+CNN para distancia



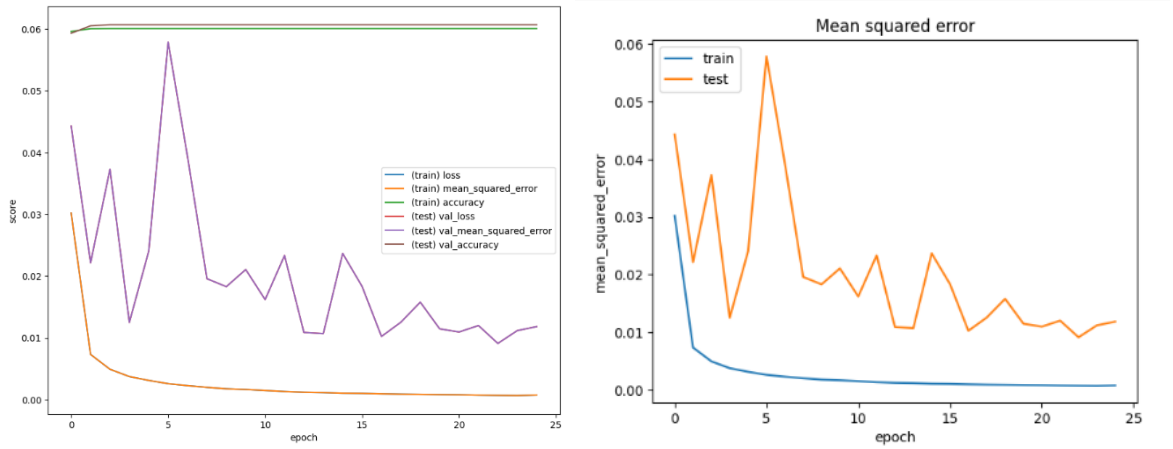
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 76: Métricas de entrenamiento Sujeto 8 Prueba 2 Modelo LSTM+CNN para distancia



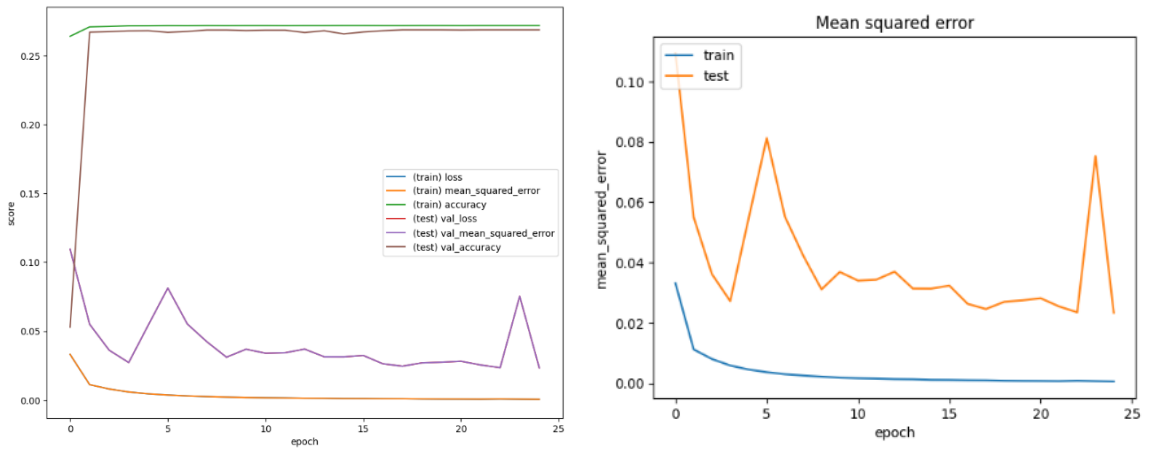
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 77: Métricas de entrenamiento Sujeto 7 Prueba 2 Modelo LSTM+CNN para distancia



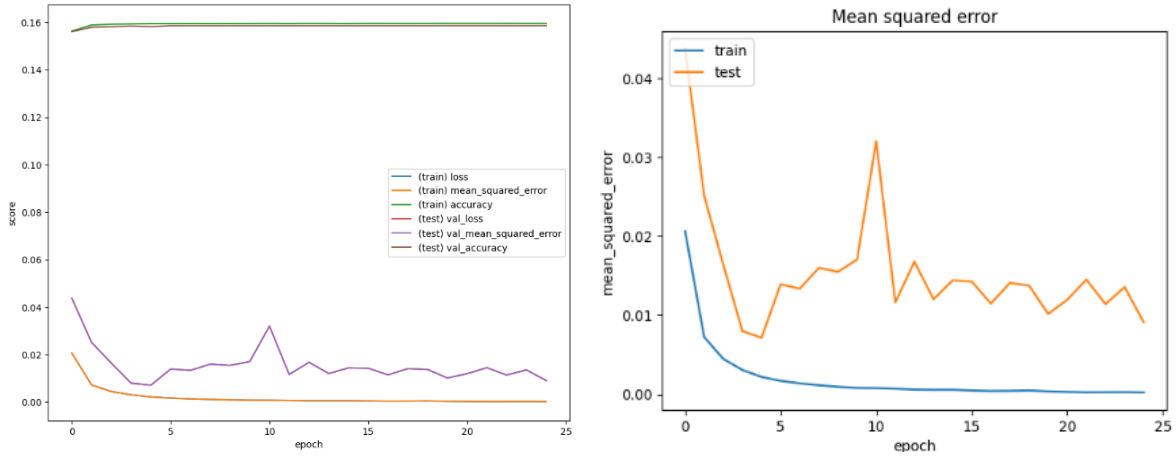
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 78: Métricas de entrenamiento Sujeto 6 Prueba 2 Modelo LSTM+CNN para distancia



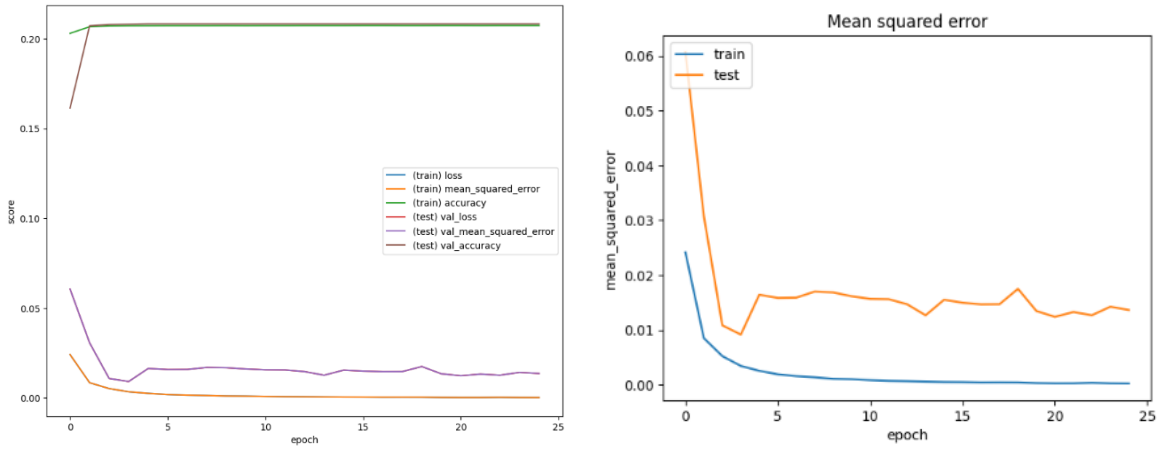
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 79: Métricas de entrenamiento Sujeto 5 Prueba 2 Modelo LSTM+CNN para distancia



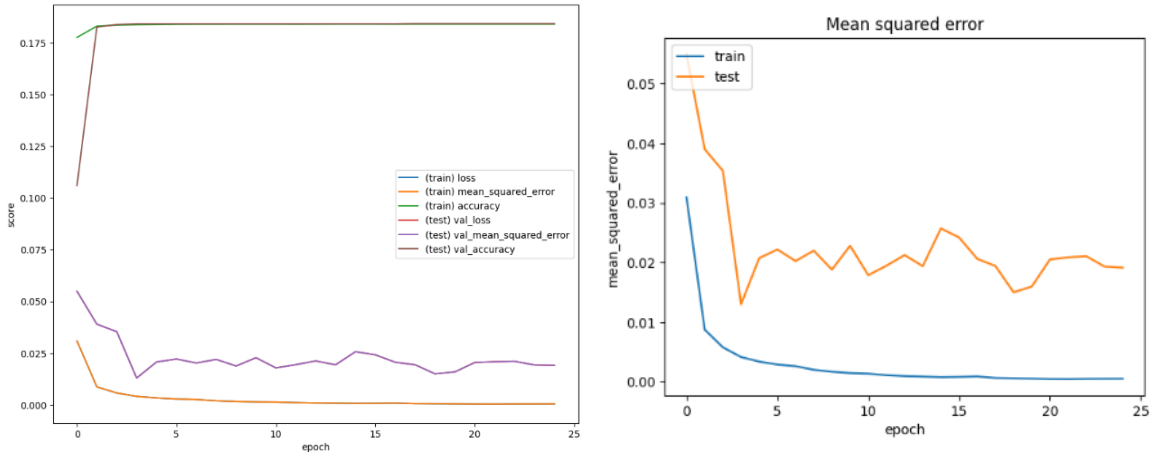
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 80: Métricas de entrenamiento Sujeto 4 Prueba 2 Modelo LSTM+CNN para distancia



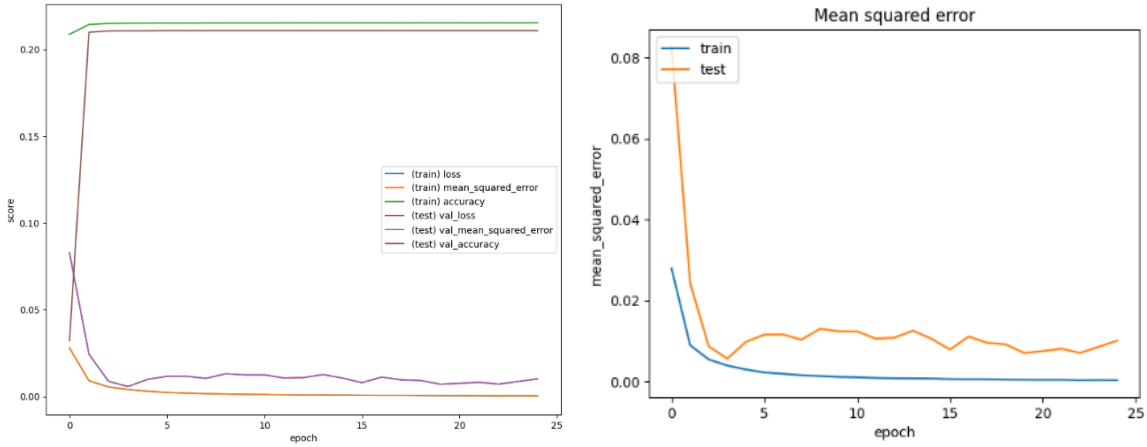
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 81: Métricas de entrenamiento Sujeto 3 Prueba 2 Modelo LSTM+CNN para distancia



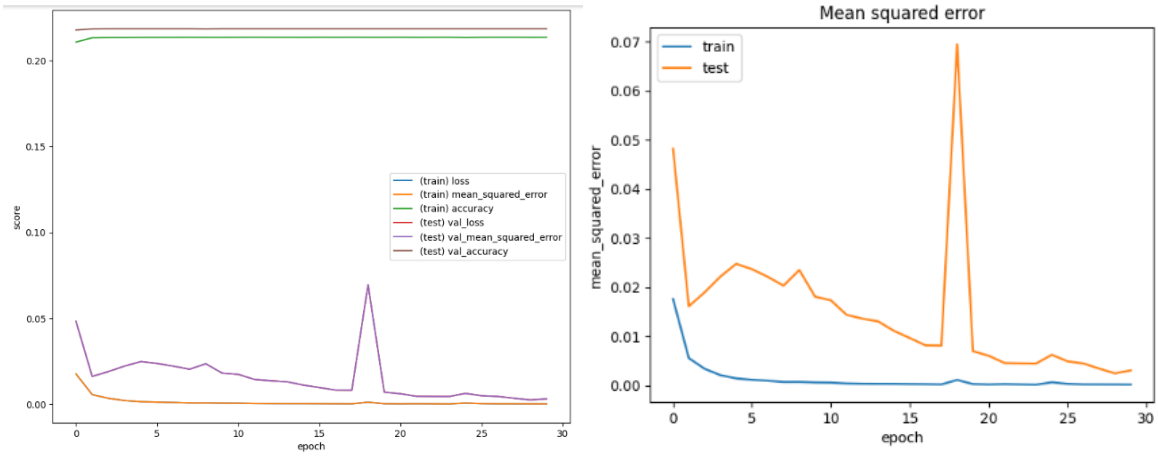
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 82: Métricas de entrenamiento Sujeto 2 Prueba 2 Modelo LSTM+CNN para distancia



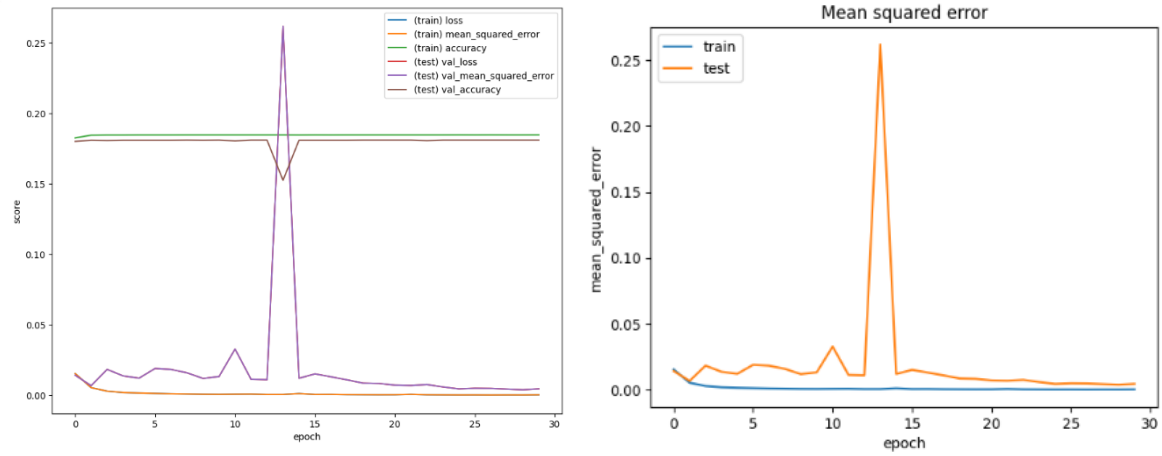
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 83: Métricas de entrenamiento Sujeto 2 Prueba 3 Modelo LSTM+CNN para distancia



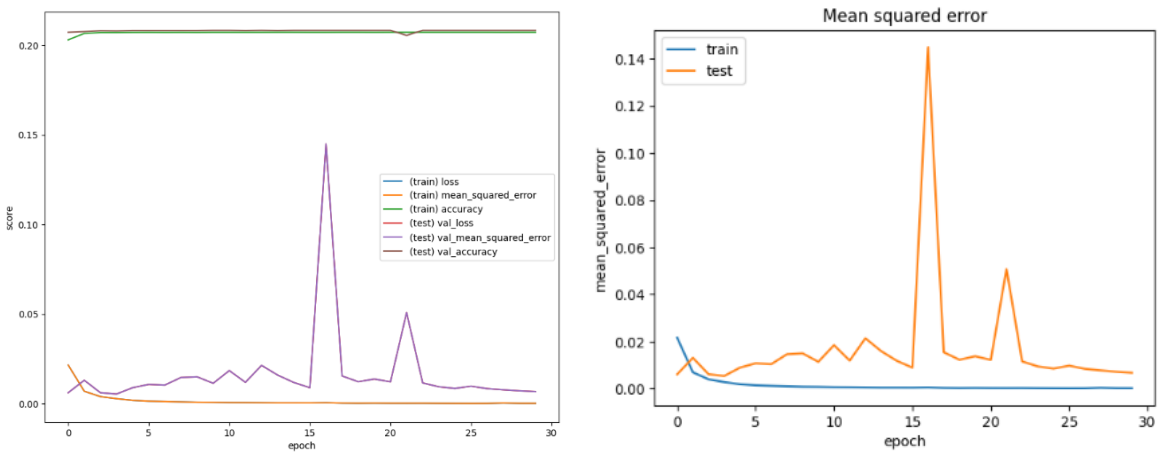
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 84: Métricas de entrenamiento Sujeto 3 Prueba 3 Modelo LSTM+CNN para distancia



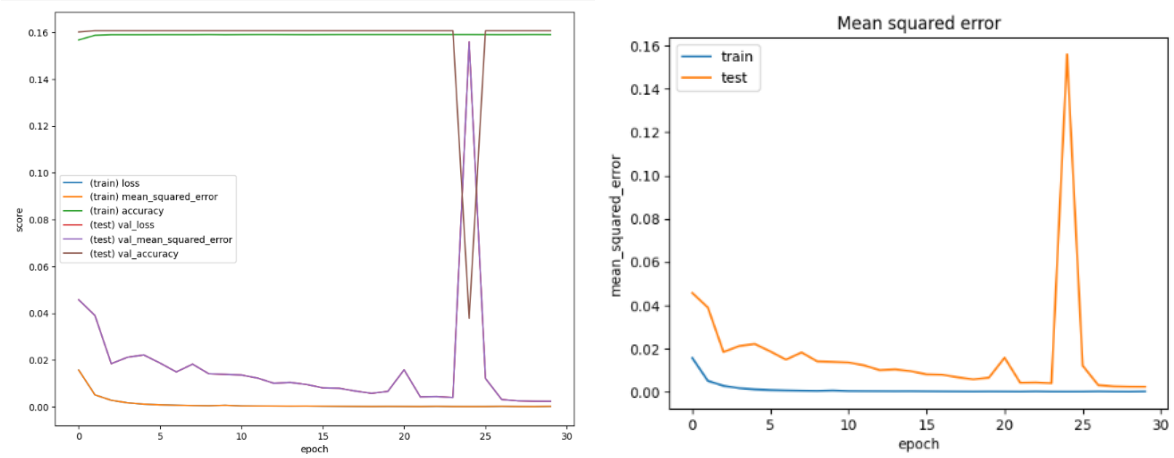
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 85: Métricas de entrenamiento Sujeto 4 Prueba 3 Modelo LSTM+CNN para distancia



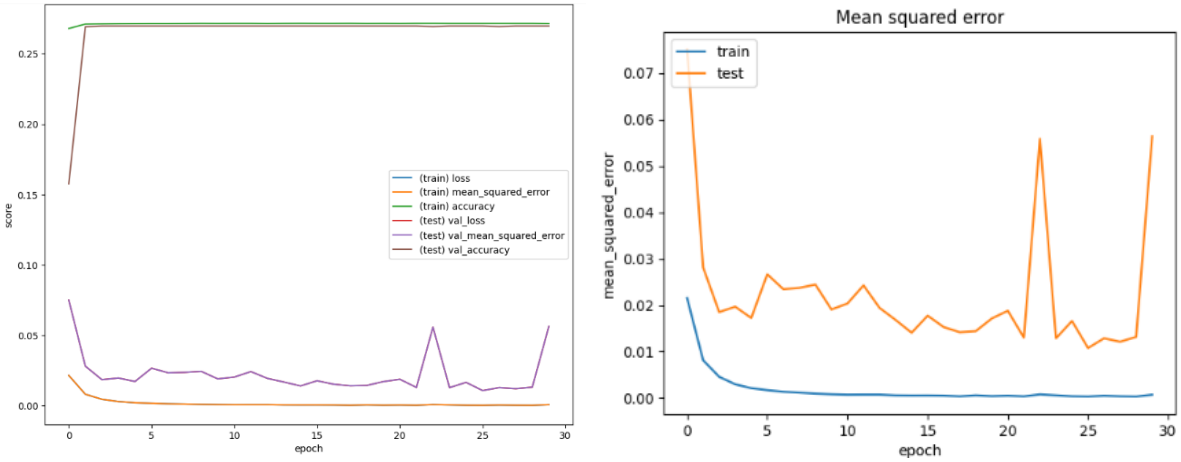
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 86: Métricas de entrenamiento Sujeto 5 Prueba 3 Modelo LSTM+CNN para distancia



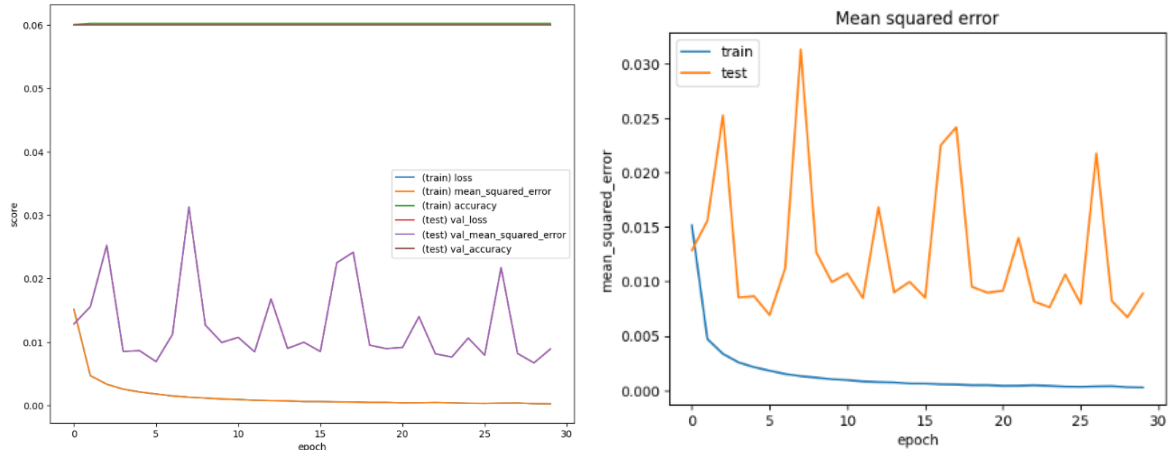
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 87: Métricas de entrenamiento Sujeto 6 Prueba 3 Modelo LSTM+CNN para distancia



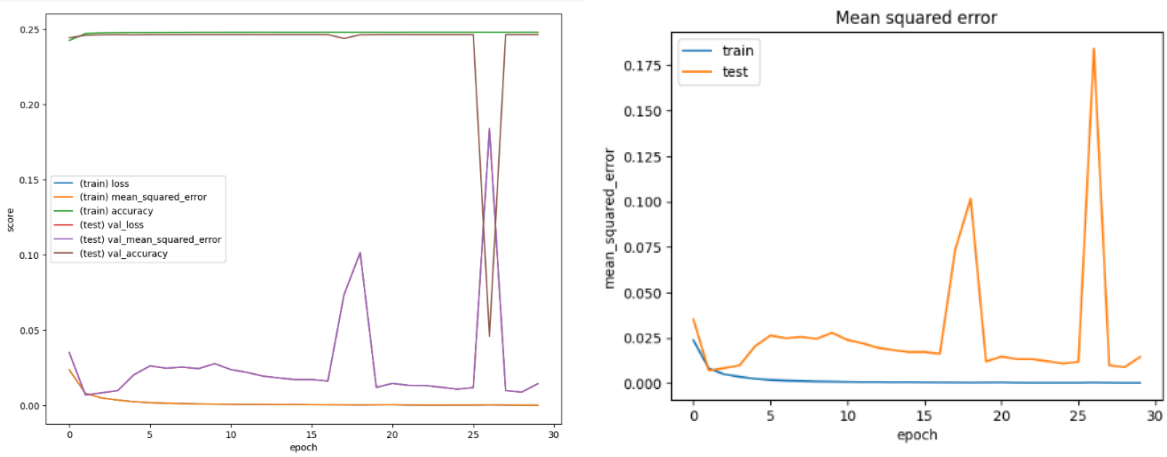
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 88: Métricas de entrenamiento Sujeto 7 Prueba 3 Modelo LSTM+CNN para distancia



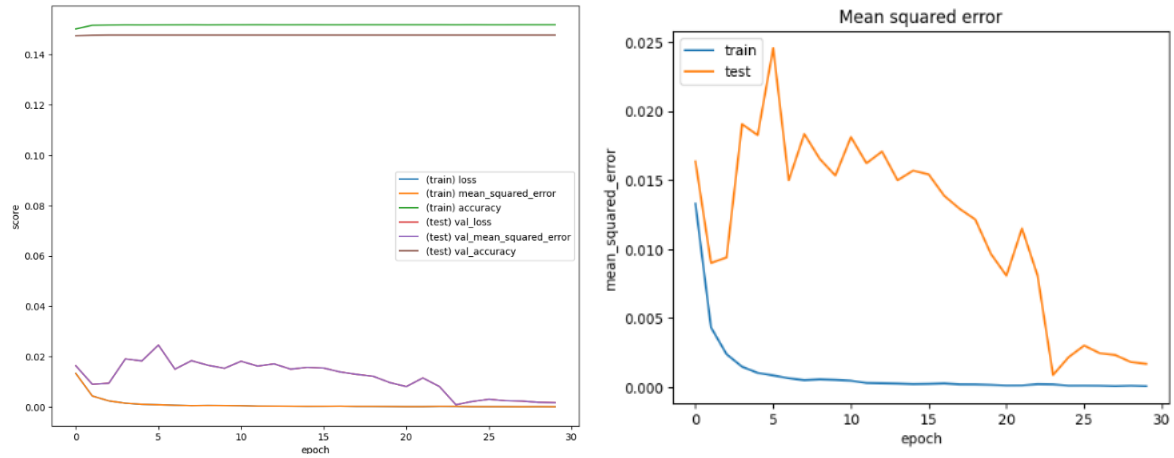
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 89: Métricas de entrenamiento Sujeto 8 Prueba 3 Modelo LSTM+CNN para distancia



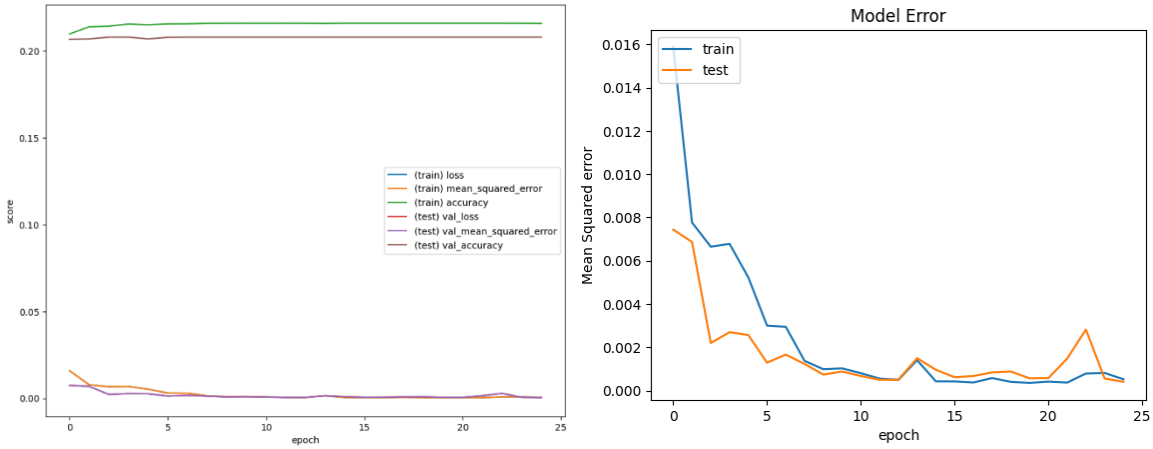
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 90: Métricas de entrenamiento Sujeto 9 Prueba 3 Modelo LSTM+CNN para distancia



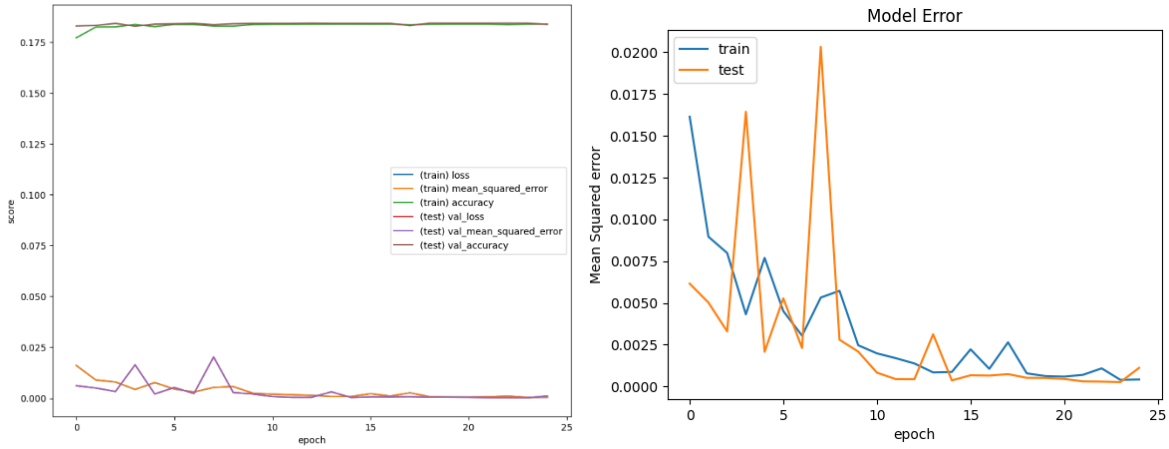
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 91: Métricas de entrenamiento Sujeto 2 Prueba 1 Modelo LSTM+FCN para distancia



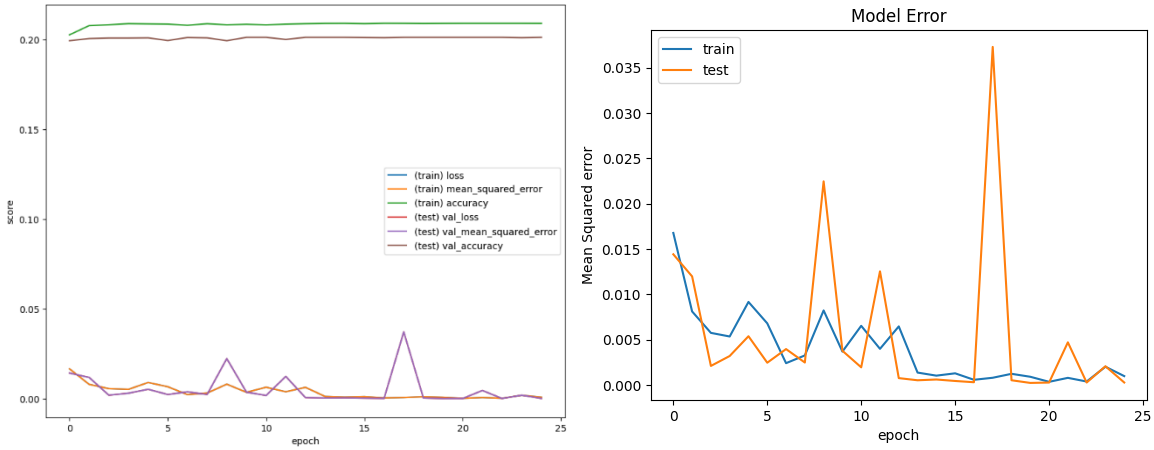
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 92: Métricas de entrenamiento Sujeto 3 Prueba 1 Modelo LSTM+FCN para distancia



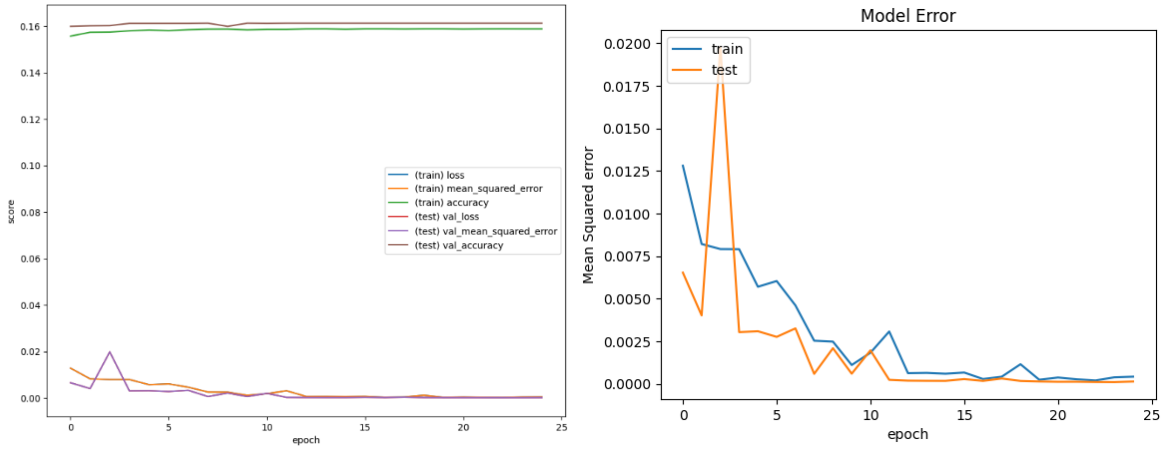
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 93: Métricas de entrenamiento Sujeto 4 Prueba 1 Modelo LSTM+FCN para distancia



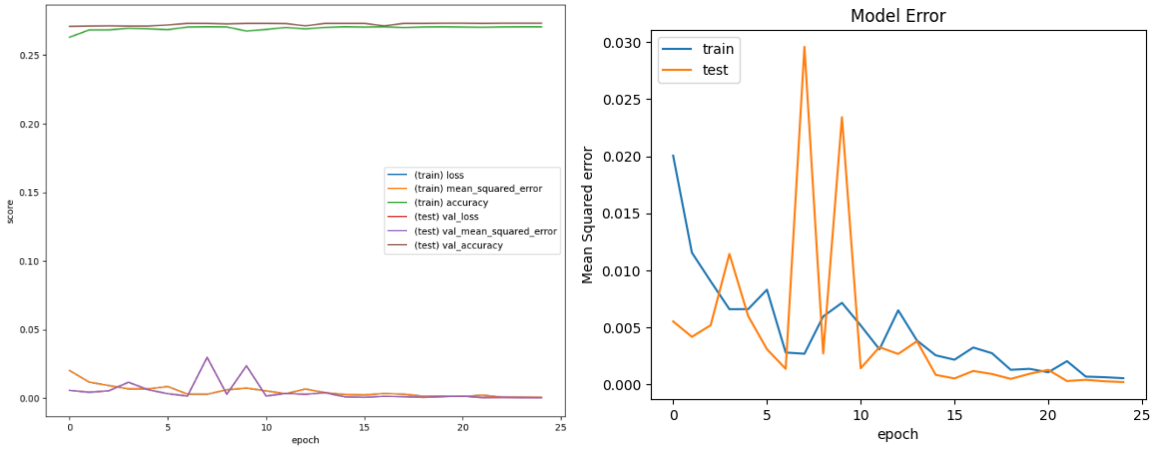
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 94: Métricas de entrenamiento Sujeto 5 Prueba 1 Modelo LSTM+FCN para distancia



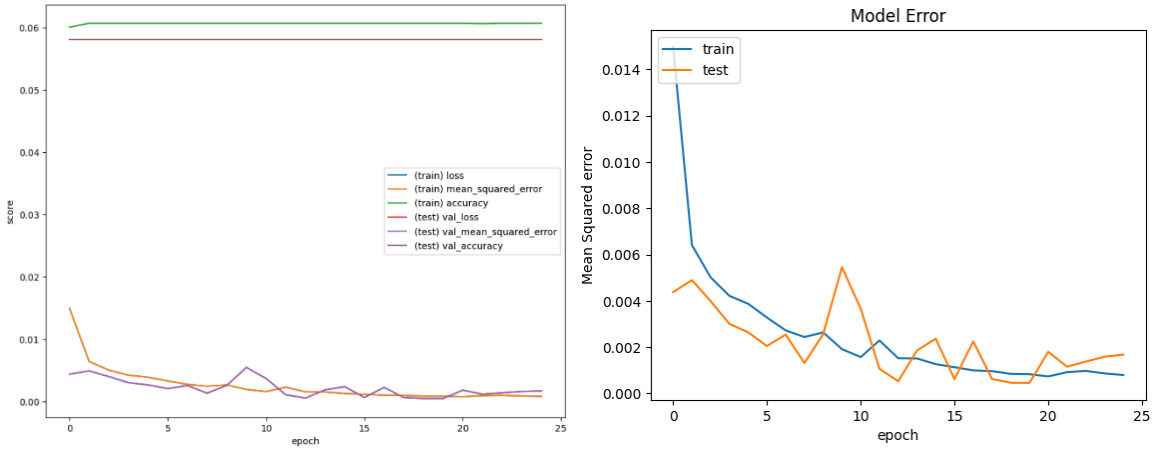
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 95: Métricas de entrenamiento Sujeto 6 Prueba 1 Modelo LSTM+FCN para distancia



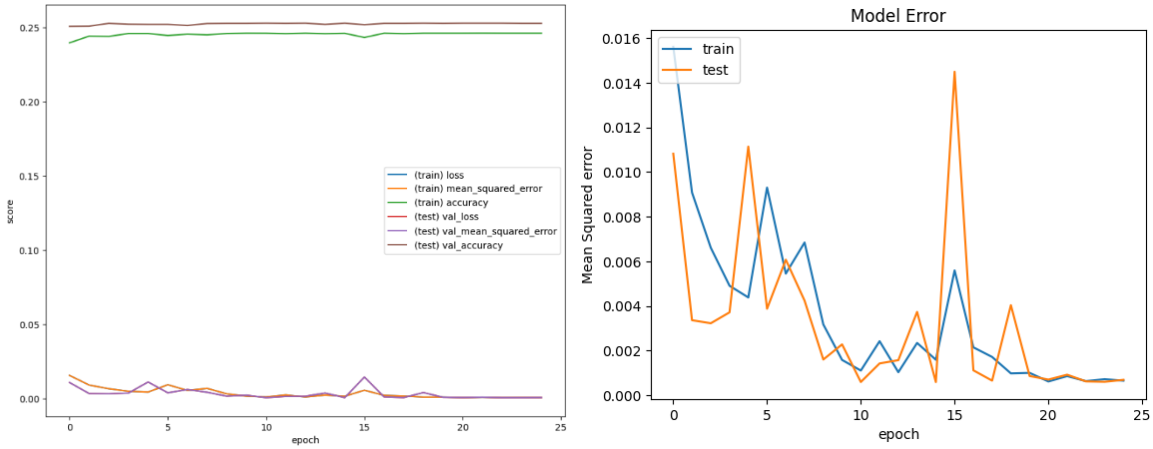
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 96: Métricas de entrenamiento Sujeto 7 Prueba 1 Modelo LSTM+FCN para distancia



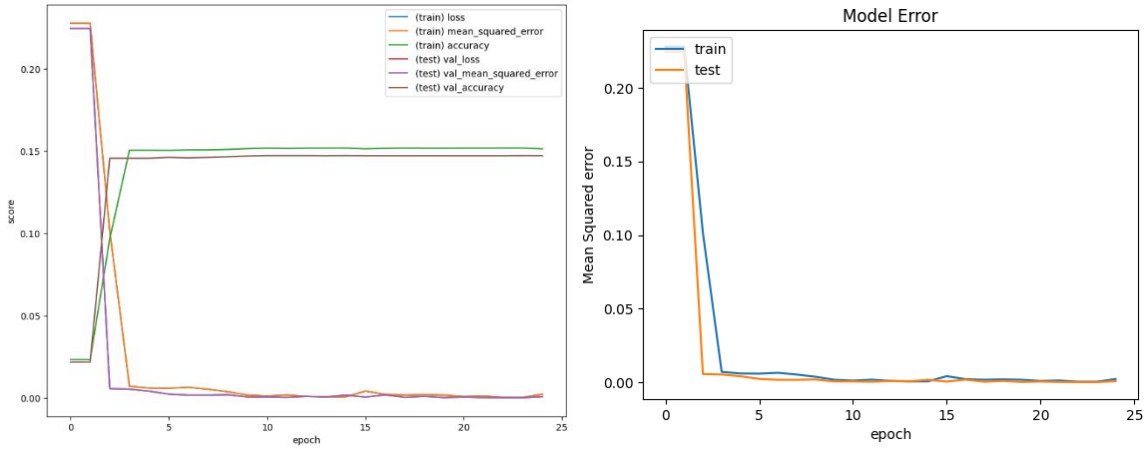
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 97: Métricas de entrenamiento Sujeto 8 Prueba 1 Modelo LSTM+FCN para distancia



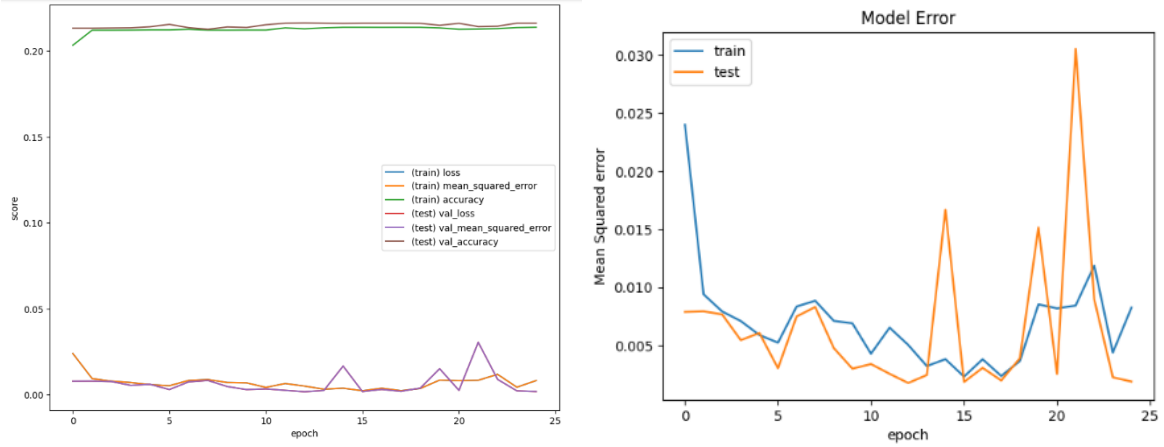
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 98: Métricas de entrenamiento Sujeto 9 Prueba 1 Modelo LSTM+FCN para distancia



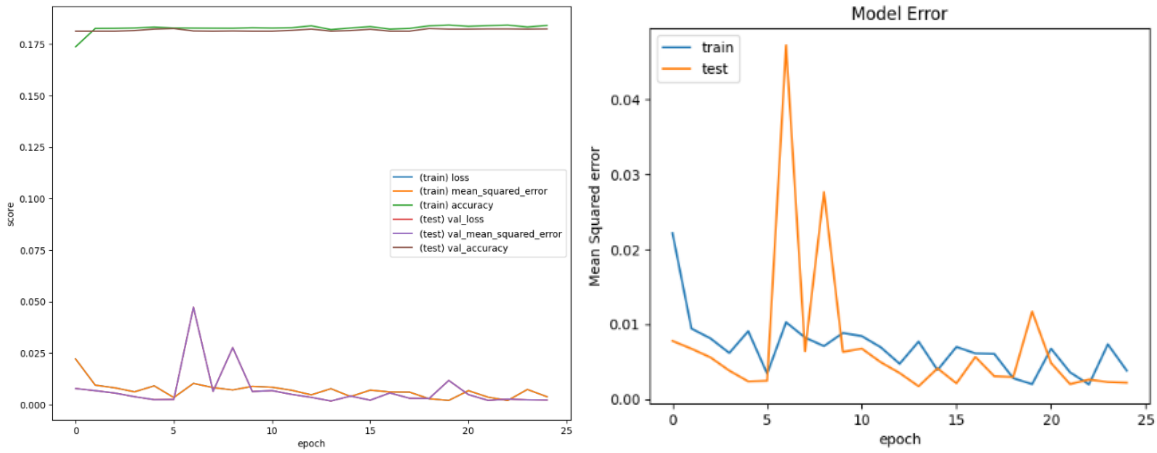
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 99: Métricas de entrenamiento Sujeto 2 Prueba 2 Modelo LSTM+FCN para distancia



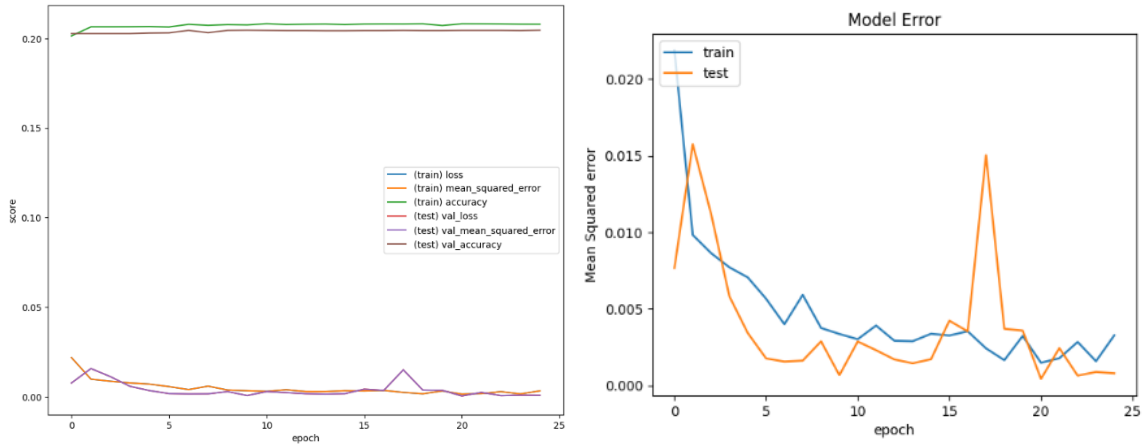
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 100: Métricas de entrenamiento Sujeto 3 Prueba 2 Modelo LSTM+FCN para distancia



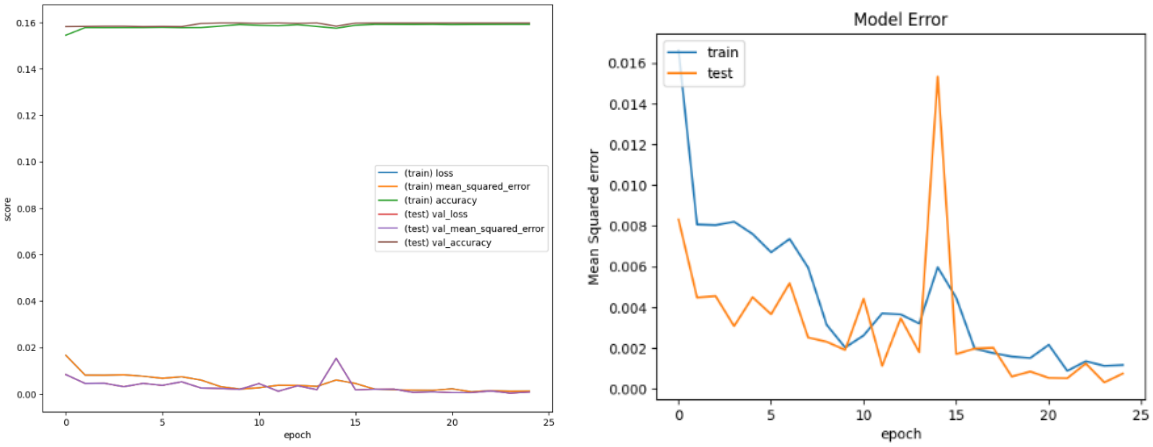
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 101: Métricas de entrenamiento Sujeto 4 Prueba 2 Modelo LSTM+FCN para distancia



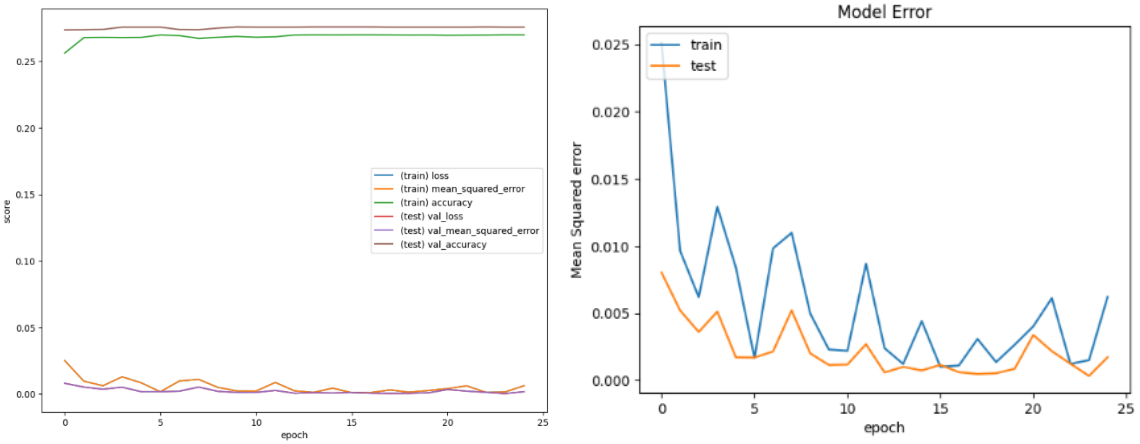
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 102: Métricas de entrenamiento Sujeto 5 Prueba 2 Modelo LSTM+FCN para distancia



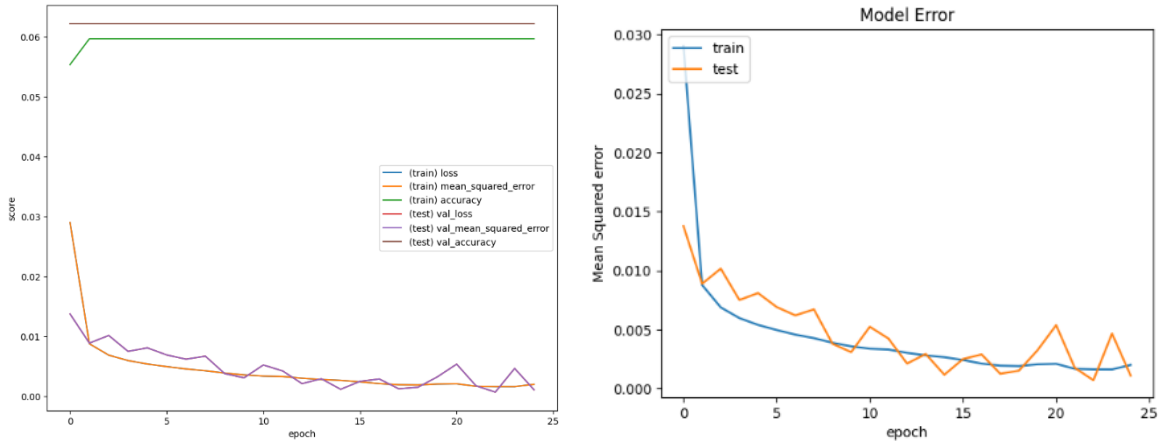
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 103: Métricas de entrenamiento Sujeto 6 Prueba 2 Modelo LSTM+FCN para distancia



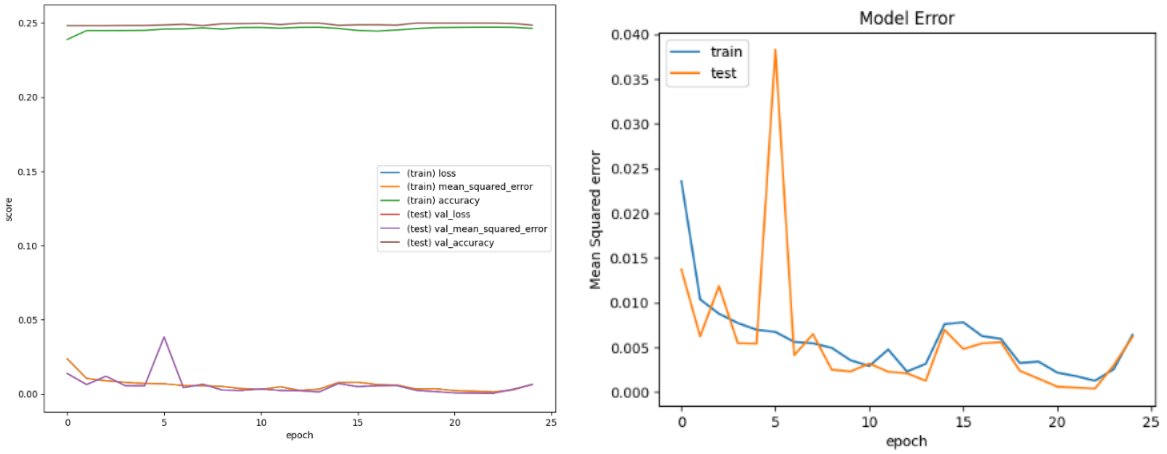
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 104: Métricas de entrenamiento Sujeto 7 Prueba 2 Modelo LSTM+FCN para distancia



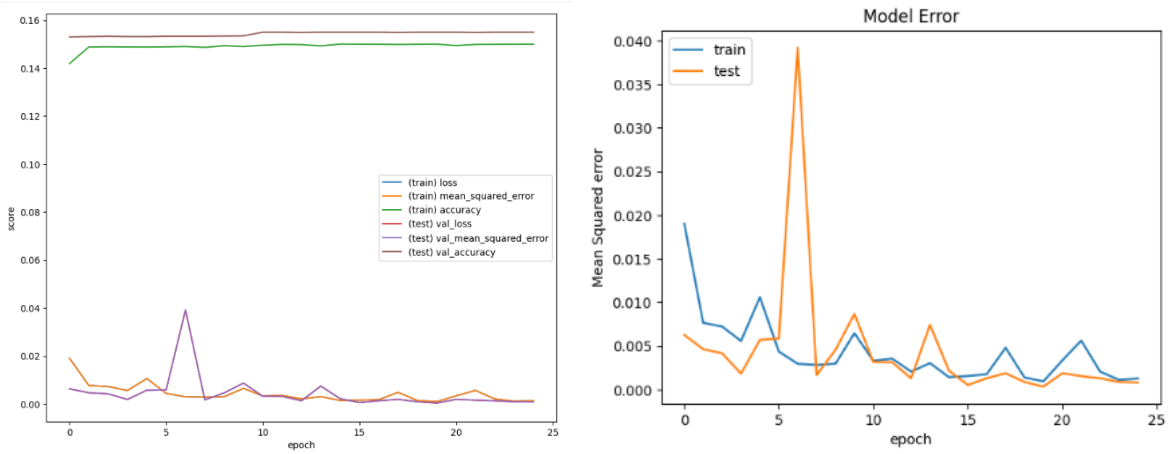
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 105: Métricas de entrenamiento Sujeto 8 Prueba 2 Modelo LSTM+FCN para distancia



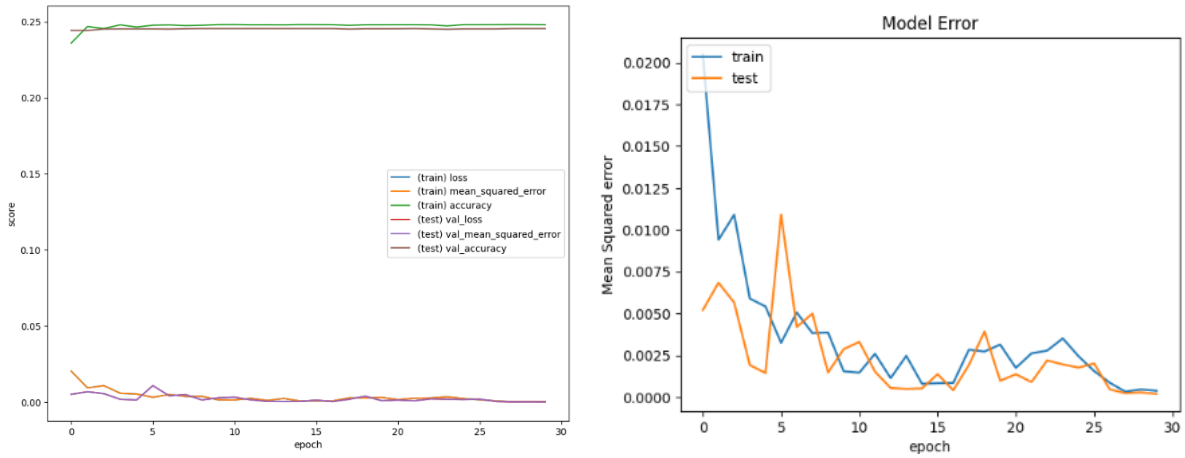
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 106: Métricas de entrenamiento Sujeto 9 Prueba 2 Modelo LSTM+FCN para distancia



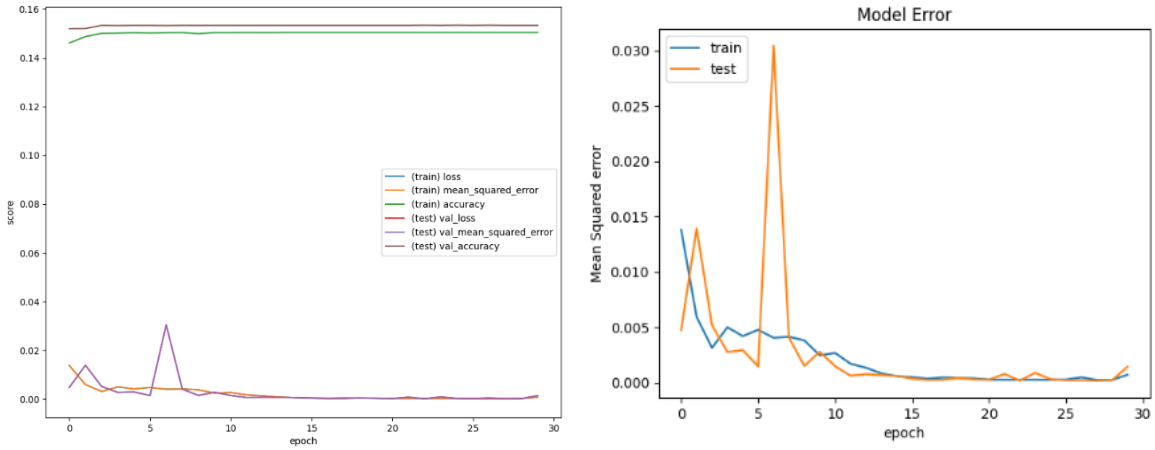
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 107: Métricas de entrenamiento Sujeto 8 Prueba 3 Modelo LSTM+FCN para distancia



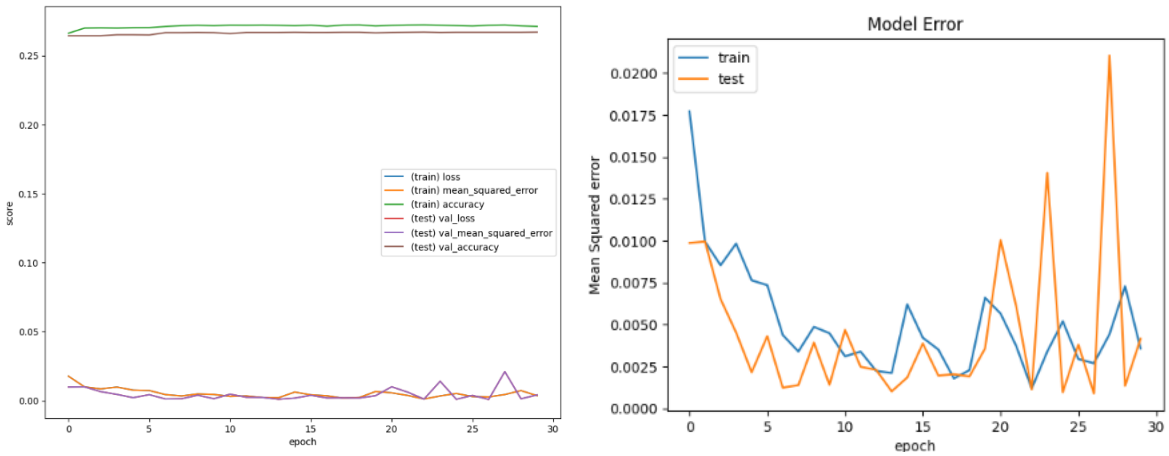
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 108: Métricas de entrenamiento Sujeto 9 Prueba 3 Modelo LSTM+FCN para distancia



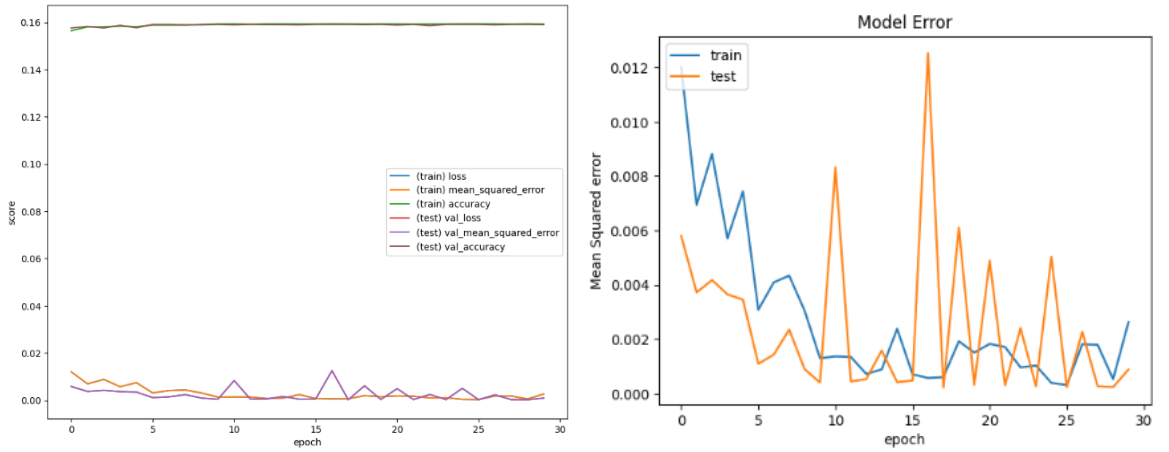
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 109: Métricas de entrenamiento Sujeto 6 Prueba 3 Modelo LSTM+FCN para distancia



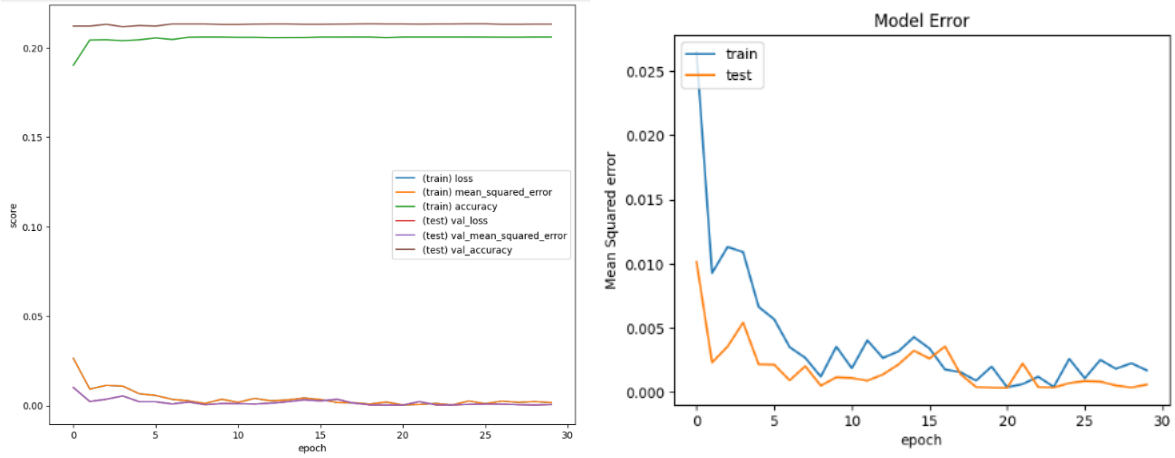
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 110: Métricas de entrenamiento Sujeto 5 Prueba 3 Modelo LSTM+FCN para distancia



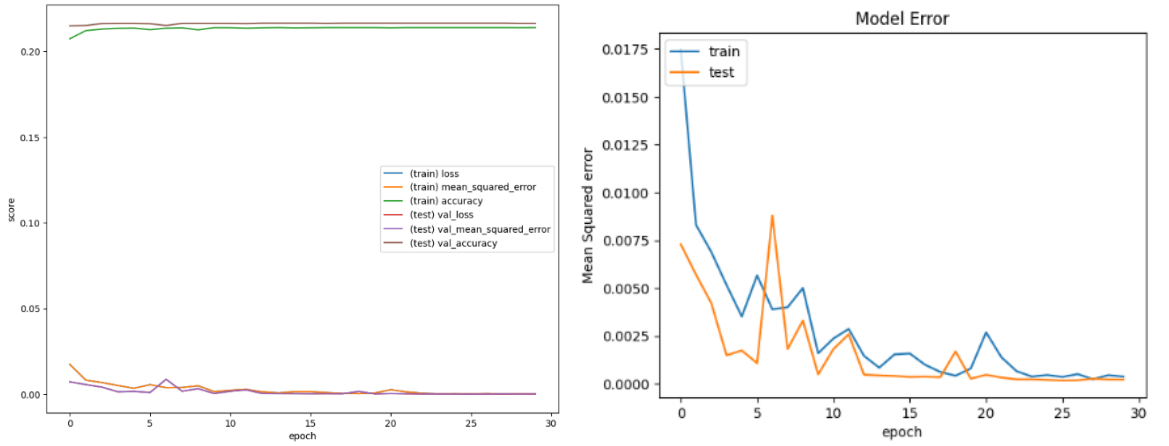
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 111: Métricas de entrenamiento Sujeto 4 Prueba 3 Modelo LSTM+FCN para distancia



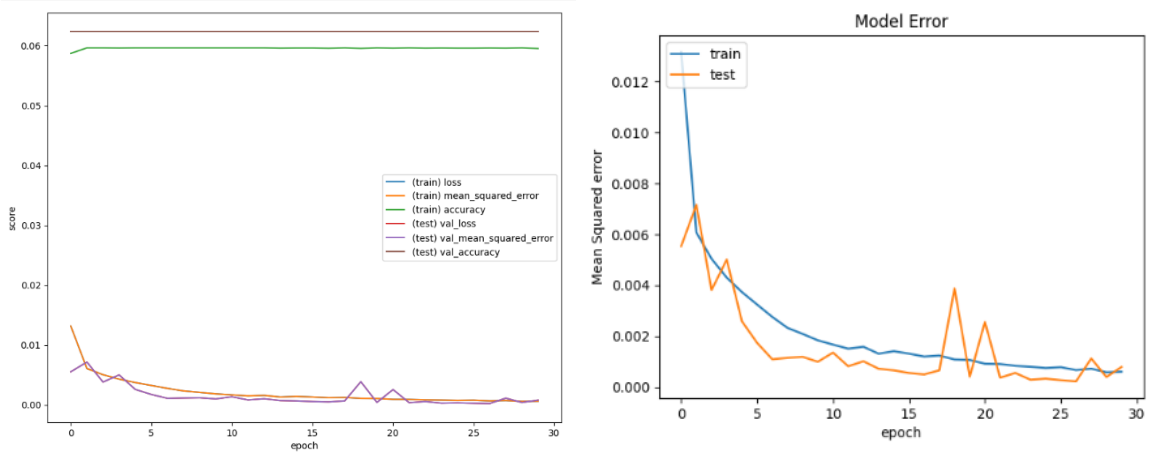
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 112: Métricas de entrenamiento Sujeto 2 Prueba 3 Modelo LSTM+FCN para distancia



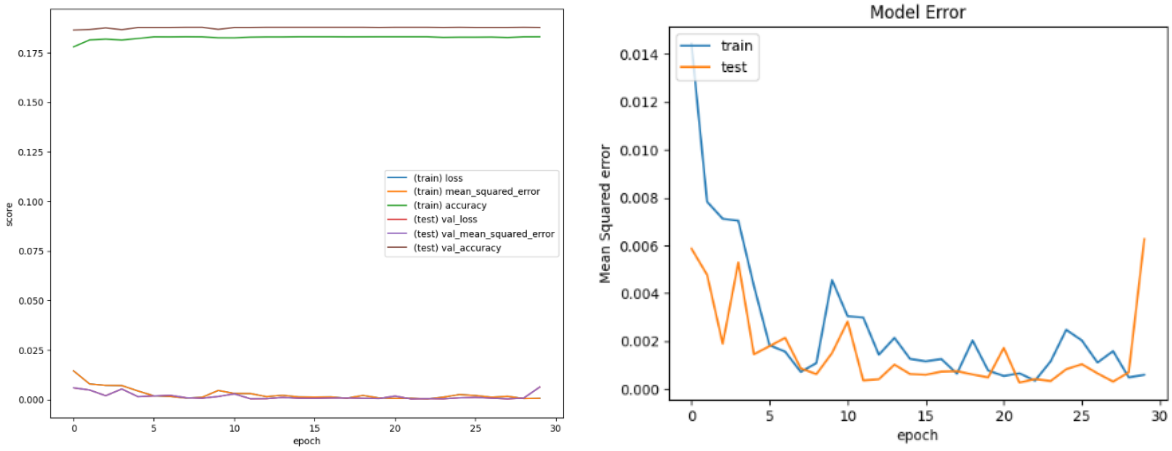
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 113: Métricas de entrenamiento Sujeto 7 Prueba 3 Modelo LSTM+FCN para distancia



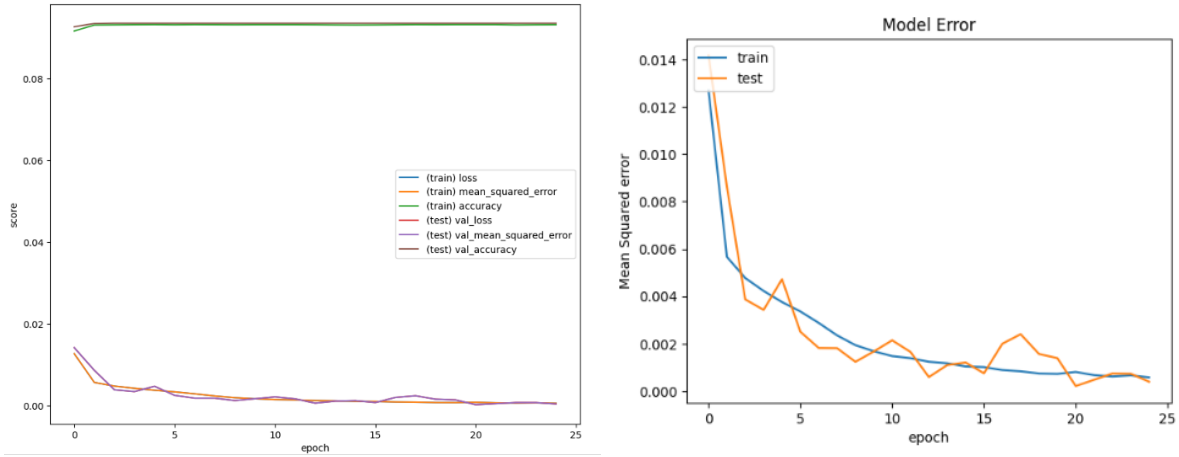
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 114: Métricas de entrenamiento Sujeto 3 Prueba 3 Modelo LSTM+FCN para distancia



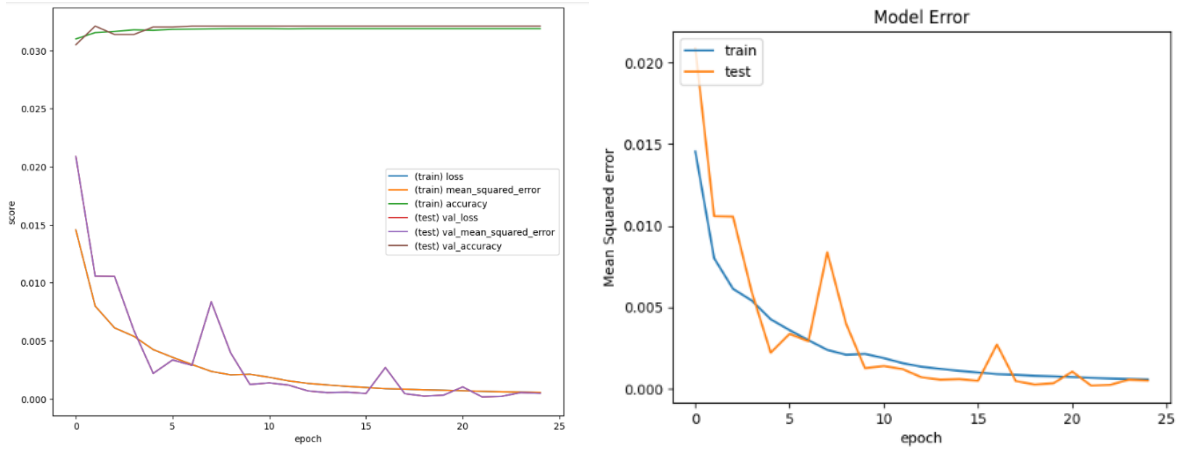
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 115: Métricas de entrenamiento Sujeto 9 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



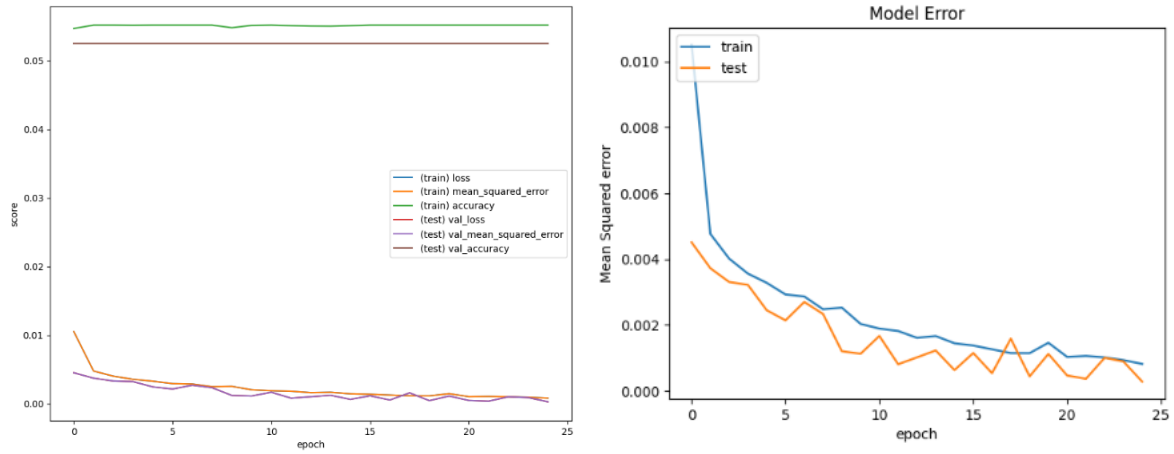
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 116: Métricas de entrenamiento Sujeto 3 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



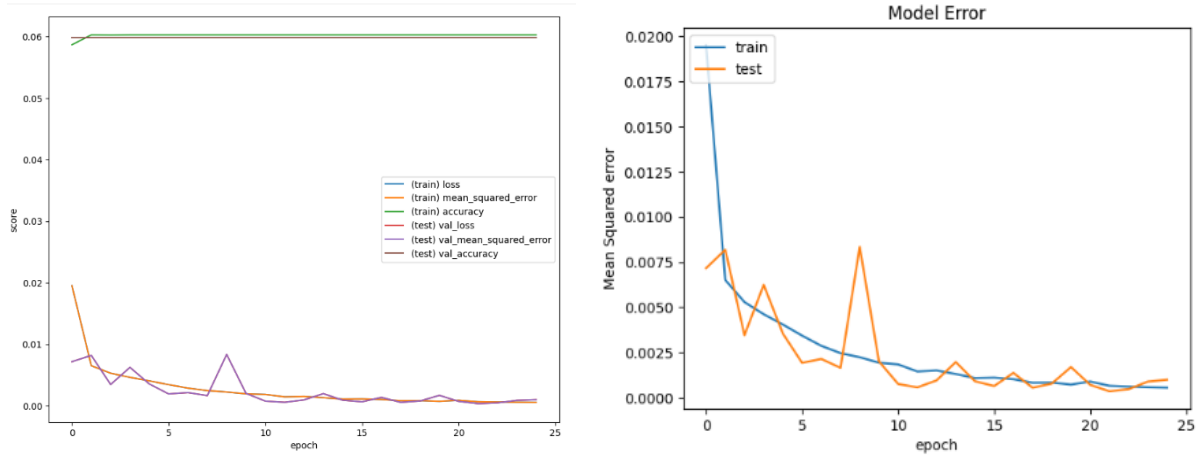
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 117: Métricas de entrenamiento Sujeto 5 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



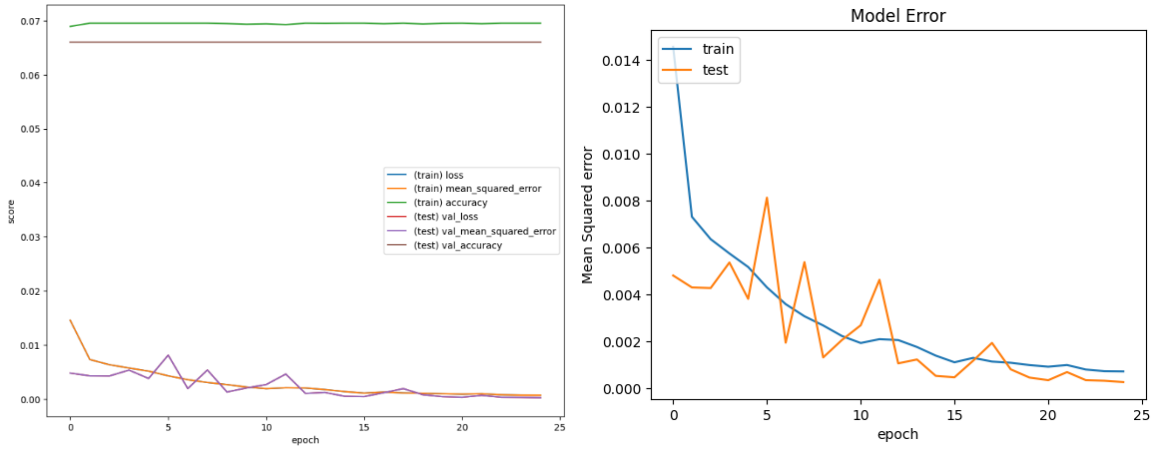
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 118: Métricas de entrenamiento Sujeto 7 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



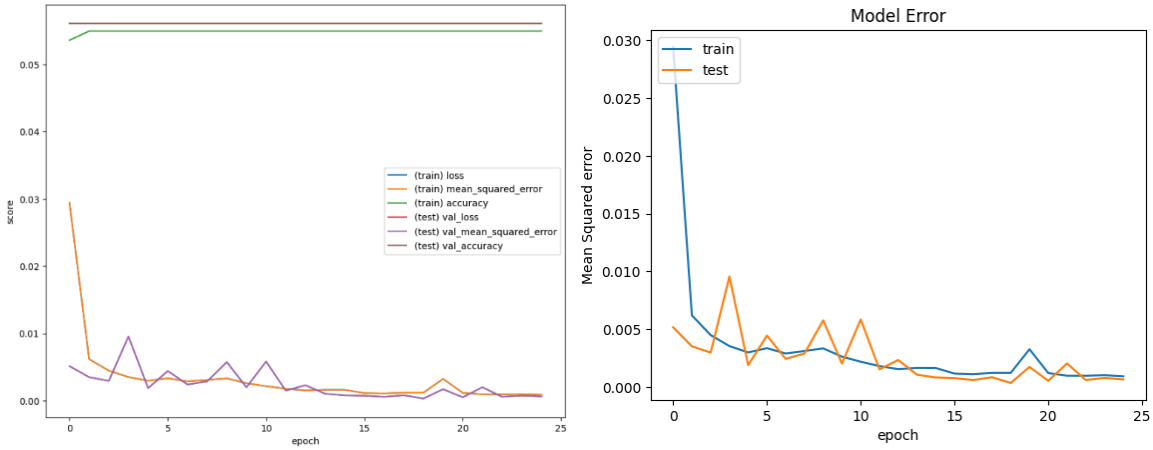
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 119: Métricas de entrenamiento Sujeto 8 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



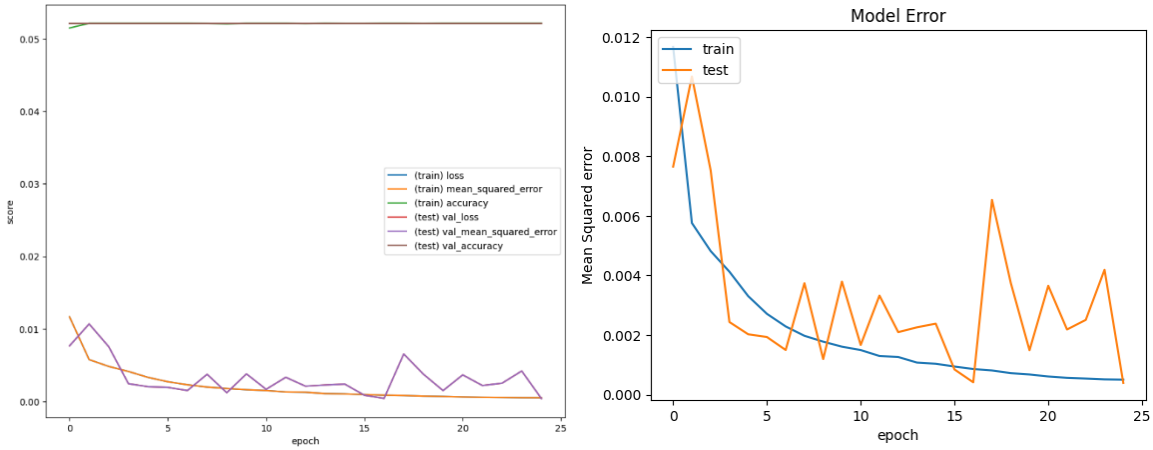
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 120: Métricas de entrenamiento Sujeto 2 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



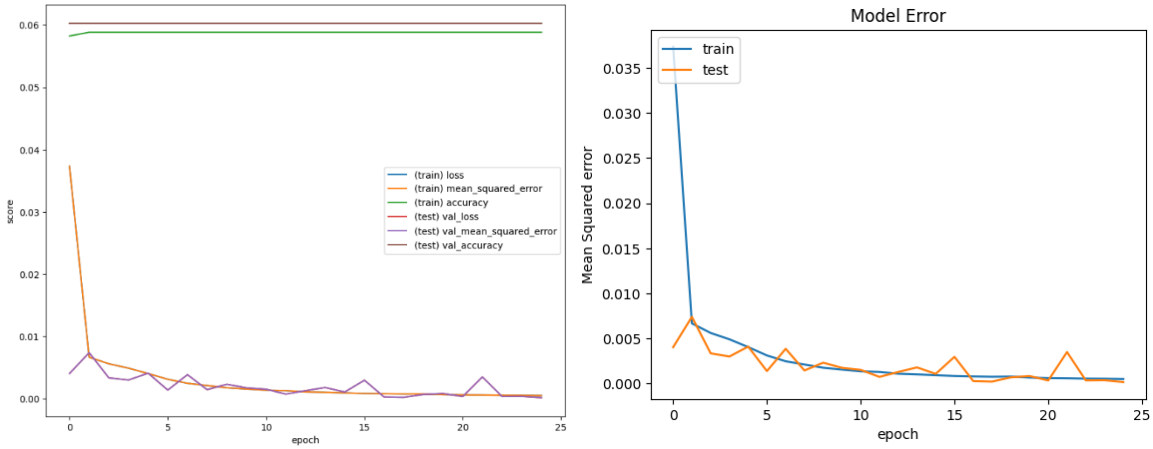
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 121: Métricas de entrenamiento Sujeto 4 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



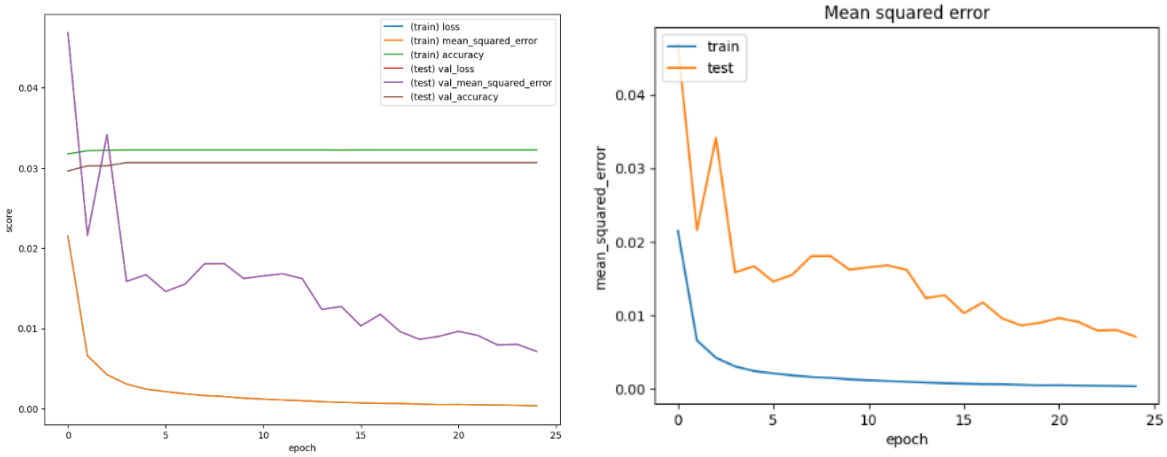
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 122: Métricas de entrenamiento Sujeto 6 Prueba 1 Modelo LSTM+FCN para tamaño de zancada



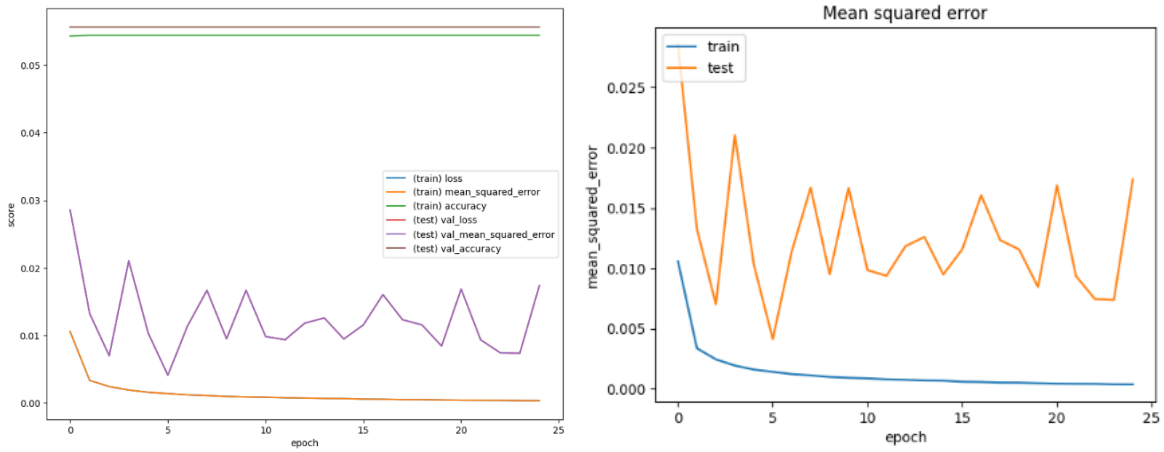
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 123: Métricas de entrenamiento Sujeto 3 Prueba 1 Modelo LSTM+CNN para tamaño de zancada



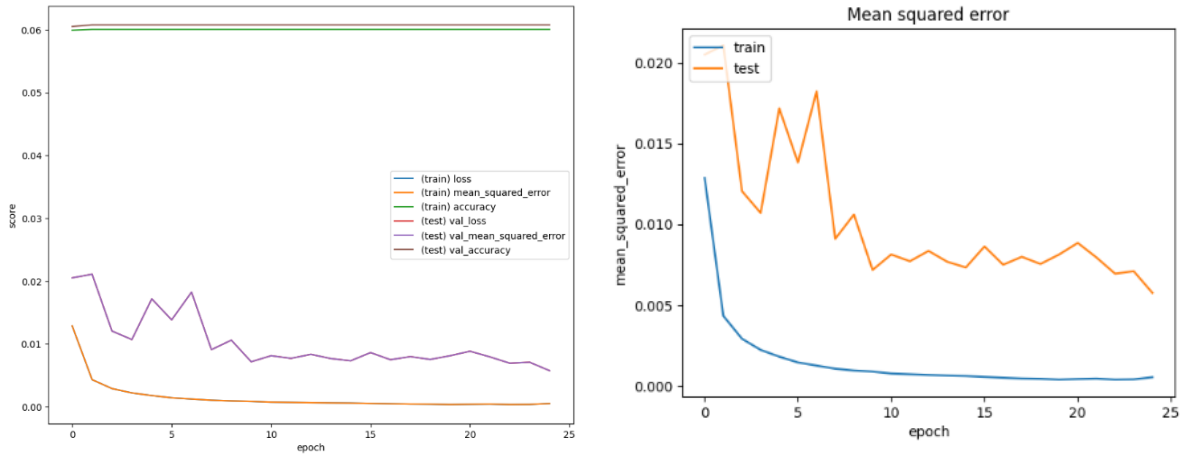
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 124: Métricas de entrenamiento Sujeto 5 Prueba 1 Modelo LSTM+CNN para tamaño de zancada



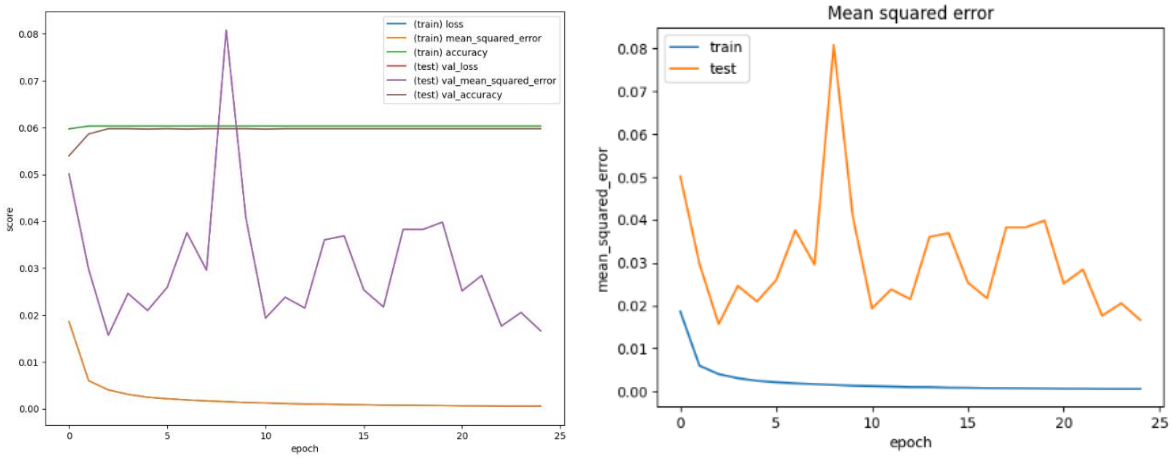
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 125: Métricas de entrenamiento Sujeto 7 Prueba 1 Modelo LSTM+CNN para tamaño de zancada



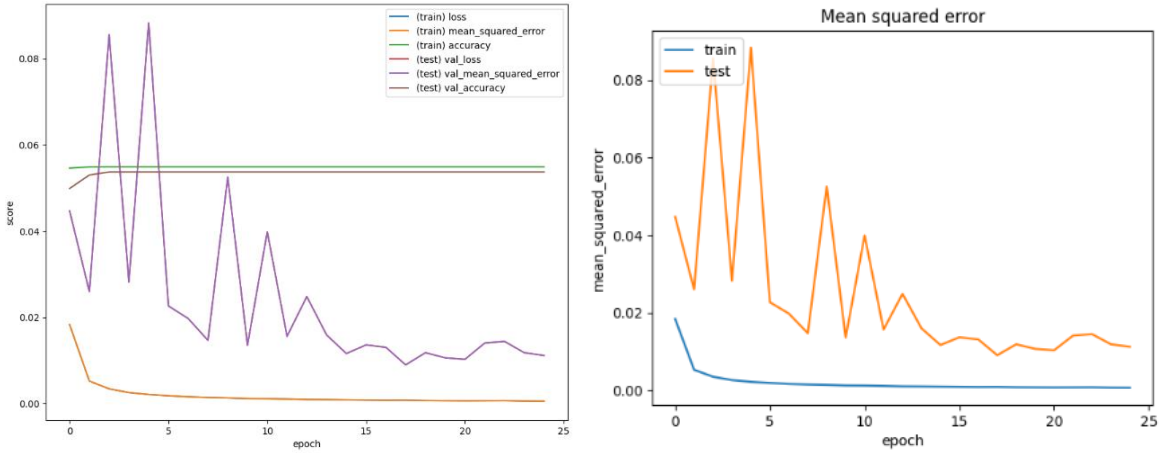
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 126: Métricas de entrenamiento Sujeto 7 Prueba 2 Modelo LSTM+CNN para tamaño de zancada



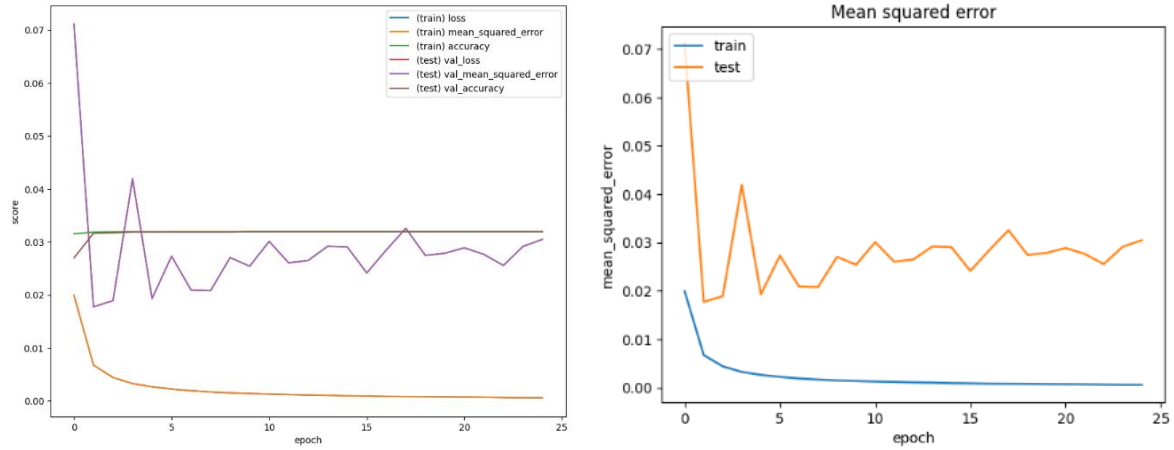
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 127: Métricas de entrenamiento Sujeto 5 Prueba 2 Modelo LSTM+CNN para tamaño de zancada



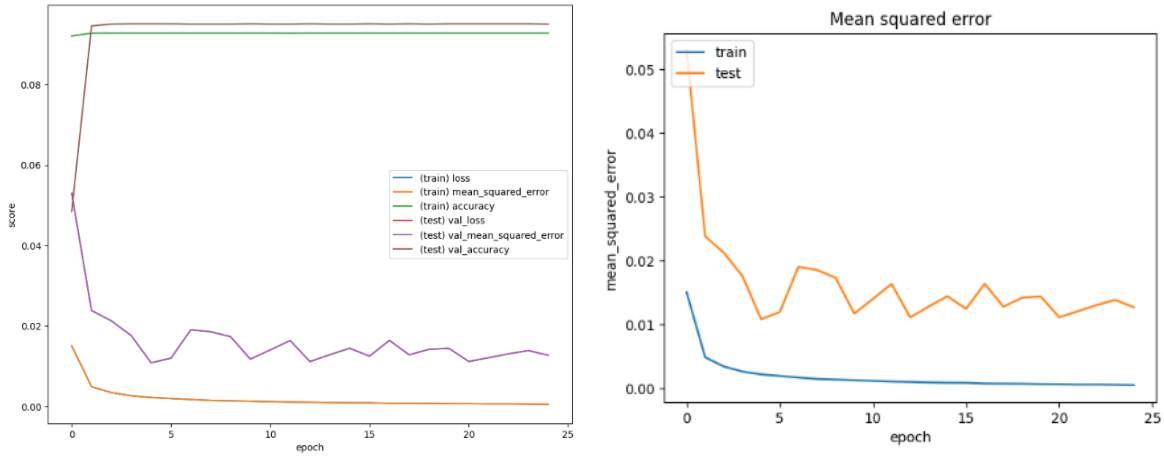
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 128: Métricas de entrenamiento Sujeto 3 Prueba 2 Modelo LSTM+CNN para tamaño de zancada



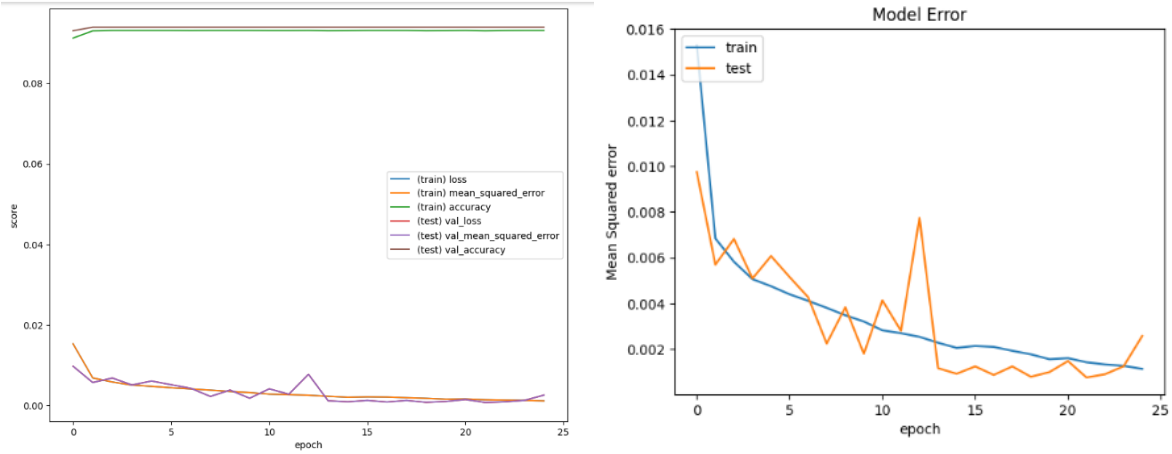
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 129: Métricas de entrenamiento Sujeto 9 Prueba 2 Modelo LSTM+CNN para tamaño de zancada



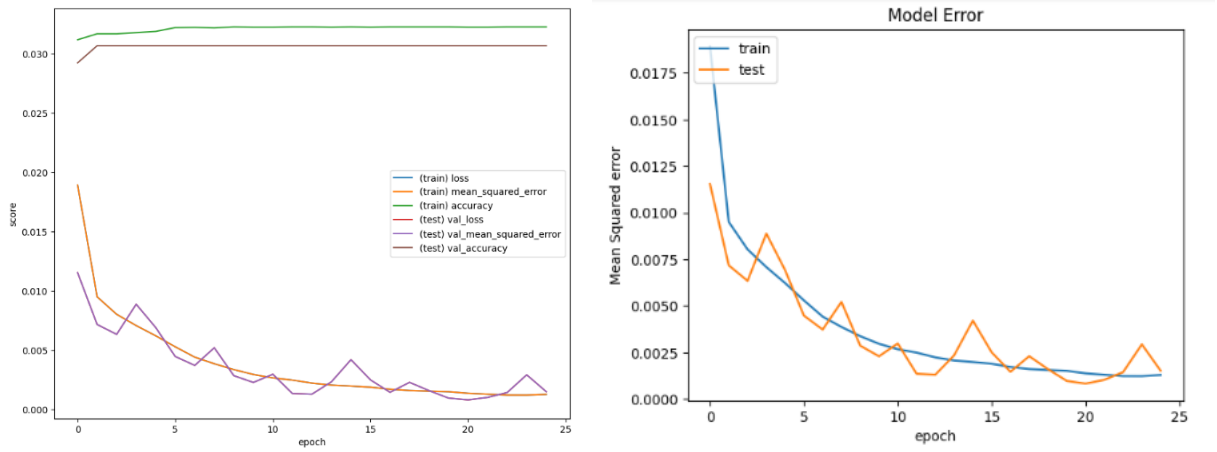
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 130: Métricas de entrenamiento Sujeto 9 Prueba 2 Modelo LSTM+FCN para tamaño de zancada



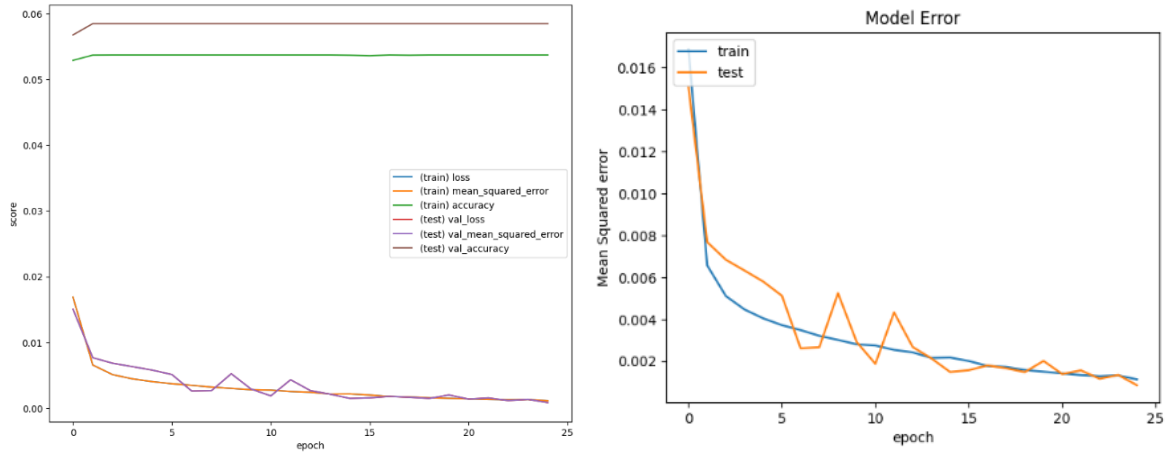
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 131: Métricas de entrenamiento Sujeto 3 Prueba 2 Modelo LSTM+FCN para tamaño de zancada



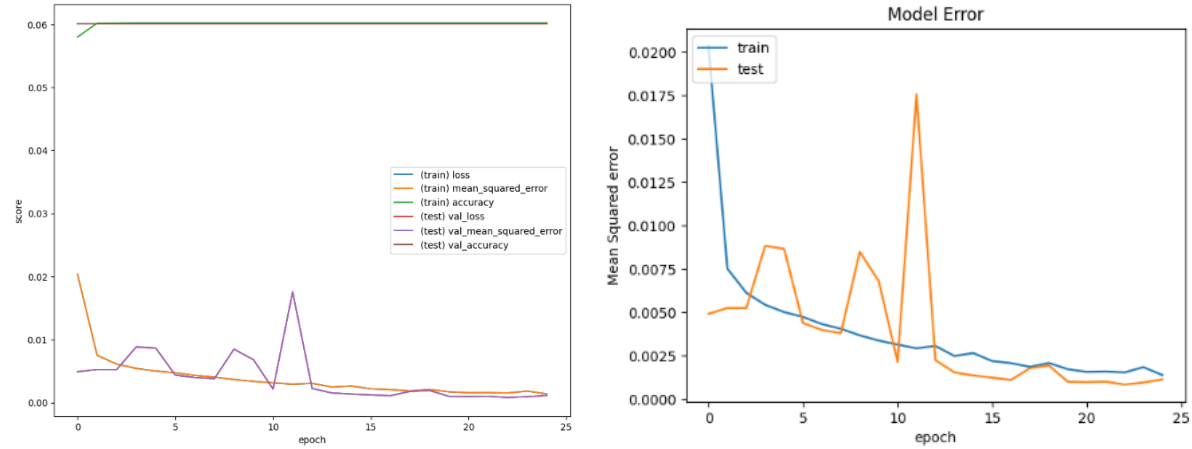
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 132: Métricas de entrenamiento Sujeto 5 Prueba 2 Modelo LSTM+FCN para tamaño de zancada



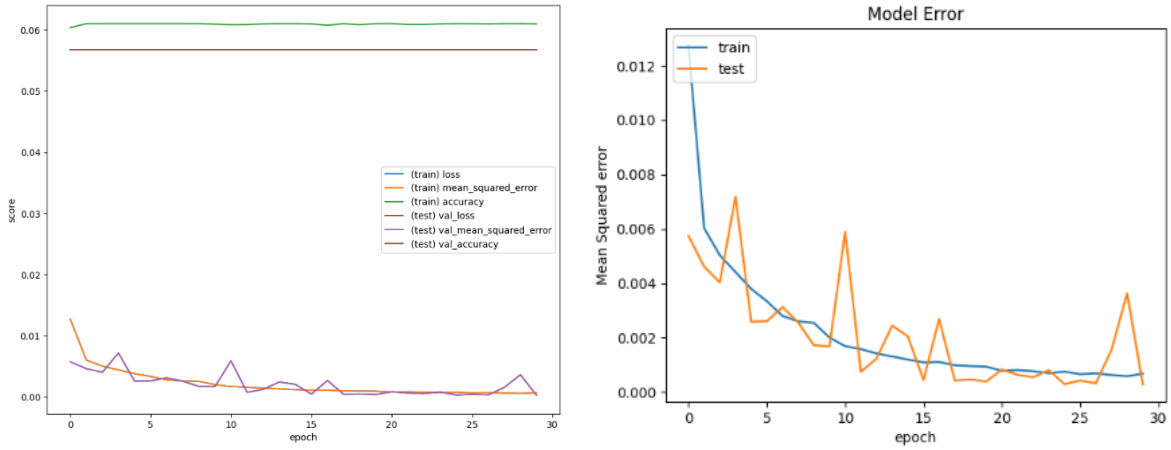
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 133: Métricas de entrenamiento Sujeto 7 Prueba 2 Modelo LSTM+FCN para tamaño de zancada



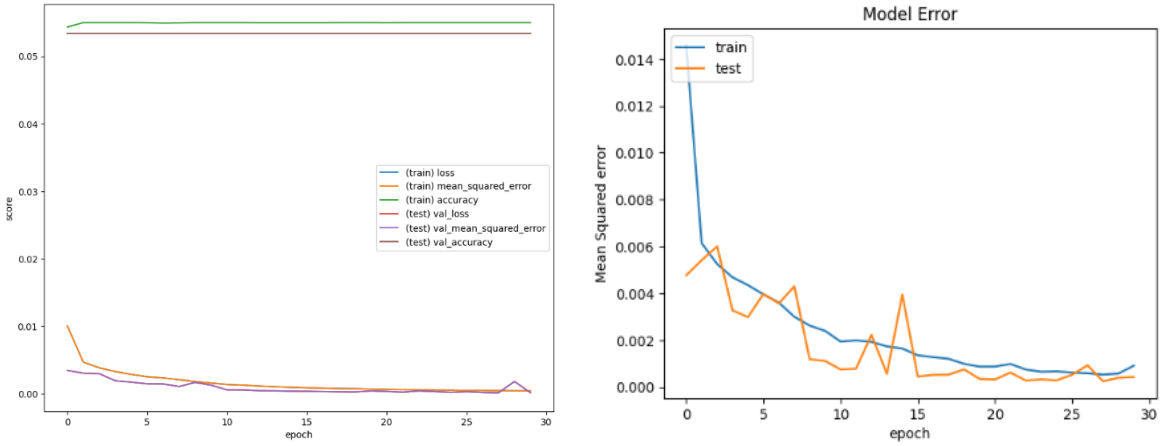
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 134: Métricas de entrenamiento Sujeto 7 Prueba 3 Modelo LSTM+FCN para tamaño de zancada



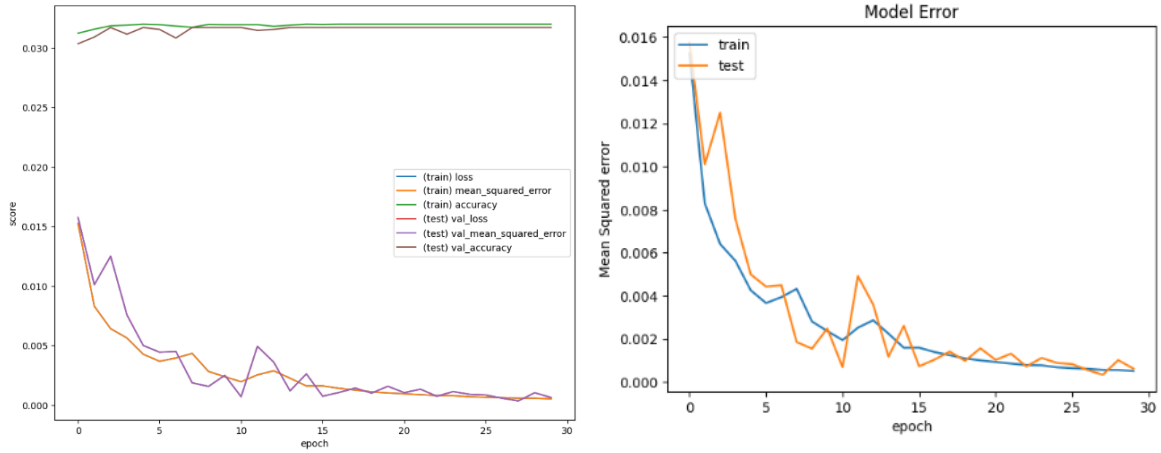
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 135: Métricas de entrenamiento Sujeto 5 Prueba 3 Modelo LSTM+FCN para tamaño de zancada



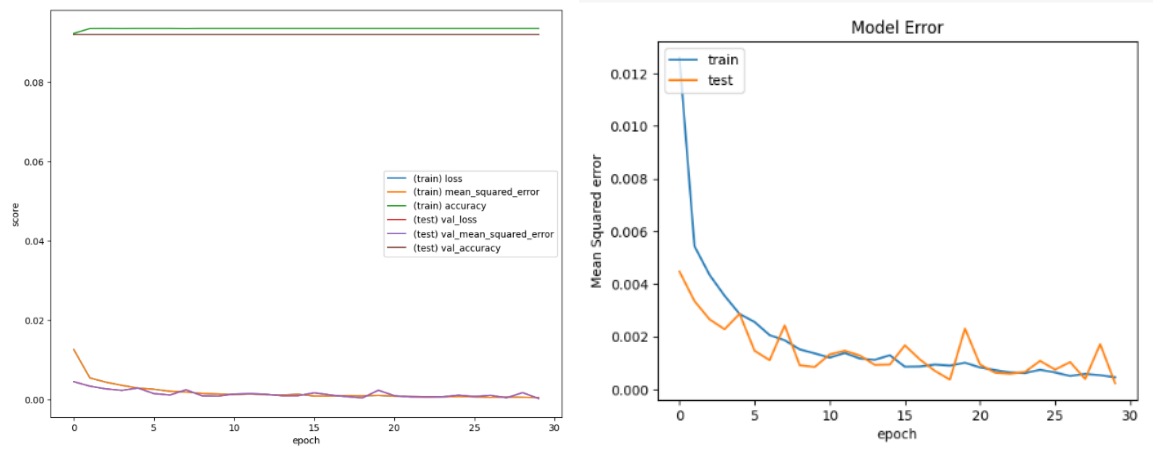
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 136: Métricas de entrenamiento Sujeto 3 Prueba 3 Modelo LSTM+FCN para tamaño de zancada



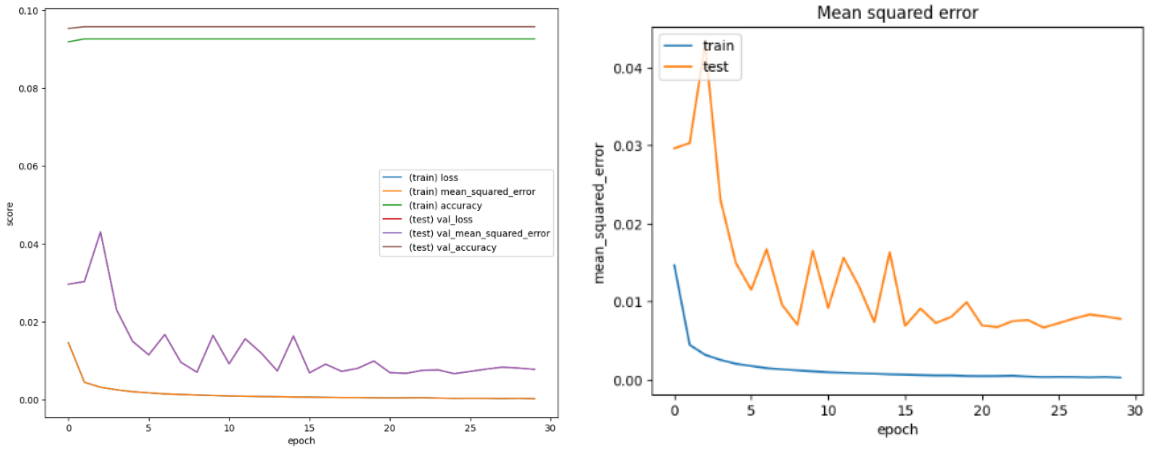
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 137: Métricas de entrenamiento Sujeto 9 Prueba 3 Modelo LSTM+FCN para tamaño de zancada



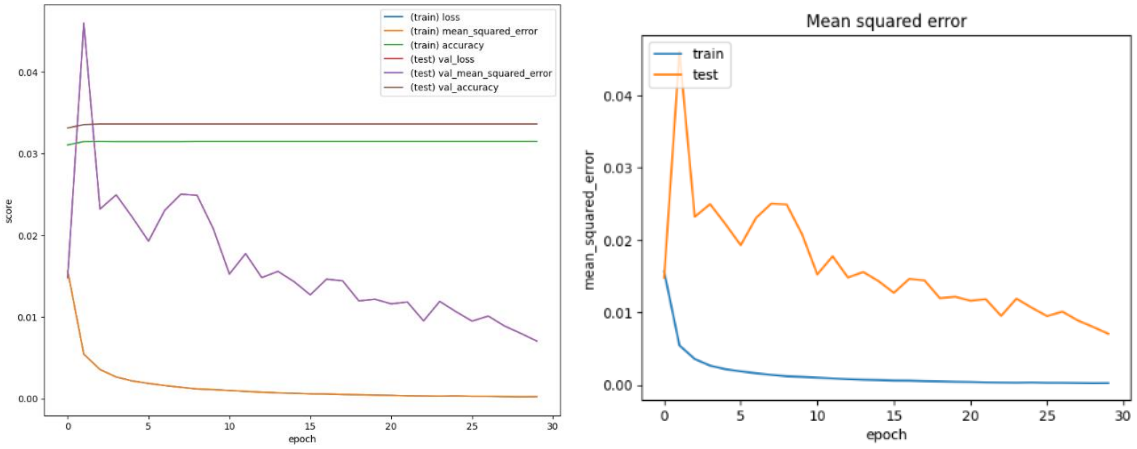
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 138: Métricas de entrenamiento Sujeto 9 Prueba 3 Modelo LSTM+CNN para tamaño de zancada



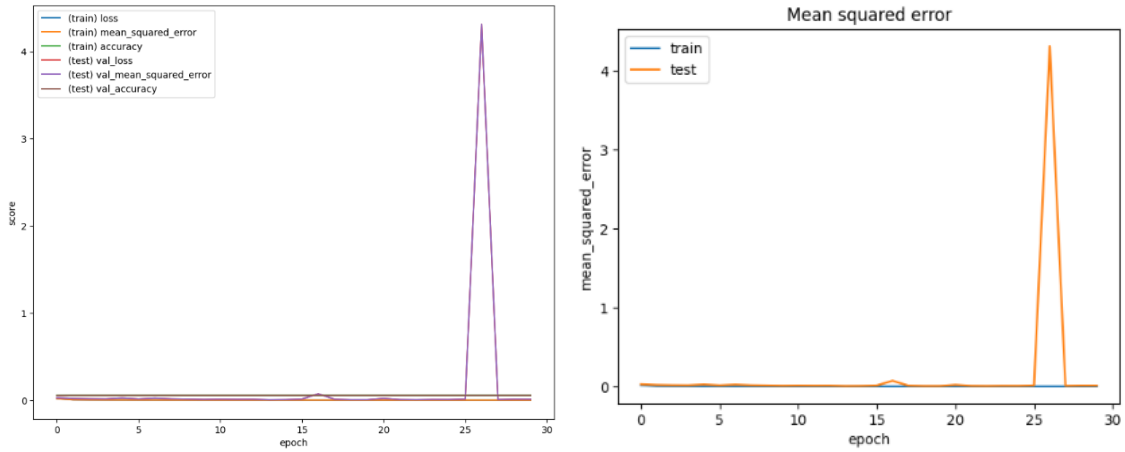
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 139: Métricas de entrenamiento Sujeto 3 Prueba 3 Modelo LSTM+CNN para tamaño de zancada



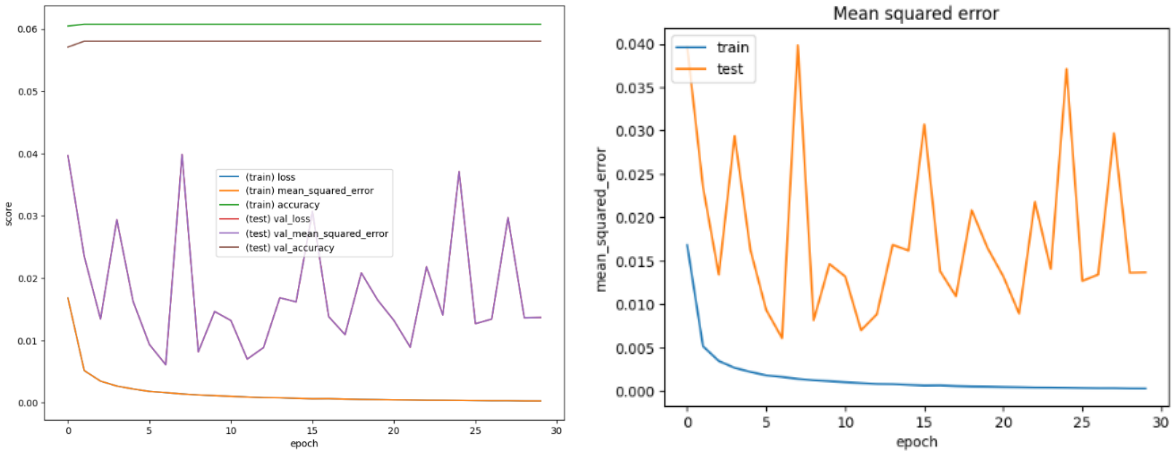
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 140: Métricas de entrenamiento Sujeto 5 Prueba 3 Modelo LSTM+CNN para tamaño de zancada



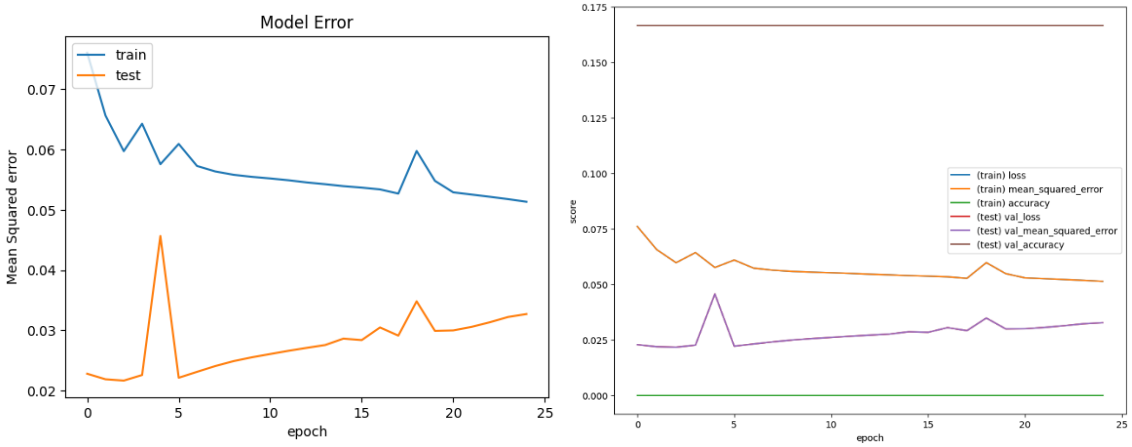
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 141: Métricas de entrenamiento Sujeto 7 Prueba 3 Modelo LSTM+CNN para tamaño de zancada



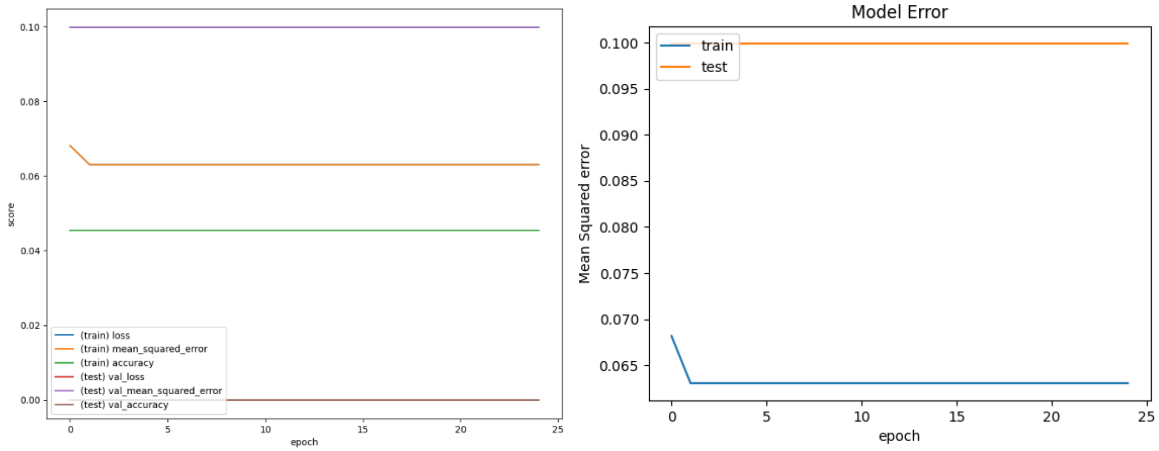
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 142: Métricas de entrenamiento Sujeto 9 del Modelo LSTM+FCN para número de pasos



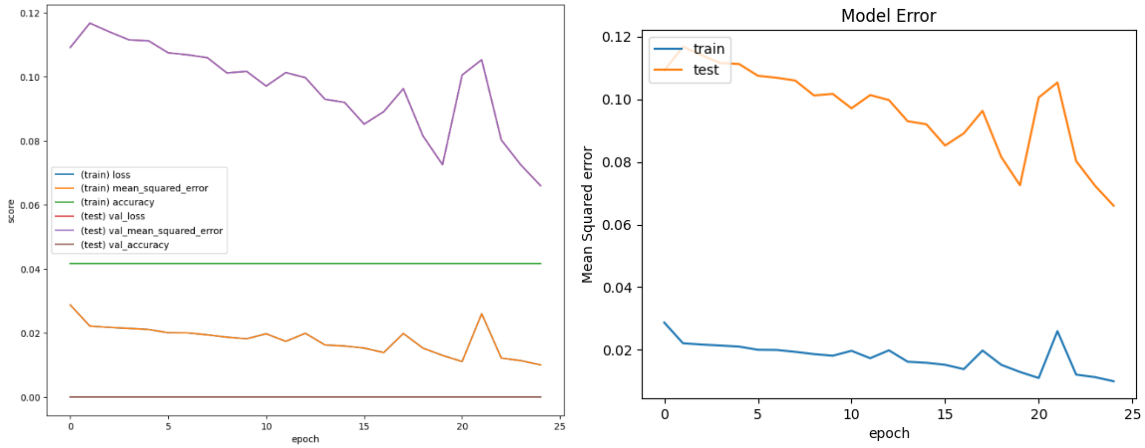
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 143: Métricas de entrenamiento Sujeto 2 del Modelo LSTM+FCN para número de pasos



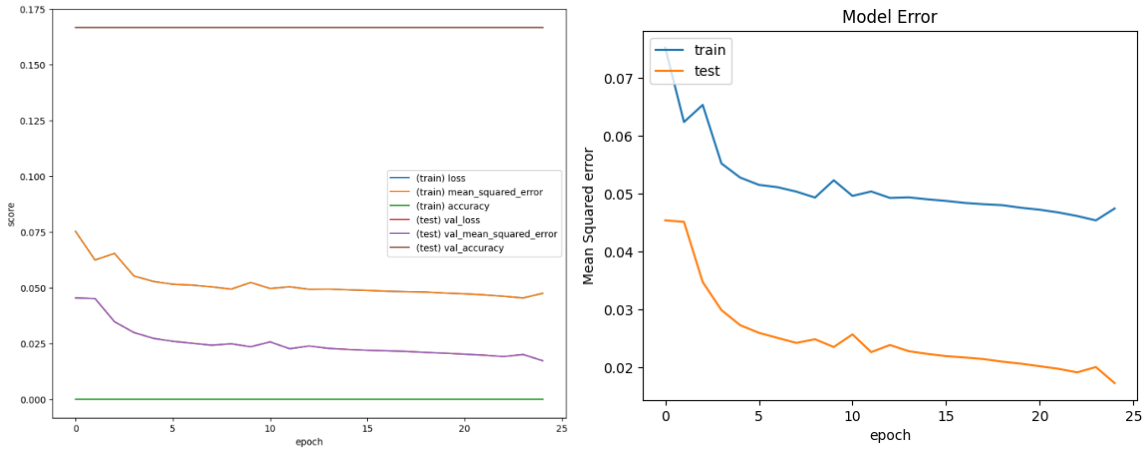
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 144: Métricas de entrenamiento Sujeto 4 del Modelo LSTM+FCN para número de pasos



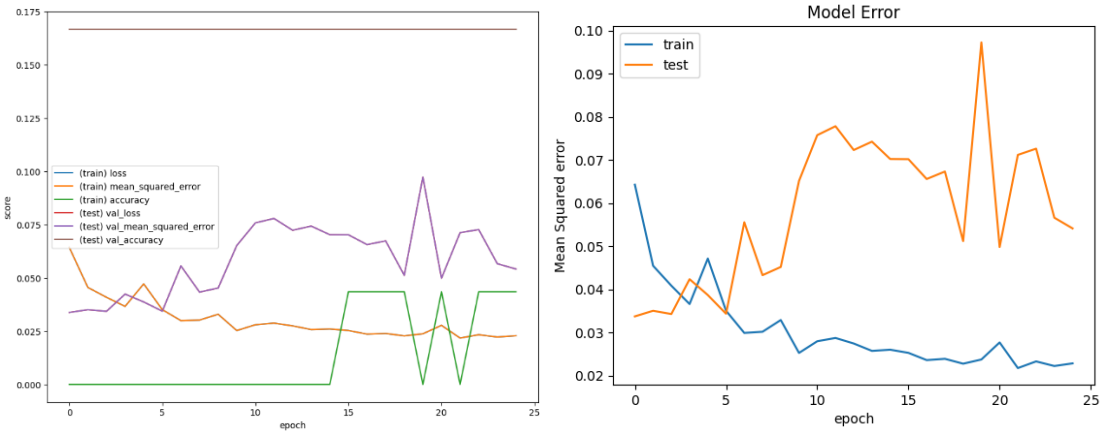
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 145: Métricas de entrenamiento Sujeto 5 del Modelo LSTM+FCN para número de pasos



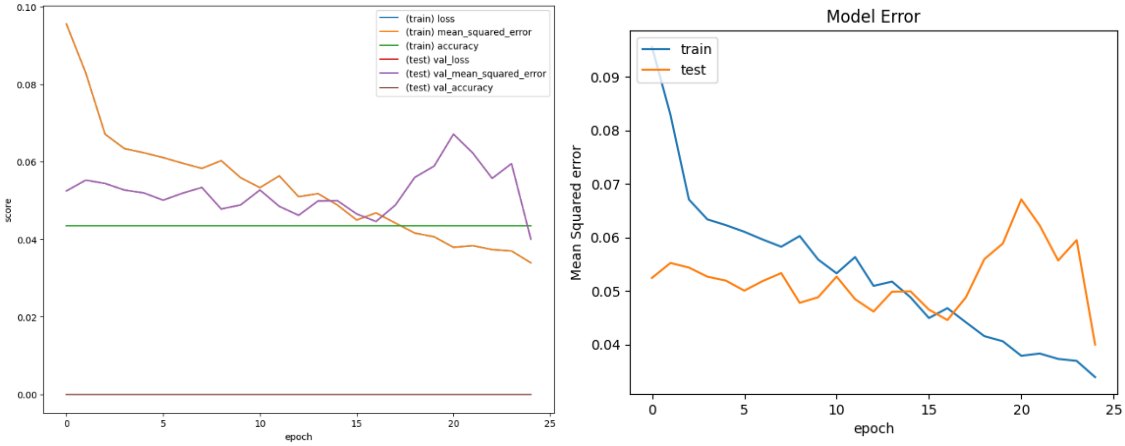
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 146: Métricas de entrenamiento Sujeto 6 del Modelo LSTM+FCN para número de pasos



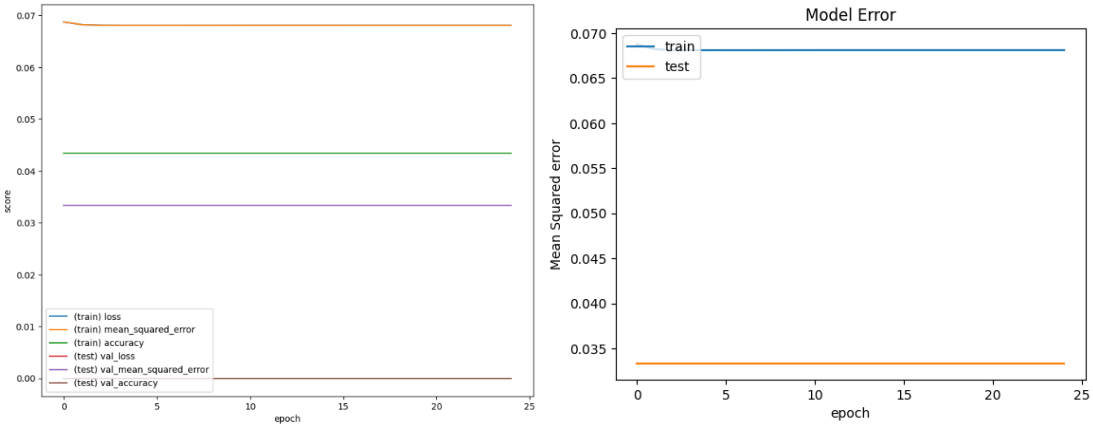
Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 147: Métricas de entrenamiento Sujeto 7 del Modelo LSTM+FCN para número de pasos



Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 148: Métricas de entrenamiento Sujeto 3 del Modelo LSTM+FCN para número de pasos



Fuente: Elaboración propia obtenida por código de Google Colab

Anexo 149: Ejemplo de código completo LSTM+FCN

```
# Data PreProcessing


_____________________________________________________________________
from google.colab import drive
drive.mount('/content/drive')



_____________________________________________________________________
#Importing packagesimport pandas as pd
import pandas as pd
import csv
import time
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
_____________________________________________________________________


_____________________________________________________________________
##Loading csv files
path = r'/content/drive/MyDrive/Bases de datos Fase 1 Jennifer y
Pilar/SDC/ASDC-S6.csv'

df= pd.read_csv(path, skiprows=1,
usecols=[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,2
6,27,28],
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_kn
ee','Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_
x_knee','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_h
ip','Gyro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip'
,'Magne_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip','Med_Stepleng
th'])
print (df.info)
print(df.shape)

#Standarizing
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler( feature_range = (0, 1) )
x = scaler.fit_transform(df)
x = pd.DataFrame(data = x, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','G
yro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee
','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','G
yro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magn
e_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip',' Med_Steplength
'])
```

```
scaler = MinMaxScaler( feature_range = (0, 1) )
df = scaler.fit_transform(df)
df = pd.DataFrame(data = df, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','G
yro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee
','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','G
yro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magn
e_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip',' Med_Steplength
'])

y = df['Real_Velocity']
y = pd.DataFrame(y)
print("y = \n",y)
print(y.shape)

print("x = \n",x)
print(x.shape)
```

#Verifying data shape

```
x.shape, y.shape



-1 in df[' Med_Steplength ']
```

Create train and test subdatasets

```
from sklearn.model_selection import train_test_split
import numpy as np
#X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 0, stratify = y)
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size =
0.2)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

Graphics

```
name = 'df'
columns_names =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','G
```

```
yro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee
','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','G
yro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magn
e_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip','Real_Velocity']
#plt.plot(df[name])
plt.plot(x)
plt.suptitle(name)
plt.legend(columns_names)
plt.show()

# plot the x, y, z acceleration and activities for multiple subjects
def plot_subject(subject):
     plt.figure()
     # create a plot for each column
     for col in range(subject.shape[0]):
          plt.subplot(subject.shape[0], 1, col+1)
          plt.plot(subject[:,col])
     plt.show()

# load
subjects = df
print('Loaded %d subjects' % len(subjects))
```

#Sliding Window Proccessing

```
print(df.head())
print(df.shape)
#df.loc[3,"y2"]
#df.loc[2:4]
#print(df.loc[2:4])

dataset_size = df.shape[0] ### how many rows
window_size = 100 ### how many time steps we want to process
simultaneously

xx = np.array([ x.loc[i:(i+window_size-1)].values for i in
range(dataset_size-window_size) ])
#print(xx[0:5]).loc
print("shape(xx) = ", xx.shape)

yy = np.array([ y.loc[i].values for i in range(dataset_size-
window_size) ])
print("shape(yy) = ", yy.shape)

#yy = np.array([ y.loc[i:(i+window_size-1)].values for i in
range(dataset_size-window_size) ])
#print(yy[0:5])
#print("shape(yy) = ", yy.shape)
#yy[2].shape
```

```
#yy.reshape(-1,2)
```

```
from sklearn.model_selection import train_test_split
import numpy as np
#X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 0, stratify = y)
X_train, X_test, y_train, y_test = train_test_split(xx, yy, test_size
= 0.2)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
print(X_train.shape[1])
print(X_train.shape[2])
print(y_train.shape[1])
print(np.unique(y_train).shape[0])
print(X_train.shape[0])
```

 # **LSTM + 3 DL Archicture**

Load Tensorflow libraries

```
import tensorflow as tf
import tensorflow.keras
import tensorflow.keras.backend as K
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, LSTM,
TimeDistributed
from tensorflow.keras.layers import Conv1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import sparse_categorical_crossentropy
import numpy as np
print(tf.version.VERSION)
print(tf.__version__)
```

```
num_features = 25
hidden_unit = 64
Batch_Size = 128
window_size = 100
epochs = 25
```

```
def create_lstm_model(X_train, y_train, window_size, num_features,
hidden_unit):
```

```
    inp1 = tf.keras.Input(shape=(window_size, num_features))


    first_LSTM_layer = tf.keras.layers.LSTM(hidden_unit,
name='lstm_1', return_sequences= False, return_state=True)
    first_layer, _, _ = first_LSTM_layer(inp1)

    Dense_layer_1 = tf.keras.layers.Dense(32, activation='relu')
    Dense_layer_2 = tf.keras.layers.Dense(16, activation='relu')
    Dense_layer_3 = tf.keras.layers.Dense(8, activation='relu')
    regressor_layer = tf.keras.layers.Dense(1, activation='relu')

    Dense_1_output =  Dense_layer_1(first_layer)
    Dense_2_output =  Dense_layer_2(Dense_1_output)
    Dense_3_output =  Dense_layer_3(Dense_2_output)
    Regressor =  regressor_layer(Dense_3_output)

    model = tf.keras.Model(inputs=inp1, outputs= Regressor)


    model.compile(optimizer='RMSprop', loss='mse',
metrics=['mean_squared_error', 'accuracy',])


    model.summary()

    ########Rebuilt Model########
    rebuilt_input = tf.keras.Input(shape=(1, num_features))
    ####'Build states input'######
    state_h_1_input = tf.keras.Input(shape=(hidden_unit,))
    state_c_1_input = tf.keras.Input(shape=(hidden_unit,))
    states_inputs_combination = [state_h_1_input, state_c_1_input]

    ###'first layer'
    rebuilt_first_layer, state_h_1, state_c_1 =
first_LSTM_layer(rebuilt_input,

initial_state=states_inputs_combination[:2])

    decoder_states = [state_h_1, state_c_1]

    D_1 = Dense_layer_1(rebuilt_first_layer)
    D_2 = Dense_layer_2(D_1)
    D_3 = Dense_layer_3(D_2)
    rebuilt_output = regressor_layer(D_3)

    rebuilt_model = tf.keras.Model([rebuilt_input] +
states_inputs_combination, [rebuilt_output] + decoder_states)
    rebuilt_model.summary()

    # Plotting model
```

```
    plot = tf.keras.utils.plot_model(model, to_file='model.png',
show_shapes=True, show_layer_names=True, rankdir='LR')

    return model, rebuilt_model, plot

# fit network
model, _, plot = create_lstm_model(X_train, y_train, window_size,
num_features, hidden_unit)
start_time = time.perf_counter()

result = model.fit(X_train, y_train, epochs=epochs,
batch_size=Batch_Size, verbose=1, shuffle=False,
validation_data=(X_test, y_test))

# evaluate model
end_time = time.perf_counter()
run_time = (end_time - start_time)
print( "time = " + str(int(run_time)) + " s = " +
str(int(run_time//60)) + " mins" )

plot
```

```
_________________________________________________________________
    Model: "model"
    _________________________________________________________________
     Layer (type)                 Output Shape              Param #
    =================================================================
     input_1 (InputLayer)         [(None, 100, 25)]         0

     lstm_1 (LSTM)                [(None, 64),              23040
                                   (None, 64),
                                   (None, 64)]

     dense (Dense)                (None, 32)                2080

     dense_1 (Dense)              (None, 16)                528

     dense_2 (Dense)              (None, 8)                 136

     dense_3 (Dense)              (None, 1)                 9

    =================================================================
    Total params: 25793 (100.75 KB)
    Trainable params: 25793 (100.75 KB)
    Non-trainable params: 0 (0.00 Byte)
    _________________________________________________________________
    Model: "model_1"

_________________________________________________________________
_________________________________
```

```
     Layer (type)                Output Shape                  Param #
Connected to

==================================================================
==============================
     input_2 (InputLayer)        [(None, 1, 25)]               0
[]

     input_3 (InputLayer)        [(None, 64)]                  0
[]

     input_4 (InputLayer)        [(None, 64)]                  0
[]

     lstm_1 (LSTM)               [(None, 64),                  23040
['input_2[0][0]',
                                 (None, 64),
'input_3[0][0]',
                                 (None, 64)]
'input_4[0][0]']

     dense (Dense)               (None, 32)                    2080
['lstm_1[1][0]']

     dense_1 (Dense)             (None, 16)                    528
['dense[1][0]']

     dense_2 (Dense)             (None, 8)                     136
['dense_1[1][0]']

     dense_3 (Dense)             (None, 1)                     9
['dense_2[1][0]']


==================================================================
==============================
    Total params: 25793 (100.75 KB)
    Trainable params: 25793 (100.75 KB)
    Non-trainable params: 0 (0.00 Byte)
    _________________________________________________________________
    Epoch 1/25
    370/370 [==============================] - 12s 14ms/step - loss:
0.0140 - mean_squared_error: 0.0140 - accuracy: 0.0583 - val_loss:
0.0085 - val_mean_squared_error: 0.0085 - val_accuracy: 0.0605
    Epoch 2/25
    370/370 [==============================] - 4s 10ms/step - loss:
0.0064 - mean_squared_error: 0.0064 - accuracy: 0.0588 - val_loss:
0.0048 - val_mean_squared_error: 0.0048 - val_accuracy: 0.0605
    Epoch 3/25
    370/370 [==============================] - 3s 9ms/step - loss:
0.0055 - mean_squared_error: 0.0055 - accuracy: 0.0588 - val_loss:
0.0055 - val_mean_squared_error: 0.0055 - val_accuracy: 0.0605
```

```
    Epoch 4/25
    370/370 [==============================] - 4s 10ms/step - loss:
0.0048 - mean_squared_error: 0.0048 - accuracy: 0.0588 - val_loss:
0.0061 - val_mean_squared_error: 0.0061 - val_accuracy: 0.0605
    Epoch 5/25
    370/370 [==============================] - 4s 11ms/step - loss:
0.0041 - mean_squared_error: 0.0041 - accuracy: 0.0588 - val_loss:
0.0048 - val_mean_squared_error: 0.0048 - val_accuracy: 0.0605
    Epoch 6/25
    370/370 [==============================] - 4s 10ms/step - loss:
0.0034 - mean_squared_error: 0.0034 - accuracy: 0.0588 - val_loss:
0.0052 - val_mean_squared_error: 0.0052 - val_accuracy: 0.0605
    Epoch 7/25
    370/370 [==============================] - 3s 9ms/step - loss:
0.0027 - mean_squared_error: 0.0027 - accuracy: 0.0588 - val_loss:
0.0051 - val_mean_squared_error: 0.0051 - val_accuracy: 0.0605
    Epoch 8/25
    370/370 [==============================] - 3s 9ms/step - loss:
0.0022 - mean_squared_error: 0.0022 - accuracy: 0.0588 - val_loss:
0.0014 - val_mean_squared_error: 0.0014 - val_accuracy: 0.0605
    Epoch 9/25
    370/370 [==============================] - 5s 12ms/step - loss:
0.0019 - mean_squared_error: 0.0019 - accuracy: 0.0588 - val_loss:
0.0016 - val_mean_squared_error: 0.0016 - val_accuracy: 0.0605
    Epoch 10/25
    370/370 [==============================] - 3s 9ms/step - loss:
0.0016 - mean_squared_error: 0.0016 - accuracy: 0.0588 - val_loss:
0.0013 - val_mean_squared_error: 0.0013 - val_accuracy: 0.0605
    Epoch 11/25
    370/370 [==============================] - 4s 12ms/step - loss:
0.0015 - mean_squared_error: 0.0015 - accuracy: 0.0588 - val_loss:
0.0011 - val_mean_squared_error: 0.0011 - val_accuracy: 0.0605
    Epoch 12/25
    370/370 [==============================] - 4s 11ms/step - loss:
0.0013 - mean_squared_error: 0.0013 - accuracy: 0.0588 - val_loss:
0.0011 - val_mean_squared_error: 0.0011 - val_accuracy: 0.0605
    Epoch 13/25
    370/370 [==============================] - 4s 11ms/step - loss:
0.0012 - mean_squared_error: 0.0012 - accuracy: 0.0588 - val_loss:
0.0021 - val_mean_squared_error: 0.0021 - val_accuracy: 0.0605
    Epoch 14/25
    370/370 [==============================] - 3s 9ms/step - loss:
0.0012 - mean_squared_error: 0.0012 - accuracy: 0.0588 - val_loss:
0.0027 - val_mean_squared_error: 0.0027 - val_accuracy: 0.0605
    Epoch 15/25
    370/370 [==============================] - 4s 10ms/step - loss:
0.0011 - mean_squared_error: 0.0011 - accuracy: 0.0588 - val_loss:
5.7880e-04 - val_mean_squared_error: 5.7880e-04 - val_accuracy: 0.0605
    Epoch 16/25
    370/370 [==============================] - 4s 11ms/step - loss:
0.0011 - mean_squared_error: 0.0011 - accuracy: 0.0588 - val_loss:
0.0014 - val_mean_squared_error: 0.0014 - val_accuracy: 0.0605
```

```
    Epoch 17/25
    370/370 [==============================] - 3s 9ms/step - loss:
9.4517e-04 - mean_squared_error: 9.4517e-04 - accuracy: 0.0588 -
val_loss: 6.5948e-04 - val_mean_squared_error: 6.5948e-04 -
val_accuracy: 0.0605
    Epoch 18/25
    370/370 [==============================] - 3s 9ms/step - loss:
9.2254e-04 - mean_squared_error: 9.2254e-04 - accuracy: 0.0588 -
val_loss: 6.1219e-04 - val_mean_squared_error: 6.1219e-04 -
val_accuracy: 0.0605
    Epoch 19/25
    370/370 [==============================] - 3s 9ms/step - loss:
8.7682e-04 - mean_squared_error: 8.7682e-04 - accuracy: 0.0588 -
val_loss: 0.0018 - val_mean_squared_error: 0.0018 - val_accuracy:
0.0605
    Epoch 20/25
    370/370 [==============================] - 4s 12ms/step - loss:
8.4183e-04 - mean_squared_error: 8.4183e-04 - accuracy: 0.0588 -
val_loss: 0.0010 - val_mean_squared_error: 0.0010 - val_accuracy:
0.0605
    Epoch 21/25
    370/370 [==============================] - 3s 9ms/step - loss:
8.1012e-04 - mean_squared_error: 8.1012e-04 - accuracy: 0.0588 -
val_loss: 5.6160e-04 - val_mean_squared_error: 5.6160e-04 -
val_accuracy: 0.0605
    Epoch 22/25
    370/370 [==============================] - 3s 9ms/step - loss:
7.6536e-04 - mean_squared_error: 7.6536e-04 - accuracy: 0.0588 -
val_loss: 0.0015 - val_mean_squared_error: 0.0015 - val_accuracy:
0.0605
    Epoch 23/25
    370/370 [==============================] - 3s 9ms/step - loss:
7.1422e-04 - mean_squared_error: 7.1422e-04 - accuracy: 0.0588 -
val_loss: 6.6600e-04 - val_mean_squared_error: 6.6600e-04 -
val_accuracy: 0.0605
    Epoch 24/25
    370/370 [==============================] - 5s 12ms/step - loss:
6.8856e-04 - mean_squared_error: 6.8856e-04 - accuracy: 0.0588 -
val_loss: 5.7372e-04 - val_mean_squared_error: 5.7372e-04 -
val_accuracy: 0.0605
    Epoch 25/25
    370/370 [==============================] - 4s 10ms/step - loss:
6.3581e-04 - mean_squared_error: 6.3581e-04 - accuracy: 0.0588 -
val_loss: 4.6692e-04 - val_mean_squared_error: 4.6692e-04 -
val_accuracy: 0.0605
    time = 145 s = 2 mins
```

```
datafolder = '/content/drive/MyDrive/IntentoCodigo'
import matplotlib.pyplot as plt
fig = plt.figure(figsize=[10,8])
ax = fig.add_subplot(111)
for key in result.history :
  #if key == "loss" or key == "val_loss": continue
  label = ("(test) " if key.startswith("val_") else "(train) ") + key
  plt.plot(result.history[key], label=label)
#ax.set(ylim=[-0.1, 1.1])no h
ax.set(xlabel="epoch", ylabel="score")
ax.legend()
plt.show()
fig.savefig(datafolder + "plot-train_test_loss_metrics_LSTM_DL.png")
```

```
# summarize history for accuracy
plt.plot(result.history['mean_squared_error'])
plt.plot(result.history['val_mean_squared_error'])
plt.title('Model Error')
plt.ylabel('Mean Squared error')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(result.history['loss'])
plt.plot(result.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Save and load network

```
### Save
dnn_filename = datafolder + "trained_network_LSTM_DLS6.h5"
model.save(dnn_filename)
print("Model saved to: " + dnn_filename)

### Load
model = tf.keras.models.load_model(datafolder +
"trained_network_LSTM_DLS6.h5")
model.summary()
```

```
    Model saved to:
/content/drive/MyDrive/IntentoCodigotrained_network_LSTM_DLS6.h5
    Model: "model"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 100, 25)]         0

 lstm_1 (LSTM)               [(None, 64),              23040
                              (None, 64),
                              (None, 64)]

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 8)                 136

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 25793 (100.75 KB)
Trainable params: 25793 (100.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Evaluate Model with new data (1 Subject)

```
_____________________________________________________________________
##Loading csv files

path = r'/content/drive/MyDrive/Bases de datos Fase 1 Jennifer y
Pilar/Copia de S6_10m_fast_velocity.CSV'

df= pd.read_csv(path, skiprows=1,
usecols=[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,2
6,27,28],
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_kn
ee','Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_
x_knee','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_h
ip','Gyro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip'
,'Magne_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip','
Med_Steplength '])
print (df.info)
print(df.shape)

#Standarizing
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler( feature_range = (0, 1) )
xtest = scaler.fit_transform(df)
xtest = pd.DataFrame(data = x, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','G
```

```
yro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee
','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','G
yro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magn
e_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip',' Med_Steplength'])

scaler = MinMaxScaler( feature_range = (0, 1) )
dftest = scaler.fit_transform(df)
dftest = pd.DataFrame(data = df, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','G
yro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee
','Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','G
yro_x_hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magn
e_z_hip','Euler_x_hip','Euler_y_hip','Euler_z_hip',' Med_Steplength
'])

ytest = df['Real_Velocity']
ytest = pd.DataFrame(y)
print("y = \n",y)
print(ytest.shape)

print("x = \n",x)
print(xtest.shape)

#Verifying data shape
xtest.shape, ytest.shape
_____
print(dftest.head())
print(dftest.shape)
#df.loc[3,"y2"]
#df.loc[2:4]
#print(df.loc[2:4])

dataset_size = dftest.shape[0] ### how many rows
window_size = 100 ### how many time steps we want to process
simultaneously

xxtest = np.array([ xtest.loc[i:(i+window_size-1)].values for i in
range(dataset_size-window_size) ])
#print(xx[0:5]).loc
print("shape(xx) = ", xxtest.shape)

yytest = np.array([ y.loc[i].values for i in range(dataset_size-
window_size) ])
print("shape(yy) = ", yytest.shape)

# Preprocess the new dataset

print(xxtest.shape)
print(yytest.shape)


# Evaluate the model on the new dataset
```

```
mse  = model.evaluate(xxtest, yytest, verbose=0)
print("Mean squared error on new dataset:", mse)
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics

model =
tf.keras.models.load_model("/content/drive/MyDrive/IntentoCodigo/Model
ostrained_network_LSTM_DLS6.h5")

# Supongamos que X_test es la lista de etiquetas reales y y_pred es la
lista de predicciones del modelo.
y_pred = model.predict(xxtest)

print(y_pred.shape)
print("Step Length Prediction", y_pred[0:1])
#filtered_data = np.where(y_pred < 0.2, 0, y_pred)
#mask = filtered_data != 0
#mediciones_filtradas = filtered_data[mask]

mean_prediction = np.mean(y_pred)
max = np.max(y_pred)
min = np.min(y_pred)
print("Max", max)
print("Min", min)

print("Mean of Predictions:", mean_prediction)

# Cálculo de métricas de regresión
mse = metrics.mean_squared_error(yytest, y_pred)
rmse = np.sqrt(mse)
mae = metrics.mean_absolute_error(yytest, y_pred)
r2 = metrics.r2_score(yytest, y_pred)


print("Mean Squared Error: {}".format(mse))
print("Mean Absolute Error: {}".format(mae))
print("Root Mean Squared Error: {}".format(rmse))
print("R-squared: {}".format(r2))
```

Anexo 150: Ejemplo código LSTM+CNN

```
# Data PreProcessing


_____________________________________________________________________
from google.colab import drive
drive.mount('/content/drive')


_____________________________________________________________________
#Importing packagesimport pandas as pd
import pandas as pd
import csv
import time
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler


_____________________________________________________________________
##Loading csv file
path = r'/content/drive/MyDrive/Bases de datos Fase 1 Jennifer y
Pilar/SDC/ASDC-S6.csv'

df= pd.read_csv(path, skiprows=1,
usecols=[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
,28],
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee',
'Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee',
'Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_
hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','
Euler_x_hip','Euler_y_hip','Euler_z_hip','Step_count'])
print (df.info)
print(df.shape)

#Standarizing
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler( feature_range = (0, 1) )
x = scaler.fit_transform(df)
x = pd.DataFrame(data = x, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','Gyro_
z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee','Euler
_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_hip','
Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','Euler_
x_hip','Euler_y_hip','Euler_z_hip','Step_count'])

scaler = MinMaxScaler( feature_range = (0, 1) )
df = scaler.fit_transform(df)
df = pd.DataFrame(data = df, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','Gyro_
z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee','Euler
_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_hip','
```

```
Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','Euler_
x_hip','Euler_y_hip','Euler_z_hip','Step_count'])

y = df['Step_count']
y = pd.DataFrame(y)
print("y = \n",y)
print(y.shape)

print("x = \n",x)
print(x.shape)

#Verifying data shape
x.shape, y.shape



-1 in df['Step_count']
```

Create train and test subdatasets

```
from sklearn.model_selection import train_test_split
import numpy as np
#X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 0, stratify = y)
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size =
0.2)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

Graphics

```
name = 'df'
columns_names =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','Gyro_
z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee','Euler
_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_hip','
Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','Euler_
x_hip','Euler_y_hip','Euler_z_hip','Step_count']
#plt.plot(df[name])
plt.plot(x)
plt.suptitle(name)
plt.legend(columns_names)
plt.show()
```

```
# plot the x, y, z acceleration and activities for multiple subjects
def plot_subject(subject):
      plt.figure()
      # create a plot for each column
      for col in range(subject.shape[0]):
            plt.subplot(subject.shape[0], 1, col+1)
            plt.plot(subject[:,col])
      plt.show()

# load
subjects = df
print('Loaded %d subjects' % len(subjects))
```

 # Proccessing into windows

```
print(df.head())
print(df.shape)

dataset_size = df.shape[0] ### how many rows
window_size = 100 ### how many time steps we want to process
simultaneously

xx = np.array([ x.loc[i:(i+window_size-1)].values for i in
range(dataset_size-window_size) ])
#print(xx[0:5]).loc
print("shape(xx) = ", xx.shape)

yy = np.array([ y.loc[i].values for i in range(dataset_size-window_size)
])
print("shape(yy) = ", yy.shape)
```

```
from sklearn.model_selection import train_test_split
import numpy as np
X_train, X_test, y_train, y_test = train_test_split(xx, yy, test_size =
0.2)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)

-1 in yy
```

```
print(X_train.shape[1])
print(X_train.shape[2])
print(y_train.shape[1])
print(np.unique(y_train).shape[0])
print(X_train.shape[0])
```

```
# **CNN-LSTM Archicture**

#### Load Tensorflow libraries


import tensorflow as tf
import tensorflow.keras
import tensorflow.keras.backend as K
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, LSTM,
TimeDistributed
from tensorflow.keras.layers import Conv1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.layers import BatchNormalization
print(tf.version.VERSION)
print(tf.__version__)

# Define model
verbose, epochs, batch_size = 1, 25, 128


n_timesteps, n_features, n_outputs = X_train.shape[1], X_train.shape[2], 1
# the Y train shape changed to one in order to have un valor continuo en
la salida
#n_timesteps, n_features, n_outputs = X_train.shape[1], X_train.shape[2],
np.unique(y_train).shape[0]
#n_timesteps, n_features, n_outputs = trainX.shape[1], trainX.shape[2],
trainy.shape[1]
#n_timesteps, n_features, n_outputs = 100, 6, 3

# reshape data into time steps of sub-sequences
n_steps, n_length = 10, 10
X_train= X_train.reshape((X_train.shape[0], n_steps, n_length,
n_features))
X_test = X_test.reshape((X_test.shape[0], n_steps, n_length, n_features))
print(X_train.shape[0])
print(X_test.shape[0])

# define model
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3,
activation='relu'), input_shape=(None,n_length,n_features)))
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3,
activation='relu')))
model.add(TimeDistributed(BatchNormalization()))
model.add(TimeDistributed(Dropout(0.5)))
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(100))
```

```
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='linear'))
model.compile(loss='mean_squared_error',
optimizer=Adam(learning_rate=0.001), metrics=['mean_squared_error',
'accuracy',])

print(n_outputs)

# Plotting model
plot = tf.keras.utils.plot_model(model, to_file='model.png',
show_shapes=True, show_layer_names=True, rankdir='LR')
print(model.summary())
plot
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDist  (None, None, 8, 64)       4864
 ributed)

 time_distributed_1 (TimeDi  (None, None, 6, 64)       12352
 stributed)

 time_distributed_2 (TimeDi  (None, None, 6, 64)       256
 stributed)

 time_distributed_3 (TimeDi  (None, None, 6, 64)       0
 stributed)

 time_distributed_4 (TimeDi  (None, None, 3, 64)       0
 stributed)

 time_distributed_5 (TimeDi  (None, None, 192)         0
 stributed)

 lstm (LSTM)                 (None, 100)               117200

 dropout_1 (Dropout)         (None, 100)               0

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 144873 (565.91 KB)
Trainable params: 144745 (565.41 KB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

```
epochs, batch_size = 25, 128
# fit network
```

```
start_time = time.perf_counter()

result = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
verbose=1, shuffle=False, validation_data=(X_test, y_test))

# evaluate model
end_time = time.perf_counter()
run_time = (end_time - start_time)
print( "time = " + str(int(run_time)) + " s = " + str(int(run_time//60)) +
" mins" )
```

```
    Epoch 1/25
    370/370 [==============================] - 20s 12ms/step - loss:
0.0234 - mean_squared_error: 0.0234 - accuracy: 0.0579 - val_loss: 0.0322
- val_mean_squared_error: 0.0322 - val_accuracy: 0.0627
    Epoch 2/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0058
- mean_squared_error: 0.0058 - accuracy: 0.0582 - val_loss: 0.0165 -
val_mean_squared_error: 0.0165 - val_accuracy: 0.0627
    Epoch 3/25
    370/370 [==============================] - 4s 10ms/step - loss: 0.0039
- mean_squared_error: 0.0039 - accuracy: 0.0582 - val_loss: 0.0139 -
val_mean_squared_error: 0.0139 - val_accuracy: 0.0627
    Epoch 4/25
    370/370 [==============================] - 4s 10ms/step - loss: 0.0030
- mean_squared_error: 0.0030 - accuracy: 0.0582 - val_loss: 0.0242 -
val_mean_squared_error: 0.0242 - val_accuracy: 0.0627
    Epoch 5/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0026
- mean_squared_error: 0.0026 - accuracy: 0.0582 - val_loss: 0.0305 -
val_mean_squared_error: 0.0305 - val_accuracy: 0.0627
    Epoch 6/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0022
- mean_squared_error: 0.0022 - accuracy: 0.0582 - val_loss: 0.0121 -
val_mean_squared_error: 0.0121 - val_accuracy: 0.0627
    Epoch 7/25
    370/370 [==============================] - 3s 9ms/step - loss: 0.0020
- mean_squared_error: 0.0020 - accuracy: 0.0582 - val_loss: 0.0180 -
val_mean_squared_error: 0.0180 - val_accuracy: 0.0627
    Epoch 8/25
    370/370 [==============================] - 4s 10ms/step - loss: 0.0017
- mean_squared_error: 0.0017 - accuracy: 0.0582 - val_loss: 0.0117 -
val_mean_squared_error: 0.0117 - val_accuracy: 0.0627
    Epoch 9/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0015
- mean_squared_error: 0.0015 - accuracy: 0.0582 - val_loss: 0.0145 -
val_mean_squared_error: 0.0145 - val_accuracy: 0.0627
    Epoch 10/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0014
- mean_squared_error: 0.0014 - accuracy: 0.0582 - val_loss: 0.0148 -
val_mean_squared_error: 0.0148 - val_accuracy: 0.0627
    Epoch 11/25
```

```
    370/370 [==============================] - 4s 10ms/step - loss: 0.0013
- mean_squared_error: 0.0013 - accuracy: 0.0582 - val_loss: 0.0146 -
val_mean_squared_error: 0.0146 - val_accuracy: 0.0627
    Epoch 12/25
    370/370 [==============================] - 4s 10ms/step - loss: 0.0012
- mean_squared_error: 0.0012 - accuracy: 0.0582 - val_loss: 0.0173 -
val_mean_squared_error: 0.0173 - val_accuracy: 0.0627
    Epoch 13/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0011
- mean_squared_error: 0.0011 - accuracy: 0.0582 - val_loss: 0.0108 -
val_mean_squared_error: 0.0108 - val_accuracy: 0.0627
    Epoch 14/25
    370/370 [==============================] - 3s 8ms/step - loss: 0.0010
- mean_squared_error: 0.0010 - accuracy: 0.0582 - val_loss: 0.0142 -
val_mean_squared_error: 0.0142 - val_accuracy: 0.0627
    Epoch 15/25
    370/370 [==============================] - 3s 9ms/step - loss:
9.1060e-04 - mean_squared_error: 9.1060e-04 - accuracy: 0.0582 - val_loss:
0.0169 - val_mean_squared_error: 0.0169 - val_accuracy: 0.0627
    Epoch 16/25
    370/370 [==============================] - 4s 10ms/step - loss:
8.4972e-04 - mean_squared_error: 8.4972e-04 - accuracy: 0.0582 - val_loss:
0.0106 - val_mean_squared_error: 0.0106 - val_accuracy: 0.0627
    Epoch 17/25
    370/370 [==============================] - 3s 8ms/step - loss:
8.2167e-04 - mean_squared_error: 8.2167e-04 - accuracy: 0.0582 - val_loss:
0.0183 - val_mean_squared_error: 0.0183 - val_accuracy: 0.0627
    Epoch 18/25
    370/370 [==============================] - 3s 8ms/step - loss:
7.3188e-04 - mean_squared_error: 7.3188e-04 - accuracy: 0.0582 - val_loss:
0.0190 - val_mean_squared_error: 0.0190 - val_accuracy: 0.0627
    Epoch 19/25
    370/370 [==============================] - 4s 10ms/step - loss:
6.8186e-04 - mean_squared_error: 6.8186e-04 - accuracy: 0.0582 - val_loss:
0.0129 - val_mean_squared_error: 0.0129 - val_accuracy: 0.0627
    Epoch 20/25
    370/370 [==============================] - 4s 11ms/step - loss:
6.8482e-04 - mean_squared_error: 6.8482e-04 - accuracy: 0.0582 - val_loss:
0.0176 - val_mean_squared_error: 0.0176 - val_accuracy: 0.0627
    Epoch 21/25
    370/370 [==============================] - 3s 8ms/step - loss:
6.2655e-04 - mean_squared_error: 6.2655e-04 - accuracy: 0.0582 - val_loss:
0.0111 - val_mean_squared_error: 0.0111 - val_accuracy: 0.0627
    Epoch 22/25
    370/370 [==============================] - 3s 8ms/step - loss:
5.6743e-04 - mean_squared_error: 5.6743e-04 - accuracy: 0.0582 - val_loss:
0.0110 - val_mean_squared_error: 0.0110 - val_accuracy: 0.0627
    Epoch 23/25
    370/370 [==============================] - 3s 8ms/step - loss:
5.0258e-04 - mean_squared_error: 5.0258e-04 - accuracy: 0.0582 - val_loss:
0.0120 - val_mean_squared_error: 0.0120 - val_accuracy: 0.0627
    Epoch 24/25
```

```
    370/370 [==============================] - 4s 12ms/step - loss:
4.8326e-04 - mean_squared_error: 4.8326e-04 - accuracy: 0.0582 - val_loss:
0.0116 - val_mean_squared_error: 0.0116 - val_accuracy: 0.0627
    Epoch 25/25
    370/370 [==============================] - 3s 8ms/step - loss:
4.5226e-04 - mean_squared_error: 4.5226e-04 - accuracy: 0.0582 - val_loss:
0.0134 - val_mean_squared_error: 0.0134 - val_accuracy: 0.0627
    time = 147 s = 2 mins
```

```
datafolder = '/content/drive/MyDrive/IntentoCodigo'
import matplotlib.pyplot as plt
fig = plt.figure(figsize=[10,8])
ax = fig.add_subplot(111)
for key in result.history :
  #if key == "loss" or key == "val_loss": continue
  label = ("(test) " if key.startswith("val_") else "(train) ") + key
  plt.plot(result.history[key], label=label)
#ax.set(ylim=[-0.1, 1.1])no h
ax.set(xlabel="epoch", ylabel="score")
ax.legend()
plt.show()
fig.savefig(datafolder + "plot-train_test_loss_metrics_1_CNN_LSTM.png")
```

```
# summarize history for mean_squared_error
plt.plot(result.history['mean_squared_error'])
plt.plot(result.history['val_mean_squared_error'])
plt.title('Mean squared error')
plt.ylabel('mean_squared_error')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(result.history['loss'])
plt.plot(result.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Save and load network

```
### Save
dnn_filename = datafolder + "trained_CNN&LSTM_Average.h5"
model.save(dnn_filename)
print("Model saved to: " + dnn_filename)
```

Load

```
model = tf.keras.models.load_model(datafolder +
"trained_CNN&LSTM_Average.h5")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDist  (None, None, 8, 64)       4864
 ributed)

 time_distributed_1 (TimeDi  (None, None, 6, 64)       12352
 stributed)

 time_distributed_2 (TimeDi  (None, None, 6, 64)       256
 stributed)

 time_distributed_3 (TimeDi  (None, None, 6, 64)       0
 stributed)

 time_distributed_4 (TimeDi  (None, None, 3, 64)       0
 stributed)

 time_distributed_5 (TimeDi  (None, None, 192)         0
 stributed)

 lstm (LSTM)                 (None, 100)               117200

 dropout_1 (Dropout)         (None, 100)               0

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 144873 (565.91 KB)
Trainable params: 144745 (565.41 KB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

Evaluate Model

****1 SUBJECT****

```
_____________________________________________________________
# Cargar el modelo entrenado
#trained_model =
tf.keras.models.load_model("/content/drive/MyDrive/IntentoCodigo/trained_C
NN&LSTM_Average.h5")
trained_model = model
```

```
path = r'/content/drive/MyDrive/Bases de datos Fase 1 Jennifer y
Pilar/Copia de S6_10m_fast_velocity.CSV'

df= pd.read_csv(path, skiprows=1,
usecols=[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
,28],
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee',
'Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee',
'Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_
hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','
Euler_x_hip','Euler_y_hip','Euler_z_hip','Step_count'])
print (df.info)
print(df.shape)

#Standarizing
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler( feature_range = (0, 1) )
xtest = scaler.fit_transform(df)
xtest = pd.DataFrame(data = x, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','Gyro_
z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee','Euler
_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_hip','
Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','Euler_
x_hip','Euler_y_hip','Euler_z_hip','Step_count'])

scaler = MinMaxScaler( feature_range = (0, 1) )
dftest = scaler.fit_transform(df)
dftest = pd.DataFrame(data = df, columns =
['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee','Gyro_
z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee','Euler
_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_hip','
Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','Euler_
x_hip','Euler_y_hip','Euler_z_hip','Step_count'])

ytest = df['Step_count']
ytest = pd.DataFrame(y)
print("y = \n",y)
print(ytest.shape)

print("x = \n",x)
print(xtest.shape)

#Verifying data shape
xtest.shape, ytest.shape


_____________________________________________________________________
print(dftest.head())
print(dftest.shape)
#df.loc[3,"y2"]
#df.loc[2:4]
#print(df.loc[2:4])
```

```
dataset_size = dftest.shape[0] ### how many rows
window_size = 100 ### how many time steps we want to process
simultaneously

xxtest = np.array([ xtest.loc[i:(i+window_size-1)].values for i in
range(dataset_size-window_size) ])
#print(xx[0:5]).loc
print("shape(xx) = ", xxtest.shape)

yytest = np.array([ y.loc[i].values for i in range(dataset_size-
window_size) ])
print("shape(yy) = ", yytest.shape)

n_features=25

# Preprocess the new dataset

xxtest = xxtest.reshape((xxtest.shape[0], n_steps, n_length, n_features))

print(xxtest.shape)
print(yytest.shape)


# Evaluate the model on the new dataset
mse  = model.evaluate(xxtest, yytest, verbose=0)
print("Mean squared error on new dataset:", mse)

from sklearn.metrics import mean_squared_error, mean_absolute_error

# Obtain predicted values for the test set
y_pred = model.predict(X_test)

print(y_pred[0:1])

mean_prediction = np.mean(y_pred)
max = np.max(y_pred)
min = np.min(y_pred)
print("Max", max)
print("Min", min)

print("Mean of Predictions:", mean_prediction)


#Compute the mean squared error and mean absolute error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)

 #Print the results
print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
print("Root Mean Squared Error: {}".format(rmse))
```

Anexo 151: Reunión con directivos de UNCIH

Fuente: Elaboración propia

Anexo 152: Actividad deportiva realizada por la UNCIH

Fuente: Elaboración propia

Anexo 153: Código de Secuencia a uno realizado para el modelo LSTM+FCN

```
Data Pre processing
from google.colab import drive
drive.mount('/content/drive')
#Importing packages
import pandas as pd
import csv
import time
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

# Loading csv file
path = r'/content/drive/MyDrive/Data Base/Distance Parameter/S7_D.csv'

df = pd.read_csv(path, skiprows=1, usecols=list(range(4, 29)),
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee',
'Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee',
'Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_
hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','
Euler_x_hip','Euler_y_hip','Euler_z_hip','Real_Velocity'])

# Standardizing
scaler = MinMaxScaler(feature_range=(0, 1))
df_scaled = scaler.fit_transform(df)

# Prepare sequences and labels
X_sequences_list = []
y_label_list = []

start_idx = 0
for idx in range(1, len(df)):
    # Check if it's the end of an experiment
    if df['Real_Velocity'].iloc[idx] != df['Real_Velocity'].iloc[idx-1]:
        end_idx = idx
        X_sequences_list.append(df_scaled[start_idx:end_idx])
        y_label_list.append(df_scaled[end_idx, -1])
        start_idx = idx + 1

# Convert lists to arrays if needed (depends on your subsequent processes)
X_sequences = np.array(X_sequences_list)
y_labels = np.array(y_label_list)

print("X_sequences shape:", X_sequences.shape)
print("y_label:", y_labels.shape)
from sklearn.model_selection import train_test_split

from keras.preprocessing.sequence import pad_sequences
```

```
# Assuming L is the sequence length you want
L = 1000  # or whatever sequence length you decide upon
num_features = 25  # Assuming each element in a sequence is of length 25

X_train_padded = pad_sequences(X_sequences, maxlen=L, padding='post',
truncating='post', dtype='float32')

# Reshaping to add the feature dimension
X_train_padded = X_train_padded.reshape((-1, L, num_features))

print("Padded X_train shape:", X_train_padded.shape)

# Split the sequences and labels into training and test sets
X_train_list, X_test_list, y_train, y_test =
train_test_split(X_train_padded, y_labels, test_size=0.2)

print("Train sequences:", len(X_train_list),len(y_train))
print("Test sequences:", len(X_test_list),len(y_test))
Graphics
import matplotlib.pyplot as plt

def plot_features(data):
    # Create a new figure with a defined size
    plt.figure(figsize=(10, 15))

    # Loop through each column in data
    for i, column in enumerate(data.columns, 1):
        plt.subplot(len(data.columns), 1, i)
        plt.plot(data[column])
        plt.title(column)

    # Adjust layout for better visualization
    plt.tight_layout()
    plt.show()

# Call the function
plot_features(df)

import tensorflow as tf
import tensorflow.keras
import tensorflow.keras.backend as K
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, LSTM,
TimeDistributed
from tensorflow.keras.layers import Conv1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import sparse_categorical_crossentropy
import numpy as np
print(tf.version.VERSION)
print(tf.__version__)

num_features = 25
hidden_unit = 64
Batch_Size = 8
```

```
window_size = 1000
epochs = 25

#Define model

def create_lstm_model(window_size, num_features, hidden_unit):
    inp = tf.keras.Input(shape=(window_size, num_features))

    # LSTM layer
    x = tf.keras.layers.LSTM(hidden_unit, name='lstm_1',
return_sequences=False)(inp)

    # Dense layers
    x = tf.keras.layers.Dense(32, activation='relu')(x)
    x = tf.keras.layers.Dense(16, activation='relu')(x)
    x = tf.keras.layers.Dense(8, activation='relu')(x)

    # Output layer
    output = tf.keras.layers.Dense(1, activation='relu')(x)

    model = tf.keras.Model(inputs=inp, outputs=output)

    # Compile the model
    model.compile(optimizer='RMSprop', loss='mse',
metrics=['mean_squared_error', 'accuracy'])

    model.summary()

    # Plotting model
    plot = tf.keras.utils.plot_model(model, to_file='model.png',
show_shapes=True, show_layer_names=True, rankdir='LR')

    return model, plot

# Fit and evaluate the network
model, plot = create_lstm_model(window_size, num_features, hidden_unit)
start_time = time.perf_counter()
result = model.fit(X_train_list, y_train, epochs=epochs,
batch_size=Batch_Size, verbose=1, shuffle=False,
validation_data=(X_test_list, y_test))
end_time = time.perf_counter()
run_time = (end_time - start_time)
print("time = " + str(int(run_time)) + " s = " + str(int(run_time // 60))
+ " mins")
plot
# datafolder = '/content/drive/MyDrive/IntentoCodigo'
import matplotlib.pyplot as plt
fig = plt.figure(figsize=[10,8])
ax = fig.add_subplot(111)
for key in result.history :
 #if key == "loss" or key == "val_loss": continue
  label = ("(test) " if key.startswith("val_") else "(train) ") + key
  plt.plot(result.history[key], label=label)
#ax.set(ylim=[-0.1, 1.1])no h
```

```
ax.set(xlabel="epoch", ylabel="score")
ax.legend()
plt.show()
# fig.savefig(datafolder + "plot-train_test_loss_metrics_LSTM_DL.png")
# summarize history for accuracy
plt.plot(result.history['mean_squared_error'])
plt.plot(result.history['val_mean_squared_error'])
plt.title('Model Error')
plt.ylabel('Mean Squared error')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(result.history['loss'])
plt.plot(result.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

##Loading csv files

```
path = r'/content/drive/MyDrive/Data Base/Subjects/S7/
S7_106m_outdoor_velocity.CSV'

experiment_df = pd.read_csv(path, skiprows=1, usecols=list(range(4, 29)),
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee',
'Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee',
'Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_
hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','
Euler_x_hip','Euler_y_hip','Euler_z_hip','Real_Velocity'])
# Let's say you have loaded your new experiment data into a DataFrame
called "experiment_df"
```

 # 1. Preprocess the new experiment data

```
experiment_scaled = scaler.transform(experiment_df)
experiment_sequences_list = []

start_idx = 0
for idx in range(1, len(experiment_df)):
    # Check if it's the end of an experiment
    if experiment_df['Real_Velocity'].iloc[idx] !=
experiment_df['Real_Velocity'].iloc[idx-1]:
        end_idx = idx

experiment_sequences_list.append(experiment_scaled[start_idx:end_idx])
        start_idx = idx + 1
```

 # Convert lists to arrays

```
experiment_sequences = np.array(experiment_sequences_list)
experiment_padded = pad_sequences(experiment_sequences, maxlen=L,
padding='post', truncating='post', dtype='float32')
experiment_padded = experiment_padded.reshape((-1, L, num_features))
```

```
# 2. Predict with the model
predictions = model.predict(experiment_padded)

# Now, "predictions" will hold the predicted values for your experiment.
print(predictions)


# 1. Create a dummy array with the shape of your original experiment data
# but filled with zeros or any arbitrary values
dummy_array = np.zeros((len(predictions), df_scaled.shape[1]))

# 2. Replace the column corresponding to `Real_Velocity` with your
predictions
dummy_array[:, -1] = predictions.flatten()

# 3. Use the `inverse_transform` method to revert the scaling
inverse_transformed_data = scaler.inverse_transform(dummy_array)

# 4. Extract the inverse transformed predictions for `Real_Velocity`
original_predictions = inverse_transformed_data[:, -1]

print("Predictions", original_predictions)

# from sklearn.metrics import mean_squared_error
# from sklearn.metrics import mean_absolute_error
# import numpy as np

# path = r'/content/drive/MyDrive/Bases de datos Fase 1 Jennifer y
Pilar/S2_30m_regular_velocity_outdoor_RealValue.csv'
# df = pd.read_csv(path, skiprows=1, usecols=list(range(4, 29)),
names=['Acc_x_knee','Acc_y_knee','Acc_z_knee','Gyro_x_knee','Gyro_y_knee',
'Gyro_z_knee','Magne_x_knee','Magne_y_knee','Magne_z_knee','Euler_x_knee',
'Euler_y_knee','Euler_z_knee','Acc_x_hip','Acc_y_hip','Acc_z_hip','Gyro_x_
hip','Gyro_y_hip','Gyro_z_hip','Magne_x_hip','Magne_y_hip','Magne_z_hip','
Euler_x_hip','Euler_y_hip','Euler_z_hip','Real_Velocity'])

# original = df['Real_Velocity']

# mask = ~np.isnan(original_predictions)

# original = original[:len(original_predictions)]

# original_filtered = original[mask]
# original_predictions_filtered = original_predictions[mask]

# # 2. Compute the MSE
# mse = mean_squared_error(original_filtered,
original_predictions_filtered)

# # Compute the MAE
# mae = mean_absolute_error(original_filtered,
original_predictions_filtered)
# print("Mean Absolute Error:", mae)
```

```
# # Compute the RMSE
# rmse = np.sqrt(mse)
# print("Root Mean Squared Error:", rmse)

# print("Mean Squared Error:", mse)
```

Fuente: Elaboración propia

XI. REFERENCIAS

1. Acevedo, E., Serna, A, & Serna, E. (2017). *Principios y características de las redes neuronales artificiales.* (Vol. 173).
2. Adamowicz, L., Christakis, Y., Czech, M. D., & Adamusiak, T. (2022). SciKit Digital Health: Python Package for Streamlined Wearable Inertial Sensor Data Processing. *JMIR mHealth and uHealth*, *10*(4), e36762. https://doi.org/10.2196/36762
3. Alemán-Ramírez, C. (2020). Marcha en personas con discapacidad visual: Revisión de literatura. *MHSalud: Revista en Ciencias del Movimiento Humano y Salud*, *17*(1), Article 1. https://doi.org/10.15359/mhs.17-1.5
4. Ali & Muhammad. (2019). *An Intuitive Guide of Artificial Neural Network with Practical Implementation in Keras & Tensorflow*.
5. Amor Perea, J. (2018). *Técnicas de ajuste de los métodos Kernel para la regresión.* Universidad de Sevilla.
6. Andrenacci, I., Boccaccini, R., Bolzoni, A., Colavolpe, G., Costantino, C., Federico, M., Ugolini, A., & Vannucci, A. (2021). A Comparative Evaluation of Inertial Sensors for Gait and Jump Analysis. *Sensors (Basel, Switzerland)*, *21*(18), 5990. https://doi.org/10.3390/s21185990
7. Arana, C. (s. f.). *Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales*.
8. Baqersad, J., Poozesh, P., Niezrecki, C., & Avitabile, P. (2017). Photogrammetry and optical methods in structural dynamics – A review. *Mechanical Systems and Signal Processing*, *86*, 17-34. https://doi.org/10.1016/j.ymssp.2016.02.011

9. Brownlee, J. (2018). *Better deep learning: Train faster, reduce overfitting, and make better predictions* (Machine Learning Mastery.).
10. Buisson Garcia, S. (2023). *SEGMENTACIÓN DE LAS FASES DE LA MARCHA HUMANA Y PREDICCIÓN DE SUS EVENTOS FUTUROS MEDIANTE SENSORES INERCIALES Y TÉCNICAS DE APRENDIZAJE MÁQUINA*. https://eciencia.urjc.es/handle/10115/22319
11. Caldas R., Fadel, T, & Buarque, F. (2020). Adaptive predictive systems applied to gait analysis: A systematic review. *Gait & Posture*, *77*, 75-82. https://doi.org/10.1016/j.gaitpost.2020.01.021
12. Cappagli, G., Finocchietti, S., Cocchi, E., Giammari, G., Zumiani, R., Cuppone, A. V., Baud-Bovy, G., & Gori, M. (2019). Audio motor training improves mobility and spatial cognition in visually impaired children. *Scientific Reports*, *9*, 3303. https://doi.org/10.1038/s41598-019-39981-x
13. Chollet, F. (2021). *Deep Learning with Python, Second Edition*. Simon and Schuster.
14. Cicirelli, G., Impedovo, D., Dentamaro, V., Marani, R., Pirlo, G., & D'Orazio, T. R. (2022). Human Gait Analysis in Neurodegenerative Diseases: A Review. *IEEE Journal of Biomedical and Health Informatics*, *26*(1), 229-242. https://doi.org/10.1109/JBHI.2021.3092875
15. C.M. Saliba, A.L. Clouthier, & S.C. Brandon. (2018). Prediction of knee joint contact forces from external measures using principal component prediction and reconstruction. *National library of medicine*, 1-27. https://doi.org/10.1123/jab.2017-0262

16. Dehghani, A., Sarbishei, O., Glatard, T., & Shihab, E. (2019). A Quantitative Comparison of Overlapping and Non-Overlapping Sliding Windows for Human Activity Recognition Using Inertial Sensors. *Sensors*, *19*(22), Article 22. https://doi.org/10.3390/s19225026
17. Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., & Dehmer, M. (2020). An Introductory Review of Deep Learning for Prediction Models With Big Data. *Frontiers in Artificial Intelligence*, *3*. https://www.frontiersin.org/articles/10.3389/frai.2020.00004
18. Eunice Amador Rosa, M., Lozano Bustillo, A., Espinoza Salvadó, I., Sierra, M., & Rivera, B. (2022). Prevalence and Causes of Avoidable Blindness in Subjects Over 50 Years of Age in Honduras. *Journal of Ophthalmic & Vision Research*, *17*(2), 225-232. https://doi.org/10.18502/jovr.v17i2.10794
19. Flores, G., & Manduchi, R. (2018). WeAllWalk: An Annotated Dataset of Inertial Sensor Time Series from Blind Walkers. *ACM Transactions on Accessible Computing*, *11*, 1-28. https://doi.org/10.1145/3161711
20. *Inertial Sensors and Their Applications | SpringerLink*. (s. f.). Recuperado 23 de julio de 2023, de https://link.springer.com/chapter/10.1007/978-3-319-91734-4_2#:~:text=Inertial%20sensors%20are%20well%20known,used%20to%20update%20position%20estimates.
21. Jaén-Vargas, M., Reyes Leiva, K. M., Fernandes, F., Barroso Gonçalves, S., Tavares Silva, M., Lopes, D. S., & Serrano Olmedo, J. J. (2022). Effects of sliding window variation in the performance of acceleration-based human activity recognition using deep learning models. *PeerJ Computer Science*, *8*, e1052. https://doi.org/10.7717/peerj-cs.1052

22. Khan, S. S., & Abedi, A. (2022). *Step Counting with Attention-based LSTM* (arXiv:2211.13114). arXiv. https://doi.org/10.48550/arXiv.2211.13114

23. *Machine Learning-Based Zero-Velocity Detection for Inertial Pedestrian Navigation | IEEE Journals & Magazine | IEEE Xplore*. (s. f.). Recuperado 23 de julio de 2023, de https://ieeexplore.ieee.org/document/9107252

24. Matich. (2001). *Redes Neuronales: Conceptos Básicos y Aplicaciones.* (Vol. 41, pp. 12-16). Universidad Tecnológica Nacional.

25. Michelucci, U. (2022). *An Introduction to Autoencoders* (arXiv:2201.03898). arXiv. https://doi.org/10.48550/arXiv.2201.03898

26. Mohri, Afshin Rostmizadeh, & Ameet Talwalkar. (2018). *Foundations of machine learning* (MIT press).

27. Mundt, M., Johnson, W. R., Potthast, W., Markert, B., Mian, A., & Alderson, J. (2021). A Comparison of Three Neural Network Approaches for Estimating Joint Angles and Moments from Inertial Measurement Units. *Sensors*, *21*(13), Article 13. https://doi.org/10.3390/s21134535

28. Muñoz, D. M., & Landínez, S. P. C. (2020). MODELO DE MACHINE LEARNING PARA LA DETECCIÓN DE EVENTOS DE MARCHA HUMANA. *Encuentro Internacional de Educación en Ingeniería*. https://doi.org/10.26507/ponencia.850

29. Nagpal, A., & Gabrani, G. (2019). Python for Data Analytics, Scientific and Technical Applications. *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 140-145. https://doi.org/10.1109/AICAI.2019.8701341

30. Naik, P., Naik, G., & M.B.Patil, M. (2022). *Conceptualizing Python in Google COLAB*.

31. Nibali, A., He, Z., Morgan, S., & Prendergast, L. (2018). *Numerical Coordinate Regression with Convolutional Neural Networks* (arXiv:1801.07372). arXiv. https://doi.org/10.48550/arXiv.1801.07372
32. Organizacion mundial de la salud. (2023). *Ceguera y discapacidad visual*. https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment
33. Paci, D., Cantoni, A., & Allegato, G. (2022). Magnetometers. En B. Vigna, P. Ferrari, F. F. Villa, E. Lasalandra, & S. Zerbini (Eds.), *Silicon Sensors and Actuators: The Feynman Roadmap* (pp. 477-501). Springer International Publishing. https://doi.org/10.1007/978-3-030-80135-9_14
34. Parreira, R. B., Grecco, L. A. C., & Oliveira, C. S. (2017). Postural control in blind individuals: A systematic review. *Gait & Posture*, *57*, 161-167. https://doi.org/10.1016/j.gaitpost.2017.06.008
35. Rebala, G, Ravi, A, & Churiwala. (2019). *An introduction to machine learning*. Springer.
36. Reyes Leiva, K. M., Gato, M. Á. C., & Olmedo, J. J. S. (2023). Estimation of Spatio-Temporal Parameters of Gait and Posture of Visually Impaired People Using Wearable Sensors. *Sensors*, *23*(12), Article 12. https://doi.org/10.3390/s23125564
37. Reyes Leiva, K. M., Jaén-Vargas, M., Codina, B., & Serrano Olmedo, J. J. (2021). Inertial Measurement Unit Sensors in Assistive Technologies for Visually Impaired People, a Review. *Sensors*, *21*(14), Article 14. https://doi.org/10.3390/s21144767
38. Reyes Leiva, K. M., Jaén-Vargas, M., Cuba, M. Á., Lara, S. S., & Olmedo, J. J. S. (2021). A Proposal of a Motion Measurement System to Support Visually Impaired People in

Rehabilitation Using Low-Cost Inertial Sensors. *Entropy*, *23*(7), Article 7. https://doi.org/10.3390/e23070848

39. Ribeiro, P. M. S., Matos, A. C., Santos, P. H., & Cardoso, J. S. (2020). Machine Learning Improvements to Human Motion Tracking with IMUs. *Sensors*, *20*(21), Article 21. https://doi.org/10.3390/s20216383
40. Salazar, R. D. V., & Mesa, A. A. C. (2019). Dispositivos de asistencia para la movilidad en personas con discapacidad visual: Una revisión bibliográfica. *Revista Politécnica*, *15*(28), Article 28. https://doi.org/10.33571/rpolitec.v15n28a10
41. *Salud visual—OPS/OMS | Organización Panamericana de la Salud*. (2018). https://www.paho.org/es/temas/salud-visual
42. Saucedo, A. C. G., Heredia, F. J. G., & Martínez, R. R. (2016). Discapacidad visual. *Cultura Científica y Tecnológica*, *51*, Article 51. Recuperado 23 de julio de 2023, de http://erevistas.uacj.mx/ojs/index.php/culcyt/article/view/954
43. Serrano, Emilio Soria, & Jose D Martin. (2009). *Redes neuronales artificiales*. Escuela Técnica Superior de Ingeniería; http://ocw.uv.es/ingenieria-y-arquitectura/1-2/libro_ocw_libro_de_redes.pdf.
44. Su, B., & Gutierrez-Farewik, E. M. (2020). Gait Trajectory and Gait Phase Prediction Based on an LSTM Network. *Sensors*, *20*(24), Article 24. https://doi.org/10.3390/s20247127
45. Teuwen, J., & Moriakov, N. (2020). Chapter 20—Convolutional neural networks. En S. K. Zhou, D. Rueckert, & G. Fichtinger (Eds.), *Handbook of Medical Image Computing and Computer Assisted Intervention* (pp. 481-501). Academic Press. https://doi.org/10.1016/B978-0-12-816176-0.00025-9

46. Vorobioff, J., Cerrotta, S., Morel, E., & Amadio, A. (2022). *Inteligencia Artificial y Redes Neuronales Fundamentos, Ejercicios y Aplicaciones:*

47. Zhang, L., & Suganthan, P. N. (2016). A survey of randomized algorithms for training neural networks. *Information Sciences*, *364-365*, 146-155. https://doi.org/10.1016/j.ins.2016.01.039