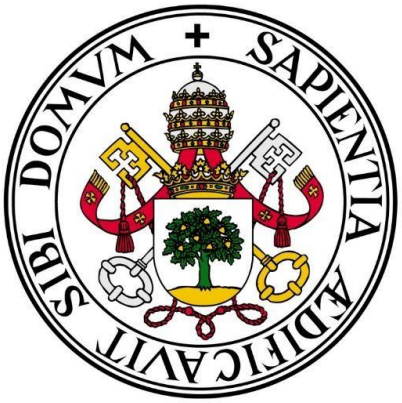

UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Estudio comparativo de redes de aprendizaje profundo para reconocimiento de actividades empleando sensores inerciales

MÁSTER UNIVERSITARIO EN INVESTIGACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Autor:

D. Daniel Antolínez López

Tutor:

Dr. D. Mario Martínez Zarzuela

Valladolid, 25 de enero de 2023

TÍTULO:	**Estudio comparativo de redes de aprendizaje profundo para reconocimiento de actividades empleando sensores inerciales**
AUTOR:	**D. Daniel Antolínez López**
TUTOR:	**Dr. D. Mario Martínez Zarzuela**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. FRANCISCO JAVIER DÍAZ PERNAS**
VOCAL:	**Dr. D. JAVIER AGUIAR PÉREZ**
SECRETARIO:	**Dra. Dña. MÍRIAM ANTÓN RODRÍGUEZ**
P.SUPLENTE:	**Dr. D. CARLOS GÓMEZ PEÑA**
S.SUPLENTE:	**Dra. Dña. MARÍA GARCÍA GADAÑÓN**
V.SUPLENTE:	**Dr. D. JESÚS POZA CRESPO**

FECHA: **Febrero de 2023**

CALIFICACIÓN:

Resumen de TFM

HAR (*Human Activity Recognition*) o reconocimiento de actividades humanas, es un campo de investigación centrado, como su nombre indica, en la capacidad de reconocer la actividad humana que está siendo realizada. Suponiendo, especialmente en la última década, un campo de elevada relevancia en investigación, dadas las posibilidades que ofrece para la mejora de la calidad de vida de las personas en ámbitos muy diversos, tales como el sanitario, el transporte o la seguridad entre otros.

Aunque este campo de investigación no es nuevo, la evolución reciente de los sistemas de sensores inerciales, y de los sistemas de computación (aprendizaje automático), han supuesto un incremento notable en la capacidad de los investigadores de trabajar en este problema. Por ejemplo, centrándonos únicamente en el campo del aprendizaje automático, es posible encontrar en la literatura más reciente propuestas de redes neuronales muy diversas, aportando diferentes opciones para poder dar una mejor solución al problema del HAR, superando los resultados anteriores.

Con esta última afirmación como base, en el presente trabajo se realiza una búsqueda de las arquitecturas de red, propuestas en la literatura de la última década, para resolver el problema del HAR con sensores inerciales. A continuación, se implementan algunas de dichas arquitecturas y se comparan los resultados con los obtenidos en un trabajo de fin de máster previo, empleando la base de datos pública REALDISP. Adicionalmente, se estudia cómo la utilización de datos de aceleración, cuaternios, orientación o su combinación afecta a la exactitud del reconocimiento. Este TFM ha permitido identificar las mejores aproximaciones de redes neuronales con sensores inerciales, así como cuáles son los mejores datos de entrada a utilizar, para el reconocimiento de actividades con la base de datos REALDISP.

Palabras Clave

Aprendizaje profundo, reconocimiento de actividades humanas, base de datos REALDISP, unidad de medida inercial.

Abstract

HAR (Human Activity Recognition) is a field of research focused, as its name suggests, on the ability to recognize the human activity being performed. It has become, especially in the last decade, a highly relevant field of research, given the possibilities it offers for improving the quality of life of people in many different fields, such as health, transportation or security, among others.

Although this field of research is not new, the recent evolution of inertial sensor systems and computing systems (machine learning) has led to a significant increase in the ability of researchers to work on this problem. For example, focusing only on the field of machine learning, it is possible to find in the most recent literature very diverse neural network proposals, providing different options to provide a better solution to the HAR problem, surpassing previous results.

With this last statement as a basis, in the present work a search of the network architectures, proposed in the literature of the last decade, to solve the HAR problem with inertial sensors is carried out. Then, some of these architectures are implemented and the results are compared with those obtained in a previous master's thesis, using the public database REALDISP. Additionally, it is studied how the use of acceleration, quaternions, orientation data or their combination affects the recognition accuracy. This TFM has allowed us to identify the best approaches of neural networks with inertial sensors, as well as the best input data to use for the recognition of activities with the REALDISP database.

Keywords

Deep learning, human activity recognition, REALDISP dataset, Inertial measurement unit.

Agradecimientos

Gracias a mi familia y a los compañeros del grupo de investigación de ITAP robótica médica, por su apoyo constante durante la realización del proyecto.

Gracias también a Javier, por encenderme en multitud de ocasiones el pc de trabajo.

Y finalmente gracias a mi tutor, por la ayuda ofrecida, para llevar a cabo el trabajo.

ÍNDICE

Índice de figuras

Índice de tablas

1.introducción y objetivos previos

1.1. Introducción, objetivos e hipótesis

Con el presente trabajo de fin de Master, se pretende realizar una "continuación" o ampliación del trabajo de fin de master previo, realizado por Sáez Bombín [1], en relación al campo de reconocimiento de actividades humanas (HAR) mediante aprendizaje automático. Empleando para la realización del presente trabajo una versión del framework creado por [2], y empleando como base de datos REALDISP [3], [4]. De tal forma que el presente trabajo sirva como ampliación y ejemplo de uso de los trabajos anteriormente mencionados.

Este trabajo, ubicado dentro del campo actual de estudio del Grupo de Telemática e Imagen (GTI) de la Universidad de Valladolid, pretende servir como continuación de la investigación de los resultados presentados en [1].

Así pues, el objetivo principal del trabajo, será dar respuesta a la pregunta de investigación principal: **¿Es posible encontrar en la literatura actual una arquitectura de red que permita obtener unos resultados iguales o superiores a los obtenidos con la red implementada por Sáez Bombín en su TFM?**, y a la secundaria **¿Son los cuaternios los mejores datos para trabajar con la base de datos REALDISP?**

Las hipótesis iniciales para ambas preguntas de investigación, son afirmativas, ya que, en el primer caso, la ciencia en el campo de reconocimiento de actividades con *Deep Learning* en los últimos años ha avanzado lo suficiente como para ofrecer alternativas que puedan dar mejores resultados que los obtenidos previamente. Y para la segunda pregunta, es probable que el empleo de otros datos diferentes a los cuaternios, o combinación de estos, aporte mejor resultados que el empleo independiente de los mismos.

1.2. Conocimientos previos

Para la realización de este trabajo, ha resultado vital el conocimiento previo adquirido, principalmente durante el curso académico del master MUITIC. Destacando principalmente la base técnica obtenida propia del campo del aprendizaje automático en las asignaturas de ARQUITECTURAS PARALELAS Y DEEP LEARNING, FUNDAMENTOS DE APRENDIZAJE AUTOMÁTICO y APRENDIZAJE AUTOMATICO AVANZADO. Así como la base para la realización de trabajos científicos vista en otras asignaturas del mismo curso.

Asimismo, también ha tenido cierta utilidad el conocimiento de sensores IMU (*inertial measurment Units*) adquirido durante un proyecto de investigación realizado en el grupo de investigación ITAP robótica médica. De forma que se tengan unos conocimientos básicos del funcionamiento de estos, así como de las medidas usuales que ofrecen.

1.3. Descripción del problema

A continuación, se realizará una breve sinopsis del problema a tratar, así como los requisitos técnicos y la estructura empleada en el presente trabajo.

1.3.1. El problema del HAR

El problema de HAR (*Human Activity Recognition*) o reconocimiento de actividades humanas, es un campo de investigación con una elevada relevancia en las últimas décadas [5], y en la actualidad, como se puede apreciar en el elevado número de bibliografía reciente del mismo, tal y como se indica en los diversos estados del arte disponibles, tales como [5]–[9] entre otros.

El objetivo de HAR es el de reconocer actividades humanas en entornos controlados y no controlados, con el potencial empleo del mismo en muy diversas áreas, como las indicadas en [8] (transporte, entretenimiento, locomoción, etc.), destacando en la actualidad en las áreas sanitarias o de bienestar, siendo tal su importancia, que existen *surveys* específicamente centrados en trabajos de HAR de este campo, como el trabajo de 2022 [6].

Si bien el problema de HAR no es nuevo, dado el avance en sensorización y computación (como el aprendizaje automático), se ha producido un importante avance en el mismo en las últimas décadas. Permitiendo que cada vez los procesadores tengan una mayor potencia de cómputo, y que por tanto se avance progresivamente en el empleo de técnicas más complejas de reconocimiento de actividades humanas imposibles previamente.

Aunque existen diversos métodos para el reconocimiento de actividades [5], [9], el método con mayor interés para la resolución del problema de HAR es el del aprendizaje automático, y dentro de este, destacando concretamente el aprendizaje profundo o Deep learning [8], dada su capacidad de procesar grandes cantidades de datos.

Entre los datos más comunes para el problema de HAR, se suele hacer uso de datos procedentes de sistemas de sensores, categorizados en [8] como sensores ambientales como ondas de radio, wifi o sistemas de navegación global por satélite (GNSS). Sensores portables (*Wearables*) como los sistemas inerciales (IMUS), constando de acelerómetros y giróscopos, cámaras, biosensores, etc…, adquiridos por sensores concretos, u otros dispositivos que pueda llevar el usuario y que contengan estos sensores (*Smartphone*).

También en la literatura, y en concreto en los *surveys* anteriormente citados [6], [7], o [8] podemos encontrar diversas arquitecturas de aprendizaje automático empleadas para el reconocimiento de actividades humanas. Categorizando los diversos tipos de preprocesado (filtrado, tratamiento, ventanas, etc…), selección de características (PCA, ICA, etc…) y redes empleadas en la literatura (supervisadas, semi-supervisadas y no supervisadas), como las redes convolucionales y recurrentes descritas en el trabajo [10]. Así como describiendo los principales datasets públicos para HAR con sus respectivas características (actividades, campo, etc.), como es el caso de la base de datos central de este trabajo REALDISP [3], [4].

1.3.2. Requisitos técnicos

Para la realización del presente trabajo, se ha empleado un ordenador del grupo de investigación GTI, contando con el sistema operativo Ubuntu, una tarjeta gráfica RTX 2080 TI de NVIDIA, y empleando el programa Visual Studio Code para la programación y entrenamientos, junto con el empleo del *framework* desarrollado en [2].

1.3.3. REALDISP

Como base de datos empleada para las comprobaciones de este trabajo, se ha escogido la base de datos REALDISP propuesta en [3], [4]. Disponible de forma abierta para su empleo en el repositorio de machine learning de la UCI. Seleccionada como base de datos para trabajar, dada la cantidad de información disponible para reconocimiento de actividades humanas, así como por ser el mismo dataset empleado por Sáez Bombín en [1].

La base de datos se compone de 1419 instancias, y 120 atributos. Con datos obtenidos de 17 sujetos realizando 33 actividades distintas, con captación mediante sensorización de aceleración, velocidad de giro, campo magnético y cuaternios. Con tres escenarios de colocación de la sensorización disponibles (ideal, auto-emplazamiento y desplazamiento inducido).

La sensorización se lleva a cabo mediante el empleo de 9 sensores colocados en diversas zonas del sujeto: S1: parte inferior del brazo derecho (RLA), S2: brazo derecho (RUA), S3: espalda (BACK), S4: parte superior del brazo izquierdo (LUA), S5: parte inferior del brazo izquierdo (LLA), S6: pantorrilla derecha (RC), S7: muslo derecho (RT), S8: muslo izquierdo (LT), S9: pantorrilla izquierda (LC). Ofreciendo cada sensor datos 3D de aceleración (accX, accY, accZ), giroscopio (gyrX, gyrY, gyrZ), orientación del campo magnético (magX, magY, magZ) y 4 dimensiones de cuaternios (Q1, Q2, Q3, Q4).

En la Tabla 1 se incluyen los nombres de las 33 actividades etiquetadas en el dataset REALDISP:

Tabla 1 Actividades de la base de datos REALDISP.

A1: Caminar	A18: Torsión opuesta de la parte superior del tronco y la parte inferior del cuerpo
A2: Correr despacio (*Jogging*)	A19: Elevación lateral de brazos
A3: Correr (*Run*)	A20: Elevación frontal de brazos
A4: Saltar hacia arriba	A21: Palmas frontales
A5: Saltar hacia delante y hacia atrás	A22: Cruce frontal de brazos
A6: Salto lateral	A23: Rotación de los hombros de gran amplitud
A7: Saltar con las piernas/los brazos abiertos/cerrados	A24: Rotación de los hombros de baja amplitud
A8: Saltar a la cuerda	A25: Rotación interna de los brazos
A9: Giro de tronco (brazos extendidos)	A26: Rodillas (alternadas) hacia el pecho
A10: Giro de tronco (codos doblados)	A27: Talones (alternativamente) hacia la espalda
A11: Flexión de cintura hacia delante	A28: Rodillas en flexión (en cuclillas)
A12: Rotación de la cintura	A29: Rodillas (alternadas) doblando hacia delante
A13: Flexiones de cintura (alcanzar el pie con la mano contraria)	A30: Rotación sobre las rodillas
A14: Alcance de los talones hacia atrás	A31: Remo
A15: Flexión lateral (10 a la izquierda + 10 a la derecha)	A32: Bicicleta elíptica
A16: Flexión lateral con el brazo hacia arriba (10_ a la izquierda + 10 a la derecha)	A33: Ciclismo
A17: Estiramiento repetitivo hacia delante	

1.3.4. Estructura de la memoria

La memoria se ha estructurado de la siguiente forma:

En el Capítulo 1, se ha realizado una introducción previa al problema a tratar, con una breve descripción del mismo, así como de los requisitos técnicos del trabajo.

En el Capítulo 2, se procederá a la revisión del estado del arte de HAR, mediante el estudio de diversos artículos que traten este problema en los últimos tiempos, tanto con otras bases de datos, como con REALDISP.

En el Capítulo 3, se procede a analizar el desarrollo del trabajo realizado, con la búsqueda de información llevada a cabo, así como las comprobaciones y pruebas realizadas, con las redes encontradas en estos últimos.

En el Capítulo 4, se analizarán las principales conclusiones del trabajo, así como las líneas futuras posibles del mismo.

Por último, en el Anexo A se incluye información adicional sobre las redes utilizadas.

2.Revisión del estado del arte

Durante el desarrollo del presente apartado, se realizará una revisión del estado del arte actual del empleo de Deep learning en el problema de HAR (*Human Activity Recognition*). De forma que se tenga una visión general del estado actual de la técnica según la literatura, así como emplear el conocimiento adquirido durante el mismo, como base fundamental del desarrollo del presente trabajo.

Para el análisis de la soluciones propuestas sobre el problema de HAR, se desglosarán algunas de las técnicas empleadas en la literatura actual, dividiéndose el estado del arte por una parte en redes de aprendizaje profundo en general para el problema del HAR (con diversas bases de datos), y posteriormente por un segundo aparatado que desglose en concreto las redes empleadas con la base de datos REALDISP [3], [4].

En ambos casos, para la búsqueda de literatura, se han empleado los buscadores Google académico y IEEEXPLORE, con las palabras clave *Deep learning*, HAR, REALDISP y sensores entre otras (de forma tanto conjunta como separada). Tratando de seleccionar aquellos trabajos más recientes (de los últimos 5 años a ser posible), que sean interesantes bien por su planteamiento del preprocesado, o de la arquitectura, y especialmente por los resultados obtenidos con estos en los entrenamientos.

Cabe destacar la existencia de diversos trabajos del estado del arte del HAR en los últimos años [5] ,[6], [9], [10] y [11]. Sirviendo como guía y como base de los principales planteamientos para abordar el reconocimiento de actividades humanas.

Hay que tener en cuenta que, dada la gran cantidad de bibliografía disponible, en este trabajo únicamente se desarrollaran algunos de los trabajos más relevantes en cuanto a arquitectura o resultados obtenidos en la literatura según el criterio del autor. Pudiendo encontrarse otros trabajos en la literatura o en los *surveys* similares a los aquí mencionados, y que pudieran o no obtener mejores resultados que los seleccionados.

2.1. Redes de aprendizaje profundo en reconocimiento de actividades

A continuación, se desglosan algunas de las redes más destacables de los últimos años encontradas durante al análisis de la literatura actual, ordenadas según el grado de complejidad aparente de las mismas, y recopiladas finalmente en una tabla resumen al final del apartado.

Algunas de las redes más destacadas encontradas en la literatura que **no** emplean REALDISP [3], [4] para el problema de HAR son:

2.1.1. LSTM

En el trabajo de Hoffman et al. [12], se propone el empleo de una red LSTM formada por 4 capas de 32 nodos y una capa *dropout*, tal y como indica se indica en la Figura 1.

Se emplearon como base de datos WISDM y GAMI. Contando WISDM [13] con 1.098.207 datos de 36 sujetos con *Smartphones* colocados en la cintura, realizando las 6 actividades diarias: *walking (38.6%), jogging (31.2%), climbing stairs up (11.2%), climbing stairs down (9.1%), sitting (5.5%) y standing (4.4%),* eliminando en el trabajo los de *climbing stairs up and down*. GAMI por su parte cuenta con 4 actividades: *walking, waiting (standing), jogging y sitting*, adquiriéndose los datos mediante un teléfono Android en 10 personas.

Para los mejores resultados obtenidos mediante el empleo de la red (*learning rate*: 0.001, *dropout rate*: 0.5, *batch size*: 64, *activation function: tanh, optimizer*: Adam), se obtuvo un *accuracy* de 98.34% con WISDM y de 96.81% con GAMI.

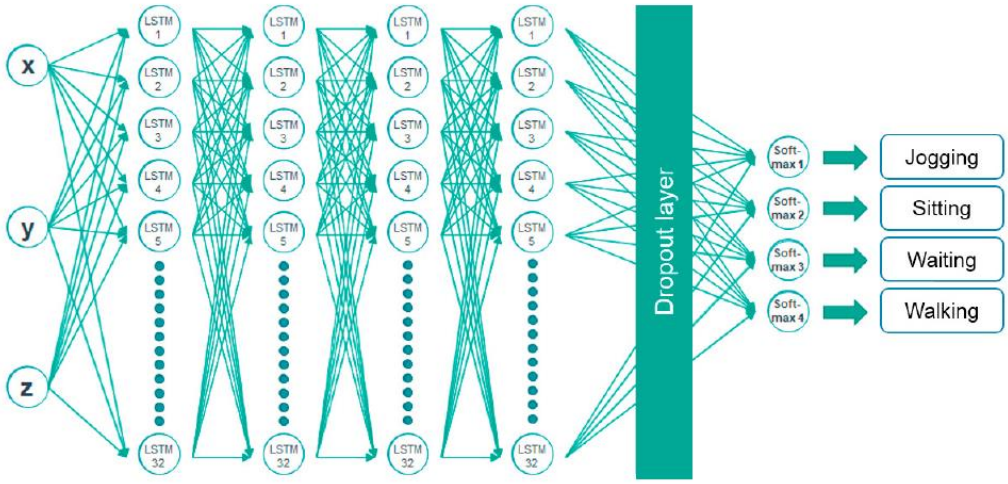

Figura 1 configuración LSTM de la red [12].

2.1.2. CNN model color coded

En el trabajo de Yatbaz et al. [14], se propone realizar un preprocesado a las imágenes empleadas como entradas (combinación de las señales), y normalizadas (min-max y filtros medianos escalonados), pasando por una red de 4 capas convolucionales de 32 canales cada uno y kernel 3x3, con activación RELU después de cada convolución y *maxpooling* con kernel 1x2. Pudiendo observar las características de la configuración en la Figura 2 y la red en la Figura 3.

La base de datos empleada en el artículo es MHEALTH [15], [16], con datos de sensores (acelerómetro, giroscopio, magnetómetro y ECG), colocados en pecho, tobillo izquierdo y brazo derecho de 10 sujetos. Incluyéndose en el conjunto de datos 12 actividades, de las cuales se emplean 8 en el trabajo (*Standing still, Sitting and relaxing, lying down,*

walking, cycling, climbing stairs, jogging y running). Empleando una proporción de los datos 80:20 (*train, test*), ventanas de 1s y solapamiento del 50%.

Obteniendo como resultado en el mejor caso empleando esta red y para el dataset MHEALTH, con los datos balanceados y 5-fold, un *accuracy* medio de 96.92%, y un *F1-score* de 95.15%.

- **Image Size:** 7x50x3
- **Batch Size:** 32
- **Number of Convolutional Layers:** 4
- **Number of Fully Connected Layers:** 3
- **Dropout Rate:** 0.25
- **Validation Method(s):** LOSO and 5-Fold
- **Pooling:** 1x2 Kernel
- **Optimizer:** Sthocastic Gradient Descent (SGD)
- **Activation Function:** RELU
- **Loss Function:** Cross-Entropy Loss
- **Learning Rate:** 0.075
- **Epochs:** 500
- **Early Stop Condition:** No improvement for 15 consecutive epochs

Figura 2 Características de la red [14].

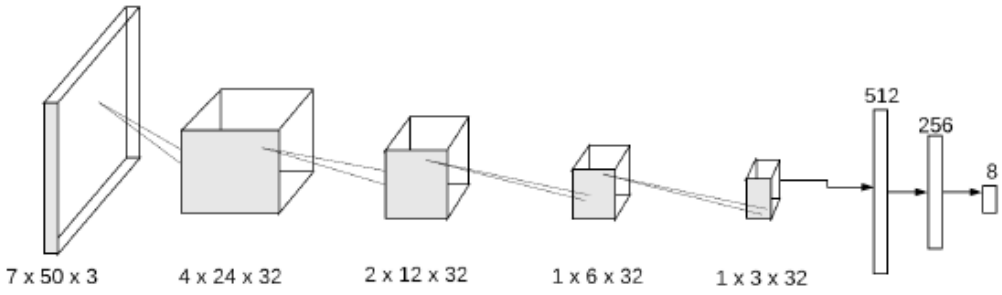

Figura 3 Arquitectura propuesta [14].

2.1.3. FFNN-CNN

En el trabajo de Gholamiangonabadi et al. [17], se propone el empleo de LOSOCV (*Leave-One-Subject-Out Cross-Validation*) para comprobar en la evaluación la capacidad de generalización de diversas arquitecturas de red. Las redes más destacadas propuestas para la evaluación con LOSOCV frente a 10k-fold son:

CNN-2C: Dos capas convolucionales con 64 neuronas, *maxpooling* de 32, convolución, *maxpooling* y dos *fully connected layers* de 64, correspondiente a la Figura 4.

FFNN-4H: Red FFNN con 4 capas ocultas de 128 neuronas, con tamaño de entrada de la ventana deslizante de 10, correspondiente a la Figura 5.

Emplea como base de datos MHEALTH al igual que en el trabajo [14], pero en este caso empleándose las 12 actividades del dataset . Consiguiendo mediante evaluación con LOSOCV un *accuracy* en el mejor escenario (con tamaño de ventana deslizante 20) de 81.1% para FFNN-4H y 85.1% para CNN-2C.

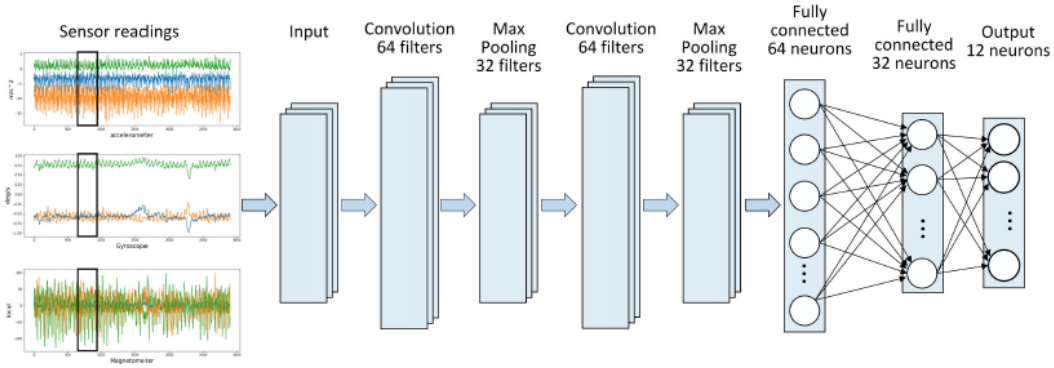

Figura 4 Arquitectura de red CNN-2C [17].

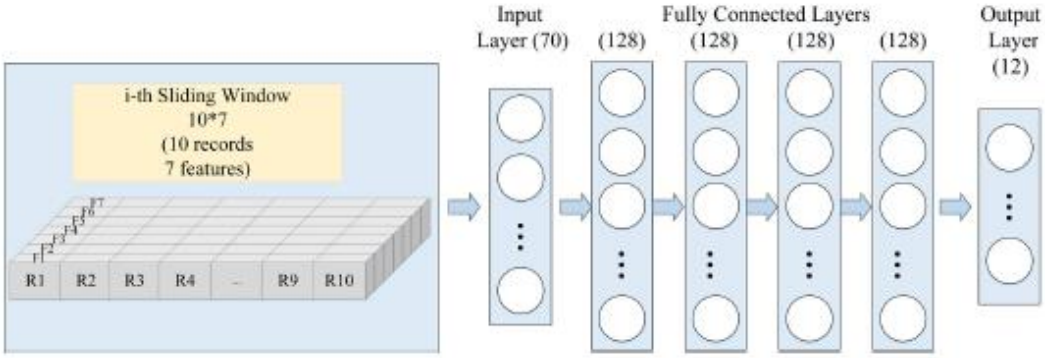

Figura 5 Arquitectura de la red FFNN-4H [17].

2.1.4. DeepConvLSTM

En el trabajo de Bock et al. [18], se propone una arquitectura *DeepConvLSTM* reducida, con solo una capa LSTM en lugar de las 2 tradicionales, tal y como se indica en Figura 6. Empleando para la comprobación de la ventaja de esta modificación 5 bases de datos: OPORTUNITY, HHAR, Wetlab, RWHAR y SBHAR.

El dataset OPORTUNITY [19], se corresponde con un dataset muy empleado en el problema de HAR, consistente en datos obtenidos de 4 sujetos realizando seis series de ejercicios diarios, constando de 17 actividades a predecir (*open/ close door 1 and 2, fridge, dishwasher and drawer 1, 2 and 3, clean table, drink from cup y toggle switch*). Aplicando a su vez en el trabajo un preprocesado a los datos similar al realizado en [20],

rellenando datos con interpolación lineal y normalizando por canal, constando finalmente el dataset de 113 canales, con datos obtenidos de los sensores inerciales a 30Hz (acelerómetros, giroscopios y magnetómetros), con 6 horas de grabaciones.

El *dataset* HHAR [21] está formado por datos de 9 sujetos realizando 6 actividades del día a día (*biking, sitting, standing, walking, walking upstairs y down stairs*) muestreados a 100Hz. En [18] únicamente se emplean los datos de los sensores de aceleración de este *dataset*.

El *dataset* Wetlab [22] consta de datos obtenidos de 22 participantes durante pruebas de extracción de ADN, captándose los datos con una unidad de detección de aceleración en la muñeca a 50Hz, distinguiendo entre 9 clases (*cutting, inverting, peeling, pestling, pipetting, pouring, stirring, transfer*).

El *dataset* RWHAR [23] consta de datos de 15 participantes y 8 actividades (*walking upstairs, walking downstairs, jumping, lying, standing, sitting, running, walking*). Siendo los datos obtenidos por un sensor colocado en la muñeca 50Hz. En [18] únicamente se emplean los datos de los sensores de aceleración de este *dataset*.

Finalmente, el *dataset* SBHAR [24] cuenta con datos de 30 participantes, realizando actividades diarias, con 6 actividades (*standing, sitting, lying, walking, walking downstairs, walking upstairs*) y 6 transiciones entre posturas (*stand-to-sit, sit-tostand, sit-to-lie, lie-to-sit, stand-to-lie, lie-to-stand*). Al igual que en el caso anterior, en el trabajo [18] se empleó únicamente los datos de acelerómetro obtenidos a 50Hz.

Con los dataset anteriormente mencionado, y la red propuesta, obtuvieron evaluando con LOSO *cross-validation* resultados de F1-SCORE con 1 layer LSTM en el mejor caso: 51% (128 *hidden units*), 46.7% (256), 39.2% (256), 74.4% (512) y 71.6% (512) respectivamente. Empleando para el entrenamiento ventanas de 0.5s y solapamiento del 50% para OPORTUNITY, y 1s y 60% en el resto, con Adam de optimizador.

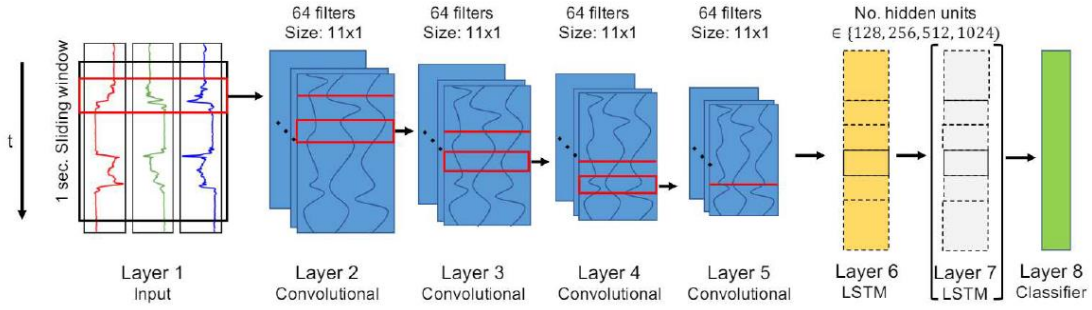

Figura 6 Arquitectura DeepConvLSTM reducida [18].

2.1.5. GAN-LSTM

En el trabajo de Moshiri et al. [25], se propone el empleo de un sistema semi-supervisado para reconocimiento de actividades basadas en la información del estado del canal (CSI), para reconocer actividades humanas en interiores y su correlación con las señales wifi reflejadas por los sujetos en movimiento. Para ello propone el empleo de PCA (*principal component análisis*), y la transformada corta de Fourier (STFT) con los datos del CSI. Para el entrenamiento proponen el empleo de una estructura GAN (*Generative Adversarial Network*) para la generación de los datos sintéticos (empleando la mitad de los datos del dataset), esquematizado en la Figura 7 .Empleando para la clasificación los datos sintéticos y originales para el entrenamiento mediante LSTM con 200 *hidden layers*.

Como dataset se empleó CSI dataset cedido por [26], compuesto por datos CSI de amplitud y fase, recogidos a 1KHz y para 7 actividades (*lying down, falling, walking, running, sitting down y standing up*) obteniendo un accuracy sin usar datos sintéticos de 92.8% y empleando 50/50 de 87.2%.

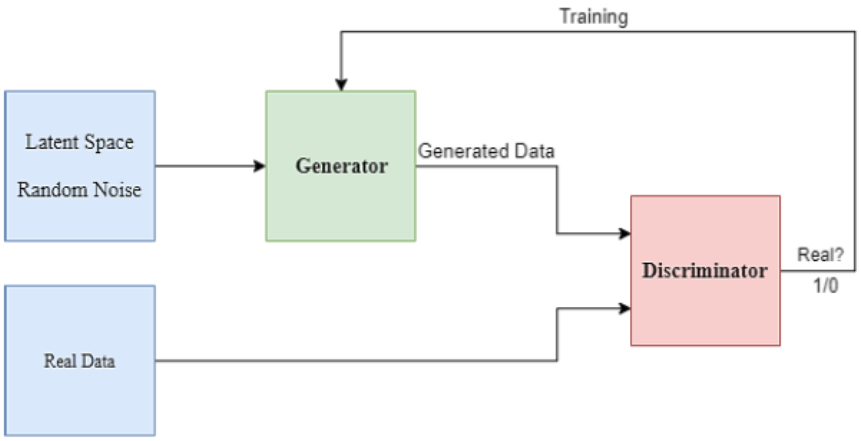

Figura 7 Diagrama del funcionamiento de la red GAN [25].

2.1.6. PSDRNN

En el trabajo de Li et al. [27], se emplea un sistema basado en extracción de características y *Deep learning*, siendo la principal diferencia respecto a las DRNN clásicas, la extracción de los vectores PSD (*power spectral density*) como entrada a la red recurrente DRNN. Calculándose en primer lugar la aceleración lineal a partir de las aceleraciones (tres ejes) ofrecidas por el acelerómetro. A partir de las cuales se obtiene el vector PSD empleando la transformada rápida de Fourier (STFT), mediante una ventana móvil en cada registro. Siendo estos vectores PSD los que se introduzcan en la red para entrenar. Un esquema de la red lo encontramos en la Figura 8.

El modelo DRNN propuesto, consiste en 2 capas ocultas LSTM de 80 *units* y otra para softmax, como el de la Figura 9. Empleando para el STFT una ventana de 30 con solapamiento de longitud 25 (83.33%).

El *dataset* empleado para los resultados es el UniMiB SHAR. Este dataset está compuesto por datos de aceleraciones recogidos por un *Smartphone* colocado en la parte trasera del pantalón de los sujetos, recolectando muestras a 50Hz, y con un total de 17 clases (*standing up from sitting, standing up from lying, walking, running, going upstairs, jumping, going downstairs, lying down from standing, sitting down, generic falling forward, falling rightward, generic falling backward, hitting on obstacle in the fall, falling with protection strategies, falling backward while sitting chair, syncope, y falling leftward*).

Consiguiendo en la evaluación con el tri-PSDRNN (PSDRNN, pero con un PSD para cada aceleración de forma independiente) un 96.52% en *weighted* F1-score, y un MAA (*macro-average accuracy*) de 94.66% con el *dataset* UniMiB SHAR.

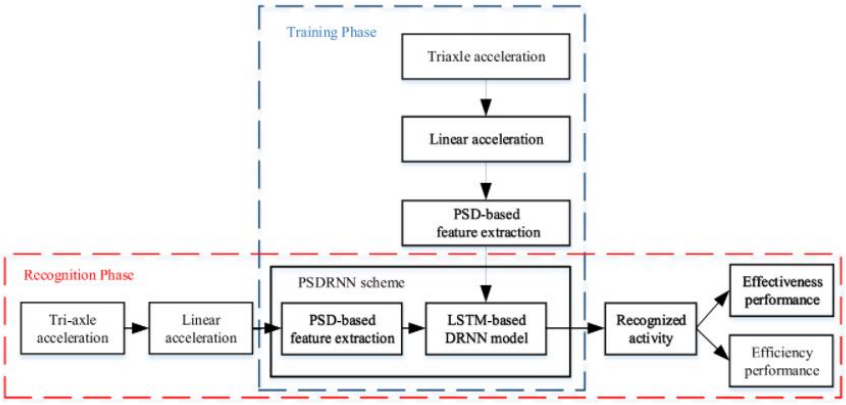

Figura 8 Arquitectura propuesta PSDRNN [27].

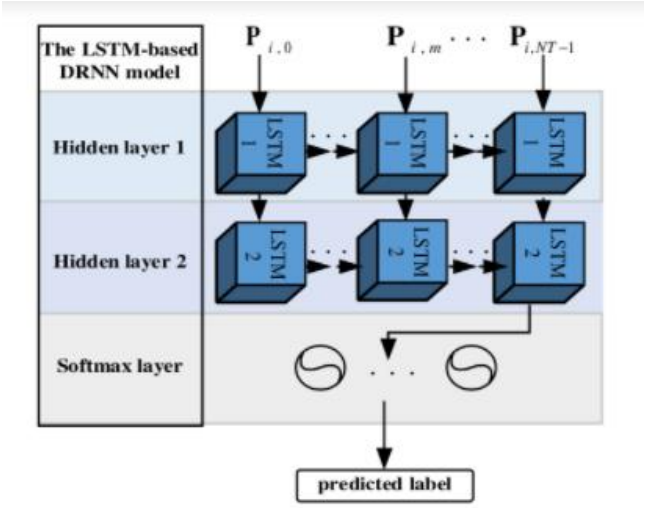

Figura 9 Arquitectura DRNN [27].

2.1.7. LSTM-CNN

En el artículo de Xia et al. [28], se propone una red de 8 Capas: 2 LSTM de 32 neuronas, otras dos capas convolucionales, una de 64 y otra de 128 con capas *maxpooling* en medio y finalmente una capa GAP (*global average pooling*), y una capa *batch normalization* (BN), observable en la **¡Error! No se encuentra el origen de la referencia.**. Empleando en el preprocesado interpolación lineal, normalizado y escalado de los datos, con los hiperparámetros de la **¡Error! No se encuentra el origen de la referencia.** .

Las bases de datos empleadas en el artículo son UCI-HAR, WISDM y OPORTUNITY. Empleando para la segmentación de datos ventana deslizante de 50% de solapamiento y tamaño 128, con excepción de OPORTUNITY, para el cual se emplea un tamaño de 24.

La base de datos UCI-HAR [24], contiene datos de 30 sujetos, con datos de los sensores inerciales de los *Smartphones* que portaban en la zona de la cintura. Constando el *dataset* de 6 actividades cotidianas (*standing, laying, walking, walking downstairs y walking upstairs*), así como las posturas intermedias (*standing to sitting, sitting to standing, sitting to laying, laying to sitting, standing to laying, laying to standing*). Captando a una frecuencia de 50Hz las aceleraciones y la velocidad angular (3 ejes) de los sensores. Empleándose en el trabajo [28] únicamente las seis cotidianas.

Las bases de datos WISDM y OPORTUNITY ya se comentaron anteriormente.

Como resultados en las mejores condiciones, se obtuvieron unos valores F1-SCORE en los dataset UCI-HAR, WISDM y OPORTUNITY de respectivamente de 95,78%, 95,85% y 92.63%.

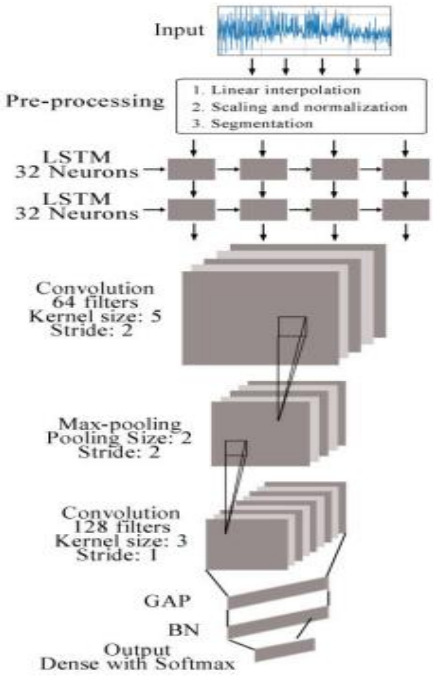

Figura 10 Esquema de la red propuesta [28] .

TABLE 6. List of selected hyper-parameters.

Stage	Hyper-parameters		Selected Values
Data Preprocessing	Window Size		128/24
	Step Size		64/12
Architecture	LSTM_1 Neurons		32
	LSTM_2 Neurons		32
	Convolution_1	Kernel Size	5
		Stride	2
		Filters	64
	Max-pooling	Pooling Size	2
		Stride	2
	Convolution_2	Kernel Size	3
		Stride	1
		Filters	128
Training	Optimizer		Adam
	Batch Size		192
	Learning Rate		0.001
	Number of Epochs		200

Figura 11 Hiperparámetros de la red [28].

2.1.8. Resumen de trabajos previos sobre HAR.

En la Tabla 2 se realizará un resumen de los datos comentados durante el presente aparatado en forma de tabla, de forma que se pueda comparar de forma sencilla las arquitecturas, bases de datos y preprocesados empleados en cada uno de los artículos anteriormente detallados.

Tabla 2 Resumen de los trabajos encontrados en la bibliografía.

Método	Red empleada	Preprocesado	Bases de datos	Actividades	Accuracy	F1-score
LSTM	LSTM de 4 capas de 32 nodos + *dropout* + *early stopping*	Ventana deslizante de 50	WISDM	*Walking, jogging, sitting y standing*	98.34%	NE
			GAMI	*walking, waiting (standing), jogging y sitting*	96.81%	NE
CNN model color coded	4 capas convolucionales de 32 canales cada uno y kernel 3x3, con activación RELU y *maxpooling* con kernel 1x2	Combinación y normalizado de las señales. ventanas de 1s y solapamiento del 50%	MHEALTH	*Standing still, Sitting and relaxing, lying down, walking, cycling, climbing stairs, jogging y running*	96.92%	95.15%
FFNN-CNN	CNN2: 2 conv. de 64, 2 Max. de 32, 2 *fully*. de 64. Evaluación LOSOCV	Ventana deslizante variadas (10,20, etc)	MHEALTH	*Standing still, Sitting and relaxing, Lying down, Walking, Climbing stairs,Waist bends forward, Frontal elevation of arms, Knees bending (crouching), Cycling, Jogging, Running, Jump front & back*	85.1%	NE
	FFNN con 4 capas ocultas de 128. Evaluación LOSOCV				81.1%	
DeepConvLSTM1	DeepConvLSTM reducida, con solo una capa LSTM	ventanas de 0.5s y solapamiento del 50% para OPORTUNITY, y 1s y 60% en el resto. Interpolación lineal de datos faltantes y normalizado	OPORTUNITY	*Open/ close door 1 and 2, fridge, dishwasher and drawer 1, 2 and 3, clean table, drink from cup y toggle switch.*	NE	51% (128 hidden units)
			HHAR	*biking, sitting, standing, walking, walking upstairs y downstairs*	NE	74.4% (512)
			Wetlab	*cutting, inverting, peeling, pestling, pipetting, pouring, stirring, transfer*	NE	46.7% (256)
			RWHAR	*walking upstairs, walking downstairs, jumping, lying, standing, sitting, running, walking*	NE	39.2% (256)
			SBHAR	*standing, sitting, lying, walking, walking downstairs, walking upstairs*	NE	71.6% (512)

GAN-LSTM	GAN (generación de datos) y LSTM de 200	PCA y STFT	CSI	*lying down, falling, walking, running, sitting down y standing up*	92.8%	NE
PSDRNN	2 capas ocultas LSTM de 80 units y otra para softmax	STFT con aceleraciones, una ventana de 30 con solapamiento de longitud 25	UniMiB SHAR	*standing up from sitting, standing up from lying, walking, running, going upstairs, jumping, going downstairs, lying down from standing, sitting down, generic falling forward, falling rightward, generic falling backward, hitting on obstacle in the fall, falling with protection strategies, falling backward while sitting chair, syncope, y falling leftward*	94.66% (WAA)	96.52%
LSTM-CNN	2 LSTM de 32, dos capas conv., una de 64 y otra de 128 con capas *maxpooling* y una capa GAP y una capa *batch normalization*	interpolación lineal, normalizado y escalado de los datos. ventana deslizante de 50% de solapamiento y tamaño 128 (salvo OPORTUNITY 24)	UCI-HAR	*standing, laying, walking, walking downstairs y walking upstairs*	NE	95,78%,
			WISDM	*Walking, jogging, sitting y standing*		95,85%
			OPORTUNITY	*Open/ close door 1 and 2, fridge, dishwasher and drawer 1, 2 and 3, clean table, drink from cup y toggle switch.*		92.63%

* NE: No empleado, o no encontrado en la bibliografía consultada.

2.2. Propuestas de aprendizaje automático para reconocimiento de actividades en el dataset REALDISP

En el presente apartado se realizará un desglose de las diferentes técnicas de aprendizaje automático empleadas para trabajar con el dataset REALDISP [3], [4] encontradas en la literatura. Para ello, como en el apartado anterior, se procederá a comentar las técnicas encontradas en orden de complejidad relativa (de menor a mayor), y con una tabla resumen al final del apartado.

2.2.1. VersaTL

El trabajo de conferencia de Abdu-Aguye y Gomaa [29] hace empleo de una red convolucional consistente en dos capas convolucionales para extracción de características, y clasificación mediante redes neuronales recurrentes de tres capas, con una capa intermedia de *pooling espacial piramidal* (SPP), para convertir las salidas del filtrado inicial (extracción), en un vector de tamaño fijo para la clasificación. De forma que se pueda realizar el aprendizaje con datos fijos y variables de entrada.

Para la evaluación se emplearon 5 *datasets*, incluido REALDISP [3], [4], siendo los otros 4: Gomaa-1 [30], HAPT [24], *Daily and Sports Activities* [31] y HAD-AW [32].

Centrándonos únicamente en este caso en REALDISP, en el trabajo se utilizaron como entradas únicamente giroscopios y acelerómetros, usando el 75% de los datos para *train* y el resto para test. Los autores consiguieron un accuracy máximo de 77.40 ± 2.28 en el caso de usar un *pooling* de 8-4-2-1 (en la pirámide).

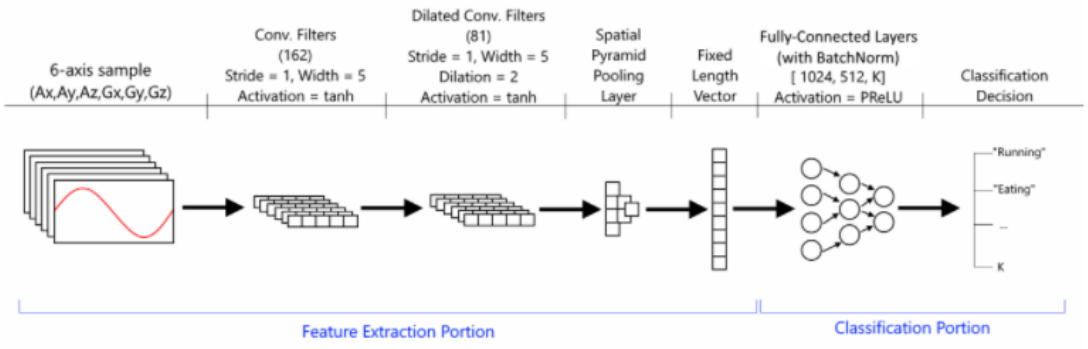

Figura 12 Arquitectura VersaTL [29].

2.2.2. FE-AT

En el trabajo de conferencia de Nguyen et al [33], proponen un sistema denominado FE-AT, un aprendizaje basado en atributos y semi supervisado (supervisado + aprendizaje basado en atributos), combinación de FE (basado en características) y AT (basado en atributos). Emplea *Random oversampling* para replicar datos de nueva actividad (evitar sistema desbalanceado). Uso de matriz actividad-característica en el AT (basado en atributos), de forma que se empleen características semánticas para la clasificación, con una arquitectura como la de la Figura 13 . Empleo de *cross-validation*, con datos de todos los usuarios, salvo uno para test.

Los experimentos se realizan con tres *dataset*: REALDISP [3], [4], MHealth [15], [16] y *Daily And Sport* [31] . Se emplean ventanas de 5 segundos, con un 50% de solapamiento para la extracción de características de los sensores.

Para la base de datos REALDISP, los autores obtuvieron un F1-SCORE máximo en torno a 0.82, de acuerdo a la Figura 14.

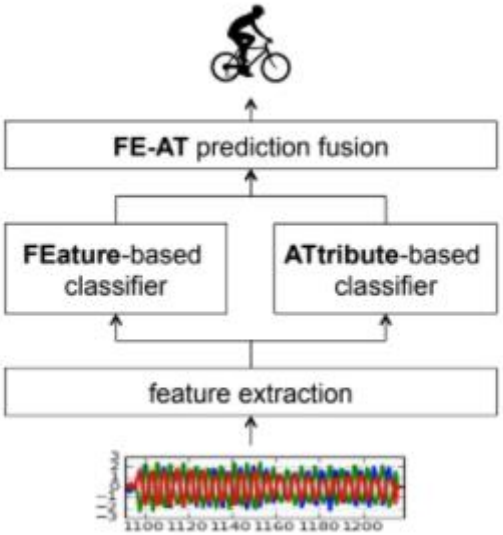

Figura 13 Arquitectura FE-AT [33].

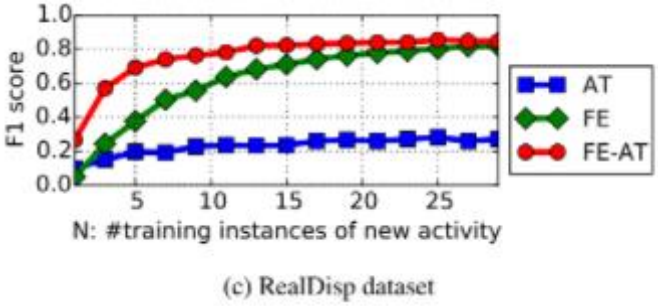

Figura 14 Resultados con REALDISP [33].

2.2.3. Adaptative Pooling

En el trabajo de conferencia de Abdu-Aguye et al. [34], se propone el empleo de *pooling* adaptativo para la clasificación con redes convolucionales (CNN) para evitar el afinado de hiperparámetros. Consistiendo el *pooling* adaptativo una generalización del *pooling* piramidal, describiendo la extensión de este a dimensiones arbitrarias (aunque se centran en una dimensión). La estructura propuesta es la de la Figura 15. Para la comprobación durante el trabajo se comparó el rendimiento empleando una configuración aleatoria de los hiperparámetros, con o sin *pooling* adaptativo.

Para la evaluación emplean REALDISP y concretamente la información de giroscopio y acelerómetro. Preprocesado los datos en ventanas de tamaño fijo de 1s, empleando el 75% de los datos para train (13.963 datos disponibles). Obteniendo al emplear el método propuesto con una convolución un *accuracy* medio de 94.14%.

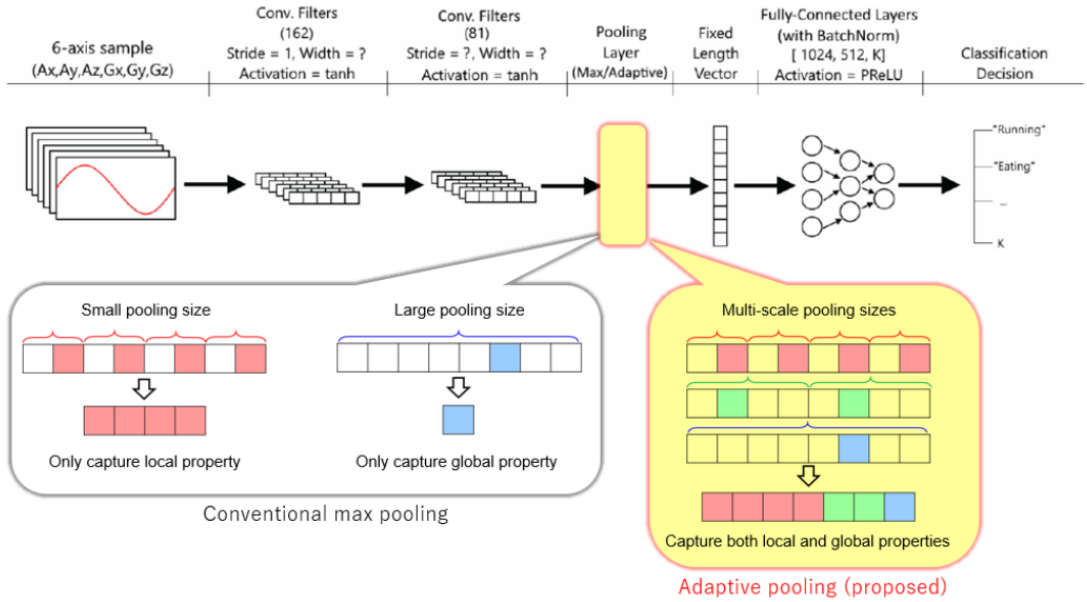

Figura 15 Arquitectura de red propuesta [34].

2.2.4. SMART

En el trabajo de conferencia de Ghorpade et al. [35], se propone una arquitectura de red de nombre SMART, realizando un sistema semi supervisado, basado en el empleo de un autoencoder con dos LSTM de 256 y 128 neuronas, 3 *layers* de densidad, con la del medio de 3 neuronas como layer embebida y 9 neuronas a la salida. Además, se hace uso de un algoritmo de *K-nearest Neighbors* para identificar las etiquetas más probables, y un *clustering DBSCAN* para la separación de las actividades y poder identificar nuevas actividades denominadas en el trabajo actividades emergentes. Arquitectura observable en la Figura 16.

La base de datos empleada es REALDISP, empleando 9 sensores y 17 sujetos, con ventana deslizante de 1s. Incluye 2 actividades emergentes (*Jogging y Waist bends forward*), y otras 6 como existentes (*Walking, Running, Jump frontyback, Frontal elevation of arms, Knee bending y Cycling*).

Para actividades existentes se empleaba 30% datos para *test*, y para actividades emergentes se seleccionaron aleatoriamente 50 registros para *test*. Creando en total 10 *sets* de *test* de referencia.

Para la evaluación se emplean las métricas del F1-score y *precision-recall*. Consiguiendo en los resultados más destacados un F1-SCORE en actividades existentes de 0.8 aproximadamente, a partir de 20 registros de entrenamientos y de 0.85 aprox. para actividades emergentes con menos de 20 registros. Resultados contemplados en la Figura 17.

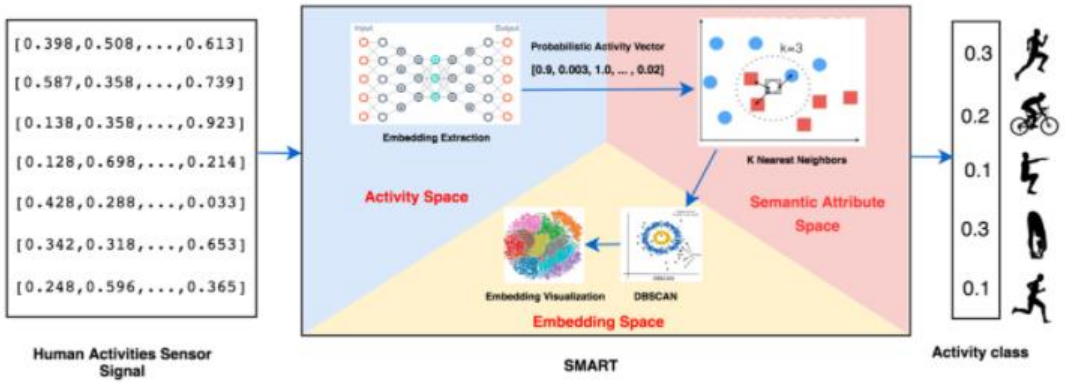

Figure 1: System Architecture

Figura 16 Arquitectura SMART [35].

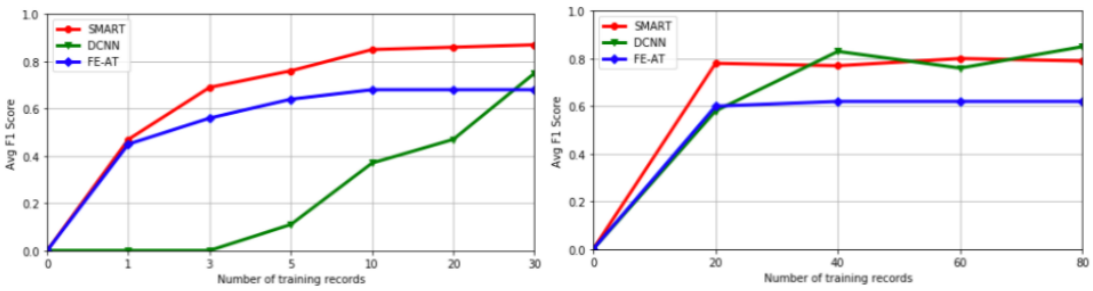

Figura 17 Resultados del empleo de SMART [35].

2.2.5. SMLDIST

En el trabajo de Chen et al. [36], los autores proponen un sistema denominado SMLDist (*Stage-Memory-Logits distillation*), basado en mejorar el funcionamiento de las redes convolucionales, contando con tres niveles de destilación de información para la red (*Stage, memory, logits*).

Mediante esta propuesta se pretendía enseñar a la red como extraer más características de las señales recibidas por los sensores, de forma que en cada una de las etapas en las que se divide el diseño, el modelo profesor generado va guiando a la red estudiante, sobre que debe extraer en cada etapa. También para la última etapa, se desarrolló un sistema auto adaptativo memoria intuición (SIMU), para enseñar al sistema como relacionar las características extraídas en la penúltima etapa. La estructura propuesta se aprecia en la Figura 18 y en la Figura 19.

Para la evaluación se emplearon 7 *datasets* (RealWorld-HAR [23], MHEALTH [15], [16], HAPT [24], HTC-TMD [37], UCI-HAR [24], DSADS [31], [38], [39]), incluido REALDISP [3], [4], para el cual emplean ventanas de 3 segundos.

En el caso del dataset REALDISP y empleando SIMU, los autores obtuvieron un *accuracy* de 94% y un F1 Macro de 93.79%.

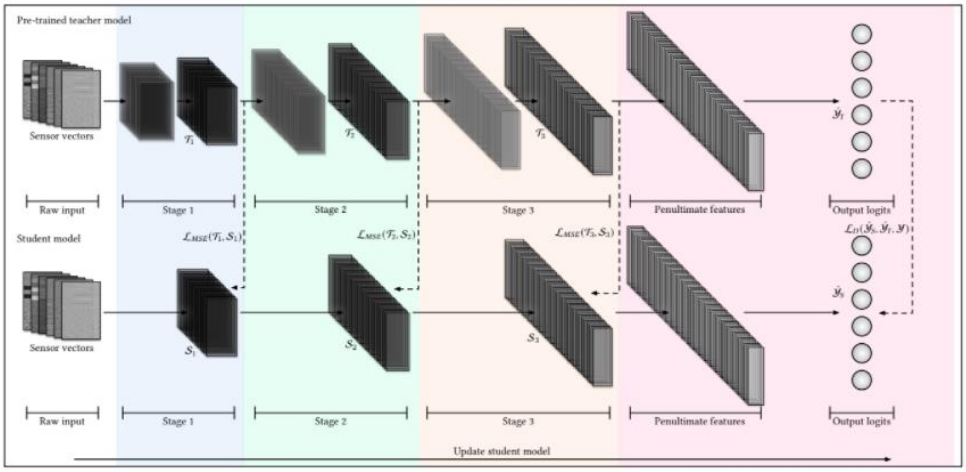

Figura 18 Arquitectura SMLDIST [36].

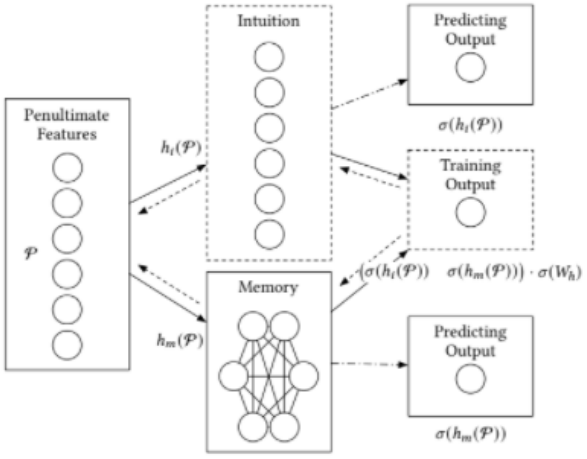

Figura 19 Diagrama SMLDIST [36].

2.2.6. FE-Random Forest

En el artículo de Zhu et al. [40], se propone la extracción de características, y el empleo de *machine learning* para clasificación, como se observa en la Figura 20. Para la extracción de características, se agrupan las señales inerciales en frames (con ventanas de 3s), para los cuales se calcula un vector de características, siendo algunas de las características obtenidas la media, la correlación o SMA entre otras. Empleándose características del dominio temporal, y de frecuencia como la energía.

Por otra parte, para la clasificación, se empleó el algoritmo de *Random forest*, realizándose diversos experimentos según la configuración.

Para la evaluación, se empleó validación cruzada, empleando los datos de un sujeto para *test*, y los demás para *train*. Comprobando los resultados obtenidos para cada tipo de dato (acelerómetro, cuaternio, giroscopio, magnetómetro, y todos los anteriores). Consiguiendo unos resultados, con la mejor combinación (con todos los datos, y truncamiento de: *Accuracy* 99.4% y *F-Measure* 0.993.

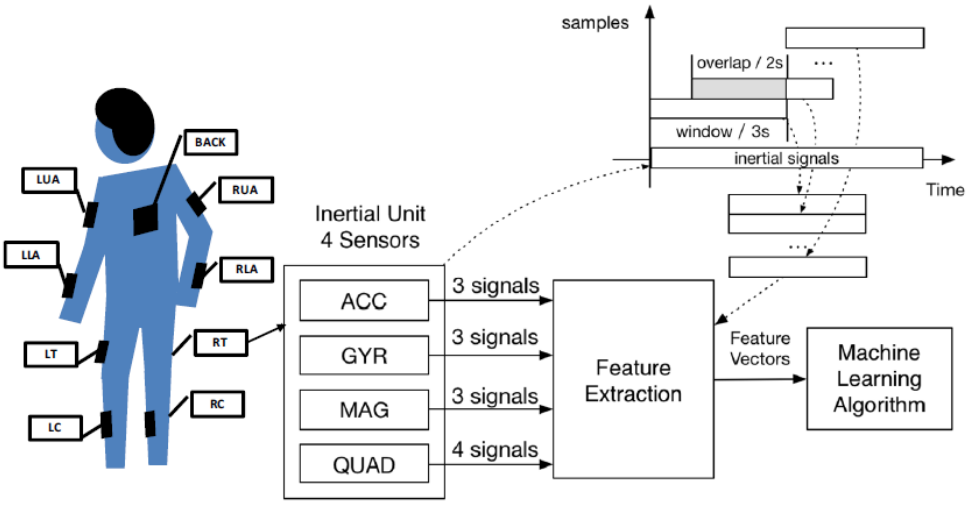

Figura 20 Diagrama de extracción de características [40].

2.2.7. Extremely randomized trees

En el artículo de conferencia de Uddin y Uddiny [41] de 2015, se propone el empleo de árboles de decisión extremadamente aleatorios. Para la selección de características se emplea el algoritmo supervisado *Guided Random Forest* (GDF), que selecciona en base a los *scores* del *Random forest*, el mínimo número de características importantes a seleccionar.

Para el reconocimiento de la actividad realizada, se emplea *Extremely randomized trees* (Extra-Trees), de manera que todo el set de entrenamiento se emplea para la construcción de cada árbol. De forma que se seleccionan aleatoriamente K atributos y valores de corte para cada atributo. Tras lo cual se asocia un score a cada atributo, y aquellos que dan la máxima reducción de la varianza se utiliza para dividir el nodo.

Para el preprocesado se eliminaron artefactos, y se realizado la selección de características a emplear (por el GDF).

Se emplearon como bases de datos REALDISP [4], PAMAP2 [42], DSA [31] y ADL [43].

Para la evaluación, se realizó una comparación con SVM y KNN, y con todas las características de REALDISP, o solo con las seleccionadas por GSR, siendo el *accuracy* máximo obtenido con *Extra-Trees* de 99.6%.

2.2.8. Adabost

En el artículo de Subasi et al. [44] de 2018, se propone el empleo de Adabost para la clasificación, de forma que la construcción de cada modelo se ve afectada de forma iterativa por la de los modelos anteriores.

En total para la comprobación de este algoritmo se emplearon 10 clasificadores diferentes y con 7 actividades de REALDISP. Los mejores resultados se obtuvieron con Adaboost y Random Forest, con un *accuracy* de 99.98% y un *F-Measure* de 1.

2.2.9. Triaxial Accelerometer

En el artículo de conferencia de Mimouna et al. [45], se dividen los datos de los sensores empleando una ventana móvil y se extraen las características de cada ventana. Calculándose la entropía de cada señal, para distinguir sensores activos de inactivos (según el ejercicio realizado) y así seleccionar las señales más significativas para identificar la acción realizada. De forma que en caso de que la entropía sea 0, el vector de características extraído contiene la media de la señal, tras lo cual, la señal se normaliza y se clasifica mediante SVM. Estructura observable en la Figura 21.

Los autores emplearon tres bases de datos para la comprobación: UC Berkeley WARD Dataset, Berkeley MHAD y REALDISP. Empleando para cada caso una ventana deslizante de longitud fija de 6, 15 y 9 respectivamente, utilizando únicamente los datos de

acelerómetro (los sensores por separado) de cada base de datos, y extracción de características mediante transformada wavelet discreta.

Los resultados obtenidos con REALDISP en el mejor caso (con un filtro de media móvil a los datos) alcanzaron un accuracy del 90%.

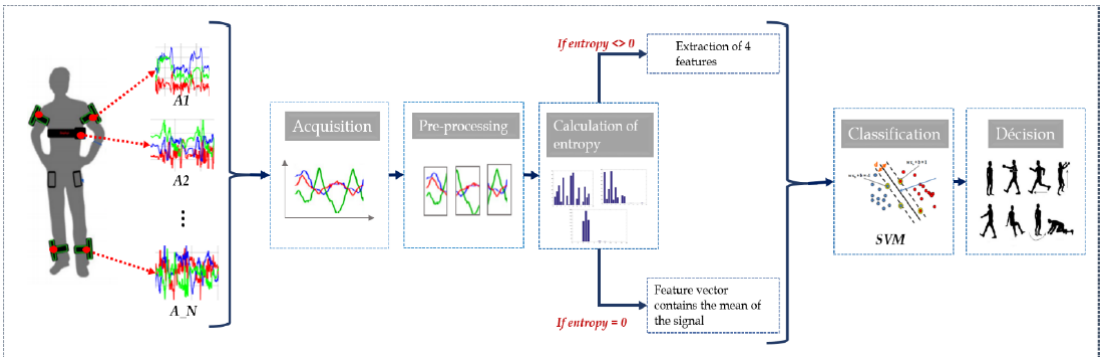

Figura 21 Arquitectura propuesta [45].

2.2.10. Wavelet

En el artículo de conferencia de Abdu-Aguye et al. [46], se pretende realizar un primer procesado de los datos mediante una transformada wavelet (da información temporal y de frecuencia), y posteriormente *pooling* adaptativo (*pooling* espacial piramidal) en la descomposición generada, obteniendo así una representación compacta y de longitud fija de los datos. De forma que se obtiene la transformada *wavelet* para cada canal de datos a distintos niveles, generándose vectores de coeficientes divididos en: detallados (componentes de alta frecuencia) y aproximados (baja frecuencia). A estos vectores se les emplea un operador *maxpooling* (para realizar *pooling* piramidal), con el que se obtiene una representación de tamaño fijo de cada vector, siendo posteriormente concatenados para obtener un único vector de características (estructura de la Figura 22). Empleándose para la clasificación *Random Forest*.

Para la evaluación se emplearon 5 *dataset*: Gomaa-1 [30], HAPT [24], *Daily and Sports Activities* [31], HAD-AW [32] y REALDISP, empleando únicamente acelerómetros y giroscopios para la clasificación, con un 75% de datos para train y 25% para test . Siendo el máximo resultado obtenido en REALDISP de 81.7% tanto en *accuracy* como en F1-score, con *pooling* 8-2-4-1 y 7 niveles de descomposición.

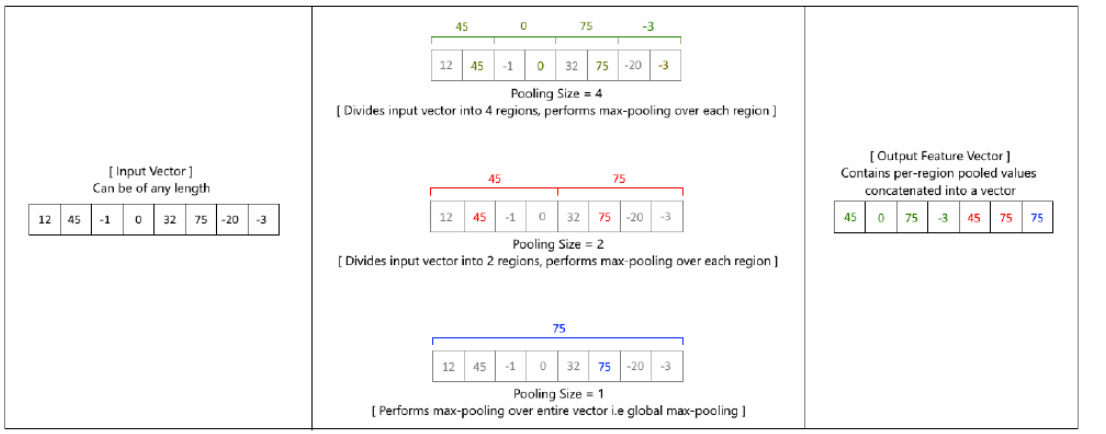

Figura 22 Ventanas propuestas [46].

2.2.11. Resumen REALDISP

A continuación, se incluye una tabla resumen de los trabajos comentados que emplean REALDISP en sus experimentos.

Tras una intensa búsqueda, se puede apreciar que, si bien existen trabajos que empleen REALDISP como base de datos para investigar el problema de HAR, son muy pocos comparados con los que emplean otras bases más conocidas como OPORTUNITY. Cabe destacar que los mejores resultados con REALDISP no se obtuvieron con técnicas de aprendizaje profundo, sino con otras técnicas de aprendizaje automático clásico como árboles de decisión y Adaboost.

Tabla 3 Resumen artículos HAR REALDISP

Método	Preprocesado	Datos empleados	clases	Método de clasificación	Accuracy	F1-score
VersaTL	Convolución y *pooling* piramidal	Acelerómetros y giroscopios	33	Redes neuronales recurrentes de tres capas	77.4%	NE
FE-AT	*Random oversampling*. Ventanas de 5s y solapamiento del 50%	Todos	33	Combinación de FE (basado en características) y AT (basado en atributos)	NE	82%
Adaptative pooling	Ventanas de tamaño fijo de 1s, empleando el 75% de los datos para *train*	Acelerómetros y giroscopios	33	*Pooling* adaptativo con CNN	94.14%	NE
SMART	Ventana deslizante de 1s	Todos	8	Autoencoder con dos LSTM, KNN y DBSCAN	NE	0.85
SMLDIST	Ventanas de 3 segundos	Todos	33	SMLDist (*Stage-Memory-Logits distillation*) con SIMU (sistema auto adaptativo)	94%	93.79%
FE-Random Forest	Agrupan las señales inerciales en frames (con ventanas de 3s) y calculo un vector de características	Todos	33	*Random Forest*	99.4%	99.3%
Extremely randomized trees	*Guided Random Forest* (GDF)	Todos o los seleccionados por GDF	33	*Extremely randomized trees*	99.6%	NE
Adaboost	Adaboost	Todos	33	*Random Forest*	99.98%	100%
Triaxial Accelerometer	Ventana deslizante de tamaño 9 y calculo entrópico	Acelerómetro	33	SVM	90%	NE
Wavelet	Transformada *Wavelet* y *Pooling* adaptativo	Acelerómetro y giroscopio	33	*Random Forest*	81.7%	81.7%

3. Estudio comparativo de redes de aprendizaje profundo

Durante el presente apartado, se tratará de detallar el proceso de desarrollo llevado a cabo para la realización del presente trabajo.

Dado que el objetivo principal del trabajo consiste en la comprobación del reconocimiento de actividades en REALDISP mediante diversas redes encontradas en la bibliografía actual, una parte importante en el desarrollo del trabajo ha sido la recopilación de información sobre el desarrollo actual de aprendizaje automático en *HAR*, y concretamente en el *dataset* REALDISP, tal y como se pudo comprobar en el capítulo anterior. De tal forma que se dispusiera de una base documentada, sobre la que comenzar a trabajar, empleando las redes propuestas en la literatura (o una versión de estas), frente al problema de *HAR* en REALDISP.

A su vez, dado que el trabajo es una ampliación del llevado a cabo por Sáez Bombín [1], empleando el *framework* desarrollado por Pardo-Villalibre [2], se requirió, al comienzo del trabajo de una familiarización inicial con el entorno de Pardo-Villalibre. De forma que se pudiera emplear este en las diversas comprobaciones realizadas en el trabajo.

De cara a las fases iniciales del trabajo, también cabe destacar el estudio de las conclusiones del trabajo de Sáez Bombín, en cuanto a la estructura de empleo del dataset. De cara a tener una base inicial de la que partir con la base de datos, y a su vez poder comparar los resultados obtenidos con el trabajo de Sáez Bombín.

Siendo por tanto imprescindible para comprender el inicio del trabajo realizado, de un apartado que defina los parámetros iniciales del trabajo planteado, tanto respecto al empleo del framework y las modificaciones realizadas al mismo, como respecto al punto de partida, con la configuración empleada en [1] .

Finalmente, una vez comentados ambos puntos introductorios, se procederá al desarrollo propio del trabajo, centrándonos en la comprobación de las preguntas de investigación planteadas originalmente:

- **¿Es posible encontrar en la literatura actual una arquitectura de red que permita obtener unos resultados iguales o superiores a los obtenidos con la red implementada por Sáez Bombín en su TFM?**

- **¿Son los cuaternios los mejores datos para trabajar con la base de datos REALDISP?**

Además de otras preguntas surgidas durante el desarrollo del trabajo. Buscando, por tanto, la verificación de las hipótesis planteadas, mediante diversas comprobaciones y con la constatación de los resultados obtenidos.

3.1. Metodología y fases de desarrollo

En la primera fase del trabajo, se realizó una búsqueda de información sobre el problema del *HAR*, bases de datos empleadas, soluciones propuestas, etc. De cara a conocer el tema en el que se pretende trabajar, y cuáles son los últimos avances propuestos por la comunidad científica. Empleando para la búsqueda de información, las herramientas de Google Académico y del IEEE, buscando entre otros criterios *HAR*, REALDISP, y aprendizaje automático entre otros.

En la segunda fase del trabajo fue fundamental aprender a utilizar de manera óptima, el *framework* desarrollado por Pardo-Villalibre [2] y sus diferentes opciones de configuración. Fue de especialidad utilidad la guía de uso elaborada por el autor, si bien hay que tener en cuenta que no se utilizó una versión final del *framework*, ya que el mismo ha continuado desarrollándose.

En la tercera fase del trabajo se seleccionaron de entre los artículos encontrados en la literatura los más viables para replicar y/o modificar, en un intento de encontrar una arquitectura de red con un desempeño (*Accuracy* y F1-SCORE) superior, o en su defecto similar al que se obtendría con la red propuesta por Sáez Bombín [1]. Algunas de las redes seleccionadas en la literatura son las comentadas en el capítulo 2 de este trabajo. Siendo las redes: CNN2C, CNN color, LSTM-CNN, SMART las seleccionadas.

Los resultados empleando la red de Sáez Bombín, serán los obtenidos con la configuración descrita en su TFM, pero empleando el *framework* de Pardo-Villalibre. Por este motivo, los resultados pueden variar respecto a los descritos en el TFM original.

3.2. Condiciones iniciales

Para la realización de las comprobaciones y pruebas, se optó por el empleo de un ordenador del Grupo de Telemática e Imagen (GTI) de la universidad de Valladolid, con el sistema operativo Linux (Ubuntu) instalado, y con el programa Visual Studio Code para el empleo del *framework* [2] (clonado localmente del GitHub del mismo). Disponiendo el ordenador de una gráfica GTX2080 TI de Nvidia.

Con el equipo anteriormente descrito, se comenzó a trabajar con el *framework*, realizando comprobaciones sencillas, para la familiarización con el mismo, dada la amplia variedad de opciones de las que dispone. Pruebas tales como el tuneado del *dataset* (con cuaternios, aceleraciones, etc), comprobación del funcionamiento de la etapa de preprocesado (*Data Augmentation*, STFT, etc..) y la etapa de entrenamiento (selección de redes a entrenar, configuración de hiperparametros, 10k-fold, etc).

Una vez realizada esta familiarización previa con el entorno de trabajo, se comenzó la fase de pruebas propuestas con el entorno, realizando modificaciones al mismo, para poder trabajar con nuevas redes de entrenamiento propuestas (descritas en el apartado posterior), y con distintos datos del dataset REALDISP.

Para la configuración inicial del sistema, se optó por el empleo de la configuración inicial (con la salvedad de hiperparametros y elementos del dataset) encontrada en [1] como la más conveniente para trabajar con REALDISP: data augmentation de los cuaternios (180º), transformada rápida de Fourier (FFT) de los datos, y datos de entrada sin normalización.

De esta forma, dados los resultados obtenidos en [1], en el desarrollo del presente trabajo se optó por no pre-procesar los datos y emplear una ventana deslizante de tamaño 128 y con 75% de solapamiento. Así como por el empleó del mismo subconjunto de actividades del dataset, que fueron seleccionadas en [1], en cuanto a que empleaban la parte superior del cuerpo.

A9: Giro de tronco (brazos extendidos), A10: Giro de tronco (codos doblados), A11: Flexión de cintura hacia delante, A12: Rotación de la cintura, A13: Flexiones de cintura (alcanzar el pie con la mano contraria), A19: Elevación lateral de brazos, A20: Elevación frontal de brazos, A21: Palmas frontales, A22: Cruce frontal de brazos, A24: Rotación de los hombros de baja amplitud, A25: Rotación interna de los brazos, A31: Remo.

Los sensores utilizados fueron consecuentemente, los mismos que en el caso del TFM anterior y correspondientes a los que recogían datos de la parte superior del cuerpo (RUA: *Right Upper Arm*, RLA: *Right Lower Arm*, LUA: *Left Upper Arm*, LLA: *Right Lower Arm* y *BACK*).

Por último, se emplearon los mismos sujetos (1, 2, 3, 5, 8, 9, 10, 11, 13, 14, 16, 17), cuyo etiquetado fue revisado en el trabajo de Sáez Bombín.

Para poder realizar una mejor comparación se ha implementado en el framework la red con mejores resultados obtenidos en [1]: una red CNN-LSTM de tres capas convolucionales y una LSTM.

Por último, antes de poder comenzar las pruebas con las redes encontradas en la literatura, fue necesario realizar una serie de ajustes en el framework [2], con el fin de poder trabajar según los criterios propuestos con el dataset REALDISP. A continuación, se detallan los cambios fundamentales y las pruebas realizadas:

3.2.1. Tunning

En la parte de *tunning* o tuneado del *framework*, se trabajó de forma general con el ejecutable para REALDISP original. Modificándose únicamente, con el objetivo de obtener de los datos originales del dataset tanto las aceleraciones de cada sensor (mismos sensores empleados por defecto), como del giroscopio de estos, además de los cuaternios de la versión original. Pudiendo de esta forma ampliar los resultados obtenidos por Sáez Bombín, empleando diversos tipos de datos con las redes propuestas.

3.2.2. Preprocesado

En el apartado de preprocesado, se configuro el archivo *pipeline config*, para realizar el preprocesado de los datos procedentes del *tunning*. Para ello se empleó una de las plantillas de ejemplo de este, modificándose los apartados pertinentes del mismo, para adecuarse a las pruebas a realizar en el presente trabajo. Empleándose diversas configuraciones de este, según el objetivo deseado, aplicándose de forma general a los 12 sujetos mencionados en el apartado de condiciones iniciales. Así como las 11 actividades que empleaba Saéz-Bombín.

A su vez cabe mencionar, que es en esta parte del *framework*, en el que se lleva a cabo la técnica de ventana deslizante de los datos, así como la transformada de Fourier y *data augmentation* (para los cuaternios).

3.2.3. Entrenamiento

Finalmente, en la etapa de entrenamiento del *framework*, se requirió, por motivos que se detallarán posteriormente, el empleo durante la fase de entrenamiento de todos los datos del dataset. Permitiendo el *framework* por defecto seleccionar el tamaño del *batch* y el de *steps* de cada entrenamiento mediante *configs*, pero sin la posibilidad de que los *steps* se calculasen de forma automática en base al número de datos de entrada.

Por tanto, para subsanar este inconveniente, se realizaron modificaciones en los archivos train.py y utils.py del *framework*, aportando esta opción.

Asimismo, para cada tipo de entrenamiento se tuvieron que generar *configs* (hiperparametros), y redes (los empleados en el trabajo), pero ambos casos se detallarán posteriormente.

Finalmente, se procedió a corregir y modificar aquellos entrenamientos disponibles en el *framework* (por defecto), que generaban errores al ser ejecutados, destacando especialmente el ejemplo de la red de CNN y LSTM.

3.3. Desarrollo

Una vez introducidas las características iniciales, y el trabajo previo desarrollado, se procede a detallar el desarrollo del trabajo llevado a cabo en orden cronológico, para dar respuesta a las preguntas de investigación planteadas.

3.3.1. Selección de la red

La primera pregunta de investigación a responder en el trabajo es:

¿Es posible encontrar en la literatura actual una arquitectura de red que permita obtener unos resultados iguales o superiores a los obtenidos con la red implementada en el TFM anterior de Sáez Bombín?

La hipótesis inicial es que sí que es factible encontrar una red en la literatura (o variante de esta) igual o superior a la anterior. Para la comprobación de la hipótesis, partimos de las condiciones iniciales definidas previamente: *framework*, base de datos REALDISP, ventana deslizante de 128 y 75% de superposición, 12 sujetos, 11 actividades, 5 sensores, *data augmentation* (cuaternios), FFT y sin normalización de los datos de entrada. Los datos del *dataset* a emplear son los cuaternios. De forma que los datos de entrada a la red, una vez sido preprocesados por el *framework*, serán de tamaño 128 x 60.

Se añadieron al *framework* una serie de arquitecturas valoradas como interesantes para el presente trabajo, ya que habían sido empleadas para el problema del HAR, aunque no necesariamente sobre la base de datos REALDISP. Siendo en algunos casos copias directas de lo expresado en la literatura, y en otras modificaciones de estos de acuerdo con el criterio del autor, en aquellos casos en los que la arquitectura a realizar se encontrara únicamente parcialmente definida.

Las siete redes seleccionadas para las comprobaciones son los siguientes:

- **CNN+LSTM-Saez**

  La arquitectura de red propuesta por Sáez Bombín en su TFM [1], formada por tres capas convolucionales (64) con *maxpooling*, *dropout* (0.5) y una capa LSTM (64), así como *batch normalization* entre capas.

- **CNN+COLOR**

  Red CNN propuesta en [14], formada por 4 capas convolucionales de 32 filtros, *maxpooling* y *dropout* de 0,25.

- **CNN2C**

  Red descrita en [17], formada por 2 capas convolucionales de 64 filtros, *maxpooling* y 3 capas densas (64, 32 y 12)

- **LSTM**

  Red descrita en [12], formada por 4 capas LSTM simples (32 neuronas), *dropout* de 0.5 y una capa densa.

- **LSTM+CNN**

  Red propuesta en [28], con dos capas LSTM (32) y dos convoluciones (64 y 128), con *global averaging*, *batch normalization* y capa densa al final.

- **LSTMAE**

  Una versión de la arquitectura propuesta en [35]. Empleando únicamente la parte de autoencoder LSTM (sin la parte de no supervisado original) con modificaciones propias dada la falta de información específica sobre la red en el artículo. Siendo la estructura propuesta un encoder-decoder de 2 capas LSTM de 256 y 128, *dropout* 0,5 y *batch normalization* (2 para el encoder y 2 para el decoder), con una capa densa al final.

- **CNN+LSTMAE**

  En este caso, estamos ante una versión alternativa de la anterior, con el añadido de una red CNN anterior al autoencoder, con una estructura similar a la red descrita en [1] (tres capas convolucionales de 64 con *maxpooling* y *dropout*) previo al auto-encoder LSTMAE.

Para una mejor comprensión de las redes anteriormente descritas, se adjuntan a continuación esquemas orientativos de las estructuras de las redes, obtenidos mediante el empleo del programa Netron. Así mismo, en los anexos se incluyen las tablas de *summary* generadas por Keras en cada red, para el caso en el que colocamos únicamente cuaternios a la entrada de la red.

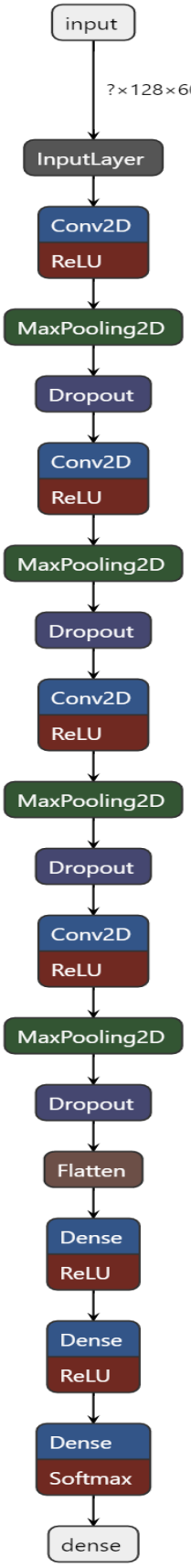

Figura 24 CNN+COLOR.

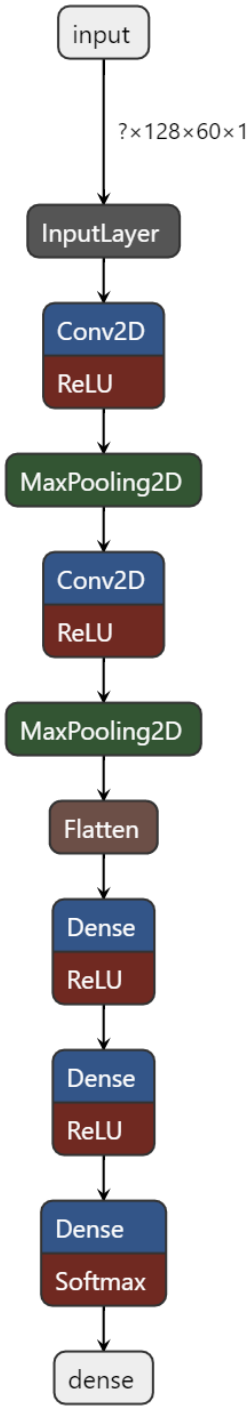

Figura 23 CNN2C.

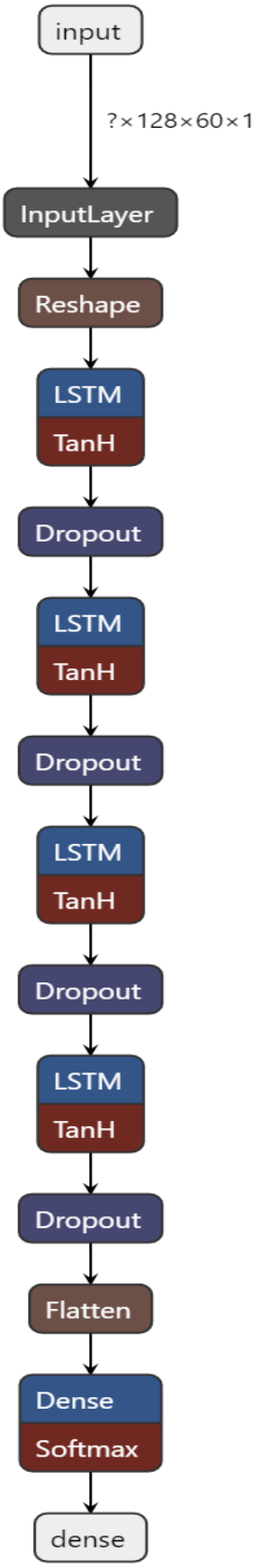

Figura 25 LSTM.

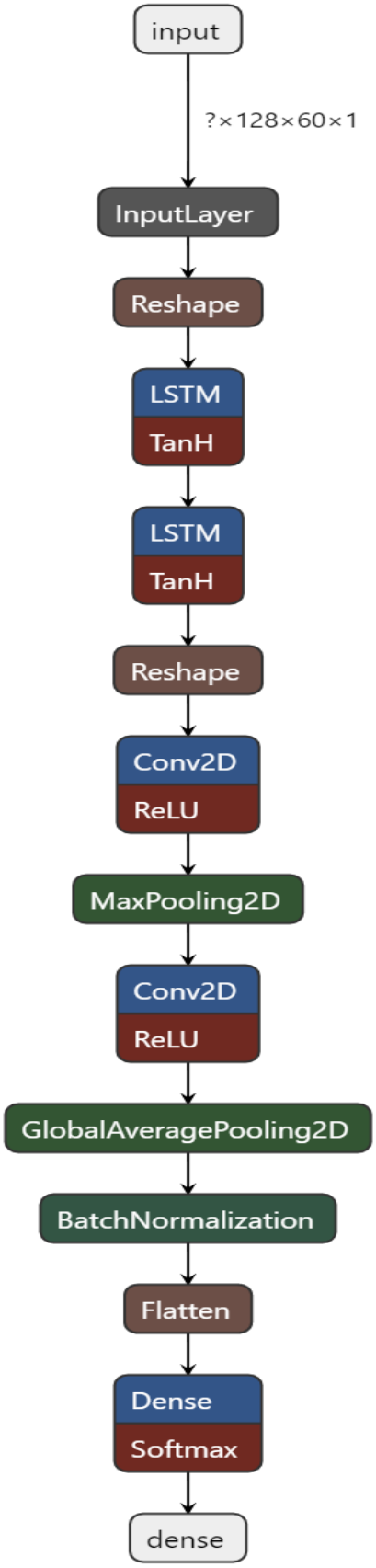

Figura 26 LSTM+CNN.

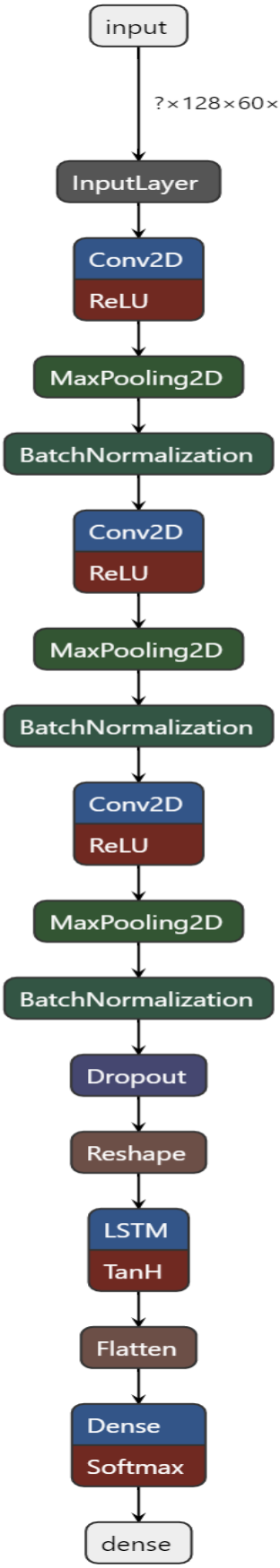

Figura 28 CNN+LSTM-Saez.

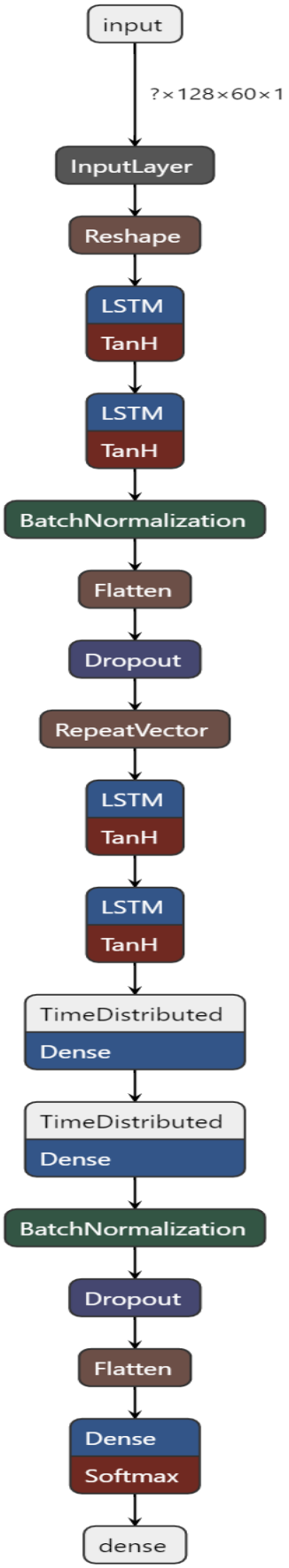

Figura 27 LSTMAE.

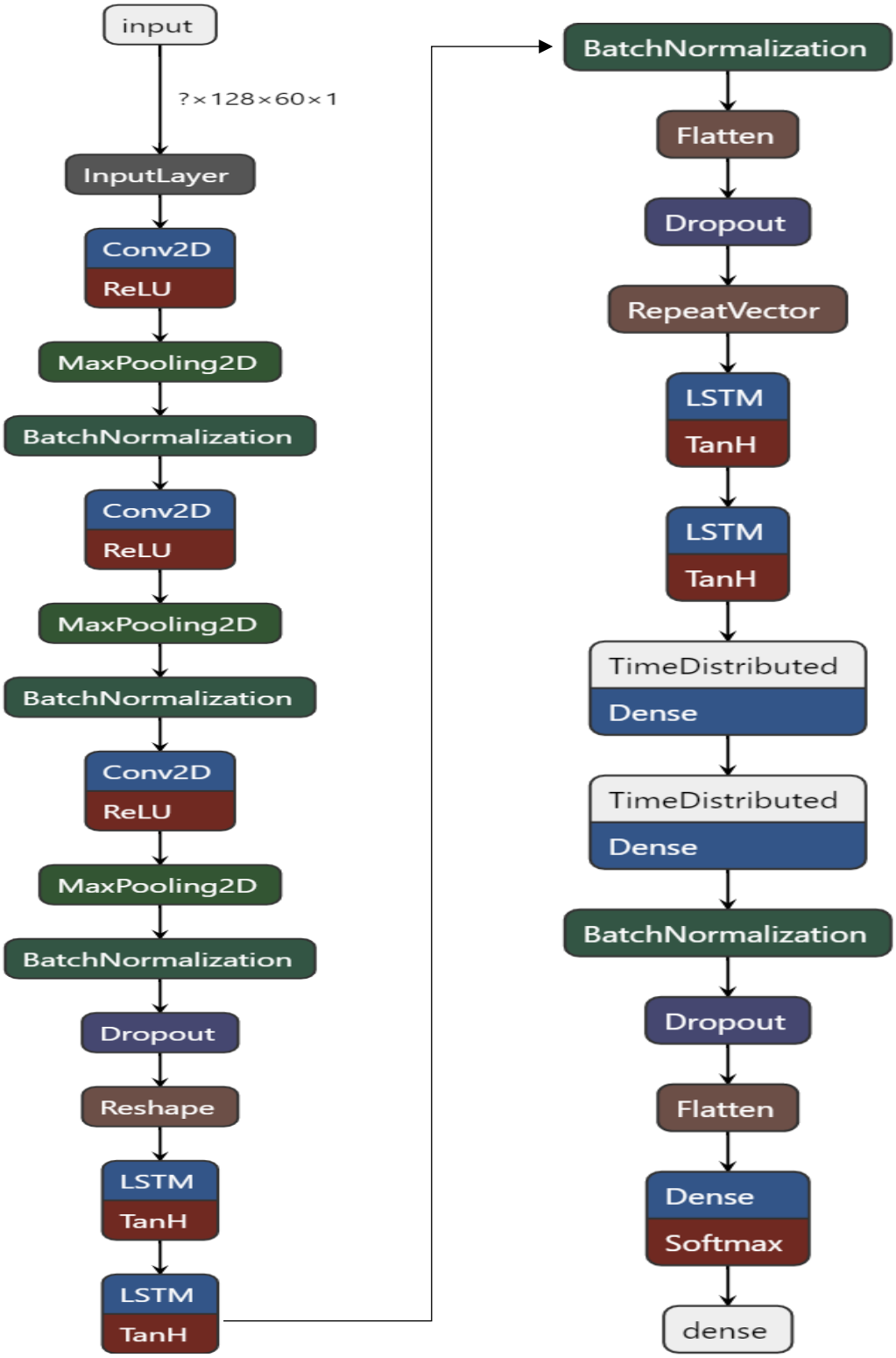

Figura 29 CNN+LSTMAE.

.

Para la comprobación de los resultados obtenidos con las diferentes arquitecturas, se optó por el empleo de 10k-fold no estricto (mismo sujeto de *test* y *validación*). Empleando como tamaños de *batch* 15 y 30, 100 *epochs*, *early stopping* de 12 (para evitar *overfitting*) y *modelCheckPoint accuracy* (la de entrenamiento y no la de validación), con un step size de 40, es decir no se emplean todos los datos en el entrenamiento (por problemas con el pc). Los resultados a destacar, serán los valores obtenidos de F1-score y de *Accuracy* de cada red, con la configuración de 10k-fold anterior.

En la **¡Error! No se encuentra el origen de la referencia.** se pueden apreciar los resultados obtenidos, empleando como dato los cuaternios, y con tamaños de batch 15 y 30.

Tabla 4 Resultados 10k-fold con batch 15 y batch 30 para cuaternios

	batch 15		batch 30	
Red	Accuracy	F1-score	Accuracy	F1-score
CNN+COLOR	87%	0,02	93%	0,56
CNN2C	98%	0,85	98%	0,86
LSTM	92%	0,47	97%	0,5
LSTM+CNN	91%	0,52	91%	0,48
CNN+LSTM-Saez	98%	0,85	98%	0,88
LSTMAE	95%	0,72	96%	0,74
CNN+LSTMAE	98%	0,89	98%	0,86

Con los resultados previos, podemos comprobar como resaltados en amarillo, los mejores resultados obtenidos son, con *batch* 15 para CNN+LSTMAE con un F1 de 0.89, y con *batch* 30 para la red de CNN+LSTM-Saez, con un F1 de 0.88, muy buenos resultados en ambos casos, pero ligeramente superiores para la red propuesta CNN+LSTMAE.

Para realizar una mejor comparación de los resultados, se adjuntan a continuación las tablas de métricas resultado del 10k-fold del *framework*, Figura 30 y Figura 31.

En estas figuras podemos apreciar más claramente, que si bien ambas redes, obtienen resultados generales bastante similares, la de CNN+LSTMAE para batch 15, es ligeramente superior a la de Sáez Bombín (más azul oscuro).

Por todo ello, tras estas comprobaciones podemos concluir que:

- Las redes LSTM con AE dan buenos resultados, especialmente la de CNN+LSTMAE para cuaternios.
- Para *batch* 15 CNN+LSTMAE supera a la red de Sáez Bombín.
- Requerimiento de más pruebas para confirmar la hipótesis inicial.

Así pues, con todo lo anterior, se procede a plantear y comprobar la segunda pregunta de investigación del trabajo. De forma que las comprobaciones de está sirvan a su vez como comprobaciones adicionales de la primera pregunta de investigación.

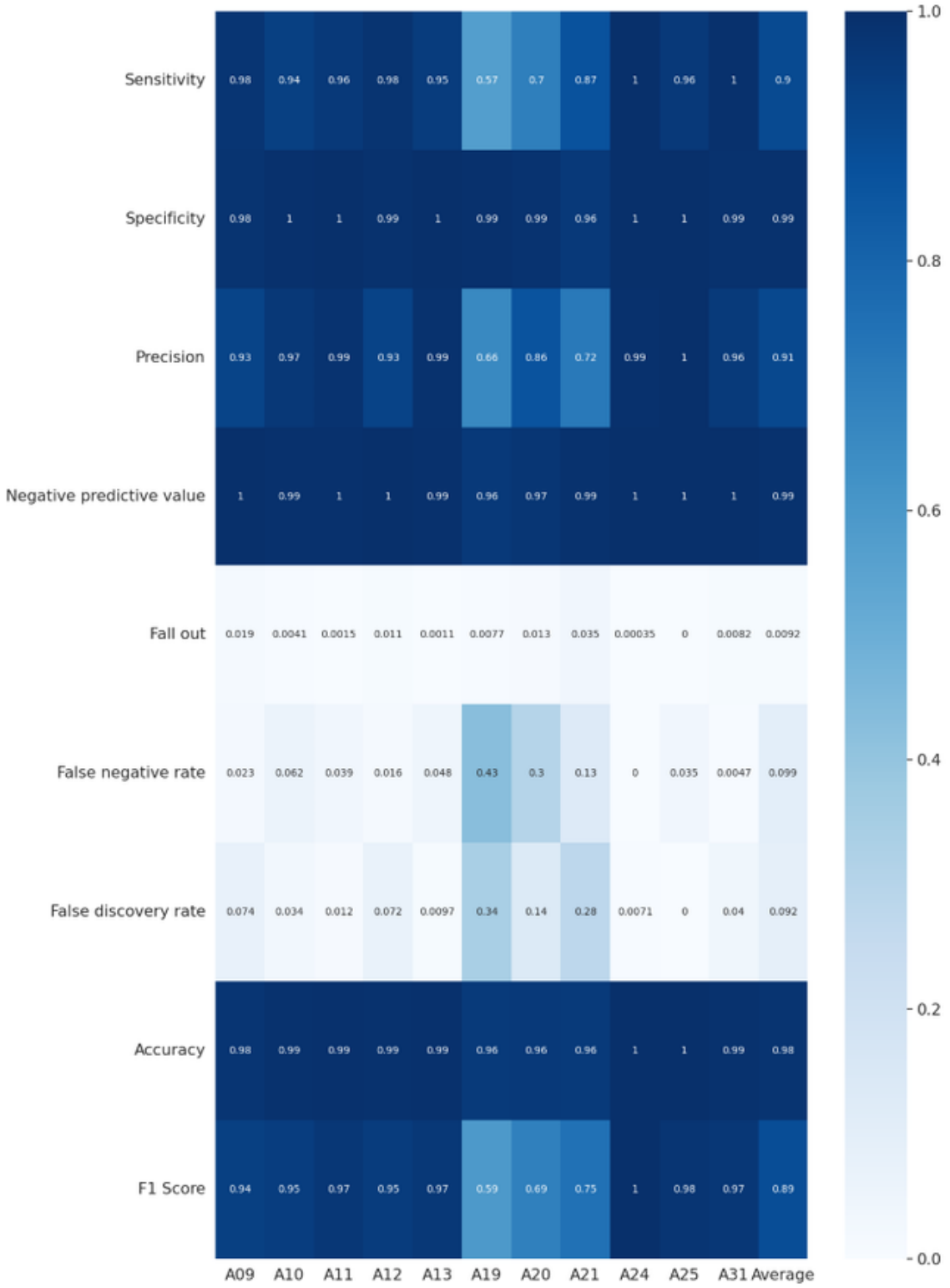

Figura 30 Resultados con batch 15, CNN+LSTMAE

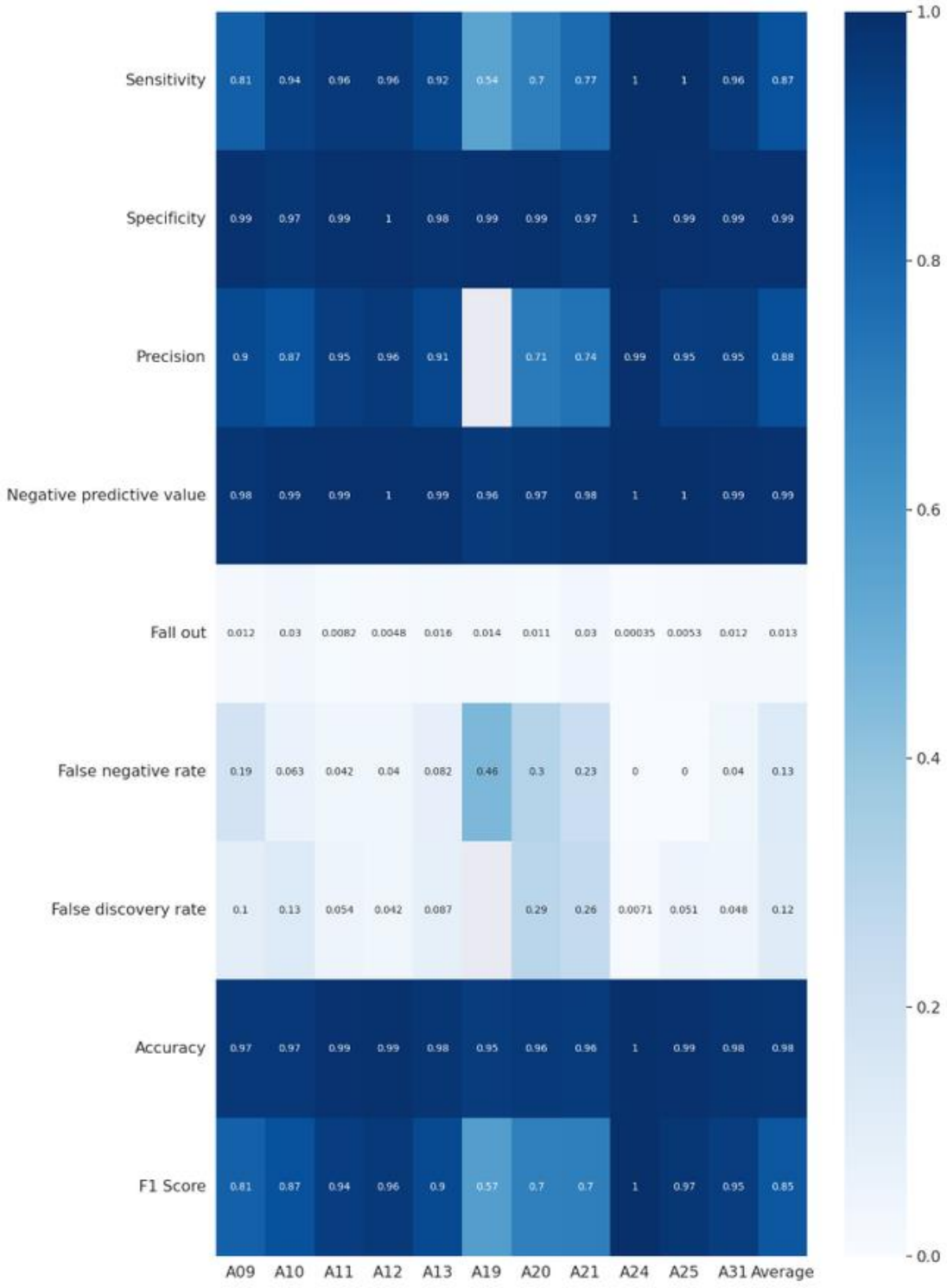

Figura 31 Resultados con batch 15, CNN+LSTM-Saez.

3.3.2. Selección de los datos

La segunda pregunta de investigación planteada es:

¿Son los cuaternios los mejores datos para trabajar con REALDISP?

Siendo la hipótesis inicial que **sí que hay datos mejores o complementarios a los cuaternios en REALDISP.**

Para la comprobación de esta hipótesis, y como comprobación adicional de la primera pregunta de investigación, se realizará un entrenamiento similar al anterior (*framework*, 10k-fold, *batch* 15 y 30, redes, etc…), pero con diferentes tipos de datos de entrada: Aceleraciones (45 columnas), Giroscopios (45), Aceleraciones + Cuaternios (105) y Aceleraciones + Giroscopios (90).

A continuación, se añade una tabla resumen con las arquitecturas de red anteriormente comentadas, y con los resultados obtenidos según los datos empleados (aceleración, cuaternios o giroscopio, y aceleración + cuaternios), así como según el tamaño del *batch* contemplado (15 o 30), con ventana deslizante de 128 muestras y 75% de solapamiento, así como *Early stopping* de 12 y 100 epochs. Encontrando resaltados en amarillo los mejores resultados en cada caso (al igual que en el entrenamiento previo).

Tabla 5 Resultados obtenidos con batch 30 y 15

Red	Batch	Aceleración		Giroscopio		Cuaternios		AC + Cuaternios		AC + GIR	
		Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
CNN + COLOR	15	84%	0.02	84%	0.02	87%	0.02	84%	0.02	84%	0.02
	30*	84%	0.02	84%	0.02	93%	0.56	84%	0.02	84%	0.02
CNN2C	15	98%	0.88	99%	0.92	98%	0.85	98%	0.9	99%	0.94
	30*	99%	0.93	98%	0.9	98%	0.86	99%	0.92	99%	0.94
LSTM	15	97%	0.73	96%	0.77	92%	0.47	97%	0.75	97%	0.79
	30	98%	0.87	98%	0.86	91%	0.47	97%	0.82	98%	0.83
LSTM +CNN	15	99%	0.93	98%	0.9	91%	0.52	99%	0.9	99%	0.91
	30	99%	0.9	98%	0.88	92%	0.49	98%	0.88	99%	0.9
CNN+ LSTM-Saez	15	99%	0.92	98%	0.88	98%	0.85	98%	0.9	98%	0.9
	30*	98%	0.9	99%	0.91	98%	0.88	99%	0.91	99%	0.93
LSTM AE	15	99%	0.93	99%	0.93	95%	0.72	99%	0.91	99%	0.95
	30	99%	0.93	99%	0.93	95%	0.71	99%	0.94	99%	0.96
CNN+ LSTM AE	15	99%	0.92	98%	0.88	98%	0.89	99%	0.93	99%	0.94
	30*	99%	0.96	99%	0.95	98%	0.86	99%	0.92	99%	0.92

*Por problemas con el pc empleado, el tamaño del batch empleado con Aceleración + Cuaternios se ha visto reducido al mínimo que no diese problemas el entrenamiento (CNN+COLOR:23, CNN2C:25, CNN+LSTM-Saez:24, CNN+LSTMAE: 23). Empleando por problemas similares con Aceleración + Giróscopo: CNN+COLOR, CNN2C y CNN + LSTMAE:22 y para CNN+LSTM-Saez 24.

A partir de los resultados anteriores, podemos concluir que como era de esperar, al añadir emplear otros datos diferentes a los cuaternios, los resultados obtenidos mejoran, obteniendo valores muy elevados tanto de *Accuracy* como del F1-SCORE, destacando especialmente la red de Sáez Bombín, la de LSTMAE, la de CNN2C y la de CNN+LSTMAE, siendo esta última la que ha obtenido mejores resultados en la mayoría de los casos comprobados, con los mejores resultados empleando solo la aceleración y con *batch* 30.

Hay que tener en cuenta que para el caso de Aceleración + Cuaternios, por problemas con el PC, no se pudo realizar la prueba de LSTMAE + CNN con *batch* 30, y si con LSTMAE, lo que podría implicar que posiblemente, a mismas condiciones, LSTMAE + CNN siguiese siendo superior.

De nuevo como en el caso anterior, se adjuntan diversas tablas de métricas obtenidas en el entrenamiento.

Con los resultados obtenidos en esta segunda comprobación, podemos concluir que:

- Los mejores resultados son los dados por LSTMAE y CNN+LSTMAE, por encima de la red de Sáez Bombín (primera hipótesis).
- Los datos con valores superiores son los de aceleraciones, y aceleraciones + giroscopio.
- En relación a lo anterior, los cuaternios como se esperaba (hipótesis) no son los mejores datos para este caso, ya que son las aceleraciones.

Con lo que resumiendo todo lo anterior, contestando a la primera pregunta, sí que existen redes derivadas de las existentes en la literatura que superan a la propuesta por Sáez Bombín en su TFM (primera hipótesis), y contestando a la segunda pregunta, las aceleraciones dan mejores resultados que los cuaternios como datos de entrada, y por tanto estos no son los mejores datos para trabajar con REALDISP.

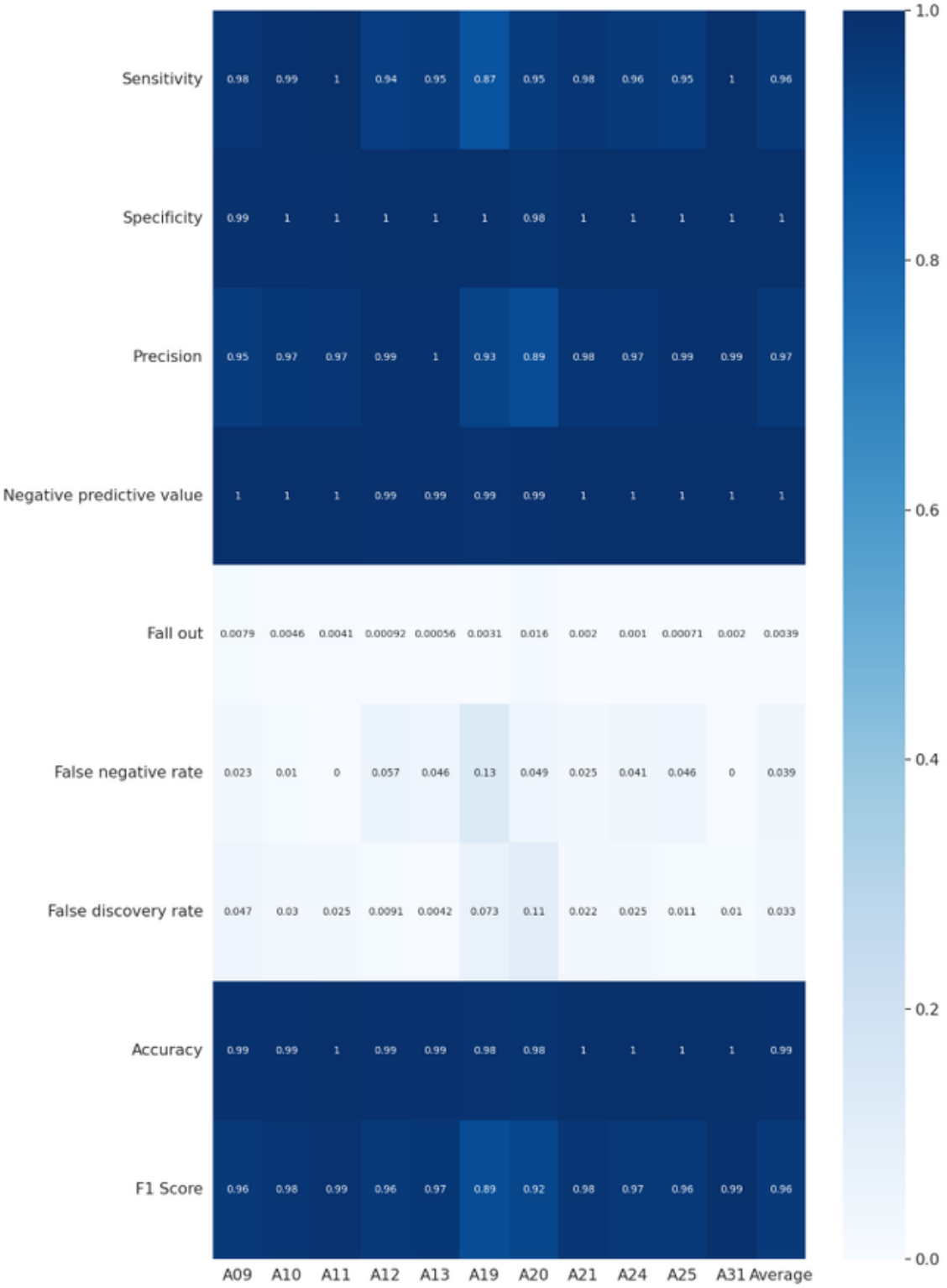

Figura 32 Resultados con aceleración, batch 30 y CNN+LSTMAE

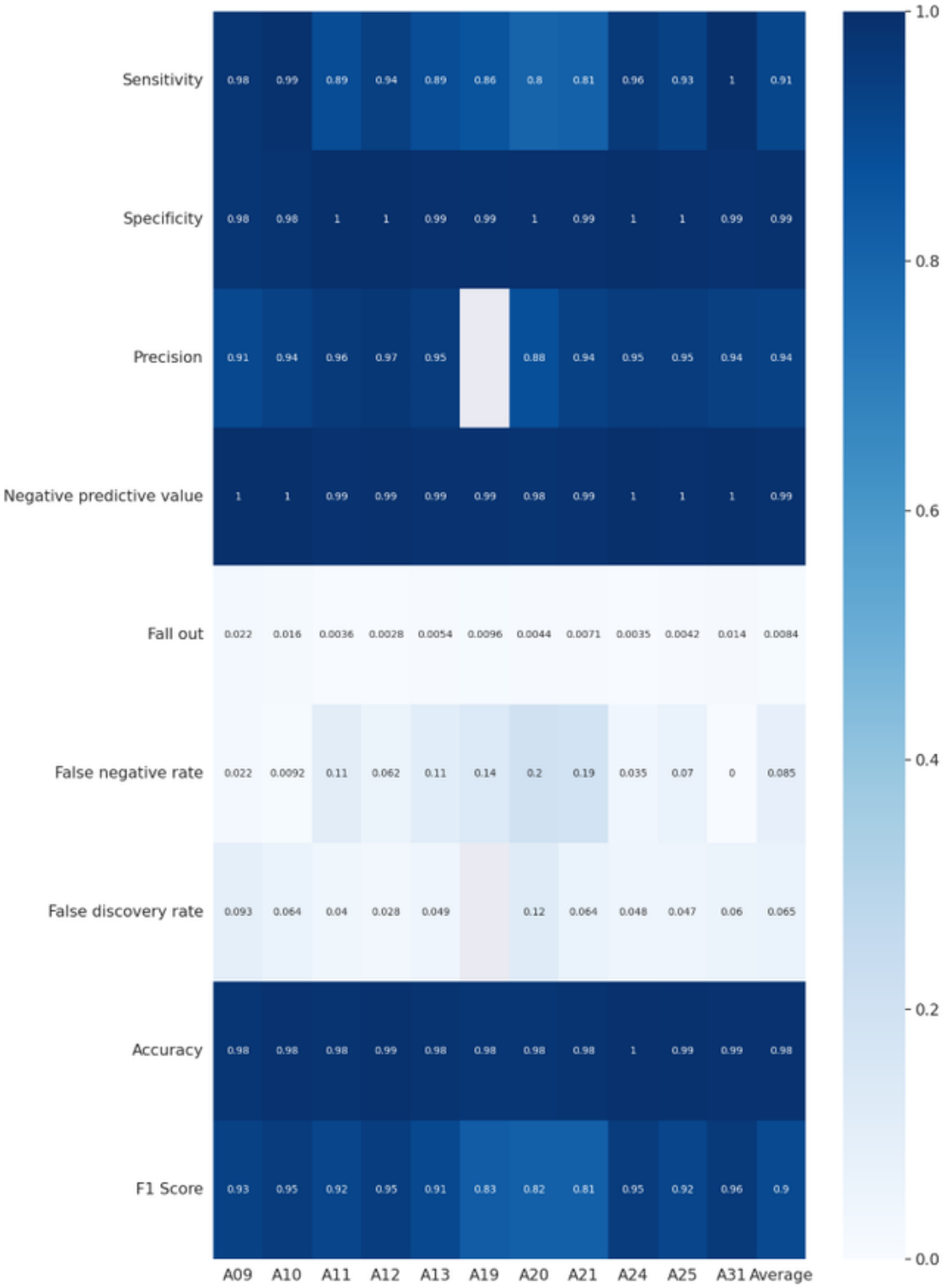

Figura 33 Resultados con aceleración, batch 30 y CNN+LSTM-Saez

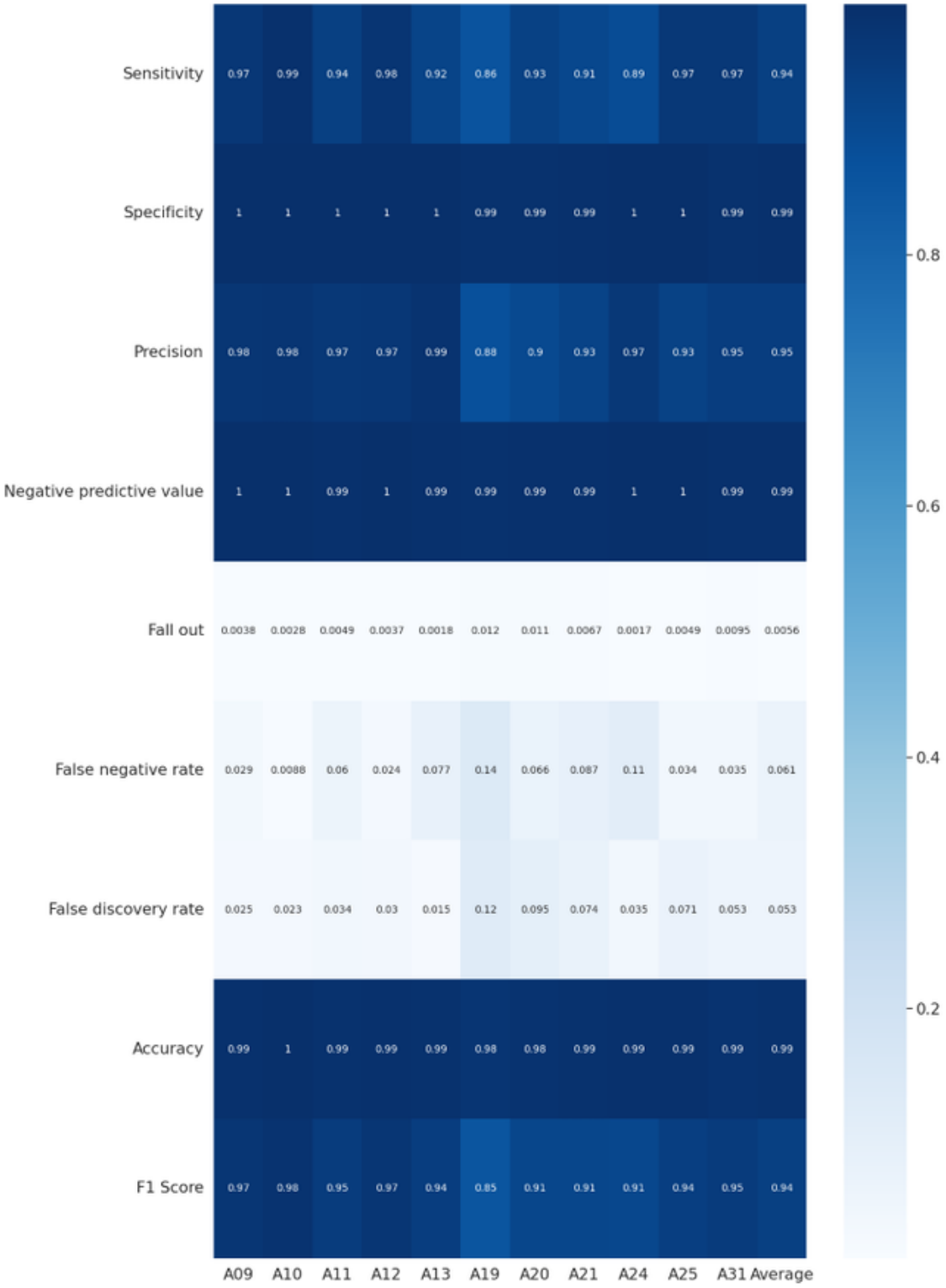

Figura 34 Resultados con Ac + Cuaternios, batch 30, LSTMAE

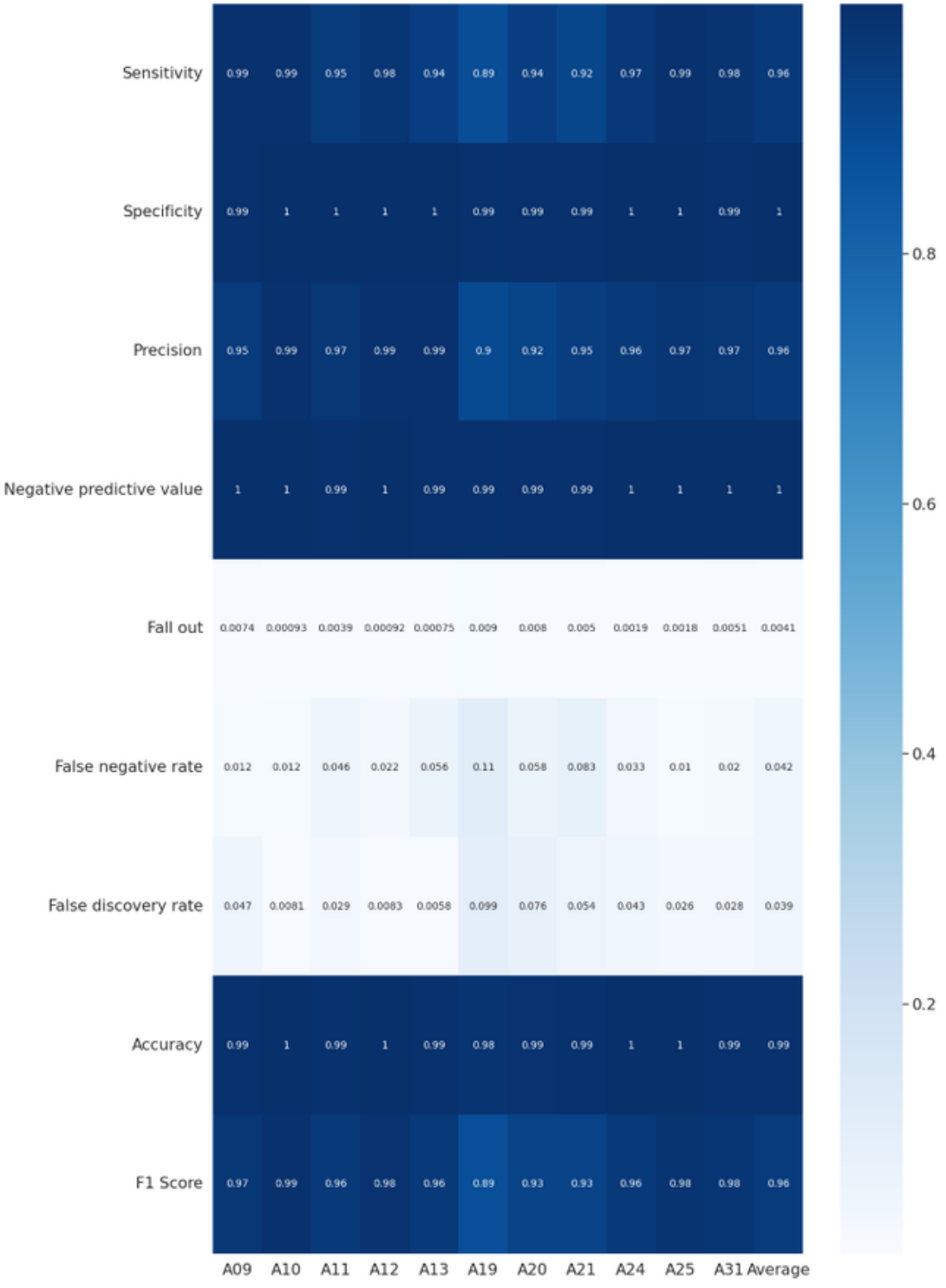

Figura 35 Resultados con Ac + Giroscopio, batch 30, LSTMAE

3.3.3. Selección del óptimo

De los resultados anteriores obtenidos, la investigación nos lleva a plantearnos una nueva pregunta de investigación: **¿Podemos encontrar otros tamaños de batch y número de epochs de entrenamiento para mejorar los resultados de las redes que obtenían los mejores resultados en los experimentos anteriores?**

Teniendo de nuevo una hipótesis inicial afirmativa: **Es posible encontrar parámetros con resultados superiores a las comprobaciones previas.**

Para esta comprobación, se requiere de dos pasos sucesivos de entrenamiento:

1) Entrenamiento con distintos *batch* (entre 4 y 26 para no tener problemas), y *early stopping* 12 (val_los en este caso), ya que se emplean 8 sujetos para *train*, 2 para validación y 2 para *test* (siempre los mismos para *train*, validación y *test*, pero seleccionados por primera vez aleatoriamente).

Se realiza el entrenamiento con las redes LSTMAE y CNN+LSTMAE (las que mejores resultados dieron) para cada tipo de dato (cuaternios, aceleración, giroscopio, aceleración + cuaternios y aceleración + giroscopio), buscando el *batch* y *epoch* óptimo en cada caso.

Se emplea Tensorboard (Tensorflow), para visualizar las gráficas seleccionadas como mejores (con mayor resultado de *accuracy* y F1), y con unas curvas más claras, seleccionando manualmente los *epoch* óptimos (buscando el *loss* mínimo sin *overfitting*). A continuación, encontramos algunas de estas gráficas obtenidas desde la Figura 36 a la Figura 41:

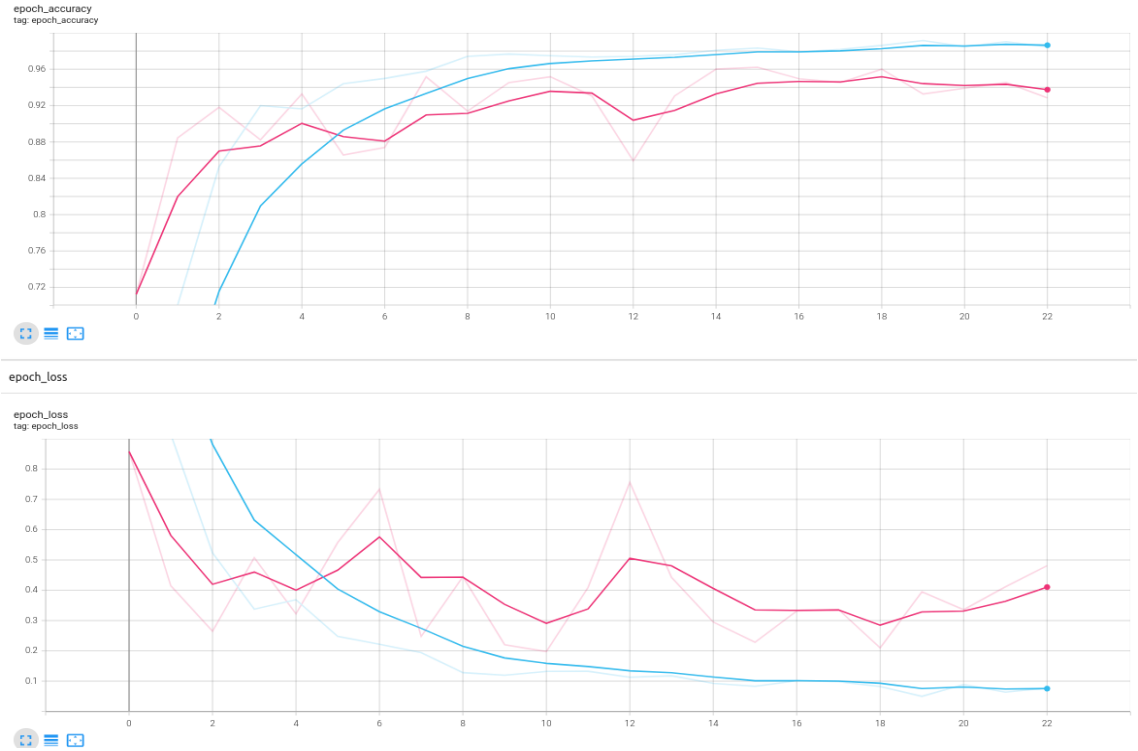

Figura 36 Curvas LSTMAE, AC+ Giroscopio, batch 14, epochs 18.

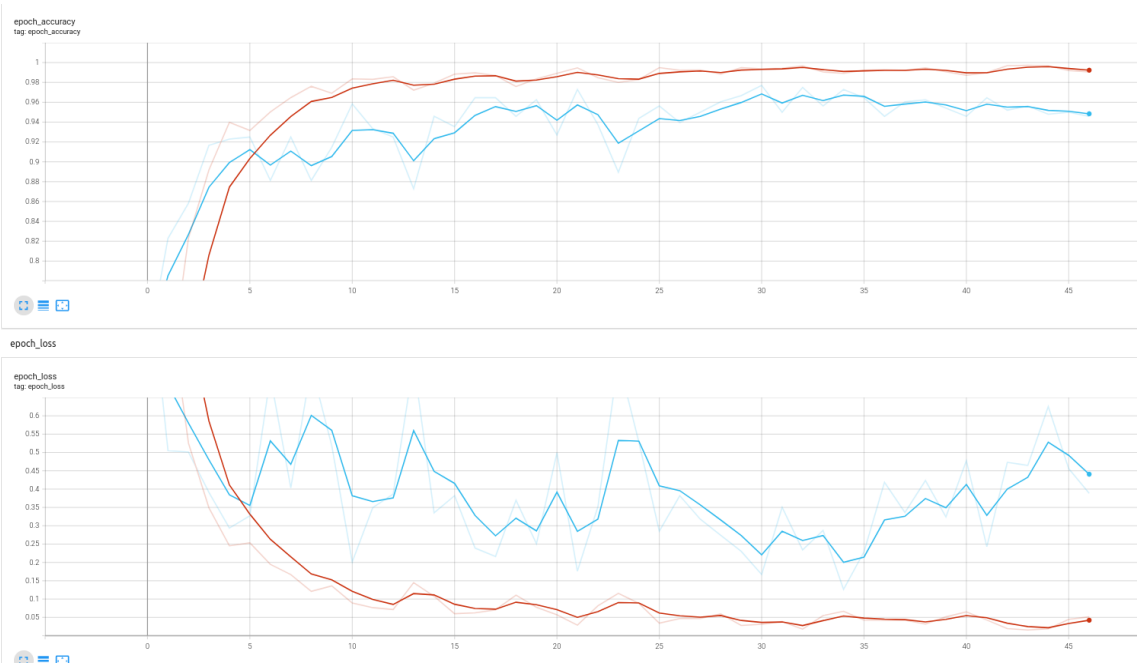

Figura 37 Curvas LSTMAE, Aceleración, batch 20, epochs 34.

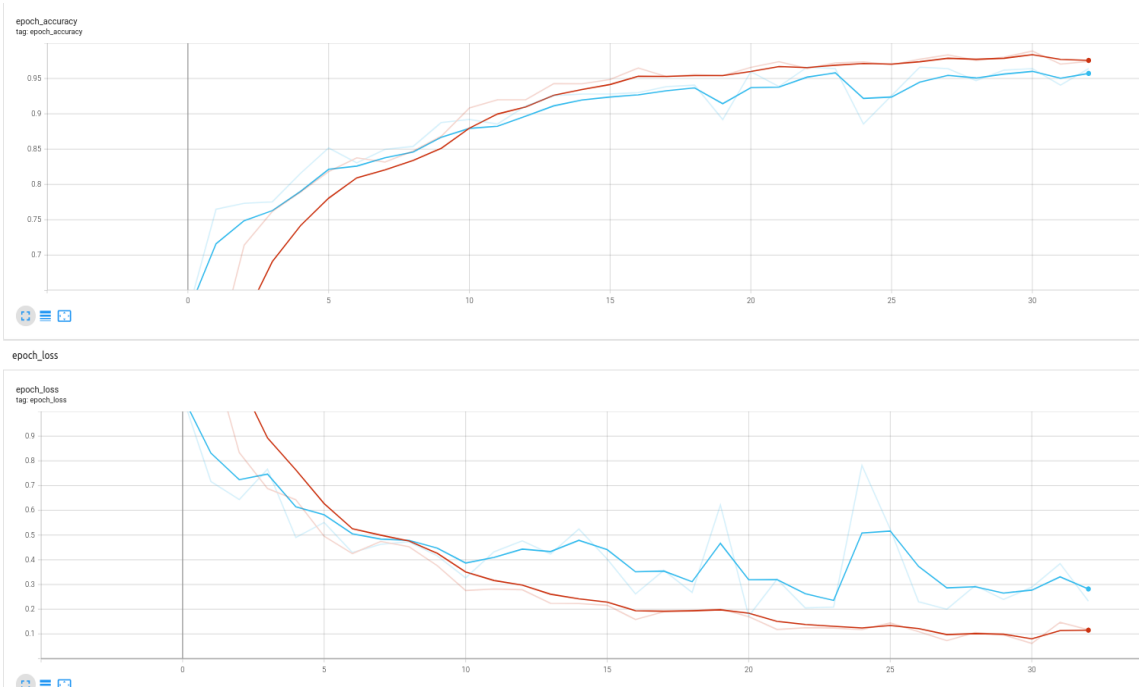

Figura 38 Curvas CNN+LSTMAE, Aceleración + Cuaternios, batch 8, epochs 30.

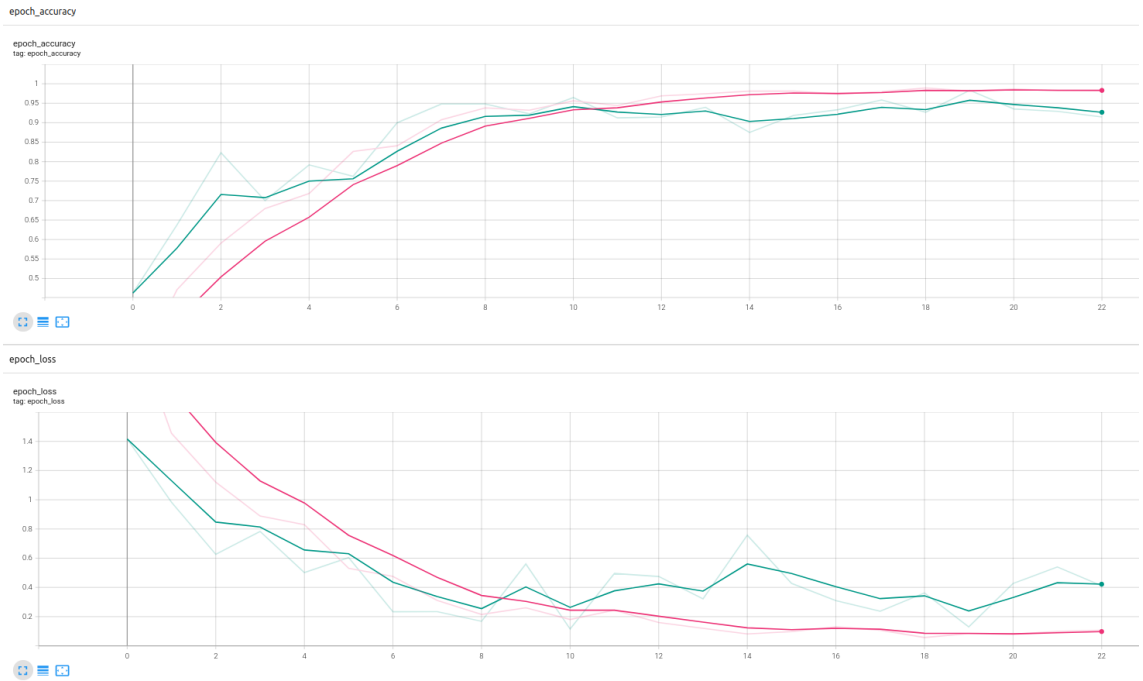

Figura 39 Curvas CNN+LSTMAE, Aceleración + Giroscopio, batch 8, epochs 22.

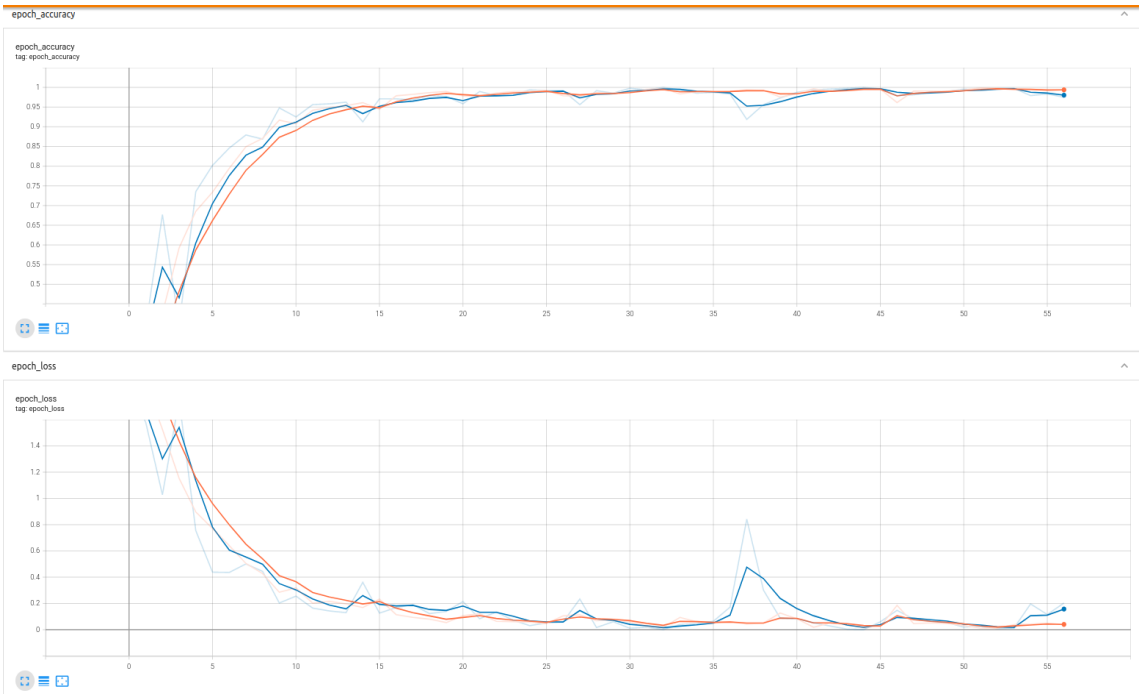

Figura 40 Curvas CNN+LSTMAE, Aceleración, batch 10, epochs 44.

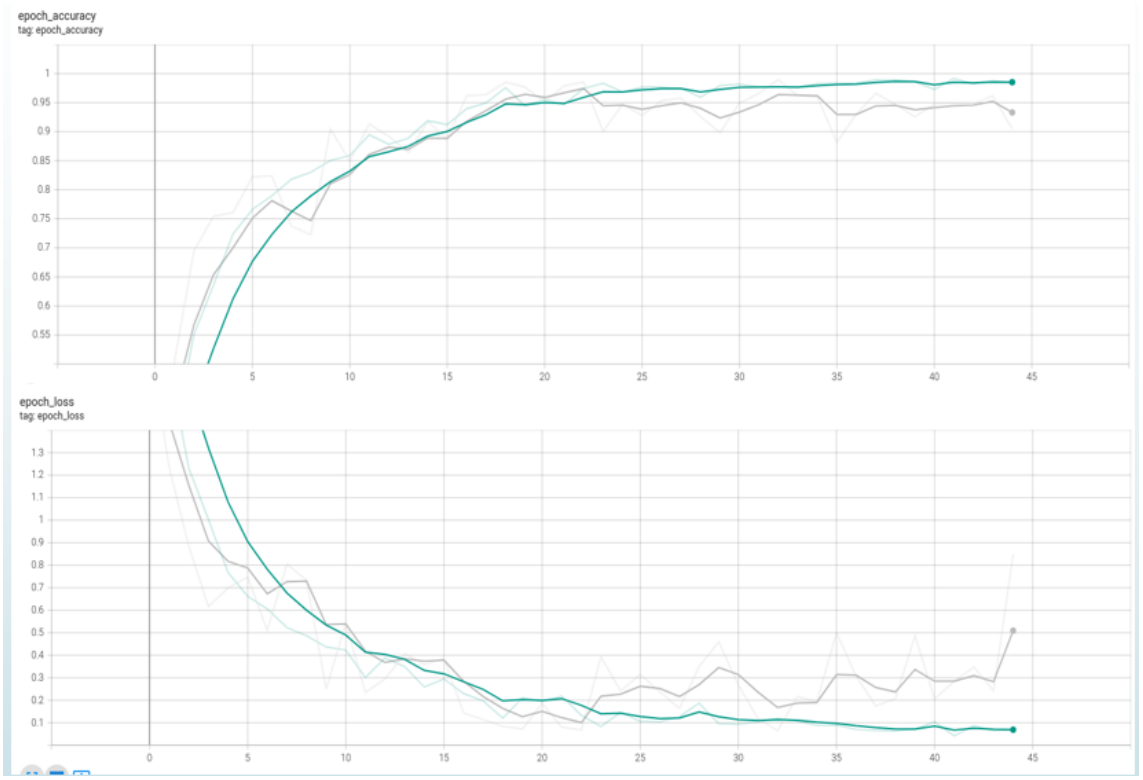

Figura 41 Curvas CNN+LSTMAE, Aceleración + Cuaternios, batch 8, epochs 22.

Una vez seleccionados los valores más adecuados en cuanto tamaño de *batch* y número de *epochs*, se realizaron experimentos del tipo 10k-fold (con el mismo sujeto para test y validación), pero en este caso sin *Early stopping*, ni ningún tipo de condicionamiento de los resultados con la validación (solo nos fijamos en el *accuracy* y loss de train) y con los steps ajustados al *batch* (número de muestras/batch). A continuación, tenemos los resultados obtenidos, resumidos en la Tabla 6 y la Tabla 7.

Tabla 6 Resultados óptimos LSTMAE

LSTMAE						
Datos	Epoch	batch	ACC	ACC_original	F1_SCORE	F1_original
Aceleración	34	20	99%	99%	0,95	0,93
Cuaternios	15	22	96%	95%	0,75	0,72
Giroscopio	8	16	99%	99%	0,93	0,93
Aceleración + Cuaternios	30	8	99%	99%	0,94	0,94
Aceleración + giroscopio	18	14	99%	99%	0,96	0,96

Tabla 7 Resultados óptimos CNN+LSTMAE.

CNN+LSTMAE						
Datos	Epoch	batch	ACC	ACC_original	F1_SCORE	F1_original
Aceleración	44	10	99%	99%	0,97	0,96
Cuaternios	29	18	99%	98%	0,93	0,89
Giroscopio	22	10	99%	99%	0,93	0,95

Aceleración + Cuaternios	22	8		99%		0,93
Aceleración + giroscopio	22	8	99%	99%	0,94	0,94

Con los resultados obtenidos, aunque se encuentren incompletos (CNN+LSTMAE) por problemas con el pc, podemos ver que en general la búsqueda del óptimo ha permitido mejorar o mantener en la mayor parte de los casos el F1 obtenido. Como excepciones tenemos el caso del giroscopio con CNN + LSTMAE (con resultados peores), probablemente debidos a no poder llegarse a un tamaño de *batch* óptimo (probablemente superior) por problemas con el pc.

Con esto podemos confirmar que, si bien en todos los casos no ha sido posible encontrar estos óptimos, sí que podemos confirmar que estos existen, con lo cual podemos afirmar que la hipótesis de la existencia de óptimos era correcta.

3.3.4. Selección del LSTMAE

Como una ampliación directa de la investigación llevada a cabo en la anterior pregunta de investigación, nos encontramos con la última pregunta de investigación a responder en el presente trabajo: **¿Hay otras configuraciones de LSTMAE que proporcionen mejores resultados?**

Siendo la hipótesis: **Existen otras configuraciones de LSTMAE que den mejores resultados.**

Para esta comprobación final, se pretenden probar dos estructuras LSTMAE alternativas:

- LSTMAE2: como LSTMAE, pero sin la etapa de *batch normalization.*
- LSTMAE3: similar a LSTMAE2 y con una capa más de 64 units en el encoder y el decoder

Para las comprobaciones se escogieron del entrenamiento 10k-fold realizado en las pruebas del óptimo el mejor y el peor resultado de k-fold para los datos de aceleración y aceleración + cuaternios (los más destacables para el presente trabajo).

Realizándose el entrenamiento de forma independiente (es decir sin 10k-fold), con las nuevas estructuras (LSTMAE2 y LSTMAE3), y con los parámetros del entrenamiento de óptimos (tamaños de *batch* y *epochs*, y mismos sujetos de *train* y *test*), así como empleando todos los datos (número de muestras/*batch*).

Así pues, los mejores resultados para aceleración son con el sujeto S09 como test, y el peor con S02, empleando *batch* 20 y 34 *epochs* sin *early stopping*.

Por otra parte, para aceleración + cuaternios, el mejor sujeto es el S11 y el peor S10, con *batch* 8 y 30 *epochs*. Con el mismo sujeto de *test* y *validación* en ambos casos (como en los entrenamientos previos).

Los resultados obtenidos con dicho entrenamiento, los tenemos presentes en la Tabla 8

Tabla 8 Resultados comparación LSTMAE

Datos	Red	ACC/F1 (peor LSTMAE)	ACC/F1 (peor obtenido)	ACC/F1 (mejor LSTMAE)	ACC/F1 (mejor obtenido)
Aceleración	LSTMAE2	(0,99;0,9)	(1;0,97)	(1,0,99)	(1;0,97)
	LSTMAE3		(0,99;0,91)		(0,99;0,95)
Aceleración + Cuaternios	LSTMAE2	(0,97,0,8)	(1;0,97)	(1,0,99)	(0,99;0,92)
	LSTMAE3		(1;0,97)		(1;0,97)

En vista de los resultados, se puede apreciar, que, para el peor caso, ambas configuraciones han obtenido un F1 superior al de la LSTMAE original, pero para el mejor caso, ninguno de los dos ha conseguido igualar los resultados de la LSTMAE original.

A mayores de lo cual, como comparación adicional, se realizó el entrenamiento con la misma configuración de 10k-fold que en la búsqueda de los datos (segunda pregunta de investigación), concretamente para la comparación con los datos de aceleración (*batch* 15, *early stopping* 12, etc.). Los resultados se pueden observar en la Tabla 9.

Tabla 9 Comparación aceleración LSTMAE.

Red	ACC	F1
LSTMAE	99%	0,93
LSTMAE2	99%	0,93
LSTMAE3	99%	0,93

También de este último entrenamiento, se adjuntan las tablas de métricas de cada uno de los LSTMAE.

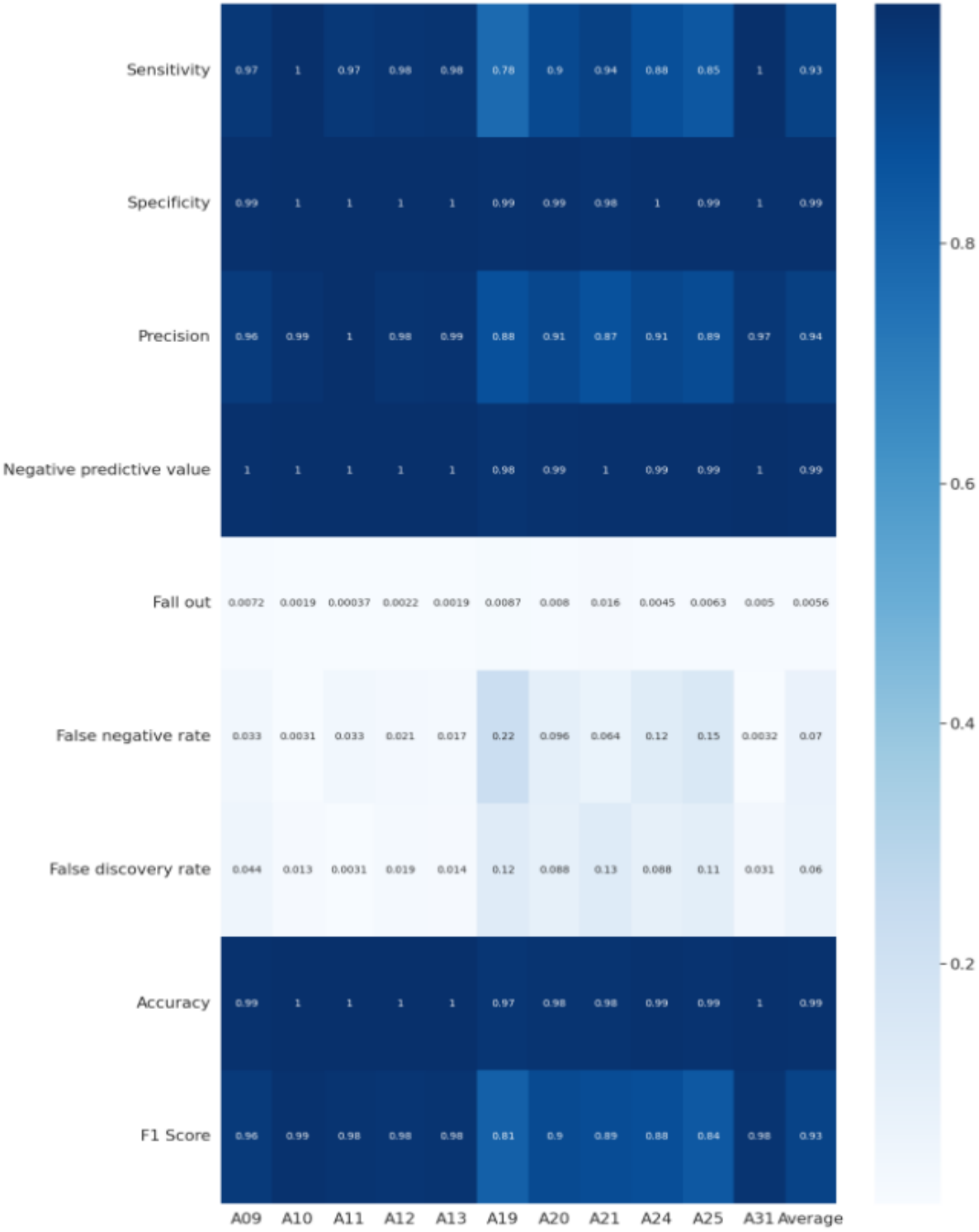

Figura 42 Resultados con LSMTAE, Aceleración, batch 15.

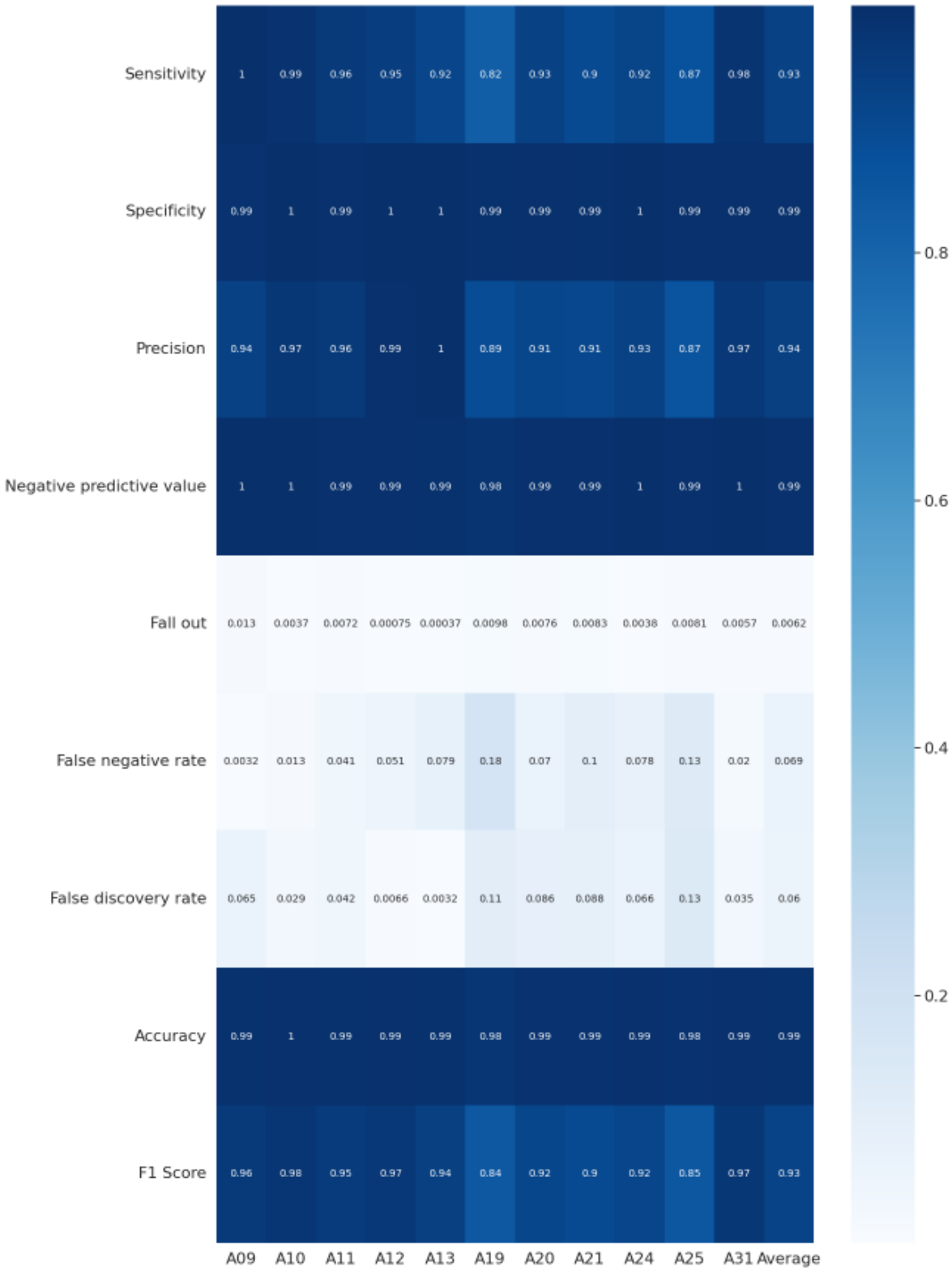

Figura 43 Resultados con LSMTAE2, Aceleración, batch 15.

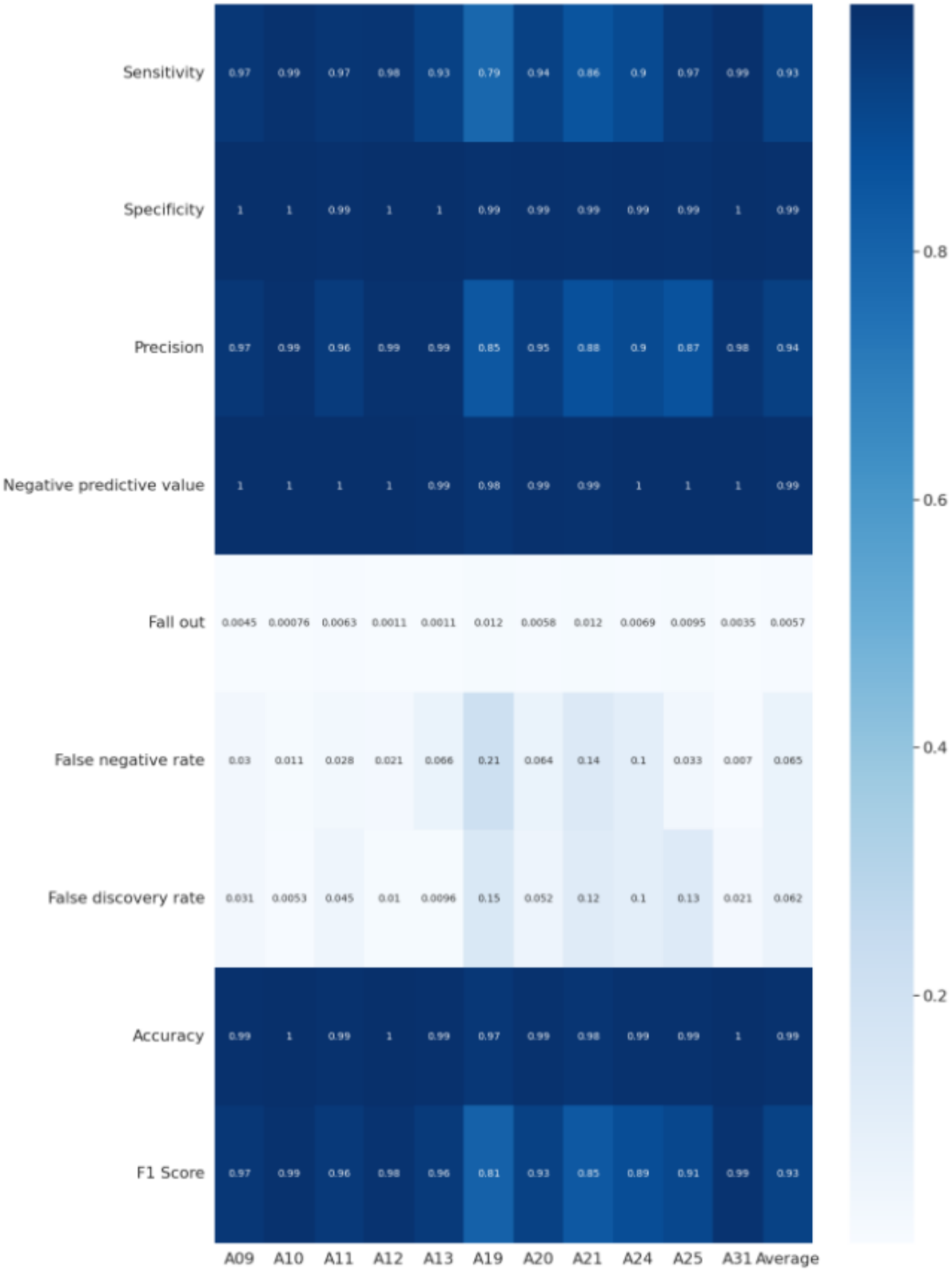

Figura 44 Resultados con LSMTAE3, Aceleración, batch 15.

Con los resultados obtenidos, podemos decir que, aunque probando con sujetos particulares de test (el peor y el mejor encontrados en 10-k fold), eliminando *batch normalization* (LSTMAE2), parece que se obtienen mejores resultados, al comprobarlo con el mismo 10-k fold de LSTMAE de la segunda pregunta de investigación (búsqueda de datos y batch 15), los resultados de las tres redes son los mismos (con ligeras variaciones en las tablas de métricas).

Es decir, es posible que para casos concretos LSTMAE2 aporte mejores resultados, pero con las pruebas planteadas no se aprecia dicha mejora de forma general, y por tanto podría ser interesante de cara al futuro, realizar un mayor número de pruebas en este sentido.

Además, viendo el resultado de LSTMAE3, también podemos aventurarnos a comentar que el añadir una capa más al auto-encoder LSTM, no proporciona ninguna mejora apreciable en los resultados.

4.Conclusiones y líneas futuras

4.1. Conclusiones

Durante el presente trabajo, tomando como base los trabajos realizados por Pardo-Villalibre y Sáez Bombín, se ha tratado de dar respuesta a la pregunta de investigación: **¿Es posible encontrar en la literatura actual una arquitectura de red que permita obtener unos resultados iguales o superiores a los obtenidos con la red implementada por Sáez Bombín en su TFM?**

Para responder esta pregunta, en primer lugar, se realizó un estudio del estado del arte, en el que se pudiera por una parte comparar los diversos trabajos centrados en la resolución del problema de *HAR*, y por otra analizar los estudios centrados en la base de datos empleada en el presente trabajo REALDISP.

De este análisis del estadio del arte previo, se obtuvo una idea inicial del estado del arte de la técnica en la última década, y se seleccionaron de las encontradas en la literatura una serie de arquitecturas de Deep learning (o variantes de las mismas), viables para una comprobación inicial de la pregunta de investigación.

Mediante el empleo de las arquitecturas de control seleccionadas, el *framework* de Pardo-Villalibre y las especificaciones seleccionadas por Sáez Bombín, se realizaron una serie de entrenamientos y comprobaciones con estas redes, de cara a la comprobación de la hipótesis de la pregunta de investigación, y de las preguntas de investigación derivadas de esta inicial, y surgidas durante el proceso de investigación.

Tras la realización de las pruebas realizadas, con el análisis de las métricas *Accuracy* y F1, obtenidas durante los entrenamientos, se pudieron confirmar, o al menos intuir la veracidad de las hipótesis planteadas durante el trabajo.

Como resumen, y en vista de los resultados obtenidos, podemos concluir para las preguntas de investigación:

- **¿Es posible encontrar en la literatura actual una arquitectura de red que permita obtener unos resultados iguales o superiores a los obtenidos con la red implementada por Sáez Bombín en su TFM?**

  Sí que es posible encontrar redes en la literatura actual, como son el caso de las redes propuestas LSTMAE y CNN+LSMTAE.

- **¿Son los cuaternios los mejores datos para trabajar con la base de datos REALDISP?**

Los cuaternios no son los mejores datos para trabajar con REALDISP, ya que los que mejores resultados dan son las aceleraciones.

- **¿Podemos encontrar otros tamaños de batch y número de epochs de entrenamiento para mejorar los resultados de las redes que obtenían los mejores resultados en los experimentos anteriores?**

  Sí que es posible encontrar parámetros de *batch* y *epochs* que mejoran los resultados anteriores, aunque hay que tener en cuenta que posiblemente con mayores tamaños de batch, o variando otros parámetros, estos valores podrían mejorar.

- **¿Hay otras configuraciones de LSTMAE que proporcionen mejores resultados?**

  Aunque hay que realizar un mayor número de comprobaciones en este campo, sí que es posible intuir que el empleo de otras configuraciones del LSTMAE (y del CNN+LSTMAE), pueden aportar resultados superiores, a los obtenidos con la versión por defecto.

En resumen, tras el trabajo realizado, se tiene un mayor conocimiento de cómo poder trabajar con REALDISP, del tipo de arquitecturas a probar, y los datos que aportarán mejores resultados en el problema de *HAR*. Pudiendo ser los resultados obtenidos sustanciales con un análisis en mayor medida de los mismos, para la posible publicación de un artículo. Pudiendo emplearse este trabajo como un punto de partida interesante para el trabajo futuro con REALDISP, en base a los resultados obtenidos.

4.2. Líneas futuras

Como posibles líneas futuras de investigación, se podrían llevar a cabo las siguientes comprobaciones:

- Comparación con el artículo de SMART [35], empleando para ello los mismos sujetos y actividades que este en el entrenamiento con 10k-fold.

- Comprobación con todo el dataset REALDISP, empleando todos los sujetos y actividades de la base de datos, para las redes propuestas (LSMTAE y CNN+LSTMAE), comparándolo así con otros trabajos que empleen REALDISP, así como con los resultados de emplear únicamente unos sujetos y datos concretos, como es el caso del presente trabajo.

- Continuación de las comprobaciones de posibles modificaciones de las redes propuestas: más o menos capas CNN, poner la CNN después del autoencoder, CNN+LSTMAE sin *batch normalization*, etc.

- Proponer redes alternativas en las que la utilización combinada de cuaternios junto con aceleraciones pudieran proporcionar mejores resultados.

De esta forma, llevando a cabo las comprobaciones anteriores, y buscando ampliar los resultados aquí obtenidos, se podrían incluso trasladar dichos resultados a una posible publicación científica.

5.Bibliografía

[1] S. Sáez Bombín, «Sistema de Aprendizaje Profundo para reconocimiento de actividades con sensores de captura de movimientos», 2020, Accedido: 5 de noviembre de 2022. [En línea]. Disponible en: https://uvadoc.uva.es/handle/10324/41353

[2] G. Pardo Villalibre, «Redes de aprendizaje profundo para reconocimiento de actividades humanas: framework de pre-procesado y entrenamiento en Tensorflow», 2021, Accedido: 13 de septiembre de 2022. [En línea]. Disponible en: https://uvadoc.uva.es/handle/10324/50023

[3] O. Banos, M. A. Toth, M. Damas, H. Pomares, y I. Rojas, «Dealing with the Effects of Sensor Displacement in Wearable Activity Recognition», *Sensors (Basel)*, vol. 14, n.º 6, pp. 9995-10023, jun. 2014, doi: 10.3390/s140609995.

[4] O. Baños, M. Damas, H. Pomares, I. Rojas, M. A. Tóth, y O. Amft, «A benchmark dataset to evaluate sensor displacement in activity recognition», en *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, New York, NY, USA, sep. 2012, pp. 1026-1035. doi: 10.1145/2370216.2370437.

[5] S. Bian, M. Liu, B. Zhou, y P. Lukowicz, «The State-of-the-Art Sensing Techniques in Human Activity Recognition: A Survey», *Sensors*, vol. 22, n.º 12, Art. n.º 12, ene. 2022, doi: 10.3390/s22124596.

[6] F. Serpush, M. B. Menhaj, B. Masoumi, y B. Karasfi, «Wearable Sensor-Based Human Activity Recognition in the Smart Healthcare System», *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1-31, feb. 2022, doi: 10.1155/2022/1391906.

[7] F. Demrozi, G. Pravadelli, A. Bihorac, y P. Rashidi, «Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey», *IEEE Access*, vol. 8, pp. 210816-210836, 2020, doi: 10.1109/ACCESS.2020.3037715.

[8] F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, y X. Liu, «A Survey on Deep Learning for Human Activity Recognition», *ACM Computing Surveys*, vol. 54, ago. 2021, doi: 10.1145/3472290.

[9] M. Vrigkas, C. Nikou, y I. A. Kakadiaris, «A Review of Human Activity Recognition Methods», *Front. Robot. AI*, vol. 2, nov. 2015, doi: 10.3389/frobt.2015.00028.

[10] N. Y. Hammerla, S. Halloran, y T. Ploetz, «Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables». arXiv, 29 de abril de 2016. Accedido: 31 de agosto de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1604.08880

[11] E. Ramanujam, T. Perumal, y S. Padmavathi, «Human Activity Recognition With Smartphone and Wearable Sensors Using Deep Learning Techniques: A Review», *IEEE Sensors J.*, vol. 21, n.º 12, pp. 13029-13040, jun. 2021, doi: 10.1109/JSEN.2021.3069927.

[12] C. Hofmann, C. Patschkowski, B. Haefner, y G. Lanza, «Machine Learning Based Activity Recognition To Identify Wasteful Activities In Production», *Procedia Manufacturing*, vol. 45, pp. 171-176, 2020, doi: 10.1016/j.promfg.2020.04.090.

[13] J. R. Kwapisz, G. M. Weiss, y S. A. Moore, «Activity recognition using cell phone accelerometers», *SIGKDD Explor. Newsl.*, vol. 12, n.º 2, pp. 74-82, mar. 2011, doi: 10.1145/1964897.1964918.

[14] H. Y. Yatbaz, E. Ever, y A. Yazici, «Activity Recognition and Anomaly Detection in E-Health Applications Using Color-Coded Representation and Lightweight CNN Architectures», *IEEE Sensors J.*, vol. 21, n.º 13, pp. 14191-14202, jul. 2021, doi: 10.1109/JSEN.2021.3061458.

[15] O. Banos *et al.*, «mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications», en *Ambient Assisted Living and Daily Activities*, Cham, 2014, pp. 91-98. doi: 10.1007/978-3-319-13105-4_14.

[16] O. Banos *et al.*, «Design, implementation and validation of a novel open framework for agile development of mobile health applications», *Biomed Eng Online*, vol. 14 Suppl 2, p. S6, 2015, doi: 10.1186/1475-925X-14-S2-S6.

[17] D. Gholamiangonabadi, N. Kiselov, y K. Grolinger, «Deep Neural Networks for Human Activity Recognition With Wearable Sensors: Leave-One-Subject-Out Cross-Validation for Model Selection», *IEEE Access*, vol. 8, pp. 133982-133994, 2020, doi: 10.1109/ACCESS.2020.3010715.

[18] M. Bock, A. Hoelzemann, M. Moeller, y K. Van Laerhoven, «Improving Deep Learning for HAR with shallow LSTMs». arXiv, 5 de agosto de 2021. doi: 10.48550/arXiv.2108.00702.

[19] D. Roggen *et al.*, «Collecting complex activity datasets in highly rich networked sensor environments», en *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, jun. 2010, pp. 233-240. doi: 10.1109/INSS.2010.5573462.

[20] F. J. Ordóñez y D. Roggen, «Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition», *Sensors*, vol. 16, n.º 1, Art. n.º 1, ene. 2016, doi: 10.3390/s16010115.

[21] A. Stisen *et al.*, «Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition», en *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, New York, NY, USA, nov. 2015, pp. 127-140. doi: 10.1145/2809695.2809718.

[22] P. M. Scholl, M. Wille, y K. Van Laerhoven, «Wearables in the wet lab: a laboratory system for capturing and guiding experiments», en *Proceedings of the 2015*

ACM International Joint Conference on Pervasive and Ubiquitous Computing, New York, NY, USA, sep. 2015, pp. 589-599. doi: 10.1145/2750858.2807547.

[23] T. Sztyler y H. Stuckenschmidt, «On-body localization of wearable devices: An investigation of position-aware activity recognition», en *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, mar. 2016, pp. 1-9. doi: 10.1109/PERCOM.2016.7456521.

[24] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, y D. Anguita, «Transition-Aware Human Activity Recognition Using Smartphones», *Neurocomputing*, vol. 171, pp. 754-767, ene. 2016, doi: 10.1016/j.neucom.2015.07.085.

[25] P. F. Moshiri, H. Navidan, R. Shahbazian, S. A. Ghorashi, y D. Windridge, «Using GAN to Enhance the Accuracy of Indoor Human Activity Recognition», p. 5.

[26] S. Yousefi, H. Narui, S. Dayal, S. Ermon, y S. Valaee, «A Survey on Behavior Recognition Using WiFi Channel State Information», *IEEE Communications Magazine*, vol. 55, pp. 98-104, oct. 2017, doi: 10.1109/MCOM.2017.1700082.

[27] X. Li, Y. Wang, B. Zhang, y J. Ma, «PSDRNN: An Efficient and Effective HAR Scheme Based on Feature Extraction and Deep Learning», *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, vol. 16, n.º 10, p. 11, 2020.

[28] K. Xia, J. Huang, y H. Wang, «LSTM-CNN Architecture for Human Activity Recognition», *IEEE Access*, vol. 8, pp. 56855-56866, 2020, doi: 10.1109/ACCESS.2020.2982225.

[29] M. Abdu-Aguye y W. Gomaa, «VersaTL: Versatile Transfer Learning for IMU-based Activity Recognition using Convolutional Neural Networks», ene. 2019, pp. 507-516. doi: 10.5220/0007916705070516.

[30] W. Gomaa, R. Elbasiony, y S. Ashry, «ADL Classification Based on Autocorrelation Function of Inertial Signals», en *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, dic. 2017, pp. 833-837. doi: 10.1109/ICMLA.2017.00-53.

[31] K. Altun, B. Barshan, y O. Tunçel, «Comparative study on classifying human activities with miniature inertial and magnetic sensors», *Pattern Recognition*, vol. 43, n.º 10, pp. 3605-3620, oct. 2010, doi: 10.1016/j.patcog.2010.04.019.

[32] S. Mohammed, R. Elbasiony, y W. Gomaa, «An LSTM-based Descriptor for Human Activities Recognition using IMU Sensors», ene. 2018, pp. 504-511. doi: 10.5220/0006902405040511.

[33] L. T. Nguyen, M. Zeng, P. Tague, y J. Zhang, «Recognizing new activities with limited training data», en *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, New York, NY, USA, sep. 2015, pp. 67-74. doi: 10.1145/2802083.2808388.

[34] M. G. Abdu-Aguye, W. Gomaa, Y. Makihara, y Y. Yagi, «Adaptive Pooling Is All You Need: An Empirical Study on Hyperparameter-insensitive Human Action Recognition Using Wearable Sensors», en *2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, United Kingdom, jul. 2020, pp. 1-6. doi: 10.1109/IJCNN48605.2020.9207082.

[35] M. Ghorpade, H. Chen, Y. Liu, y Z. Jiang, «SMART: Emerging Activity Recognition with Limited Data for Multi-modal Wearable Sensing», en *2020 IEEE International Conference on Big Data (Big Data)*, Atlanta, GA, USA, dic. 2020, pp. 1316-1321. doi: 10.1109/BigData50022.2020.9377746.

[36] R. Chen, H. Luo, F. Zhao, X. Meng, Z. Xie, y Y. Zhu, «Modeling Accurate Human Activity Recognition for Embedded Devices Using Multi-level Distillation». arXiv, 11 de agosto de 2021. doi: 10.48550/arXiv.2107.07331.

[37] M.-C. Yu, T. Yu, S.-C. Wang, C.-J. Lin, y E. Y. Chang, «Big data small footprint: the design of a low-power classifier for detecting transportation modes», *Proc. VLDB Endow.*, vol. 7, n.º 13, pp. 1429-1440, ago. 2014, doi: 10.14778/2733004.2733015.

[38] B. Barshan y M. C. Yüksek, «Recognizing Daily and Sports Activities in Two Open Source Machine Learning Environments Using Body-Worn Sensor Units», *The Computer Journal*, vol. 57, n.º 11, pp. 1649-1667, nov. 2014, doi: 10.1093/comjnl/bxt075.

[39] K. Altun y B. Barshan, «Human Activity Recognition Using Inertial/Magnetic Sensor Units», en *Human Behavior Understanding*, Berlin, Heidelberg, 2010, pp. 38-51. doi: 10.1007/978-3-642-14715-9_5.

[40] J. Zhu, R. San-Segundo, y J. M. Pardo, «Feature extraction for robust physical activity recognition», *Hum. Cent. Comput. Inf. Sci.*, vol. 7, n.º 1, p. 16, dic. 2017, doi: 10.1186/s13673-017-0097-2.

[41] Md. T. Uddin y Md. A. Uddiny, «Human activity recognition from wearable sensors using extremely randomized trees», en *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, Savar, Dhaka, Bangladesh, may 2015, pp. 1-6. doi: 10.1109/ICEEICT.2015.7307384.

[42] A. Reiss y D. Stricker, «Introducing a New Benchmarked Dataset for Activity Monitoring», en *2012 16th International Symposium on Wearable Computers*, jun. 2012, pp. 108-109. doi: 10.1109/ISWC.2012.13.

[43] P. Casale, O. Pujol, y P. Radeva, «Human Activity Recognition from Accelerometer Data Using a Wearable Device», en *Pattern Recognition and Image Analysis*, Berlin, Heidelberg, 2011, pp. 289-296. doi: 10.1007/978-3-642-21257-4_36.

[44] A. Subasi *et al.*, «Sensor Based Human Activity Recognition Using Adaboost Ensemble Classifier», *Procedia Computer Science*, vol. 140, pp. 104-111, 2018, doi: 10.1016/j.procs.2018.10.298.

[45] A. Mimouna, A. B. Khalifa, y N. E. Ben Amara, «Human Action Recognition Using Triaxial Accelerometer Data: Selective Approach», en *2018 15th International Multi-Conference on Systems, Signals & Devices (SSD)*, Hammamet, mar. 2018, pp. 491-496. doi: 10.1109/SSD.2018.8570429.

[46] M. G. Abdu-Aguye y W. Gomaa, «Competitive Feature Extraction for Activity Recognition based on Wavelet Transforms and Adaptive Pooling», en *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, jul. 2019, pp. 1-8. doi: 10.1109/IJCNN.2019.8852299.

6.Anexo A

6.1. Summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 60, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 30, 32)	0
dropout (Dropout)	(None, 64, 30, 32)	0
conv2d_1 (Conv2D)	(None, 64, 30, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 32, 15, 32)	0
dropout_1 (Dropout)	(None, 32, 15, 32)	0
conv2d_2 (Conv2D)	(None, 32, 15, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 16, 8, 32)	0
dropout_2 (Dropout)	(None, 16, 8, 32)	0
conv2d_3 (Conv2D)	(None, 16, 8, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 8, 4, 32)	0
dropout_3 (Dropout)	(None, 8, 4, 32)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 512)	524800
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 11)	2827

Total params: 687,019
Trainable params: 687,019
Non-trainable params: 0

Figura 45 Summary de CNN+COLOR.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 60, 64)	640
max_pooling2d (MaxPooling2D)	(None, 64, 30, 64)	0
conv2d_1 (Conv2D)	(None, 64, 30, 64)	65600
max_pooling2d_1 (MaxPooling2	(None, 32, 15, 64)	0
flatten (Flatten)	(None, 30720)	0
dense (Dense)	(None, 64)	1966144
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 11)	363

Total params: 2,034,827
Trainable params: 2,034,827
Non-trainable params: 0

Figura 46 Summary CNN2C.

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 128, 60)	0
lstm (LSTM)	(None, 128, 32)	11904
dropout (Dropout)	(None, 128, 32)	0
lstm (LSTM)	(None, 128, 32)	8320
dropout (Dropout)	(None, 128, 32)	0
lstm (LSTM)	(None, 128, 32)	8320
dropout (Dropout)	(None, 128, 32)	0
lstm (LSTM)	(None, 32)	8320
dropout (Dropout)	(None, 32)	0
flatten (Flatten)	(None, 32)	0
dense (Dense)	(None, 11)	363

Total params: 37,227
Trainable params: 37,227
Non-trainable params: 0

Figura 47 Summary LSTM.

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 128, 60)	0
lstm (LSTM)	(None, 128, 32)	11904
lstm (LSTM)	(None, 128, 32)	8320
reshape (Reshape)	(None, 1, 128, 32)	0
conv2d (Conv2D)	(None, 1, 64, 64)	51264
max_pooling2d (MaxPooling2	(None, 1, 32, 64)	0
conv2d (Conv2D)	(None, 1, 32, 128)	73856
global_average_pooling2d (	(None, 128)	0
batch_normalization (Batch	(None, 128)	512
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 11)	1419

Total params: 147,275
Trainable params: 147,019
Non-trainable params: 256

Figura 48 Summary LSTM+CNN.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 60, 64)	640
max_pooling2d (MaxPooling2D)	(None, 64, 30, 64)	0
batch_normalization (BatchNo	(None, 64, 30, 64)	256
conv2d_1 (Conv2D)	(None, 64, 30, 64)	65600
max_pooling2d_1 (MaxPooling2	(None, 32, 15, 64)	0
batch_normalization_1 (Batch	(None, 32, 15, 64)	128
conv2d_2 (Conv2D)	(None, 32, 15, 64)	65600
max_pooling2d_2 (MaxPooling2	(None, 16, 8, 64)	0
batch_normalization_2 (Batch	(None, 16, 8, 64)	64
dropout (Dropout)	(None, 16, 8, 64)	0
reshape (Reshape)	(None, 16, 512)	0
lstm (LSTM)	(None, 64)	147712
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 11)	715

Total params: 280,715
Trainable params: 280,491
Non-trainable params: 224

Figura 49 Summary CNN+LSTM-Saez.

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 128, 60)	0
lstm (LSTM)	(None, 128, 256)	324608
lstm (LSTM)	(None, 128)	197120
batch_normalization (Batch	(None, 128)	512
flatten (Flatten)	(None, 128)	0
dropout (Dropout)	(None, 128)	0
repeat_vector (RepeatVecto	(None, 11, 128)	0
lstm (LSTM)	(None, 11, 128)	131584
lstm (LSTM)	(None, 11, 256)	394240
time_distributed (TimeDist	(None, 11, 256)	65792
time_distributed (TimeDist	(None, 11, 3)	771
batch_normalization (Batch	(None, 11, 3)	12
dropout (Dropout)	(None, 11, 3)	0
flatten (Flatten)	(None, 33)	0
dense (Dense)	(None, 11)	374

Total params: 1,115,013
Trainable params: 1,114,751
Non-trainable params: 262

Figura 50 Summary LSTMAE.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 60, 64)       640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 30, 64)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 30, 64)        256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 30, 64)        65600
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 15, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 15, 64)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 15, 64)        65600
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 8, 64)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 8, 64)         64
_________________________________________________________________
dropout (Dropout)            (None, 16, 8, 64)         0
_________________________________________________________________
reshape (Reshape)            (None, 16, 512)           0
_________________________________________________________________
lstm (LSTM)                  (None, 16, 256)           787456
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               197120
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
repeat_vector (RepeatVector) (None, 11, 128)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 11, 128)           131584
_________________________________________________________________
lstm_3 (LSTM)                (None, 11, 256)           394240
_________________________________________________________________
time_distributed (TimeDistri (None, 11, 256)           65792
_________________________________________________________________
time_distributed_1 (TimeDist (None, 11, 3)             771
_________________________________________________________________
batch_normalization_4 (Batch (None, 11, 3)             12
_________________________________________________________________
dropout_2 (Dropout)          (None, 11, 3)             0
_________________________________________________________________
flatten_1 (Flatten)          (None, 33)                0
_________________________________________________________________
dense_2 (Dense)              (None, 11)                374
=================================================================
Total params: 1,710,149
Trainable params: 1,709,663
Non-trainable params: 486
```

Figura 51 Summary CNN+LSTMAE.